SOFTIC 2022-1
ソフトウェア等の権利保護に関する調査研究

ソフトウェア等の権利保護に関する調査研究

報告書
（資料編）

— 2022（令和 4）年度 —

2023（令和 5）年 3 月

一般財団法人ソフトウェア情報センター

本事業は、一般社団法人授業目的公衆送信補償金等管理協会（SARTRAS）の
共通目的基金の助成を受け実施されています。

目　次

Google v. Oracle事件最高裁判決

2022年12月1日

慶應義塾大学大学院法務研究科教授
奥邨　弘司

© Koji OKUMURA

▸訴訟の経緯

1996年　SunがJavaを開発　（2010年OracleがSunを買収）

2005年　GoogleがAndroidを開発開始
　　Javaのライセンスを受けようとするが頓挫。JavaAPIの一部を利用してAndroidプラットフォームを開発

2010年　Oracleが著作権侵害及び特許権侵害でGoogleを提訴

2012年　陪審は特許権侵害否定、APIの著作物性と著作権侵害を認定
　　地裁は、陪審評決を覆し、APIの著作物性を否定
　　〔872 F. Supp. 2d 974〕

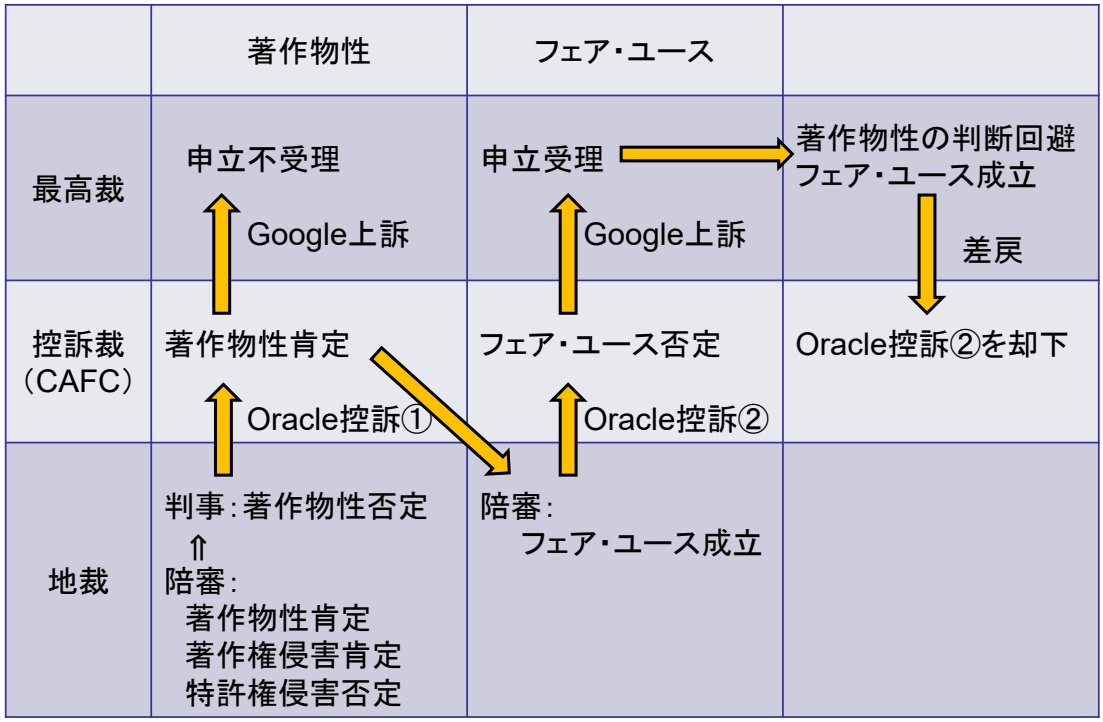
2014年　CAFCがAPIの著作物性を認めて、地裁判決を取り消し、差戻し
　　〔750 F.3d 1339〕　⇒　Googleの裁量上訴の申立却下

2016年　地裁はフェアユースを認める

2018年　CAFCがフェアユースを否定して、事件を差戻し
　　〔886 F.3d 1179〕 ⇒　Googleの裁量上訴の申立認容

2021年　Googleにフェア・ユース成立の最高裁判決
　　⇒　事件はCAFCに差戻し

▸訴訟経緯



■Google LLC v. Oracle Am., Inc., 141 S. Ct. 1183 (2021)　の概要

≪法廷意見≫　6人
API宣言コードの著作物性の判断は回避した上で、フェア・ユース成立と判断

ブライヤー判事・ソトマイヨール判事・ケイガン判事　・・・　リベラル派
ロバーツ長官　・・・　中間派
ゴーサッチ判事・カバノー判事　・・・　保守派

≪反対意見≫　2人
APIの宣言コードには著作物性があり、またその利用はフェア・ユースではない

トーマス判事・アリート判事　・・・　保守派

＊新任のバレット判事（保守派）は、就任前の事件であり、参加せず

■ フェア・ユース （米国著作権法107条）

106条および106A条の規定にかかわらず、批評、論評、ニュース報道、教育（教室での使用のための複数複製を含む）、研究、調査などの目的で、著作権のある著作物を公正に利用すること――複製物またはレコードの形での複製による利用、または該当条に特掲された他の方法による利用を含む――は、著作権の侵害とはならない。個々の事件における著作物の利用が公正な利用といえるか否かを決定する上で、考慮されるべき要素には以下のものが含まれる。

① 当該利用の目的および性格。なお、当該利用が商業的性質のものか、非営利的教育目的かといったことも含む
② 当該著作権のある著作物の性質
③ 当該著作権のある著作物全体との関係で利用される部分の量および実質性ならびに
④ 当該利用が、当該著作権のある著作物の潜在的な市場や価値に与える影響

上記要素のすべてを考慮した上で、公正な利用であるとされた場合、著作物が未発行であるという事実は、それ自体では、公正な利用であるとすることを妨げない。

【5】

■ フェア・ユース

フェア・ユースは、制定法ではなく判例法によって生み出された概念である

⇓

Folsom v. Marsh, 9. F.Cas. 342 (C.C.D. Mass. 1841)

・被告は、原告の著作物（初代大統領ワシントンの著作集）から、無断で転載・要約を行って被告書籍を完成させたため、著作権侵害に問われた

ストリー判事による判決

「要するに、本件のような問題を判断する際には、行われた行為の性質と目的、利用された既存著作物の量と価値、そして、その利用によって、原作品の販売を害したり、原作品の利益を減少させたり、原作品の目的に取って代わったりする程度に注目しなければならない。」

行われた行為の性質と目的	⇒	第1要素
利用された著作物の価値	⇒	第2要素
利用された著作物の量	⇒	第3要素
市場代替性・市場への害	⇒	第4要素

Folsom判決に始まる一連の判例法を、現行著作権法制定時に明文化したのが107条

＊ 厳密な意味では、Folsom事件判決はフェア・ユース（権利制限）の起源ではない。実は当時の著作権法には翻案権が存在しなかった。結果、被告の行為は著作権侵害ではなかった。ストーリー判事は、複製だけではなくて翻案にも権利が及ぶとしつつ、その範囲を画するために前記説示を行った。つまり権利拡張の説示だった。ただ、権利の範囲は裏返せば権利制限の範囲であり、実質的にフェアユースの起源と考えられるようになった。

【6】

■ 現行法制定後のフェア・ユースに関する最高裁判決のポイント

Sony v. Universal, 464 U.S. 417 (1984)

無料TV放送番組のタイムシフト録画をフェア・ユースと認めた

・商業的使用は不公正と推定。非商業的使用は逆の推定　＜第1要素＞
・潜在的市場への影響＝将来の損害の可能性。商業的な場合可能性は推定される
　非商業的な場合、著作権者が可能性を証明する必要がある　＜第4要素＞

Harper & Row v. Nation, 471 U.S. 539 (1985)

刊行前の大統領の手記の一部をスクープした報道についてフェア・ユースを否定

・未公表の著作物の利用がフェア・ユースとなる範囲は狭い　＜第2要素＞
　　⇒　後の法改正で当てはまらなくなる
・量は少なくとも核心部分を利用すれば実質的である　＜第3要素＞
・第4要素は、唯一の最も重要なフェア・ユースの構成要素である　＜第4要素＞
・市場への害の検討に際しては、二次的著作物の市場も考慮すべき　＜第4要素＞

Stewart v. Abend, 495 U.S. 207 (1990)

小説の映画化についてフェア・ユースを否定

・フェア・ユースは、法の厳格適用が創造性を窒息させることを避けるためのもの
・事実的な作品の場合は、創造的な作品の場合よりも、フェア・ユースは認められやすい　＜第2要素＞

【7】

Campbell v. Acuff-Rose, 510 U.S. 569 (1994)

ラップグループがリリースしたパロディ曲の歌詞について、フェア・ユース肯定

＜総論＞

・ フェア・ユース判断に一律の明確な基準はなく、事例毎に、個別に4要素の検討結果を総合考慮して判断すること

＜第1要素＞

・ 検討の中心は、後続の作品が先行作品を代替するだけか、それとも変容力を有しているか、それはどの程度か
・ 商業性の有無は検討事項の1つ。変容力があるほど、その重要性は小さくなる
・ パロディ（原作品を批評・批判するもの）は、変容力を有する

＜第2要素＞

・ パロディ（のような変容力のある作品）の場合、第2要素は重要ではない

＜第3要素＞

・ 許される複製の程度（量的・質的）は、利用の目的と性格によって変化する

＜第4要素＞

・ ソニー判決の推定は、単純複製の場合にのみ当てはまる
・ 後続の利用が変容力を有するほど、市場代替性・市場への影響は不明確になる
・ 辛辣な書評がもたらす悪影響は、著作権法上の評価の対象外
・ 二次的利用のための潜在的な市場には、原作品の権利者自身が一般的に活用する市場と、活用のために他者にライセンスする一般的な市場のみが含まれる

【8】

Campbell v. Acuff-Rose, 510 U.S. 569 (1994)

ラップグループがリリースしたパロディ曲の歌詞について、フェア・ユース肯定

＜総論＞

- フェア・ユース判断に一律の明確な基準はなく、事例毎に、個別に4要素の検討結果を総合考慮して判断すること

＜第1要素＞

- 検討の中心は、後続の作品が先行作品を代替するだけか、それとも変容力を有しているか、それはどの程度か

Sony判決の影響力減

- 商業性の有無は検討事項の1つ。変容力があるほど、その重要性は小さくなる
- パロディ（原作品を批評・批判するもの）は、変容力を有する

＜第2要素＞

Stewart判決の影響力減

- パロディ（のような変容力のある作品）の場合、第2要素は重要ではない

＜第3要素＞

Harper & Row判決の影響力減

- 許される複製の程度（量的・質的）は、利用の目的と性格によって変化する

＜第4要素＞

Sony判決の影響力減

- ソニー判決の推定は、単純複製の場合にのみ当てはまる
- 後続の利用が変容力を有するほど、市場代替性・市場への影響は不明確になる
- 辛辣な書評がもたらす悪影響は、著作権法上の評価の対象外
- 二次的利用のための潜在的な市場には、原作品の権利者自身が一般的に活用する市場と、活用のために他者にライセンスする一般的な市場のみが含まれる

【9】

■ 最高裁判例について

① （著作権に関する）最高裁判例は必ずしも多くない

② 後続する最高裁判例は・・・

・事案に関係しないところは先行最高裁判例に触れない
・関係する部分で方向性が同じ場合は先行最高裁判例を引用する
・一方、衝突する場合は切り分けにより先行最高裁判例の影響力を削ぐ

③ 結果、直近の判例の影響力が加重される

⇓

これまでは、Campbell事件最判の影響力が極めて強かった ： 今後は？

・Campbell事件（以降）においては、第1要素の検討＝変容力の有無が鍵だった
（変容力無双状態！）

・具体的には、変容力がある場合・・・

商業性や第2要素は重要性を失う。同時に、変容力のある利用方法のために必要なら、全部利用（量）や核心部分利用（質）も許される。市場代替性も低いとされて、（潜在的）市場への影響も否定されがち。

第1要素の検討の中心は、既存の著作物を代替するだけか、それとも、既存の著作物を新しい表現や意味、または主張によって変化させることで、さらなる目的や異なる性格を伴う、何か新しいものを付け加えるのかであり、言葉を借りれば、新しい作品が「変容力を有する」のか、それはどの程度かを問うことである。

【10】

■Google LLC v. Oracle Am., Inc., 141 S. Ct. 1183 (2021) の概要

≪法廷意見≫ 6人
API宣言コードの著作物性の判断は回避した上で、フェア・ユース成立と判断

ブライヤー判事・ソトマイヨール判事・ケイガン判事	・・・	リベラル派
ロバーツ長官	・・・	中間派
ゴーサッチ判事・カバノー判事	・・・	保守派

≪反対意見≫ 2人
APIの宣言コードには著作物性があり、またその利用はフェア・ユースではない

トーマス判事・アリート判事	・・・	保守派

＊新任のバレット判事（保守派）は、就任前の事件であり、参加せず

【11】

■Google LLC v. Oracle Am., Inc., 141 S. Ct. 1183 (2021) の概要

≪総論≫
- 〔著作権法の意義、フェア・ユース規定の歴史と意義などについて簡単に述べる〕
- 争点は、①Java API（の宣言コードとその体系を含む）全体の著作物性と、②Googleによる（宣言コードとその体系の）利用がフェア・ユースとなるか。
- ①については、著作物性があることを仮定した上で、②のフェア・ユースの成立の可否について検討する。

≪コンピュータプログラムの特殊性とフェア・ユース≫
- コンピュータ・プログラムは、伝統的な言語の著作物とは異なり、機能的な目的を果たすためのものであるという点で特徴を有する。
- 議会は、コンピュータプログラムを著作物として保護することとした。
- 結果、コンピュータプログラムは、他の著作物同様に保護される一方、他と同様に著作権の制限（フェア・ユースを含む）に服すこととなった。
- フェア・ユースは、コンピュータプログラムが著作権で保護される範囲を画する上で重要な役割を果たす。

≪フェア・ユースは事実問題か法律問題か≫
- フェア・ユースは事実と法律の混合問題であり、事実問題の部分は陪審員に委ね、法律問題は裁判官が判断するべき。
- 法律審では、法律問題の部分について、（裁量権の濫用があるかどうかだけを判断するのではなくて）一から判断すべきである。

【12】

≪フェアユースの４要素の検討≫

＜第2要素＞　⇒　フェア・ユースに有利

・コンピュータが実行する特定のタスクにどのようなラベルを付け、どのように体系化するかという点について著作物性が認められるかは議論がありうるだろう。
〔この点は、本判決では議論されない〕

・宣言コードは、コンピュータプログラムの一部であるが、タスクを分割するという一般的なシステム、タスクを体系化するというアイデア、コマンドの呼び出し方、実装コードと、密接に結びついているという点で特徴的である。
〔宣言コード・実装コードと分けられているが、別々に存在するのではなくて、1つのプログラム中に、宣言部分・実装部分という具合に連続して存在する〕

・実装コードの作成にあたっては、どうすれば効率とスピードを両立させてコンピュータを稼働させられるかなどの点に創造性を発揮する必要がある。
一方、宣言コードの場合は、プログラマーが、宣言コードの名前を覚えやすくする点に創造性が発揮された。

・SunのJava APIは、ユーザーインターフェースであり、宣言コードは、ユーザーインターフェースの一部であるため、他のプログラムとは異なる。

・宣言コードの価値は、プログラマーが時間と労力を費やしてAPIシステムを習得したことに由来する。

・以上から、仮に著作権があるとしても、宣言コードは、（コンピュータプログラムの中でさえも）著作権保護の中核から遠い存在であるといえる。

【13】

＜第１要素＞　⇒　フェア・ユースに有利

・第１要素の検討においては、従来から変容力のある利用か否かを重視してきた。

・Googleは、機能を呼び出すために宣言コードを複製したが、機能を呼び出すという目的自体は、プログラムの複製全てに共通するものなので、そこから先、より具体的な目的とその特徴について検討しなければならない。

・（より具体的に見ると）GoogleがJava APIを利用したのは、スマートフォン向けの創造的で核心的なプラットフォームを生み出すためであった。

・Googleは、再実装（既存のシステムを学習したプログラマーが、その基本的なスキルを新しいシステムで使用できるようにする）ために、必要な範囲でAPIを複製したに過ぎない。

・インターフェイスの再実装が、コンピュータプログラムの開発を促進することは、十分に陪審員に対して示されている。

・以上は、Googleの複製の「目的と性格」に、変容力があることを示している。

・非商業的な利用がフェア・ユース認定に有利であることは事実だが、Campbell事件最判で説示したように逆は必ずしも真ではない。変容力ある利用の場合はなおさら。

・同じくCampbell事件最判で説示したように、被疑侵害者の誠実・不誠実が、フェア・ユースの判断に影響を与えることには懐疑的である。
この点、一般論とは別に、本件では、他の要素がフェア・ユースの成立を支持していることと、陪審員がGoogleに有利な判断を示したことを考慮すると、なおさらである。

【14】

＜第3要素＞　⇒　フェア・ユースに有利

- Googleは、合計約11,500行を複製したが、これは、GoogleがJavaとAndroidで共通化させようと考えたAPIの宣言コードのほぼすべてに相当する。
- Java API全体は286万行にのぼるので、複製されたのは0.4%に過ぎない。
- 従来から、複製された部分が原著作物の創造的表現の核心である場合は、少量の複製であってもフェア・ユースとは言えないとする一方で、大量の複製でも、複製された部分があまり創造的表現を含まない場合や、正当な目的に必須の複製の場合は、フェア・ユースに該当するとしてきた。
- 長編小説から1行を複製しても実質的とは言えないが、世界で一番短い小説（「彼が目覚めたとき、恐竜はまだそこにいた」(注)という1行からなる）からの複製だったら話は変わってくる。
- Googleはインターフェースの再実装という変容力のある目的のために複製を行った
- 本件のように、複製の量が、有効でかつ変容力のある目的と結びついたものであるならば、第3要素は、フェア・ユースに有利傾く。
- 控訴裁は、もっと少ない複製でも、Javaプログラムを Androidで動くようにすることはできたとするが、それはGoogleの目的ではない。Googleの目的は、プログラマーが、Javaで身につけたスキルをAndroidで活用できるように再実装をすることであった。
- Googleが行った複製なくして、再実装が不可能だったことは陪審も同意するはず。

（注）　A. Monterroso, El Dinosaurio, in Complete Works & Other Stories 42 (E. Grossman transl. 1995)

【15】

＜第4要素＞　⇒　フェア・ユースに有利

- 権利者が損失を被るか否かだけではなくて、その原因も考慮すべき。
  Campbell事件最判で指摘したように、例えば、辛辣なパロディによる原作品の売上減少は、著作権法で保護される損害ではない。
- 本件では、権利者の損害と、複製が公共の利益に与える影響との比較考慮も必要
- 第1審で提示された多数の証拠によって、陪審員は、次の判断が可能であった
  - AndroidがJavaの現実の市場、または潜在的な市場に、損害を与えていないこと
  - GoogleがAPIの一部を複製したかどうかにかかわらず、Sunがスマートフォンの市場に参入できなかったであろうこと
- 潜在的な市場＝理論的な市場と解するべきではない。潜在的な市場＝理論的な市場だとすると、すべての場合に損失が概念されてしまう。
- 複製の結果、GoogleはAndroidプラットフォームから莫大な利益を得たが、その源泉は、プログラマーのJavaへの投資に大きく関係する一方で、Sun（＝Oracle）がJava APIを作成するために行った投資との関係は薄い。よって、Oracleが前記利益の分配を受ける合理的な理由がない。
- 多数のプログラマーがJava API学習のために投資をしてきたことを考慮すると、本件でOracleに著作権の行使を認めることは、公衆に害を及ぼす危険性がある。宣言コードについての権利行使を認めると、将来の新しいプログラムの創造を制限することになる。
- ①Sunはスマートフォン市場で競争する力に欠けていたこと、②Googleの利益の源泉、③公衆の創造性を害するリスクを総合すると、第4要素もフェア・ユースに有利

【16】

≪まとめ≫

- 本件では、伝統的な著作権法の概念も、フェア・ユースに関する先例も、変更していない
- Googleは、ユーザーが蓄積した能力を、新しくかつ変容力のあるプログラムに活かすために必要なものだけを抽出して、ユーザーインターフェースを再実装したのであるから、GoogleによるJava APIの複製は、法律問題としてフェア・ユースである。
- 連邦巡回控訴裁判所のこれに反する判決は取り消され、事件は、本判決に従う形でさらに手続きが進められるべく、差し戻される。

【17】

■検討

≪全体≫

- 判決自身も指摘するように、先例を、明示的には変更していない
- （Campbell事件最判で「変容力」概念が導入されたような）新概念の導入もない
  ⇒ 今後も、フェア・ユース判断にあたっては、Campbell事件最判の説示内容が基本となるだろう

<u>先例のReplacementはなかった。しかし・・・</u>

【18】

■検討

先例のReplacementはなかった。しかし、Refinement はあった。具体的には・・・

≪全体≫

・ 検討が、第2要素からはじめられたのは、最近の判決には珍しい
　⇒ 下級審は、本件の事情の故と理解するか？　それとも、これがきっかけとなって、4要素の相対化につながるか？

・ 各要素の検討において、法的な観点から結論を導くというよりも、陪審員の判断を踏まえて結論を導くような傾向がある
　⇒ 単純に言えば、既に陪審員による審理が終わっているという本件の事情を反映してのことと思われる
　⇒ ただ、本判決では、フェア・ユースが法律問題か事実問題かについて詳しく判断しているため、本判決の傾向に、下級審が（過剰反応して）影響を受ける可能性は高いのではないか？
　　例えば、サマリージャッジメント段階でのフェア・ユース認定に影響する？
　　（被疑侵害者側にとって、負担が増える？）

・著作権法が独占を認めるべき範囲はどこまでか、という問題がかなり前面に出ている
　⇒（下級審判決の）予測可能性が、従来よりも下がるのではないか？

【19】

≪第2要素≫

・ 従来、Campbell事件最判の影響でほとんど無視されてきた第2要素であるが、本判決によって、その位置づけが見直されることになるのではないか？
　⇒ パロディの事案（Campbell事件）で軽視されるのは当然だが、それ以外の本来重視すべきだった事案でも軽視・無視されてきたのは、下級審の過剰反応
・ 書きぶりを見る限り、API宣言コードの著作物性には否定的と思われる

≪第1要素≫

・ 第1要素の検討で、変容力の有無が、最重要であることに変化はない
・ 既存著作物の表現形式に変更がなくても、変容力ありと最高裁が判断したのは重要

	目的・性格の変更ある利用	目的・性格の変更なき利用 ＝代替的な利用
表現形式の変更あり	パロディ Campbell事件	単なる翻案　⇒　侵害
表現形式の変更なし	検索エンジン Perfect10事件 本件	単なる複製　⇒　侵害

・ リバースエンジニアリングや互換製品の開発は、今まで以上にフェア・ユースとされやすくなった
・ 商業性や不誠実性がフェア・ユース判断に悪影響を与えるという考え方は、既にCampbell事件最判で否定されていたが、本判決でとどめを刺された感じ

【20】

≪第3要素≫
- 許される複製の程度（量的・質的）は、利用の目的・性格に照らして判断するという、Campbell事件最判の考え方が再確認された
  ⇒ Campbell事件の場合は、質的な面が検討の中心であったが、今回は量的な面が検討の中心となった
- リバースエンジニアリング、検索エンジン、機械学習など、全部複製しないと目的を達し得ない場合について、この要件が不利に働くことはなくなった

≪第4要素≫
- Campbell事件以降、下級審は、変容力があれば市場代替性は低く、市場への害も少ない、という一種の「公式」を多用してきたが、本件ではそこに言及がない
  ⇒ 下級審（特に地裁に）にどう影響するか ⇐「巡回区判例による遅延現象」＊
- 利益の源泉を問うのは特徴的な視点
  ⇒ 著作権が保護すべき利益かどうかという問題設定につながる
  ⇒ 当然の問題設定ではあるが、予測可能性は下がるのではないか？
- 陪審員の判断に大きく依拠している
  ⇒ サマリージャッジメントなどでの判断が難しくなるのでは？ （既述）
- Nimmerの論文を引用する形で、潜在的な市場を、理論的な市場と解すべきではないことが明記されたのは大きい
  ⇒ この考え方は、日本の権利制限規定の、「権利者の利益を不当に害す場合」の判断にも示唆的では？ （権利制限規定に基づいて利用が行われれば、大なり小なり権利者に不利益が発生する。問題は「不当」かどうか）

【21】

■トーマス判事の反対意見 （⇒それに対する簡単なコメント）

- 宣言コードと実装コードは密接に結びついており、分離して考えるのは間違い
  ⇒ 小説でも同じだが、そこから、非創作的部分を特定するのは普通ではないか？
- AppleやMicrosoftが独自の宣言コードを書くことができた以上、融合理論は非適用
  ⇒ AppleなどのAPIは、Javaとは別物ではないか？ アイデアが多数あるに過ぎない。
- 法廷意見が、法定の順番でも重要性（①Harper&Row最判に基づき第4要素 ②第1要素）でもなく、第2要素から検討し、宣言コードを保護の必要性が薄いものとした上で、他の要素を検討しているのは大きな間違い。結果、フェア・ユース判断が間違ってる
  ⇒ 第4要素が最重要というのは現状少数説
- 全ての著作物の表現はアイデアと結びついているし、著作物の習熟にその利用者が投資するのは宣言コードに限らない。法廷意見の第2要素判断は間違い。
- Googleの行為によって、Oracleは多大な損害を被っている（Javaプラットフォームの価値が下がり、またスマフォメーカーにJavaPFのライセンスする機会を失った）
- 商業性は決め手ではないが、何百億ドルもの無許諾複製をFUとしたことはない
  ⇒ 著作権で保護されるべき損害かという視点に欠ける。第4要素が最重要という少数説の立場の帰結
- GoogleはOracleと同じ目的で宣言コードを複製しており、変容力はない。法廷意見では、変容力がある＝他者が新しいプログラムを創作する、であり間違い。
  ⇒ 法廷意見の指摘のように、機能を呼び出すことを利用目的と捉えると、全ての場合となる
- 宣言コードは核心部分であり、変容力がない以上、大量の複製は正当化不可

【22】

〔参考〕「巡回区判例による遅延現象」について

- 「巡回区判例による遅延現象」は、奥邨が名付けたものであり、一般的用語ではない
- 著作権分野で新しい最高裁判決が出た後も、地裁判決にその影響が現れるのに時間がかかる状況がしばしば見られる
- この原因は、地裁にとって、従うべき判例には、最高裁のものだけでなく、自身が属する巡回区の控訴裁のものも含まれていることが指摘できる
- （新しい）最高裁判例と、（従来からある）控訴裁判例が、必ずしも整合しないとき、地裁としてはどうすべきかジレンマに陥る
  - ⇒ 本来は、控訴裁判例を無視すべきなのだろうが、なんとか整合させようと工夫を凝らすケースが少なくない。結果、最高裁判例の影響が減衰する。
  - ⇒ 最高裁判例の対象となっていない巡回区で生じやすい。
    例えば、最高裁が、第3巡回区控訴裁の判決を破棄差し戻しした場合、第3巡回区については控訴裁判例が軌道修正されるが、他の巡回区はそうではない
    そのため、例えば、第7巡回区の地裁は、（整合しなさそうに見える自身の巡回区）控訴裁判例を、どう扱うべきかジレンマに陥る
- 控訴裁が、その判例について、軌道修正を行ってくれるまで、この現象は続く
  - ⇒ よって「巡回区判例による遅延現象」

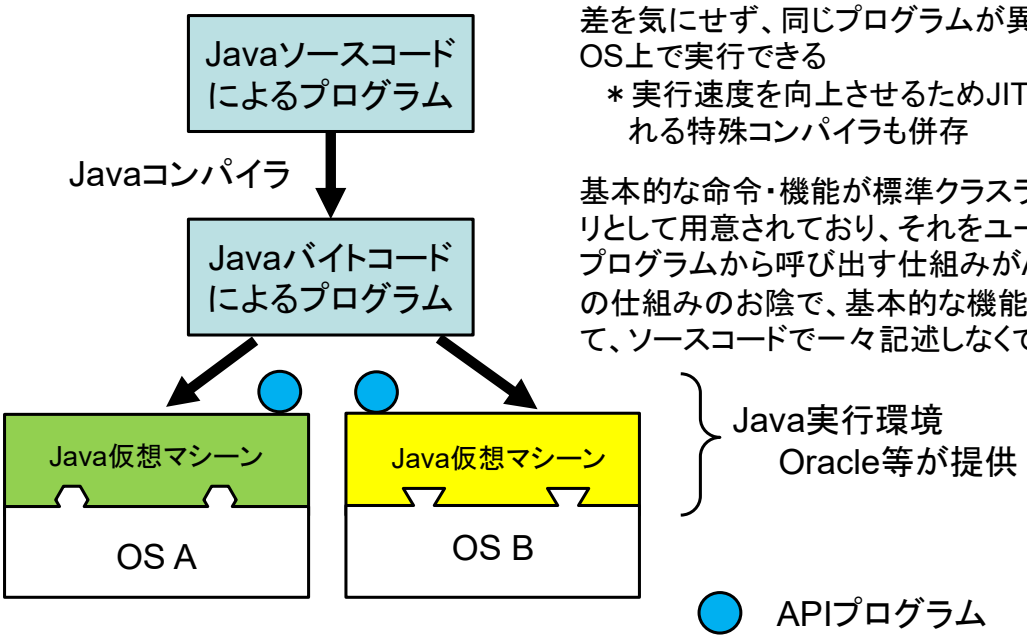
【23】

▸Javaの仕組み

Javaは、言語仕様、コンパイラ、実行環境（仮想マシーン＋APIプログラム）からなる

仮想マシーン（バイトコードを機械語に変換するインタプリター*）のお陰で、OS毎の差を気にせず、同じプログラムが異なるOS上で実行できる
＊実行速度を向上させるためJITと呼ばれる特殊コンパイラも併存

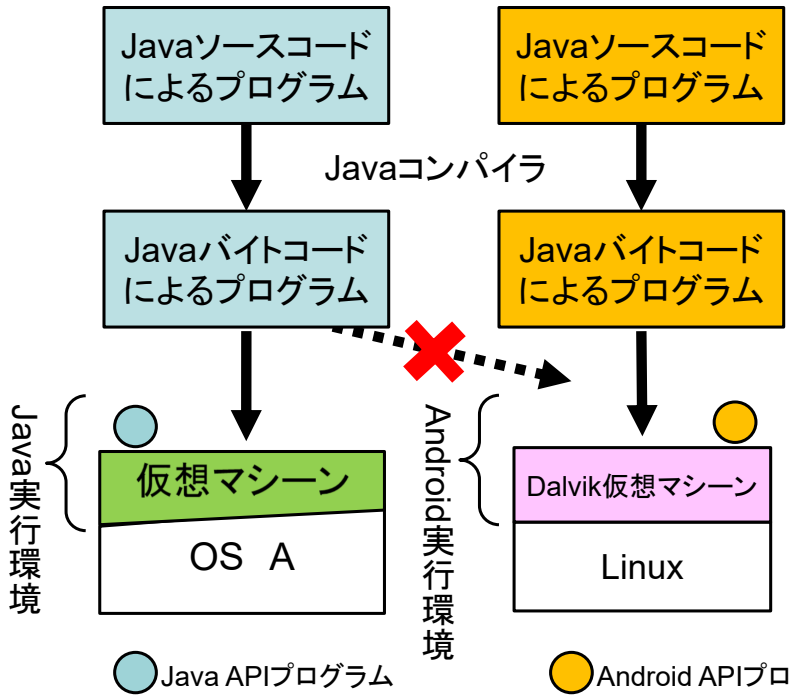
基本的な命令・機能が標準クラスライブラリとして用意されており、それをユーザープログラムから呼び出す仕組みがAPI。この仕組みのお陰で、基本的な機能について、ソースコードで一々記述しなくてすむ



【24】

▸事実関係

Android実行環境はGoogleが提供



〔ポイント〕
同じJava言語で記述されていても、Android用のソフト以外は、Androidでは実行できない

⇩

ソフトの互換性なし

⇩

G社はAndroid用ソフトの開発者としてJavaプログラマの取り込みを狙った

【25】

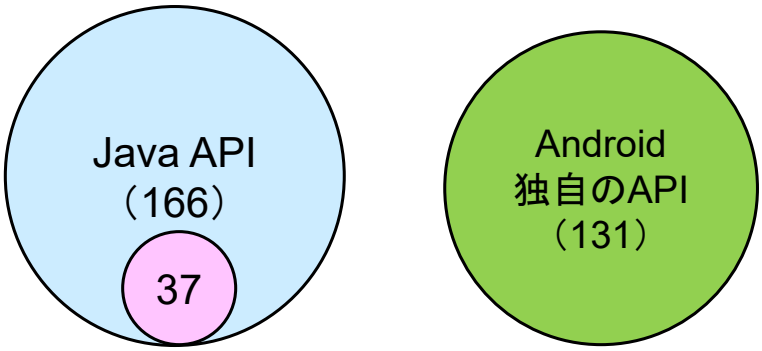
▸事実関係

AndroidのAPIは・・・

総計166パッケージあるJava APIの一部（37パッケージ）

＋

Android独自のAPI（131パッケージ）

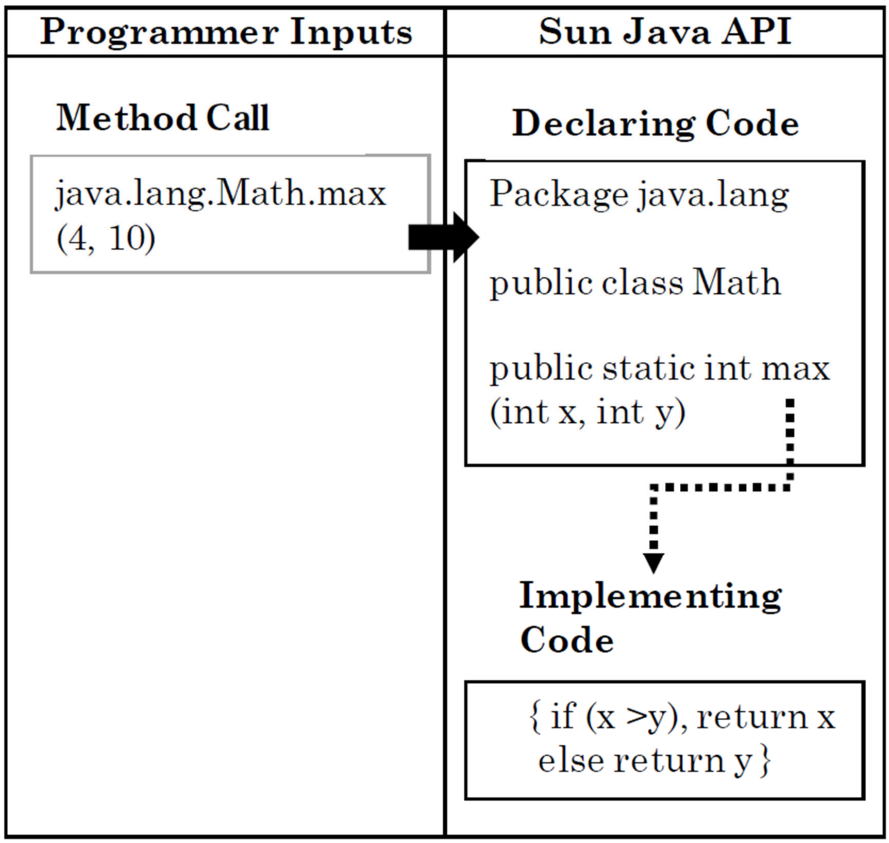


Java API: 166パッケージ、3000クラス、30000メソッド

G社が利用した一部：（赤い部分） 37パッケージ、600クラス、6000メソッド

【26】

Sun Java API Diagram



API プログラム【パッケージ　java lang：クラス　Math】のソースコード　のイメージ

黄色マーカーは宣言コード

クラス Math を宣言

メソッド sin を宣言

水色マーカー部分は実装コード

```
package java.lang;
import java.math.BigDecimal;
public final class Math {
    private Math() {}
    public static double sin(double a) {
        return StrictMath.sin(a); // default impl. delegates to StrictMath
    }
    public static double cos(double a) {
        return StrictMath.cos(a); // default impl. delegates to StrictMath
    }
    public static int round(float a) {
        int intBits = Float.floatToRawIntBits(a);
        int biasedExp = (intBits & FloatConsts.EXP_BIT_MASK)
                >> (FloatConsts.SIGNIFICAND_WIDTH - 1);
        int shift = (FloatConsts.SIGNIFICAND_WIDTH - 2
                + FloatConsts.EXP_BIAS) - biasedExp;
        if ((shift & -32) == 0) { // shift >= 0 && shift < 32
            int r = ((intBits & FloatConsts.SIGNIF_BIT_MASK)
                    | (FloatConsts.SIGNIF_BIT_MASK + 1));
            if (intBits < 0) {
                r = -r;
            }
            return ((r >> shift) + 1) >> 1;
        } else {
            return (int) a;
        }
    }
    public static long round(double a) {
        long longBits = Double.doubleToRawLongBits(a);
        long biasedExp = (longBits & DoubleConsts.EXP_BIT_MASK)
                >> (DoubleConsts.SIGNIFICAND_WIDTH - 1);
        long shift = (DoubleConsts.SIGNIFICAND_WIDTH - 2
                + DoubleConsts.EXP_BIAS) - biasedExp;
        if ((shift & -64) == 0) { // shift >= 0 && shift < 64
            long r = ((longBits & DoubleConsts.SIGNIF_BIT_MASK)
                    | (DoubleConsts.SIGNIF_BIT_MASK + 1));
            if (longBits < 0) {
                r = -r;
            }
            return ((r >> shift) + 1) >> 1;
        } else {
            return (long) a;
        }
    }
```

API プログラム【パッケージ　java.lang：クラス　Math】のソースコード　のイメージ


```
package java.lang;
import java.math.BigDecimal;
public final class Math {
    private Math() {}
    public static double sin(double a) {
        return StrictMath.sin(a); // default impl. delegates to StrictMath
    }
    public static double cos(double a) {
        return StrictMath.cos(a); // default impl. delegates to StrictMath
    }
    public static int round(float a) {
        int intBits = Float.floatToRawIntBits(a);
        int biasedExp = (intBits & FloatConsts.EXP_BIT_MASK)
                >> (FloatConsts.SIGNIFICAND_WIDTH - 1);
        int shift = (FloatConsts.SIGNIFICAND_WIDTH - 2
                + FloatConsts.EXP_BIAS) - biasedExp;
        if ((shift & -32) == 0) { // shift >= 0 && shift < 32
            int r = ((intBits & FloatConsts.SIGNIF_BIT_MASK)
                    | (FloatConsts.SIGNIF_BIT_MASK + 1));
            if (intBits < 0) {
                r = -r;
            }
            return ((r >> shift) + 1) >> 1;
        } else {
            return (int) a;
        }
    }
    public static long round(double a) {
        long longBits = Double.doubleToRawLongBits(a);
```

■そのAPIは何を指していますか？

APIー般と考えると間違い

- Google v. Oracle事件に関する議論では、Java APIという言葉で何を指しているかが結構混乱気味ではないだろうか？
- Java APIは、Javaプラットフォームが用意する標準クラス・メソッド（≒基本命令とその機能）を利用する仕組みのこと
- 問題はこの「仕組み」の中に実は3つのものが混じっていること

①標準クラス・メソッド（≒命令とその機能）の呼び出し方法

②呼び出される標準クラス・メソッド（≒命令とその機能）そのもの

③標準クラス・メソッド（≒命令とその機能）を実現するプログラム

例えば、判決がJava APIの宣言コードと名づけているのは、それは③のクラス・メソッドの宣言部分という意味。決して①の意味ではない！
今回の問題は、③のプログラム中の、①や②の影響を強く受ける部分（宣言コードやその構成）に著作物性があるか否か、利用はフェア・ユースか否か。
①や②については議論の対象外。
そもそも、①や②は、方法や機能なので、著作権法上はアイデアの部類であり保護の対象外のはず。

▸事実関係

Java APIとは何か・・・

⇒ 一般に、API（Application Program Interface）とは、アプリケーションプログラムが、OSなどの機能を利用するために、OSなどに予め用意されている仕組みをいう

⇒ 例えば、MS-Wordでファイルを HDDに保存する機能について考えると、Wordのプログラマは、自分でそういう機能をプログラムする必要はなくて、OSに用意されている仕組みを利用すれば済むのと同じようなイメージ

⇒ Javaの場合、Java仮想マシーンの機能を利用する仕組みという意味で、APIと呼ばれているが、標準クラスライブラリと呼ばれることもある

⇒ クラスというのは、昔々のプログラミングモデルでいえば、モジュールとかサブルーチンに近いイメージ

【31】

▸事実関係

Java APIとは何か・・・

⇒ つまり、頻繁に利用されるモジュール群を、Oracle側が作成してライブリーとして配布してくれていることになる

⇒ ここで、重要なのは、Javaのプログラマにとって、標準クラスライブラリに含まれるクラスは、Java言語の命令語の一種のような感じで使われていることである

⇒ 例えば画面に円を描く場合、本来なら、sin・cosなどの三角関数を利用して1ドットずつ表示位置を求めていくことになる
しかし、Javaの標準クラスライブラリ（＝API）には java.awt.Graphicsというクラスが用意されており、当該クラスのdrawOvalという機能（＝メソッド）を利用すれば、四角形の４つの頂点を与えるだけで、それに内接する円を描いてくれる

⇒ つまり、円を描く専用命令みたいなもの

【32】

▸事実関係

Java APIとは何か・・・

⇒ このように、モジュールを一種の専用命令的に再利用しやすくするのは、オブジェクト指向言語（C++やJavaが代表）の特徴

⇒ なお、Javaプログラマは、各クラス（＝モジュール）がどういう仕組み（＝コード）で実現されているかは知らない。知っているのは、クラスの名前と、その機能（＝メソッド）、処理して欲しいデータ（＝引数）の渡し方、などだけである

＊ 命令語の場合に、命令語の名前と文法さえ知っていれば利用できるのと同じ

例： クラス名： java.awt.Graphics
メソッド名 ： drawOval (左隅X, 左隅Y, 幅, 高さ)

⇒ 同じ機能を果たすものでも、クラス名やメソッド名が異なると、プログラマは容易に見つけられない

【33】

▸事実関係

⇒ もっとも、G社は37のJava APIを丸々コピーしたわけではない

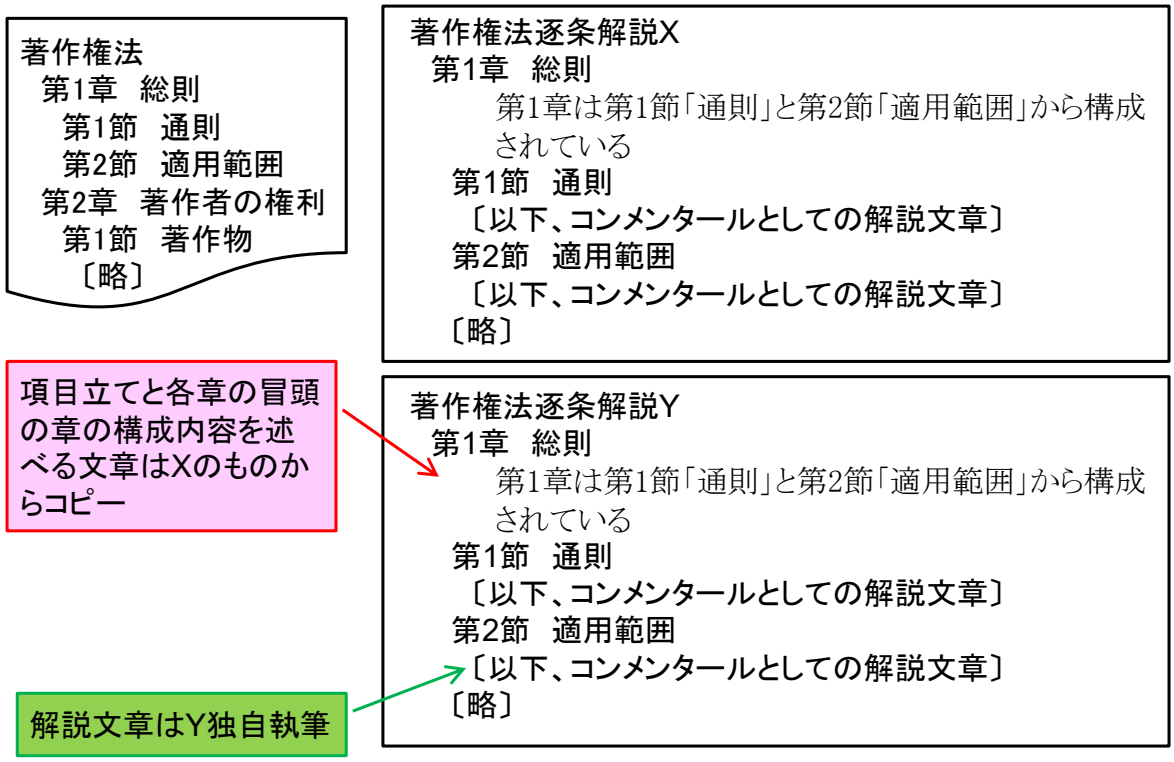
⇒ G社は、API（＝クラス）の中身のコードは、全部自分たちで一から記述した（公開されている、各クラスの機能についての資料をもとにして独自創作したものと思われる）。O社もその点で、著作権侵害があったと主張しているわけではない。

⇒ G社がコピーしたのは、クラスの名前、メソッドの名前、そしてパッケージ内でのクラスの構成、各クラスに含まれるメソッドの組合わせだった。

⇒ そして、それを可能にするために、APIの宣言コード（クラスの名前が一覧化されている）を逐語的に複製した

【34】

今回の問題を喻えると・・・



【35】

今回の問題を喻えると・・・

Xは、著作権法の項目（章・節・款・条）・その体系・見出しをそっくりそのまま利用した、逐条解説を執筆。各章・各節・各款・各条の冒頭には、その章・節・款・条の構成内容を簡潔に述べる行（例：第１章は、第１節「通則」および第2節「適用範囲」から構成される）が存在。その行の後、詳しい解説文章が続く。

Yは、Xの逐条解説の項目（章・節・款・条）・その体系・見出しと、各章・各節・各款・各条の冒頭の、その章・節・款・条の構成内容を簡潔に述べる行の全てをそのまま複製。具体的な解説文章自体は、Yが独自に著作。

著作権法の項目・その体系・見出し	⇒	Java言語の命令語や文法に相当（クラスなど）
各章などの冒頭の簡単な文章	⇒	宣言コードに相当
詳しい解説文章	⇒	実装コードに相当

【36】

Google v. Oracle事件最高裁判決

【37】

Google v. Oracle
事件の経過

2022年12月1日

弁護士　石新智規

シドリーオースティン
法律事務所・外国法共同事業
SIDLEY AUSTIN | FOREIGN LAW JOINT ENTERPRISE

背景事情

1996年 ❖ Sum MicrosystemsがJavaを開発
Javaは、その後、最も人気のあるコンピュータ言語のひとつとなる。

2005年 ❖ Googleは携帯電話のプラットフォーム構築のためにAndroid社を買収する。
- Androidプラットフォームとして Java Platformを採用することを検討
- Sum Microsystemsとライセンス交渉を開始したが、最終的に交渉はまとまらず
- Googleはライセンスを断念、アンドロイドプラットフォームを自社で開発することに
- その過程で、Java APIの166パッケージのうち37のパッケージの名称と機能を複製したが、実装プログラムは自社で開発

2007年 ❖ Google は携帯電話向けのAndroid プラットフォームを公開

2008年 ❖ Android携帯の販売開始

2010年 ❖ OracleはSun Microsystemsを買収し、Oracle Americaと社名を変更

Java APIをめぐる紛争の経緯

2010年 8月 ❖ Oracle：Javaの特許権および著作権を侵害されたと主張し、Googleを提訴

2012年 5月 ❖ カリフォルニア北部地区連邦地裁（以下「連邦地裁」）の陪審：著作権侵害あり（陪審は著作物性を前提とするよう指示される）、特許侵害なしと評決（フェアユースについて意見が分かれ結論でず）。連邦地裁：Java APIのGoogleが複製した部分は102条(b)に基づき、著作権で保護されないと判断。

2014年 5月 ❖ 連邦巡回控訴裁判所（以下「CAFC」）：Java API（Declaring codeとSSO）は著作権で保護されると判断し、フェアユースの審理のため事件を連邦地裁に差し戻す。

10月 ❖ Google: 著作物性を肯定したCAFCの判断に対する裁量上訴

2015年 1月 ❖ 連邦最高裁：Solicitor Generalに対し、政府の見解を提出するよう要請

6月 ❖ 連邦最高裁：Googleの裁量上訴の申立てを受理せず（第1ラウンド終了）

2016年 5月 ❖ 連邦地裁：GoogleによるJava APIの複製はフェアユースに当たると判断（第2ラウンド）

2018月 3月 ❖ CAFC：GoogleによるJava APIの複製はフェアユースに当たらないと判断

8月 ❖ CAFC：全裁判官による審理を求めるGoogleの請求を却下

2019年 1月 ❖ GoogleがCAFCの判断に対して再び連邦最高裁に裁量上訴の申立て

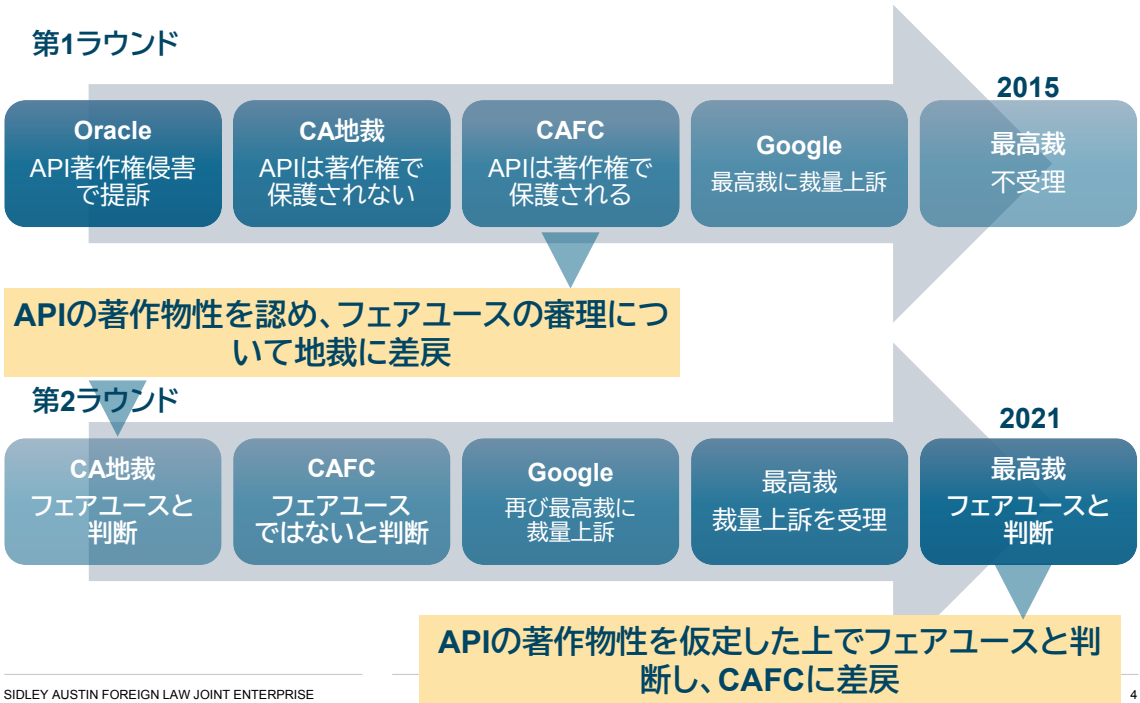
4月 ❖ 連邦最高裁：Solicitor Generalに対し、政府の見解を提出するよう要請

11月 ❖ 連邦最高裁：Googleの裁量上訴を受理

2020年10月 ❖ 連邦最高裁：口頭弁論

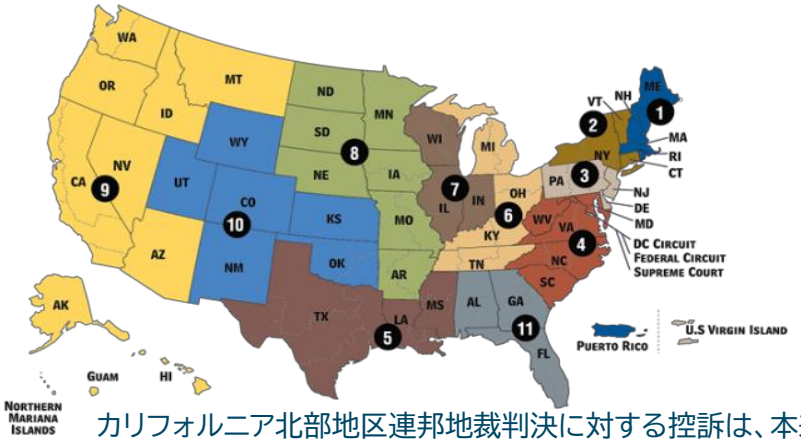
2021年 4月 ❖ 連邦最高裁：GoogleによるAPIの複製をフェアユースと判断し、差戻。

著作物性とフェアユースの各論点で地裁と控訴審の判断が割れた



巡回区控訴裁判所（Circuit Court）の管轄

94の連邦地裁裁判所が12の巡回区控訴裁判所の管轄下にあり、各連邦裁判所の判断は、連邦最高裁と各裁判所を管轄する巡回区控訴裁判所の判断に拘束される。

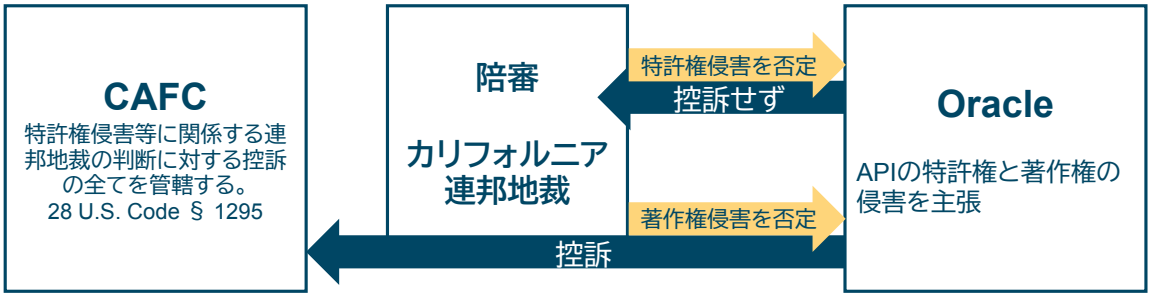


カリフォルニア北部地区連邦地裁判決に対する控訴は、本来、第9巡回区控訴裁判所で審理されるはず。本件で、なぜCAFCが審理したのか？

https://www.uscourts.gov/about-federal-courts/court-role-and-structure

連邦控訴裁判所（CAFC）の管轄

- 特許権侵害等に関係する連邦地裁の判断に対する控訴の全てを管轄する。
- 本件でOracleはAPIの特許権と著作権の侵害を主張していた。
- Oracleは特許権侵害を否定した陪審の判断については控訴せず、APIの著作物性を否定（よって著作権侵害も否定）したカリフォルニア連邦地裁の判断のみを控訴したが、当初の請求に特許権が関係しているため、第9巡回区ではなく、連邦控訴裁判所が審理することとなった。
- ただし、著作権についてCAFCは専属管轄を有しないので、CAFCは、原審を管轄する第9巡回区控訴裁判所の先例に照らし、原審の是非を判断する。*Atari Games Corp. v. Nintendo of Am., Inc., 897 F. 2d 1572, 1575 (Fed. Cir. 1990)*



第1ラウンドの争点

- Googleが（166あるAPIパッケージのうち）37のdeclaring codeの名称とその体系（package-class-method）（37のAPIの3%）を複製した事実とimplementing codeを自ら作成した事実（37のAPIの97%）について争いはない。
- 争点は、declaring code の名称とその体系が著作物性を有するか否か、言い換えると、declaring codeの名称及びその構成・体系は、method of operation として、102条(b)に基づき著作権の対象から除外されるのか。一定の創作性がある以上（102条(a)）、著作権で保護されるのか。

102条

- (a)Copyright protection subsists, in accordance with this title, in original works of authorship fixed in any tangible medium of expression, now known or later developed, from which they can be perceived, reproduced, or otherwise communicated, either directly or with the aid of a machine or device. Works of authorship include the following categories:
- (1)literary works;
- (2)musical works, including any accompanying words;
- (3)dramatic works, including any accompanying music;
- (4)pantomimes and choreographic works;
- (5)pictorial, graphic, and sculptural works;
- (6)motion pictures and other audiovisual works;
- (7)sound recordings; and
- (8)architectural works.
- (b)In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work.

