論　説

プログラムにおける相互運用性の意義と著作権の保護範囲

—— Google 対 Oracle 事件米国最高裁判決を素材に——

泉　　克　幸*

はじめに

2021 年 4 月 5 日，米合衆国連邦最高裁判所は，Google 社（以下「Google」）による Java API の複製はフェア・ユースに当たるため著作権を侵害しないとの判決（以下，「本最高裁判決」）[1]を下した[2]。Oracle 社（以下「Oracle」）が，自社の Java API を Google が無断複製したとして著作権侵害訴訟を提起してから 10 年以上に及ぶ年月の後のことである。Java API がどのようなものかについて詳しくは後述するが，いずれにせよ，コンピュータ・プログラムと密接に関わっている。

米国において著作権法の一部が改正され，コンピュータ・プログラムの定義規定が新設されたのは 1980 年のことである[3),4)]。その定義規定からも分かるように，プログラムの本質は機能であり，それ自体を鑑賞して楽しむといった類のものではないため[5)]，小説や絵画，音楽といった他の著作物とは異なる独特の問題が存在する。たとえば，ア

＊ 中央大学法科大学院兼任講師，関西大学総合情報学部教授

イデアと表現の区別の困難性，相互運用性や互換性[6]の確保の必要性，リバース・エンジニアリングの是非などである。そして，上述の著作権法一部改正以降，米国ではかなりの数の判決例が，これらの問題と関連しつつプログラムの保護範囲[7]を巡って出されている[8]。本最高裁判決はこれら判決例の流れの中で位置づけることができるものであり，プログラム著作物の保護範囲を明らかにしたものと評価することができる。ところで，最高裁にはGoogle，Oracleのそれぞれを支持する第三者からの意見書，アミカス・ブリーフ（amicus brief）が数多く提出されたが[9),10)]その1つに，72人の知的財産法学者からのもの[11)]がある。この意見書はGoogleを支持するものであるが，本件の第一次控訴審であった連邦巡回区控訴裁判所（Court of Appeals for the Federal Circuit。以下「CAFC」）によるJava APIの著作物性を肯定する判決[12)]は破棄されるべきであるとの見解を示している。

わが国のコンピュータ・プログラムの法的保護のあり方を巡り，旧通産省による独自立法と文化庁による著作権法改正の双方が提言されたものの，最終的には著作権法改正案だけが残り，改正法（法律62号）として成立したのは米国著作権法改正から遅れること5年の1985年（昭和60年）6月であった。同改正法は翌年の1986年1月1日から施行されている[13)]。この改正により，「プログラム」の定義（著2条1項10号の2）が置かれ，著作物の例示規定に「プログラムの著作物」（同10条1項9号）が追加され現在に至っている。わが国でもプログラム著作物の保護範囲に関する事例はあるものの，米国と比較して決して多くはない[14)]。また，インターフェイスや相互運用性の議論[15)]を含む本最高裁判決と同様の事例は見当たらない。他方で，Javaプログラミング言語はわが国でも広く普及しており，同言語を用いたプログラムが日々作り出され，また，様々な環境で使用されている。

以上のことから，本最高裁判決を取り上げ，相互運用性の観点から検討することは，米国におけるプログラム著作物の保護範囲の理解にとってはもちろんであるが，わが国のそれにとっても非常に有益だと思われる。

Ⅰ　本最高裁判決を検討するための前提および予備知識

本最高裁判決を理解・検討するためには，①相互運用性を実現するのに必要となるソフトウェア・インターフェイスに対して，米国の判決例は従来どのような議論・対処を行ってきたのか，②本最高裁判決に至る訴訟経緯，および，③Java APIとはいかなる

ものか，ということについての最低限の知識が欠かせない。そこで，本章ではまずこれらについて明らかにすることにする。

1．プログラム著作物の保護範囲に関する重要な先例
――特に相互運用性との関係で

(1) アイデアと表現の二分論を明らかにした Baker 対 Selden 事件

著作権法の保護は抽象的なアイデアには及ばず，それを具体化した表現にのみ及ぶ。換言すれば，保護対象である著作物は具体的表現であるということになる（「アイデアと表現の二分論／ idea-expression dichotomy」）。これは世界中全ての著作権法において共通する基本原則である。米国著作権法 102 条(a)(1)は，著作権の対象となる著作物の種類の 1 つとして「言語著作物（literary works）」を挙げるが，一般的にコンピュータ・プログラムはこの言語著作物に当たると理解されている[16]。他方で，102 条(b)は，「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」と規定し，著作権の保護がアイデアには認められないというアイデアと表現の二分論を明示する。

このアイデアと表現の二分論を確立させた判例が Baker 対 Selden 事件最高裁判決[17]である。本件において原告 Selden が著作物性を主張したのは，簿記システムを説明する文書と添付されていた特定の記入用紙や白紙から構成されている書籍であった。記入用紙は罫線と見出しで構成されており，簿記システムが図解され，実際の使用・実行方法が示されていた。このシステムは，複式簿記と同じ結果をもたらすが，列と見出しの特殊な配置により，1 日，1 週間，または 1 か月分の営業全体を，1 ページまたは見開き 2 ページに表示させることができるというものであった。被告である Baker は同じような結果が得られる図面を用いた書籍を販売したが，列の配列を変えており見出しも異なるものを使用していた。Selden の簿記システムに関して最高裁は以下のように判示し，その著作物性を認めなかった。「科学や有用な技芸に関する本を出版する目的は，そこに含まれる有用な知識を世界に伝えることである。しかしながら，有用な知識の利用が本の著作権侵害を引き起こすとすれば，この目的は達成されない。そして，その本が教授する技芸が，本を説明するために使われている方法や図，あるいはそれに類似したものを用いることなく利用できない場合，そのような方法や図は，当該技芸に不可欠な付随物であるとみなされ，当該技芸と共に公衆に帰属する」[18]。「当裁判所は，白紙の

帳簿は著作権の対象ではなく，Selden の書籍の著作権だけでは，Selden が設計し当該書籍に記載・図示されているように罫線が引かれ配列されている帳簿を作成して使用する独占的権利が与えられるわけではないという結論に達した」[19]。

以上のように，Beker 事件最判は，書籍が説明しようとしている技芸（＝アイデア）と，当該技芸をその人なりのやり方で説明した記述（＝表現）とを明確に区別し，著作権法で保護されるのは後者のみであること，そして前者の独占は許されず誰でもが自由に利用できるというアイデアと表現の二分論を明らかにしている。この最高裁の考え方は，後に，マージ理論もしくは融合理論（merger doctrine）に発展した。マージ理論とは，あるアイデアを説明しようとする際，どうしても特定の表現を用いなければならないケースでは，アイデアと表現は不可分一体であり分離できないのであって，表現を著作権法で保護することはアイデアの独占につながるため，その保護を否定するという考え方である。マージ理論は Baker 事件最判以降，合衆国の各巡回区における判決の中で現れているが，(i) アイデアとマージした表現については著作物であること自体を否定するとの考え方と，(ii) そのような表現の著作物性は肯定するものの，実際の行使の場面で侵害の範囲を限定するという考え方の二者に，大きく分けることができる[20]。

(2) プログラムの SSO の保護と Whelan 対 Jaslow 事件

Whelan 事件では，Whelan 社が開発したプログラム（「Dentalab」）について著作権侵害が争われた。Dentalab はデンタル・ラボラトリー（歯科医院向けの機材などを販売する会社）の業務に利用されるものであるが，EDL という特殊なプログラム言語で書かれており，IBM 社の「シリーズ 1（Series One）」コンピュータでしか使用できなかった。Jaslow 社はより広く普及している IBM の「PC」コンピュータで使用できるように Dentalab を BASIC 言語で書き換えたプログラム（「Dentcom」）を作成し，販売を開始した。そこで，Whelan 社が Jaslow 社を提訴したのが本件である。ペンシルベニア東部地区連邦地方裁判所は Whelan 社の主張を認め，Jaslow 社の Dentcom は Dentalab の著作権を侵害していると判断した[21]。Jaslow 社は控訴したが，第 3 巡回区連邦控訴裁判所は原審を支持する判決を下した[22]。

この控訴審判決について最も着目すべき点は，「コンピュータ・プログラムの著作権保護は，プログラムの文字通りのコードを超えて，その構造（structure），順序（sequence）および組織（organization）に及ぶ可能性がある」[23]と判示したことにある。この，「構造，順序および組織」は「SSO」と称されることが多い[24]。控訴裁判所はプログラムの著作権法上の保護範囲に関する議論において，アイデアと表現との境界線を引くために

Baker 事件最判を取り上げ，空欄の用紙が Selden の会計手法にとって不可欠に付随する場合には，その用紙には著作権法上の保護は認められないとする同最高裁の判断から，「Baker 事件最判では Selden の書籍が達成しようとしている目的に焦点が当られているのと同じように，アイデアと表現の境界線は，争いとなっている作品が達成しようとしている目的を考慮することで引くことができる。換言すれば，実用品の目的または機能はアイデアであり，他方で，目的や機能にとって必要でないものは全て，アイデアの表現の一部である」[25]との理解を示した。そして，「目的を達成するために様々な手段が存在している場合には，選択された特定の手段はその目的のために必ずしも必要なものではない。それゆえアイデアではなく表現である」[26]との一般的なルールを明らかにした。控訴裁判所はこの一般論を Dentalab プログラムに当てはめて検討し，実用的性格を有する Dentalab プログラムの目的がデンタル・ラボラトリーの事業運営にあることを指摘し，さらに，「プログラムの構造がそのタスクに不可欠なものでないことも明らかである。なぜなら，同じ機能を実行するが構造とデザインが異なる，Dentalab および Dentcom と競合する他のプログラムが市場に存在するからである」[27]と述べた。そして，「ソフトウェア・コンピュータ・プログラムにおける『アイデアの表現』とは，有益な情報を受け取り，整理し，計算し，保持し，相関させ，スクリーン上に表示，プリントアウト，または音声通信によって生成させるときに，プログラムがコンピュータを作動，管理，および制御する際の方法のことである」との地裁判決の判断を取り上げ，これを肯定した上で，「Dentalab プログラムの詳細な構造は，このプログラムのアイデアではなく表現の一部であると，結論せざるを得ない」[28]とした。この後，控訴裁判所はファイル構造（file structure），スクリーン・アウトプット，およびサブルーチンの比較によって実質的類似性を肯定した地裁判決には，明らかな誤りはないと判断した。