102条 (a)

- コンピュータプログラムは1980年に著作権の保護対象として加えられた。

➢A “computer program” is a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result.

- 独立創作されたオリジナリティーと最小限度の創作性は要求される。
  Feist Publications, Inc. v. Rural Telephone Service Co., 499 U. S. 340, 345 (1991)

102 (b)

(b)　いかなる場合にも、著作者が作成した創作的な著作物に対する著作権による保護は、着想、手順、プロセス、方式、操作方法、概念、原理または発見(これらが著作物において記述され、説明され、描写され、または収録される形式の如何を問わない)には及ばない。

訳は、著作権情報センターの翻訳を利用させていただいた
https://www.cric.or.jp/db/world/america/america_c1a.html#102

Declaring code の著作物性

連邦地裁

ルールに従う以上、プログラマーは同じ機能を指定する方法を示すために、同一のdeclaration／header を利用しなければならない。Java パッケージのそれぞれに対するdeclaring codeを書く方法は一つしかない、よって、Merger理論により、著作権性が認められない。

Declaring code は、著作権の保護はショートフレーズで構成されているので、著作物性が認められない。

37のJava APIパッケージのグルーピングやコードがありふれているものであるという主張を支える証拠がないとして、Scenes a Faire（ありふれた表現）の抗弁は認めなかった。

CAFC

- Mergerは抗弁であり、著作物性の問題ではないとした上で、それしか表現の方法がないか、他の表現手法が極めて限られている場合でなければ、Mergerは成立しない→3つのパッケージを除き本件ではMergerを否定した。
- Googleが複製した7000 linesの選択と配列には無制限の選択があり得た。例えば、“java.lang.Math.max”というcodeの名称は、“Math.maximum”、“Arith.larger”などでもよく、そこに選択の余地があった。
- 著作物性の問題は、Googleによる「複製時」ではなく、API パッケージの「創作時」の選択可能性を検討しなければならない。Sun Microsystems（当時）は、様々な呼称が可能な中で、“java.lang.Math.max”という名称で呼ぶことを選択した。

ショートフレーズの集合体（編集物に類似）をとらえれば、十分に創作性があり、ショートフレーズドクトリンの適用は誤り。

Scenes a Faireの抗弁も、「複製時」の状況ではなく、「創作」時の状況に基づいて判断されるべきで、創作時において、外的要因によって、そのような表現にならざるを得ないという証拠がなければならないが、提示されなかった。

APIの構成・体系の著作物性

連邦地裁

第9巡回区控訴裁判所の先例に従えば、APIの構成・体系（いわゆるSSO）は著作権によって保護される可能性はある。
Johnson Controls v. Phoenix Control Sys., 886 F.2d 1173, 1175 (9th Cir. 1989).

しかし、102(b)に照らし、相互運用性(interoperability)の確保のために不可欠な機能的な要素部分には著作物性はない。*Sega Entertainment Ltd. v. Accolade, Inc., 977 F.2d 1510 (9th Cir. 1992)*

Sega判決は、Sega社製のコンソール上で作動するゲームカートリッジを開発するため、相互運用に不可欠なインターフェースを確認するためにソースコードを複製することは、102(b)で保護されない機能的部分を確認する行為としてフェアユースであると判断した。

CAFC

原審が依拠したLotus判決（第1巡回区控訴審裁判所）は、第9巡回区控訴裁判所の先例ではない。第9巡回区の先例は、SSOの著作物性を肯定する。
Johnson Controls (1989), Atari Games (1990)

Lotus判決：Commandのヒエラルキー（階層）は、102(b)の「操作方法」（method of operation）であるため、著作権による保護を受けないと判断し、さらに、特定の言葉がある動作に不可欠なものである場合には、その言葉が「操作方法」の一部になるため著作権の保護対象とならないと判示。*Lotus Development Corp. v. Borland International, Inc., 49 F.3d 807 (1st Cir. 1995), aff'd without opinion by equally divided court, 516 U.S. 233, 116 S. Ct. 804, 133 L. Ed. 2d 610 (1996)*

コンピュータプログラムは本来機能的なものであり、機能という性質を理由にカテゴリカルに102(b)で保護を否定されない。機能を表現する方法が複数あり得るのであれば、その表現は保護される

相互運用性（interoperability）は著作物性の問題ではなくフェアユースの問題である。

第2ラウンドの争点

- 第1ラウンドで連邦地裁は、陪審員に対し、Declaring codeの名称とその構成・体系（API)には著作物性があることを前提として、APIの複製の有無とフェアユース抗弁の成否について判断（事実認定の問題）するよう指示をしたが、陪審員はフェアユースについて判断が分かれた。
- 原審：APIの著作物性を否定したので、フェアユースに関する再審理をしなかった。
- CAFC：APIに著作物性を認めたものの、GoogleのAPI複製がTransformativeといえるか、相互運用性のために複製することが必要であるか（Javaを利用する者にとって利用が不可避なものであるか）、GoogleのAndroidへのAPIの複製によりOracleが損害を蒙っているかなどの点について十分な証拠を有していないことを理由に、フェアユースについて再審理を求め、地裁に差し戻した。
- 再度の審理で陪審員は、10人全員一致でGoogleによるAPIの複製をフェアユースと判断

107条

第106条及び第106A条の規定（筆者注、著作権の支分権規定）にかかわらず、批評、解説、ニュース報道、授業（教室における利用のために複数のコピーを作成する行為を含む）、研究又は調査等を目的とする、複製物、レコードへの複製又は同条に定められるその他の手段による利用を含む著作物のフェアユース（公正利用）は、著作権の侵害とならない。

特定の場合に著作物の利用がフェアユースとなるか否かを決定する際に考慮すべき要素は、以下を含むものとする。

(1) 利用の目的及び性質（当該利用が商業性を有するか否か、又は非営利的な教育目的か否かを含む）

(2) 著作物の性質

(3) 著作物全体との関連において利用された部分の量及び重要性

(4) 著作物の潜在的市場又は価値に対する当該利用の影響

上記のすべての要素を考慮してフェアユースが認定される場合、作品が未発行であるという事実それ自体は、フェアユースの認定を妨げないものとする。

APIの複製はフェアユースか

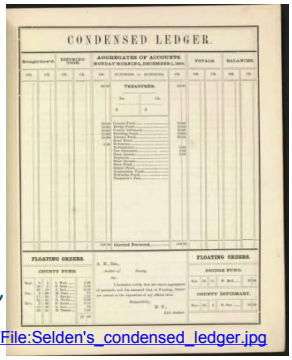
連邦地裁	CAFC
(1) スマートフォン用に必要な範囲でのみ複製されており、自らImplementation codeを付加しているから、商業性はあるが、トランスフォーマティブである。	(1) Declaring codeを原著作物と全く同じ機能と目的で複製しており、何らかの表現やメッセージの追加がない。スマートフォンに利用するという利用形態の変化だけではトランスフォーマティブと評価するに不十分である。
(2) Declaring codeとその構成・体系は著作物性を認めるために十分に創作性を有するものであるが、非常に高い創作性を有するものとまでは言えない。	(2) 創作性は認められるが、declaring codeの一部はJavaを利用する際に不可欠なものであり、機能性と創作性を区別するのは困難。機能性の強い著作物である。
(3) Googleはトランスフォーマティブの目的のために合理的に必要と考えられる量の複製をしているだけで、複製されたlineの量はわずかである。	(3) 複製された量は多くないが、Javaの利用に不可避な部分(170 lines)以上 (11,500 lines) の複製をしており、複製された部分は、Androidプラットフォーム構築にとって非常に重要なものであった。
(4) AndroidにおけるAPIの利用は、デスクトップやラップトップコンピュータの市場に害を及ぼすものではない。	(4) JavaはAndroid以前より携帯電話向けにライセンスされており競合している。Amazon KindleはJava SEからAndroidに切り替えたという事実もあり、実損が生じている。また、スマートフォン市場にOracleが参入していないとしても、スマートフォンへのライセンスは企図していた。
フェアユース(全要素)	フェアユースではない(第1と第4)

Baker判決と102(b)の関係

- *Baker v. Selden, 101 U.S. 99, 101 (1879)*

会計士Charles Seldenは、会計帳簿の独特のシステム(項目区分、項目名称など)の使い方を説明する文章とフォームからなる書籍を発行したが、Seldenの死後、類似するフォームを含む帳簿を発行したBakerに対し、遺族が、書籍の著作権に基づき著作権侵害訴訟を提起した事件。

- 書籍に記載された技芸(art)に対し財産権を著作者に付与することは、公衆に対する不意打ちかつ詐欺となる。それは、著作権ではなく、特許権の対象である。


CONDENSED LEDGER

- 102条(b)はこの判決を具体化したものと解されている。
- アイデアと表現の二分論
- Merger理論(アイデアの表現手段が限られる場合)
- 例示された除外対象の意義 (Lotus判決)

反対意見、CAFC、機能性があるからといって、102(b)で保護は否定されない

https://en.wikipedia.org/wiki/Baker_v._Selden#/media/File:Selden's_condensed_ledger.jpg
https://tile.loc.gov/storage-services/service/rbc/rbc0001/2011/2011gen155867/2011gen155867.pdf

コンピュータプログラムの著作権法による保護

Whelan (1986)
- デンタルラボ（在庫管理・顧客管理・請求を効率的に行うシステム）のSSOの著作物性を認め、異なるプログラミング言語・アルゴリズムを利用してSSOだけを複製した場合にも著作権侵害を肯定。
- デンタルラボ管理の効率化を実現するプログラムの構成（SSO）は様々あり得る。アイデアに必然的に必要な部分以外は全て「表現」である。

Altai (1992)
- 保護されないideaの範囲を非常に狭く解するWhelan判決を批判
- Abstraction-Filtration-Comparison Test
- ①効率性によって決定づけられるプログラム設計の要素、②プログラムがともに機能するハードウェア・ソフトウェアといった外部要因によって制約を受けるプログラム設計、③プログラムのパブリックドメインの要素（一般化したプログラミング技術など）を著作権で保護される対象から除外（フィルタリング）

Lotus (1995)
- 被告のBorlandは、原告LotusのLotus1-2-3のメニューコマンド構造とは異なるスプレッドシート（ユーザーインターフェース）を自ら開発していたが、Lotus1-2-3の既存利用者の移行を誘引するため、Lotus1-2-3で構築されたMacro（命令）をそのままBorlandのスプレッドシートで利用できるようにするエミュレーション機能を付加する目的でLotus1-2-3のメニューコマンド構造を複製した。
- Lotus 1-2-3のコマンド構造は「method of operation」あるとして102(b)を理由に著作物性を否定

2010 Oracleによる提訴

本件下級審の位置づけ

- CAFCは、Declaring codeの体系（SSO）の著作物性を肯定するにあたり、第9巡回区控訴裁判所の先例として、SSOの著作物性を肯定した*Johnson Controls (1989)* とこれを引用する *Atari Games (1990)* を指摘。
- *Johnson Controls* は仮処分事件の上訴に過ぎず、本案判決ではないため先例としての重みは大きくない。また、*Atari*もCAFCが第9巡回区の先例を検討したもの。
- 90年代以降、第9巡回区控訴裁判所は、*Altai*のアプローチを是認している。*Sega Entertainment Ltd. v. Accolade, Inc., 977 F.2d 1510,1525 (9th Cir. 1992)*
- *Johnson Controls*は、コンピュータプログラム著作権保護の裁判例の歴史からみれば、*Altai*前の80年代のものであり、今回、第9巡回区控訴裁判所が本件を審理していれば、*Altai*又は*Lotus*のようなアプローチを採用していた（地裁判決が是認されていた）可能性はあった。
- カリフォルニア北部連邦地裁の判決に対する控訴をCAFCが担当したことは、Oracleには有利に働いたように思われる。

Beijing
Boston
Brussels
Century City
Chicago
Dallas
Geneva
Hong Kong
Houston
London
Los Angeles
Munich
New York
Palo Alto
San Francisco
Shanghai
Singapore
Sydney
Tokyo
Washington, D.C.

Java (ジャバ/ジャヴァ) 概説

富士通株式会社
ジャパン・グローバルゲートウェイ
アドバンスドテクノロジー推進統括部
シニアディレクター
野山 孝太郎 (noyama@fujitsu.com)

アジェンダ

FUJITSU

- Java とは
- Java Specification Requestとは
- Java APIとは

© 2022 FUJITSU LIMITED

Java とは

© 2022 FUJITSU LIMITED

Java とは

○**プログラミング言語**の名前。
- ○1995年にサン・マイクロシステムズ社によってリリースされた。
  2010年に同社がオラクル社に買収された事によって版権が移行。
- ○特徴は、**オブジェクト指向**、**プラットフォーム非依存**、マルチスレッド、メモリ操作の隠蔽など。

○または、Java (プログラム) を開発し動作させる為の
プラットフォーム (開発環境・実行環境) の名前。

【参考】What is Java technology and why do I need it?

© 2022 FUJITSU LIMITED

【ご参考】代表的なプログラミング言語

FUJITSU

言語	初版	主な用途					実行モデル			プログラムの特徴									型付け							
		Webアプリケーション	デスクトップアプリケーション	モバイルアプリケーション	サーバーアプリケーション	AI・機械学習	ネイティブ	JIT	インタプリタ	手続き型（命令型）	オブジェクト指向	関数型	コンポーネント指向	イベント駆動型	ジェネリック	リフレクション	並列コンピューティング	マルチプラットフォーム	宣言型	型推論	強い	弱い	安全である	安全でない	静的	動的
C	1973年				○		○			○												○		○	○	
C++	1980年				○		○			○	○				○				○		○			○	○	
C#	2000年	○	○	○	○			○	○	○	○	○			○			○			○		○	○	○	
																					○		○			○
COBOL	1959年				○		○			○	○										○	○				
Go	2009年	○		○			○			○	△			○		○	○			○	○				○	
Java	1994年	○	○	○	○			○	○	○	○				○			○			○				○	
JavaScript	1995年	○							○	○	○	○									○					○
Objective-C	1986年		○	○			○			○	○										○				○	○
Perl	1987年	○						○		○	○	○									○					○
PHP	1995年	○							○	○	○										○					○
Python	1991年	○				○			○	○	○	○									○					○
Ruby	1995年	○							○	○	○	○									○					○
Swift	2014年		○	○			○			○	○	○			○					○	○				○	
VisualBasic	1991年		○				○						○	○					○		○		○		○	○
VisualBasic.NET	2001年		○					○			○			○							○				○	

© 2022 FUJITSU LIMITED

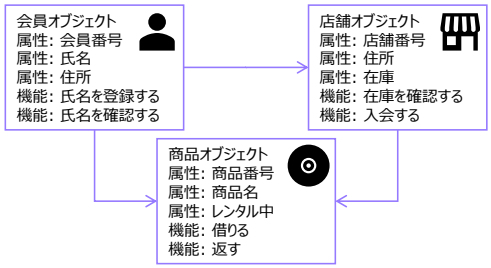
オブジェクト指向 とは

FUJITSU

○プログラミングパラダイム (プログラムの設計の考え方) の一つ。

○有名なプログラミングパラダイムとしては他に手続き型 (COBOL、C言語) などがある。

「オブジェクト指向」の考え方



実現したい処理に関係する概念を
属性や機能を持つ「**オブジェクト**」として表現する

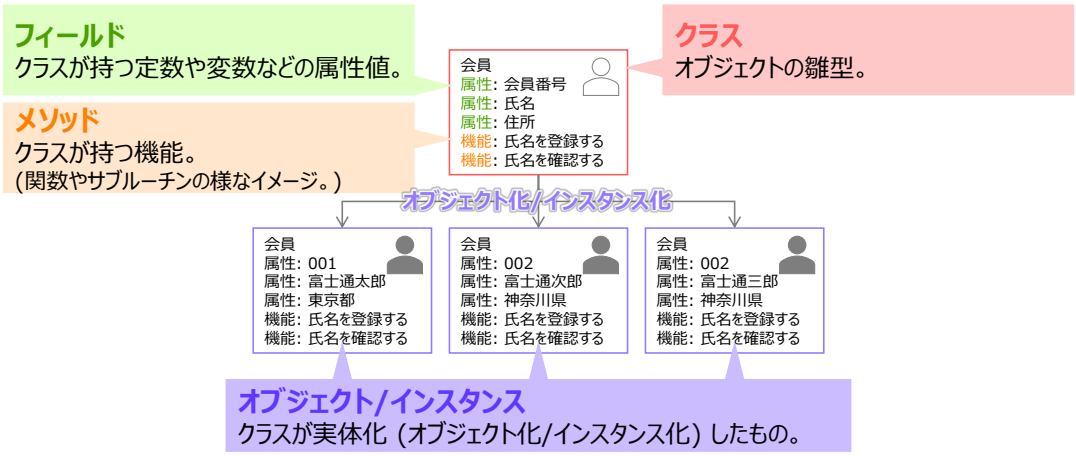
「手続き型」の考え方

1. 商品番号を入力する
もし商品が複数ある場合
　① 1.の処理を繰り返す
2. 会員番号を入力する
もし会員ではない場合
　① 氏名を入力する
　② 住所を入力する
　　:
3. 価格を表示する
4. 支払金を入力する
　　:

実現したい処理をそのまま記述する

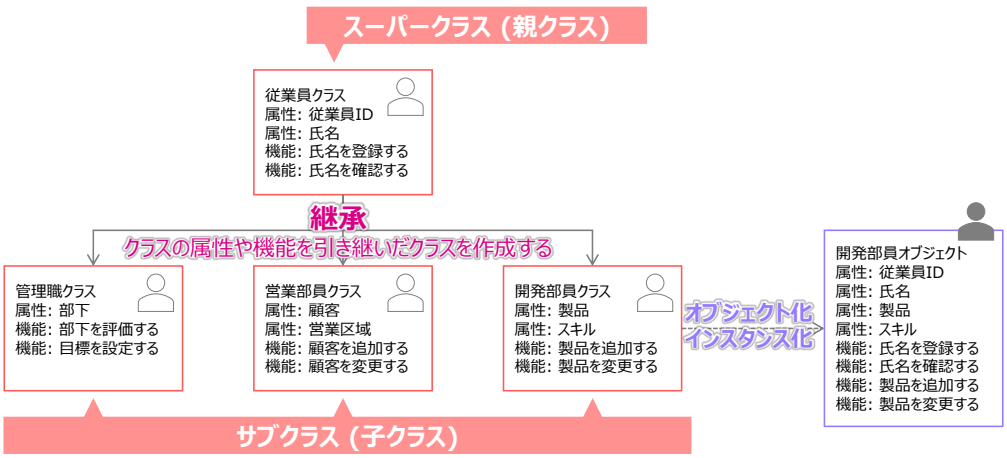
© 2022 FUJITSU LIMITED

オブジェクト指向における用語（Javaの場合）



© 2022 FUJITSU LIMITED

オブジェクト指向における用語（Java）



© 2022 FUJITSU LIMITED

会員クラスの実装イメージ（簡易サンプル）

FUJITSU

```
class Member {
  int id;
  String name;
  String address;

  void setName(String n) {
    this.name = n;
  }

  String getAddress() {
    return address;
  }
}
```

※赤太字はJavaとして決められている単語（予約語）

これは会員（Member）クラスです。
数値型の会員番号（id）フィールドを持ちます。
文字列型の氏名（name）フィールドを持ちます。
文字列型の住所（address）フィールドを持ちます。

会員
属性：会員番号
属性：氏名
属性：住所
機能：氏名を確認する
機能：住所を確認する

何も返却しないsetNameという機能を持ちます。
この機能は、引数として文字列nを受け取り、nameフィールドに代入します。

文字列を返却するgetNameという機能を持ちます。
※nameフィールドの値を返却します。

※Memberクラスを呼び出す側のイメージ

```
Member m = new Member();
m.setName("富士通太郎");
```

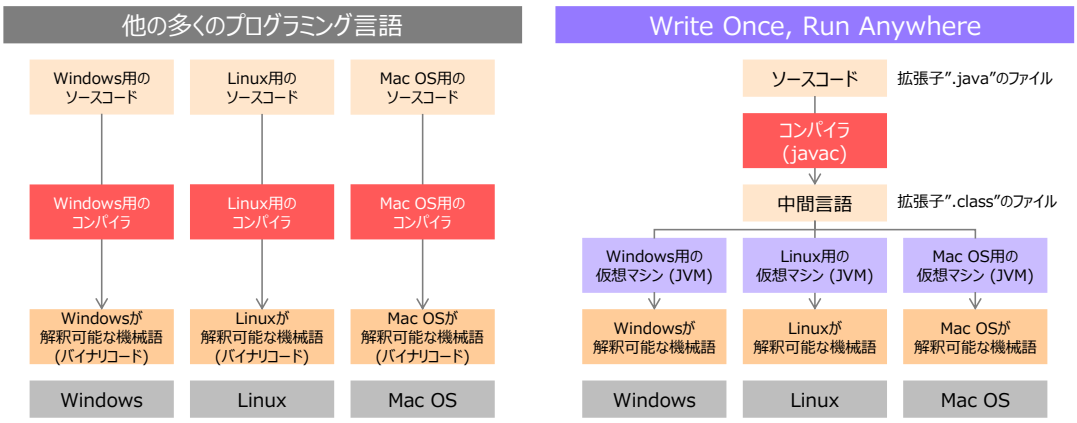
Memberクラスからmオブジェクトを作ります。
mオブジェクトのsetName機能に引数"富士通太郎"を渡して呼び出します。

© 2022 FUJITSU LIMITED

Java はプラットフォーム非依存という特徴を持つ

FUJITSU

【基本思想】Write Once, Run Anywhere（一度書けば、どこでも実行できる）



© 2022 FUJITSU LIMITED

Java実行環境/開発環境の配布形態

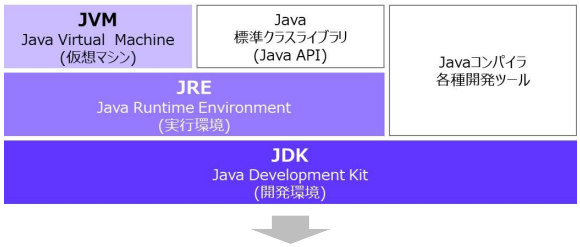
FUJITSU

○JRE (実行環境) = JVM + Java標準クラスライブラリ (Java API)
○JDK (開発環境) = JRE + Javaコンパイラなど各種開発ツール

© 2022 FUJITSU LIMITED

Javaの種類 (エディション)

FUJITSU



Java SE (Java Platform, Standard Edition)	一般ユーザー向け。
Java EE (Java Platform, Enterprise Edition)	企業ユーザー向け。業務システム用の拡張クラスライブラリなどを追加。
Java ME (Java Platform, Micro Edition)	組み込みシステム向け。専用クラスライブラリの他、JVMも軽量化。
Java Card	スマートカードなど小型メモリデバイス向け。

© 2022 FUJITSU LIMITED

java.lang.Mathの一部を読んでみる

FUJITSU

package/import
→Javaのパッケージ管理
package=自分の場所
import=利用する他の
クラスの場所

public/private
→クラス、フィールド、
メソッドの公開/非公開

```
package java.lang;

import java.math.BigDecimal;
import java.util.Random;
import jdk.internal.math.FloatConsts;
import jdk.internal.math.DoubleConsts;
import jdk.internal.vm.annotation.IntrinsicCandidate;

public final class Math {
  private Math() {}

  public static final double E = 2.718281828459045;
  public static final double PI = 3.141592653589793;
        :

  public static int max(int a, int b) {
    return (a >= b) ? a : b;
  }

  public static long max(long a, long b) {
    return (a >= b) ? a : b;
  }
        :
}
```

final
→変更できない事を表す

static
→オブジェクト化しなくても
利用可能である事を表す

© 2022 FUJITSU LIMITED

Javaにおける基本データ型

FUJITSU

データ型	意味	値
boolean	真偽値	true または false
char	16ビットUnicode文字	¥u0000～¥uFFFF
byte	8ビット整数	-128～127
short	16ビット整数	-32,768～32,767
int	32ビット整数	-2,147,483,648～2,147,483,647
long	64ビット整数	-9,223,372,036,854,775,808～ 9,223,372,036,854,775,807
float	32ビット単精度浮動小数点数	
double	64ビット単精度浮動小数点数	

© 2022 FUJITSU LIMITED

package/import

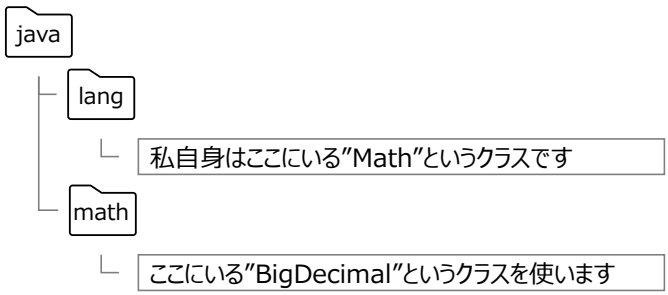
FUJITSU

Javaはクラスを**フォルダの様な階層構造で管理**
package [自分の場所]
import [利用したい他のクラスの場所].[利用したい他のクラス]

```
package java.lang;

import java.math.BigDecimal;
          :

public final class Math {
          :
}
```



© 2022 FUJITSU LIMITED

public/private

FUJITSU

クラスやフィールド、メソッドが**どこからアクセス可能であるか**を定義したもの。
「アクセス修飾子」と呼ばれる。

アクセス修飾子	意味
public	すべてのクラスからアクセスできる。
protected	現在のクラスとサブクラスからアクセスできる。
(省略)	現在のクラスト同じパッケージのクラスからアクセスできる。
private	現在のクラスからのみアクセスできる。 (他のクラスからアクセスできない。)

© 2022 FUJITSU LIMITED

final

FUJITSU

クラスやフィールド、メソッド、また、メソッドへの引数が変更不可である事を表す。
(それぞれ変更不可の意味は下表の通り。)

対象	意味
クラス	このクラスは継承できない。
メソッド	このメソッドはオーバーライドできない。
フィールド	定数。(final修飾子が付かない場合は値を自由に変えられる変数)
引数	メソッドの中で引数として渡された値を変更できない。

© 2022 FUJITSU LIMITED

static

FUJITSU

	static修飾子なし	static修飾子あり
フィールド	インスタンス変数	static変数
メソッド	インスタンスメソッド	staticメソッド

オブジェクト化/インスタンス化しなくても利用できる

```
public class Value {
  public String getValue() {
    :
  }
}
```

このメソッドを呼び出す為にはまずオブジェクト化が必要

```
Value v = new Value();
v.getValue();
```

```
public class Value {
  public static String getValue() {
    :
  }
}
```

オブジェクト化せずにメソッドの呼び出しが可能

```
Value.getValue();
```


© 2022 FUJITSU LIMITED

改めてjava.lang.Mathの一部を読んでみる

FUJITSU

```
package java.lang;

import java.math.BigDecimal;
import java.util.Random;
import jdk.internal.math.FloatConsts;
import jdk.internal.math.DoubleConsts;
import jdk.internal.vm.annotation.IntrinsicCandidate;

public final class Math {
  private Math() {}

  public static final double E = 2.718281828459045;
  public static final double PI = 3.141592653589793;
         :

  public static int max(int a, int b) {
    return (a >= b) ? a : b;
  }

  public static long max(long a, long b) {
    return (a >= b) ? a : b;
  }
         :
}
```

私は、java¥langにいます。

java¥mathにあるBigDecimalクラスを使います。
java¥utilにあるRandomクラスを使います。
jdk¥internal¥mathにあるFloatConstsクラスを使います。
jdk¥internal¥mathにあるDoubleConstsクラスを使います。
jdk¥internal¥vm¥annotationにあるIntrunsicCandidateクラスを使います。

私は、公開している変更（継承）できないMathというクラスです。
オブジェクト化する事はできません。(オブジェクト化する時に呼び出されるメソッドを隠蔽)

公開している64ビット単精度浮動小数点数の定数Eがあり、その値は2.7…です。
公開している64ビット単精度浮動小数点数の定数PIがあり、その値は3.1…です。

公開しているmaxという静的メソッドがあり、32ビット整数のaとbを受け取って、32ビット整数を返します。(aとbの大きさを比べて、大きい方の値を返します。)

公開しているmaxという静的メソッドがあり、64ビット整数のaとbを受け取って、32ビット整数を返します。(aとbの大きさを比べて、大きい方の値を返します。)

© 2022 FUJITSU LIMITED

FUJITSU

Java Specification Request とは

© 2022 FUJITSU LIMITED

JSR (Java Specification Request)

FUJITSU

- Javaに関連する技術は (Java自信を含めて)
  JSR (Java Specification Request) という名前で仕様を定めている。
  - この仕様は、JCP (Java Community Process) という団体で策定する。
  - JCPでは、Executive Committeeにおいて意思決定を行う。
    現在のメンバーは、Oracleの他、Amazon、富士通、IBM、Intel、Microsoftなど。
- 例えば、Java SE 19は”JSR 394”において仕様定義されている。
  - なお、Copyrightは、Oracle and/or its affiliates になっている。

© 2022 FUJITSU LIMITED

【参考】Java SEのバージョンと準拠しているJSR

FUJITSU

Javaのバージョン	リリース日	準拠している仕様 (JSR)
JDK 1.0	1996年1月23日	–
JDK 1.1	1997年2月19日	–
J2SE 1.2	1998年12月8日	–
J2SE 1.3	2000年5月8日	JSR 58
J2SE 1.4	2002年2月6日	JSR 59
J2SE 5.0	2004年9月30日	JSR 176
Java SE 6	2006年12月11日	JSR 270
Java SE 7	2011年7月28日	JSR 336
Java SE 8	2014年3月18日	JSR 337
Java SE 9	2017年9月21日	JSR 379
Java SE 10	2018年3月20日	JSR 383
Java SE 11	2018年9月25日	JSR 384
Java SE 12	2019年3月19日	JSR 386
Java SE 13	2019年9月17日	JSR 388
Java SE 14	2020年3月17日	JSR 389
Java SE 15	2020年9月15日	JSR 390
Java SE 16	2021年3月16日	JSR 391
Java SE 17	2021年9月14日	JSR 392
Java SE 18	2022年3月22日	JSR 393
Java SE 19	2022年9月20日	JSR 394

© 2022 FUJITSU LIMITED

JSR 394におけるjava.lang.Mathの仕様定義

FUJITSU

【JSR 394の定義抜粋】

```
package java.lang;

import java.math.BigDecimal;
import java.util.Random;
import jdk.internal.math.FloatConsts;
import jdk.internal.math.DoubleConsts;
import jdk.internal.vm.annotation.IntrinsicCandidate;

public final class Math {
  private Math() {}

  public static final double E = 2.718281828459045;
  public static final double PI = 3.141592653589793;
         :

  public static int max(int a, int b) {
    return (a >= b) ? a : b;
  }

  public static long max(long a, long b) {
    return (a >= b) ? a : b;
  }
         :
}
```

Package java.lang

public final class Math

public static final double E
The double value that is closer than any other to e, the base of the natural logarithms.
See Also: Constant Field Values

定数が定義されたページへのリンク

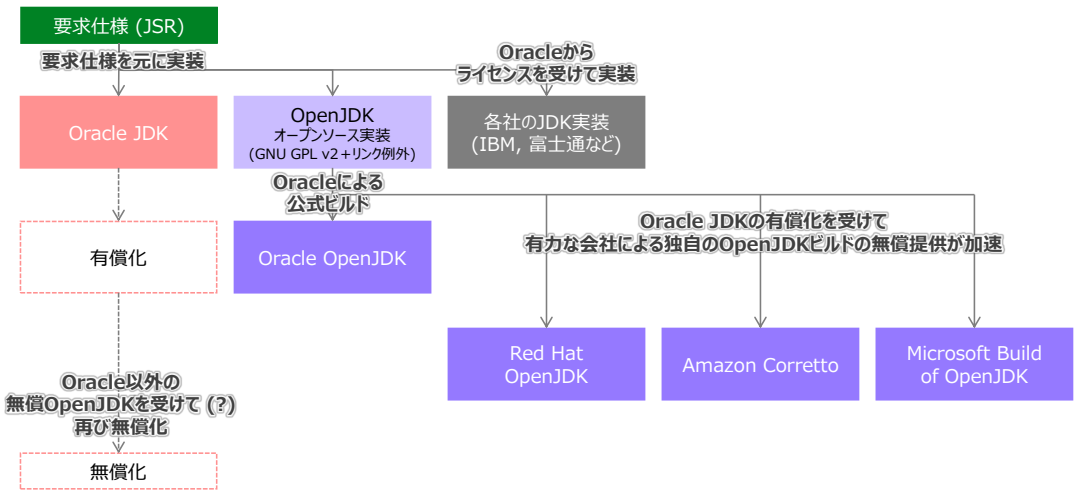
2.718281828459045

public static int max(int a, int b)
Returns the greater of two int values. That is, the result is the argument closer to the value of Integer.MAX_VALUE. If the arguments have the same value, the result is that same value.
Parameters: a - an argument. b - another argument.
Returns: the larger of a and b.

© 2022 FUJITSU LIMITED

Java実行環境/開発環境のバリエーション

FUJITSU



© 2022 FUJITSU LIMITED

License (JSR 394) [1/4]

FUJITSU

JSR-000394 Java SE 19 Final Release for Evaluation

ORACLE AMERICA, INC. IS WILLING TO LICENSE THIS SPECIFICATION TO YOU ONLY UPON THE CONDITION THAT YOU ACCEPT ALL OF THE TERMS CONTAINED IN THIS LICENSE AGREEMENT ("AGREEMENT"). PLEASE READ THE TERMS AND CONDITIONS OF THIS AGREEMENT CAREFULLY. BY DOWNLOADING THIS SPECIFICATION, YOU ACCEPT THE TERMS AND CONDITIONS OF THIS AGREEMENT. IF YOU ARE NOT WILLING TO BE BOUND BY THEM, SELECT THE "DECLINE" BUTTON AT THE BOTTOM OF THIS PAGE.

Specification: JSR-000394 Java SE 19 ("Specification")
Version: 19
Status: Final Release
Release: September 2022

Copyright 2022 Oracle America, Inc.
500 Oracle Parkway, Redwood City, California 94065, U.S.A.

All rights reserved.

NOTICE

The Specification is protected by copyright and the information described therein may be protected by one or more U.S. patents, foreign patents, or pending applications. Except as provided under the following license, no part of the Specification may be reproduced in any form by any means without the prior written authorization of Oracle America, Inc. ("Oracle") and its licensors, if any. Any use of the Specification and the information described therein will be governed by the terms and conditions of this Agreement.

Subject to the terms and conditions of this license, including your compliance with Paragraphs 1 and 2 below, Oracle hereby grants you a fully-paid, non-exclusive, non-transferable, limited license (without the right to sublicense) under Oracle's intellectual property rights to:

© 2022 FUJITSU LIMITED

License (JSR 394) [2/4]

FUJITSU

1.Review the Specification for the purposes of evaluation. This includes: (i) developing implementations of the Specification for your internal, non-commercial use; (ii) discussing the Specification with any third party; and (iii) excerpting brief portions of the Specification in oral or written communications which discuss the Specification provided that such excerpts do not in the aggregate constitute a significant portion of the Technology.

2.Distribute implementations of the Specification to third parties for their testing and evaluation use, provided that any such implementation:
(i) does not modify, subset, superset or otherwise extend the Licensor Name Space, or include any public or protected packages, classes, Java interfaces, fields or methods within the Licensor Name Space other than those required/authorized by the Specification or Specifications being implemented;
(ii) is clearly and prominently marked with the word "UNTESTED" or "EARLY ACCESS" or "INCOMPATIBLE" or "UNSTABLE" or "BETA" in any list of available builds and in proximity to every link initiating its download, where the list or link is under Licensee's control; and
(iii) includes the following notice:
"This is an implementation of an early-draft specification developed under the Java Community Process (JCP) and is made available for testing and evaluation purposes only. The code is not compatible with any specification of the JCP."

The grant set forth above concerning your distribution of implementations of the specification is contingent upon your agreement to terminate development and distribution of your "early draft" implementation as soon as feasible following final completion of the specification. If you fail to do so, the foregoing grant shall be considered null and void.

No provision of this Agreement shall be understood to restrict your ability to make and distribute to third parties applications written to the Specification.

Other than this limited license, you acquire no right, title or interest in or to the Specification or any other Oracle intellectual property, and the Specification may only be used in accordance with the license terms set forth herein. This license will expire on the earlier of: (a) two (2) years from the date of Release listed above; (b) the date on which the final version of the Specification is publicly released; or (c) the date on which the Java Specification Request (JSR) to which the Specification corresponds is withdrawn. In addition, this license will terminate immediately without notice from Oracle if you fail to comply with any provision of this license. Upon termination, you must cease use of or destroy the Specification.

"Licensor Name Space" means the public class or interface declarations whose names begin with "java", "javax", "com.oracle" or their equivalents in any subsequent naming convention adopted by Oracle through the Java Community Process, or any recognized successors or replacements thereof

© 2022 FUJITSU LIMITED

License (JSR 394) [3/4]

FUJITSU

TRADEMARKS

No right, title, or interest in or to any trademarks, service marks, or trade names of Oracle or Oracle's licensors is granted hereunder. Oracle, the Oracle logo, and Java are trademarks or registered trademarks of Oracle America, Inc. in the U.S. and other countries.

DISCLAIMER OF WARRANTIES

THE SPECIFICATION IS PROVIDED "AS IS" AND IS EXPERIMENTAL AND MAY CONTAIN DEFECTS OR DEFICIENCIES WHICH CANNOT OR WILL NOT BE CORRECTED BY ORACLE. ORACLE MAKES NO REPRESENTATIONS OR WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT THAT THE CONTENTS OF THE SPECIFICATION ARE SUITABLE FOR ANY PURPOSE OR THAT ANY PRACTICE OR IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADE SECRETS OR OTHER RIGHTS. This document does not represent any commitment to release or implement any portion of the Specification in any product.

THE SPECIFICATION COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION THEREIN; THESE CHANGES WILL BE INCORPORATED INTO NEW VERSIONS OF THE SPECIFICATION, IF ANY. ORACLE MAY MAKE IMPROVEMENTS AND/OR CHANGES TO THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THE SPECIFICATION AT ANY TIME. Any use of such changes in the Specification will be governed by the then-current license for the applicable version of the Specification.

LIMITATION OF LIABILITY

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ORACLE OR ITS LICENSORS BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION, LOST REVENUE, PROFITS OR DATA, OR FOR SPECIAL, INDIRECT, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF OR RELATED TO ANY FURNISHING, PRACTICING, MODIFYING OR ANY USE OF THE SPECIFICATION, EVEN IF ORACLE AND/OR ITS LICENSORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

© 2022 FUJITSU LIMITED

License (JSR 394) [4/4]

FUJITSU

You will hold Oracle (and its licensors) harmless from any claims based on your use of the Specification for any purposes other than the limited right of evaluation as described above, and from any claims that later versions or releases of any Specification furnished to you are incompatible with the Specification provided to you under this license.

RESTRICTED RIGHTS LEGEND

If this Software is being acquired by or on behalf of the U.S. Government or by a U.S. Government prime contractor or subcontractor (at any tier), then the Government's rights in the Software and accompanying documentation shall be only as set forth in this license; this is in accordance with 48 C.F.R. 227.7201 through 227.7202-4 (for Department of Defense (DoD) acquisitions) and with 48 C.F.R. 2.101 and 12.212 (for non-DoD acquisitions).

REPORT

You may wish to report any ambiguities, inconsistencies or inaccuracies you may find in connection with your evaluation of the Specification ("Feedback"). To the extent that you provide Oracle with any Feedback, you hereby: (i) agree that such Feedback is provided on a non-proprietary and non- confidential basis, and (ii) grant Oracle a perpetual, non-exclusive, worldwide, fully paid-up, irrevocable license, with the right to sublicense through multiple levels of sublicensees, to incorporate, disclose, and use without limitation the Feedback for any purpose related to the Specification and future versions, implementations, and test suites thereof.

GENERAL TERMS

Any action related to this Agreement will be governed by California law and controlling U.S. federal law. The U.N. Convention for the International Sale of Goods and the choice of law rules of any jurisdiction will not apply.

The Specification is subject to U.S. export control laws and may be subject to export or import regulations in other countries. Licensee agrees to comply strictly with all such laws and regulations and acknowledges that it has the responsibility to obtain such licenses to export, re-export or import as may be required after delivery to Licensee.

This Agreement is the parties' entire agreement relating to its subject matter. It supersedes all prior or contemporaneous oral or written communications, proposals, conditions, representations and warranties and prevails over any conflicting or additional terms of any quote, order, acknowledgment, or other communication between the parties relating to its subject matter during the term of this Agreement. No modification to this Agreement will be binding, unless in writing and signed by an authorized representative of each party.

© 2022 FUJITSU LIMITED

Java API とは

FUJITSU

© 2022 FUJITSU LIMITED

API とは

FUJITSU

Application Programing Interface

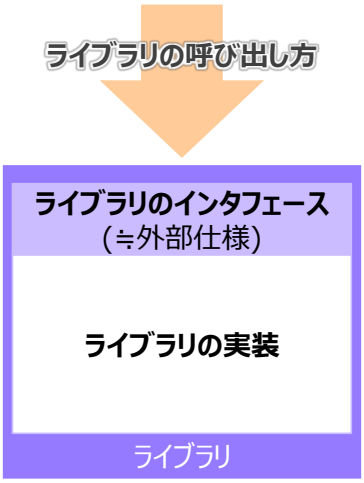
- ソフトウェアコンポーネント (プログラム、サービスなど) 同士が互いに情報をやり取りする為に使用するインターフェースの仕様のこと。
- 関数/サブルーチン/メソッドの呼び出し方、インプットする変数 (引数)、アウトプットされる変数 (戻り値)、データ構造などが定義される。

国際標準化されている規約・プロトコル	POSIX など
ベンダーによる定義	Windows API など
プログラミング言語の標準ライブラリ	Standard Template Library (C++) Java API

© 2022 FUJITSU LIMITED

JavaにおけるAPIとは

FUJITSU



一般的にはここをAPIと呼ぶことも多いが、
Javaの場合は開発者がインタフェースを読み解く必要あり。

この部分は特に”Java API仕様”と呼ばれる。
インタフェースを実装するコードに近い定義になっている。

一般的に、Java APIと言うとここを指している事が多い。
(厳密な定義ではない。)

© 2022 FUJITSU LIMITED

API仕様の位置付けの違い (開発者？利用者？)

FUJITSU

`https://www.googleapis.com/upload/drive/v3/files?uploadType=media&Content-Type=...`	利用者	・java.langにあるのでimportの必要はない。 ・staticメソッドなのでインスタンス化の必要はない。 ・int型の変数を2つ渡すと、int型の戻り値がもらえる。 `int num = Math.max(100, 1000);`
ほぼそのまま実装可能		読み解きが必要
Create a POST request to the method's /upload URI with the query parameter of uploadType=media: **https://www.googleapis.com/upload/drive/v3/files?uploadType=media** Add the file's data to the request body. Send the request. If the request succeeds, the server returns the HTTP 200 OK status code along with the file's metadata. {HTTP}	API仕様	**Package java.lang** **public final class Math** **public static int max(int a, int b)** Returns the greater of two int values. That is, the result is the argument closer to the value of Integer.MAX_VALUE. If the arguments have the same value, the result is that same value. Parameters: a - an argument. b - another argument. Returns: the larger of a and b.
読み解きが必要		ほぼそのまま実装可能

・HTTP/HTTPSプロトコルの受け口が必要。
・HTTP/HTTPSプロトコルの解釈器が必要。
・パラメータごとに該当する動作の実装が必要。
	ライブラリの開発者	`package java.lang;`
		
		`public final class Math {`
		`  public static int max(int a, int b) {}`
		`}`

© 2022 FUJITSU LIMITED

Java API Specification

FUJITSU

JavaのAPI Specification (仕様) には以下の様に記されている。

○本書は2つのセクションに分かれています。

- ○Java Platform, Standard Edition (Java SE) APIは、汎用コンピューティングのためのJavaプラットフォームを定義します。
  これらのAPIは、名前がjavaで始まるモジュール内にあります。
- ○Java Development Kit（JDK）APIはJDK固有のものであり、必ずしもJava SEプラットフォームのすべての実装で利用できるとは限りません。
  これらの API は、jdkで始まる名前のモジュール内にあります。

【参考】https://www.oracle.com/jp/java/technologies/documentation.html
【参考】https://docs.oracle.com/en/java/javase/19/

© 2022 FUJITSU LIMITED

Java SEに含まれるAPI (クラスライブラリ) (1/3)

FUJITSU

モジュール	説明
java.base	**Java SE Platformの基盤となるAPIを定義します。**
java.compiler	言語モデル、注釈処理、およびJavaコンパイラAPIを定義します。
java.datatransfer	アプリケーション間およびアプリケーション内でデータを転送するためのAPIを定義します。
java.desktop	AWTとSwingのユーザー・インタフェース・ツール・キットとアクセシビリティ、オーディオ、イメージング、印刷、およびJavaBeans用のAPIを定義します。
java.instrument	エージェントがJVM上で実行されているプログラムを計測できるようにするサービスを定義します。
java.logging	Java Logging APIを定義します。
java.management	Java Management Extensions (JMX) APIを定義します。
java.management.rmi	Java Management Extensions (JMX)リモートAPIの「RMIコネクタ」を定義します。
java.naming	Java Naming and Directory Interface (JNDI) APIを定義します。
java.net.http	HTTPクライアントおよびWebSocket APIを定義します。
java.prefs	Preferences APIを定義します。
java.rmi	Remote Method Invocation (RMI) APIを定義します。
java.scripting	Scripting APIを定義します。
java.se	Java SE PlatformのAPIを定義します。
java.security.jgss	IETF Generic Security Services API (GSS-API)のJavaバインディングを定義します。
java.security.sasl	IETF Simple Authentication and Security Layer (SASL)のJavaサポートを定義します。
java.smartcardio	Java Smart Card I/O APIを定義します。
java.sql	JDBC APIを定義します。
java.sql.rowset	JDBC RowSet APIを定義します。
java.transaction.xa	JDBCで分散トランザクションをサポートするためのAPIを定義します。
java.xml	Java API for XML Processing (JAXP)、Streaming API for XML (StAX)、Simple API for XML (SAX)、およびW3C Document Object Model (DOM) APIを定義します。
java.xml.crypto	XML暗号化のためのAPIを定義します。

© 2022 FUJITSU LIMITED

Java SEに含まれるAPI (クラスライブラリ) (2/3)

FUJITSU

モジュール	説明
jdk.accessibility	Assistive Technologiesの実装者が使用するJDKユーティリティ・クラスを定義します。
jdk.attach	アタッチAPIを定義します。
jdk.charsets	java.base (主に2バイト文字とIBM文字セット)にないcharsetsを提供します。
jdk.compiler	「システムJavaコンパイラ」の実装とそのコマンドラインの等価javacを定義します。
jdk.crypto.cryptoki	SunPKCS11セキュリティ・プロバイダの実装を提供します。
jdk.crypto.ec	SunECセキュリティ・プロバイダの実装を提供します。
jdk.dynalink	高水準の操作をオブジェクトに動的にリンクするためのAPIを定義します。
jdk.editpad	jdk.jshellによって使用される編集パッド・サービスの実装を提供します。
jdk.hotspot.agent	HotSpot Serviceability Agentの実装を定義します。
jdk.httpserver	JDK固有のHTTPサーバーAPIを定義し、最小限のHTTPサーバーを実行するためのjwebserverツールを提供します。
jdk.incubator.foreign	外部メモリーにアクセスし、Javaから直接外部ファンクションをコールするためのAPIを定義します。
jdk.incubator.vector	実行時にSIMD命令(x64でのAVX命令やAArch64でのNEON命令など)に確実にコンパイルできる計算を表現するためのAPIを定義します。
jdk.jartool	jarツールやjarsignerツールなど、Javaアーカイブ(JAR)ファイルを操作するためのツールを定義します。
jdk.javadoc	「システム・ドキュメンテーション・ツール」とそれと同等のコマンドラインjavadocの実装を定義します。
jdk.jcmd	診断ツールを定義し、jcmd、jps、jstatツールなどのJVMをトラブルシューティングします。
jdk.jconsole	実行中のアプリケーションをモニタリングおよび管理するための、JMXグラフィカル・ツールjconsoleを定義します。
jdk.jdeps	jdeps、javapおよびjdeprscanツールを含む、Javaライブラリおよびプログラムの依存性を分析するためのツールを定義します。
jdk.jdi	Java Debug Interfaceを定義します。
jdk.jdwp.agent	Java Debug Wire Protocol (JDWP)エージェントの実装を提供します。
jdk.jfr	JDK Flight RecorderのAPIを定義します。
jdk.jlink	実行時イメージの作成用のjlinkツール、JMODファイルの作成と操作用のjmodツール、およびクラスとリソース用のJDK実装固有のコンテナ・ファイルの検査用にjimageツールを定義します。
jdk.jpackage	Javaパッケージ化ツールのjpackageを定義します。
jdk.jshell	Javaコードのスニペットを評価するためのjshellツールを提供し、スニペットをモデリングおよび実行するためのJDK固有のAPIを定義します。

© 2022 FUJITSU LIMITED

Java SEに含まれるAPI (クラスライブラリ) (3/3)

FUJITSU

モジュール	説明
jdk.jsobject	JavaScriptオブジェクトのAPIを定義します。
jdk.jstatd	jstatツールのデーモンを起動してJVM統計をリモートで監視するためのjstatdツールを定義します。
jdk.localedata	「米国のロケール」以外のロケールのロケール・データを提供します。
jdk.management	JVMのJDK固有の管理インタフェースを定義します。
jdk.management.agent	JMX管理エージェントを定義します。
jdk.management.jfr	JDK Flight Recorderの管理インタフェースを定義します。
jdk.naming.dns	DNS Java Namingプロバイダの実装を提供します。
jdk.naming.rmi	RMI Java Namingプロバイダの実装を提供します。
jdk.net	JDK固有のNetworking APIを定義します。
jdk.nio.mapmode	JDK固有のファイル・マッピング・モードを定義します。
jdk.sctp	SCTP用のJDK固有のAPIを定義します。
jdk.security.auth	javax.security.auth.*インタフェースとさまざまな認証モジュールの実装を提供します。
jdk.security.jgss	JDK拡張機能をGSS-APIに定義し、SASL GSSAPIメカニズムを実装します。
jdk.xml.dom	Java SE APIの一部ではないW3C Document Object Model (DOM) APIのサブセットを定義します。
jdk.zipfs	Zipファイル・システム・プロバイダの実装を提供します。

© 2022 FUJITSU LIMITED

java.baseに含まれるパッケージ (1/3)

FUJITSU

パッケージ	説明
java.io	このパッケージは、データ・ストリーム、直列化、ファイル・システムによるシステム入出力用に提供されています。
java.lang	Javaプログラミング言語の設計にあたり基本的なクラスを提供します。★このパッケージはimportしなくても利用できます。
java.lang.annotation	Javaプログラミング言語の注釈機能のライブラリ・サポートを提供します。
java.lang.constant	クラスやメソッド・ハンドルなどのランタイム・エンティティ用に「名目記述子」を表すクラスおよびインタフェース、および定数プール・エントリやinvokedynamicコール・サイトなどのクラス・ファイル・エンティティ用に「名目記述子」を表すクラスおよびインタフェース。
java.lang.invoke	java.lang.invokeパッケージは、Java Virtual Machineとやりとりするための低レベルのプリミティブを提供します。
java.lang.module	モジュール記述子をサポートし、解決とサービス・バインディングによってモジュールの構成を作成するクラス。
java.lang.ref	参照オブジェクト・クラスを提供し、限定されたレベルでのガベージ・コレクタとの対話を可能にします。
java.lang.reflect	クラスとオブジェクトに関するリフレクト情報を取得するための、クラスとインタフェースを提供します。
java.lang.runtime	java.lang.runtimeパッケージでは、Java言語の低レベルのランタイム・サポートが提供されます。
java.math	任意精度の整数演算(BigInteger)および任意精度の10進演算(BigDecimal)を行うクラスを提供します。
java.net	ネットワーク・アプリケーションを実装するためのクラスを提供します。
java.net.spi	java.netパッケージのサービス・プロバイダ・クラス。
java.nio	データのコンテナであるバッファについて定義し、その他のNIOパッケージの概要情報を提供します。
java.nio.channels	入出力操作を実行できるエンティティ(ファイル、ソケットなど)への接続を表すチャネルや、多重化された非ブロック入出力操作用のセレクタを定義します。
java.nio.channels.spi	java.nio.channelsパッケージのサービス・プロバイダ・クラス。
java.nio.charset	byteとUnicode文字の相互変換を行うため、文字セット、デコーダ、およびエンコーダを定義します。
java.nio.charset.spi	java.nio.charsetパッケージのサービス・プロバイダ・クラス。
java.nio.file	ファイル、ファイル属性、およびファイル・システムにアクセスするためのJava仮想マシン用のインタフェースとクラスを定義します。
java.nio.file.attribute	ファイルおよびファイル・システム属性へのアクセスを提供するインタフェースとクラスです。
java.nio.file.spi	java.nio.fileパッケージのサービス・プロバイダ・クラス。
java.security	セキュリティ・フレームワークのクラスとインタフェースを提供します。
java.security.cert	証明書、証明書失効リスト(CRL)、証明書パスを解析および管理するためのクラスとインタフェースを提供します。

© 2022 FUJITSU LIMITED

java.baseに含まれるパッケージ (2/3)

FUJITSU

パッケージ	説明
java.security.interfaces	RSA Laboratory Technical Note PKCS#1で定義されているRSA (Rivest, Shamir and Adleman AsymmetricCipher algorithm)キーと、NISTのFIPS-186で定義されているDSA (Digital Signature Algorithm)キーを生成するためのインタフェースを提供します。
java.security.spec	キー仕様およびアルゴリズム・パラメータ仕様のクラスおよびインタフェースを提供します。
java.text	テキスト、日付、数値、およびメッセージを自然言語に依存しない方法で処理するためのクラスとインタフェースを提供します。
java.text.spi	java.textパッケージに含まれているクラスのサービス・プロバイダ・クラスです。
java.time	日付、時間、インスタント、デュレーションのメインAPI。
java.time.chrono	デフォルトのISO暦以外の暦体系の汎用API。
java.time.format	日付と時間を出力し、解析するクラスを提供します。
java.time.temporal	フィールドと単位を使用した日時へのアクセス、および日時アジャスタ。
java.time.zone	タイムゾーンおよびそのルールのサポート。
java.util	コレクション・フレームワーク、国際化サポート・クラス、サービス・ローダー、プロパティ、乱数生成、文字列解析とスキャン・クラス、Base64エンコーディングとデコード、ビット配列、およびその他のユーティリティ・クラスが含まれています。
java.util.concurrent	並行プログラミングでよく使用されるユーティリティ・クラスです。
java.util.concurrent.atomic	単一の変数に対するロックフリーでスレッドセーフなプログラミングをサポートするクラスの小規模なツールキットです。
java.util.concurrent.locks	組込みの同期および監視から区別された状態をロックおよび待機するためのフレームワークを提供するインタフェースおよびクラス。
java.util.function	関数型インタフェースは、ラムダ式やメソッド参照のターゲットとなる型を提供します。
java.util.jar	JAR (Java ARchive)ファイル形式の読み込みと書込みに使うクラスを提供します。JARは、必要に応じてマニフェスト・ファイルを付随させることのできる、標準的なZIPに基づくファイル形式です。
java.util.random	このパッケージには、乱数生成のための汎用APIをサポートするクラスとインタフェースが含まれています。
java.util.regex	正規表現で指定されたパターンに対して文字シーケンスをマッチングするためのクラス。
java.util.spi	java.utilパッケージに含まれているクラスのサービス・プロバイダ・クラスです。
java.util.stream	コレクションに対するマップ-リデュース変換など、要素のストリームに対する関数型の操作をサポートするクラスです。
java.util.zip	標準のZIPおよびGZIPファイル形式の読み込みおよび書込み用クラスを提供します。
javax.crypto	暗号化操作のクラスとインタフェースを提供します。

© 2022 FUJITSU LIMITED

java.baseに含まれるパッケージ (3/3)

FUJITSU

パッケージ	説明
javax.crypto.interfaces	RSA LaboratoriesのPKCS#3で定義されているDiffie-Hellmanキーのインタフェースを提供します。
javax.crypto.spec	キー仕様およびアルゴリズム・パラメータ仕様のクラスおよびインタフェースを提供します。
javax.net	ネットワーク・アプリケーションのためのクラスを提供します。
javax.net.ssl	セキュア・ソケット・パッケージのクラスを提供します。
javax.security.auth	このパッケージは、認証と承認に関するフレームワークを提供します。
javax.security.auth.callback	このパッケージは、情報(ユーザー名やパスワードなどの認証データ)の取得や情報(エラーおよび警告メッセージなど)の表示のためにサービスがアプリケーションとやり取りするために必要なクラスを提供します。
javax.security.auth.login	このパッケージは、プラグイン可能な認証フレームワークを提供します。
javax.security.auth.spi	このパッケージは、プラグイン可能な認証モジュールの実装に使用されるインタフェースを提供します。
javax.security.auth.x500	このパッケージには、X500プリンシパルおよびX500非公開資格をSubjectに格納する際に使用すべきクラスが含まれています。
javax.security.cert	公開キー証明書用のクラスを提供します。

© 2022 FUJITSU LIMITED

Java標準クラスライブラリ (Java API) の位置付け 特に"java.lang"

FUJITSU

Javaプログラミング開発においては**必須**

java.lang.Object

■Javaでは、すべてのクラスのスーパークラスとして Objectクラスが用意されている。
■すべてのクラスはOcjectクラスを継承している為、様々なオブジェクトの操作や情報入手が可能。 (これらはObjectクラスに実装されている。)

```
System.out.println(obj.getClass());
```

objオブジェクトのクラスが何か調べて出力する

```
class java.lang.String
```

java.lang.Exception

■Javaでは、ライブラリなどの処理中に予期せぬ事象が発生した場合、例外 (Exceptionクラスの) オブジェクトが生成される。
■生成された例外オブジェクトの内容に従って エラー処理を実装する。

```
try {
  // 処理
}
catch (Exception e) {
  e.printStackTrace();
}
```

プログラミングする上でほぼ必須となる機能も標準クラスライブラリとして提供

© 2022 FUJITSU LIMITED

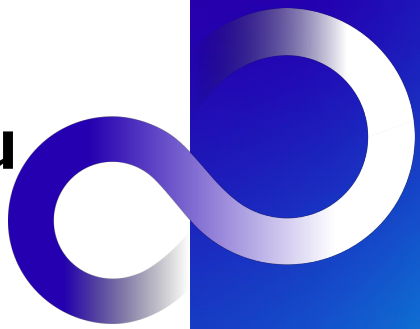
関連情報

FUJITSU

- Java Community Process (https://jcp.org/en/home/index)
- Java Specification Requests (https://jcp.org/en/jsr/overview)
- Oracle Java (https://www.oracle.com/jp/java/technologies/)
- Java SE 日本語ドキュメント (https://www.oracle.com/jp/java/technologies/documentation.html)
- OpenJDK (https://openjdk.org/)
- GitHub OpenJDK (https://github.com/openjdk)
- Android API Reference (https://developer.android.com/reference)

© 2022 FUJITSU LIMITED

Thank you



FUJITSU

© 2022 FUJITSU LIMITED

Google v. Oracle事件最高裁判決

2023年1月16日

慶應義塾大学大学院法務研究科教授
奥邨　弘司

© Koji OKUMURA

▸訴訟の経緯

1996年　SunがJavaを開発　（2010年OracleがSunを買収）

2005年　GoogleがAndroidを開発開始
　　Javaのライセンスを受けようとするが頓挫。JavaAPIの一部を利用してAndroidプラットフォームを開発

2010年　Oracleが著作権侵害及び特許権侵害でGoogleを提訴

2012年　陪審は特許権侵害否定、APIの著作物性と著作権侵害を認定
　　地裁は、陪審評決を覆し、APIの著作物性を否定
　　〔872 F. Supp. 2d 974〕

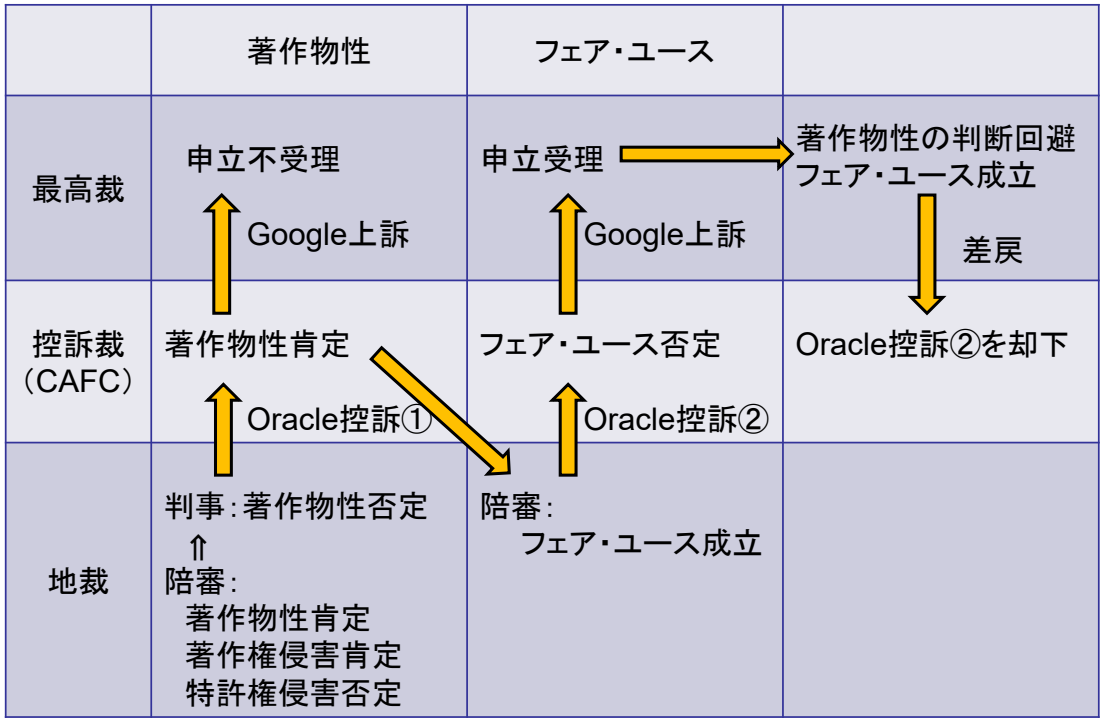
2014年　CAFCがAPIの著作物性を認めて、地裁判決を取り消し、差戻し
　　〔750 F.3d 1339〕　⇒　Googleの裁量上訴の申立却下

2016年　地裁はフェアユースを認める

2018年　CAFCがフェアユースを否定して、事件を差戻し
　　〔886 F.3d 1179〕 ⇒　Googleの裁量上訴の申立認容

2021年　Googleにフェア・ユース成立の最高裁判決
　　⇒　事件はCAFCに差戻し

▸訴訟経緯



【3】

■Google LLC v. Oracle Am., Inc., 141 S. Ct. 1183 (2021)　の概要

≪法廷意見≫　6人
API宣言コードの著作物性の判断は回避した上で、フェア・ユース成立と判断

ブライヤー判事・ソトマイヨール判事・ケイガン判事　・・・　リベラル派
ロバーツ長官　・・・　中間派
ゴーサッチ判事・カバノー判事　・・・　保守派

≪反対意見≫　2人
APIの宣言コードには著作物性があり、またその利用はフェア・ユースではない

トーマス判事・アリート判事　・・・　保守派

＊新任のバレット判事（保守派）は、就任前の事件であり、参加せず

【4】

■ フェア・ユース （米国著作権法107条）

106条および106A条の規定にかかわらず、批評、論評、ニュース報道、教育（教室での使用のための複数複製を含む）、研究、調査などの目的で、著作権のある著作物を公正に利用すること――複製物またはレコードの形での複製による利用、または該当条に特掲された他の方法による利用を含む――は、著作権の侵害とはならない。個々の事件における著作物の利用が公正な利用といえるか否かを決定する上で、考慮されるべき要素には以下のものが含まれる。

① 当該利用の目的および性格。なお、当該利用が商業的性質のものか、非営利的教育目的かといったことも含む
② 当該著作権のある著作物の性質
③ 当該著作権のある著作物全体との関係で利用される部分の量および実質性ならびに
④ 当該利用が、当該著作権のある著作物の潜在的な市場や価値に与える影響

上記要素のすべてを考慮した上で、公正な利用であるとされた場合、著作物が未発行であるという事実は、それ自体では、公正な利用であるとすることを妨げない。

■ フェア・ユース

フェア・ユースは、制定法ではなく判例法によって生み出された概念である

⇓

Folsom v. Marsh, 9. F.Cas. 342 (C.C.D. Mass. 1841)

・被告は、原告の著作物（初代大統領ワシントンの著作集）から、無断で転載・要約を行って被告書籍を完成させたため、著作権侵害に問われた

ストリー判事による判決

「要するに、本件のような問題を判断する際には、行われた行為の性質と目的、利用された既存著作物の量と価値、そして、その利用によって、原作品の販売を害したり、原作品の利益を減少させたり、原作品の目的に取って代わったりする程度に注目しなければならない。」

行われた行為の性質と目的	⇒	第1要素
利用された著作物の価値	⇒	第2要素
利用された著作物の量	⇒	第3要素
市場代替性・市場への害	⇒	第4要素

Folsom判決に始まる一連の判例法を、現行著作権法制定時に明文化したのが107条

＊ 厳密な意味では、Folsom事件判決はフェア・ユース（権利制限）の起源ではない。実は当時の著作権法には翻案権が存在しなかった。結果、被告の行為は著作権侵害ではなかった。ストーリー判事は、複製だけではなくて翻案にも権利が及ぶとしつつ、その範囲を画するために前記説示を行った。つまり権利拡張の説示だった。ただ、権利の範囲は裏返せば権利制限の範囲であり、実質的にフェアユースの起源と考えられるようになった。

■ 現行法制定後のフェア・ユースに関する最高裁判決のポイント

Sony v. Universal, 464 U.S. 417 (1984)

無料TV放送番組のタイムシフト録画をフェア・ユースと認めた

・商業的使用は不公正と推定。非商業的使用は逆の推定　＜第1要素＞
・潜在的市場への影響＝将来の損害の可能性。商業的な場合可能性は推定される
　非商業的な場合、著作権者が可能性を証明する必要がある　＜第4要素＞

Harper & Row v. Nation, 471 U.S. 539 (1985)

刊行前の大統領の手記の一部をスクープした報道についてフェア・ユースを否定

・未公表の著作物の利用がフェア・ユースとなる範囲は狭い　＜第2要素＞
　　⇒　後の法改正で当てはまらなくなる
・量は少なくとも核心部分を利用すれば実質的である　＜第3要素＞
・第4要素は、唯一の最も重要なフェア・ユースの構成要素である　＜第4要素＞
・市場への害の検討に際しては、二次的著作物の市場も考慮すべき　＜第4要素＞

Stewart v. Abend, 495 U.S. 207 (1990)

小説の映画化についてフェア・ユースを否定

・フェア・ユースは、法の厳格適用が創造性を窒息させることを避けるためのもの
・事実的な作品の場合は、創造的な作品の場合よりも、フェア・ユースは認められやすい　＜第2要素＞

【10】

Campbell v. Acuff-Rose, 510 U.S. 569 (1994)

ラップグループがリリースしたパロディ曲の歌詞について、フェア・ユース肯定

＜総論＞
・フェア・ユース判断に一律の明確な基準はなく、事例毎に、個別に4要素の検討結果を総合考慮して判断すること

＜第1要素＞
・検討の中心は、後続の作品が先行作品を代替するだけか、それとも変容力を有しているか、それはどの程度か　Sony判決の影響力減
・商業性の有無は検討事項の1つ。変容力があるほど、その重要性は小さくなる
・パロディ（原作品を批評・批判するもの）は、変容力を有する

＜第2要素＞　Stewart判決の影響力減
・パロディ（のような変容力のある作品）の場合、第2要素は重要ではない

＜第3要素＞　Harper & Row判決の影響力減
・許される複製の程度（量的・質的）は、利用の目的と性格によって変化する

＜第4要素＞　Sony判決の影響力減
・ソニー判決の推定は、単純複製の場合にのみ当てはまる
・後続の利用が変容力を有するほど、市場代替性・市場への影響は不明確になる
・辛辣な書評がもたらす悪影響は、著作権法上の評価の対象外
・二次的利用のための潜在的な市場には、原作品の権利者自身が一般的に活用する市場と、活用のために他者にライセンスする一般的な市場のみが含まれる

【14】

■　最高裁判例について

①　（著作権に関する）最高裁判例は必ずしも多くない

②　後続する最高裁判例は・・・
・事案に関係しないところは先行最高裁判例に触れない
・関係する部分で方向性が同じ場合は先行最高裁判例を引用する
・一方、衝突する場合は切り分けにより先行最高裁判例の影響力を削ぐ

③　結果、直近の判例の影響力が加重される
⇓
これまでは、Campbell事件最判の影響力が極めて強かった　：　今後は？
・Campbell事件（以降）においては、第１要素の検討＝変容力の有無が鍵だった
（変容力無双状態！）
・具体的には、変容力がある場合・・・
商業性や第２要素は重要性を失う。同時に、変容力のある利用方法のために必要なら、全部利用（量）や核心部分利用（質）も許される。市場代替性も低いとされて、（潜在的）市場への影響も否定されがち。

第１要素の検討の中心は、既存の著作物を代替するだけか、それとも、既存の著作物を新しい表現や意味、または主張によって変化させることで、さらなる目的や異なる性格を伴う、何か新しいものを付け加えるのかであり、言葉を借りれば、新しい作品が「変容力を有する」のか、それはどの程度かを問うことである。

【15】

■Google LLC v. Oracle Am., Inc., 141 S. Ct. 1183 (2021)　の概要

≪法廷意見≫　６人
API宣言コードの著作物性の判断は回避した上で、フェア・ユース成立と判断

ブライヤー判事・ソトマイヨール判事・ケイガン判事	・・・	リベラル派
ロバーツ長官	・・・	中間派
ゴーサッチ判事・カバノー判事	・・・	保守派

≪反対意見≫　２人
APIの宣言コードには著作物性があり、またその利用はフェア・ユースではない

トーマス判事・アリート判事	・・・	保守派

＊新任のバレット判事（保守派）は、就任前の事件であり、参加せず

【16】

■Google LLC v. Oracle Am., Inc., 141 S. Ct. 1183 (2021) の概要

≪総論≫
- 〔著作権法の意義、フェア・ユース規定の歴史と意義などについて簡単に述べる〕
- 争点は、①Java API（の宣言コードとその体系を含む）全体の著作物性と、②Googleによる（宣言コードとその体系の）利用がフェア・ユースとなるか。
- ①については、著作物性があることを仮定した上で、②のフェア・ユースの成立の可否について検討する。

≪コンピュータプログラムの特殊性とフェア・ユース≫
- コンピュータ・プログラムは、伝統的な言語の著作物とは異なり、機能的な目的を果たすためのものであるという点で特徴を有する。
- 議会は、コンピュータプログラムを著作物として保護することとした。
- 結果、コンピュータプログラムは、他の著作物同様に保護される一方、他と同様に著作権の制限（フェア・ユースを含む）に服すこととなった。
- フェア・ユースは、コンピュータプログラムが著作権で保護される範囲を画する上で重要な役割を果たす。

≪フェア・ユースは事実問題か法律問題か≫
- フェア・ユースは事実と法律の混合問題であり、事実問題の部分は陪審員に委ね、法律問題は裁判官が判断するべき。
- 法律審では、法律問題の部分について、（裁量権の濫用があるかどうかだけを判断するのではなくて）一から判断すべきである。

【17】

≪フェアユースの４要素の検討≫

＜第２要素＞ ⇒ フェア・ユースに有利
- コンピュータが実行する特定のタスクにどのようなラベルを付け、どのように体系化するかという点について著作物性が認められるかは議論がありうるだろう。
  〔この点は、本判決では議論されない〕
- 宣言コードは、コンピュータプログラムの一部であるが、タスクを分割するという一般的なシステム、タスクを体系化するというアイデア、コマンドの呼び出し方、実装コードと、密接に結びついているという点で特徴的である。
  〔宣言コード・実装コードと分けられているが、別々に存在するのではなくて、１つのプログラム中に、宣言部分・実装部分という具合に連続して存在する〕
- 実装コードの作成にあたっては、どうすれば効率とスピードを両立させてコンピュータを稼働させられるかなどの点に創造性を発揮する必要がある。
  一方、宣言コードの場合は、プログラマーが、宣言コードの名前を覚えやすくする点に創造性が発揮された。
- SunのJava APIは、ユーザーインターフェースであり、宣言コードは、ユーザーインターフェースの一部であるため、他のプログラムとは異なる。
- 宣言コードの価値は、プログラマーが時間と労力を費やしてAPIシステムを習得したことに由来する。
- 以上から、仮に著作権があるとしても、宣言コードは、（コンピュータプログラムの中でさえも）著作権保護の中核から遠い存在であるといえる。

【19】

＜第１要素＞　⇒　フェア・ユースに有利

- 第１要素の検討においては、従来から変容力のある利用か否かを重視してきた。
- Googleは、機能を呼び出すために宣言コードを複製したが、機能を呼び出すという目的自体は、プログラムの複製全てに共通するものなので、そこから先、より具体的な目的とその特徴について検討しなければならない。
- （より具体的に見ると）GoogleがJava APIを利用したのは、スマートフォン向けの創造的で核心的なプラットフォームを生み出すためであった。
- Googleは、再実装（既存のシステムを学習したプログラマーが、その基本的なスキルを新しいシステムで使用できるようにする）ために、必要な範囲でAPIを複製したに過ぎない。
- インターフェイスの再実装が、コンピュータプログラムの開発を促進することは、十分に陪審員に対して示されている。
- 以上は、Googleの複製の「目的と性格」に、変容力があることを示している。
- 非商業的な利用がフェア・ユース認定に有利であることは事実だが、Campbell事件最判で説示したように逆は必ずしも真ではない。変容力ある利用の場合はなおさら。
- 同じくCampbell事件最判で説示したように、被疑侵害者の誠実・不誠実が、フェア・ユースの判断に影響を与えることには懐疑的である。
  この点、一般論とは別に、本件では、他の要素がフェア・ユースの成立を支持していることと、陪審員がGoogleに有利な判断を示したことを考慮すると、なおさらである。

【21】

＜第３要素＞　⇒　フェア・ユースに有利

- Googleは、合計約11,500行を複製したが、これは、GoogleがJavaとAndroidで共通化させようと考えたAPIの宣言コードのほぼすべてに相当する。
- Java API全体は286万行にのぼるので、複製されたのは0.4%に過ぎない。
- 従来から、複製された部分が原著作物の創造的表現の核心である場合は、少量の複製であってもフェア・ユースとは言えないとする一方で、大量の複製でも、複製された部分があまり創造的表現を含まない場合や、正当な目的に必須の複製の場合は、フェア・ユースに該当するとしてきた。
- 長編小説から１行を複製しても実質的とは言えないが、世界で一番短い小説（「彼が目覚めたとき、恐竜はまだそこにいた」(注)という１行からなる）からの複製だったら話は変わってくる。
- Googleはインターフェースの再実装という変容力のある目的のために複製を行った
- 本件のように、複製の量が、有効でかつ変容力のある目的と結びついたものであるならば、第3要素は、フェア・ユースに有利傾く。
- 控訴裁は、もっと少ない複製でも、Javaプログラムを Androidで動くようにすることはできたとするが、それはGoogleの目的ではない。Googleの目的は、プログラマーが、Javaで身につけたスキルをAndroidで活用できるように再実装をすることであった。
- Googleが行った複製なくして、再実装が不可能だったことは陪審も同意するはず。

（注）　A. Monterroso, El Dinosaurio, in Complete Works & Other Stories 42 (E. Grossman transl. 1995)

【23】

＜第４要素＞　⇒　フェア・ユースに有利

- 権利者が損失を被るか否かだけではなくて、その原因も考慮すべき。
  Campbell事件最判で指摘したように、例えば、辛辣なパロディによる原作品の売上減少は、著作権法で保護される損害ではない。
- 本件では、権利者の損害と、複製が公共の利益に与える影響との比較考慮も必要
- 第１審で提示された多数の証拠によって、陪審員は、次の判断が可能であった
  - AndroidがJavaの現実の市場、または潜在的な市場に、損害を与えていないこと
  - GoogleがAPIの一部を複製したかどうかにかかわらず、Sunがスマートフォンの市場に参入できなかったであろうこと
- 潜在的な市場＝理論的な市場と解するべきではない。潜在的な市場＝理論的な市場だとすると、すべての場合に損失が概念されてしまう。
- 複製の結果、GoogleはAndroidプラットフォームから莫大な利益を得たが、その源泉は、プログラマーのJavaへの投資に大きく関係する一方で、Sun(＝Oracle)がJava APIを作成するために行った投資との関係は薄い。よって、Oracleが前記利益の分配を受ける合理的な理由がない。
- 多数のプログラマーがJava API学習のために投資をしてきたことを考慮すると、本件でOracleに著作権の行使を認めることは、公衆に害を及ぼす危険性がある。宣言コードについての権利行使を認めると、将来の新しいプログラムの創造を制限することになる。
- ①Sunはスマートフォン市場で競争する力に欠けていたこと、②Googleの利益の源泉、③公衆の創造性を害するリスクを総合すると、第４要素もフェア・ユースに有利

【25】

≪まとめ≫

- 本件では、伝統的な著作権法の概念も、フェア・ユースに関する先例も、変更していない
- Googleは、ユーザーが蓄積した能力を、新しくかつ変容力のあるプログラムに活かすために必要なものだけを抽出して、ユーザーインターフェースを再実装したのであるから、GoogleによるJava APIの複製は、法律問題としてフェア・ユースである。
- 連邦巡回控訴裁判所のこれに反する判決は取り消され、事件は、本判決に従う形でさらに手続きが進められるべく、差し戻される。

【27】

■検討

≪全体≫

・ 判決自身も指摘するように、先例を、明示的には変更していない
・ （Campbell事件最判で「変容力」概念が導入されたような）新概念の導入もない
　⇒ 今後も、フェア・ユース判断にあたっては、Campbell事件最判の説示内容が基本となるだろう

先例のReplacementはなかった。しかし・・・

【28】

■検討

先例のReplacementはなかった。しかし、Refinement はあった。具体的には・・・

≪全体≫

・ 検討が、第2要素からはじめられたのは、最近の判決には珍しい
　⇒ 下級審は、本件の事情の故と理解するか？　それとも、これがきっかけとなって、4要素の相対化につながるか？

・ 各要素の検討において、法的な観点から結論を導くというよりも、陪審員の判断を踏まえて結論を導くような傾向がある
　⇒ 単純に言えば、既に陪審員による審理が終わっているという本件の事情を反映してのことと思われる
　⇒ ただ、本判決では、フェア・ユースが法律問題か事実問題かについて詳しく判断しているため、本判決の傾向に、下級審が（過剰反応して）影響を受ける可能性は高いのではないか？
　　例えば、サマリージャッジメント段階でのフェア・ユース認定に影響する？
　　（被疑侵害者側にとって、負担が増える？）

・ 著作権法が独占を認めるべき範囲はどこまでか、という問題がかなり前面に出ている
　⇒（下級審判決の）予測可能性が、従来よりも下がるのではないか？

【30】

≪第2要素≫
- 従来、Campbell事件最判の影響でほとんど無視されてきた第2要素であるが、本判決によって、その位置づけが見直されることになるのではないか？
  ⇒ パロディの事案（Campbell事件）で軽視されるのは当然だが、それ以外の本来重視すべきだった事案でも軽視・無視されてきたのは、下級審の過剰反応
- 書きぶりを見る限り、API宣言コードの著作物性には否定的と思われる

≪第1要素≫
- 第1要素の検討で、変容力の有無が、最重要であることに変化はない
- 既存著作物の表現形式に変更がなくても、変容力ありと最高裁が判断したのは重要

	目的・性格の変更ある利用	目的・性格の変更なき利用 ＝代替的な利用
表現形式の変更あり	パロディ Campbell事件	単なる翻案 ⇒ 侵害
表現形式の変更なし	検索エンジン Perfect10事件 本件	単なる複製 ⇒ 侵害

- リバースエンジニアリングや互換製品の開発は、今まで以上にフェア・ユースとされやすくなった
- 商業性や不誠実性がフェア・ユース判断に悪影響を与えるという考え方は、既にCampbell事件最判で否定されていたが、本判決でとどめを刺された感じ

【32】

≪第3要素≫
- 許される複製の程度（量的・質的）は、利用の目的・性格に照らして判断するという、Campbell事件最判の考え方が再確認された
  ⇒ Campbell事件の場合は、質的な面が検討の中心であったが、今回は量的な面が検討の中心となった
- リバースエンジニアリング、検索エンジン、機械学習など、全部複製しないと目的を達し得ない場合について、この要件が不利に働くことはなくなった

≪第4要素≫
- Campbell事件以降、下級審は、変容力があれば市場代替性は低く、市場への害も少ない、という一種の「公式」を多用してきたが、本件ではそこに言及がない
  ⇒ 下級審（特に地裁に）にどう影響するか ⇐「巡回区判例による遅延現象」＊
- 利益の源泉を問うのは特徴的な視点
  ⇒ 著作権が保護すべき利益かどうかという問題設定につながる
  ⇒ 当然の問題設定ではあるが、予測可能性は下がるのではないか？
- 陪審員の判断に大きく依拠している
  ⇒ サマリージャッジメントなどでの判断が難しくなるのでは？ （既述）
- Nimmerの論文を引用する形で、潜在的な市場を、理論的な市場と解すべきではないことが明記されたのは大きい
  ⇒ この考え方は、日本の権利制限規定の、「権利者の利益を不当に害す場合」の判断にも示唆的では？ （権利制限規定に基づいて利用が行われれば、大なり小なり権利者に不利益が発生する。問題は「不当」かどうか）

【34】

■トーマス判事の反対意見 （⇒それに対する簡単なコメント）

・宣言コードと実装コードは密接に結びついており、分離して考えるのは間違い
　⇒ 小説でも同じだが、そこから、非創作的部分を特定するのは普通ではないか？
・AppleやMicrosoftが独自の宣言コードを書くことができた以上、融合理論は非適用
　⇒ AppleなどのAPIは、Javaとは別物ではないか？ アイデアが多数あるに過ぎない。
・法廷意見が、法定の順番でも重要性（①Harper&Row最判に基づき第4要素 ②第1要素）でもなく、第2要素から検討し、宣言コードを保護の必要性が薄いものとした上で、他の要素を検討しているのは大きな間違い。結果、フェア・ユース判断が間違ってる
　⇒ 第4要素が最重要というのは現状少数説
・全ての著作物の表現はアイデアと結びついているし、著作物の習熟にその利用者が投資するのは宣言コードに限らない。法廷意見の第2要素判断は間違い。
・Googleの行為によって、Oracleは多大な損害を被っている（Javaプラットフォームの価値が下がり、またスマフォメーカーにJavaPFのライセンスする機会を失った）
・商業性は決め手ではないが、何百億ドルもの無許諾複製をFUとしたことはない
　⇒ 著作権で保護されるべき損害かという視点に欠ける。第4要素が最重要という少数説の立場の帰結
・GoogleはOracleと同じ目的で宣言コードを複製しており、変容力はない。法廷意見では、変容力がある＝他者が新しいプログラムを創作する、であり間違い。
　⇒ 法廷意見の指摘のように、機能を呼び出すことを利用目的と捉えると、全ての場合となる
・宣言コードは核心部分であり、変容力がない以上、大量の複製は正当化不可

【36】

〔参考〕「巡回区判例による遅延現象」について

・「巡回区判例による遅延現象」は、奥邨が名付けたものであり、一般的用語ではない
・著作権分野で新しい最高裁判決が出た後も、地裁判決にその影響が現れるのに時間がかかる状況がしばしば見られる
・この原因は、地裁にとって、従うべき判例には、最高裁のものだけでなく、自身が属する巡回区の控訴裁のものも含まれていることが指摘できる
・（新しい）最高裁判例と、（従来からある）控訴裁判例が、必ずしも整合しないとき、地裁としてはどうすべきかジレンマに陥る
　⇒ 本来は、控訴裁判例を無視すべきなのだろうが、なんとか整合させようと工夫を凝らすケースが少なくない。結果、最高裁判例の影響が減衰する。
　⇒ 最高裁判例の対象となっていない巡回区で生じやすい。
　例えば、最高裁が、第3巡回区控訴裁の判決を破棄差し戻しした場合、第3巡回区については控訴裁判例が軌道修正されるが、他の巡回区はそうではない
　そのため、例えば、第7巡回区の地裁は、（整合しなさそうに見える自身の巡回区）控訴裁判例を、どう扱うべきかジレンマに陥る
・控訴裁が、その判例について、軌道修正を行ってくれるまで、この現象は続く
　⇒ よって「巡回区判例による遅延現象」

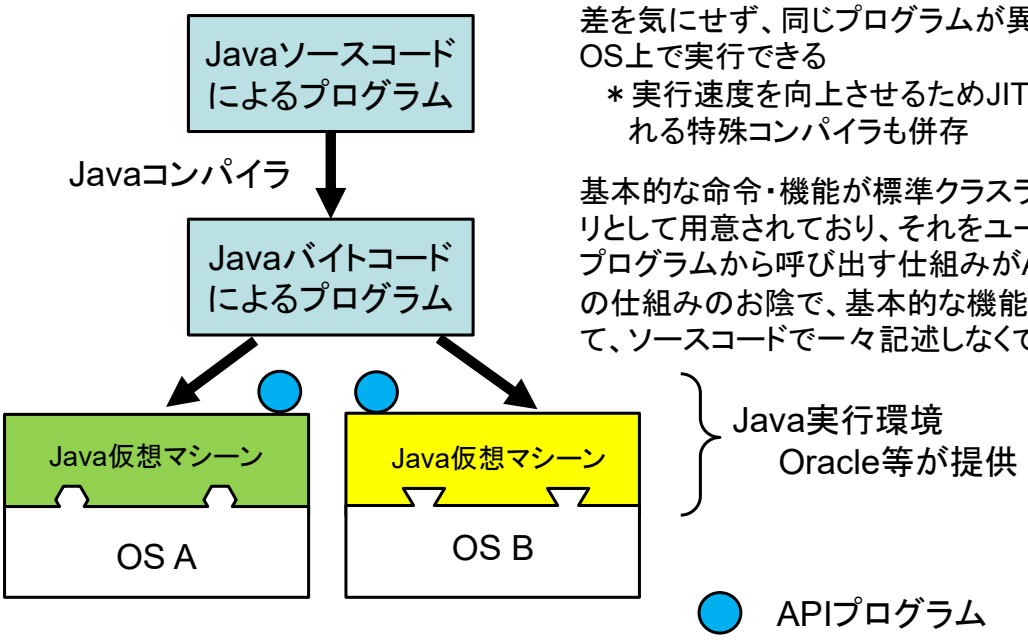
【37】

▸Javaの仕組み

Javaは、言語仕様、コンパイラ、実行環境（仮想マシーン＋APIプログラム）からなる

仮想マシーン（バイトコードを機械語に変換するインタプリター*）のお陰で、OS毎の差を気にせず、同じプログラムが異なるOS上で実行できる
＊実行速度を向上させるためJITと呼ばれる特殊コンパイラも併存

基本的な命令・機能が標準クラスライブラリとして用意されており、それをユーザープログラムから呼び出す仕組みがAPI。この仕組みのお陰で、基本的な機能について、ソースコードで一々記述しなくてすむ

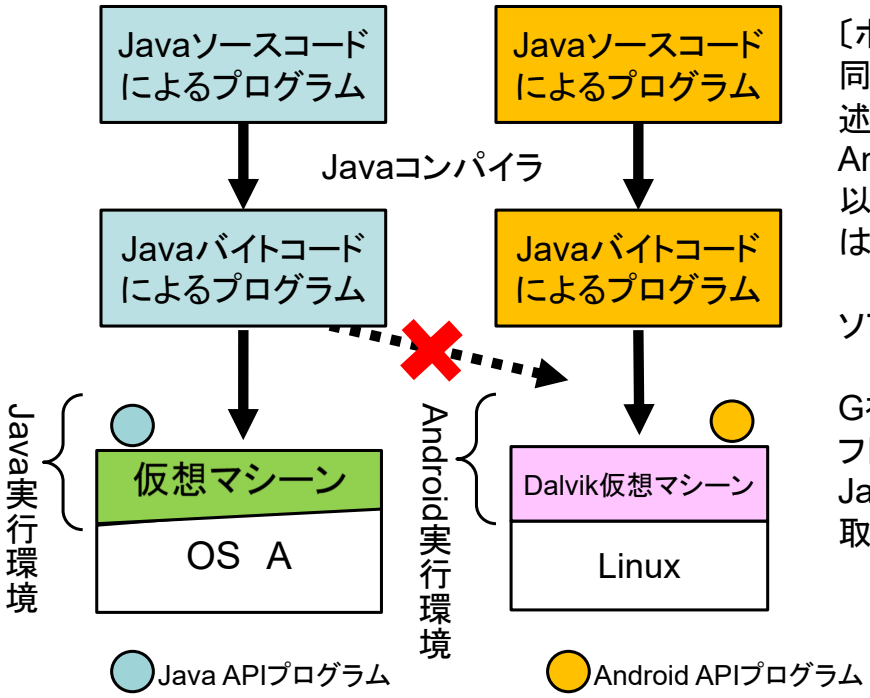


【38】

▸事実関係

Android実行環境はGoogleが提供

〔ポイント〕
同じJava言語で記述されていても、Android用のソフト以外は、Androidでは実行できない
⇩
ソフトの互換性なし
⇩
G社はAndroid用ソフトの開発者としてJavaプログラマの取り込みを狙った



【39】

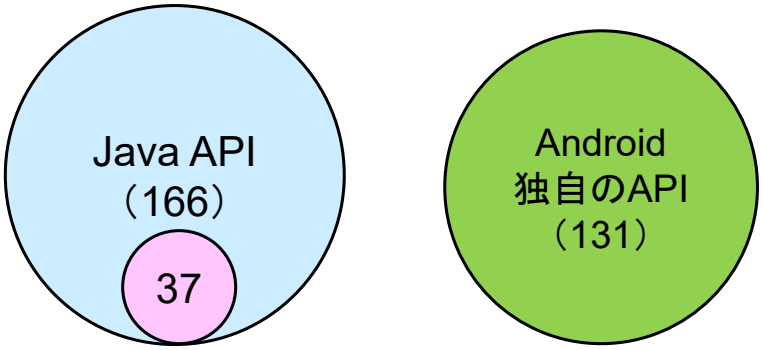
▸事実関係

AndroidのAPIは・・・

総計166パッケージあるJava APIの一部（37パッケージ）

＋

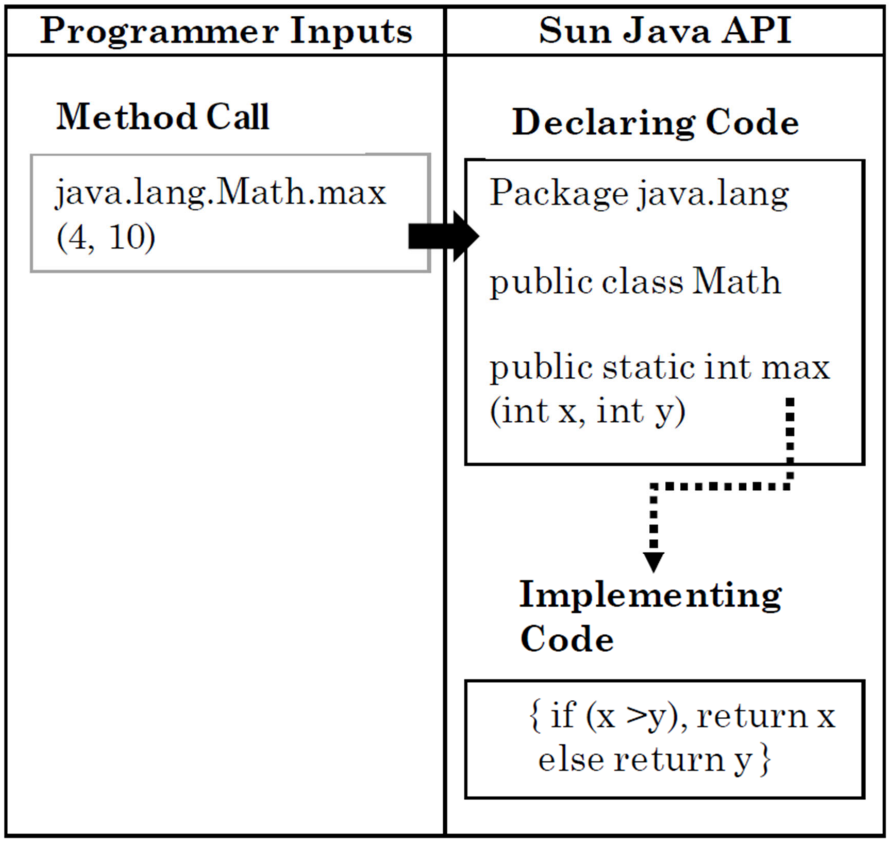
Android独自のAPI（１３１パッケージ）



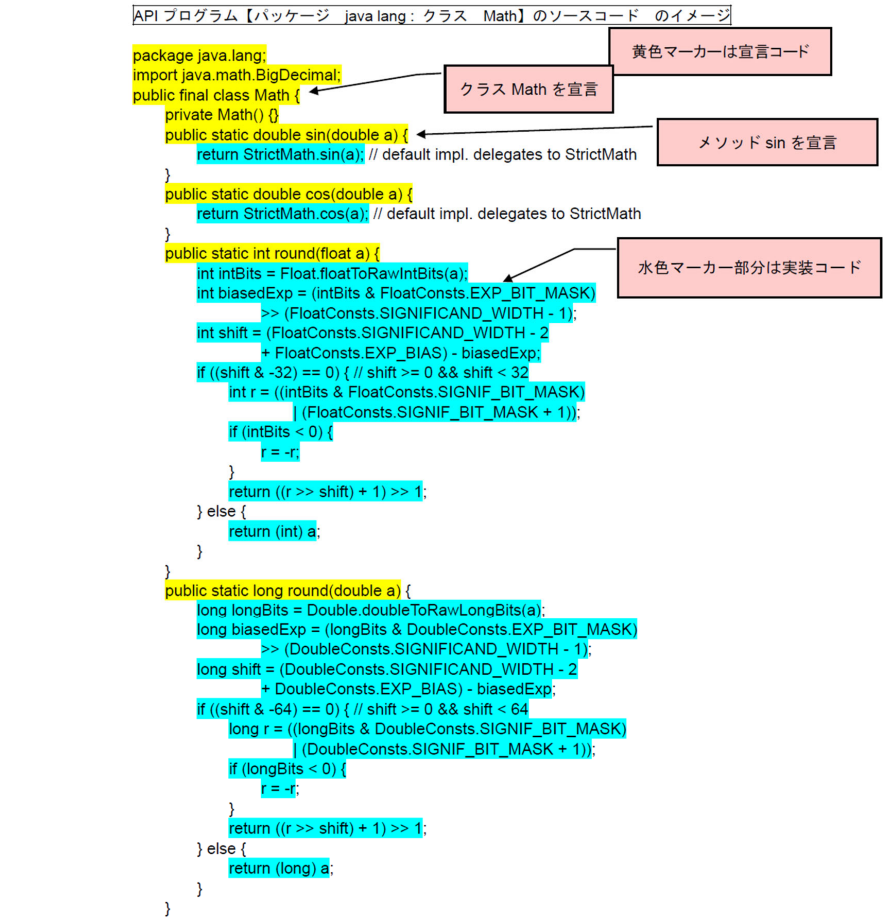
Java API:	166パッケージ、3000クラス、30000メソッド
G社が利用した一部：（赤い部分）	37パッケージ、600クラス、6000メソッド

【40】

Sun Java API Diagram

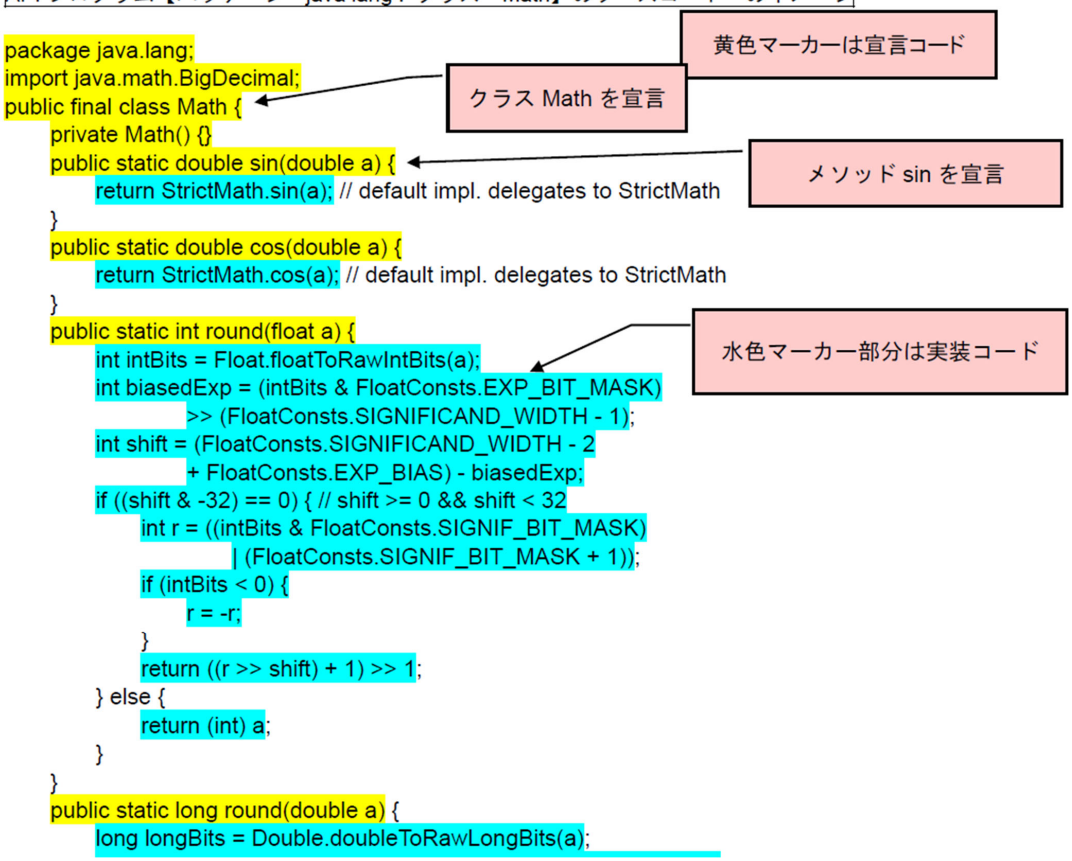


API プログラム【パッケージ　java lang：クラス　Math】のソースコード　のイメージ




```
package java.lang;
import java.math.BigDecimal;
public final class Math {
    private Math() {}
    public static double sin(double a) {
        return StrictMath.sin(a); // default impl. delegates to StrictMath
    }
    public static double cos(double a) {
        return StrictMath.cos(a); // default impl. delegates to StrictMath
    }
    public static int round(float a) {
        int intBits = Float.floatToRawIntBits(a);
        int biasedExp = (intBits & FloatConsts.EXP_BIT_MASK)
                >> (FloatConsts.SIGNIFICAND_WIDTH - 1);
        int shift = (FloatConsts.SIGNIFICAND_WIDTH - 2
                + FloatConsts.EXP_BIAS) - biasedExp;
        if ((shift & -32) == 0) { // shift >= 0 && shift < 32
            int r = ((intBits & FloatConsts.SIGNIF_BIT_MASK)
                    | (FloatConsts.SIGNIF_BIT_MASK + 1));
            if (intBits < 0) {
                r = -r;
            }
            return ((r >> shift) + 1) >> 1;
        } else {
            return (int) a;
        }
    }
    public static long round(double a) {
        long longBits = Double.doubleToRawLongBits(a);
        long biasedExp = (longBits & DoubleConsts.EXP_BIT_MASK)
                >> (DoubleConsts.SIGNIFICAND_WIDTH - 1);
        long shift = (DoubleConsts.SIGNIFICAND_WIDTH - 2
                + DoubleConsts.EXP_BIAS) - biasedExp;
        if ((shift & -64) == 0) { // shift >= 0 && shift < 64
            long r = ((longBits & DoubleConsts.SIGNIF_BIT_MASK)
                    | (DoubleConsts.SIGNIF_BIT_MASK + 1));
            if (longBits < 0) {
                r = -r;
            }
            return ((r >> shift) + 1) >> 1;
        } else {
            return (long) a;
        }
    }
```

API プログラム【パッケージ　java lang：クラス　Math】のソースコード　のイメージ




```
package java.lang;
import java.math.BigDecimal;
public final class Math {
    private Math() {}
    public static double sin(double a) {
        return StrictMath.sin(a); // default impl. delegates to StrictMath
    }
    public static double cos(double a) {
        return StrictMath.cos(a); // default impl. delegates to StrictMath
    }
    public static int round(float a) {
        int intBits = Float.floatToRawIntBits(a);
        int biasedExp = (intBits & FloatConsts.EXP_BIT_MASK)
                >> (FloatConsts.SIGNIFICAND_WIDTH - 1);
        int shift = (FloatConsts.SIGNIFICAND_WIDTH - 2
                + FloatConsts.EXP_BIAS) - biasedExp;
        if ((shift & -32) == 0) { // shift >= 0 && shift < 32
            int r = ((intBits & FloatConsts.SIGNIF_BIT_MASK)
                    | (FloatConsts.SIGNIF_BIT_MASK + 1));
            if (intBits < 0) {
                r = -r;
            }
            return ((r >> shift) + 1) >> 1;
        } else {
            return (int) a;
        }
    }
    public static long round(double a) {
        long longBits = Double.doubleToRawLongBits(a);
```

■そのAPIは何を指していますか？

APIー般と考えると間違い

- Google v. Oracle事件に関する議論では、Java APIという言葉で何を指しているかが結構混乱気味ではないだろうか？
- Java APIは、Javaプラットフォームが用意する標準クラス・メソッド（≒基本命令とその機能）を利用する仕組みのこと
- 問題はこの「仕組み」の中に実は3つのものが混じっていること