(3) 抽象化・濾過テストによりプログラムの保護範囲を限定した CA 対 Altai 事件

次に取り上げるのは，Computer Associates（CA）対 Altai 事件[29]である。CA 社は IBM メインフレーム（大型コンピュータ）で実行できるジョブ・スケジューリング・プログラム（job scheduling program）[30]，「CA-SCHEDULER」（以下「CAS」）を開発した。この CAS は「ADAPTER」というサブプログラムを含んでいた。IBM コンピュータは大きさによって 3 つの異なる OS が搭載されているが，ADAPTER はこれらの OS いずれにおいても，CAS が実行できるようにするものである。その意味で，ADAPTER は翻訳者としての役目を果たしており，「OS 互換性部品（operating system compatibility component）」といえる。ADAPTER のようなプログラムを用いれば OS の種類に関係なくソフ

トウェアが利用でき，ユーザーにとってメリットが大きい。

Altai 社は「ZEKE」というジョブ・スケジューリング・プログラムを開発したが，これは，1 つの OS（「VSE」）で動くように設計されていた。顧客からの要望もあって，Altai 社は別の OS（「MVS」）で動く ZEKE の開発を決定し，それに当たり，元 CA 社員のプログラマ（以下，「P」）を雇い，「OSCAR 3.4」（以下，「3.4 版」）というプログラムを開発した。この 3.4 版を利用すれば，ZEKE が MVS でも利用できることになる。P は 3.4 版の開発に当たり，約 30％を ADAPTER からコピーした。CA 社からの提訴を受けこの事実を知った Altai 社は 3.4 版を合法的に書き換えることとし，P をプロジェクトから外し，ADAPTER のコードは利用せずに「OSCAR 3.5」（以下，「3.5 版」）を開発した。

3.4 版および 3.5 版は ADAPTER の著作権を侵害しているとの CA 社の申立てに対し（なお，3.5 版プログラムについては，コードは書き換えられたものの，ADAPTER の構造と実質的に類似していると CA 社は主張した），連邦地方裁判所は 3.4 版については約 30％のコピーを Altai 社も認めていたことから侵害を認定し損害賠償を命じる一方，3.5 版については Whelan 判決での考え方を否定し，実質的類似性はないとして侵害を認めなかった[31]。控訴審である第 2 巡回区連邦控訴裁判所も，3.5 版の著作権侵害を認めなかった。以下では，この控訴審判決の考え方[32]を紹介する。

まず控訴裁判所は，「Baker 事件で最高裁は，著作物が記述するアイデア，システムまたはプロセスに付随物として使用されることが必要不可欠な著作物の側面は，著作権保護の対象ではないと結論付けた……この理由付けから当裁判所は，コンピュータ・プログラムの要素のうちその機能にとって必要不可欠に付随する要素は同じように保護されないと結論する」[33]とした。 次に，実質的に同一の問題に直面した Whelan 事件を取り上げ，「Whelan 判決が打ち立てた表現からアイデアを区別する基準は概念的に広過ぎるとして，学界では評判が芳しくない。この分野の代表的な専門家は，『「（Whelan 判決が示した）理由付けの決定的な欠陥は，いかなるコンピュータ・プログラムであっても，著作権法上の用語としてたった 1 つの“アイデア”しか根底にはないとしており，ひとたび分離可能なアイデアが特定されれば，それ以外のものは全て表現であると考えていることである』と述べている……コンピュータ・プログラムの最終的な機能や目的は相互に作用するサブルーチンの複合結果である。それぞれのサブルーチンはそれ自体がプログラムであり，すなわち，それ自身の“アイデア”を有している。それゆえ，プログラム全体の目的がプログラムのアイデアと同じであるとする Whelan 判決の一般公式は説明として不十分である」[34]と述べて Whelan 判決を批判し，1 審判事が同判決に従わなかったことは賢明であったとする。そして，コンピュータ・プログラムの構造に関す

る実質的類似性のテストとして新たに，「抽象化・濾過・比較（abstraction-filtration-comparison）」テスト（以下，「AFC テスト」）を提示した[35]。

最初の段階である「抽象化」の作業は，表現をアイデアから分離する具体的な手法として Learnead Hand 判事が Nichols 事件[36]において提唱した「抽象化テスト」をプログラムに適用することで行う。抽象化テストというのは，最初に具体的な作品を想定し，そこから個別事象を次々に取り除いていく。そうすると，最終的には何についての作品かという一般的な説明やタイトルだけになるが，抽象化の作業のどこかの段階で，アイデアがもはや保護されない地点が存在し，そこでアイデアと表現の境界線が引かれる，というものである[37]。控訴裁判所は抽象化のステップに関して，「理論的な面でリバース・エンジニアリングに似た方法で，コピーされたと申し立てられているプログラムの構造を分解し，構造部分に含まれる抽象度のレベルを分離する。このプロセスはコードから始まり，プログラムの最終的な機能を明確にすることで終わる。その過程では，設計者のとった各ステップを，プログラムの作成時とは逆の順序で辿り，図で示していく必要がある」[38]と述べた。

AFC テストの第 2 段階である「濾過」の段階では，「抽象度の各レベルの構造上の要素を分析して，そのレベルに含まれる要素がアイデアか，または保護されない表現（①効率性を考慮したものであり，そのアイデアに必然的に付随するもの，②プログラム自体の外部要因が必要とするもの，あるいは，③パブリックドメインから取ったもの）かどうかを検討する」[39]。

最後の「比較」の段階では，濾過の作業によってふるいにかけられた結果，残った表現について，「被告がこの保護された表現のいずれかの側面をコピーしたかどうか，および，原告のプログラム全体に対するコピーされた部分の相対的な重要性に焦点を当てる」[40]。

控訴裁判所は，AFC テストについて以上の一般的な基準を明確にした後，地裁判決の検討に移った。地裁判決では，専門家証人がコード，パラメータ・リストとマクロ，サービスのリスト，および組織図の要素について比較分析を行い，その結果，ADAPTER と 3.5 版の実質的類似性を認めなかった。地裁判事は専門家の意見に基づき，両プログラムの実質的類似性を否定した。控訴裁判所は，この地裁判決の判断を支持するとの結論を下した。

この Altai 事件は，他のプログラムとの互換性を達成するために必要となるプログラムのインターフェイスが著作権の保護範囲に含まれるかについて控訴裁判所で争われた最初のものであると理解されている[41]。控訴裁判所は，濾過のステップにおいて，著

作権では保護されない要素として，①効率性によって決定される要素，②外部要因によって決定される要素，および，③パブリックドメインから取られた要素，を挙げた。そして②について，著作権法分野の著名な専門家の次のような考え方を好意的に引用し，その考え方を明らかにしている。「多くの場合，標準的な技術を利用せずに，ある特定のコンピュータ環境下で特定の機能を実行するプログラムを書くことは事実上，不可能である。これは，設計選択に関するプログラマの自由が以下のような外在的な考慮要因によって制限されていることがしばしばであることの結果である。すなわち，(i) 特定のプログラムが動くことを意図しているコンピュータの機械的仕様，(ii) 当該プログラムと連動して動作するよう設計された他のプログラムとの互換性要件，(iii) コンピュータ生産者の設計基準，(iv) サービスの提供を受ける業界の要望，(v) コンピュータ業界において広く受け入れられているプログラミング慣行，などである」[42]。

この控訴裁判所の一般的な考え方は，ADAPTER と 3.5 版の比較の作業において具体的に現れている。すなわち，パラメータ・リストとマクロについて，「ADAPTER の保護される要素の僅かしか類似しておらず，それ以外はパブリックドメインか，もしくはプログラムの機能的要求によって決定された。上記のように，機能的要素およびパブリックドメインから取得された要素は著作権保護の対象ではない。僅かに類似しているパラメータ・リストおよびマクロに関しては，プログラム全体に対する相対的な寄与を考慮すれば，侵害の認定を正当化しなかった地裁の結論は妥当であった」と指摘している [43]。また，ADAPTER と 3.5 版の両方に必要なサービスのリストに関するオーバーラップについても，「ADAPTER または 3.5 版を介して接続される OS とアプリケーション・プログラムの要求によって決定された」と述べた地裁判決を引用し，続けて，「言い換えれば，プログラム構造のこの側面は，相互作用するように設計された他のプログラムの性質によって決定されたのであり，それゆえ，著作権では保護されない」[44), 45)] とも述べている。

(4) メニュー・コマンドの階層構造を「操作方法」とした Lotus 対 Borland 事件

この事件は Lotus 社が開発した表計算ソフトとも呼ばれるスプレッドシート・プログラム，「Lotus1-2-3」の著作権を，Borland 社が開発した同種のプログラム，「Quattro」および「Quattro Pro」が侵害したかどうかで争われたものである。本稿で取り上げる第 1 巡回区連邦控訴裁判所 1995 年 3 月 9 日判決 [46)] およびその 1 審判決 [47)] 以前にも，両製品の間では，数度の判決が出されていた [48)]。

Lotus1-2-3 には「Copy」や「Print」といった 469 のコマンドが 50 以上のメニュー

やサブメニューに配置されている。Borland 社は，Lotus 社のメニュー・コマンドの階層を自社のプログラムに取り込み，Lotus1-2-3 との互換性を持たせることで，Lotus1-2-3 に慣れ親しんだユーザーが，新しいコマンドを学んだり，1-2-3 のマクロを書き換えたりすることなく，Borland 社のプログラムに乗り換えることができるようにした。その際，Borland 社は Lotus1-2-3 の基本的なコンピュータ・コードを一切コピーせず，メニュー・コマンドの階層の単語と構造のみをコピーした。第 1 審のマサチューセッツ地区連邦地方裁判所は，Borland 社の「キー・リーダー（Key Reader）」[49]は Lotus1-2-3 の著作権を侵害すると判断したので，Boland 社が控訴した。

控訴裁判所は以下のとおりの判断を示し，地裁判決を破棄した。

まず控訴裁判所は，Baker 事件の事案と本件とは同一であり，「Selden〔会計〕システムの“ユーザー・インターフェイス”がコンピュータではなく紙とペンによって提供されていたという違いだけである」との Borland 社の主張に対し，「Baker 事件と本件控訴審の事案とは Borland 社が主張するほどは類似しているとは思われない…… Selden とは異なり，Lotus 社は会計システムについての独占権を主張しているわけではない。むしろ，本控訴審ではコンピュータの操作に使用するコマンドに関する Lotus 社の独占が争点なのである」[50]と述べ，Borland 社の主張を認めなかった。