①標準クラス・メソッド（≒命令とその機能）の呼び出し方法

②呼び出される標準クラス・メソッド（≒命令とその機能）そのもの

③標準クラス・メソッド（≒命令とその機能）を実現するプログラム

例えば、判決がJava APIの宣言コードと名づけているのは、それは③のクラス・メソッドの宣言部分という意味。決して①の意味ではない！
今回の問題は、③のプログラム中の、①や②の影響を強く受ける部分（宣言コードやその構成）に著作物性があるか否か、利用はフェア・ユースか否か。
①や②については議論の対象外。
そもそも、①や②は、方法や機能なので、著作権法上はアイデアの部類であり保護の対象外のはず。

【44】

▸事実関係

Java APIとは何か・・・

⇒一般に、API（Application Program Interface）とは、アプリケーションプログラムが、OSなどの機能を利用するために、OSなどに予め用意されている仕組みをいう

⇒例えば、MS-Wordでファイルを HDDに保存する機能について考えると、Wordのプログラマは、自分でそういう機能をプログラムする必要はなくて、OSに用意されている仕組みを利用すれば済むのと同じようなイメージ

⇒Javaの場合、Java仮想マシーンの機能を利用する仕組みという意味で、APIと呼ばれているが、標準クラスライブラリと呼ばれることもある

⇒クラスというのは、昔々のプログラミングモデルでいえば、モジュールとかサブルーチンに近いイメージ

【45】

▸事実関係

Java APIとは何か・・・

⇒ つまり、頻繁に利用されるモジュール群を、Oracle側が作成してライブリーとして配布してくれていることになる

⇒ ここで、重要なのは、Javaのプログラマにとって、標準クラスライブラリに含まれるクラスは、Java言語の命令語の一種のような感じで使われていることである

⇒ 例えば画面に円を描く場合、本来なら、sin・cosなどの三角関数を利用して1ドットずつ表示位置を求めていくことになる
しかし、Javaの標準クラスライブラリ（＝API）には
java.awt.Graphicsというクラスが用意されており、当該クラスのdrawOvalという機能（＝メソッド）を利用すれば、四角形の4つの頂点を与えるだけで、それに内接する円を描いてくれる

⇒ つまり、円を描く専用命令みたいなもの

【46】

▸事実関係

Java APIとは何か・・・

⇒ このように、モジュールを一種の専用命令的に再利用しやすくするのは、オブジェクト指向言語（C++やJavaが代表）の特徴

⇒ なお、Javaプログラマは、各クラス（＝モジュール）がどういう仕組み（＝コード）で実現されているかは知らない。知っているのは、クラスの名前と、その機能（＝メソッド）、処理して欲しいデータ（＝引数）の渡し方、などだけである
  ＊ 命令語の場合に、命令語の名前と文法さえ知っていれば利用できるのと同じ

例： クラス名： java.awt.Graphics
メソッド名 ： drawOval (左隅X, 左隅Y, 幅, 高さ)

⇒ 同じ機能を果たすものでも、クラス名やメソッド名が異なると、プログラマは容易に見つけられない

【47】

▸事実関係

⇒ もっとも、G社は37のJava APIを丸々コピーしたわけではない

⇒ G社は、API（＝クラス）の中身のコードは、全部自分たちで一から記述した（公開されている、各クラスの機能についての資料をもとにして独自創作したものと思われる）。O社もその点で、著作権侵害があったと主張しているわけではない。

⇒ G社がコピーしたのは、クラスの名前、メソッドの名前、そしてパッケージ内でのクラスの構成、各クラスに含まれるメソッドの組合わせだった。

⇒ そして、それを可能にするために、APIの宣言コード（クラスの名前が一覧化されている）を逐語的に複製した

【48】

今回の問題を喩えると・・・ ★

公認　野球規則
1.00　試合の目的
2.00　競技場
2.01　競技場の設定
〔略〕
5.00　試合の進行
5.01　試合の開始

野球規則　逐条解説X
1.00　試合の目的
本章は試合の目的について述べる。
〔以下、逐条解説としての解説文章〕
2.00　競技場
本章は、1）競技場の設定、2）本塁、3）塁・・・から構成される
2.01　競技場の設定
〔以下、逐条解説としての解説文章〕
〔略〕

項目立てと各章・各節の冒頭の章・節の構成内容を述べる文章はXのものからコピー

野球規則　逐条解説Y
1.00　試合の目的
本章は試合の目的について述べる。
〔以下、逐条解説としての解説文章〕
2.00　競技場
本章は、1）競技場の設定、2）本塁、3）塁・・・から構成される
2.01　競技場の設定
〔以下、逐条解説としての解説文章〕
〔略〕

解説文章はY独自執筆

【49】

今回の問題を喩えると・・・ ★

Xは、公認野球規則の項目（章・節・款・条）・その体系・見出しをそっくりそのまま利用した、逐条解説を執筆。各章・各節・各款・各条の冒頭には、その章・節・款・条の構成内容を簡潔に述べる行（例：本章は、1 競技場の設定、2 本塁、3 塁・・・から構成される）が存在。その行の後、詳しい解説文章が続く。

Yは、Xの逐条解説の項目（章・節・款・条）・その体系・見出しと、各章・各節・各款・各条の冒頭の、その章・節・款・条の構成内容を簡潔に述べる行の全てをそのまま複製。具体的な解説文章自体は、Yが独自に著作。

野球規則の項目・その体系・見出し	⇒	Java言語の命令語や文法に相当（クラスなど）
各章などの冒頭の簡単な文章	⇒	宣言コードに相当
詳しい解説文章	⇒	実装コードに相当

【50】

Google v. Oracle事件最高裁判決

【51】

SOFTIC　権利保護委員会（01/16/2023）

Google　v. Oracle 事件連邦最高裁判決における API の著作物性について
―議論のための素材提供

平嶋　竜太

1. Google　v. Oracle 事件連邦最高裁判決―Thomas 判事による反対意見への注目
2. 学説における議論と私見－著作物性の評価に関連して
3. 日本法に置き換えた著作物性の検討

1. Google　v. Oracle 事件連邦最高裁判決―Thomas 判事による反対意見への注目

・多数意見が著作物性についての判断を示すことなく、フェアユースにより非侵害との判断に至った点を批判。

⇒JavaAPI の著作物性を肯定して、フェアユースによる著作権侵害を否定する結論も誤っているとの見解。

そのロジックの概要は以下。

宣言コードの著作物性については、著作権法の文言（"'computer program'"－ "a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result."§101.）より肯定されることは明らかとの立場を示す。

「宣言コード→機能的な表現→操作方法（102 条（b））に該当→著作物性なし」というロジックは間違い！

⇔機能的という意味では宣言コードも実装コードも変わらないし、宣言コードを用いるというアイデアは保護されないとしても、そのアイデアに対する特定の表現は著作物たりうるのにもかかわらず、多数意見は、立法の趣旨に反して、宣言コードと実装コードの区別を主たる根拠としている（The majority holds otherwise—concluding that every factor favors Google—by relying, in large part, on a distinction it draws between declaring and implementing code, a distinction that the statute rejects.）、と批判。

2. 学説における議論と私見－著作物性の評価に関連して

（学説における議論概観）⇔あくまでパテント論文執筆時点の状況！

・本件事案について、API の著作物性そのものを否定したうえで結論が導かれるべきとする見解が学説では当初から広く展開されてきた。

（例えば、Peter S. Menell," RISE OF THE API COPYRIGHT DEAD?: AN UPDATED EPITAPH FOR COPYRIGHT PROTECTION OF NETWORK AND FUNCTIONAL FEATURES OF COMPUTER SOFTWARE",31 Harv.J. L.&Tech305(2018)）

・加えて、コンピュータプログラムにおける互換性（compatibility/interoperability）確保のために、著作権侵害の否定やプログラムにおける一定部分については著作物として保護すること自体を否定する考え方が判例上も以前から確立されていたといえる。

（Computer Associates Int'l, Inc. v. Altai, Inc. 982 F.2d 693 (2d Cir. 1992)（いわゆる"abstraction-filtration-comparison" (AFC) test を提示して、異なるソフトウエア間における compatibility に係る要素における類似性があることをもっても、著作権侵害を否定)、Sega Enterprises Ltd. v. Accolade, Inc., 977 F.2d 1510 (9th Cir. 1992)（ソフトウエアのリヴァースエンジニアリングに関連して、ソフトウエアの構成要素のうち一定要素について、著作権法 102 条 (b) により著作物性を否定)、Sony Computer Entm't, Inc. v. Connectix Corp., 203 F.3d 596, 606 (9th Cir. 2000)（テスト目的での複製について、フェアユースが成立。)、Lotus Development Corp. v. Borland International, Inc., 49 F.3d 807 (1st Cir. 1995)（表計算ソフト階層構造について、著作権法 102 条（b）により著作物性を否定。))

⇔Google も、JavaAPI の宣言コードは 102 条（b）の「操作方法」に該当するという主張を積極的に行ってきた。

・従来からの学説でも、interoperability 確保のため API の著作物性についてはアプリオリに否定されるべきことが妥当であることを提示する説はかなり有力といえる。このような立場からは、本件判決の結論自体については妥当とするものの、著作物性についての明確な判断を示さなかった点について批判もみられる。(Lemley, Mark A. and Samuelson, Pamela, Interfaces and Interoperability After Google v. Oracle (August 2, 2021). Available at SSRN: https://ssrn.com/abstract=3898154　)

・これまでの学説等において特に興味深い指摘

本件判決の論理構成（フェアユース法理の適用により Oracle の著作権行使を否定）→フェアユース法理の遠大なる（いき過ぎた？）拡大 (a far-reaching expansion of the fair use doctrine)。→結論のためにフェアユースに依存した点であるとする見解がみられる。フェアユース法理適用の分析において著作物性について検討されるべき事柄を取り入れているという点で、プログラムにおけるフェアユース法理の新しい地平を拓いたもの、とも評価する。

（Balganesh, Shyamkrishna, The Institutionalist Turn in Supreme Court Copyright Jurisprudence (May 5, 2021). 2021 Supreme Court Review (forthcoming), Available at SSRN: https://ssrn.com/abstract=3842276)

この見解によると、著作権法 102 条（b）により宣言コードの著作物性を否定する論理構成を採るとプログラムのうち著作物性を否定されるものとそうでないものを分けることになり、結果的にあらゆるコンピュータプログラムについても著作物性を否定する結論に偏ることになるのではないかという懸念が oral argument を通じて

も連邦最高裁判事の間である程度醸成されており、そのような条文解釈による著作物性を示すことに対する懸念と著作権行使に否定的なIT業界の意見との間を「調和」させるべく、フェアユース法理の枠組みの中で著作物性を否定するような論理が取り込まれたものがBreyer判事による多数意見ではないのか、という分析が示されている。

（とはいえ、何ゆえにストレートに著作物性の是非を検討することに対してそこまで及び腰なのかいま一つ判然としないが……）

（私見）

本件事案におけるJavaAPIの位置付け

－元々はSun自らがJavaAPIの標準化作業の中心的な役割を果たしながらも、個別の仕様についてはコミュニティの参画を通じて形成されてきたという経緯。

その後はOracle自体もオープンソースライセンスの下でJavaAPIの提供を始めた。

→このような一連の形成過程における「特殊性」についても注目すべきではないのか。

すなわち、JavaAPI自体が、元々、社会に対して広く開放された状況の下で形成されてきたという経緯をもってすれば、API 自体の著作物性が肯定されるという仮定の下にあるとしても、フェアユース法理を適用して、JavaAPIに係る著作権行使を否定するという結論自体についての違和感は緩和されているようにも思われる。

⇔これまで構築されてきたフェアユース法理の枠組みでうまく収まるものであったのかということは別の問題。

JavaAPIのこのような形成過程と異なり、仮に一つの企業によって完全に閉鎖的な形で開発されてproprietaryに提供されているプログラムであれば、そのプログラムの著作物性を肯定しつつも、多数意見のようなロジックによってフェアユース法理を適用することの妥当性が担保され得るのかという形で捉えなおすとすれば、やはり本件判決の論理構成がAPI一般について妥当するといえるのか、疑念も残るところであると考える。

より一般的に適応しうる論理構成を考えるのであれば、やはりAPI自体の著作物性を一律に否定するか、一定条件を充たすAPIについての著作物性を定型的に否定するという論理構成で結論を導出する方が説得的であったといえるかもしれない。

また、本件事案におけるGoogleによるJava APIの一部の複製利用行為とは、従前の判例で問題となったような純粋な意味での互換性確保やリヴァースエンジニアリングといったものとは、かなり性質が異なるものであるようにも考えられる。そのため、従前のinteroperability確保を巡る議論の延長線上に本件事案の評価をすることについて

は若干慎重であるべきであるようにも考えられる。

→JavaAPI とはあくまでもアプリ開発等のためのプログラミングを支援するツール。片や Android はスマホ用 OS を核として成立するプラットフォームであり、同 OS 上で動作するアプリ開発のためのツールとして観点から、JavaAPI との間に（ある意味で）「バリアフリー化」を図ることでアプリ開発者の層を拡大しうることを大きな目的として、Java API の宣言コードが Android に取り込まれたといえる。

互換性という概念について、仮に、それが実現されることによって、相互にメリットを享受しうるものであると観念するのであれば、Google にとってみれば Java API の宣言コードを Android に取り込むことには明らかに多大なメリットがあったといえるものの、反対に、Java API を基にした JavaSE 等の認証を主なビジネスモデルとしている Oracle 側にとって Android 上のアプリ開発に JavaAPI の宣言コードが活用されることが直ちに大きなメリットをもたらしたといえるのかという点では疑問も残る。

その意味では本件事案における interoperability のベネフィットはやはり片面的であるといえる。

⇔Java 言語をプログラミングに用いるユーザは元々相当程度存在していたことから、Android 上のアプリ開発に Java 言語を利用することかが可能となること自体が Oracle にとって直ちにそのビジネスモデルの上でメリットが生じるとは簡単には評しえないからである。

（もっとも、Lemley, Mark A. and Samuelson,supra は、そもそも interoperability を巡っては、一方向（oneway）的なものとなりうることは当然想定しうるという理解をしているようである。）

したがって、従来議論されてきたコンピュータプログラムにおける互換性（compatibility/interoperability）確保だけを根拠として著作物性を否定することを、本件事案のような場合にそのまま適用することは適切とは言えないようにも考えられる。

→その意味では Thomas 判事の反対意見にも一定の合理性は見いだせるように思われる。

その他、

宣言コードを「操作方法」と同様に単なる技術的アイデアと同質のものとして捉えることによって、解釈論上も類型的に著作物性を否定するという考え方

－理論的には否定することはできないとしても、パッケージ、クラスとしての分類も含めた宣言コード自体の名称記述については本来的には無限に存在しうる多種多様な記述を選択肢とすることもありうる。その各表現について基本的には創作性が高いとはいいがたいものが多くを占めるのかもしれないが、アプリオリに著作物性がないことを肯定することにもやや疑念が残る。

3. 日本法に置き換えた著作物性の検討

本件事案について、仮に日本法を前提として考える場合には、Java API についての著作物性はどのように評価されるべきであろうか？

Java API 自体－Java 言語という特定のプログラム言語で記述された実装コードとしてのプログラムを前提として、一定のタスクを実行させるための特定の処理手順に対応している個々のパーツのようなプログラム（メソッドと呼ばれている）であって、個々に対して特定の名称を付与し、さらにそれぞれの機能ごとに適切な集合たる、クラス、パッケージといった群に分類を行うことで体系的に構成された類型を意味するものといえる。

日本の著作権法 10 条 3 項では、プログラム著作物を作成するために用いるプログラム言語、規約及び解法には著作権法による保護は及ばない旨を定めており、**JavaAPI は、「規約」と解釈できるのか否か？**

→同規定における「規約」とは、「特定のプログラムにおける前号のプログラム言語の用法についての特別の約束」とされており、例えば、特定のプログラムと連結して使うために則る必要のある一定のルール、データを複数の電子計算機で処理するための共通の約束事（加戸守行・著作権法逐条講義（七訂新版）129 頁）

学説では、規約とはプログラムにおける約束事であって、インタフェースについては一般論として「規約」に該当すると解する見解（中山信弘・著作権法（第 3 版）137 頁）もみられる。

しかしながら、「インタフェース」の概念の中に、JavaAPI のようにプログラミングに際して共通に用いられるソフトウエアの部品の集合体のようなものまで包含されるものと解されるのかについては、学説上の明確な見解がみられるとはいえない状況にあると考えられる。当然ながら JavaAPI が言葉としてはインタフェースという呼称を含むものであることをもって直ちに著作権法上の「規約」と評価できるとはいえないと考えられる。

Java 言語を用いたプログラム開発の現実として、メソッドやクラスが既に用意されており、それを用いて実現できる役割機能である場合には、Java 言語を用いてわざわざ一から独自に別途のコードを記述するという作業が行われることはおそらく限られている（⇔むしろ現実にはほとんどない！）ものと考えられ、JavaAPI に含まれているパッケージ、クラス、メソッドを呼び出して用いるということの方がむしろ一般的であると考えられる。

このため、JavaAPI の構成についても、Java 言語における用法の一種として解することは解釈論としてもさほど無理はないものと考えられる。このようなことから、JavaAPI の「構成」自体については、Java 言語というプログラム言語における「用法

についての特別の約束」に近似するものと解することによって、「規約」として著作権法による保護は及ぶものではないという結論を導くことも一般論としては可能であると考えられる。

ところで、著作権法 10 条 3 項とは、そもそも技術的アイデア自体が著作物として保護されないとする考え方を踏まえた確認規定と解されている。（加戸守行・著作権法逐条講義（七訂新版）135 頁）

→**JavaAPI 自体が技術的アイデア自体と評価しうるのか**検討することも要すると考えられる。

JavaAPI については、一定のタスクを実行させるための特定の処理手順に対応している個々のパーツのようなプログラム（メソッドと呼ばれている）に振り分けて、個々に対して特定の名称を付与し、さらにそれぞれの機能ごとに適切な集合たる、クラス、パッケージといった群に分類を行うことで体系的な類型とするという構成自体については、まさに Java 言語によるプログラミング体系についてのアイデアそのものと考えられることから、そのような体系的構成を Google がそのまま Android に取り入れることについては、なんら著作権侵害を構成するものとは評価し得ないと考えられる。

他方、本件事案で Google による行為が問題となったのは、JavaAPI の体系的構成を Android に取り入れたということよりも、むしろ、Java API に包含される個々のプログラムのパーツを呼び出すための個々の API を呼び出すために必要となる名称である宣言コードの部分についての複製行為であることに注意すべきである。

宣言コードとして、メソッドやクラス、パッケージに付与された名称記述については、主としてその機能を示す単語や文章を短縮化したような記述表現が多くみられることから、創作性の高い表現と評価されうるものではないと考えられる。とはいえ、宣言コードの表現自体に限ってみれば、Java 言語というプログラム言語における「用法についての特別の約束」、すなわち著作権法上の「規約」とまで評価しうるのかという点では疑問が生じる。

宣言コードは、あくまで、本来であれば Java 言語の基本体系から記述される特定の機能について記述された「文章」を、いわばショートカットするような形での「略称」として記述されたものと捉えることができるものと考えられる。

創作性の高い表現とはいえないものの技術的アイデア自体やスポーツやゲームのルールと同質のものとまで評価できるかについてはやや躊躇が生じるところである。

加えて、多種多様な表現形態をとっている宣言コードの記述について、画一的に「規約」に属するものと解することには慎重であるべきであるようにも考えられる（もちろん個別の API についてみれば、ほぼ「規約」に相当するものとして、著作

権法による保護は肯定しえないものと解せるものは少なからず存在しているといえよう。)。

また、異なる観点から、**宣言コードについては、そもそも著作権法の下での「プログラム」ではないと解せるのか、**という点も併せて検討する必要があろう。

日本著作権法の下でのプログラムの定義（法 2 条 1 項 10 号の 2)：「電子計算機を機能させて一の結果を得ることができるようにこれに対する指令を組み合わせたものとして表現したものをいう。」－あくまでコンピュータに対しての指令を構成しうるものに限られるものと解される。

JavaAPI の場合、パッケージやクラスに含まれる各機能がメソッドにまで「分解」され、最終的には、対応する実装コードが電子計算機上で処理されることで「電子計算機に対する指令」となっている。その意味では、JavaAPI の宣言コードについては直接的には電子計算機に対する指令を構成するものではなく、プログラム著作物ではないと評価しうるのかということが一応問題となりうる。

→IBF ファイル事件（東京高決・平成 4 年 3 月 31 日・平成 3 年（ラ）142 号）

「電子計算機によるプログラム処理に当たり、あるシステムにおけるプログラムを稼動させ一定の処理をさせるためには、そのプログラムの他、それに処理情報を与えるデータが必要であるが、システムの効率上、データを本体プログラムとは別個のファイルに記録させることがよく行われる。その場合、該ファイルは、プログラムに読み取られその結果電子計算機によって処理されるものではあるが、電子計算機に対する指令の組合せを含むものではないので、著作権法上のプログラムではない。」

「データを記述するに当たり、プログラム自身が規定した一定の記号又は文字（以下「記号等」という。）が記述されていれば、プログラムがそれを読み取ってその記号等に意味付けられた処理を行うとしても、それは、プログラムがその記号等をデータとして読み取り所定の処理を行うものにすぎず、その記号等をもって電子計算機に対する指令であるということはできない。したがって、また、そのような記号等が付されたデータをもって、著作権法上のプログラムであるということはできない。」

JavaAPI の宣言コードについては、直接的にはハードウエアに対する「指令」ではなく、一見上記の「データファイル」であるかのようにも思われるが、宣言コード→Java 言語によるソースコード→オブジェクトコードと変換されることにより、最終的には実装コードとして、ハードウエアによって実行される命令群に変換される以上、データと解することはできないであろう。

以上のように、日本法の下では JavaAPI の構成自体は「規約」と評価しうるかもしれないし、宣言コードそれぞれについてみれば著作物性を充足しないものとして評価さ

れるものが少なからず存在しうるものと考えられる。それらについて著作権侵害は当然問題となり得ない。しかしながら、他方で、JavaAPI の宣言コード全てについて、画一的に著作物性がないものとして著作権法の下での保護を否定することについては慎重であるべきようにも考えられる。

↑

もちろん、個々の宣言コードの著作物性の評価の結論としては、ありふれた表現として著作物性が否定されるものも数多く存するとは考えられるものの、宣言コードといえども創作性の低いものの著作物と解しうるものが存在する余地については否定しきれない。（著作物性が肯定されうる宣言コードについても一部存在すると解すべきように考えられる。）

すると、著作物性が肯定されうる宣言コードについて、本件事案における Google のような行為を行う者に対する著作権エンフォースメントをどのように考えるべきであろうかという問題は残るようにも考えられる。

創作性の低い著作物であるとしても、デッドコピー行為に対して複製権による権利行使は肯定されうる余地はあるものと考えられ、著作物性が肯定される API の宣言コードをデッドコピーして Android に利用した Google の行為に対する著作権行使は日本法の下では許容されるものと解する余地はあるのであろうか？

⇔そこで、日本版フェアユース規定と位置付けられることの多い著作権法 30 条の 4 の適用が注目されるところとなる。

（補）

東京地判・平成 11 年 1 月 29 日・判時 1680 号 119 頁（古文単語語呂合わせ事件）

「著作権法の保護の対象となる著作物については、思想又は感情を創作的に表現したものであることが必要である。ところで、**創作的に表現したものというためには、当該作品が、厳密な意味で、独創性の発揮されたものであることは必要でないが、作成者の何らかの個性の表現されたものであることが必要である。**文章表現に係る作品において、ごく短いものや表現形式に制約があり、他の表現が想定できない場合や、表現が平凡、かつありふれたものである場合には、筆者の個性が現われていないものとして、創作的な表現であると解することはできない。原告語呂合わせに創作性が認められるか否かについては、右の観点に照らして判断すべきである。」

「1　原告書籍一　朝めざましに驚くばかり
　　被告書籍一　朝目覚ましに驚き呆れる」

「原告語呂合わせ 1 は、古語「あさまし」及び古語「めざまし」の二語について、その共通する現代語訳「驚くばかりだ」を一体的に連想させて、容易に記憶ができるようにする目的で、二つの古語のいずれにも発音が類似し、かつ、現代語訳と意味のつながる「朝目覚まし」

という語句を選択して、これに「驚くばかりだ」を続けて、短い文章にしたものである。

右語呂合わせは，極めて短い文であるが、二つの古語を同時に連想させる語句を選択するという工夫が凝らされている点において、原告の個性的な表現がされているので、**著作物性を肯定することができる。**」

「４　原告書籍一　志賀直哉もガーナチョコレートを食べたい

被告書籍一　もー、ガーナチョコがあればなあ」

「原告語呂合わせ４は、古語「しがな」及び「もがな」とその現代語訳「～したい、～たらなあ」を一体的に連想させて、容易に記憶ができるようにする目的で、二つの古語のいずれにも発音が類似し、かつ、現代語訳と意味のつながる「志賀直哉もガーナチョコレートを」という語句を選択して、これに「食べたい」を続けて、短い文章にしたものである。右語呂合わせは、二つの古語を同時に連想させる語句を選択する工夫が凝らされている点において原告の個性が発揮されているので、**著作物性を肯定することができる。」**

「１０　原告書籍一　ゆううつな井伏氏

被告書籍一　井伏氏は憂鬱だ」

「原告語呂合わせ１０は、古語「いぶせし」とその現代語訳「憂鬱だ」を一体的に連想させて、容易に記憶ができるようにする目的で、古語と発音が類似し、かつ、現代語訳と意味のつながる「井伏氏」という人名を選択して、これを「ゆううつな」に続けて、短い文章にしたものである。**右語呂合わせは、ごく平凡で、ありふれたものであり、筆者の個性を発揮した創作的表現とまではいえないから、著作物とは認められない。」**

「２４　原告書籍一　ウ！ツクシかわいい

被告書籍一　うっ、つくしんぼかわいいぞ

原告語呂合わせ２４は、古語「うつくし」とその現代語訳「かわいい」を一体的に連想させて、容易に記憶ができるようにする目的で、古語と発音が類似し、かつ、現代語訳と意味のつながる「ウ！ツクシ」という語句を選択して、これに「かわいい」を続けて、短い文章にしたものである。**右語呂合わせは、ごく平凡で、ありふれたものであり、筆者の個性を発揮した創作的表現とまではいえないから、著作物とは認められない。」**

「３８　原告書籍二　バーヤも行きたい大学へ

被告書籍一　ばーやは死体

原告語呂合わせ３８は、古語「ばや」とその現代語訳「～たい」を一体的に連想させて、容易に記憶ができるようにする目的で、古語と発音が類似し、かつ、現代語訳と意味のつながる「バーヤも」という語句を選択して、「行きたい」及び「大学へ」を続けて、短い文章にしたものである。**右語呂合わせは、ごく平凡で、ありふれたものであり、筆者の個性を発揮した創作的表現とまではいえないから、著作物とは認められない。**」

AWF v. Goldsmith の概要

2023・1・30

弁護士　石新智規

1

The Andy Warhol Foundation For The Visual Arts, Inc., v. Lynn Goldsmith, Lynn Goldsmith, Ltd.

ゴールドスミスは，1984年にヴァニティ・フェア誌にプリンスの肖像写真を利用許諾した。ヴァニティ・フェアは，このライセンスに基づきゴールドスミスが描いたプリンスの肖像画を掲載した。

ゴールドスミスが撮影した写真

アンディーウォーフォール作品

2

AWF v. Goldsmith

- ✓ ウォーホールは、雑誌に掲載した肖像画のほかに同じ写真に基づき14のシルクスクリーン印刷画と2つの鉛筆画のプリンスの肖像画を描いた
- ✓ ウォーホールの死後、ウォーホール財団は，これらの肖像画をプリンスシリーズとしてライセンス
- ✓ 2016年のプリンスの死後、ゴールドスミスはプリンスシリーズの存在を知り、写真の著作権を侵害するものと警告
- ✓ ウォーホール財団は2017年4月に著作権侵害不存在の確認の訴えを提起

3

AWF v. Goldsmith

4

事案の概要

- ✓ 2019年7月1日、ニューヨーク州南部連邦地方裁判所は，ウォーホールの作品がプリンスを生きた人間以上のアイコンとして表現するのに、ゴールドスミスの写真の創作的部分はほとんど捨象されていると判断し，プリンスシリーズの制作を写真のフェアユースであると判断した。
- ✓ ゴールドスミスが控訴

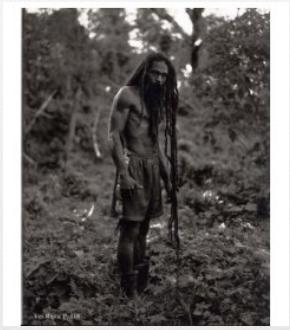
5

第2巡回区控訴審判決

- ✓ 2021年３月26日、第２巡回区控訴裁判所は本件写真の利用をフェアユースとした原審を破棄
- ✓ 第１要素：両者が広い意味で視覚著作物として創作されたというだけでなく、同一人物の肖像であるという点で同一
- ✓ プリンスシリーズが全て２次的著作物に該当するものではないが、その一部は本件写真を高コントラストでスクリーン印刷された異なる方式で描いたものであり、原作品を変容的に利用したものでなく２次的著作物に該当する
- ✓ 第１要素はフェアユースに不利なものである。
- ✓ 同じ第２巡回区においてアプロプリエーション芸術に写真を利用したことがフェアユースと判断された2013年のCariou v. Prince判決では，写真家であるCariouの原作品とは全く異なる美術上の目的を示唆する方法で芸術家のPrinceが表現を付加しており，プリンスシリーズとは対照的であり、この判断は第２巡回区控訴裁判所の先例にも反しない。

6

Cariou v. Prince



Patrick Cariou “Yes Rasta”

Richard Prince “Canal Zone series”

7

第2巡回区控訴審判決

- ✓ 第2要素はフェアユースの判断において重要な役割を果たすことは少ない
- ✓ 本件写真は創作性があり未公表であるという性質から、第2要素についてはフェアユースに不利
- ✓ 第3要素について、プリンスシリーズは本件写真の創作的な部分を削ぐのではなく、むしろ部分的に増幅するものであるから、本件写真の本質的な部分は複製されていると判断し、フェアユースに不利なものである。

8

第2巡回区控訴審判決

- ✓ ミュージシャンの肖像写真を典型的な2次的利用のためにライセンスする市場はすでに存在し、ウォーホールの利用はその市場を害する
- ✓ ウォーホールの無許諾利用を認めれば，そのような典型的な二次的利用の市場をも破壊し，著作権者にインセンティブを与えようとする著作権法に反する結果を招く
- ✓ 第4要素もフェアユースに不利である。

9

判決後の経緯

- 2021年3月26日　第2巡回区控訴審判決
- 2021年4月5日　Google v. Oracle判決
- 2021 年4月23日　AWFがGoogle 判決に基づき再審理申立て
- 2021年8月24日　第2巡回区控訴審が判決を修正するが、結論は維持。
- 2021年9月10日　全員再審理の申立て却下
- 2021年12月9日　上告受理申立て
- 2022年3月28日　上告受理
- 2022年10月12日　口頭弁論

10

Google v. Oracle

- In the context of fair use, we have considered whether the copier's use "adds something new, with a further purpose or different character, altering" the copyrighted work "with new expression, meaning or message." Id., at 579.
- we have used the word "transformative" to describe a copying use that adds something new and important. Campbell, 510 U. S., at 579. An "'artistic painting'" might, for example, fall within the scope of fair use even though it precisely replicates a copyrighted "'advertising logo to make a comment about consumerism.'"
- Or, as we held in Campbell, a parody can be transformative because it comments on the original or criticizes it, for "[p]arody needs to mimic an original to make its point."

11

再審理

[1] After our initial disposition of this appeal, *see Andy Warhol Found. for the Visual Arts, Inc. v. Goldsmith*, 992 F.3d 99 (2d Cir. 2021), the Supreme Court issued its decision in *Google LLC v. Oracle America, Inc.*., 141 S. Ct. 1183 (2021), which discussed the fair-use factors implicated in this case. Shortly thereafter, Plaintiff-Appellee filed a "Petition for Panel Rehearing and Rehearing En Banc" (the "petition"). Apart from its reliance on the *Google* opinion, the petition mostly recycles arguments already made and rejected, and requires little comment. Nevertheless, in order to carefully consider the Supreme Court's most recent teaching on fair use, we hereby GRANT the petition, conclude that additional oral argument is unnecessary, *see* Fed R. App. P. 40(a)(4)(A), withdraw our opinion of March 26, 2021, and issue this amended opinion in its place.

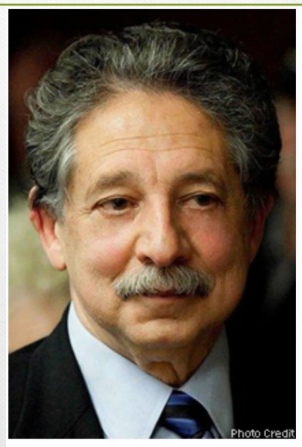
12

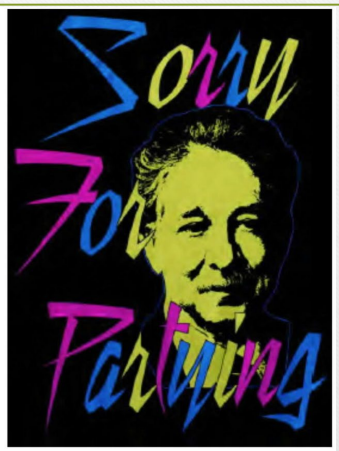
Google判決の影響について

- ✓ Google判決に第2巡回区控訴審の判断が抵触するという主張を認めず、むしろ、その判断はGoogle判決に沿うと指摘。
- ✓ Google判決は、フェアユース判断が極めて事例に応じた判決となるということを明確にしており、コンピュータコードという特殊な事例に関する判断は、本件には当てはまらない（著作権の保護は、著作物の実用的なものではなく、芸術的なものである方が強い）。
- ✓ "[t]he fact that computer programs are primarily functional makes it difficult to apply traditional copyright concepts in that technological world."

13

Kienitz v. Sconnie Nation LLC 766 F.3d 756 (7th Cir. 2014)





UCLA Netanel 教授のPPTより

14

ソフトウェア等の法的保護に関する調査研究委員会
第４回

―APIの法的保護―
日本の著作権法のもとでの
権利制限規定の適用可能性

2023.1.30
シティライツ法律事務所
弁護士　伊藤　雅浩

CITY LIGHTS LAW

テーマ

- 仮にJava APIに著作物性が認められた場合、フェア・ユース規定がない日本の著作権法のもとで、権利制限規定の適用によってJava APIを適法に利用することはできるか。

©Masahiro Ito

著作物性がある場合の権利制限①

（著作物に表現された思想又は感情の享受を目的としない利用）

第三十条の四　著作物は、次に掲げる場合その他の**当該著作物に表現された思想又は感情を自ら享受し又は他人に享受させることを目的としない場合**には、その必要と認められる限度において、いずれの方法によるかを問わず、利用することができる。ただし、**当該著作物の種類及び用途並びに当該利用の態様に照らし著作権者の利益を不当に害することとなる場合**は、この限りでない。

一　二　（略）

三　前二号に掲げる場合のほか、著作物の表現についての人の知覚による認識を伴うことなく当該著作物を電子計算機による情報処理の過程における利用その他の利用（**プログラムの著作物にあつては、当該著作物の電子計算機における実行を除く。**）に供する場合

©Masahiro Ito

「享受」

一般的には「精神的にすぐれたものや物質上の利益などを，受け入れ味わいたのしむこと」を意味するとされており，ある行為が本条に規定する「著作物に表現された思想又は感情」の「享受」を目的とする行為に該当するか否かは，先に述べた立法趣旨及び「享受」の一般的な語義を踏まえ，**著作物等の視聴等を通じて，視聴者等の知的・精神的欲求を満たすという効用を得ることに向けられた行為であるか否か**という観点から判断されることとなるものと考えられる。

文化庁『デジタル化・ネットワーク化の進展に対応した柔軟な権利制限規定に関する基本的な考え方』（2019）38頁

©Masahiro Ito

30条の4における「プログラム」

- プログラムの著作物は表現と機能の複合的性格を有しており，プログラムの著作物に「表現された思想又は感情」とは当該プログラムの機能を意味するものと考えられるところ，その「表現された思想又は感情」の**「享受」に該当するか否かは，当該プログラムを実行等することを通じて，その機能に関する効用を得ることに向けられた行為であるか**という観点から判断されるものと考えられる。（続く）

文化庁『デジタル化・ネットワーク化の進展に対応した柔軟な権利制限規定に関する基本的な考え方』（2019）39頁

30条の4における「プログラム」（承前）

- プログラムの著作物について対価回収の機会が保障されるべき利用は，プログラムの実行等を通じて，プログラムの機能に関する効用を得ることに向けられた利用行為であると考えられることから，そのような目的のない利用行為については権利制限の対象とするのが本条の趣旨に合致するものと考えられる。このようなプログラムの著作物の性質を踏まえると，例えば，プログラムの調査解析を目的とするプログラムの著作物の利用（いわゆる「リバース・エンジニアリング」）は，プログラムを実行すること等によってその機能を享受することに向けられた利用行為ではないと評価できることから，「著作物に表現された思想又は感情を自ら享受し又は他人に享受させることを目的としない場合」に該当するものと考えられる。

30条の4における「プログラム」

- リバース・エンジニアリングは、機能を享受することに向けられた利用行為ではないから、非享受利用だとされた。
- 他方で、API（宣言コード）の場合、当該部分を自己のプログラムに複製することで、コンピュータに解析・実行され、プログラムの機能に関する効用を得ることになる。

©Masahiro Ito

著作物性がある場合の権利制限②

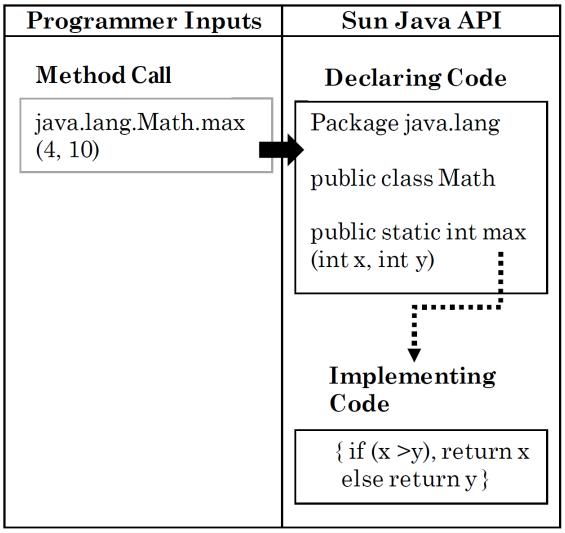
（引用）

第三十二条　公表された著作物は、引用して利用することができる。この場合において、その引用は、**公正な慣行に合致するもの**であり、かつ、報道、批評、研究その他の**引用の目的上正当な範囲内**で行なわれるものでなければならない。

©Masahiro Ito

明瞭区別性と主従関係

- 最判昭55・3・28判時967号45頁（モンタージュ写真事件）
  - ➢旧法下の事件
  - ➢写真著作物の同一性保持権に関するもので、傍論部分
- Declaring CodeとImplementing Codeは、明瞭に区別して認識できる。
- 利用者がImplementing Codeを作成していれば、そちらが「主」で、Declaring Codeは「従」とならないか。



©Masahiro Ito

絵画鑑定書事件（知財高判平22・10・13判時2092号135頁）

「引用としての利用に当たるか否かの判断においては，他人の著作物を利用する側の利用の目的のほか，その方法や態様，利用される著作物の種類や性質，当該著作物の著作権者に及ぼす影響の有無・程度などが総合考慮されなければならない。」

- APIのDeclaring Codeの利用
  - ➢目的：PCソフト開発者のスキルをモバイル開発でも生かせるように
  - ➢方法・態様：宣言コードのみで、実装コードはオリジナル
  - ➢種類や性質：わかりやすく、機能性が求められる部分
  - ➢著作権者に及ぼす影響：市場が異なる

©Masahiro Ito

Google v. Oracle 事件セミナー

日本法との関係

2021.5.10 （財）ソフトウェア情報センター
シティライツ法律事務所
弁護士　伊藤　雅浩

CITY LIGHTS LAW

日本法との関係

- APIの著作物性
  - ➢10条3項との関係
  - ➢部分的な考察・全体的な考察
  - ➢編集著作物
- 権利制限規定に基づく利用可能性
  - ➢30条の4（非享受利用）
  - ➢32条（引用）

©Masahiro Ito

10条3項との関係

そもそもAPIは保護の枠外か

CITY LIGHTS LAW

10条3項

プログラムの著作物

3　第一項第九号に掲げる著作物に対するこの法律による保護は、その著作物を作成するために用いるプログラム言語、規約及び解法に及ばない。この場合において、これらの用語の意義は、次の各号に定めるところによる。

一　**プログラム言語**　プログラムを表現する手段としての文字その他の記号及びその体系をいう。

二　**規約**　特定のプログラムにおける前号のプログラム言語の用法についての特別の約束をいう。

三　**解法**　プログラムにおける電子計算機に対する指令の組合せの方法をいう。

APIは「プログラム言語」「規約」「解法」に含まれるか

©Masahiro Ito

プログラム言語（1号）

- プログラム言語
  - C言語，BASIC，Java，PHPなどの言語のほか，アセンブラ言語や機械語などをいう。
  - 言語の構成要素である，**用語，文法**が含まれる。
  - この言語とその文法は，原理または思想に過ぎず，著作物ではない。

APIは「プログラム言語」そのものや文法ではない

©Masahiro Ito

規約（2号）

- 特定のプログラムにおける表現の特別の約束
- 「例えば同一計算機内で複数のプログラムを用いて連携して**一連の処理をするためにそれらの約束事を共通にする**必要があるということ」（加戸逐条）
- 「**具体的には，インターフェース**やプロトコールのことと考えてよい」（中山『（新版）ソフトウェアの法的保護』43頁）
  - 自己のプログラムの特別な約束事についての独占権の主張を認めたならば・・競争政策観点からは極めて好ましくない
  - スポーツやゲームのルールのようなもの

©Masahiro Ito

規約（2号）承前

- プログラムと規約の境目（前掲・中山45頁）
  （インターフェイスが具体的なプログラムの形で記述されている場合）
  - そのインターフェイスを実現するには当該プログラムへのアクセスが必要になる
  - 「もしこのようなプログラムに著作物性が認められるとすれば，現実には著作物の類似性だけで侵害か否かが決まることになろう」
  - しかし，インターフェイスのプログラムの記述の自由度は極めて狭いため，これらに著作物性を認めるとアイデアの保護になりかねない
  - 「**アイディアと表現とが密接な関係にあるプログラムの規約につき，それがいかなる形態であれ保護しないことを規定したのが，著作権法10条3項2号である**と解すべき」
  - 「保護範囲もまた極めて狭く解すべきであり，**事実上デッド・コピーのみが侵害**となろう」

原則として保護の枠外だが，表現・内容次第か

解法（3号）

- プログラムが処理するための手順をいい，アルゴリズムと呼ばれる
- プログラムの構造・順序・構成（structure, sequence and organization, SSO）のような非言語的要素は原則として著作権による保護対象外

本件で問題となるAPIは「解法」ではない

解法（3号）承前

参考

- 東京地判平15.1.31（製図プログラム事件）
  - ➢原告プログラムの**メニュー表示部における処理の流れ**は、①画面上に、データ作成（入力）、修正、描画、説明、終了の順に各メニューメッセージを表示し、②ユーザーにいずれかのメニュー番号を選択（入力）させ、③ユーザーが入力したメニュー番号に応じて、各機能を実行ファイルを呼び出すものであるが、これらの流れは、**法一〇条三項三号所定の「解法」に当たるというべき**であって、著作権の保護が及ばない。
  - ➢また、「キロ行程最初の値」、「キロ行程オフセット値」、「縮尺」、「用紙サイズ」の順序で変数に値を設定するという処理の流れも、法一〇条三項三号所定の「解法」に当たり、著作物としての保護を受けない。

解法（3号）承前

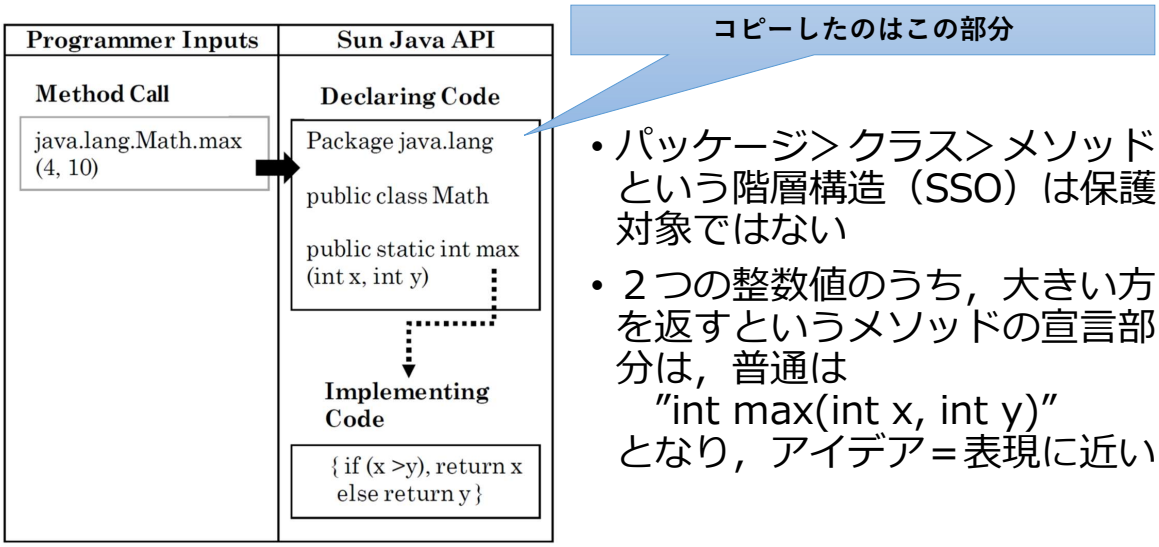
参考

- 知財高判平23.2.28（携帯電話コンテンツ）
  - ➢「②恋愛の神様」のプログラムの一部を構成する日干計算のプログラムは，年，月，日の値を受け取り，１月と２月を前年の１３番目，１４番目の月として取り扱う処理の後に，年を５倍したものと日の値とを加算し，閏年数等に基づく補正を行った上で１０で割った余りの整数値を求め，この整数値を，干支を示す数値として返す関数を，if文と算術演算子を用いて表現するものである。干支を示す数値を求めるために，上記の計算や処理を行うことは，その際にZellerの公式を応用することも含めてアイデアであるといえる。**同プログラムは，上記アイデアを実現するための解法を，計算式によりそのまま記述したもの**であり，その長さも極めて短いものであるから，作成者の個性が発揮された表現とはいえない。

検討

- 一般に，APIは，外部プログラムの呼び出しや，プログラム間のデータの受け渡しについて定めるものであって「規約」（プログラム言語の用法についての特別の約束）であるといえる
- 本件APIは，その規約を具体的なプログラムの形式で記述しており（Declaring Code），プログラムの一部でもあるから，プログラムの著作物としての保護が完全に排除されるわけではない
- しかし，個々のAPIはアイデアと表現がほぼ1対1に対応するものであり，要保護性は高くない

APIの具体例



- パッケージ> クラス> メソッドという階層構造（SSO）は保護対象ではない
- ２つの整数値のうち，大きい方を返すというメソッドの宣言部分は，普通は
  ”int max(int x, int y)”
  となり，アイデア＝表現に近い

関数の呼び出し側に関する判断

参考

- 知財高判平26.3.12（ディスクパブリッシャー事件）
  - 第三者が用意した関数の呼び出し部分が共通したという事例
  - 「IMPLEMENT_DYNAMIC 命令について本件プログラムの記述は，「IMPLEMENT_DYNAMIC（CjobsListPanel，CPanel）」であり，被控訴人プログラムの記述は，「IMPLEMENT_DYNAMIC（CProjectListBox，CBox）」である。
本件プログラムと被控訴人プログラムとの共通部分は，「IMPLEMENT_DYNAMIC 命令」を使用している部分であるが，証拠によれば，**「IMPLEMENT_DYNAMIC命令」は，マイクロソフト社があらかじめ用意している関数であるから，当該関数を使用していることをもって，当該表現に創作性を認めることはできない**。
控訴人は，“CJobListPanel や”CTaskListPanel”の各クラスの上位クラスとして“CPanel”クラスを用意し，共通機能を“CPanel”クラスにまとめ，“CJobListPanel や”CTaskListPanel”を派生クラスと定義することにより，同じ機能を実現するに当たり，記述の重複やコーディング量を減らす等の表現の工夫をしているなどと主張する。
しかしながら，控訴人の上記主張は，「IMPLEMENT_DYNAMIC 命令」の一般的な機能を用いたプログラム作成上の工夫（アイデア）を説明するものにすぎず，上記共通部分に係る創作性について具体的に主張するものではない。
また，控訴人は，「IMPLEMENT_DYNAMIC」は，いわゆるオブジェクトクラスの承継命令であり，クラス毎にコーディングしていくＣ＋＋言語における構造部分にも当たる重要な箇所であるところ，被控訴人プログラムと本件プログラムのソースコードが同じ場所で同じように承継命令を記載しているのは，クラス分けの仕方がほぼ同じであるからにほかならないなどと主張するが，**クラス分け（分類と整理）の仕方は，プログラムのアイデア又は解法に当たるもの**であって，プログラムの表現上の創作性を認める根拠となるものではない。」

©Masahiro Ito

部分的な考察と全体的な考察

質的な基準と量的な基準

ひとつひとつのAPIの創作性を判断するのが適切か

- 知財高判平24.8.8（釣りゲーム事件）
  - ➢第１審原告は，個々の要素がそれぞれバラバラでは表現上の創作性を有しない場合でも，複数の要素が全体として表現上の創作性を有することがあるから，一つのまとまりのある著作物を個々の構成部分に分解して，パーツに分けて創作性の有無や，アイデアか表現かを判断することは妥当ではないと主張する。
    しかしながら，著作物の創作的表現は，様々な創作的要素が集積して成り立っているものであるから，原告作品と被告作品の共通部分が表現といえるか否か，また表現上の創作性を有するか否かを判断する際に，**その構成要素を分析し，それぞれについて，表現といえるか否か，また表現上の創作性を有するか否かを検討することは，有益であり，かつ必要なこと**であって，その上で，作品全体又は侵害が主張されている部分全体について，表現といえるか否か，また表現上の創作性を有するか否かを判断することは，正当な判断手法ということができる。

原則的な考え方（質的基準）

- 個々の表現について創作性の有無を判断するもの
- 前掲東京地判平15.1.31
  - ➢プログラム相互の同一性等を検討する際にも，プログラム表現には上記のような特性が存在することを考慮するならば，**プログラムの具体的記述の中で，創作性が認められる部分を対比することにより，実質的に同一であるか否か，あるいは，創作的な特徴部分を直接感得することができるか否かの観点から判断すべき**であって，単にプログラムの全体の手順や構成が類似しているか否かという観点から判断すべきではない

※質的基準，量的基準という考え方は，田村善之『裁判例にみるプログラムの著作物の保護範囲の確定手法』知財管理Vol.65 No.10 1305頁，同Vol.66 No.11 1480頁(2015)に倣った。

膨大な量がコピーされている場合

- 大量のプログラムには何らかの著作物性があるという推定
- 東京地判平23.1.28（増田足事件）
  - ➢原告プログラムは，上記アのとおり，株価チャート分析のための多様な機能を実現するものであり，**膨大な量のソースコードからなり，そこに含まれる関数も多数にのぼるものであって，原告プログラムを全体としてみれば，そこに含まれる指令の組合せには多様な可能性があり得るはずである**から，特段の事情がない限りは，原告プログラムにおける具体的記述をもって，誰が作成しても同一になるものであるとか，あるいは，ごくありふれたものであるなどとして，作成者の個性が発揮されていないものと断ずることは困難ということができる。
  - ➢約300の関数のうち，103の関数は全く同一で，148の関数は処理手順同一

©Masahiro Ito

膨大な量がコピーされている場合（承前）

- 東京地判平23.5.26（おまかせ君事件。控訴審判決あり。）
  - ➢測量用のプログラムという機能を達成するためには，単純に，機能ごとに処理式を表現すれば足りるにもかかわらず，原告プログラムは，上記のとおり，共通化できる部分を選択し，これらを抽出して１つのファイルにまとめている。**これらのサブルーチンを各ファイル中のどの処理ステップ部分から切り出してサブルーチン化するのか，その際に，引数として，どのような型の変数をいくつ用いるか，あるいは，いずれかのシステム変数で値を引き渡すのか，などの選択には，多様な選択肢があり得るはず**であるから，この点にも，プログラム作成者の個性が表れているといえる。
  - ➢36個のファイルが全く同一