次に裁判所は，Altai 事件で提唱された「抽象化・濾過・比較」の 3 段階で行う AFC テストが適用可能かどうかを判断した。しかしながら，AFC テストが用いられた同事件は，あるプログラムが別のプログラムから非文字的な（nonliteral）表現をコピーしたかどうかが問題になった事案であったとして，本件に AFC テストを適用することは適切ではないとの考えを示している。すなわち，「Altai テストは，コンピュータ・コードの文字通りのコピーが主張されている事案を評価するには有益なフレームワークを提供するが，メニュー・コマンド階層の非文字的な複製が著作権侵害を構成するかどうかを評価する場合には，ほとんど役に立たないと当裁判所は考える。事実，この文脈で Altai テストを用いることは裁判所を実際，誤解に導くおそれがある。というのは，裁判所に様々なレベルに抽象化するように指示することで，文字通りコピーされた場合には，コピーを行った者が著作権侵害の責任を負うことになるような，著作権で保護される主題を含む基本的なレベルを認定することを促しているように思われるからである。その基本的な（あるいは文字通りの）レベルは，Altai 事件のような非文字的なコピーのケースでは問題にならないが，本控訴審の当該事案は，まさにそうしたケースである。Altai 事件での AFC テストと地方裁判所のテストが求めているように，メニュー・コマンド階層を個々の語句とメニューのレベルにまで抽象化し，その段階で表現からアイデアを

濾過することは，メニュー・コマンド階層について著作権が認められるかどうかという，より根本的な問題を，完全に不明確にしてしまうと考える」[51]。

控訴裁判所は以上のように述べたのに続けて，メニュー・コマンド階層の著作物性の評価に移った。そして，次のような説示の上，Lotus 1-2-3 のメニュー・コマンド階層は，著作権法 102 条(b)が著作権の保護対象から除外する「操作方法（method of operation）」[52]に該当すると判断した。「102 条(b)にいう『操作方法』とは，車やフードプロセッサー，あるいはコンピュータなど，人が何かを操作する手段を意味する。すなわち，何かを操作する方法を説明するテキストは，操作方法自体に著作権保護を拡大することはない。他の人々はその方法を自由に利用できるし，自らの言葉で説明することもできる。同様に，1 つの新しい操作方法が説明されずに利用されている場合でも，他の人々はその方法を利用することも説明することもできる。当裁判所は Lotus のメニュー・コマンド階層は著作物性が認められない『操作方法』だと判示する。Lotus のメニュー・コマンド階層は，ユーザーが Lotus 1-2-3 をコントロールし操作する手段を提供する……メニュー・コマンド階層がなければ，ユーザーは Lotus 1-2-3 の機能にアクセスし，コントロールし，そして実際に利用したりすることができない」[53]。

また，控訴裁判所は互換性あるいは相互運用性に関しても見解を示している。すなわち，「Lotus のメニュー・コマンドが『操作方法』であることは，プログラムの互換性を考慮すれば，より明確となる。Lotus 社の理論に従えば，ユーザーが複数の異なるプログラムを使用する場合，使用するプログラムごとに異なる方法で同じ操作を実行する方法を学習する必要がある……これは馬鹿げたこと（absurd）であると考える。プログラムを操作する多くの異なる方法，あるいは，階層的に配置されたコマンド用語のセットを使用してプログラムを操作する多くの異なる方法が存在し得るという事実は，選択された実際の操作方法を著作権の対象とすることにはならない。それは依然としてコンピュータを操作する方法として機能しているのであり，著作権の保護対象ではない」[54]と述べる。

さらに，次のようにも述べている。「〔著作権侵害を認めた〕地方裁判所の判決に従うなら，ユーザーが Lotus 1-2-3 において特定の操作を行うのに必要な時間を短縮するためにマクロを書いた場合，ユーザーは別のプログラムにおいて，同じ操作を行うのに必要な時間を短縮するためにそのマクロを使うことができない。ユーザーは別のプログラムのメニュー・コマンド階層を使用してマクロを書き直す必要がある。これは，マクロが明らかにユーザー自身の成果物であるにも拘わらずである。ユーザーがコンピュータに同じ操作を別の方法で行うことを強制することは，『操作方法』は著作権の対象ではな

いという議会の 102 条(b)の指示を無視するものであると，当裁判所は考える…… Lotus のメニュー・コマンド階層は，Lotus 1-2-3 のマクロの基礎となっているため，Lotus のメニュー・コマンド階層は『操作方法』である」[55]。

(5) 小　括

さて，これまで，プログラム著作物の保護範囲の画定に際して互換性や相互運用性の観点から見て重要な意義を有する米国の判例を紹介してきた。その結果，互換性を実現するのに必要な側面については，プログラム著作物の保護を認めないとの一般的な理解があることが分かる。その具体的手法としては，まず第一に，Altai 事件控訴審判決における抽象化・濾過・比較の 3 段階によって著作権の保護要素を特定するという AFC テストを挙げることができる。これは，濾過の段階において，当該プログラムと連動して動作するよう設計された他のプログラムとの互換性を，外部要因によって決定される要素の 1 つとして考慮し，これを著作権によって保護される対象から除くという手法である。第 2 巡回区の Altai 事件判決の考え方は，その後，① Sega 社のビデオ・ゲーム機で動くゲーム・カートリッジを，Accolade 社がリバース・エンジニアリングを行って作成販売した行為がフェア・ユースに当たるとされた Sega 対 Accolade 事件（第 9 巡回区）[56]，② Sony 社のプレイステーション用のゲームソフトをパソコン上で利用できるようにするエミュレータの作成のために行ったリバース・エンジニアリングがフェア・ユースに当たるとされた Sony 対 Connectix 事件（同）[57]，③ある OS で動くアプリケーション・プログラムを別の OS で動くようにする目的で，元の OS コードを一部コピーする行為には，著作権は及ばないとされた Bateman 対 Mnemonics 事件（第 11 巡回区）[58]，④プリンタのカートリッジに埋め込まれたプログラムを，互換品の製造を目的にコピーする行為は著作権侵害に当たらないとされた Lexmark 対 Static 事件（第 6 巡回区）[59]，⑤電話のコール・コントローラー機の開発に当たり，競争企業の機器との互換性を確保するために，競争企業のコマンドコードをコピーした行為が，ありふれた場面の法理（*scnes a afre*）によって侵害とならないと判断された Mitel 対 Iqtel 事件（第 10 巡回区）[60]などに反映されていると理解されている[61]。

なお，前記①の Sega 対 Accolade 事件はフェア・ユース法理の適用事例であるが，フェア・ユースの 4 つの判断要素の 1 つである「著作物の性質」の検討において，プログラムの実用性に照らせば Altai 判決のアプローチが適切であると指摘されている[62]。事件①と同じリバース・エンジニアリングの事例である事件②は，事件①の理解をほぼ踏襲している。

また，事件④および同⑤ではありふれた場面の法理が用いられている。ありふれた場面（または，ありふれた情景）の法理とは，特定の場面やシーンを描写する際に，誰でもがイメージする一般的で平凡な出来事や登場人物などに対して著作物性を認めないという考え方である。この法理は，Baker 事件最判がアイデアと表現の二分論から発展させたマージ理論からさらに派生したともいわれており，マージ理論はアイデアの表現方法が 1 つである場合を想定しているのに対し，ありふれた場面の法理は，表現方法は複数ある場合にも，通常用いられる表現についての保護を否定するという違いがある[63]。あるいは，マージ理論は事実的または機能的な著作物に用いられるが，ありふれた場面の法理の方はフィクションの作品に用いられ易いとも言われている[64]。このありふれた法理の適用についても，Altai 事件控訴審が打ち立てた AFC テストの分析手法で触れられている。すなわち，AFC テストの第 2 段階の濾過の作業では，著作権の保護が認められない要素を取り除くのであるが，その 1 つが当該プログラムの外的要因が必要とするため決定される要素であった。同判決は，次のように述べる。「当裁判所は，特定の『陳腐な（stock）』または標準的な文学的工夫を使わずに特定の歴史的時代または架空のテーマについて書くことが事実上不可能な場合，そのような表現は著作権で保護されない，と判示したことがある……これは，ありふれた場面の法理として知られているものであり，マージ理論と同様，コンピュータ・プログラムにも類推適用される」[65]。

互換性を実現するのに必要な側面についてプログラム著作物の保護を認めないとする別の手法としては，Lotus 事件控訴審が示したように，著作権法 102 条(b)が規定する著作権の保護対象から除外される「操作方法」に該当するとして，保護を否定するやり方がある。

以上のように，プログラムの互換性は強く達成されるべきものであり，したがって，著作権法上の保護を認めないとするのが各連邦巡回区の判例法であること，そして，その具体的手法として，① AFC テスト，②ありふれた場面の法理，③マージ理論，④ 102 条(b)の「操作方法」該当性，および⑤フェア・ユースが用いられてきたことが分かる[66]。

2．Java API とは何か

(1) 概　説

本最高裁判決の理解には，無断複製が問題となっている Java API（Java Application Programming Interface）がそもそもいかなるものかについての知識が大きな助けとなる。

そこで次に，Java を含め，Java API について説明することとする[67]。

Java は C や Python（パイソン），Perl（パール）などと同じく，プログラミング言語の 1 つである。プログラムは出来るだけ多くのプラットフォーム（具体的にはハードウェアや OS など）で動くことがユーザーから求められ，また，開発企業にとって戦略上有利である。この点 Java は仮想マシン（VM（Virtual Machine））という方式をとっていることが特徴的である。前記のプログラミング言語で書かれたプログラムはソースプログラム（またはソースコード）と呼ばれるが，コンピュータを動作させるためにオブジェクトプログラム（またはオブジェクトコード）への変換（コンパイル）が必要となる（このとき用いるプログラムを「コンパイラ」という）。Java のシステムでは，ソースコードはコンパイラを用いて，まず Java バイトコードと呼ばれる中間的な言語に変換される。このバイトコードは OS（Windows や macOS，Linux など）に依存せず，OS ごとに用意された Java VM（JVM とも呼ばれる）上で解釈・実行されることになる。別の言い方をすれば，Java 言語で書かれたプログラムは，JVM を通じて複数の OS 上で実行できるため，OS ごとにプログラムを書き直す必要がない。このことが，Java を開発した Sun Microsystems 社（以下，「Sun」）が掲げた「一度書けば，どこでも動く（Write once, run everywhere）」というスローガンの所以である。