※「引数」とは，max(a, b)におけるaやbなど，APIを介してやり取りされるデータのことをいう。

©Masahiro Ito

検討

- 個々のAPIの仕様はアイデアに過ぎないかありふれた表現
  - ➢メソッド名をMath.max以外にも”Math.maximum”,”Calc.larger”などの選択の幅があることは創作性を基礎づけるか
- 7000以上ものDeclaring Codeが複製されたということは，その中には創作性があるものが含まれていたのではないか
  - ➢メソッドの中にはmaxやsin, cosのような単純なものだけではない

プログラムの著作物性　＝　Σ 創作性（個々のAPI）
or
max（創作性（個々のAPI））

©Masahiro Ito

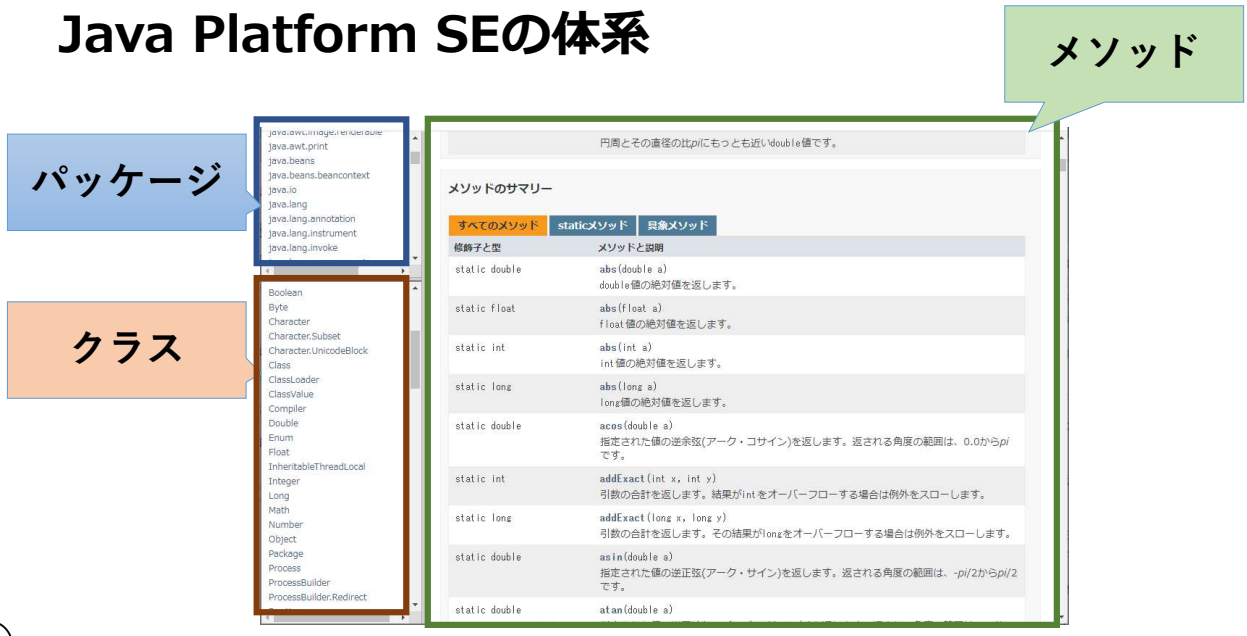
編集著作物

APIを著作権法で保護する手段はないか

編集著作物（12条1項）

- 編集物（略）でその**素材の選択又は配列によつて創作性を有するもの**は、著作物として保護する
- 素材（個々のAPI，関数など）の選択・配列の編集方針によって創作性が判断される
  - ➢独自の選択を受け容れる余地がどの程度あるか（質的評価），どの程度の量の選択を行っているか（量的評価）とを相関的に評価するという考え方
  - ➢情報収集・選択に要する投資を保護するものではない
- 素材の選択または配列の創作性が利用された場合にのみ侵害となる（Googleが利用したAPIの部分について創作性の有無が問題となる）

Java Platform SEの体系



検討

- java.lang.mathの各メソッドは，「しらみつぶし」「他の言語でも用意されているような関数」という印象があり，ただちに「素材の選択」に創作性があるとは言い難い
- パッケージ，クラスによる整理も一般的であり「素材の配列」にも直ちに創作性があるとは言い難い
- しかし，7000以上ものメソッドをコピーしており，その「選択」には多様な可能性が認められうるため，全体として編集著作物になり得るのではないか（個々の素材であるAPIの著作物性は不要）

権利制限規定による利用可能性

30条の4（非享受利用）

30条の4

（著作物に表現された思想又は感情の享受を目的としない利用）

第三十条の四　著作物は、次に掲げる場合その他の**当該著作物に表現された思想又は感情を自ら享受し又は他人に享受させることを目的としない場合**には、その必要と認められる限度において、いずれの方法によるかを問わず、利用することができる。ただし、**当該著作物の種類及び用途並びに当該利用の態様に照らし著作権者の利益を不当に害することとなる場合**は、この限りでない。

一　二　（略）

三　前二号に掲げる場合のほか、著作物の表現についての人の知覚による認識を伴うことなく当該著作物を電子計算機による情報処理の過程における利用その他の利用（**プログラムの著作物にあつては、当該著作物の電子計算機における実行を除く。**）に供する場合

©Masahiro Ito

30条の4における「プログラム」の扱い

- リバースエンジニアリングが非享受利用になるとの文脈にて
  - プログラムの著作物は表現と機能の複合的性格を有しており，プログラムの著作物に「表現された思想又は感情」とは当該プログラムの機能を意味するものと考えられるところ，その「表現された思想又は感情」の**「享受」に該当するか否かは，当該プログラムを実行等することを通じて，その機能に関する効用を得ることに向けられた行為であるか**という観点から判断されるものと考えられる。

文化庁『デジタル化・ネットワーク化の進展に対応した柔軟な権利制限規定に関する基本的な考え方』（2019）39頁

©Masahiro Ito

権利制限規定による利用可能性

32条1項（引用）

CITY LIGHTS LAW

32条1項

（引用）

第三十二条　公表された著作物は、引用して利用することができる。この場合において、その引用は、**公正な慣行に合致するもの**であり、かつ、報道、批評、研究その他の**引用の目的上正当な範囲内**で行なわれるものでなければならない。

©Masahiro Ito

SLN SOFTIC LAW NEWS

一般財団法人ソフトウェア情報センター
発行・編集　専務理事　光主清範

No. 138(2014/8)

ORACLE AMERICA, INC v. GOOGLE INC
米連邦控訴審裁判所（CAFC）2014 年 5 月 9 日判決[1]
～アプリケーションプログラミングインターフェースの著作物性が肯定された事例～

弁護士　石新智規

◆ 目 次

１　はじめに

近時、米国やヨーロッパにおいて著作権リフォームが注目されている。インターネット時代の本格化した2000年以降に普及したデジタル技術と著作権の対立の止揚が求められている時代が到来していると言えそうである。

[1] Oracle Am., Inc. v. Google Inc.,750 F.3d 1339

SOFTIC
© 2014 (一財)ソフトウェア情報センター
〒105-0001 東京都港区虎ノ門 5-1-4 東都ビル 4 階
TEL. (03)3437-3071　FAX. (03)3437-3398
WebSite　http://www.softic.or.jp/

歴史的にみて、技術と著作権の対立は、著作権という法概念（又は権利）の誕生以来の宿命であり、米国では、ビデオカセットテープへのテレビ番組の複製がフェアユースかが争われた 1984 年のソニー判決以降、数多くの著作権裁判においてテクノロジーと著作権の調和が論点とされてきた。

特に、ソニー判決後の 1980 年代後半から 90 年代は、Whelan 判決（1986）[2]がプログラムの文字的(literal)要素であるコードだけではなく、非文字的（nonliteral）な要素であるコンピュータプログラムの構造（Structure）、シークエンス（Sequence）及び組織（Organization）（以下、「SSO」と略す。）にも広範に著作権の保護を及ぼしたことを契機に、コンピュータプログラム著作権の保護範囲を巡る議論が盛んに行われた時期であった。

SSO に対しても著作権保護を及ぼす Whelan 判決はその後厳しい批判に晒され、その批判を受け入れる形で、Altai 判決（1992）は、Whelan 判決とは異なる、非文字的要素の保護範囲を限定的に解釈する基準を採用し、その基準は多くの支持を得た。[3]

さらに、Accolade 判決（1992）[4]は、Sega 製のゲームコンソール（Game Console）上で動作するゲームカートリッジ（Game Cartridge）を製作する過程で、コンソール・カートリッジ間のインターフェースの機能を理解するために、Sega のプログラムをリバースエンジニリングすることをフェアユースとして許容した。こうした流れの中で、ソフトウェアの著作権による保護は相当程度厳格に解釈されるようになり、SSO に法的保護を求める者は特許権による保護を求めるべきとの思潮が一般的なものとなっていた。

ところが、本判決は、すでに定着したと考えられていた裁判例の動向を踏まえ API（アプリケーションインターフェース、以下「API」という。）の SSO の著作物性を否定した原判決[5]の判断[6]を覆したため、驚きをもって受け止められている。[7]

２　事件の経緯

2005 年、Google は、携帯電話用のプラットフォーム構築を計画し、そのプラットフォーム構築のためにオープンソースの Java のライセンスを著作権者である Sun Microsystem と協議した。しかし、ライセンスの妥結には至らず[8]、Google は独自のプラットフォーム構築

---

[2] 【日本・アメリカ】コンピュータ・著作権法（デニス＝カージャラ・椙山敬士共著、日本評論社（1989）、「カージャラ＝椙山」）168 頁以下

[3] Dennis S. Karjala, A Coherent Theory For The Copyright Protection of Computer Software and Recent Judicial Interpretations, 66 U. Chi. L. Rev.53, 83 は、Altai 判決「に続くほとんど全ての裁判所がその（Altai 判決の）理由づけと分析手法を受け入れ、従おうとしている」と評価する。

[4] *Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1525 (9th Cir. 1992)*

[5] Appeals from the United States District Court for the Northern District of California in No. 10-CV-3561, Judge William H. Alsup.
Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974, 2012 U.S. Dist. LEXIS 75896 (N.D. Cal., 2012)

[6] 著作物性に関する原決定（CAFC が Copyrightablity Decision と呼ぶ決定）は、Whelan 判決から Accolade 判決にいたる過去の裁判例を第 9 巡回区の裁判例に限らず広く検討し、結論を導いている。

[7] Altai @ 21: Software Copyrights Revisited,　http://www.law.berkeley.edu/15775.htm　本判決前から、果たして、CAFC（連邦巡回区控訴裁判所）が回帰的にプログラムの SSO を保護するのかどうかに注目が集まっていた。

[8] 交渉がまとまらなかったのは、Google が Android プラットフォームに Java virtual machine やその他の Java プログラムと互換性を持たせることを拒絶したためである。Sun の“write once, run anywhere”のポリシーが Google の姿勢と合致しなかったようである。

に乗り出した。Google は最終的に Java 言語を利用することに決め、168 の API を開発したが、そのうち 37 は Java API パッケージを実装（implementation）したものであった。

なお、API とは、一般的なコンピュータの機能を実行させるプログラムを事前に書いておき、それをいくつかにまとめてパッケージ化したものである。API を利用することによりプログラマーは、一定の機能を実現するために最初からプログラムを構築する必要がなく、一定の機能を実現するために事前に用意されたプログラムを自らのプログラムに流用することができる。いわばプログラミングのショートカット（近道）をプログラマーに提供するものである。

Google は 2007 年、Android プラットフォームをリリースし[9]、翌年、Android ベースの携帯電話が発売され、300 万台以上の Android 携帯電話が販売された。

2010 年、Sun を買収した Oracle は、Google に対して、カリフォルニア連邦地方裁判所にて著作権侵害に基づく訴訟を提起した（当初は特許権侵害も主張していたが[10]、その主張は陪審によって否定され、その点について Oracle は不服申立てしていない）。

Google は上記 37 の Java API パッケージを実装する際に、37 のパッケージ、その下に連なる 600 を超える Class、さらにその下の 6000 を超える methods の組織化された体系を全て複製したが（7000 の declaring code の複製を含む）、実装プログラムについては、①rangeCheck 機能と②8 つの逆コンパイルされたセキュリティファイル[11]を除き、Google が自ら書いていることについて当事者間に争いはない。

2012 年 4 月 16 日に審理が開始され、Java API の著作物性、Google のフェアユースその他の抗弁に関して 24 人の証人尋問が行われた。2012 年 5 月 7 日、陪審は、Google による 37 の Java API と rangeCheck の複製を認め、8 つのセキュリティファイルについては非侵害としたが[12]、フェアユースの抗弁については意見が分かれた。[13]

その後、2012 年 5 月 31 日、カリフォルニア連邦地裁は、Java API のパッケージの SSO には著作物性が認められないと判断し、6 月 20 日、①rangeCheck 機能と②8 つの逆コンパイルされたセキュリティファイルの侵害の点を除き、Google の主張（著作権侵害不成立）を認める最終的な判断をした。その判断を不服として Oracle と Google 双方が控訴したのが本件である。[14]

なお、前述のとおり提訴はカリフォルニアであるが、特許権侵害も主張されているので、その不服申し立てについて CAFC が管轄を持つが、その準拠すべき判例法は第 9 巡回区控訴審裁判所の判例法である[15]。[16]

---

[9] Google は Java 言語を用いたものの、最終的に Android プラットフォームは、Java プラットフォームとは互換性を有さず、一方のプラットフォームで動作するソフトウェアは他方では動作しない。

[10] そのため、著作物性の判断に対する上訴の管轄が、カリフォルニアを管轄する第 9 巡回区控訴審裁判所ではなく、CAFC となった。

[11] Google はこの部分の複製は認めたうえで、de minimis の抗弁を主張した。

[12] この非侵害の認定について連邦地裁は Oracle の異議を認めた。

[13]裁判所は陪審員に対して、API の著作物性を前提とした検討をするように指示（instruction）を与え、著作物性については別途裁判所が検討を進めた。

[14] Google の控訴は、rangeCheck とセキュリティファイルの侵害を認めた点についてであるが、本件の中心争点ではないので、この点は省略する。

[15] この点で、第 9 巡回区の判例法といえる *Sega Enterprises Ltd. v. Accolade, Inc., 977 F2d. 1510(9th Cir.1992)* の先例としての位置づけ（CAFC はこの判決をフェアユース判例であると位置づけ、著作物性が唯一の争

３　論点

コンピュータプログラムの著作権法上の保護が及ぶ範囲はどこまでか（102 条(a)(b)の解釈）。保護される部分か否かの基準はどのように判断するのかが論点である。

米国著作権法上、コンピュータプログラムは保護を受ける。同法は、107 条で「コンピュータプログラム」とは「ある一定の結果を生じさせるためにコンピュータにおいて直接的又は間接的に用いられる言明又は指示の組合せ」（A “computer program” is a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result）であると定義しているため、言語の著作物（literary work）として保護される。[17][18]

しかし、コンピュータプログラムのどこまでが保護されるのか条文上明確ではない。

37 の API を実装する際に Google が複製した、Java API パッケージのパッケージ、class 及び methods の名称とそれが組織化された体系は、冒頭で言及したプログラムの SSO（Structure、Sequence と Organization）であるところ、Oracle の主張は、Google が Oracle の許諾なく 37 の Java API の declaring code と SSO を無断で複製したことが著作権侵害であるというものであった。

連邦地裁は、Java API の SSO に創作性・独創性があることを認めながらも、コマンド構造・システム・動作方法（事前に付与された一定の機能を実行するコマンドの階層構造）、すなわち Java API の SSO は、著作権法 102 条(b)を理由に著作権による保護を受けないと判断した。

４　CAFC の判断

（１）Java API の著作物性について

CAFC は、連邦地裁の判断を覆し、Java API の SSO の著作物性を肯定し、Google のフェアユース抗弁についてさらに審理をするように地裁に差し戻した。

CAFC は、コンピュータプログラムの著作権の保護範囲について詳細に議論を展開しつつ概要、以下のような点から、Java API の著作物性（declaring code という literal[文字的]な部分だけではなく、SSO という nonliteral[非文字的]な部分についても）を認める Oracle の主張を支持した。

点である本件において重要視していないように思われる。）も重要な意味を持つ。SSO の著作物性を否定するために Google の援用する他の裁判所の判断に対する CAFC の評価（特に Lotus 判決の軽視）に影響があった。なお、CAFC は、Lotus 判決を引用した先例を有する。

[16] 先例として挙げられているのは、CAFC が第 9 巡回区の判例法の解釈を行った *Atari Games Corp. v. Nintendo of Am., Inc., 975 F.2d 832, 838 (Fed. Cir. 1992)*である。手続法に明文でその旨を定める規定はなく、判例法としてそのように解釈されている。判決は、当事者も異議を述べていないことについても言及している。

[17] Article 102(a), “Copyright protection subsists, in accordance with this title, in original works of authorship fixed in any tangible medium of expression, now known or later developed, from which they can be perceived, reproduced, or otherwise communicated, either directly or with the aid of a machine or device. Works of authorship include the following categories:
(1) literary works;…” 102 条は著作物の列挙規定である。

[18] *See Atari Games Corp. v. Nintendo of Am., Inc., 975 F.2d 832, 838 (Fed. Cir. 1992)* ("As literary works, copyright protection extends to computer programs.").

・Baker v. Selden[19]判決（以下「Baker 判決」」という。）と米国著作権法 102 条(a)(b)項の意義について

著作物性の判断において CAFC は、まず、102 条(a)において最小限の創作性が必要とされるとともに、同 102 条(b)において、著作権法の保護はアイデアではなく、表現にだけ及ぶという原則を確認する。特に、102 条(b)の規定をアイデア／表現二分論を明文化したもの（逆にいえば、それ以上の意味を持たない確認規定である）と理解する。

“In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work.[20]”

（著作物に対する著作権による保護は、それが表現、説明、描写又は具現化される形態に関係なく、いかなるアイデア、手続、プロセス、システム、動作方法、概念、原則又は発見に対しても及ばない。）

そして、この 102 条(b)は Baker 判決によって確立された原則を法文化したものであることを指摘し、Baker 判決が、①著作権保護が表現にのみ及び、アイデア、システム、プロセスには及ばないこと、②機能と不可分なコンピュータプログラムの要素は著作権による保護を受けないという原則の法源となっているという。

Google は、問題となっている Java API に 102 条(a)で要求される創作性があることを認める一方、102 条(b)によって、それが機能的なものである場合には著作権による保護を認めないのが Baker 判決であり、102 条（b）であるという立場を採ったが、CAFC は、Google の主張を排斥し、次のように述べた。

Section 102(b) "in no way enlarges or contracts the scope of copyright protection" and that its "purpose is to restate . . . that the basic dichotomy between expression and idea remains unchanged." *Feist, 499 U.S. at 356* (quoting H.R. Rep. No. 1476, 94th Cong., 2d Sess. 54, *reprinted in* 1976 U.S.C.C.A.N. 5659, 5670). "*Section 102(b)* does not extinguish the protection accorded a particular expression of an idea merely because that expression is embodied in a method of operation." *Mitel, Inc. v. Iqtel, Inc., 124 F.3d 1366, 1372 (10th Cir. 1997).*

102 条(b)は、「著作権保護の範囲を拡大ないし制約するものではなく」、そしてその「目的は、アイデアと表現の基本的な区分は変わらないままであることを再度述べること」（引用省略）である。102 条(b)は、表現が動作方法（a method of operation）に具現化されているというだけの理由で、あるアイデアの特定の表現に付与される著作権保護を消滅させるものではない（引用省略）。

他方で、アイデアと表現とを区別する作業が容易ではなく、裁判所の適用する基準が統一していないことを指摘、**Whelan 判決**（3d Cir.1986、「ある作品の目的又は機能に必要とはされないものは全て表現である」）と **Lotus 判決**（9th Cir. 1992、「動作方法（method of

---

[19] *Baker v. Selden*、*101U.S. 99, 101(1879)* 会計士の Charles Selden は、会計帳簿の独特のシステムの使い方を説明する簡単文章と 20 頁のフォームから成る書籍を発行したが、その死後、その酷似するシステムを発表した Baker に対し、遺族が、書籍及び書籍で示されているシステムの著作権に基づき著作権侵害訴訟を提起したものである。本件については、カージャラ＝椙山 231 頁以下も参照。Baker 判決は、102 条(b)の意義を理解するうえで極めて重要なものである。

[20] See, *Golan v. Holder, 132 S. Ct. 873, 890, 181 L. Ed. 2d 835 (2012)* ("The idea/expression dichotomy is codified at 17 U.S.C. § 102(b).").

operation）は、それによって、ユーザーが何かを動作させる手段であり、その動作を実行するために用いられる言語は、著作権による保護を受けない表現である」）を対照的な（両極端の）裁判例として挙げつつ、第 9 巡回区裁判所の採る判例法は、そのいずれでもなく、**Altai 判決**の **Abstraction-Filtration-Comparison** テスト（以下「AFC テスト」という。）[21]を採用したという。　AFC テストは次の 3 段階からなるテストである。

（i）抽象化ステップ：侵害されたプログラムを構成要素に細分化する。

(ii) ろ過ステップ：全ての保護されない部分（アイデアだけでなく、あるアイデアと必然的に同一のものとなる表現も）を除外する。[22][23]

(iii) 比較ステップ：ろ過後に残った創作的表現と侵害プログラムと主張されているプログラムの比較を行う。

この AFC テストは、ある機能を実行する対象が動作方法（method of operation）であることをもって著作物性を否定するという Lotus 判決の概念を拒絶する（つまり、「動作方法」と分類される場合でも、それにもかかわらず著作物性が肯定される場合もあるという考え方を採る）一方で、同テストは、いったんあるアイデアを分離できれば、その他を全て表現と扱う Whelan 判決の仮定も拒絶するものだと CAFC は論じる。

そのうえで CAFC は、連邦地裁が AFC テストに言及しながら、AFC テストを適用しようとせず、102 条(b)に基づき、形態を問わず、機能性のゆえに Java API が著作権による保護を受けないと判断したことを批判し、AFC テストを適用した。

① Declaring Code に対するあてはめ

（A）Merger 理論

前述のとおり、Google による Declaring Code の複製及び同 Code の創作性の点については争いがなかったものの、地裁は、102 条(b)（同条に具現化されている Merger【アイデアと表現が混同（一体化）するため、著作権による保護を否定する】理論と短いフレーズ法理）に基づき、著作物性を否定した。すなわち、地裁は、Java ルールに従う以上、プログラマーは同じ機能を指定する方法を示すために、同一の declaration または方法の header lines を利用しなければならないとし、Java パッケージのそれぞれに対する declaring code を書く方法は一つしかない、よって、Merger 理論により、著作権侵害は成立しないと判断した。

---

[21] *Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1525 (9th Cir. 1992)* ("In our view, in light of the essentially utilitarian nature of computer programs, the Second Circuit's approach is an appropriate one."[コンピュータプログラムの本質的に実用的な性質に照らし、第 2 巡回区控訴審裁判所のアプローチは適切なものであると考える]). 詳細については、SLN38 号（1992.8.21）参照。

[22] より具体的には、原決定 20－21 頁、Altai, at 709-710, (1)特定のプログラムが動作することが意図されているコンピュータの機械的な仕様、(2)他のプログラムとの互換性の必要性、(3)コンピュータ製造業者の標準設計、(4)業界の需要、(5)コンピュータ産業において広く受け入れられているプログラミングの実務といった要素が挙げられている。

[23] Altai 判決のフィルタリングテストは、Melville Nimmer 教授の息子である David Nimmer の提唱する基準に由来する。See, 3 Nimmer on Copyright§13.03[F][3], at 13-65, David Nimmer et al., A Structured Approach to Analyzing the Substantial Similarity of Computer Software in Copyright Infringement Cases, 20 ARIZ, ST. L.J.625 (1988).

しかし、CAFC は、アイデアと表現の Merger を否定し、著作物性を肯定した。[24]

(a) 証拠によれば、Google が複製した 7000 ラインの選択と配列には無制限の選択があり得た。例えば、“java.lang.Math.max”という code の名称は、“Math.maximum”、“Arith.larger”などでもよく、そもそも予定されたものなどなく、そこに選択の余地があった。

(b) 地裁は、Google が複製する時点で他の選択をする余地があったか否かを検討しているが、著作物性の問題は、複製時ではなく、API パッケージの創作時の選択可能性を検討しなければならない。この点、前述のとおり、Sun（当時）は、様々な呼称が可能な中で、“java.lang.Math.max”という名称で呼ぶことを選択した。

（B）短いフレーズ（Short Phrase） 法理

地裁は、Android プラットフォームにおける名称の選択可能性を認めるものの、名前であるとか短いフレーズには著作権の保護は及ばないと判断した（Short Phrase 法理）。

CAFC は、問題となるのはフレーズの短さではなく、当該フレーズが創作的なものであるか否かであるかであるとして、前述のとおり、declaring code の記述や API パッケージには創作性があるとした。また、地裁が個々の line に細分化して短いフレーズとして著作物性を検討していること自体も適切ではなく、Oracle の主張は、細分化された個々の短いフレーズそれ自体ではなく、（いわば line の集合体としての）編集著作物としての全体の保護であると指摘した。

（C）Scenes a Faire 法理

Google は、Scenes a Faire 法理（あるテーマや状況について、ありふれた、標準的な表現であるという場合に著作物性を否定する法理）に基づき、declaring code の著作物性が否定される旨も主張していた。地裁では、ありふれた表現であることの立証がされていないと判断されていたが、CAFC は、さらに、第 9 巡回区控訴審裁判所の判例法においては、Scenes a Faire 法理は著作物性の判断要素ではなく、抗弁であると位置づけているので、仮にそのような法理を認めるとしても、API の著作物性を否定する理由にはならないとした。

また、Merger 理論の適用時と同様、この法理が問題となるのは Java API の複製時ではなく、創作時に創作者が置かれた状況が問題とされるべきであり、創作時に Sun のプログラマーによる Class、Method 及び Code の選択を指図する外部的要因の有無を問うべきだが、そのような事情を Google が立証していないと結論づけた。

② API パッケージの SSO について

（A）連邦地裁の 102(b)の解釈―Lotus 判決への依拠[25]

連邦地裁は、Java API パッケージの SSO が創作的かつ独自性のあるものであると判断したが、しかし、それが米国著作権法 102 条(b)の「システム又は動作方法であり…それゆえ著作権保護を受けられない」と判断した。その結論に到達するうえで、連邦地裁は、

[24] 但し、CAFC は、2 つのパッケージについて Merger 理論の適用の可能性を示唆したが、Google がその点を主張していないと指摘した。

[25] Pamela Samuelson, “Legally Speaking, Are APIs Copyrightable? ” (Vol.25. No11. Communication of the ACM, 25), available at http://www.aipla.org/committees/committee_pages/Copyright-Law/Committee%20Documents/Oracle%20v.%20Google%20-%20AIPLA%20Primer%20on%20APIs/Samuelson%20Article.pdf

Lotus 判決[26]（第 1 巡回区控訴審裁判所）に大きく依拠した。すなわち、同判決は、コマンドのヒエラルキー（階層）は、102 条(b)の「動作方法」（method of operation）であるため、著作権による保護を受けないと判断し、さらに、特定の言葉がある動作に不可欠なものである場合には、その言葉が「動作方法」の一部になるため、それ自体が著作権の保護の対象となり得ないとも判示した。

（B）Oracle の主張

Oracle は､本件と Lotus 判決(の前提事実)との違いを指摘して地裁の判断を批判した。

（a） Lotus 判決で被告 Borland は、Google とは異なり Code を複製していないので、事実関係がそもそも異なる。

(b) Lotus 判決は、問題となったコマンド（copy、print など）は創作性がないとしたが、本件では、declaring code、API パッケージの SSO に創作性があることに争いがない。

(c) Lotus 判決では、問題となったコマンドが動作に不可欠のものであると判断されたのに対して、本件では、3 パッケージを除き、Java 言語でプログラムを書くために、Java API パッケージを複製する必要がなかった。

（C）CAFC の判断：Oracle の主張を認める。

(a)「method of operation」に該当したら著作権保護を否定する Lotus 判決は、第 9 巡回区控訴審裁判所の判例法ではなく、また、アイデアそれ自体ではなく、アイデアを表現するものであればコンピュータプログラムの SSO に著作物性を認める第 9 巡回区の判例法にも抵触する。[27]

(b) Lotus 判決のような解釈は、「プロセスや方法の表現」[28]を著作権が保護することを認め、アイデアと表現とを区分する Altai 判決が採用した AFC テストを是認する第 9 巡回区の考え方に合致しない。

(c) そして、Lotus 判決の解釈と議会の意思との不整合を指摘する。

“If we were to accept the district court's suggestion that a computer program is uncopyrightable simply because it "carr[ies] out pre-assigned functions," no computer program is protectable. That result contradicts Congress's express intent to provide copyright protection to computer programs,…, despite their utilitarian or functional purpose.”

（コンピュータプログラムは「事前に与えられた機能を実行」というだけの理由で著作物性がないという地裁の示唆を我々が万が一受け入れたら､コンピュータプログラムは法的に保護されなくなる。その結果は、その実用的・機能的な目的にも関わらずコンピュータプログラムに著作権の保護を与えた議会の明白な意思と矛盾するものである。）

(d) CAFC は、以上のような理由に基づき、著作者が根底にあるアイデアを表現する複数の方法を持つ限り､機能的な創作物に対しても著作権を付与するものであると解釈し、Google が依拠する Lotus 判決が解釈したように、102 条(b)は機能的なコンピュータプログラムの要素に対する著作権の保護を（カテゴリカルに）否定するものではなく、伝統的なアイデア／表現の二分論を定めたに過ぎないものとした。

---

[26] *Lotus Development Corp. v. Borland International, Inc., 49 F.3d 807 (1st Cir. 1995), aff'd without opinion by equally divided court, 516 U.S. 233, 116 S. Ct. 804, 133 L. Ed. 2d 610 (1996)* 上告受理申立てが受理されたが、裁判官の意見が等しく分かれたたため、第 1 巡回区控訴審裁判所の判断が是認された。

[27] *Johnson Controls, Inc. v. Phoenix Control Sys., Inc., 886 F.2d 1173,1175-76 (9th Cir. 1989).*

[28] *Altai, 975 F 2d. at 839*

(e)　Java API の SSO により実現する機能を表現する方法は複数ある以上、SSO が機能を提供するというだけでは、102 条(b)によって著作権保護を否定されないとした。

（２）フェアユースについて

①コンピュータの互換性を達成する必要性と著作物性の関係性

連邦地裁は、Java API の SSO を「method of operation」と特徴付ける際に「コマンド構造の複製は互換性の達成のために必要である」と説明した。すなわち、pre-Android Java code を Android 上で実行するためには、Google は同一名称・同一分類・同一の機能仕様を有するコマンドシステムである Java package, class, method を提供する必要があったと判断し、このことが Java API の著作物を否定する結論に結びついた。

その結論を導くに際し、連邦地裁は第 9 巡回区裁判所の互換性に関連する裁判例である Accolade 事件と Connectix 判決に主に依拠した。[29]

この点、CAFC は、上記 2 事件の判決は、被告が著作物の機能的な側面を理解するために中間的な複製物を作ったうえで最終的に新たな著作物を作っている点で、Google が Java API をリバースエンジニアリングして、著作物性のない機能的な部分へアクセスしたのではなく、declaring code と 37 の Java API パッケージの SSO を全て複製した本件は、事案を異にするものであるとした。

そして、両事件はあくまでフェアユースの成否が問題になったものであり、API の著作物性を問題とする本件とは異なることを強調し、Google が示唆した（連邦地裁は採用しているようにも思われる）著作物性に対する「互換性の抗弁」を正面から否定した。[30]

CAFC は、互換性の必要に関する Google の利益は[31]、開発者がすでに有するベースを利用するために Java を梃子として開発プロセスを加速化することにあるとし、そのような利益は、著作物性ではなく、フェアユースの分析の中で検討されるべき問題であるとした。

②Google の Java API の複製はフェアユースか

この点について CAFC はフェアユース法理の一般論について詳述するものの、具体的な事実関係が十分に明らかになっていないとして、API の著作物性を認めたうえで、Google による複製行為がフェアユースに当たるか否かの審理に差し戻した。[32]

---

[29] In reaching this conclusion, the court relied primarily on two Ninth Circuit decisions[この結論に達するため、裁判所は主に二つの第 9 巡回区裁判所の判断に依拠した。]: *Sega Enterprises v. Accolade, Inc., 977 F.2d 1510 (9th Cir. 1992), and Sony Computer Entertainment, Inc. v. Connectix, Corp., 203 F.3d 596 (9th Cir. 2000).*

[30] Google は、主張書面において、102 条(b)に基づき、互換性に関連する要素は著作権の対象とならないという前提を認めるものとして Accolade 事件と Connectix 事件を引用している。

[31] 互換性の必要性が AFC テストの仕様を画するような外部要因として機能するのかを検討することがあることは認めるものの、CAFC は、そのような検討は「創作者」の互換性実現の必要性であって、本件のように「複製者」（後行者）の必要性ではないと指摘する。しかし、互換性を求めるのは通常後行者であり、そのような必要性の有無がフィルタリングの際の判断対象になるのであるから、CAFC の解釈は説得的とは思われない。

[32] その他、rangeCheck file と逆コンパイルされたセキュリティファイルの著作権侵害を認めた連邦地裁の判断に対し Google が de minimis 法理に基づく異議申立てを行ったが、Google の主張が退けられている。

５　コンピュータプログラムの保護範囲を巡る裁判例の動向

1980 年代から 90 年代にかけて米国においてコンピュータプログラムの著作権保護に関する議論を牽引したのは、Pamela Samuelson 教授と Dennis Karjala 教授である。[33]本判決が驚きをもって受け止められた理由を理解するためには、これまでの裁判所の動向を大雑把にも押さえておく必要がある。[34]

ここでは、主に Samuelson 教授の整理に従って、本判決でも言及されているコンピュータプログラムを巡る裁判例の動向を確認しておきたい。[35]

（１）　Franklin 事件から Paperback 事件まで

最初のソフトウェア著作権をめぐる事案は、Apple がそのオペレーティングシステムを複製した Franklin を訴えた事件（3d Cir. 1983）である。本件で Franklin は、Apple のオペレーティングシステムプログラムは、Baker v. Selden 判決と 102 条(b)により著作権保護の対象から外されると反論した。

第 3 巡回区控訴審裁判所は、102 条(b)をアイデア／表現の二分論を規定したものとして、そして Baker 判決は Merger（アイデアを表現する方法が一つ又は非常に限られた数しかない場合には、その表現に著作物性を認めない）理論を確立した裁判例として扱った。同裁判所は、その前提から、Apple のオペレーティングシステムとして同じ機能を実行するために複数のプログラムを書ける限り（その結果 Merger（混同）は否定される）、[36]プログラムはアイデアの「表現」であり、それゆえに著作物性が肯定されると判断し、Franklin の著作権侵害を認定した。

1986 年の Whelan 判決では、デンタルラボの SSO の著作物性を広く認め、異なるプログラミング言語・アルゴリズムを利用して SSO だけを複製した場合にも著作権侵害を肯定した。その際、Franklin 判決と同様、102 条(b)は、アイデア／表現の二分論を再確認するにすぎず、Baker 判決は Merger に関する裁判例であるとの立場をとった。

102 条(b)を制限的に解釈し、コンピュータプログラムの機能的側面を保護する（プログラムの複雑さとは関係がなく、一つのアイデアに基づきプログラムの全体を保護してしまう）ため、Whelan 判決に対する批判は強かった。[37]

主な争点ではないので、本稿では省略する。

[33] Google の準備書面（Brief）にも、両教授の論文が引用されている。

[34] 前掲注 6 に指摘したとおり、原決定はコンピュータプログラムの著作権保護の歴史を丁寧に遡って検証している。Samuelson, supra note 25 は、この点を高く評価する。

[35] Pamela Samuelson, Why Copyright Law Excludes Systems and Processes from the Scope of Its Protection, 85 Tex. L. Rev. 1921 (2007), at 1921

[36] カージャラ＝椙山 105 頁以下及びその原文にあたる、Dennis S. Karjala, Copyright, Computer Software, and The New Protectionism, 28 Jurimetrics J. 33（1987）を参照。

機能的著作物に対する侵害判断について、「伝統的な著作権が技術的な『表現』について著作権侵害を認定するにはほとんど文字通りの類似性（筆者注、デットコピーに近い類似性という意味であろう。）を要求しているにもかかわらず、小説や戯曲に適用されるべき表現についての広範な考え方を技術に対して誤って適用している。」とする。

[37] カージャラ＝椙山 118 頁以下を参照。

その批判の要は、①102 条(b)の下地となった Baker 判決の理解が間違っていることと、②コンピュータプログラムという機能的著作物と他の文芸的著作物との差異を全く無視していることにある。「アイデアを表現から区別するための基準を決定するにあたり、（筆者注、Whelan 判決の）裁判所は、Baker v. Selden 判決を読み違え、同判決が、目的を達成するためのさまざまな方法があることから、それぞれの方法が

しかし、Whelan 判決に対する強い批判にも関わらず、Paperback 事件[38]において、裁判所は Whelan 判決の結論に従った。

被告の Paperback は、Lotus1-2-3 のメニューコマンドシステムの構造を複製したとして著作権侵害に問われた。Paperback は、一定の場合 SSO は著作権法の保護対象となることを認めたが、メニューコマンドは「system」（システム）であり、102 条(b)によって著作権法による保護の対象から外れる旨を主張した。

裁判所は、Whelan 判決と同様、102 条(b)と Baker 判決は、抽象的アイデアと保護される表現とを区別する規定であるとし、他の手段の存在を SSO が著作権法によって保護されるものかどうかを判断するための鍵となる要素であると判断した。すわなち、「（Lotus）の一つのメニュー構造の特定の表現は、電子的なスプレッドシートに不可欠なものでも、スプレッドシートに関する一つのメニュー構造の少し抽象的なアイデアと混同（merge）するものでもない。」から、そのようなアイデアは、「文字通り無制限ではないにせよ、数多くの方法で表現することができる」と指摘し、Lotus のメニューコマンドシステムの SSO は著作権法上保護されると判断した。

（２） Altai 事件から Lotus 事件まで

この流れを大きく変えたのが Altai 判決である。

Altai は、ZEKE というスケジュールソフトを様々なオペレーティングシステムにおいて動作するように修正を施すにあたり、Computer Associates International Inc.（以下「CA」という。）の元従業員を雇用したところ、同従業員は、CA が販売していたスケジューラーの互換性を達成する ADAPTER というモジュールのソースコードの 30％を無断で複製し利用していたため（ソフト名は OSCAR）、CA が Altai に対して著作権侵害訴訟を提起したものである。

Altai は、ソースコード（文字的部分）の複製については著作権侵害を認めてコードを書き直し、新しいバージョンのスケジュールソフトを制作・販売した。しかし CA は、この新しいバージョンのソフトも実質的に CA のスケジュールソフトのパラメータリスト（モジュール間でやりとりされる情報の形態）と Macros（命令）を実質的に複製するものであり、著作権侵害であると主張した。パラメータリスト及び Macros はモジュールの構造を決定づけるもので SSO の一つであり、Whelan 判決と同様、SSO の著作物性が問われた。

Altai 判決は、プログラムの非文字的な（nonliteral）要素が著作権法上保護されることを認めながら、Whelan 判決が「ある実用的な著作物の目的又は機能はその作品のアイデアであり、その目的又は機能にとって必要ではない部分はそのアイデアの表現の部分というこ

保護された表現であるということを暗示しているとの前提を指示するものとして理解している。Baker 判決の裁判所は、帳簿を付ける他の方法があるかないかについては注意を払わず、たんにその表に著作権を認めることが原告が開発した特別のシステムを独占することになるかどうかについて注意を払った。さらに重要なことは、Whelan 事件の裁判所は、（略）製作物の質的に重要な部分を利用することは仮にその部分が量的に僅少であってもその著作権を侵害する、と判示した文芸作品に関するケースを参照した。このアプローチは、技術的製作物とその他の製作物との重要な違いを無視している。革新的な技術開発は、それが極めて重要であったとしても、著作権によっては保護されていない。」（カージャラ＝椙山 120 頁、下線は筆者）

[38] *Lotus Development Corp. v. Paperback Software International, 740 F.Supp.37 (D. Mass.1990).*

とになろう。望ましい目的を達成する手段が様々ある場合、そこから選ばれた一つの特定の手段は、その目的に必然的なものではない。よって、それは表現であり、アイデアではない」とする点を誤りであるという。すわなち、各モジュールが一つのプログラムであり、「アイデア」を持っているので、一つの全体的な目的とそのプログラムのアイデアを等置する Whelan 判決の一般的公式は、説明としても不十分なものであると指摘する。

そして Altai 判決は、前出の AFC テスト[39]に従って著作権侵害の有無を判断すべきだとし、特に、AFC のフィルタリングテストにおいて、①効率性によって決定づけられるプログラム設計の要素、②プログラムがともに機能するハードウェア・ソフトウェアといった外部要因によって制約を受けるプログラム設計、③プログラムのパブリックドメインの要素（一般化したプログラミング技術など）を除くこととする点が、Whelan の一つのアイデアを抽出したほか全てを「表現」とする基準とは全く異なる。

Altai 判決は、AFC テストのフィルタリングの結果を比較し、SSO であるパラメータリスト及び Macros について、パブリックドメイン又はプログラムの機能上の制約（それが作動する IBM オペレーティングシステムのインターフェースによる制約）に基づく類似点であるとして、侵害を認めなかった地裁の判断を支持した。

その後、コンピュータプログラムの著作権に基づくユーザーインターフェースの保護の可否が争われた Lotus 判決（1995）は、Altai 判決の AFC テストを利用せず、102 条(b)を根拠に端的に著作権侵害を否定した。

被告の Borland は、原告 Lotus の Lotus1-2-3 のメニューコマンド構造とは異なるスプレッドシート（ユーザーインターフェース）を自ら開発していたが、Lotus1-2-3 の既存利用者の移行を誘引すべく、Lotus1-2-3 で構築された Macro（命令）をそのまま Borland のスプレッドシートで利用できるようにするエミュレーション機能を付加する目的で Lotus1-2-3 のメニューコマンド構造を複製したため、Lotus が著作権侵害訴訟を提起したものである。

連邦地裁は、コマンド名称を変えることで、非常に多くのメニュー、サブメニューについて非常に多くの Lotus とは異なる選択肢があること、さらにコマンドの配列や階層化にも同様に選択肢があることから、Lotus のコマンド構造に著作物性を認めた。

これに対して第 1 巡回区控訴審裁判所は、102 条(b)及び Baker 判決に依拠し、Lotus 1-2-3 のコマンド構造は「method of operation」(動作方法)であるとして著作物性を否定し、地裁の判断を覆した。

第 1 巡回区控訴審裁判所の判断（Lotus 判決）は、102 条(b)の明文上著作権の保護対象から外す「method of operation」であるとして、Altai 判決のような AFC テストを経由せずに著作権法による保護を否定した点に特徴があり、Altai 判決のような複雑な評価を経ることなく[40]、端的に、コンピュータプログラムの著作権保護の範囲を絞るものといえよう。[4142]

---

[39] AFC テストは、文芸の著作物で確立した侵害判断の基準(抽象化テスト)をコンピュータプログラム事件に応用したものである。Karjala, supra note 3, at 79, *Nichols v. Universal Pictures Co., 45 F 2d. 119(2d Cir. 1930) (L. Hand, J.), cert. denied, 282 U.S. 902(1931)* 参照。Altai 判決は、Whelan 判決との決定的な違いは、抽象化テストが、著作物が無数のアイデアと表現が混合されて構成されている可能性を暗黙の前提としている点であるという。また、Altai 判決は、コンピュータプログラムという、その機能的性質ゆえに特別な著作物の特質を Whelan 判決よりも意識しており、「コンピュータプログラムの本質的に実用的な性質が表現からアイデアを分離する作業を更に複雑なものとする」（Altai,982 F 2d. at 704）と述べている。

[4040] Pamela Samuelson, A Fresh Look At Tests for Nonliteral Copyright Infringement, 107 Nw. U. L. Rev. 1821,

６　本件の評価

（１）Baker 判決と著作権法 102 条(b)の解釈

５で検討したとおり、コンピュータプログラムの著作権法の下での保護範囲について Whelan 判決と Lotus 判決を左右の両極としてその中間点に Altai 判決が位置付けることが可能であり、本判決も同様の理解を示している。[43]

本判決は、連邦地裁が Lotus 判決に依拠した点を批判した Oracle の主張を是認した。

主な批判は、①Lotus 判決（第 1 巡回区控訴審裁判所）は本件事件が準拠すべき第 9 巡回区控訴審裁判所の判例法ではないこと、②Lotus 判決の 102 条(b)の解釈によれば、同条(a)でコンピュータプログラムの創作性・独創性を認めながら、コンピュータプログラムが（性質上当然に）「システム」であり、「method of operation （動作方法）」であることから、同条(b)によってその保護が否定されてしまうが、かかる解釈は、コンピュータプログラムを著作権により保護することを認めた議会の意思に反する点にある。

コンピュータプログラムの保護範囲を巡る裁判例の動向は前述のように 80 年代から 90 年代にかけて大きく揺れた。その分岐点は、コンピュータプログラムという特殊な著作物を著作権法によって保護することの意味と 102 条(b)を定めた議会の意思をどのように解釈するか、という点にある。

CAFC は、Lotus 判決ではなく Altai 判決を基礎に、102 条は「システム」「動作方法」をカテゴリカルに著作物から除外したのではなく、Baker 判決で確立した Merger 理論を定めたものだとした。[44]　その上で、Java API の SSO により実現する機能を表現する方法は

1839, 1840 (2013) は、これまでの裁判例上の nonliteral な著作権侵害の判断基準を考察する中で、Altai 判決を評価しつつ、102 条(b)が明確に「手続、プロセス、システム、操作方法」を著作権の保護対象から除外しているにもかかわらず AFC テストがそれを求めなかった点を指摘し、102 条(b)の除外を AFC テストのろ過プロセスに位置づけるべきであったが、そうしなかったと述べる。

[41] Lotus はこれを不服として上告受理申立てを行った。その申立理由の一つは、102 条(b)の著作権からの除外対象とされる「システム」や「method of operation(動作方法)」を文字通りに解釈し、Lotus 判決のように著作物性を否定することが、コンピュータプログラムに著作権の保護を及ぼすものとした議会の意思を否定することなるため、裁判所はそのような解釈を採るべきではない、という点にあった。Lotus は、102 条(b)はそのような特殊な除外事由を設けたものではなく、アイデア／表現の二分論を単に条文化したものに過ぎないと主張した。なお、この上告受理申立ては受理されたものの、裁判官の意見が分かれた結果、Lotus 判決が是認された点は注 26 で述べた。

[42] Mark Lemley et al., Software and Internet Law (Third edition), at 73 は、Lotus 判決によれば、API の著作物性は完全に否定されることになるが、ほとんどの裁判所はこうした絶対的なアプローチを採っていないと評価する。

[43] 本判決は、SSO の選択肢があったことを極めて重視し、フィルタリングにおける除外事由を詳細に検討した様子がないため、Whelan 判決の復活ともいえるように思われる面もあるが、CAFC は、自らの判決を Whelan 判決ではなく Altai 判決の系譜に位置づけている。判決 22 頁以下。

[44] Oracle は、102 条(b)の列挙が特別な除外事由を定めたとする Lotus 判決の解釈が不当であると主張する際、元著作権局長の Ralph Oman の見解や Altai 判決ほかの裁判例を援用するだけではなく、現在 Google の Copyright Counsel を務める William Party 氏が（皮肉なことに）過去に（今回 Google が大きく依拠した）Lotus 判決を、全てのコンピュータプログラムを保護されない「method of operation」とするものだと批判していた（まさに本件における Oracle の主張そのものである）ことに言及している。 See, William F. Patry, Copyright and Computer Programs: It’s All in the Definitions, 14 Cardozo Arts& Ent. L.J.1, 5-8, 13, 59-63(1996),

複数ある以上、仮に SSO が機能を提供するというだけでは、102 条(b)によって著作権保護を否定されないとしたのは前述のとおりである。しかし、他の選択肢があるというだけで SSO の保護を認めるのは行き過ぎではなかろうか。ある機能を実現するために SSO の選択肢など複数あるのが通常であり、それだけでは保護される表現とアイデアとを区別できるものではないし、ほとんど全てが表現と評価されてしまう結果（その結果は Whelan のような表現部分を非常に広くとる見解と軌を一にするように思われる。）を招来する。[45]

かかる CAFC の判断は、Altai 判決の AFC テストのフィルタリングのテストの中身が不明瞭であるという弱点[46]を浮き彫りにした。実際に AFC テストの濾過テストを通過して残る SSO の創作性とはどのようなものか、想起しにくいように思われる。

今後は、再審理を求められたフェアユースの成否とともに、Lotus 判決の際に最高裁で決着のつかなかった 102 条(b)に関する CAFC の解釈が是認されるか否かも議論の的となると思われる。

この点、Samuelson 教授は、CAFC の採った 102 条の狭い（列挙されている語に特別な意義を見出さず、アイデア／表現二分論を採用したに過ぎないとする）解釈が、Melville B. Nimmer による Baker 判決[47]の誤った解釈[48]に起因することを、102 条(b)の legislative history（制定経緯）に遡って丁寧に論証し、改正法前から存在した、コンピュータプログラムを著作権法の下で保護することへの懐疑を背景に、102 条(b)があえて「システム」、「動作方法」等を除外事由と定めたものであると主張している[49]。

CAFC は、102 条(b)の列挙事由に特別な意味を見出さない立場を採用したが、その過程で立法経緯が詳細に検討された様子は伺えない。Google は当然のことながら、Baker 判決の正当な評価に依拠して 102 条(b)を解釈すべきであることを主張しているが、CAFC は 102 条(b)の立法経緯を詳細に検討することなくその主張を退けている。

なお、かかる Patry の見解については、Karjala, supra note3, at 72 footnote54 参照。

[45] Karjala, supra note 3, at 89 footnote124 参照。

[46] Altai 判決も「率直に言って、非文字的なプログラム構造に対する著作権の保護の正確な境界は完全にははっきりしていない」と明言する。また、Karjala, supra not 3, at 80, 81 は、Altai 判決のフィルタリングテストの結果、ほとんどの非文字的要素が創作的表現として重要な要素として残る余地がないのではないか、と指摘し、AFC テストの結果は、コードのデッドコピーまたはそれに近い実質的な複製のみを保護する結果となるはずだが、Altai 判決がその点を明確しなかったため、その後の Altai 判決に従おうとする裁判所がまごつく結果を生んでいると述べる。今回の CAFC による判断は Karjala 教授の懸念が顕在化した結果というべきであろう。

[47] Samuelson, supra 35, at 1928-1936,連邦最高裁は、「科学上の真実又は技術上の方法は世界共通の財産であり、いかなる著作者も自分独自の方法でそれを表現し、説明し、利用する権利を有する。」と判断し、端的に、「未記入の会計帳簿は、著作物の対象足り得ない」として Selden の主張を退けた。

[48] Lloyd L. Weinreb, Copyright For Functional Expression, 111 Harv. L. Rev. 1149, 1173 (1998) は、この判断は「システム又は実用性のある技芸（art）は、それがどのような方法で表現されていようとも、著作物の対象足り得ない、という簡潔かつわかりやすいものである」と評価する（下線は筆者）。これに対して Melville Nimmer は、Baker 判決から著作物性を判断するための基準を見出すことに反対し、Baker 判決によって、いかなる著作物の著作物性も否定されてはならないとした。Nimmer は、あるシステムの表現がシステムそれ自体と区別できる以上、著作権の対象であると主張した。しかし、かかる Nimmer の解釈は、Baker 判決が、システムの表現方法が他にあるか否かさえ検討せずに著作権による保護を否定していることと整合しない。

[49] Samuelson, supra 35, at 1953-1961,

よって、仮に本件が最高裁で審理されることになれば、この点が詳細に争われる可能性も相当程度あると思われる。[5051]

（2）コンピュータプログラムの法的保護の在り方

CAFC は、判決の中で、独立して「Google' Policy-Based Argument」という項目を立て、Google がその立場をサポートするものとして大きく依拠した Samuelson 教授を批判的に採り上げている。ソフトウェアの著作権による保護は制限し、むしろ特許による保護を求めるべきであるという教授の見解[52]に疑問を投げかけ、むしろ著作権法による保護はもっともソフトウェアに適したものであり[53]、かつ、著作権の保護と特許法の保護が重複することは連邦最高裁も認めているところであるとし[54]、議会又は最高裁が立場を変えない以上、ソフトウェアプログラムの保護は特許法に限られるべきであるという主張は支持できないと結論づけている。[55]

Samuelson 教授は、102 条(b)の Nimmer による解釈が誤っていることを指摘し、Baker 判決の正当な理解と 102 条(b)を定めた議会の意思からすれば、コンピュータプログラムの著作権による保護は謙抑的であるべきとすべきもので、CAFC の指摘は、同教授が立法論としてそのような主張をしているかのごとき理解をするもので、同教授の主張を誤解するものと言わざるをえない。

---

[50] なお、102 条(b)をアイデア／表現二分論の確認にすぎないという Nimmer の解釈は、Lotus 事件の原審が採用する立場であり、その解釈には次のような批判が加えられている。そのような解釈に従えば「Lotus が行ったように、Selden がコンピュータプログラムによって帳簿システムを実行していたら、原判決の分析の下では、（筆者注、Baker 判決の判示内容とは全く異なり）その『規則に従った線と会計項目』及びシステムの体系化の方法を擁護する権利を与えられることになろう。」が、それは、Baker 判決を無視するものと批判する。See, Dennis S. Karjala and Peter S. Menell, Applying Fundamental Copyright Principles to Lotus Development Corp. v. Borland International, Inc., 10 Berkeley Tech. L. J 177, 189 (1995)

[51] Samuelson 教授の私信によれば、今秋にも上告受理申立ての可能性があるようである。ちなみに、Copyright minimalist （著作権の強化に懐疑的な論者）と評価されている Justice Stephan Breyer は、tenure を取得した論文において、著作権によるコンピュータプログラム保護に懐疑的な姿勢を見せている。Stephen Breyer, The Uneasy Case for Copyright: A Study of Copyright in Books, Photocopies, and Computer Programs, 84 Harv. L. Rev. 281(1970).