Java の別の特性としてオブジェクト指向がある。これは，大型のプログラムを設計する場合でいえば，1 つの複雑で巨大なプログラムではなく，複数のモジュールと呼ばれる独立した部分（いわば「部品」）から構成されていると把握する考え方のことである。Java にはグラフィカル・ユーザーインターフェイスやデータベースへのアクセスなど，様々な機能を提供するクラスライブラリ[68]が予め標準で用意されており，これが Java API である。前記部品としての処理の最小単位は「メソッド（method）」と呼ばれるが，複数のメソッドを束ねる単位が「クラス（class）」であり，さらに，複数のクラスを集めたものを「パッケージ（package）」と呼ぶ。2012 年の本件連邦地方裁判所の認定によれば，「2008 年当時，Java API には 166 のパッケージがあり，600 以上のクラスに分割され，全部で 6,000 以上のメソッドに分割されていた。このことは，Java API には 166 のフォルダー（＝パッケージ）があり，その中には 6,000 以上のサブルーチン（＝メソッド）を実行するための 600 以上のプログラム（＝クラス）が事前に記述されている，ということに近い」[69]ということになる。また，本最高裁判決は，第一次 CAFC 判決を引用し，API というのは，「ある機能を実行するためにコードを自分で一から書くのではなく，予め書かれたコードを使用して自分のプログラムにその機能を組み込むことを可能にする」道具であると理解している[70]。Java API はインターネット上に公開されている。従っ

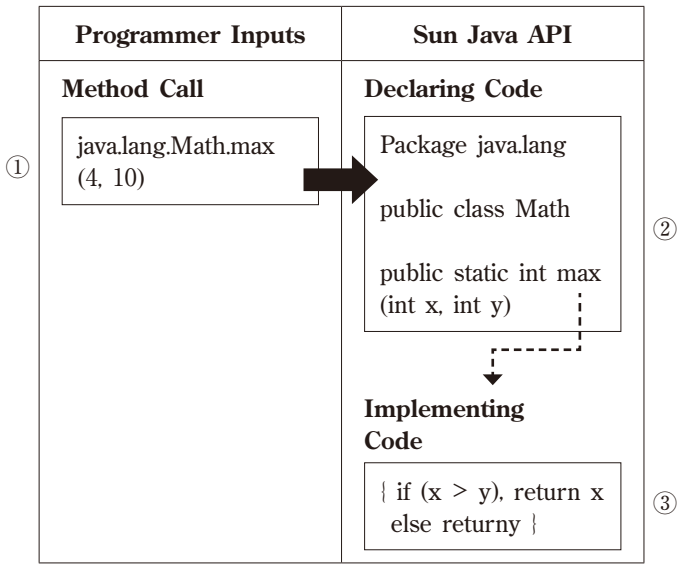
て，誰でもが自ら作成するプログラムの中にJava APIの機能を埋め込ませることで両者を連携させ，機能を共有させることができるのである。

⑵　最高裁の理解

次に，本件最高裁のJava APIについての理解を紹介しておく。

下記に示した**参考図**は本件最高裁の判決文に「APPENDIX B」として添付されているものである[71]。最高裁は「メソッド・コール（Method Call）」，「宣言コード（Declaring Code）」および「実行コード（Implementing Code）」という語句を用いて，Java API技術を説明している。上述したようにJava APIには6,000以上ものメソッドが準備されており，それぞれが様々なタスク（例えば，「2つの数字のうち，大きい方を示す」）を行う。このタスクを実際にコンピュータに実行するよう指示するコードが「実行コード」である[72]。プログラマは，多数の実行プログラムから目的のタスクを行う実行プログラムを選択し，コンピュータに伝えるために，自分のプログラムにそのタスクに対応するコマンドを入力して，呼び出す。こうしたコマンドを「メソッド・コール」という[73]。

参考図　「APPENDIX B」



プログラマが入力したメソッド・コールと，目的のタスクをコンピュータに実行させる実行コードとを結びつける役割を果たすのが「宣言コード」という別のコードである。参考図が示すように，宣言コードはJava APIの一部であり，プログラマが入力した特定のコマンドは，Java APIの中にある特定の宣言コードと一致する。宣言コードは，

それぞれのタスクの名前，Java API 全体の組織体系におけるタスクの位置（具体的には，当該メソッドがいずれのパッケージの，いずれのクラスに配置されているか）を示している。この意味で宣言コードとメソッド・コールはリンクを形成しており，プログラマは予め実行コードに書かれた多くのタスク（メソッド）を利用できる。

最高裁はこのような宣言コードには２つの重要な機能があると指摘する。１つは，プログラマのショートカットを可能にすることである。これは，非常に長いものになる可能性のある実行コードを自ら書く必要がなく，簡単なコマンド以上のことを学ぶ必要がない点を捉えたものである。最高裁はこのショートカット機能について，「車でいうアクセルペダル」，「タイプライターでいう QWERTY キーボード」になぞらえ，「宣言コードは人間と機械のインターフェイスの一部と考えることができる」[74]と述べる。

宣言コードのもう１つの機能は，「Java の開発者が様々なタスクの潜在的な世界を現実の世界に分割した方法を反映していることであり，具体的には，Java ベースのコンピュータ・システムを実行させたい潜在的な何百万もの様々なタスクのセットはどれなのか，また，それらのタスクをどのように整理しグループ化するかということである」[75]。やや分かりにくい表現であるが，最高裁は続けて「この意味で，宣言コードは組織的な機能を果たす。Java の開発者が構築することを決定したタスクライブラリの構造を決定している。この組織システムを理解するには，書籍をアクセス可能なシステムに分類するデューイ十進分類法……を思い浮かべればよい」[76]と述べていることから，数多くのメソッドの１つ１つはどのクラスに置かれているのか，さらに個々のクラスをどのパッケージに分類するのか，という構造化・組織化のことを指しているものと思われる。