[52] See also, Dennis S. Karjala, Distinguishing Patent and Copyright Subject Matter, 35 Conn. L. Rev. 439(2002).

[53] CAFC は、ソフトウェアの特許による保護に疑問を呈する見解として、次の 2 つの記事を紹介している。*See* Technology Quarterly, *Stalking Trolls*, ECONOMIST, Mar. 8, 2014, http://www.economist.com/news/technology-quarterly/21598321-intellectual-property-after-being-blamed-stymying-innovation-america-vague;Timothy B. Lee, *Will the Supreme Court save us from software patents?*, WASH. POST, Feb. 26, 2014, 1:13 PM,http://www.washingtonpost.com/blogs/the-switch/wp/2014/02/26/will-the-supreme-court-save-us-from-software-patents/.

[54] *Mazer v. Stein, 347 U.S. 201, 217, 74 S. Ct. 460, 98 L. Ed. 630, 1954 Dec. Comm'r Pat. 308 (1954)*.

[55] Oracle は、Google の依拠する Samuelson 教授がコンピュータプログラムを著作権法で保護するのは間違いであると主張した論文に言及し、Google の主張が立法論である（議会の意図とは別のことを言っている）かのような主張をしている。CAFC の判断を見る限り、Oracle の戦略は成功したというべきであろう。しかし、同教授は言うまでもなく現行法を無視しているのではなく、Baker 判決と 102 条(b)の正確な理解に基づいて（Nimmer の解釈の誤りを正すべく）議論していることは、Samuelson, supra 35 の論文を読めば明らかであり（Google はこの論文を引用しているが、Oracle がその主張に用いたのは、Pamela Samuelson, CONTU Revisited: The Case Against Copyright Protection for Computer Programs in Machine-Readable Form, 1984 Duke L.J.663(1984)である。）、CAFC の批判は当たらないように思われる。

コンピュータプログラムの法的保護の在り方は 80 年代～90 年代に大きく議論されたテーマであり、著作権法と特許法という異なる法律を用意する知的財産法体系の下でそれをどのように位置づけるのかは重要な視点である。

その技術的性格に付随する社会の恩恵ということを考えた場合、後行者による改良が徐々に加えられていくことが他の著作物とは大きく異なる特徴[56]であり、かかる特徴うえに、他の伝統的著作物に比べ、コンピュータプログラムの著作権による保護はより制限的・謙抑的に解釈されるべきである。具体的には、文字的要素（ソースコードとオブジェクトコード）のデッドコピー又はそれに類する複製に限るべきであろう。CAFC は、機能性の保護が特許ではなく著作権で保護されることの意味を十分に咀嚼していないようである。かかる解釈は、特許法において保護されない機能[57]が著作権法によって最小限度の創作性を充足するだけで特許より長い法的保護を享受することを認めるものだが、コンピュータプログラムの機能（特に nonliteral な部分）がそこまで強固な法的保護を受けることを正当化する事情を提示していない。

前述のとおり、Whelan 判決は、コンピュータプログラムのアイデアに当たるものにまで、特許法の求める厳格な要件を充足しないにも関わらず特許以上の長期間にわたり独占権を付与する（法的保護を与える）結果を生じさせるものであったが、Altai 判決及び Lotus 判決によって、コンピュータプログラムの著作権による法的保護の領域が極めて限定的になものになったと理解されたため、ソフトウェアの SSO について保護を求める者は特許権を取得することが一般的なものとなっていったと評価されていた。すなわち、著作権による保護の強化傾向は Whelan 判決以後一時強まったものの、90 年代半ばまでには適正なレベルまでに弱まり、その取扱いが安定した。[58][59]

本決定は、そのような理解を根底から覆す判断である。CAFC の判断が米国のソフトウェア産業にとってどのような影響を及ぼすのか、この事件の進展とともに、米国内での議論の推移を見守る必要がある。

また、各論的には、後発者の互換性を実現するための SSO の複製がフェアユースを構成

---

[56] Karjala, supra note 3, at 61, 「文芸の著作物に関しては、最終章だけが異なっている 100 冊の「戦争と平和」よりも、100 の異なった戦争小説を持つことを我々は望む。その結果、小説に対する幅広い長期間の保護が、後行者を不当に妨害することなく、また、社会から望ましい作品を奪うことなく創作者を評価するという目的を達成する。しかし、技術は、消費者にとってより望ましい又は有用なものとするために後行発明者がわずかな改変を加えることにより徐々に改善されていくものである。」。かかる根本的な相違が著作権法と特許法のポリシーにあることを指摘し、このような基本的な視座から、コンピュータプログラムが著作権法の下で保護されることの意味を考える必要を説く。

[57] *Alice Corp. Pty v. CLS Bank Int'l, 573 U.S. ___ (2014)* によって、ソフトウェア特許の法的保護の基準は従前よりも厳格なものとなると推測される。同判決について SLN137 号(2014.7)参照。
USPTO のメモランダムも参照。http://www.uspto.gov/patents/announce/alice_pec_25jun2014.pdf　CAFC のような解釈は、そこからこぼれた特許に値しないコンピュータプログラムの機能を著作権法によって保護する道を開くものとなろう。

[58] Pamela Samuelson, The Uneasy Case for Software Copyrights Revisited, 79 Geo. Wash. L. Rev. 1746, 1775-1782, 1980 年代～90 年代のコンピュータプログラム著作権をめぐる動向を総括し、コンピュータプログラムに対する著作権による保護が弱まる形で安定したことを肯定的に評価する。特に 90 年代の裁判例の動向がソフトウェア産業にマイナスの影響を与えたとは評価できないとする。

[59] 原決定も同様の理解をしている。原決定 32 頁。

するのか、つまり、互換性・技術の標準化という事情に基づきフェアユースが認められるのか、という点の差戻し後の原審の判断も注目される。[60]

なお、本稿の作成にあたり椙山敬士弁護士より様々な助言及び示唆を頂いた。最後に記し、お礼を申し上げたい。

以上

[60]いわゆる変容的利用（transformative use）法理では、直ちにフェアユースとの判断が導き出せないように思われる。互換性を実現するためのリバースエンジニアリングについてフェアユースとするのが第 9 巡回区控訴審裁判所の判例ではあるが、Google の行為は、リバースエンジニアリングをして新たな物を創作したものではないため（当然 Oracle は Sega 事件等との相違を強調する。）、その点がどのような判断がなされるのかも注目される。互換性実現に対する需要を考えれば、フェアユースとの判断が出る可能性も相当程度あるように思われる。米国が 107 条の柔軟な解釈によって産業を発展させてきた面は否定できず、このような米国特有の社会背景も加味する必要がある。See, Pamela Samuelson, Unbundling Fair Uses, *77 Fordham L. Rev. 2537, 2605* (2007), Fred von Lohmann, Fair Use as Innovation Policy, 23 Berkeley Tech. L.J. 829 (2008).

SLN SOFTIC LAW NEWS

一般財団法人ソフトウェア情報センター
発行・編集　専務理事　亀井 正博

No. 159(2018/4)

ORACLE AMERICA, INC., v.　GOOGLE LLC
米連邦巡回区控訴裁判所（CAFC）２０１８年３月２７日判決
―フェアユースの適用を否定―

弁護士　椙山 敬士

１．はじめに

SLN138 号で石新智規弁護士が詳細に報告した事件の第２ラウンドである。2010 年、ORACLE は自己が保有する Java のアプリケーション・プログラム・インターフェイス(API)の著作権が、GOOGLE の携帯用 OS である Android に複製されたとして、カリフォルニア州北部地方裁判所に提訴した。同裁判所は、API の著作物性を否定した[1]。

控訴審である CAFC[2]は、GOOGLE が Java API の 37 パッケージの declaring code 及び SSO(structure, sequence, organization)を侵害したとして原審判決を破棄し、フェアユースの判断をするために原審に差し戻した[3]。

GOOGLE は certiorari（裁量上告）の申立をしたが、最高裁は却下したため、差戻審であるカリフォルニア州北部地方裁判所に係属したが、同裁判所はフェアユース（アメリカ著作権法 107 条[4]）の適用を認めた[5]。

---

[1] Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012).

[2] この事件は特許権侵害も同時に主張されていたため、控訴審は特許に関する専属的控訴審である CAFC に係属した（特許権侵害については控訴されなかったが、１審で特許権侵害が含まれていれば控訴審は CAFC になる）。著作権事件の控訴審はその地区を管轄する全米の 12 の巡回区控訴裁判所に係属することになるので、CAFC は通常は著作権事件を扱わない。本件のように特許権侵害も含まれていたために CAFC に控訴審が係属する場合、CAFC は当該地裁の地域を管轄する巡回区控訴裁判所（本件では第９巡回区）の先例に従うことになる。

[3] Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1348 (Fed. Cir. 2014), SLN138 号。

[4] （107 条）106 条及び 106A 条の規定にかかわらず、批評、解説、ニュース報道、授業（教室において使用するための多数の複製を含む。）、研究又は調査等を目的とする著作物の公正使用（複製物又はレコードへの複製その他同条に明記する手段による使用を含む）は著作権侵害とならない。特定の場合に著作物の使用が公正使用となるかどうかを判定する場合には、考慮すべき要素として以下を含むものとする。

（１）使用の目的及び性格（使用が商業性を有するかどうか又は非営利の教育を目的とするかどうかの別を含む。）

© 2018 (一財)ソフトウェア情報センター

これに対し、ORACLE は控訴した。CAFC はフェアユースの適用を否定し、原審判決を破棄、損害賠償額の決定のために再び原審に差戻した（本件判決）。

２．フェアユースを認めた原審判断

原審のカリフォルニア州北部地方裁判所は、フェアユースの４つの判断要素につき次のように判断して、フェアユースを認めた。

第１要素（使用の目的及び性格）GOOGLE の使用は商業的であるが、スマートフォンのために必要な要素を選択し自分のコードと統合した点で transformative[6]であるとした。

第２要素（著作物の性質）「高度に創作的」とは言えないし、「設計においては機能的考慮が支配的である。」とした。

第３要素（使用された部分の量及び実質性）transformative な使用のために合理的に必要なだけしか GOOGLE はコピーしておらず、コピーされた行数は最小限である。

第４要素（市場への害）著作物の市場であるデスクトップやラップトップには害がない。

３．フェアユースを否定した CAFC の判断

CAFC のフェアユースに関する実体的判断は次のとおり。

（１）第１要素（使用の目的及び性格）

CAFC は、この要素を商業的利用、transformative な使用、悪意の３つの項に分けて論じる。

a. 商業的利用

まず一般論として、従来の判例を踏まえ、利用の商業性はフェアユースの認定に不利に働くが、107 条の柱書の利用もほぼ商業的であるとして、二次的使用者の商業的利用の程度（商業における付随的利用と比して）が意味を持つとする。

GOOGLE はこの点について、（１）Android はオープンソースとして提供しているから非商業的である、（２）GOOGLE の収入は Android 以前のサーチエンジン上の広告から得ている、と主張した。

CAFC は次のようにこの主張を認めなかった。（１）については、Android が無償であるからといって、GOOGLE の Java API の利用が非商業的になるわけではない。ナップスター判決[7]を引いて、「利用者が通常購入するものをただで提供することも商業的利用たり得る。」。

GOOGLE が非商業的動機を持っていても法律問題としては無関係である。Harper & Row 最高裁判決[8]を引いて、被告が原告の本から抜粋したことは、部分的には公衆に

（２）著作物の性質

（３）著作物全体との関連において、使用された部分の量及び実質性

（４）著作物の潜在的市場又は価値に対する使用の影響。作品が未公表であるという事実は、上記全要素を考慮した上認定される限り、それ自体で公正使用の認定を妨げるものではない。

[5] Oracle Am., Inc. v. Google Inc., No. 3:10-cv-3561 (N.D. Cal. June 8, 2016)

[6] フェアユースを認めた有名な最高裁判決（プリティーウーマン判決）で用いられた用語。Cambell v. Acuff-Rose Music Inc., 510 U.S. 569(1994), SLN56 号。変容的などと訳されるが後に詳述する。

[7] A & M Records, Inc., v. Napster, Inc., 239 F.3d 1004 (9th Cir. 2001), SLN87 号。

[8] Harper & Row v. Nation Enterprises, 471 U.S. 539 (1985)

ニュース的価値のある情報を提供するという目的のためであっても、問題は「利用の目的がもっぱら金銭的目的であったかどうかではなく、ユーザーが通常の料金を支払わずに著作物を利用することから利益を得ることになるかどうか、である。」。

（2）GOOGLE は収入を Android からではなく広告から得ているとしても、商業性は GOOGLE がどのように収入を得るかに依存しない。「商業的利用を立証するために直接的経済的利益は要求されない。[9]」。GOOGLE の商業的利用はフェアユースの認定に不利に働く。

b. transformative な利用

著作権法に transformative という言葉はないが、最高裁はプリティーウーマン判決で、第１要素の「中心的目的は、新しい作品が transformative であるかどうか、どの程度 transformative かである」とした。利用が transformative になるのは、「それが何か新しいものを付け加え、さらなる目的または別の性格をもって、先行作を新しい表現、意味又はメッセージで変更すること」をいう。決定的な問題は「新しい作品が原作の目的を単に代替するものか、・・・新しいものを付け加えているかどうか」にある。

プリティーウーマン判決は、「transformative な利用がフェアユースの認定のために絶対に必要なわけではない」としつつ、「新しい作品が transformative であればあるだけ、（フェアユースの認定に不利に働きうる）商業性といった他の要素の重要性が低くなる。」。

原審は、コピーされた declarations が同じ機能、目的であることを認めつつ、GOOGLE がコピーする部分を選択し、スマートフォンの限定的な作動環境に適合するようコードをインプリメントしたことは、「コピーされたコードに新しい表現、意味又はメッセージを付与する新規なコンテキスト」を作り出しているとした。

GOOGLE は、Java API パッケージの小部分を使って、「デスクトップやサーバーではなく、スマートフォンのプラットフォームである Android」という新しいコンテキストにおいて新しい作品を創造した、と主張する。

当審は、GOOGLE の議論は認められないとする。GOOGLE の利用は transformative ではない。その理由として、（１）107 条の柱書に列挙する利用に合致しない、（２）Android における Java API パッケージの目的は Java プラットフォームにおけるパッケージと同じ利用である、（３）スマートフォンは新しいコンテキストではない、とした。

（１）について、GOOGLE の利用は 107 条柱書の「批判、コメント、又はニュース報道等」に当たらない。

（２）について、API パッケージは両作品において同じ機能に仕えることは争いがない。GOOGLE が選択的にコピーしたとしても、それ自体 transformative とは言えない。自らコードをインプリメントしたかどうかも transformative かどうかについて関連性がない。ただコピーして別のプラットフォームに変更なく移し替えることは transformative ではない。

（３）について、Java SE は、ライセンス等でスマートフォンで既に使われており、

[9] A & M Records, 239 F.3d at 1005

新しいコンテキストではない。そもそも、あるものを新しいコンテキストに移してもそれ自体で transformative ではない。第９巡回区控訴裁判所は、GOOGLE によるサムネイル画像のサーチエンジンでの利用を認めた判決[10]で、「利用が transformative と考えられるのは、被告が原告作品を変更したか、原告著作物を別のコンテキストで利用して原告作品を新しい創作の中に移されるようにした場合である。」と述べている。同審が transformative としたのは、原作品が娯楽、審美的又は情報提供的であるのに対し、サーチエンジンでは画像をユーザーに情報の出所を指摘するポインターにしている。したがって、同事件において、コンテキストの変更自体が重要だったのではなく、コンテキストの変更が目的の変更を促進したことが利用を transformative にしたことは明らかである。そのままのコピーであり、同じ機能と目的の為であり、表現内容又はメッセージに何の変更もない場合、単なるフォーマットの変更（例えば、デスクトップやラップトップコンピューターからスマートフォンやタブレットへの変更）は、transformative な利用となるには不十分である。

c. 悪意(bad faith)

フェアユースは衡平法(equity) 上の原理なので、悪意はフェアユースの認定を妨げる。原審において ORACLE は GOOGLE がライセンスの必要性を認識していたことなどから悪意を主張したが、原審は認めなかった。当審としては、陪審は端的に GOOGLE の悪意を認定する証拠を認定しなかったのだと考える。それ以上に、作品の利用許諾を否定されただけでフェアユースの認定に不利に働くわけではない。

結論として、GOOGLE が悪意で行動したことに陪審が説得されなかったとしても、GOOGLE の利用は高度に商業的であり、transformative でない性質であるから、第１の要素はフェアユースの認定に不利に働く。

（２）第２要素（著作物の性質）

この要素は、作品が情報提供的か創作的かという点にかかる。地裁は、「declaring code や SSO は著作権保護を認められるのに十分な創作性を有するが、その設計には機能的な考慮が支配的である、と陪審は合理的に認定したはずである」と結論した。

当審も、機能的考慮は実質的であり、重要だと結論する。第２の要素はフェアユースの認定に有利に働く。しかし、第９巡回区控訴裁判所は、この第２の要素は「類型的にフェアユースの全体的な判定において特に重要ではない」としてきた。したがって、陪審の見解として第２の要素がフェアユースに有利だと想定したとしても、全体的な分析の上で重要性は低い。

（３）第３の要素（使用された部分の量及び実質性）

この「使用された量及び実質性は」侵害作品においてではなく、非侵害作品において見ることになる。この要素の判断は、量と質的価値の双方をみることになる。この判断は柔軟なもので単なるパーセンテージの問題ではない。地裁は、GOOGLE が ORACLE の作品の１％という小さな部分をコピーした点に注目した。両当事者は Java 言語で書くのに必要なのは 170 行のコードに過ぎないことに同意したが、GOOGLE は 11,500 行もコピーしたので、11,330 行も必要な数より多い。また、37 の API パッケー

[10] Perfect 10, Inc. v. Google, Inc. 508 F. 3d 1146 (9th Cir. 2007)

ジのSSOもコピーしている。地裁は、Javaのプログラマー達がJavaのシステムとAndroidのシステムの間で混同しないようシステム間の一貫性を保持しようとしたGOOGLEの希望を強調した。GOOGLE は当審において相互運用性(interoperability)に依拠していない（GOOGLE は Java プラットフォームと非互換に設計している）。GOOGLE は、ソフトウエア開発者が問題の Java API パッケージを使うのに既に訓練と経験をしているという事実を利用することを求めた。しかし、著作物の人気を利用することや想定されるユーザーの期待に合うためにコピーするという固有の権利などない。ソフトウェア開発者が慣れている著作物のアスペクトをこれらの同じ開発者に受けるために取ることはフェアユースではない。小部分をコピーしたにすぎないとしても、質的に重要でないとはいえない。

以上のような理由から、第３の要素は、フェアユースの判断においてせいぜい中立的であり、むしろ不利と言えよう。

（４）第４の要素（潜在的市場への影響）

この要素は、フェアユースは「コピーされる作品の市場性を実質的に損なわないような他人によるコピーに限定される」[11]という考えを反映している最高裁はかつて第４の要素が「疑いもなくフェアユースの最も重要な要素である[12]」と言った。しかし、その後プリティーウーマン判決では４つの要素を別個に考察すべきではなく、すべてを検討し、著作権の目的に照らして結果を合せて衡量すべきであるとした[13]。

第４の要素の評価に当たっては、著作物の現実的又は潜在的な市場への害だけでなく、「潜在的な派生的使用の市場」への害も考慮すべきであり、それには、「原著作物の創作者が一般に展開するだろう市場や他人に展開を許諾するだろう市場をふくむ。」[14]。

原審は、Android がリリースされる以前に、Sun（ORACLE の先代）がすべての Java API パッケージを OpenJDK という名称で無償のオープンソースで提供していた（GPL の条件にのみ服する）ことに注目し、Android における使用はデスクトップやラップトップの市場の害にならないとした。そして、著作物の市場への Android の影響は、Sun が OpenJDK によって既に予想していたことに相当するとした。

当審において ORACLE は現実的、潜在的市場への影響は甚大であると主張するが、当審は同意する。

第一に、現実的害に関し、Java SE は Android のリリース以前に Blackberry, SavaJe, Danger, and Nokia 等のスマートフォンで使用されていた。タブレットについては、Amazon はキンドルで Java SE を使っていたが、Android に乗り換えた。Android は Java SE の替わりに使われたのであり、直接的な市場の害があった。

仮に、ORACLE が Java SE をライセンスしていたかどうかにつき争いがあるとしても、フェアユースは現実的だけでなく潜在的な市場も問題にする。Android におけるコピーは、ORACLE が参入する市場、自ら創作したり他人に創作をライセンスして派生的市

[11] Harper & Row, 471 U.S. at 566-67
[12] Harper & Row, 471 U.S. at 566
[13] Cambell, 510 U.S. at 578
[14] Cambell, 510 U.S. at 592

場に影響を与える。ORACLE が装置メーカーでなく、今までスマートフォンを作っていなかったとしても、ライセンスできるのであるから、市場が害される。

以上より、第４の要素は ORACLE におおいに有利に働く。

（５）４つの要素のバランス

著作権法の目的に照らして、GOOGLE に ORACLE の作品の商業的利用を許すことは著作権の目的を高めることにならない。GOOGLE は自分自身の API を開発することにより、または新しいプラットフォームの開発に ORACLE の API のライセンスを受けることにより、創作的表現やイノベーションを促進するという著作権の目標を増進することができたはずであるにもかかわらず、GOOGLE は ORACLE の創作的努力をコピーすることを選んだ。著作物をそのまま奪い競合するプラットフォームにおいて原作品と同じ目的や機能のために使用することはフェアではない。

第２の要素は GOOGLE に有利に、第３の要素はせいぜい中立的であるが、全ての要素を考慮するとフェアユースには決してならない。

（結論）ORACLE の申立を認め、地裁判決を破棄し、損害額の認定のために差し戻す。

４．検討

いくつかの問題点を検討する。

（１）CAFC による著作権判断の問題点

先に述べたように、アメリカの特許と著作権の管轄の在り方からして、CAFC は普段は著作権事件を扱わないし、各事件の原審が帰属する各巡回区控訴裁判所の先例に従うことになる。いわば付け焼刃で事案を処理することになる。普段から著作権事件を扱っているなら、実際の事件を多数経験できるだけでなく、他の裁判所も含めた過去の事例を、歴史的経過も踏まえてバックグラウンドとして持ちうることになる。しかし、CAFC はそのような立場に立ちえない。このことは、１回目の控訴審において、1986 年の Whelan 判決[15]の認めた SSO（structure, sequence, organization）の保護に無批判に従ったことに端的に表れていたといえよう。たしかに、同判決は第９巡回区控訴裁判所の判例であるが、アメリカ全体の流れを見ると、1991 年の Feist 判決[16]におけるシンプルなインセンティブ論の否定を受けて、1992 年の第２巡回区の CA 対アルタイ判決[17]でプログラムの特性に応じた制限的な保護に落ち着いたという歴史があり、この流れは大多数の学者、実務家から２０年以上是認されてきたのである。CAFC の１回目の判決は先祖返りのような驚きを斯界に与えた。このことは、２回目の判決においても指摘しなければならない。すなわち、フェアユースの判断においても、SSO の保護は持ち出されており、これを疑いなく墨守しているからである。

総じていえば、CAFC による著作権法判断は、経験と厚みを欠いたものであり、プログラムを著作権で保護する際の特殊性（技術的特性）に対する十分な配慮がなされていないと思われる。

---

[15] Whelan Assocs., Inc. v. Jaslow Dental Lab. Inc., 97 F.2d 1222 (3rd Cir. 1986), SLN1 号。

[16] Feist Publications, Inc. v. Rural Tel. Serv. Co. 111 S.Ct 1282 (1991), SLN26 号。

[17] Computer Assocs., Int'l v. Altai, Inc., 982 F.2d 693 (2nd Cir. 1992), SLN39 号。

（２）transformative の意義

次に、transformative の意義について一言しておく。trans とは、transportation, transfer, tranlation などのように別の状態に移ることという。form とは形である。よって、transformative は移形的ということになるが、日本語にならないので、原語のまま用いることとする。

プリティーウーマン事件の事案は、オリジナルのポップスをラップに変えたというもので作品自体を変更する態様のものであった。しかるに、前出の perfect10 の事件では、サムネイル画像というだけで作品としての変更はなかった。それでも裁判所は索引的な利用という態様をみて、transformative であってフェアユースにあたるとした。これは、本 CAFC 判決も感じているようにプリティーウーマン判決の用語法とは合わないと思われる。transformative という語の「変容」が生じた一つの原因は、プリティーウーマン判決の、先に引用した説示自体にあったように思われる。すなわち、「決定的な問題は、新しい作品が原作の目的を単に代替するものか、・・・新しいものを付け加えているかどうか」にある。同事件において、被告作品は「新しいものを付け加えている」し、「単に代替するもの」ではない。しかし、この２つの事柄はきれいに反対概念をなしているわけではない[18]。すなわち、「代替的利用」ではないからといって作品として「新しいものを付け加えている」とは限らない。したがって、「代替的でなければよい」としてしまえば、サムネイルの事案でも（作品の変更はなくとも）代替的ではないからフェアユースにあたる、とすることができるのである。ただ、このような拡張的論法を用いると、判断基準としては極めてあいまいなものとなる。現に、他の教会で用いられているテキストは別の教会で用いることは新たな提供になるから transformative だとか、通信線を伸ばして放送をより広い範囲に及ぼすことは transformative だというような事案も見られるようになった。

確かに、プリティーウーマン判決が transformative という語をキーワードとして用い、代替的利用と対置させたことがフェアユースに関する裁判のその後の混乱の一因となったように思われる。したがって、CAFC がこの語を本来の意義に近づけて判断したのも無理からぬ面がある。

しかし、翻って考えてみると、107 条は本来 transformative という一語で代表させるようなものではなかったはずである。著作物を巡る様々な事情を考慮に入れることは当然に予定されているし、107 条の文言もそうであろう。本件のような事案でも、フェアユースの判断においてプログラムの特性という点について、よくよく吟味される必要がある。

（３）技術的特性

本件において、フェアユースの判断にあたっては、（CAFC の判断による）ＳＳＯ保護を不適切とし、コードの量的な少なさを前提としたうえで、技術的特性として評価するべき最重要な点は「ソフトウェア開発者が問題の Java API パッケージを使うのに既に訓練と経験をしているという」事実である。通常の著作物であれば、このような現象は全くおきようがない。しかし、社会全体の効率、効用を考えると、当然に一考しなければ

[18] 「使用が商業性を有するかどうか又は非営利の教育を目的とするかどうかの別を含む。」という第１要素のカッコ書きもそうである。「商業性」と「非営利」はほぼ反対概念であるとしても、「商業性」の反対が「非営利の教育」に限定されるわけではない。

ならない事柄である。このことは、技術そのものというのではなく、この技術に関わる社会的対応の問題ということになる。ユーザーインターフェイスの保護は限定的である[19]。これは、ユーザーが一定の操作になれると別のインターフェイスに移り難いという、いわゆるユーザーロックインという現象を考慮に入れているからだと言えよう。このような特性は他の著作物には見出しがたい（簿記とか書式とかの技術的要素があるものではありうる）。同様に、開発者が一定の言語、ツールに慣れている場合にも似たようなことが起きうるわけである。この場合は、デベロッパーロックインということができよう。

本件は間違いなく最高裁で判断が出るであろう。グーグルブックス事件[20]にみるように、アメリカの裁判所の傾向としてはグーグルのイノベーションについて寛容な政策的判断をしているようにも思われる。特許事件で CAFC の多くの判例が最高裁で覆されていることからも、予断を許さないところといえよう。今後のプラットフォームの帰趨にも大きな影響を与えることだろう。

以　上

[19] Apple Computer, Inc. v. Microsoft Corporation, 35 F.3d 1435 (9th Cir. 1994), SLN59 号、Lotus Dev. Corp. v. Borland Int'l Inc., 49 F.3d 807 (1st Cir. 1995), SLN62 号。

[20] The Authors Guild, Inc.　v. Google Inc., 2d Cir. 2015 804 f 3d 202, SLN149 号。　Certiorari denied, April 18, 2016.

SLN SOFTIC LAW NEWS

一般財団法人ソフトウェア情報センター
発行・編集　専務理事　亀井 正博

No. 165 (2020/10)

Google LLC v. Oracle America, INC.

米連邦最高裁口頭弁論*

－ 米国ソフトウェア著作権の行方 －

弁護士　石新 智規

目　次

* 本件の理解に必要な情報は、https://www.scotusblog.com/case-files/cases/google-llc-v-oracle-america-inc/ で入手できる。

© 2020 一般財団法人ソフトウェア情報センター

〒105-0003 東京都港区西新橋 3-16-11
愛宕イーストビル 14 階
TEL: 03-3437-3071 / E-mail: res@softic.or.jp

１　はじめに

本件は、SLN138 号で私が報告した第 1 ラウンド[1]、SLN159 号で椙山敬士弁護士が報告した第 2 ラウンドの事件[2]に続く第 3 ラウンドである。舞台は連邦最高裁である。最高裁判決の行方を占う口頭弁論が 10 月 7 日に開かれた[3]。従来、法廷傍聴しなければ、連邦最高裁から反訳および録音が後日公開されるまではその内容を正確に知ることはできなかったが、コロナ感染症拡大防止策として連邦最高裁は 5 月から口頭弁論を電話会議方式で行っており、それをオンラインで同時公開している（年内の口頭弁論はこの形態で実施するようである）。本件口頭弁論も日本時間午後 11 時（アメリカ東部時間午前 10 時）から聴くことができた。知財関係者による実況中継や弁護士や裁判官の発言に対するコメントが Twitter 上に現れ、非常に参考になった。日本時間の翌朝には録音反訳がウェブ上で公開された。

本件は、Google と Oracle という IT 業界を代表する企業間で争われ、Oracle が勝訴すれば非常に高額の損害賠償が認容される可能性があること、連邦最高裁の判断が米国におけるソフトウェア著作権の保護範囲を決定づけ、ソフトウェア産業の今後に非常に大きな影響を与えるものであるため、口頭弁論前から非常に注目を集めていた[4]。

冒頭に述べた通り本件はすでに第 3 ラウンドではあるが、経緯を簡単に振り返る。

2010 年、Oracle は Java のアプリケーション・プログラム・インターフェイス（API）が Google の携帯用 OS である Android に許諾なく複製されたとして、Google に対して著作権侵害訴訟を提起した。

原審（カリフォルニア州連邦地裁）では Java API の 37 の declaring code 及び SSO (structure, sequence, organization) の著作物性が否定されたが、控訴審である CAFC は、API の著作物性を認め原審判決を破棄し、フェアユース抗弁（アメリカ著作権法 107 条）の成否を判断するために事件を原審に差し戻した（2014 年判決[5]）。Google は certiorari（裁量上訴）の申立てをしたが、最高裁は受理しなかった。差戻し後の原審（陪審）は Google による複製をフェアユースと認めたが、Oracle が控訴し、CAFC は原判決を破棄し、損害賠償額の審理のために再び原審に差し戻した（2018 年判決[6]）。本件は、2018 年判決に対する裁量上告の申立てが受理されたものである。

2014 年判決を受理しなかった連邦最高裁が今回は受理を決めたこと、さらに 2014 年判決の際に論点となっていた API の著作物性についても審理の対象に含めたこと、さらに、連邦最高裁が合衆国訟務長官（Solicitor General、以下単に「政府」という）に対して政府の見解を提出するよう促したうえで受理している[7]ことから、連邦最高裁が本件の論争に決着をつけようという姿勢が感じられるものであった。

２　裁量上訴の対象となった論点

本件の主要な論点は以下の 2 点である[8]。

---

[1] 石新智規「*ORACLE AMERICA, INC v. GOOGLE INC* 米連邦控訴審裁判所（CAFC）2014 年 5 月 9 日判決 ～アプリケーションプログラミングインターフェースの著作物性が肯定された事例～」(SOFTIC LAW NEWS 138 号)

[2] 椙山敬士「*ORACLE AMERICA, INC., v. GOOGLE LLC* 米連邦巡回区控訴裁判所（CAFC）2018 年 3 月 27 日判決 ―フェアユースの適用を否定―」 (SOFTIC LAW NEWS 159 号)

[3] 裁量上告は 2019 年 11 月 15 日に受理され、2020 年 3 月に口頭弁論が開催される予定であった。しかし、コロナ感染症の拡大のために延期され、10 月となった。

[4] 非常に多くの関係者からアミカス・ブリーフが提出されている。膨大なアミカス・ブリーフを分析し、本件の予備的考察を行った論考として、玉井克哉「裁判所における『熟議』－グーグル対オラクル著作権侵害事件におけるアミカス・ブリーフを素材に－」Nextcom 42 号 4 頁（2020）参照。

[5] Oracle Am., Inc. v. Google Inc.,750 F.3d 1339 (Fed. Cir. 2014)

[6] Oracle Am., Inc. v. Google LLC, 886 F.3d 1179 (Fed. Cir. 2018)

[7] https://www.supremecourt.gov/orders/courtorders/042919zor_f2q3.pdf

[8] https://www.supremecourt.gov/docket/docketfiles/html/qp/18-00956qp.pdf

(1) 著作権の保護はソフトウェアのインターフェースに及ぶのか。

関係する条文は、102 条 (a)[9]及び 102 条 (b)[10]である。102 条 (a) は著作物を列挙しており（コンピュータプログラムは言語著作物として保護される）、102 条 (b) は著作権で保護されないものを規定している。102 条 (b) は、「著作物に対する著作権による保護は、それが表現、説明、描写又は具現化される形態に関係なく、いかなるアイデア、手続、プロセス、システム、動作方法、概念、原則又は発見に対しても及ばない。」と定める。

(2) 陪審が認めたように、ソフトウェアインターフェースの利用は、新しいコンピュータプログラムを創作する文脈でフェアユースを構成するか。

関係する条文は 107 条である。

「第 106 条及び第 106A 条の規定（筆者注、著作権の支分権規定である。）にかかわらず、批評、解説、ニュース報道、授業（教室における利用のために複数のコピーを作成する行為を含む）、研究又は調査等を目的とする、複製物、レコードへの複製又は同条に定められるその他の手段による利用を含む著作物のフェアユース（公正利用）は、著作権の侵害とならない。

特定の場合に著作物の利用がフェアユースとなるか否かを決定する際に考慮すべき要素は、以下を含むものとする。

(1) 利用の目的及び性質（当該利用が商業性を有するか否か、又は非営利的な教育目的か否かを含む）

(2) 著作物の性質

(3) 著作物全体との関連において利用された部分の量及び重要性

(4) 著作物の潜在的市場又は価値に対する当該利用の影響

上記のすべての要素を考慮してフェアユースが認定される場合、作品が未発行であるという事実それ自体は、フェアユースの認定を妨げないものとする。」

2020 年 5 月になり、連邦最高裁は両当事者に対し次の論点について補充書面を提出するように指示し、手続法上の論点が 1 つ増えた。

(3) 陪審裁判（トライアル）を受ける権利を保障している合衆国憲法修正 7 条等の観点から、CAFC がトライアルを経た陪審の判断を覆しフェアユースを否定した判断手法は適法か。

関係する条文は連邦民事訴訟規則 50[11]である。同条は、裁判所が陪審の評決を覆すことができるのは、合理的な陪審員がそのような評決に至る十分な証拠を有していないと

---

[9] Article 102 (a), “Copyright protection subsists, in accordance with this title, in original works of authorship fixed in any tangible medium of expression, now known or later developed, from which they can be perceived, reproduced, or otherwise communicated, either directly or with the aid of a machine or device. Works of authorship include the following categories:
(1) literary works;…”

[10] Article 102 (b), “In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work.”

[11] Rule 50 (a) Judgment as a Matter of Law.
(1) In General. If a party has been fully heard on an issue during a jury trial and the court finds that a reasonable jury would not have a legally sufficient evidentiary basis to find for the party on that issue, the court may:
(A) resolve the issue against the party; and
(B) grant a motion for judgment as a matter of law against the party on a claim or defense that, under the controlling law, can be maintained or defeated only with a favorable finding on that issue.
(https://www.law.cornell.edu/rules/frcp/rule_50)

言える場合に限られると定めている。本件で CAFC が陪審がフェアユースとした評決を覆し、自ら判断することは違法ではないか（誤りがあるのなら陪審に差し戻すべきではないか）という論点である。

３　口頭弁論

(1)　審理の進行[12]

Roberts 長官が弁論を主宰する。まず申立人 Google の代理人に対し意見陳述の機会が与えられた後、Roberts 長官からの指名で順に各裁判官と代理人の質疑応答が行われる。その後、代理人に短時間だが再び意見陳述の時間が与えられる。次に Oracle 代理人の意見陳述、裁判官との質疑応答が同じように進行する。さらに本件は政府が参加しているため、政府についても同様の手続が進行した。最後に、Oracle と政府の陳述を踏まえ Google 代理人との質疑応答と最終陳述がなされた。

(2)　Google 代理人の意見陳述の概要[13]

API の利用はソフトウェア産業において相互運用性を確保するために当然に行われてきた。Baker 判決[14]及び 102 条 (b)に基づき、いかに創作性があろうとも、method of operation に該当する場合には、著作物性が否定される（機能性は著作権では保護されず、特許権で保護される）。Java ソフトウェア開発者はアンドロイドスマートフォンのためのアプリケーションを開発することができるが、それを実現させるためには、Google は Java から declaration コードをそのまま複製する必要が生じる。そのような相互運用を可能にするためのコードは一つであり、そのため Merger（混同、以下単に“Merger”と記す）が生じる。

Google は、API を、ロックを開錠するための鍵であるというアナロジーを用いて説明しようとした。

また、API を無許諾で複製することは、相互運用するソフトウェアを開発するために長年にわたり行われており、そのような社会的に有用な利用はフェアユースであると主張する。そして、一定の事実関係に基づき判断されるべきフェアユースにつき、陪審は十分な証拠に基づきフェアユースとして判断した。フェアユースという陪審の評決を覆した控訴裁判所はこれまで存在しないと手続上の違法も指摘した。

(3)　Oracle 代理人の意見陳述の概要[15]

Oracle の主張は以下のとおり非常にシンプルである。

連邦議会は、コードがオリジナルなものである以上、コンピュータプログラムを著作物として保護すると定めたと主張。Google は declaring code と implementing code を分け、前者は保護されないと主張するが、declaring code の表現方法は複数あり、オリジナリティがある以上、両者を区別することはできない。

現著作物を代替（supersede）する利用がフェアユースとならないことは、Harper 判決

---

[12] Roberts 長官は厳格に進行管理をしており、代理人が発言中でも遮り、次の裁判官の質問に移行していた。そのため、概要では本来の主張の一部しか回答していないように見える部分がある。それは、時間の制約のため全てを発言できていない場合も多かったというに過ぎない点に注意いただきたい。各当事者の主張の詳細は、主張書面で確認いただきたい。

[13] Oral argument transcripts at 1-5
https://www.supremecourt.gov/oral_arguments/argument_transcripts/2020/18-956_kifl.pdf

[14] Baker v. Selden, 101U.S. 99, 101(1879), 会計士 Charles Selden は、会計帳簿の独特のシステムの使い方を説明する簡単な文章と 20 頁のフォームからなる書籍を発行したが、その死後、その酷似するシステムを発表した Baker に対し、書籍及び書籍で示されているシステムの著作権に基づき遺族が著作権侵害訴訟を提起したが、システムには著作権が及ばないとされた。

[15] Oral argument transcripts at 38-40

と Stewart 判決が示すとおりである。相互運用についても、ライセンスなくそれを許容する必要は全くなく、現に Apple も Microsoft も競合するプラットフォームを作るために Java コードを複製していない。また、IBM や SAP は Oracle からライセンスを受けて利用している。連邦最高裁がフェアユースを認めれば、多くのビジネスに多大な損害を及ぼすことになり、著作権法の目的に反する結果となる。

(4) 合衆国政府代理人の意見陳述の概要[16]

1970 年代に政府はコンピュータプログラムと著作権法を研究する委員会を設置し、1978 年に、CONTU[17]として知られるリポートを発表した。その中で、コンピュータコードに著作権による保護を与えることを推奨している。それが正当化される理由は、コンピュータコードの作成には相当の投資がなされており、それを自由に複製できるとすると、コンピュータコードの創作インセンティブが相殺されるからである。

(5) 質疑応答の概要

多数の書面が提出されており、口頭弁論の印象と今後の裁判官会議を経た結論とは一致するものではないが、口頭弁論は、各論点に関し裁判官が何を検討しているかを示唆しており、興味深い。総論としては、各裁判官が API の著作物性という論点を理解するのにちょうどよいアナロジーを探して質問をしていること（これに対し当初 Google 代理人が的確な回答ができず裁判官とのやり取りがかみ合っていない印象も受けた）、コンピュータプログラムの相互運用性の確保が（これまでの類型とは異なる）フェアユースの一類型として必要なのかという意識が感じられること、この判決がソフトウェア産業に与える影響を裁判官が非常に気にしていることが印象的であった。

ア API の著作物性について

（ア）Breyer、Kagan、Sotomayor の 3 裁判官は 102 条 (b) を根拠として declaring code の著作物性を否定する立場、Thomas、Alito、Gorsuch は 102 条 (b)で著作物性は否定されず、フェアユースの成否の問題と解釈する立場（後述のとおりフェアユースに対する態度は異なる）、Roberts 及び Kavanagh はその中間に位置するように思われた。

（イ）Roberts 長官は冒頭、「準備書面の項目と配列を複製することは自由にできるのか」と Google 代理人に対し尋ねた。おそらく、declaration を「項目」に、SSO を「配列」になぞらえて質問をしたのだと思われるが、Google 代理人がそのアナロジーはコンピュータコードとは異なるとして、コンピュータコードでは表現方法が限られる場合 Merger が生じるといった自己の主張を繰り返したため、長官は Merger について聞いているのではないと指摘し、やや議論がかみ合わない[18]。Google 代理人が、「そうする必要がないのであれば複製できない」と答えたため、議論が Merger に移行した。Roberts 長官は、Google がその主張書面で提供したロックと開錠キーのアナロジーに沿って、仮に金庫を開ける必要があるとしても、鍵をこじ開けたり、鍵の番号の組み合わせを複製することが直ちに許されるのかと指摘。これに対し、Google 代理人は、金庫や鍵の開錠の仕方の書籍を書いても、その金庫や鍵の複製をコントロールする権利は著作権では得られない（それは特許権の対象である）と

---

[16] *Id.*at 64-65

[17] Final Report on the National Commission on New Technological Uses of Copyrighted Works (1981). https://repository.jmls.edu/cgi/viewcontent.cgi?article=1573&context=jitpl

[18] Roberts 長官は、Java API パッケージの class 及び methods の名称並びにそれが組織化された体系を準備書面の「項目」と「配列」になぞらえ、それらは「機能」なので 102 条（b）に基づき（他の選択手段の検討する必要なく）複製できるとの回答を想定していたようにも思える。Oral argument transcripts at 5-6

Baker 判決に即し回答していた[19]。

Roberts 長官は、Oracle 代理人に対しては、レストランのシェフが考えだしたメニューを他のレストランが模倣する例を出し、Google が行っていることは許容されるのではないかと尋ねた[20]。Oracle 代理人は、レストランのメニューが比較的平準化しているのに対して、Java API はずっと複雑であると反論した。興味深いことに、Roberts 長官はフェアユースの成否については、いずれの当事者にも特に質問を発しなかった。

（ウ）Roberts 長官に比べ、Oracle の主張により親和性を感じさせたのは、Thomas 判事と Alito 判事である。

Thomas 判事は Google が 102 条（b）の例外を強調することに疑義を呈し、むしろコンピュータプログラムを定義する 101 条が重視されるべきではないかと指摘し、Google の Merger の主張について疑義を呈している。Oracle に対しては、declaring code のほかの論点については判断する必要はあると考えるかと尋ね、Oracle 代理人から、「コードはコードであり、法的に両者を区別する必要はなく、オリジナリティがあれば著作物性を認めるのが議会の意思である」との Oracle の主張を引き出している[21]。

また、Alito 判事も、Google の主張によれば、コンピュータプログラムは全て method of operation として著作権保護の対象から外れるのではないかという懸念を指摘する一方[22]、Oracle や政府に対しては著作物性に関する質問はなかった。

（エ）Gorsuch 判事は、コンピュータプログラムの定義に従い、その一部が method of operation としてカテゴリカルに著作物として保護されないという解釈を嫌う様子で、Merger の議論に焦点を当てた。もっとも、Merger 理論に納得している様子ではなく、Google の利用は、独占禁止法上の法理である不可欠施設法理に類するものではないかと指摘していた[23]。そして Gorsuch 判事は、Alito 判事と同様、Oracle と政府に対しては著作物性について質問をしなかった。条文どおり著作物としての該当性を認めた上で、何らかの根拠に基づくフェアユースの成立を検討しているように思われる。

（オ）Kavanaugh 判事は、Google に対し、「method of operation による保護の否定はコンピュータプログラムの著作権による保護をすべて放棄することになる」という政府の主張に対する反論を求めるとともに、Google の Merger の主張について、コード作成時には何通りも作成方法がある以上 Merger は生じないのではないか、すなわち、コードの書き方は複数あるのに後になって Google が複製する必要がある対象として選んだものは一つしかないというのは循環論法ではないか、「ある音楽の表現は一つだが、それを複製することは許されない」[24]という当然の原則がなぜここでは妥当しないのか、と指摘した。また、政府に対し、Google の Merger と method of operation の主張に対して意見を求めており[25]、Google の主張に疑義を有しているようにも思われた。

（カ）Sotomayor、Kagan、Breyer 各判事は、Google の主張に理解を示していたと思われる。Breyer 判事は、自ら設例を作り、declaring code と implementing code がどう違うのかわかりやすく説明をするように求め、Google 代理人から「declaring code は Java プログラムを呼び起こす言語上のルールである」（要するに method of operation に該当する）という回答を引き出している[26]。Breyer の質疑応答は、受験生を正解

[19] *Id.* at 8-9
[20] *Id.* at 40-42
[21] *Id.* at 43-44
[22] *Id.* at 17-18
[23] *Id.* at 29-33
[24] *Id*. at 33-35
[25] *Id*. at 79-82
[26] *Id*. at 14-16

に導こうとしている口述試験官による誘導のようにも感じられた。Oracle 代理人に対しては、Qwerty キーボードの例を出し、最初の創作者が著作権を有したら、タイプライターをコントロールすることになるのではないかと指摘していた[27]。

（キ）Sotomayor 判事は、Google 代理人に対して、複製が許されるコードとそうでないコードの基準を明確にするよう求め、それが問題の核心だと発言した[28]。その確認の様子からは、declaring code と implementing code の違いを理解しようと努めているように感じられた。Google 代理人との質疑時点では Google の主張に親和性までは感じられなかったが、Sotomayor 判事は、後に Oracle 代理人に対し、「API は著作権で保護されないが、implementing code は著作権で保護されると長年認識されてきたものをこの時点でなぜひっくり返す必要があるのか」と説明を求めており[29]、基本的に Google の主張に理解を示しているのではないかと感じられる。

（ケ）Kagan 判事は、Sotomayor 判事よりはっきりと declaring code と implementing code の違いを意識していた。Google 代理人が declaring code は 102 条 (b) の method of operation であり著作物性を欠くという本来の主張を明確に述べず、口頭弁論当初から、Merger に焦点を当てていたため、議論が「その他の方法があったのか」という点に流れていた。Kagan 判事は、「書面から理解していた主張と若干異なる。そもそも著作物性がないという主張はしない趣旨か」と尋ね、Google 代理人に自らの主張を整理させ[30]、declaring code はショートカットプログラムに対する指示という method of operation であり、そもそも著作物性がないと Google が主張していること、そして、第 2 巡回区控訴裁判所の判決である Altai 事件[31]のように、コンピュータプログラムの表現部分とそうでない部分とを分析するアプローチに類するアプローチを採るものであることを明確にさせている。口頭弁論開直後から主張の方向性がはっきりしていなかった Google の主張を綺麗に整理する助け舟を出していたように見受けられた。なお、Kagan 判事は、数学の証明問題において複数の証明方法がある中で、Oracle が一番よい証明方法を発見した、というアナロジーを提供し、その解法には著作権が認められないという Google の主張を引き出している[32]。また、Oracle 代理人との質疑では、本件の状況を食料品店の野菜や果物の陳列方法にオリジナリティがある場合になぞらえ、代理人に対し、このユニークな陳列方法に著作権が認められるのか、このように配列方法はいくらでも考えられるが、その最初に考え出した者に著作権が認められるのか。また、認められないのであれば、本件とどう違うのかを尋ねた。これに対し Oracle 代理人は、文書の形になっていなければ著作権保護はないと述べるほか、Kagan 判事の挙げた例に比べ本件のコードの配列や体系が非常に複雑であることを強調した[33]。

イ　フェアユース

（ア）トランスフォーマティブな（変容的な）利用か。

興味深いことに、フェアユースを認めなかった CAFC 判決に対する裁量上訴であるにもかかわらず、フェアユースの要件について議論がほとんどなされず、むしろ著作物性に議論が集中していた。

フェアユースの成否については、Sotomayor 判事と Kagan 判事がフェアユースに前向きな姿勢を前提として（前述のとおり両判事はそもそも著作物性に否定的だと思われる）、それを否定する政府の考えを質した印象を受けた。

---

[27] *Id.*at 46-48
[28] *Id.*at 21-24
[29] *Id.*at 53-54
[30] *Id.*at 25-26
[31] Computer Associates International, Inc. v. Altai, Inc., 982 F.2d 693 (2d Cir. 1992)
[32] Oral argument transcripts at 27-29
[33] *Id.*at 55-56

Sotomayor 判事は、Java のプラットフォームを PC から携帯に移した大きなステップがなぜトランスフォーマティブと言えないのかを尋ねた。これに対し政府は、API を Java プログラムを実行するために用いており、同じ目的で利用している。いわば、劇場公開された映画をインターネット上でライブストリームするのと同じであると説明した[34]。

また Kagan 判事は、Qwerty キーボード以上により便利なものを創作したとして仮定し、それを携帯電話プラットフォームに採用した場合のフェアユースの成否を尋ねた。政府は、相互運用の目的がフェアユースの判断に有利に働く可能性を認めたものの、判事の挙げた例では利益を受けるのが消費者であるのに対し、本件は Java プログラマーであるという点が異なり、本件でフェアユースを認めることは、自ら API を制作するという大変な作業を回避するために利用されるだけである旨の説明をした[35]。

（イ）107 条の定める 4 要素以外にも考慮するべき要素はあるか。

Thomas 判事は、Oracle に対して、コンピュータプログラムにおけるフェアユースはどのような場合に認められるかと尋ねた。Oracle は、研究目的での利用は認められるが競合品に複製することは認められないと回答した[36]。この質疑からは、同判事が Oracle の主張に沿う考えを有していることが感じられたが、他方、政府に対し、107 条の考慮要素が例示列挙にすぎないという連邦最高裁の判例を前提に、4 要素のほかに何か付け加えるものがあるかと質問しており、フェアユースに含みを持たせていたと思われる。

相互運用を認めることで技術革新を生むという公益性をフェアユースの中で考慮すべきかという問題意識を感じたが、政府は、追加の要素については言明せず、裁判所はそのような利用を認めることが消費者にとって利益になるかという点だけを考慮してはならず、将来のイノベーションに対するインセンティブへの影響も考慮しなければならないと指摘した[37]。

また、Kagan 判事は、API をそのまま複製している本件にトランスフォーマティブ（変容的）か否かというテストはそもそもミスマッチではないかと Google 代理人に対し補充質問し、同代理人から、トランスフォーマティブであること（変容性）は法文上の要件ではなく、フェアユースの成否において問題とされるべきはあくまで利用行為の性質である旨の回答を引き出している[38]。

パロディの成否を判断した Campbell 判決[39]が示したトランスフォーマティブな利用（変容的利用）か否かで説明するよりも、107 条の 4 要素以外に考慮する事情を加えるというアプローチがあり得ることを示唆している。

ウ CAFC が陪審に審理を差し戻すのではなく自らフェアユースを否定する判断をしたのは正しかったのか。

手続の瑕疵に関する論点なので本号では詳細は省略するが、口頭弁論では、フェアユースの実体的な要件以上に複数の判事がこの点について各当事者に質問をしており、関心の高さが伺えた。米国憲法修正 7 条が陪審による事実認定を受ける権利を保障していることがその背景にあるようである。

Thomas 判事、Alito 判事は著作物性に関し Oracle に有利な姿勢に傾いているように

[34] *Id.*at 73-75
[35] *Id.*at 75-77
[36] *Id.*at 44-45
[37] *Id.*at 68-69
[38] *Id.*at 89
[39] Campbell v. Acuff-Rose Music, Inc., 510 U.S. 569

思えたが[40]、この論点について両判事ともに積極的に意見を求めており、瑕疵ありと考えているのかもしれない。

Gorsuch 判事は、Oracle 代理人に対し、個別の事実関係を基礎とした利益衡量を行うフェアユースでなぜ（事実認定の権限を有する）陪審の裁量を尊重する差戻しをすべきではないのかと尋ねた。Oracle 代理人は最高裁がそう考えるのであれば差し戻すことができるとやや譲歩にも思われる回答をしている。また、Gorsuch 判事は、政府に対し、どうして合理的な陪審がフェアユースとの判断に至ることができないと言えるのか（すなわち、CAFC は、民事訴訟規則 50 条に照らし陪審判断を覆すことができないのではないか、という趣旨だと考えられる。）と質した。これに対して政府は、フェアユースが個別の事実の下で判断されることを認めながらも、トランスフォーマティブ（変容的）か否かの判断は事実認定ではなく法律判断であるから、民事訴訟規則 50 条に反しないという反論をした。それに対して Gorsuch 判事は、仮に最高裁が政府の見解を支持しないとしたら、最高裁はどうすべきかとまで尋ね、陪審に差し戻すことになるとの回答を政府から引き出している[41]。

エ　本判決の影響について

口頭弁論において顕著だったのは、判決がソフトウェア業界に与える影響について裁判官が非常に意識している点である。

例えば、Kavanagh 判事は、83 名のコンピュータサイエンティストのアミカス・ブリーフが、Google が敗訴すれば壊滅的な事態を生じるだろう（sky will fall）と指摘しているが、2014 年 CAFC 判決以降 6 年が経過してもそのような事態になっていないのではないかと尋ねた。Google 代理人は、「フェアユース抗弁で差戻し審で勝訴し、当該事件はいまだ係属中であるから CAFC 判決の影響は今は見えない。しかし、アプリケーションインターフェースは何十年も許諾なく再利用されてきており、純粋に機能的なものだと業界では考えられているので、そうでないという最高裁の判断が出れば、大変な事態になる」と回答している[42]。Kavanagh 判事は同じ質問を Oracle 代理人に対しても行っており、Google とは全く逆の回答を得ている。すわなち「ソフトウェア産業の繁栄はコンピュータプログラムを著作権により保護した結果であり、CAFC の判決後も壊滅的な事態になっていないことがそれを証明している。逆の判断をすれば、コンピュータプログラムを制作するインセンティブを失わせてしまうだろう」と[43]。

Roberts 長官は、裁判所が Oracle を勝訴させたら、米国産業を壊滅させると指摘されているが、なぜそれは事実ではないと言えるのかと政府に尋ねた。政府は、「CAFC 判決後何も悪影響が出ていない。また、インターフェースというが、そのセグメントの中には創作性のないものもあり、許諾なく複製できる場合もあるだろう。そして、ライセンス又はオープンライセンスが普及しているから、壊滅的な影響が出るというコンピュータサイエンティストの指摘は間違っている」と反論した[44・45]。

4　判決の行方

本稿では口頭弁論の様子を概観した。筆者の主観的な感触では、Breyer, Kagan, Sotomayor 各判事は Google の主張に理解を示しており、102 条 (b) に基づき API の著作物性を否定するのではないかと思われる。

これに対し、Thomas, Alito, Gorsuch 各判事は、declaring code と implementing code の

---

40 Oral argument transcripts　at 13, 19-20, 67-68
41 *Id.*at 77-79
42 *Id.* at 35-36
43 *Id.* at 61-62
44 *Id.* at 65-66
45 *Id.* at 71-72 (Alito 判事も同じ質問をしている)

区別を認めず一律に著作権で保護される対象とすると思われるが、フェアユースを認める可能性は残っているように思われた。CAFC が陪審評決を破棄し自ら判断を下したことについて手続上の瑕疵がないかに強く関心を示したのがこの 3 判事であったことは、Google に有利に働くかもしれない。

なお、Roberts 長官と Kavanaugh 判事は、その発言やトーンからは上記両翼の中間にあるように思われるが、コンピュータサイエンティストのアミカス・ブリーフの“sky will fall”を意識した発言をしており、その最終的な評価に基づき態度を決めるのかもしれない。

口頭弁論直後のウェブ上での評価では、Google 代理人の対応が不十分であったという指摘が多く、Google に不利な予測をする意見が多く見られた。

確かに、口頭弁論で、裁判官が事案を理解するための適切なアナロジー探しをしている様子が窺える中、Google 代理人が裁判官を納得させるぴったりのアナロジーを提供できなかったかもしれない。しかし、上記に見るように、必ずしも Google に一方的に不利な状況とも思えない。また、裁判官が 83 人のコンピュータサイエンティストのアミカス・ブリーフに言及し、多数のアミカス・ブリーフが非常に役に立っているとの発言もあった。多数のアミカス・ブリーフが（その中には別のアナロジーを提供するものもある）裁判官に大きく影響を与えることも十分に考えられる。

蓋を開けてみないとわからない。Ruth Bader Ginsburg 判事が 9 月 18 日に亡くなったため、本件は 8 人で審理された。かつてインターフェースの複製をめぐり争われた Lotus 事件[46・47]は最高裁で 4 対 4 となり、原審（第 1 巡回区控訴裁判所）の判断が是認される結果となったが、今回は、米国ソフトウェア著作権の長年の争点に最高裁判決で決着がつくこと（提訴から 10 年を経過しており、手続的瑕疵を理由とする差戻しも適切ではないように思う）を期待したい。

以上

[46] Lotus Development Corp. v. Borland International, Inc., 49 F.3d 807 (1st Cir. 1995), aff'd without opinion by equally divided court, 516 U.S. 233, 116 S. Ct. 804, 133 L. Ed. 2d 610 (1996)

[47] 島並良「ロータス対ボーランド事件が残したもの」知財研フォーラム 34 号 36 頁（1998）参照。http://www.lib.kobe-u.ac.jp/repository/90002658.pdf

SLN SOFTIC LAW NEWS

一般財団法人ソフトウェア情報センター
発行・編集　専務理事　亀井 正博

No. 167 (2021/4)

Google LLC v. Oracle America, Inc. 連邦最高裁判決

－ Google による Java API の複製はフェアユースに該当、下級審に差戻し ―

調査研究部

目　次

© 2021 一般財団法人ソフトウェア情報センター

https://www.softic.or.jp/

〒105-0003 東京都港区西新橋 3-16-11
愛宕イーストビル 14 階
TEL: 03-3437-3071 / E-mail: res@softic.or.jp

１　はじめに

米国連邦最高裁判所は 2021 年 4 月 5 日、Oracle America Inc.（以下「Oracle」という。）が提供する Java SE (Java Platform, Standard Edition)の API として使用される"declaring code"（以下「本件コード」という。）を Google LLC（以下「Google」という。）が許諾なく複製したとして著作権侵害が争われていた事件で、Google による本件コードの複製行為は米国著作権法第 107 条に規定される「フェアユース」に該当すると判断、本件コードの著作物性を認めた上で Google によるその複製行為がフェアユースに該当しないとして著作権侵害を認定した米国連邦控訴裁判所の判決のうち「フェアユース」に関する判断部分を破棄し、下級審に差し戻す判決を行った。

以下、本号では、判決の概要のみ、主に判決要旨（Syllabus）の記載に基づき、速報することとしたい。

なお、本件事件が連邦最高裁判所に至るまでの連邦地方裁判所及び連邦控訴裁判所における各裁判所の判断や経緯については、SLN 第 138 号及び第 165 号を参照されたい。

２　連邦最高裁判所判決の概要

Google は上告審において、複製された本件コードには著作権保護が及ばないこと（米国著作権法第 102 条(b)）及び（仮に著作権保護が及ぶとしても）著作権者は他人が著作物の「フェアユース」を行うことを妨げることはできないこと（米国著作権法第 107 条）の 2 点を主張した。各争点に対する連邦最高裁判所の判断の概要は次のとおりである。

(1)　本件コードの著作物性について

連邦最高裁判所は、「急速に変化する技術、経済、ビジネスに関する諸状況に鑑み、我々は、当事者の紛争を解決するために必要なことを超えて回答すべきではないと考える。」、「議論を進めるために、複製された本件コードは著作権保護を享受し得ると仮定して、Google のコードの使用が「フェアユース」に該当するか否かに焦点を当てる。」として、本件コードが著作物性を有するか否かの判断を回避した。

(2)　フェアユースの成否について

連邦最高裁判所は、米国著作権法第 107 条に定められる 4 つの要素についてそれぞれ下記のとおり述べ、Google による本件コードの複製は「フェアユース」に該当するとの判断を示した。

なお、連邦最高裁判所は同時に、コンピュータプログラムという主に実用品としての性質を有する著作物に伝統的な著作物の概念をあてはめることの難しさを述べつつ、本件において Google は、ユーザーインターフェイスを再実装するために API を複製したこと、ユーザー（プログラマー）がその才能を新規かつ変容的なプログラムに対して発揮することができるようにする限りにおいて複製を行っていることから、法律問題としてフェアユースを構成すると判断したものであって従前の判例を覆したり変更したりするものではないことを注記している。

ア　著作物の性質（フェアユースの第 2 要素）

複製されたコードは、プログラマーが既存のコンピュータコードに単純な命令を通じてアクセスする方法を提供するための「ユーザーインターフェイス」の一部である。結果、このコードは、コンピュータにタスクの実行を命ずるコードのような多くの他の種類のコードとは異なるものである。複製されたコードは、著作権保護を受けないアイデア（API の全ての体系）及び新たな創作的表現（Google により独立して記述されたコード）と、インターフェイスの一部として生来的に結びついている。他の多くのコンピュータプログラムとは異なり、複製されたコードの価値は、かなりの部分がユーザー（ここではコンピュータプログラマー）による投資から派生するものである。そうした相違から、ここでのフェアユースの適用は、連邦議会がコンピュータプログラムに与えた一般的な著作権保護を害するものではない。

イ　使用の目的及び性質（フェアユースの第 1 要素）

Google による API の一定の複製行為は、変容的利用である。Google は、プログラマーが使い慣れたプログラミング言語の一部を捨て去ることなく異なるコンピュータ利用環境において仕事をすることを可能にするために必要とされる部分のみを複製した。Google の目的は、異なるコンピュータ利用環境（スマートフォン）のためのタスク関連システム（task-related system）を作り出すことであり、また、その目的の達成及び普及に資するプラットフォーム、すなわち Android プラットフォームを作り出すことであった。訴訟記録によれば、インターフェイスの再実装の方法が多数存在することがコンピュータプログラムのさらなる開発を促進するとされる。したがって、Google の目的は、著作権それ自体の根本的な憲法上の目的である創造の促進と一致していた。

ウ　著作物全体との関連で使用された量及び実質性（フェアユースの第 3 要素）

Google は API から、数百の異なるタスクを呼び出すのに必要なほぼ全てとなる約 11,500 行の declaring code を複製した。しかしながら、それら 11,500 行は、争われている 286 万行から成る API 全体のわずか 0.4%である。「使用されている部分の量及び実質性」を考察するに当たっては、11,500 行のコードは、相当に大規模な全体のうちのわずかな部分に過ぎないとみるべきである。複製されたコードは、インターフェイスの一部として、他のプログラマーからアクセスされる他のコードと不可分に結びつけられている。Google は、彼らの創造性や美観によってではなく、プログラマーがその能力を新たなスマートフォンのコンピュータ利用環境に向けることを可能にするためにそれらのコードを複製した。「実質性」の要素は、本件におけるように、複製の量が正当かつ変容的な目的に結びついているとき、一般的にフェアユースに有利に働く。

エ　使用が著作物の潜在的市場又は価値に与えた影響（フェアユースの第 4 要素）

訴訟記録によれば、Google の新しいスマートフォンは市場における Java SE の代替品

ではない。また、訴訟記録によれば、Java SE の著作権者はそのインターフェイスが異なる市場に再実装されることから利益を得る可能性がある。これらの事実に対して著作権を主張することは、公衆に対して、創造性に関する害を生じさせる危険がある。

３　反対意見

Brayer、Roberts、Sotomayor、Kagan、Gorsuch、Kavanaugh の 6 名の判事による多数意見に対し、Thomas 判事及び Altio 判事は反対意見を述べ、多数意見が本件コードの著作物性に関する判断を回避していることや、フェアユースに該当すると判断したことを批判している。なお Barrett 判事は審理に参加していない。

４　今後の課題

本件判決により Google による本件コードの複製行為が「フェアユース」に該当し著作権侵害に当たらないとされたことで、Android プラットフォームに対する実務的な影響は一応は回避された格好になるが、上記２(1)に記載のとおり、連邦最高裁判所は、本件判決に当たり、本件コードの著作物性そのものに関する同裁判所としての判断を示さなかった。一般論として、米国著作権法において本件コード、ひいては API が著作権保護の対象となるのかは、大きな論点として残されたこととなる。今後の議論が引き続き注目される。

以上

Syllabus

NOTE: Where it is feasible, a syllabus (headnote) will be released, as is being done in connection with this case, at the time the opinion is issued. The syllabus constitutes no part of the opinion of the Court but has been prepared by the Reporter of Decisions for the convenience of the reader. See *United States* v. *Detroit Timber & Lumber Co.,* 200 U. S. 321, 337.

SUPREME COURT OF THE UNITED STATES

Syllabus

GOOGLE LLC *v.* ORACLE AMERICA, INC.

CERTIORARI TO THE UNITED STATES COURT OF APPEALS FOR THE FEDERAL CIRCUIT

No. 18–956. Argued October 7, 2020—Decided April 5, 2021

Oracle America, Inc., owns a copyright in Java SE, a computer platform that uses the popular Java computer programming language. In 2005, Google acquired Android and sought to build a new software platform for mobile devices. To allow the millions of programmers familiar with the Java programming language to work with its new Android platform, Google copied roughly 11,500 lines of code from the Java SE program. The copied lines are part of a tool called an Application Programming Interface (API). An API allows programmers to call upon prewritten computing tasks for use in their own programs. Over the course of protracted litigation, the lower courts have considered (1) whether Java SE's owner could copyright the copied lines from the API, and (2) if so, whether Google's copying constituted a permissible "fair use" of that material freeing Google from copyright liability. In the proceedings below, the Federal Circuit held that the copied lines are copyrightable. After a jury then found for Google on fair use, the Federal Circuit reversed, concluding that Google's copying was not a fair use as a matter of law. Prior to remand for a trial on damages, the Court agreed to review the Federal Circuit's determinations as to both copyrightability and fair use.

Held: Google's copying of the Java SE API, which included only those lines of code that were needed to allow programmers to put their accrued talents to work in a new and transformative program, was a fair use of that material as a matter of law. Pp. 11–36.

(a) Copyright and patents, the Constitution says, serve to "promote the Progress of Science and useful Arts, by securing for limited Times to Authors and Inventors the exclusive Right to their respective Writings and Discoveries." Art. I, §8, cl. 8. Copyright encourages the production of works that others might cheaply reproduce by granting the

Syllabus

author an exclusive right to produce the work for a period of time. Because such exclusivity may trigger negative consequences, Congress and the courts have limited the scope of copyright protection to ensure that a copyright holder's monopoly does not harm the public interest.

This case implicates two of the limits in the current Copyright Act. First, the Act provides that copyright protection cannot extend to "any idea, procedure, process, system, method of operation, concept, principle, or discovery . . . ." 17 U. S. C. §102(b). Second, the Act provides that a copyright holder may not prevent another person from making a "fair use" of a copyrighted work. §107. Google's petition asks the Court to apply both provisions to the copying at issue here. To decide no more than is necessary to resolve this case, the Court assumes for argument's sake that the copied lines can be copyrighted, and focuses on whether Google's use of those lines was a "fair use." Pp. 11–15.

(b) The doctrine of "fair use" is flexible and takes account of changes in technology. Computer programs differ to some extent from many other copyrightable works because computer programs always serve a functional purpose. Because of these differences, fair use has an important role to play for computer programs by providing a context-based check that keeps the copyright monopoly afforded to computer programs within its lawful bounds. Pp. 15–18.

(c) The fair use question is a mixed question of fact and law. Reviewing courts should appropriately defer to the jury's findings of underlying facts, but the ultimate question whether those facts amount to a fair use is a legal question for judges to decide *de novo*. This approach does not violate the Seventh Amendment's prohibition on courts reexamining facts tried by a jury, because the ultimate question here is one of law, not fact. The "right of trial by jury" does not include the right to have a jury resolve a fair use defense. Pp. 18–21.

(d) To determine whether Google's limited copying of the API here constitutes fair use, the Court examines the four guiding factors set forth in the Copyright Act's fair use provision: the purpose and character of the use; the nature of the copyrighted work; the amount and substantiality of the portion used in relation to the copyrighted work as a whole; and the effect of the use upon the potential market for or value of the copyrighted work. §107. The Court has recognized that some factors may prove more important in some contexts than in others. *Campbell* v. *Acuff-Rose Music, Inc.*, 510 U. S. 569, 577. Pp. 21–35.

(1) The nature of the work at issue favors fair use. The copied lines of code are part of a "user interface" that provides a way for programmers to access prewritten computer code through the use of simple commands. As a result, this code is different from many other types of code, such as the code that actually instructs the computer to

execute a task. As part of an interface, the copied lines are inherently bound together with uncopyrightable ideas (the overall organization of the API) and the creation of new creative expression (the code independently written by Google). Unlike many other computer programs, the value of the copied lines is in significant part derived from the investment of users (here computer programmers) who have learned the API's system. Given these differences, application of fair use here is unlikely to undermine the general copyright protection that Congress provided for computer programs. Pp. 21–24.

(2) The inquiry into the "the purpose and character" of the use turns in large measure on whether the copying at issue was "transformative," *i.e.*, whether it "adds something new, with a further purpose or different character." *Campbell*, 510 U. S., at 579. Google's limited copying of the API is a transformative use. Google copied only what was needed to allow programmers to work in a different computing environment without discarding a portion of a familiar programming language. Google's purpose was to create a different task-related system for a different computing environment (smartphones) and to create a platform—the Android platform—that would help achieve and popularize that objective. The record demonstrates numerous ways in which reimplementing an interface can further the development of computer programs. Google's purpose was therefore consistent with that creative progress that is the basic constitutional objective of copyright itself. Pp. 24–28.

(3) Google copied approximately 11,500 lines of declaring code from the API, which amounts to virtually all the declaring code needed to call up hundreds of different tasks. Those 11,500 lines, however, are only 0.4 percent of the entire API at issue, which consists of 2.86 million total lines. In considering "the amount and substantiality of the portion used" in this case, the 11,500 lines of code should be viewed as one small part of the considerably greater whole. As part of an interface, the copied lines of code are inextricably bound to other lines of code that are accessed by programmers. Google copied these lines not because of their creativity or beauty but because they would allow programmers to bring their skills to a new smartphone computing environment. The "substantiality" factor will generally weigh in favor of fair use where, as here, the amount of copying was tethered to a valid, and transformative, purpose. Pp. 28–30.

(4) The fourth statutory factor focuses upon the "effect" of the copying in the "market for or value of the copyrighted work." §107(4). Here the record showed that Google's new smartphone platform is not a market substitute for Java SE. The record also showed that Java SE's copyright holder would benefit from the reimplementation of its interface into a different market. Finally, enforcing the copyright on

Syllabus

these facts risks causing creativity-related harms to the public. When taken together, these considerations demonstrate that the fourth factor—market effects—also weighs in favor of fair use. Pp. 30–35.

(e) The fact that computer programs are primarily functional makes it difficult to apply traditional copyright concepts in that technological world. Applying the principles of the Court's precedents and Congress' codification of the fair use doctrine to the distinct copyrighted work here, the Court concludes that Google's copying of the API to reimplement a user interface, taking only what was needed to allow users to put their accrued talents to work in a new and transformative program, constituted a fair use of that material as a matter of law. In reaching this result, the Court does not overturn or modify its earlier cases involving fair use. Pp. 35–36.

886 F. 3d 1179, reversed and remanded.

BREYER, J., delivered the opinion of the Court, in which ROBERTS, C. J., and SOTOMAYOR, KAGAN, GORSUCH, and KAVANAUGH, JJ., joined. THOMAS, J., filed a dissenting opinion, in which ALITO, J., joined. BARRETT, J., took no part in the consideration or decision of the case.

NOTICE: This opinion is subject to formal revision before publication in the preliminary print of the United States Reports. Readers are requested to notify the Reporter of Decisions, Supreme Court of the United States, Washington, D. C. 20543, of any typographical or other formal errors, in order that corrections may be made before the preliminary print goes to press.

SUPREME COURT OF THE UNITED STATES

No. 18–956

GOOGLE LLC, PETITIONER *v.*
ORACLE AMERICA, INC.

ON WRIT OF CERTIORARI TO THE UNITED STATES COURT OF APPEALS FOR THE FEDERAL CIRCUIT

[April 5, 2021]

JUSTICE BREYER delivered the opinion of the Court.

Oracle America, Inc., is the current owner of a copyright in Java SE, a computer program that uses the popular Java computer programming language. Google, without permission, has copied a portion of that program, a portion that enables a programmer to call up prewritten software that, together with the computer's hardware, will carry out a large number of specific tasks. The lower courts have considered (1) whether Java SE's owner could copyright the portion that Google copied, and (2) if so, whether Google's copying nonetheless constituted a "fair use" of that material, thereby freeing Google from copyright liability. The Federal Circuit held in Oracle's favor (*i.e.*, that the portion is copyrightable and Google's copying did not constitute a "fair use"). In reviewing that decision, we assume, for argument's sake, that the material was copyrightable. But we hold that the copying here at issue nonetheless constituted a fair use. Hence, Google's copying did not violate the copyright law.

I

In 2005, Google acquired Android, Inc., a startup firm

that hoped to become involved in smartphone software. Google sought, through Android, to develop a software platform for mobile devices like smartphones. 886 F. 3d 1179, 1187 (CA Fed. 2018); App. 137–138, 242–243. A platform provides the necessary infrastructure for computer programmers to develop new programs and applications. One might think of a software platform as a kind of factory floor where computer programmers (analogous to autoworkers, designers, or manufacturers) might come, use sets of tools found there, and create new applications for use in, say, smartphones. (For visual explanations of "platforms" and other somewhat specialized computer-related terms, you might want to look at the material in Appendix A, *infra*.)

Google envisioned an Android platform that was free and open, such that software developers could use the tools found there free of charge. Its idea was that more and more developers using its Android platform would develop ever more Android-based applications, all of which would make Google's Android-based smartphones more attractive to ultimate consumers. Consumers would then buy and use ever more of those phones. *Oracle America, Inc.* v. *Google Inc.*, 872 F. Supp. 2d 974, 978 (ND Cal. 2012); App. 111, 464. That vision required attracting a sizeable number of skilled programmers.

At that time, many software developers understood and wrote programs using the Java programming language, a language invented by Sun Microsystems (Oracle's predecessor). 872 F. Supp. 2d, at 975, 977. About six million programmers had spent considerable time learning, and then using, the Java language. App. 228. Many of those programmers used Sun's own popular Java SE platform to develop new programs primarily for use in desktop and laptop computers. *Id.*, at 151–152, 200. That platform allowed developers using the Java language to write programs that were able to run on any desktop or laptop computer, regardless of the underlying hardware (*i.e.*, the programs were in

large part "interoperable"). 872 F. Supp. 2d, at 977. Indeed, one of Sun's slogans was "'write once, run anywhere.'" 886 F. 3d, at 1186.

Shortly after acquiring the Android firm, Google began talks with Sun about the possibility of licensing the entire Java platform for its new smartphone technology. *Oracle*, 872 F. Supp. 2d, at 978. But Google did not want to insist that all programs written on the Android platform be interoperable. 886 F. 3d, at 1187. As Android's founder explained, "[t]he whole idea about [an] open source [platform] is to have very, very few restrictions on what people can do with it," App. 659, and Sun's interoperability policy would have undermined that free and open business model. Apparently, for reasons related to this disagreement, Google's negotiations with Sun broke down. Google then built its own platform.

The record indicates that roughly 100 Google engineers worked for more than three years to create Google's Android platform software. *Id.*, at 45, 117, 212. In doing so, Google tailored the Android platform to smartphone technology, which differs from desktop and laptop computers in important ways. A smartphone, for instance, may run on a more limited battery or take advantage of GPS technology. *Id.,* at 197–198. The Android platform offered programmers the ability to program for that environment. To build the platform, Google wrote millions of lines of new code. Because Google wanted millions of programmers, familiar with Java, to be able easily to work with its new Android platform, it also copied roughly 11,500 lines of code from the Java SE program. 886 F. 3d, at 1187. The copied lines of code are part of a tool called an Application Programming Interface, or API.

What is an API? The Federal Circuit described an API as a tool that "allow[s] programmers to use . . . prewritten code to build certain functions into their own programs, rather than write their own code to perform those functions

from scratch." *Oracle America, Inc.* v. *Google, Inc.*, 750 F. 3d 1339, 1349 (2014). Through an API, a programmer can draw upon a vast library of prewritten code to carry out complex tasks. For lay persons, including judges, juries, and many others, some elaboration of this description may prove useful.

Consider in more detail just what an API does. A computer can perform thousands, perhaps millions, of different tasks that a programmer may wish to use. These tasks range from the most basic to the enormously complex. Ask the computer, for example, to tell you which of two numbers is the higher number or to sort one thousand numbers in ascending order, and it will instantly give you the right answer. An API divides and organizes the world of computing tasks in a particular way. Programmers can then use the API to select the particular task that they need for their programs. In Sun's API (which we refer to as the Sun Java API), each individual task is known as a "method." The API groups somewhat similar methods into larger "classes," and groups somewhat similar classes into larger "packages." This method-class-package organizational structure is referred to as the Sun Java API's "structure, sequence, and organization," or SSO.

For each task, there is computer code, known as "implementing code," that in effect tells the computer how to execute the particular task you have asked it to perform (such as telling you, of two numbers, which is the higher). See *Oracle*, 872 F. Supp. 2d, at 979–980. The implementing code (which Google independently wrote) is not at issue here. For a single task, the implementing code may be hundreds of lines long. It would be difficult, perhaps impossible, for a programmer to create complex software programs without drawing on prewritten task-implementing programs to execute discrete tasks.

But how do you as the programmer tell the computer which of the implementing code programs it should choose,

i.e., which task it should carry out? You do so by entering into your own program a command that corresponds to the specific task and calls it up. Those commands, known as "method calls," help you carry out the task by choosing those programs written in implementing code that will do the trick, *i.e.*, that will instruct the computer so that your program will find the higher of two numbers. If a particular computer might perform, say, a million different tasks, different method calls will tell the computer which of those tasks to choose. Those familiar with the Java language already know countless method calls that allow them to invoke countless tasks.

And how does the method call (which a programmer types) actually locate and invoke the particular implementing code that it needs to instruct the computer how to carry out a particular task? It does so through another type of code, which the parties have labeled "declaring code." Declaring code is part of the API. For each task, the specific command entered by the programmer matches up with specific declaring code inside the API. That declaring code provides both the name for each task and the location of each task within the API's overall organizational system (*i.e.,* the placement of a method within a particular class and the placement of a class within a particular package). In this sense, the declaring code and the method call form a link, allowing the programmer to draw upon the thousands of prewritten tasks, written in implementing code. See *id.,* at 979–980. Without that declaring code, the method calls entered by the programmer would not call up the implementing code.

The declaring code therefore performs at least two important functions in the Sun Java API. The first, more obvious, function is that the declaring code enables a set of shortcuts for programmers. By connecting complex implementing code with method calls, it allows a programmer to

pick out from the API's task library a particular task without having to learn anything more than a simple command. For example, a programmer building a new application for personal banking may wish to use various tasks to, say, calculate a user's balance or authenticate a password. To do so, she need only learn the method calls associated with those tasks. In this way, the declaring code's shortcut function is similar to a gas pedal in a car that tells the car to move faster or the QWERTY keyboard on a typewriter that calls up a certain letter when you press a particular key. As those analogies demonstrate, one can think of the declaring code as part of an *interface* between human beings and a machine.

The second, less obvious, function is to reflect the way in which Java's creators have divided the potential world of different tasks into an actual world, *i.e.*, precisely which set of potentially millions of different tasks we want to have our Java-based computer systems perform and how we want those tasks arranged and grouped. In this sense, the declaring code performs an organizational function. It determines the structure of the task library that Java's creators have decided to build. To understand this organizational system, think of the Dewey Decimal System that categorizes books into an accessible system or a travel guide that arranges a city's attractions into different categories. Language itself provides a rough analogy to the declaring code's organizational feature, for language itself divides into sets of concepts a world that in certain respects other languages might have divided differently. The developers of Java, for example, decided to place a method called "draw image" inside of a class called "graphics."

Consider a comprehensive, albeit farfetched, analogy that illustrates how the API is actually used by a programmer. Imagine that you can, via certain keystrokes, instruct a robot to move to a particular file cabinet, to open a certain drawer, and to pick out a specific recipe. With the proper

recipe in hand, the robot then moves to your kitchen and gives it to a cook to prepare the dish. This example mirrors the API's task-related organizational system. Through your simple command, the robot locates the right recipe and hands it off to the cook. In the same way, typing in a method call prompts the API to locate the correct implementing code and hand it off to your computer. And importantly, to select the dish that you want for your meal, you do not need to know the recipe's contents, just as a programmer using an API does not need to learn the implementing code. In both situations, learning the simple command is enough.

Now let us consider the example that the District Court used to explain the precise technology here. *Id.,* at 980–981. A programmer wishes, as part of her program, to determine which of two integers is the larger. To do so in the Java language, she will first write **java.lang**. Those words (which we have put in bold type) refer to the "package" (or by analogy to the file cabinet). She will then write **Math**. That word refers to the "class" (or by analogy to the drawer). She will then write **max**. That word refers to the "method" (or by analogy to the recipe). She will then make two parentheses ( ). And, in between the parentheses she will put two integers, say 4 and 6, that she wishes to compare. The whole expression—the method call—will look like this: "**java.lang.Math.max(4, 6)**." The use of this expression will, by means of the API, call up a task-implementing program that will determine the higher number.

In writing this program, the programmer will use the very symbols we have placed in bold in the precise order we have placed them. But the symbols by themselves do nothing. She must also use software that connects the symbols to the equivalent of file cabinets, drawers, and files. The API is that software. It includes both the declaring code that links each part of the method call to the particular task-implementing program, and the implementing code

that actually carries it out. (For an illustration of this technology, see Appendix B, *infra*.)

Now we can return to the copying at issue in this case. Google did not copy the task-implementing programs, or implementing code, from the Sun Java API. It wrote its own task-implementing programs, such as those that would determine which of two integers is the greater or carry out any other desired (normally far more complex) task. This implementing code constitutes the vast majority of both the Sun Java API and the API that Google created for Android. App. 212. For most of the packages in its new API, Google also wrote its own declaring code. For 37 packages, however, Google copied the declaring code from the Sun Java API. *Id.*, at 106–107. As just explained, that means that, for those 37 packages, Google necessarily copied both the names given to particular tasks and the grouping of those tasks into classes and packages.

In doing so, Google copied that portion of the Sun Java API that allowed programmers expert in the Java programming language to use the "task calling" system that they had already learned. As Google saw it, the 37 packages at issue included those tasks that were likely to prove most useful to programmers working on applications for mobile devices. In fact, "three of these packages were . . . fundamental to being able to use the Java language at all." *Oracle*, 872 F. Supp. 2d, at 982. By using the same declaring code for those packages, programmers using the Android platform can rely on the method calls that they are already familiar with to call up particular tasks (*e.g.*, determining which of two integers is the greater); but Google's own implementing programs carry out those tasks. Without that copying, programmers would need to learn an entirely new system to call up the same tasks.

We add that the Android platform has been successful. Within five years of its release in 2007, Android-based devices claimed a large share of the United States market.

Id., at 978. As of 2015, Android sales produced more than $42 billion in revenue. 886 F. 3d, at 1187.

In 2010 Oracle Corporation bought Sun. Soon thereafter Oracle brought this lawsuit in the United States District Court for the Northern District of California.

II

The case has a complex and lengthy history. At the outset Oracle complained that Google's use of the Sun Java API violated both copyright and patent laws. For its copyright claim, Oracle alleged that Google infringed its copyright by copying, for 37 packages, both the literal declaring code and the nonliteral organizational structure (or SSO) of the API, *i.e.,* the grouping of certain methods into classes and certain classes into packages. For trial purposes the District Court organized three proceedings. The first would cover the copyright issues, the second would cover the patent issues, and the third would, if necessary, calculate damages. *Oracle*, 872 F. Supp. 2d, at 975. The court also determined that a judge should decide whether copyright law could protect an API and that the jury should decide whether Google's use of Oracle's API infringed its copyright and, if so, whether a fair use defense nonetheless applied. *Ibid.*

After six weeks of hearing evidence, the jury rejected Oracle's patent claims (which have since dropped out of the case). It also found a limited copyright infringement. It deadlocked as to whether Google could successfully assert a fair use defense. *Id.,* at 976. The judge then decided that, regardless, the API's declaring code was not the kind of creation to which copyright law extended its protection. The court noted that Google had written its own implementing code, which constituted the vast majority of its API. It wrote that "anyone is free under the Copyright Act to write his or her own code to carry out exactly the same" tasks that the Sun Java API picks out or specifies. *Ibid.* Google copied

only the declaring code and organizational structure that was necessary for Java-trained programmers to activate familiar tasks (while, as we said, writing its own implementing code). Hence the copied material, in the judge's view, was a "system or method of operation," which copyright law specifically states cannot be copyrighted. *Id.,* at 977 (citing 17 U. S. C. §102(b)).

On appeal, the Federal Circuit reversed. That court held that both the API's declaring code and its organizational structure could be copyrighted. *Oracle*, 750 F. 3d, at 1354. It pointed out that Google could have written its own declaring code just as it wrote its own implementing code. And because in principle Google might have created a whole new system of dividing and labeling tasks that could be called up by programmers, the declaring code (and the system) that made up the Sun Java API was copyrightable. *Id.,* at 1361.

The Federal Circuit also rejected Oracle's plea that it decide whether Google had the right to use the Sun Java API because doing so was a "fair use," immune from copyright liability. The Circuit wrote that fair use "both permits and requires 'courts to avoid rigid application of the copyright statute when, on occasion, it would stifle the very creativity which that law is designed to foster.'" *Id.,* at 1372–1373. But, it added, this "is not a case in which the record contains sufficient factual findings upon which we could base a de novo assessment of Google's affirmative defense of fair use." *Id.,* at 1377. And it remanded the case for another trial on that question. Google petitioned this Court for a writ of certiorari, seeking review of the Federal Circuit's copyrightability determination. We denied the petition. *Google, Inc.* v. *Oracle America, Inc.*, 576 U. S. 1071 (2015).

On remand the District Court, sitting with a jury, heard evidence for a week. The court instructed the jury to answer one question: Has Google "shown by a preponderance of the evidence that its use in Android" of the declaring code

and organizational structure contained in the 37 Sun Java API packages that it copied "constitutes a 'fair use' under the Copyright Act?" App. 294. After three days of deliberation the jury answered the question in the affirmative. *Id.*, at 295. Google had shown fair use.

Oracle again appealed to the Federal Circuit. And the Circuit again reversed the District Court. The Federal Circuit assumed all factual questions in Google's favor. But, it said, the question whether those facts constitute a "fair use" is a question of law. 886 F. 3d, at 1193. Deciding that question of law, the court held that Google's use of the Sun Java API was not a fair use. It wrote that "[t]here is nothing fair about taking a copyrighted work verbatim and using it for the same purpose and function as the original in a competing platform." *Id.,* at 1210. It remanded the case again, this time for a trial on damages.

Google then filed a petition for certiorari in this Court. It asked us to review the Federal Circuit's determinations as to both copyrightability and fair use. We granted its petition.

III

A

Copyright and patents, the Constitution says, are to "promote the Progress of Science and useful Arts, by securing for limited Times to Authors and Inventors the exclusive Right to their respective Writings and Discoveries." Art. I, §8, cl. 8. Copyright statutes and case law have made clear that copyright has practical objectives. It grants an author an exclusive right to produce his work (sometimes for a hundred years or more), not as a special reward, but in order to encourage the production of works that others might reproduce more cheaply. At the same time, copyright has negative features. Protection can raise prices to consumers. It can impose special costs, such as the cost of contacting owners to obtain reproduction permission. And the

exclusive rights it awards can sometimes stand in the way of others exercising their own creative powers. See generally *Twentieth Century Music Corp.* v. *Aiken*, 422 U. S. 151, 156 (1975); *Mazer* v. *Stein*, 347 U. S. 201, 219 (1954).

Macaulay once said that the principle of copyright is a "tax on readers for the purpose of giving a bounty to writers." T. Macaulay, Speeches on Copyright 25 (E. Miller ed. 1913). Congress, weighing advantages and disadvantages, will determine the more specific nature of the tax, its boundaries and conditions, the existence of exceptions and exemptions, all by exercising its own constitutional power to write a copyright statute.

Four provisions of the current Copyright Act are of particular relevance in this case. First, a definitional provision sets forth three basic conditions for obtaining a copyright. There must be a "wor[k] of authorship," that work must be "original," and the work must be "fixed in any tangible medium of expression." 17 U. S. C. §102(a); see also *Feist Publications, Inc.* v. *Rural Telephone Service Co.*, 499 U. S. 340, 345 (1991) (explaining that copyright requires some original "creative spark" and therefore does not reach the facts that a particular expression describes).

Second, the statute lists certain kinds of works that copyright can protect. They include "literary," "musical," "dramatic," "motion pictur[e]," "architectural," and certain other works. §102(a). In 1980, Congress expanded the reach of the Copyright Act to include computer programs. And it defined "computer program" as "'a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result.'" §10, 94 Stat. 3028 (codified at 17 U. S. C. §101).

Third, the statute sets forth limitations on the works that can be copyrighted, including works that the definitional provisions might otherwise include. It says, for example, that copyright protection cannot be extended to "any idea, procedure, process, system, method of operation, concept,

principle, or discovery . . . ." §102(b). These limitations, along with the need to "fix" a work in a "tangible medium of expression," have often led courts to say, in shorthand form, that, unlike patents, which protect novel and useful ideas, copyrights protect "expression" but not the "ideas" that lie behind it. See *Sheldon* v. *Metro-Goldwyn Pictures Corp.*, 81 F. 2d 49, 54 (CA2 1936) (Hand, J.); B. Kaplan, An Unhurried View of Copyright 46–52 (1967).

Fourth, Congress, together with the courts, has imposed limitations upon the scope of copyright protection even in respect to works that are entitled to a copyright. For example, the Copyright Act limits an author's exclusive rights in performances and displays, §110, or to performances of sound recordings, §114. And directly relevant here, a copyright holder cannot prevent another person from making a "fair use" of copyrighted material. §107.

We have described the "fair use" doctrine, originating in the courts, as an "equitable rule of reason" that "permits courts to avoid rigid application of the copyright statute when, on occasion, it would stifle the very creativity which that law is designed to foster." *Stewart* v. *Abend*, 495 U. S. 207, 236 (1990) (internal quotation marks omitted). The statutory provision that embodies the doctrine indicates, rather than dictates, how courts should apply it. The provision says:

> "[T]he fair use of a copyrighted work, . . . for purposes such as criticism, comment, news reporting, teaching . . . scholarship, or research, is not an infringement of copyright. In determining whether the use made of a work in any particular case is a fair use the factors to be considered shall include—
>
> "(1) the purpose and character of the use, including whether such use is of a commercial nature or is for nonprofit educational purposes;
>
> "(2) the nature of the copyrighted work;

> "(3) the amount and substantiality of the portion used in relation to the copyrighted work as a whole; and
>
> "(4) the effect of the use upon the potential market for or value of the copyrighted work." §107.

In applying this provision, we, like other courts, have understood that the provision's list of factors is not exhaustive (note the words "include" and "including"), that the examples it sets forth do not exclude other examples (note the words "such as"), and that some factors may prove more important in some contexts than in others. See *Campbell* v. *Acuff-Rose Music, Inc.*, 510 U. S. 569, 577 (1994); *Harper & Row, Publishers, Inc.* v. *Nation Enterprises*, 471 U. S. 539, 560 (1985); see also Leval, Toward a Fair Use Standard, 103 Harv. L. Rev 1105, 1110 (1990) (Leval) ("The factors do not represent a score card that promises victory to the winner of the majority"). In a word, we have understood the provision to set forth general principles, the application of which requires judicial balancing, depending upon relevant circumstances, including "significant changes in technology." *Sony Corp. of America* v. *Universal City Studios, Inc.*, 464 U. S. 417, 430 (1984); see also *Aiken*, 422 U. S., at 156 ("When technological change has rendered its literal terms ambiguous, the Copyright Act must be construed in light of its basic purpose").

B

Google's petition for certiorari poses two questions. The first asks whether Java's API is copyrightable. It asks us to examine two of the statutory provisions just mentioned, one that permits copyrighting computer programs and the other that forbids copyrighting, *e.g.*, "process[es]," "system[s]," and "method[s] of operation." Pet. for Cert. 12. Google believes that the API's declaring code and organization fall into these latter categories and are expressly excluded from copyright protection. The second question asks us to determine whether Google's use of the API was a "fair

use." Google believes that it was.

A holding for Google on either question presented would dispense with Oracle's copyright claims. Given the rapidly changing technological, economic, and business-related circumstances, we believe we should not answer more than is necessary to resolve the parties' dispute. We shall assume, but purely for argument's sake, that the entire Sun Java API falls within the definition of that which can be copyrighted. We shall ask instead whether Google's use of part of that API was a "fair use." Unlike the Federal Circuit, we conclude that it was.

IV

The language of §107, the "fair use" provision, reflects its judge-made origins. It is similar to that used by Justice Story in *Folsom* v. *Marsh*, 9 F. Cas. 342, 348 (No. 4,901) (CC Mass. 1841). See *Campbell*, 510 U. S., at 576 (noting how "Justice Story's summary [of fair use considerations] is discernable" in §107). That background, as well as modern courts' use of the doctrine, makes clear that the concept is flexible, that courts must apply it in light of the sometimes conflicting aims of copyright law, and that its application may well vary depending upon context. Thus, copyright's protection may be stronger where the copyrighted material is fiction, not fact, where it consists of a motion picture rather than a news broadcast, or where it serves an artistic rather than a utilitarian function. See, *e.g.*, *Stewart*, 495 U. S., at 237–238; *Harper & Row*, 471 U. S., at 563; see also 4 M. Nimmer & D. Nimmer, Copyright §13.05[A][2][a] (2019) (hereinafter Nimmer on Copyright) ("[C]opyright protection is narrower, and the corresponding application of the fair use defense greater, in the case of factual works than in the case of works of fiction or fantasy"). Similarly, courts have held that in some circumstances, say, where copyrightable material is bound up with uncopyrightable material, copyright protection is "thin." See *Feist*,

499 U. S., at 349 (noting that "the copyright in a factual compilation is thin"); see also *Experian Information Solutions, Inc.* v. *Nationwide Marketing Servs. Inc.*, 893 F. 3d 1176, 1186 (CA9 2018) ("In the context of factual compilations, . . . there can be no infringement unless the works are virtually identical" (internal quotation marks omitted)).

Generically speaking, computer programs differ from books, films, and many other "literary works" in that such programs almost always serve functional purposes. These and other differences have led at least some judges to complain that "applying copyright law to computer programs is like assembling a jigsaw puzzle whose pieces do not quite fit." *Lotus Development Corp.* v. *Borland Int'l, Inc.*, 49 F. 3d 807, 820 (CA1 1995) (Boudin, J., concurring).

These differences also led Congress to think long and hard about whether to grant computer programs copyright protection. In 1974, Congress established a National Commission on New Technological Uses of Copyrighted Works (CONTU) to look into the matter. §§201–208, 88 Stat. 1873–1875. After several years of research, CONTU concluded that the "availability of copyright protection for computer programs is desirable." Final Report 11 (July 31, 1978). At the same time, it recognized that computer programs had unique features. Mindful of not "unduly burdening users of programs and the general public," it wrote that copyright "should not grant anyone more economic power than is necessary to achieve the incentive to create." *Id.,* at 12. And it believed that copyright's existing doctrines (*e.g.*, fair use), applied by courts on a case-by-case basis, could prevent holders from using copyright to stifle innovation. *Ibid.* ("Relatively few changes in the Copyright Act of 1976 are required to attain these objectives"). Congress then wrote computer program protection into the law. See §10, 94 Stat. 3028.

The upshot, in our view, is that fair use can play an important role in determining the lawful scope of a computer

program copyright, such as the copyright at issue here. It can help to distinguish among technologies. It can distinguish between expressive and functional features of computer code where those features are mixed. It can focus on the legitimate need to provide incentives to produce copyrighted material while examining the extent to which yet further protection creates unrelated or illegitimate harms in other markets or to the development of other products. In a word, it can carry out its basic purpose of providing a context-based check that can help to keep a copyright monopoly within its lawful bounds. See H. R. Rep. No. 94–1476, pp. 65–66 (1976) (explaining that courts are to "adapt the doctrine [of fair use] to particular situations on a case-by-case basis" and in light of "rapid technological change"); see, *e.g., Lexmark Int'l, Inc.* v. *Static Control Components, Inc.*, 387 F. 3d 522, 543–545 (CA6 2004) (discussing fair use in the context of copying to preserve compatibility); *Sony Computer Entertainment, Inc.* v. *Connectix Corp.*, 203 F. 3d 596, 603–608 (CA9 2000) (applying fair use to intermediate copying necessary to reverse engineer access to unprotected functional elements within a program); *Sega Enterprises Ltd.* v. *Accolade, Inc.*, 977 F. 2d 1510, 1521–1527 (CA9 1992) (holding that wholesale copying of copyrighted code as a preliminary step to develop a competing product was a fair use).

JUSTICE THOMAS' thoughtful dissent offers a very different view of how (and perhaps whether) fair use has any role to play for computer programs. We are told that no attempt to distinguish among computer code is tenable when considering "the nature of the work," see *post*, at 10, even though there are important distinctions in the ways that programs are used and designed, *post*, at 18 ("The declaring code is what attracted programmers"). We are told that no reuse of code in a new program will ever have a valid "purpose and character," *post*, at 16, even though the reasons for copying computer code may vary greatly and differ from

those applicable to other sorts of works, *ibid.* (accepting that copying as part of "reverse engineer[ing] a system to ensure compatibility" could be a valid purpose). And we are told that our fair use analysis must prioritize certain factors over others, *post*, at 9, n. 5, even though our case law instructs that fair use depends on the context, see *Campbell*, 510 U. S., at 577–578.

We do not understand Congress, however, to have shielded computer programs from the ordinary application of copyright's limiting doctrines in this way. By defining computer programs in §101, Congress chose to place this subject matter within the copyright regime. Like other protected works, that means that the owners of computer programs enjoy the exclusive rights set forth in the Act, including the right to "reproduce [a] copyrighted work" or to "prepare derivative works." 17 U. S. C. §106. But that also means that exclusive rights in computer programs are limited like any other works. Just as fair use distinguishes among books and films, which are indisputably subjects of copyright, so too must it draw lines among computer programs. And just as fair use takes account of the market in which scripts and paintings are bought and sold, so too must it consider the realities of how technological works are created and disseminated. We do not believe that an approach close to "all or nothing" would be faithful to the Copyright Act's overall design.

V

At the outset, Google argues that "fair use" is a question for a jury to decide; here the jury decided the question in Google's favor; and we should limit our review to determining whether "substantial evidence" justified the jury's decision. The Federal Circuit disagreed. It thought that the "fair use" question was a mixed question of fact and law; that reviewing courts should appropriately defer to the jury's findings of underlying facts; but that the ultimate

question whether those facts showed a "fair use" is a legal question for judges to decide *de novo*.

We agree with the Federal Circuit's answer to this question. We have said, "[f]air use is a mixed question of law and fact." *Harper & Row*, 471 U. S., at 560. We have explained that a reviewing court should try to break such a question into its separate factual and legal parts, reviewing each according to the appropriate legal standard. But when a question can be reduced no further, we have added that "the standard of review for a mixed question all depends—on whether answering it entails primarily legal or factual work." *U. S. Bank N. A.* v. *Village at Lakeridge, LLC*, 583 U. S. ___, ___(2018) (slip op., at 9).

In this case, the ultimate "fair use" question primarily involves legal work. "Fair use" was originally a concept fashioned by judges. *Folsom*, 9 F. Cas., at 348. Our cases still provide legal interpretations of the fair use provision. And those interpretations provide general guidance for future cases. See, *e.g.*, *Campbell*, 510 U. S., at 592–593 (describing kinds of market harms that are not the concern of copyright); *Harper & Row*, 471 U. S., at 564 ("scope of fair use is narrower with respect to unpublished works"); *Sony*, 464 U. S., at 451 (wholesale copying aimed at creating a market substitute is presumptively unfair). This type of work is legal work. *U. S. Bank*, 583 U. S., at ___ (slip op., at 8) ("When applying the law involves developing auxiliary legal principles for use in other cases[,] appellate courts should typically review a decision *de novo*").

Applying a legal "fair use" conclusion may, of course, involve determination of subsidiary factual questions, such as "whether there was harm to the actual or potential markets for the copyrighted work" or "how much of the copyrighted work was copied." 886 F. 3d, at 1196; see, *e.g., Peter F. Gaito Architecture, LLC* v. *Simone Development Corp.*, 602 F. 3d 57, 63 (CA2 2010) (noting that in an infringement suit "the question of substantial similarity typically presents an

extremely close question of fact"). In this case the Federal Circuit carefully applied the fact/law principles we set forth in *U. S. Bank*, leaving factual determinations to the jury and reviewing the ultimate question, a legal question, *de novo*.

Next, Google argues that the Federal Circuit's approach violates the Seventh Amendment. The Amendment both requires that "the right of trial by jury . . . be preserved" and forbids courts to "re-examin[e]" any "fact tried by a jury." U. S. Const., Amdt. 7; see also *Gasperini* v. *Center for Humanities, Inc.*, 518 U. S. 415, 432–433 (1996). The Reexamination Clause is no bar here, however, for, as we have said, the ultimate question here is one of law, not fact. It does not violate the Reexamination Clause for a court to determine the controlling law in resolving a challenge to a jury verdict, as happens any time a court resolves a motion for judgment as a matter of law. See, *e.g., Neely* v. *Martin K. Eby Constr. Co.*, 386 U. S. 317, 322 (1967).

Nor is Google correct that "the right of trial by jury" includes the right to have a jury resolve a fair use defense. That Clause is concerned with "the particular trial decision" at issue. *Markman* v. *Westview Instruments, Inc.*, 517 U. S. 370, 376 (1996). Even though it is possible to find pre-Revolutionary English cases in which a judge sent related questions like fair abridgment to a jury, those questions were significantly different from the "fair use" doctrine as courts apply it today. See, *e.g.*, *Gyles* v. *Wilcox*, 2 Atk. 141, 142–144, 26 Eng. Rep. 489, 490–491 (Ch. 1740) (asking the Court to resolve the narrow question whether a shortened work could be considered a new work); *Sayre* v. *Moore*, 1 East 361, n., 102 Eng. Rep. 138, 139, n. (K. B. 1785) (discussing the jury's role in resolving whether copying constituted infringement). As far as contemporary fair use is concerned, we have described the doctrine as an "equitable," not a "legal," doctrine. We have found no case suggesting that application of *U. S. Bank* here would fail "to preserve

the substance of the common-law [jury trial] right as it existed in 1791." *Markman*, 517 U. S., at 376.

VI

We turn now to the basic legal question before us: Was Google's copying of the Sun Java API, specifically its use of the declaring code and organizational structure for 37 packages of that API, a "fair use." In answering this question, we shall consider the four factors set forth in the fair use statute as we find them applicable to the kind of computer programs before us. We have reproduced those four statutory factors *supra*, at 13–14. For expository purposes, we begin with the second.

A. "The Nature of the Copyrighted Work"

The Sun Java API is a "user interface." It provides a way through which users (here the programmers) can "manipulate and control" task-performing computer programs "via a series of menu commands." *Lotus Development Corp.*, 49 F. 3d, at 809. The API reflects Sun's division of possible tasks that a computer might perform into a set of actual tasks that certain kinds of computers actually will perform. Sun decided, for example, that its API would call up a task that compares one integer with another to see which is the larger. Sun's API (to our knowledge) will not call up the task of determining which great Arabic scholar decided to use Arabic numerals (rather than Roman numerals) to perform that "larger integer" task. No one claims that the decisions about what counts as a task are themselves copyrightable—although one might argue about decisions as to how to label and organize such tasks (*e.g.*, the decision to name a certain task "max" or to place it in a class called "Math." Cf. *Baker* v. *Selden*, 101 U. S. 99 (1880)).

As discussed above, *supra,* at 3–5, and in Appendix B, *infra*, we can think of the technology as having three essential parts. First, the API includes "implementing code," which

actually instructs the computer on the steps to follow to carry out each task. Google wrote its own programs (implementing programs) that would perform each one of the tasks that its API calls up.

Second, the Sun Java API associates a particular command, called a "method call," with the calling up of each task. The symbols **java.lang.**, for example, are part of the command that will call up the program (whether written by Sun or, as here, by Google) that instructs the computer to carry out the "larger number" operation. Oracle does not here argue that the use of these commands by programmers itself violates its copyrights.

Third, the Sun Java API contains computer code that will associate the writing of a method call with particular "places" in the computer that contain the needed implementing code. This is the declaring code. The declaring code both labels the particular tasks in the API and organizes those tasks, or "methods," into "packages" and "classes." We have referred to this organization, by way of rough analogy, as file cabinets, drawers, and files. Oracle does claim that Google's use of the Sun Java API's declaring code violates its copyrights.

The declaring code at issue here resembles other copyrighted works in that it is part of a computer program. Congress has specified that computer programs are subjects of copyright. It differs, however, from many other kinds of copyrightable computer code. It is inextricably bound together with a general system, the division of computing tasks, that no one claims is a proper subject of copyright. It is inextricably bound up with the idea of organizing tasks into what we have called cabinets, drawers, and files, an idea that is also not copyrightable. It is inextricably bound up with the use of specific commands known to programmers, known here as method calls (such as **java.lang.Math.max**, etc.), that Oracle does not here contest. And it is inextricably bound up with implementing

code, which is copyrightable but was not copied.

Moreover, the copied declaring code and the uncopied implementing programs call for, and reflect, different kinds of capabilities. A single implementation may walk a computer through dozens of different steps. To write implementing programs, witnesses told the jury, requires balancing such considerations as how quickly a computer can execute a task or the likely size of the computer's memory. One witness described that creativity as "magic" practiced by an API developer when he or she worries "about things like power management" for devices that "run on a battery." App. 143; see also *id.*, at 147, 204. This is the very creativity that was needed to develop the Android software for use not in laptops or desktops but in the very different context of smartphones.

The declaring code (inseparable from the programmer's method calls) embodies a different kind of creativity. Sun Java's creators, for example, tried to find declaring code names that would prove intuitively easy to remember. *Id.*, at 211. They wanted to attract programmers who would learn the system, help to develop it further, and prove reluctant to use another. See *post*, at 10 ("Declaring code . . . is user facing. It must be designed and organized in a way that is intuitive and understandable to developers so that they can invoke it"). Sun's business strategy originally emphasized the importance of using the API to attract programmers. It sought to make the API "open" and "then . . . compete on implementations." App. 124–125. The testimony at trial was replete with examples of witnesses drawing this critical line between the user-centered declaratory code and the innovative implementing code. *Id.*, at 126–127, 159–160, 163–164, 187, 190–191.

These features mean that, as part of a user interface, the declaring code differs to some degree from the mine run of computer programs. Like other computer programs, it is functional in nature. But unlike many other programs, its

use is inherently bound together with uncopyrightable ideas (general task division and organization) and new creative expression (Android's implementing code). Unlike many other programs, its value in significant part derives from the value that those who do not hold copyrights, namely, computer programmers, invest of their own time and effort to learn the API's system. And unlike many other programs, its value lies in its efforts to encourage programmers to learn and to use that system so that they will use (and continue to use) Sun-related implementing programs that Google did not copy.

Although copyrights protect many different kinds of writing, Leval 1116, we have emphasized the need to "recogni[ze] that some works are closer to the core of [copyright] than others," *Campbell*, 510 U. S., at 586. In our view, for the reasons just described, the declaring code is, if copyrightable at all, further than are most computer programs (such as the implementing code) from the core of copyright. That fact diminishes the fear, expressed by both the dissent and the Federal Circuit, that application of "fair use" here would seriously undermine the general copyright protection that Congress provided for computer programs. And it means that this factor, "the nature of the copyrighted work," points in the direction of fair use.

B. "The Purpose and Character of the Use"

In the context of fair use, we have considered whether the copier's use "adds something new, with a further purpose or different character, altering" the copyrighted work "with new expression, meaning or message." *Id.*, at 579. Commentators have put the matter more broadly, asking whether the copier's use "fulfill[s] the objective of copyright law to stimulate creativity for public illumination." Leval 1111. In answering this question, we have used the word "transformative" to describe a copying use that adds something new and important. *Campbell*, 510 U. S., at 579. An

"'artistic painting'" might, for example, fall within the scope of fair use even though it precisely replicates a copyrighted "'advertising logo to make a comment about consumerism.'" 4 Nimmer on Copyright §13.05[A][1][b] (quoting Netanel, Making Sense of Fair Use, 15 Lewis & Clark L. Rev. 715, 746 (2011)). Or, as we held in *Campbell*, a parody can be transformative because it comments on the original or criticizes it, for "[p]arody needs to mimic an original to make its point." 510 U. S., at 580–581.

Google copied portions of the Sun Java API precisely, and it did so in part for the same reason that Sun created those portions, namely, to enable programmers to call up implementing programs that would accomplish particular tasks. But since virtually any unauthorized use of a copyrighted computer program (say, for teaching or research) would do the same, to stop here would severely limit the scope of fair use in the functional context of computer programs. Rather, in determining whether a use is "transformative," we must go further and examine the copying's more specifically described "purpose[s]" and "character." 17 U. S. C. §107(1).

Here Google's use of the Sun Java API seeks to create new products. It seeks to expand the use and usefulness of Android-based smartphones. Its new product offers programmers a highly creative and innovative tool for a smartphone environment. To the extent that Google used parts of the Sun Java API to create a new platform that could be readily used by programmers, its use was consistent with that creative "progress" that is the basic constitutional objective of copyright itself. Cf. *Feist*, 499 U. S., at 349–350 ("The primary objective of copyright is not to reward the labor of authors, but '[t]o promote the Progress of Science and useful Arts'" (quoting U. S. Const., Art. I, §8, cl. 8)).

The jury heard that Google limited its use of the Sun Java API to tasks and specific programming demands related to

Android. It copied the API (which Sun created for use in desktop and laptop computers) only insofar as needed to include tasks that would be useful in smartphone programs. App. 169–170. And it did so only insofar as needed to allow programmers to call upon those tasks without discarding a portion of a familiar programming language and learning a new one. *Id.*, at 139–140. To repeat, Google, through Android, provided a new collection of tasks operating in a distinct and different computing environment. Those tasks were carried out through the use of new implementing code (that Google wrote) designed to operate within that new environment. Some of the *amici* refer to what Google did as "reimplementation," defined as the "building of a system . . . that repurposes the same words and syntaxes" of an existing system—in this case so that programmers who had learned an existing system could put their basic skills to use in a new one. Brief for R Street Institute et al. as *Amici Curiae* 2.

The record here demonstrates the numerous ways in which reimplementing an interface can further the development of computer programs. The jury heard that shared interfaces are necessary for different programs to speak to each other. App. 125 ("We have to agree on the APIs so that the application I write to show a movie runs on your device"). It heard that the reimplementation of interfaces is necessary if programmers are to be able to use their acquired skills. *Id.*, at 191 ("If the API labels change, then either the software wouldn't continue to work anymore or the developer . . . would have to learn a whole new language to be able to use these API labels"). It heard that the reuse of APIs is common in the industry. *Id.*, at 115, 155, 663. It heard that Sun itself had used pre-existing interfaces in creating Java. *Id.*, at 664. And it heard that Sun executives thought that widespread use of the Java programming language, including use on a smartphone platform, would benefit the company. *Id.*, at 130–133.

Amici supporting Google have summarized these same points—points that witnesses explained to the jury. See, *e.g.*, Brief for Copyright Scholars as *Amici Curiae* 25 ("[T]he portions of Java SE that Google reimplemented may have helped preserve consistency of use within the larger Java developer community"); Brief for Microsoft Corporation as *Amicus Curiae* 22 ("[A]llowing reasonable fair use of functional code enables innovation that creates new opportunities for the whole market to grow"); Brief for 83 Computer Scientists as *Amici Curiae* 20 ("Reimplementing interfaces fueled widespread adoption of popular programming languages" (emphasis deleted)); Brief for R Street Institute et al. as *Amici Curiae* 15–20 (describing Oracle's reimplementation of other APIs); see also Brief for American Antitrust Institute as *Amicus Curiae* 7 ("Copyright on largely functional elements of software that [have] become an industry standard gives a copyright holder anti-competitive power").

These and related facts convince us that the "purpose and character" of Google's copying was transformative—to the point where this factor too weighs in favor of fair use.

There are two other considerations that are often taken up under the first factor: commerciality and good faith. The text of §107 includes various noncommercial uses, such as teaching and scholarship, as paradigmatic examples of privileged copying. There is no doubt that a finding that copying was not commercial in nature tips the scales in favor of fair use. But the inverse is not necessarily true, as many common fair uses are indisputably commercial. For instance, the text of §107 includes examples like "news reporting," which is often done for commercial profit. So even though Google's use was a commercial endeavor—a fact no party disputed, see 886 F. 3d, at 1197—that is not dispositive of the first factor, particularly in light of the inherently transformative role that the reimplementation played in the new Android system.

As for bad faith, our decision in *Campbell* expressed some

skepticism about whether bad faith has any role in a fair use analysis. 510 U. S., at 585, n. 18. We find this skepticism justifiable, as "[c]opyright is not a privilege reserved for the well-behaved." Leval 1126. We have no occasion here to say whether good faith is as a general matter a helpful inquiry. We simply note that given the strength of the other factors pointing toward fair use and the jury finding in Google's favor on hotly contested evidence, that fact-bound consideration is not determinative in this context.

C. "The Amount and Substantiality of the Portion Used"

If one considers the declaring code in isolation, the quantitative amount of what Google copied was large. Google copied the declaring code for 37 packages of the Sun Java API, totaling approximately 11,500 lines of code. Those lines of code amount to virtually all the declaring code needed to call up hundreds of different tasks. On the other hand, if one considers the entire set of software material in the Sun Java API, the quantitative amount copied was small. The total set of Sun Java API computer code, including implementing code, amounted to 2.86 million lines, of which the copied 11,500 lines were only 0.4 percent. App. 212.

The question here is whether those 11,500 lines of code should be viewed in isolation or as one part of the considerably greater whole. We have said that even a small amount of copying may fall outside of the scope of fair use where the excerpt copied consists of the "'heart'" of the original work's creative expression. *Harper & Row*, 471 U. S., at 564–565. On the other hand, copying a larger amount of material can fall within the scope of fair use where the material copied captures little of the material's creative expression or is central to a copier's valid purpose. See, *e.g.*, *Campbell*, 510 U. S., at 588; *New Era Publications Int'l, ApS* v. *Carol Publishing Group*, 904 F. 2d 152, 158 (CA2 1990). If a defendant had copied one sentence in a novel, that copying may

well be insubstantial. But if that single sentence set forth one of the world's shortest short stories—"When he awoke, the dinosaur was still there."—the question looks much different, as the copied material constitutes a small part of the novel but the entire short story. See A. Monterroso, El Dinosaurio, in Complete Works & Other Stories 42 (E. Grossman transl. 1995). (In the original Spanish, the story reads: "Cuando despertó, el dinosaurio todavía estaba allí.")

Several features of Google's copying suggest that the better way to look at the numbers is to take into account the several million lines that Google did not copy. For one thing, the Sun Java API is inseparably bound to those task-implementing lines. Its purpose is to call them up. For another, Google copied those lines not because of their creativity, their beauty, or even (in a sense) because of their purpose. It copied them because programmers had already learned to work with the Sun Java API's system, and it would have been difficult, perhaps prohibitively so, to attract programmers to build its Android smartphone system without them. Further, Google's basic purpose was to create a different task-related system for a different computing environment (smartphones) and to create a platform—the Android platform—that would help achieve and popularize that objective. The "substantiality" factor will generally weigh in favor of fair use where, as here, the amount of copying was tethered to a valid, and transformative, purpose. *Supra*, at 25–26; see *Campbell*, 510 U. S., at 586–587 (explaining that the factor three "enquiry will harken back to the first of the statutory factors, for . . . the extent of permissible copying varies with the purpose and character of the use").

We do not agree with the Federal Circuit's conclusion that Google could have achieved its Java-compatibility objective by copying only the 170 lines of code that are "necessary to write in the Java language." 886 F. 3d, at 1206. In

our view, that conclusion views Google's legitimate objectives too narrowly. Google's basic objective was not simply to make the Java programming language usable on its Android systems. It was to permit programmers to make use of their knowledge and experience using the Sun Java API when they wrote new programs for smartphones with the Android platform. In principle, Google might have created its own, different system of declaring code. But the jury could have found that its doing so would not have achieved that basic objective. In a sense, the declaring code was the key that it needed to unlock the programmers' creative energies. And it needed those energies to create and to improve its own innovative Android systems.

We consequently believe that this "substantiality" factor weighs in favor of fair use.

D. Market Effects

The fourth statutory factor focuses upon the "effect" of the copying in the "market for or value of the copyrighted work." 17 U. S. C. §107(4). Consideration of this factor, at least where computer programs are at issue, can prove more complex than at first it may seem. It can require a court to consider the amount of money that the copyright owner might lose. As we pointed out in *Campbell*, "verbatim copying of the original in its entirety for commercial purposes" may well produce a market substitute for an author's work. 510 U. S., at 591. Making a film of an author's book may similarly mean potential or presumed losses to the copyright owner. Those losses normally conflict with copyright's basic objective: providing authors with exclusive rights that will spur creative expression.

But a potential loss of revenue is not the whole story. We here must consider not just the amount but also the source of the loss. As we pointed out in *Campbell*, a "lethal parody, like a scathing theatre review," may "kil[l] demand for the original." *Id.*, at 591–592. Yet this kind of harm, even if

directly translated into foregone dollars, is not "cognizable under the Copyright Act." *Id.,* at 592.

Further, we must take into account the public benefits the copying will likely produce. Are those benefits, for example, related to copyright's concern for the creative production of new expression? Are they comparatively important, or unimportant, when compared with dollar amounts likely lost (taking into account as well the nature of the source of the loss)? Cf. *MCA, INC.* v. *Wilson*, 677 F. 2d 180, 183 (CA2 1981) (calling for a balancing of public benefits and losses to copyright owner under this factor).

We do not say that these questions are always relevant to the application of fair use, not even in the world of computer programs. Nor do we say that these questions are the only questions a court might ask. But we do find them relevant here in helping to determine the likely market effects of Google's reimplementation.

As to the likely amount of loss, the jury could have found that Android did not harm the actual or potential markets for Java SE. And it could have found that Sun itself (now Oracle) would not have been able to enter those markets successfully whether Google did, or did not, copy a part of its API. First, evidence at trial demonstrated that, regardless of Android's smartphone technology, Sun was poorly positioned to succeed in the mobile phone market. The jury heard ample evidence that Java SE's primary market was laptops and desktops. App. 99, 200. It also heard that Sun's many efforts to move into the mobile phone market had proved unsuccessful. *Id.,* at 135, 235, 671. As far back as 2006, prior to Android's release, Sun's executives projected declining revenue for mobile phones because of emerging smartphone technology. *Id.,* at 240. When Sun's former CEO was asked directly whether Sun's failure to build a smartphone was attributable to Google's development of Android, he answered that it was not. *Id.,* at 650. Given

the evidence showing that Sun was beset by business challenges in developing a mobile phone product, the jury was entitled to agree with that assessment.

Second, the jury was repeatedly told that devices using Google's Android platform were different in kind from those that licensed Sun's technology. For instance, witnesses explained that the broader industry distinguished between smartphones and simpler "feature phones." *Id.,* at 237. As to the specific devices that used Sun-created software, the jury heard that one of these phones lacked a touchscreen, *id.*, at 359–360, while another did not have a QWERTY keyboard, *id.*, at 672. For other mobile devices, the evidence showed that simpler products, like the Kindle, used Java software, *id.*, at 396, while more advanced technology, like the Kindle Fire, were built on the Android operating system, *id.*, at 206. This record evidence demonstrates that, rather than just "repurposing [Sun's] code from larger computers to smaller computers," *post*, at 16, Google's Android platform was part of a distinct (and more advanced) market than Java software.

Looking to these important differences, Google's economic expert told the jury that Android was not a market substitute for Java's software. As he explained, "the two products are on very different devices," and the Android platform, which offers "an entire mobile operating stack," is a "very different typ[e] of produc[t]" than Java SE, which is "just an applications programming framework." App. 256; see also *id.*, at 172–174. Taken together, the evidence showed that Sun's mobile phone business was declining, while the market increasingly demanded a new form of smartphone technology that Sun was never able to offer.

Finally, the jury also heard evidence that Sun foresaw a benefit from the broader use of the Java programming language in a new platform like Android, as it would further expand the network of Java-trained programmers. *Id.*, at 131–133; see also *id.*, at 153 ("Once an API starts getting

reimplemented, you know it has succeeded"). In other words, the jury could have understood Android and Java SE as operating in two distinct markets. And because there are two markets at issue, programmers learning the Java language to work in one market (smartphones) are then able to bring those talents to the other market (laptops). See 4 Nimmer on Copyright §13.05[A][4] (explaining that factor four asks what the impact of "widespread conduct of the sort engaged in by the defendant" would be on the market for the present work).

Sun presented evidence to the contrary. Indeed, the Federal Circuit held that the "market effects" factor militated against fair use in part because Sun had tried to enter the Android market. 886 F. 3d, at 1209 (Sun sought licensing agreement with Google). But those licensing negotiations concerned much more than 37 packages of declaring code, covering topics like "the implementation of [Java's] code" and "branding and cooperation" between the firms. App. 245; see also 4 Nimmer on Copyright §13.05[A][4] (cautioning against the "danger of circularity posed" by considering unrealized licensing opportunities because "it is a given in every fair use case that plaintiff suffers a loss of a *potential* market if that potential is defined as the theoretical market for licensing the very use at bar"). In any event, the jury's fair use determination means that neither Sun's effort to obtain a license nor Oracle's conflicting evidence can overcome evidence indicating that, at a minimum, it would have been difficult for Sun to enter the smartphone market, even had Google not used portions of the Sun Java API.

On the other hand, Google's copying helped Google make a vast amount of money from its Android platform. And enforcement of the Sun Java API copyright might give Oracle a significant share of these funds. It is important, however, to consider why and how Oracle might have become entitled to this money. When a new interface, like an API or a spreadsheet program, first comes on the market, it may

attract new users because of its expressive qualities, such as a better visual screen or because of its superior functionality. As time passes, however, it may be valuable for a different reason, namely, because users, including programmers, are just used to it. They have already learned how to work with it. See *Lotus Development Corp.*, 49 F. 3d, at 821 (Boudin, J., concurring).

The record here is filled with evidence that this factor accounts for Google's desire to use the Sun Java API. See, *e.g.,* App. 169–170, 213–214. This source of Android's profitability has much to do with third parties' (say, programmers') investment in Sun Java programs. It has correspondingly less to do with Sun's investment in creating the Sun Java API. We have no reason to believe that the Copyright Act seeks to protect third parties' investment in learning how to operate a created work. Cf. *Campbell*, 510 U. S., at 591–592 (discussing the need to identify those harms that are "cognizable under the Copyright Act").

Finally, given programmers' investment in learning the Sun Java API, to allow enforcement of Oracle's copyright here would risk harm to the public. Given the costs and difficulties of producing alternative APIs with similar appeal to programmers, allowing enforcement here would make of the Sun Java API's declaring code a lock limiting the future creativity of new programs. Oracle alone would hold the key. The result could well prove highly profitable to Oracle (or other firms holding a copyright in computer interfaces). But those profits could well flow from creative improvements, new applications, and new uses developed by users who have learned to work with that interface. To that extent, the lock would interfere with, not further, copyright's basic creativity objectives. See *Connectix Corp.*, 203 F. 3d, at 607; see also *Sega Enterprises*, 977 F. 2d, at 1523–1524 ("An attempt to monopolize the market by making it impossible for others to compete runs counter to the statutory purpose of promoting creative expression");

Lexmark Int'l, 387 F. 3d, at 544 (noting that where a subsequent user copied a computer program to foster functionality, it was not exploiting the programs "commercial value *as a copyrighted work*" (emphasis in original)). After all, "copyright supplies the economic incentive to [both] create and disseminate ideas," *Harper & Row*, 471 U. S., at 558, and the reimplementation of a user interface allows creative new computer code to more easily enter the market.

The uncertain nature of Sun's ability to compete in Android's market place, the sources of its lost revenue, and the risk of creativity-related harms to the public, when taken together, convince that this fourth factor—market effects—also weighs in favor of fair use.

* * *

The fact that computer programs are primarily functional makes it difficult to apply traditional copyright concepts in that technological world. See *Lotus Development Corp.*, 49 F. 3d, at 820 (Boudin, J., concurring). In doing so here, we have not changed the nature of those concepts. We do not overturn or modify our earlier cases involving fair use—cases, for example, that involve "knockoff" products, journalistic writings, and parodies. Rather, we here recognize that application of a copyright doctrine such as fair use has long proved a cooperative effort of Legislatures and courts, and that Congress, in our view, intended that it so continue. As such, we have looked to the principles set forth in the fair use statute, §107, and set forth in our earlier cases, and applied them to this different kind of copyrighted work.

We reach the conclusion that in this case, where Google reimplemented a user interface, taking only what was needed to allow users to put their accrued talents to work in a new and transformative program, Google's copying of the Sun Java API was a fair use of that material as a matter of law. The Federal Circuit's contrary judgment is reversed,

and the case is remanded for further proceedings in conformity with this opinion.

It is so ordered.

JUSTICE BARRETT took no part in the consideration or decision of this case.

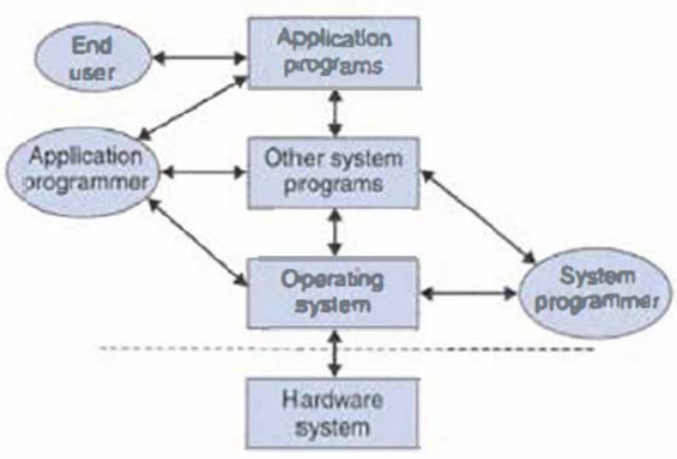
APPENDIXES

A

Computer System Diagram

Some readers might find it helpful to start with an explanation of what a "software platform" is. Put simply, a software platform collects all of the software tools that a programmer may need to build computer programs. The Android platform, for instance, includes an "operating system," "core libraries," and a "virtual machine," among other tools. App. 197–198.

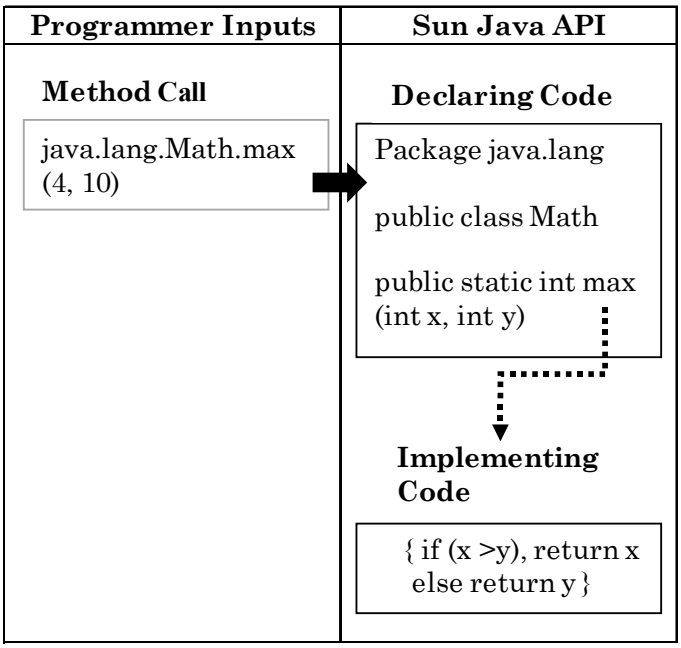
The diagram below illustrates the general features of a standard computer system, with the dotted line reflecting the division between a computer's hardware and its software. (It is not intended to reflect any specific technology at issue in this case.)



J. Garrido & R. Schlesinger, Principles of Modern Operating Systems 8 (2008) ("Figure 1.4. An External View of a Computer System").

B

Sun Java API Diagram



This image depicts the connection between the three parts of the Sun Java API technology at issue, using the District Court's example. *Oracle*, 872 F. Supp. 2d, at 980–981. The programmer enters a method call to invoke a task from within the API (the solid arrow). The precise symbols in the method call correspond to a single task, which is located within a particular class. That class is located within a particular package. All of the lines of code that provide that organization and name the methods, classes, and packages are "declaring code." For each method, the declaring code is associated with particular lines of implementing code (the dotted arrow). It is that implementing code

(which Google wrote for its Android API) that actually instructs the computer in the programmer's application.

SUPREME COURT OF THE UNITED STATES

No. 18–956

GOOGLE LLC, PETITIONER *v.*
ORACLE AMERICA, INC.

ON WRIT OF CERTIORARI TO THE UNITED STATES COURT OF APPEALS FOR THE FEDERAL CIRCUIT

[April 5, 2021]

JUSTICE THOMAS, with whom JUSTICE ALITO joins, dissenting.

Oracle spent years developing a programming library that successfully attracted software developers, thus enhancing the value of Oracle's products.[1] Google sought a license to use the library in Android, the operating system it was developing for mobile phones. But when the companies could not agree on terms, Google simply copied verbatim 11,500 lines of code from the library. As a result, it erased 97.5% of the value of Oracle's partnership with Amazon, made tens of billions of dollars, and established its position as the owner of the largest mobile operating system in the world. Despite this, the majority holds that this copying was fair use.

The Court reaches this unlikely result in large part because it bypasses the antecedent question clearly before us: Is the software code at issue here protected by the Copyright Act? The majority purports to assume, without deciding, that the code is protected. But its fair-use analysis is wholly inconsistent with the substantial protection Congress gave to computer code. By skipping over the copy-

[1] A different company, Sun, created the library. But because Oracle later purchased Sun, for simplicity I refer to both companies as Oracle.

rightability question, the majority disregards half the relevant statutory text and distorts its fair-use analysis. Properly considering that statutory text, Oracle's code at issue here is copyrightable, and Google's use of that copyrighted code was anything but fair.

I

In the 1990s, Oracle created a programming language called Java. Like many programming languages, Java allows developers to prewrite small subprograms called "methods." Methods form the building blocks of more complex programs. This process is not unlike what legislatures do with statutes. To save space and time, legislatures define terms and then use those definitions as a shorthand. For example, the legal definition for "refugee" is more than 300 words long. 8 U. S. C. §1101(42). Rather than repeat all those words every time they are relevant, the U. S. Code encapsulates them all with a single term that it then inserts into each relevant section. Java methods work similarly. Once a method has been defined, a developer need only type a few characters (the method name and relevant inputs) to invoke everything contained in the subprogram. A programmer familiar with prewritten methods can string many of them together to quickly develop complicated programs without having to write from scratch all the basic subprograms.

To create Java methods, developers use two kinds of code. The first, "declaring code," names the method, defines what information it can process, and defines what kind of data it can output. It is like the defined term in a statute. The second, "implementing code," includes the step-by-step instructions that make those methods run.[2] It is like the detailed definition in a statute.

---

[2] Consider what the relevant text of a simple method—designed to return the largest of three integers—might look like:

Oracle's declaring code was central to its business model. Oracle profited financially by encouraging developers to create programs written in Java and then charging manufacturers a fee to embed in their devices the Java software platform needed to run those programs. To this end, Oracle created a work called Java 2 Platform, Standard Edition, which included a highly organized library containing about 30,000 methods. Oracle gave developers free access to these methods to encourage them to write programs for the Java platform. In return, developers were required to make their programs compatible with the Java platform on any device. Developers were encouraged to make improvements to the platform, but they were required to release beneficial modifications to the public. If a company wanted to customize the platform and keep those customizations secret for business purposes, it had to pay for a separate license.

By 2005, many companies were racing to develop operating systems for what would become modern smartphones. Oracle's strategy had successfully encouraged millions of programmers to learn Java. As a result, Java software platforms were in the vast majority of mobile phones. Google wanted to attract those programmers to Android by including in Android the declaring code with which they were now familiar. But the founder of Android, Andrew Rubin, understood that the declaring code was copyrighted, so Google sought a custom license from Oracle. At least four times between 2005 and 2006, the two companies attempted to

---

```
public static int MaxNum (int x, int y, int z) {
  if (x >= y && x >= z) return x;
  else if (y >= x && y >= z) return y;
  else return z;
  }
```

The first line is declaring code that defines the method, including what inputs (integers x, y, and z) it can process and what it can output (an integer). The remainder is implementing code that checks which of the inputs is largest and returns the result. Once this code is written, a programmer could invoke it by typing, for example, "MaxNum (4, 12, 9)."

negotiate a license, but they were unsuccessful, in part because of "trust issues." App. 657.

When those negotiations broke down, Google simply decided to use Oracle's code anyway. Instead of creating its own declaring code—as Apple and Microsoft chose to do—Google copied verbatim 11,500 lines of Oracle's declaring code and arranged that code exactly as Oracle had done. It then advertised Android to device manufacturers as containing "Core Java Libraries." *Id.*, at 600. Oracle predictably responded by suing Google for copyright infringement. The Federal Circuit ruled that Oracle's declaring code is copyrightable and that Google's copying of it was not fair use.

II

The Court wrongly sidesteps the principal question that we were asked to answer: Is declaring code protected by copyright? I would hold that it is.

Computer code occupies a unique space in intellectual property. Copyright law generally protects works of authorship. Patent law generally protects inventions or discoveries. A library of code straddles these two categories. It is highly functional like an invention; yet as a writing, it is also a work of authorship. Faced with something that could fit in either space, Congress chose copyright, and it included declaring code in that protection.

The Copyright Act expressly protects computer code. It recognizes that a "computer program" is protected by copyright. See 17 U. S. C. §§109(b), 117, 506(a). And it defines "'computer program'" as "a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result." §101. That definition clearly covers declaring code—sets of statements that indirectly perform computer functions by triggering prewritten implementing code.

Even without that express language, declaring code

would satisfy the general test for copyrightability. "Copyright protection subsists . . . in original works of authorship fixed in any tangible medium of expression." §102(a). "Works of authorship include . . . literary works," which are "works . . . expressed in words, numbers, or other verbal or numerical symbols." §§101, 102(a). And a work is "original" if it is "independently created by the author" and "possesses at least some minimal degree of creativity." *Feist Publications, Inc.* v. *Rural Telephone Service Co.*, 499 U. S. 340, 345 (1991). The lines of declaring code in the Java platform readily satisfy this "extremely low" threshold. *Ibid.* First, they are expressed in "words, numbers, or other verbal or numerical symbols" and are thus works of authorship. §101. Second, as Google concedes, the lines of declaring code are original because Oracle could have created them any number of ways.

Google contends that declaring code is a "method of operation" and thus excluded from protection by §102(b). That subsection excludes from copyright protection "any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied." This provision codifies the "idea/expression dichotomy" that copyright protection covers only the "the author's expression" of an idea, not the idea itself. *Golan* v. *Holder*, 565 U. S. 302, 328 (2012). A property right in the idea itself "can only be secured, if it can be secured at all, by letters-patent." *Baker* v. *Selden*, 101 U. S. 99, 105 (1880). Thus, for example, a "method of book-keeping" is not protected by copyright, but the expression describing that accounting method is. *Id.*, at 101–102. So too, a person who writes a book inventing the idea of declaring code has a copyright protection in the expression in the book, but not in the idea of declaring code itself. Google acknowledges that implementing code is protected by the Copyright Act, but it contends that declaring

code is much more functional and thus is a "method of operation" outside the scope of protection.

That argument fails. As the majority correctly recognizes, declaring code and implementing code are "inextricably bound" together. *Ante*, at 22. Declaring code defines the scope of a set of implementing code and gives a programmer a way to use it by shortcut. Because declaring code incorporates implementing code, it has no function on its own. Implementing code is similar. Absent declaring code, developers would have to write every program from scratch, making complex programs prohibitively time consuming to create. The functionality of both declaring code and implementing code will thus typically rise and fall together.

Google's argument also cannot account for Congress' decision to define protected computer code as "a set of statements or instructions to be used *directly or indirectly* in a computer in order to bring about a certain result." §101 (emphasis added). Hence, Congress rejected any categorical distinction between declaring and implementing code. Implementing code orders a computer operation directly. Declaring code does so indirectly by incorporating implementing code. When faced with general language barring protection for "methods of operation" and specific language protecting declaring code, the "'specific governs the general.'" *RadLAX Gateway Hotel, LLC* v. *Amalgamated Bank*, 566 U. S. 639, 645 (2012).

This context makes clear that the phrase "method of operation" in §102(b) does not remove protection from declaring code simply because it is functional. That interpretation does not, however, render "method of operation" meaningless. It is "given more precise content by the neighboring words with which it is associated." *United States* v. *Williams*, 553 U. S. 285, 294 (2008). Other terms in the same subsection such as "idea," "principle," and "concept" suggest that "method of operation" covers the functions and

ideas implemented by computer code—such as math functions, accounting methods, or the idea of declaring code—not the specific expression Oracle created. Oracle cannot copyright the idea of using declaring code, but it can copyright the specific expression of that idea found in its library.

Google also contends that declaring code is not copyrightable because the "merger doctrine" bars copyright protection when there is only one way to express an idea. That argument fails for the same reasons Google's §102(b) argument fails. Even if the doctrine exists, Google admits that it is merely an application of §102(b). And, in any event, there may have been only one way for Google to copy the lines of declaring code, but there were innumerable ways for Oracle to write them. Certainly, Apple and Microsoft managed to create their own declaring code.

III

The Court inexplicably declines to address copyrightability. Its sole stated reason is that "technological, economic, and business-related circumstances" are "rapidly changing." *Ante*, at 15. That, of course, has been a constant where computers are concerned.

Rather than address this principal question, the Court simply assumes that declaring code is protected and then concludes that every fair-use factor favors Google. I agree with the majority that Congress did not "shiel[d] computer programs from the ordinary application" of fair use. *Ante,* at 18. But the majority's application of fair use is far from ordinary. By skipping copyrightability, the majority gets the methodology backward, causing the Court to sidestep a key conclusion that ineluctably affects the fair-use analysis: Congress rejected categorical distinctions between declaring and implementing code. But the majority creates just such a distinction. The result of this distorting analysis is an opinion that makes it difficult to imagine any circum-

stance in which declaring code will remain protected by copyright.

I agree with the majority that, under our precedent, fair use is a mixed question of fact and law and that questions of law predominate.[3] Because the jury issued a finding of fair use in favor of Google, we must construe all factual disputes and inferences in Google's favor and ask whether the evidence was sufficient as a matter of law to support the jury's verdict. See Fed. Rule Civ. Proc. 50(b). But whether a statutory fair-use factor favors one side or the other is a legal question reviewed *de novo*. Congress has established four statutory fair-use factors for courts to weigh.[4] Three decisively favor Oracle. And even assuming that the remaining factor favors Google, that factor, without more, cannot legally establish fair use in this context.

The majority holds otherwise—concluding that *every* factor favors Google—by relying, in large part, on a distinction it draws between declaring and implementing code, a distinction that the statute rejects. Tellingly, the majority

---

[3] I would not, however, definitively resolve Google's argument that the Seventh Amendment commits the question of fair use to a jury. I tend to agree with the Court that fair use was not "itself necessarily a jury issue" when the Constitution was ratified. *Markman* v. *Westview Instruments*, *Inc.*, 517 U. S. 370, 376–377 (1996). Google cites cases about "fair abridgment," but Congress has since made clear that copyright holders have "exclusive rights" over any "abridgment." 17 U. S. C. §§101, 106. And in any event, judges often declined to refer these issues to juries. See, *e.g.*, *Gyles* v. *Wilcox*, 2 Atk. 141, 144, 26 Eng. Rep. 489, 490–491 (Ch. 1740); *Folsom* v. *Marsh*, 9 F. Cas. 342, 345–349 (No. 4,901) (CC Mass. 1841) (Story, J). Still, we should not so casually decide this question when the parties barely addressed it.

[4] The factors are: "(1) the purpose and character of the use, including whether such use is of a commercial nature or is for nonprofit educational purposes; (2) the nature of the copyrighted work; (3) the amount and substantiality of the portion used in relation to the copyrighted work as a whole; and (4) the effect of the use upon the potential market for or value of the copyrighted work." §§107(1)–(4).

evaluates the factors neither in sequential order nor in order of importance (at least two factors are more important under our precedent[5]). Instead, it starts with the second factor: the nature of the copyrighted work. It proceeds in this manner in order to create a distinction between declaring and implementing code that renders the former less worthy of protection than the latter. Because the majority's mistaken analysis rests so heavily on this factor, I begin with it as well.

A. The Nature of the Copyrighted Work

This factor requires courts to assess the level of creativity or functionality in the original work. It generally favors fair use when a copyrighted work is more "informational or functional" than "creative." 4 M. Nimmer & D. Nimmer, Copyright §13.05[A][2][a] (2019). Because code is predominantly functional, this factor will often favor copying when the original work is computer code. But because Congress determined that declaring and implementing code are copyrightable, this factor alone cannot support a finding of fair use.

The majority, however, uses this factor to create a distinction between declaring and implementing code that in effect removes copyright protection from declaring code. It concludes that, unlike implementing code, declaring code is far "from the core of copyright" because it becomes valuable only when third parties (computer programmers) value it and because it is "inherently bound together with uncopyrightable ideas." *Ante*, at 23–24.

---

[5] The fourth factor—the effect of Google's copying on the potential market for Oracle's work—is "undoubtedly the single most important element of fair use." *Harper & Row, Publishers, Inc.* v. *Nation Enterprises*, 471 U. S. 539, 566 (1985). The first factor—the purpose and character of the use, including whether the use is commercial—is the second-most important because it can prove dispositive. See *id.*, at 550 ("[In general,] the fair use doctrine has always precluded a use that 'supersede[s] the use of the original'").

Congress, however, rejected this sort of categorical distinction that would make declaring code less worthy of protection. The Copyright Act protects code that operates "in a computer in order to bring about a certain result" both "directly" (implementing code) and "indirectly" (declaring code). §101. And if anything, declaring code is *closer* to the "core of copyright." *Ante*, at 24. Developers cannot even see implementing code. *Oracle Am., Inc.* v. *Google Inc.*, 2016 WL 3181206, *4 (ND Cal., June 8, 2016); see also *ante*, at 23 (declaring code is "user-centered"). Implementing code thus conveys *no* expression to developers. Declaring code, in contrast, is user facing. It must be designed and organized in a way that is intuitive and understandable to developers so that they can invoke it.

Even setting those concerns aside, the majority's distinction is untenable. True, declaring code is "inherently bound together with uncopyrightable ideas." *Ante,* at 23–24. Is anything not? Books are inherently bound with uncopyrightable ideas—the use of chapters, having a plot, or including dialogue or footnotes. This does not place books far "from the core of copyright." And implementing code, which the majority concedes is copyrightable, is inherently bound up with "the division of computing tasks" that cannot be copyrighted.[6] *Ante*, at 22. We have not discounted a work of authorship simply because it is associated with noncopyrightable ideas. While ideas cannot be copyrighted, expressions of those ideas can. *Golan*, 565 U. S., at 328.

Similarly, it makes no difference that the value of declaring code depends on how much time third parties invest in

[6]The majority also belittles declaring code by suggesting it is simply a way to organize implementing code. *Ante*, at 22–23. Not so. Declaring code *defines* subprograms of implementing code, including by controlling what inputs they can process. Similarly, the majority is wrong to suggest that the purpose of declaring code is to connect pre-existing method calls to implementing code. *Ante,* at 5. Declaring code *creates* the method calls.

learning it. Many other copyrighted works depend on the same. A Broadway musical script needs actors and singers to invest time learning and rehearsing it. But a theater cannot copy a script—the rights to which are held by a smaller theater—simply because it wants to entice actors to switch theaters and because copying the script is more efficient than requiring the actors to learn a new one.

What the majority says is true of declaring code is no less true of implementing code. Declaring code is how programmers access prewritten implementing code. The value of that implementing code thus is directly proportional to how much programmers value the associated declaring code. The majority correctly recognizes that declaring code "is inextricably bound up with implementing code," *ante*, at 22–23, but it overlooks the implications of its own conclusion.

Only after wrongly concluding that the nature of declaring code makes that code generally unworthy of protection does the Court move on to consider the other factors. This opening mistake taints the Court's entire analysis.

B. Market Effects

"[U]ndoubtedly the single most important element of fair use" is the effect of Google's copying "'upon the potential market for or value of [Oracle's] copyrighted work.'" *Harper & Row, Publishers, Inc.* v. *Nation Enterprises*, 471 U. S. 539, 566 (1985). As the Federal Circuit correctly determined, "evidence of actual and potential harm stemming from Google's copying was 'overwhelming.'" 886 F. 3d 1179, 1209 (2018). By copying Oracle's code to develop and release Android, Google ruined Oracle's potential market in at least two ways.

First, Google eliminated the reason manufacturers were willing to pay to install the Java platform. Google's business model differed from Oracle's. While Oracle earned revenue by charging device manufacturers to install the Java platform, Google obtained revenue primarily through ad

sales. Its strategy was to release Android to device manufacturers for free and then use Android as a vehicle to collect data on consumers and deliver behavioral ads. With a free product available that included much of Oracle's code (and thus with similar programming potential), device manufacturers no longer saw much reason to pay to embed the Java platform.

For example, before Google released Android, Amazon paid for a license to embed the Java platform in Kindle devices. But after Google released Android, Amazon used the cost-free availability of Android to negotiate a 97.5% discount on its license fee with Oracle. Evidence at trial similarly showed that right after Google released Android, Samsung's contract with Oracle dropped from $40 million to about $1 million. Google contests none of this except to say that Amazon used a different Java platform, Java Micro Edition instead of Java Standard Edition. That difference is inconsequential because the former was simply a smaller subset of the latter. Google copied code found in both platforms. The majority does not dispute—or even mention—this enormous harm.

Second, Google interfered with opportunities for Oracle to license the Java platform to developers of smartphone operating systems. Before Google copied Oracle's code, nearly every mobile phone on the market contained the Java platform. Oracle's code was extraordinarily valuable to anybody who wanted to develop smartphones, which explains why Google tried no fewer than four times to license it. The majority's remark that Google also sought other licenses from Oracle, *ante*, at 33, does not change this central fact. Both parties agreed that Oracle could enter Google's current market by licensing its declaring code. But by copying the code and releasing Android, Google eliminated Oracle's opportunity to license its code for that use.

The majority writes off this harm by saying that the jury could have found that Oracle might not have been able to

enter the modern smartphone market successfully.[7] *Ante*, at 31–32. But whether Oracle could itself enter that market is only half the picture. We look at not only the potential market "that creators of original works would in general develop" but also those potential markets the copyright holder might "license others to develop." *Campbell* v. *Acuff-Rose Music*, *Inc.*, 510 U. S. 569, 592 (1994). A book author need not be able to personally convert a book into a film so long as he can license someone else to do so. That Oracle could have licensed its code for use in Android is undisputed.

Unable to seriously dispute that Google's actions had a disastrous effect on Oracle's potential market, the majority changes course and asserts that enforcing copyright protection could harm the public by giving Oracle the power to "limi[t] the future creativity" of programs on Android. *Ante*, at 34. But this case concerns only versions of Android released through November 2014. Order in No. 3:10–cv–3561 (ND Cal., Feb. 5, 2016), Doc. 1479, p. 2 (identifying versions through Android Lollipop 5.0). Google has released six major versions since then. Only about 7.7% of active Android devices still run the versions at issue.[8] The majority's concern about a lock-in effect might carry more weight if this suit concerned versions of Android widely in use or that will be widely in use. It makes little sense in a suit about versions that are close to obsolete.

The majority's concern about a lock-in effect also is speculation belied by history. First, Oracle never had lock-in

---

[7] It also suggests that Oracle may have received some incidental benefit from Android. *Ante*, at 32–33. But even assuming that is true, it would go to the question of damages, not fair use. And there is no evidence that any benefit came even close to offsetting Oracle's enormous loss.

[8] Rahman, Android Version Distribution Statistics Will Now Only Be Available in Android Studio (Apr. 10, 2020), https://www.xda-developers.com/android-version-distribution-statistics-android-studio.

power. The majority (again) overlooks that Apple and Microsoft created mobile operating systems without using Oracle's declaring code. Second, Oracle always made its declaring code freely available to programmers. There is little reason to suspect Oracle might harm programmers by stopping now. And third, the majority simply assumes that the jury, in a future suit over current Android versions, would give Oracle control of Android instead of just awarding damages or perpetual royalties.

If the majority is going to speculate about what Oracle *might* do, it at least should consider what Google *has* done. The majority expresses concern that Oracle might abuse its copyright protection (on outdated Android versions) and "'attempt to monopolize the market.'" *Ante*, at 34–35. But it is Google that recently was fined a record $5 billion for abusing Android to violate antitrust laws. Case AT.40099, *Google Android*, July 18, 2018 (Eur. Comm'n-Competition); European Comm'n Press Release, Commission Fines Google €4.34 Billion for Illegal Practices Regarding Android Mobile Devices to Strengthen Dominance of Google's Search Engine, July 18, 2018. Google controls the most widely used mobile operating system in the world. And if companies may now freely copy libraries of declaring code whenever it is more convenient than writing their own, others will likely hesitate to spend the resources Oracle did to create intuitive, well-organized libraries that attract programmers and could compete with Android. If the majority is worried about monopolization, it ought to consider whether Google is the greater threat.

By copying Oracle's work, Google decimated Oracle's market and created a mobile operating system now in over 2.5 billion actively used devices, earning tens of billions of dollars every year. If these effects on Oracle's potential market *favor* Google, something is very wrong with our fair-use analysis.

C. The Purpose and Character of the Use

The second-most important factor—"the purpose and character of the use, including whether such use is of a commercial nature or is for nonprofit educational purposes," §107(1)—requires us to consider whether use was "commercial" and whether it was "transformative." *Campbell*, 510 U. S., at 578–579. Both aspects heavily favor Oracle.

Begin with the overwhelming commercial nature of Google's copying. In 2015 alone, the year before the fair-use trial, Google earned $18 billion from Android. That number has no doubt dramatically increased as Android has grown to dominate the global market share.[9] On this scale, Google's use of Oracle's declaring code weighs heavily—if not decisively—against fair use.

The majority attempts to dismiss this overwhelming commercial use by noting that commercial use does "not necessarily" weigh against fair use. *Ante*, at 27. True enough. Commercial use sometimes can be overcome by use that is sufficiently "transformative." *Campbell*, 510 U. S., at 579. But "we cannot ignore [Google's] *intended purpose* of supplanting [Oracle's] commercially valuable" platform with its own. *Harper*, 471 U. S., at 562 (emphasis in original). Even if we could, we have never found fair use for copying that reaches into the tens of billions of dollars and wrecks

[9]The real value also may be much higher because Android indirectly boosts other sources of revenue. For years Google has set its search engine as the default engine on Android. Google can use that engine to collect reams of data used to deliver behavioral advertisements to consumers on desktops. Using control over Android to choose a default search engine may seem trivial, but Google certainly does not think so. According to a Goldman Sachs analysis, Google paid Apple *$12 billion* to be the default search engine for Safari, Apple's web browser, for just one year. Leswing, Apple Makes Billions From Google's Dominance in Search—And It's a Bigger Business Than iCloud or Apple Music, Business Insider, Sept. 29, 2018. Google does not appear to have disputed this figure.

the copyright holder's market.

Regardless, Google fairs no better on transformative use. A court generally cannot find fair use unless the copier's use is transformative.[10] A work is "transformative" if it "adds something new, with a further purpose or different character, altering the first with new expression, meaning, or message." *Campbell*, 510 U. S., at 579. This question is "guided by the examples [of fair use] given in the preamble to §107." *Id.*, at 578. Those examples include: "criticism, comment, news reporting, teaching . . . , scholarship, or research." §107. Although these examples are not exclusive, they are illustrative, and Google's repurposing of Java code from larger computers to smaller computers resembles none of them. Google did not use Oracle's code to teach or reverse engineer a system to ensure compatibility. Instead, to "avoid the drudgery in working up something fresh," *id.*, at 580, Google used the declaring code for the same exact purpose Oracle did. As the Federal Circuit correctly determined, "[t]here is nothing fair about taking a copyrighted work verbatim and using it for the same purpose and function as the original in a competing platform." 886 F. 3d, at 1210.

The majority acknowledges that Google used the copied declaring code "for the same reason" Oracle did. *Ante*, at 25. So, by turns, the majority transforms the definition of "transformative." Now, we are told, "transformative" simply means—at least for computer code—a use that will help others "create new products." *Ibid;* accord, *ante*, at 26 (Google's copying "can further the development of computer programs").

---

[10] Although "transformative use is not *absolutely* necessary" every time, *Campbell* v. *Acuff-Rose Music, Inc.*, 510 U. S. 569, 579, and n. 11 (1994) (emphasis added), as a general matter "the fair use doctrine has always precluded a use that 'supersedes the use of the original,'" *Harper*, 471 U. S., at 550 (brackets omitted).

That new definition eviscerates copyright. A movie studio that converts a book into a film without permission not only creates a new product (the film) but enables others to "create products"—film reviews, merchandise, YouTube highlight reels, late night television interviews, and the like. Nearly every computer program, once copied, can be used to create new products. Surely the majority would not say that an author can pirate the next version of Microsoft Word simply because he can use it to create new manuscripts.[11]

Ultimately, the majority wrongly conflates transformative use with derivative use. To be transformative, a work must do something fundamentally different from the original. A work that simply serves the same purpose in a new context—which the majority concedes is true here—is derivative, not transformative. Congress made clear that Oracle holds "the exclusive rights . . . to prepare derivative works." §106(2). Rather than create a transformative product, Google "profit[ed] from exploitation of the copyrighted material without paying the customary price." *Harper*, 471 U. S., at 562.

D. The Amount and Substantiality of the Portion Used

The statutory fair-use factors also instruct us to consider "the amount and substantiality of the portion used in relation to the copyrighted work as a whole." §107(3). In general, the greater the amount of use, the more likely the copying is unfair. *Ibid.* But even if the copier takes only a small amount, copying the "'heart'" or "focal points" of a work weighs against fair use, *Harper*, 471 U. S., at 565–566, unless "'no more was taken than necessary'" for the copier to achieve transformative use, *Campbell*, 510 U. S., at 589.

[11] Because the majority's reasoning would undermine copyright protection for so many products long understood to be protected, I understand the majority's holding as a good-for-declaring-code-only precedent.

Google does not dispute the Federal Circuit's conclusion that it copied the heart or focal points of Oracle's work. 886 F. 3d, at 1207. The declaring code is what attracted programmers to the Java platform and why Google was so interested in that code. And Google copied that code "verbatim," which weighs against fair use. *Harper*, 471 U. S., at 565. The majority does not disagree. Instead, it concludes that Google took no more than necessary to create new products. That analysis fails because Google's use is not transformative. *Campbell*, 510 U. S., at 586 (recognizing that this fourth factor "will harken back to the [purpose-and-character] statutory facto[r]"). This factor thus weighs against Google.

Even if Google's use were transformative, the majority is wrong to conclude that Google copied only a small portion of the original work. The majority points out that the 11,500 lines of declaring code—enough to fill about 600 pages in an appendix, Tr. of Oral Arg. 57—were just a fraction of the code in the Java platform. But the proper denominator is *declaring code*, not all code. A copied work is quantitatively substantial if it could "serve as a market substitute for the original" work or "potentially licensed derivatives" of that work. *Campbell*, 510 U. S., at 587. The declaring code is what attracted programmers. And it is what made Android a "market substitute" for "potentially licensed derivatives" of Oracle's Java platform. Google's copying was both qualitatively and quantitatively substantial.

* * *

In sum, three of the four statutory fair-use factors weigh decidedly against Google. The nature of the copyrighted work—the sole factor possibly favoring Google—cannot by itself support a determination of fair use because holding

otherwise would improperly override Congress' determination that declaring code is copyrightable.[12]

IV

The majority purports to save for another day the question whether declaring code is copyrightable. The only apparent reason for doing so is because the majority cannot square its fundamentally flawed fair-use analysis with a finding that declaring code is copyrightable. The majority has used fair use to eviscerate Congress' considered policy judgment. I respectfully dissent.

[12] To be sure, these factors are not necessarily exclusive, but they are "especially relevant," *Harper*, 471 U. S., at 560; the majority identifies no other relevant factors; and I can think of none that could overcome the overwhelming weight of these key factors.

ソフトウェア等の権利保護に関する調査研究報告書（資料編） － 2022（令和 4）年度 －

2023（令和 5）年 3 月発行

発行所　一般財団法人ソフトウェア情報センター

〒105-0003 東京都港区西新橋 3-16-11

Tel: 03-3437-3071

E-mail: res@softic.or.jp

© 2023 Software Information Center