さて，**参考図**ではプログラマが，「4 と 10 のうち，大きい方の数字を求める」という処理を自分のプログラムの中で行いたというケースを想定したものとなっている。Java API には「整数 x と y の大きい方の値を返す」というタスクを行う「max」[77]というクラスメソッド[78]が準備されている。メソッド「max」は，「Math」[79]というクラスの中に含まれており，クラス「Math」は「java.lang」[80]というパッケージに含まれている。このような場合，Java 言語でメソッド max を用いるときに，プログラマはまず「**java.lang**」と書く。この記述により当該パッケージが参照される。次に「**Math**」と書くことで，同様に「Math」クラスが参照される[81]。さらに，「**max**」と記述することで，当該メソッドが参照される。最後に「( )」（括弧）を設けて，その中に比較したい２つの整数（ここでは 4 と 10）を記入する。こうして出来上がった式全体（「**java.lang.Math.max**（**4,6**））」（＝メソッド・コール）を用いると，API によって，大きい方の数値を決定す

るタスクを実現するプログラムが呼び出される[82]。このプログラム（メソッド・コール）を作成する際，太字で表した記号を，上述したとおりの正確な順序で使用するが，記号を記述するだけでは何も行うことができない。特定のパッケージやクラス，そしてメソッドと結び付けるソフトウェアも使用する必要があり，それが API である。**参考図**が示すように，API はメソッド・コール（Method Call）の各部分（**参考図**①の部分）を特定のタスクを実行するプログラムと結び付ける宣言コード（Declaring Code）（同②の部分）と，そのタスクを実際に実行する実行コード（Implementing Code）（同③の部分）の両方が含まれている[83]。

3．本最高裁判決に至る訴訟経緯

(1) 提訴に至るまでの経緯と Oracle の主張

Oracle の提訴から本最高裁判決に至るまで，その時間的な長さもさることながら，途中での各裁判所の判断内容もやや複雑である。そこで，以下では，本件の訴訟経緯について明らかにすることにする。

2005 年，Google は Android, Inc.（以下，「Android」）を買収し，スマートフォンに代表されるモバイル機器用のソフトウェア・プラットフォーム[84]を構築しようとした。当時，多くのソフトウェア開発者は，Java SE（Java Platform Standard Edition）[85]プラットフォームを使用してデスクトップやラップトップのコンピュータで使用するためのプログラムを開発していた[86]。Java 言語を用いると，基盤となるハードウェアに関係なくプログラムを書くことができた。すなわち，プログラムの大部分は相互運用可能（interoperable）であった[87]。

Android を買収した後，Google はスマートフォン技術に対し，Java プラットフォーム全体をライセンスすることについて Sun と協議したが，最終的にはこの協議は決裂するに終わった[88]。Google はその後，独自の Android プラットフォームの構築を計画することになる。同プラットフォームは当然，スマートフォンの特性（例：制約のあるバッテリーで動くこと，GPS 技術を利用する可能性があることなど）に対応することが必要である。Android プラットフォームの構築のために，Google は何百万行のコードを新しく書いた。Google は，Java に慣れ親しんだ何百万人ものプログラマが，自分の Android プラットフォームを利用して簡単に作業できるようにしたいと考えた。そこで，Java SE プログラムから約 11,500 行のコードをコピーした。このコピーした行が Java API である。Android プラットフォームは商業的に成功し，2007 年のリリースから 5 年のうちに

Android ベースの機器は米国市場で大きなシェアを占め，2015 年の時点で Android の売上げは 420 億ドルを超えた。

2010 年，Oracle は Sun を買収し，その直後に，Oracle はカリフォルニア北部地区連邦地方裁判所（以下，「地方裁判所」または「地裁」ともいう）に本訴訟を提起した。Oracle の当初の申立ては，Google による Java API の使用は著作権法と特許法の両者に違反しているというものであった。このうち，著作権侵害の主張は，37 個のパッケージ（前記 11,500 行のコード）について，文字的な宣言コード，および Java API の非文字的な組織構造（すなわち，SSO）の両方をコピーすることで複製権を侵害した，というものである。なお，Google は API の実行コードは独自に開発し，Java API のそれは利用しておらず，このことについては争いがない。

(2) 各裁判所の判断

Oracle の提訴を受けた地方裁判所は，Java API の著作物性については裁判官が判断すべきであるが，その使用が著作権侵害に当たるかどうか，また，当たるとすればフェア・ユース抗弁が適用されるかどうかについては陪審員が判断すべきであると決定した[89]。陪審員は先に，Google による複製権侵害について認定したが，フェア・ユースの判断については議論が行き詰った。著作物性の検討において地裁は，当該第 9 巡回区の先例のみならず他の地区の先例[90]，さらには CONTU 報告書を取り上げて分析を行った。その際，第 9 巡回区の先例である Accolade 事件と Connectix 事件の検討において，「これら 2 つの判決の下では，相互運用性のために必要なインターフェイス手順は『互換性に必要な機能的要素』とみなされ，102 条(b)によって著作物性を有さないとされた」との指摘を行っている[91]。結局，地裁は Google がコピーした 37 パッケージに含まれるメソッド名やクラス名自体に著作権の保護は及ばないし，SSO についても，「37 パッケージのコマンド構造がコピーされた場合には，少なくとも，複製された当該コマンドを使用するコードの相互運用性が実現される」[92]と述べ，前述した Accolade 事件と Connectix 事件との類似性から，当該コマンド構造も 102 条(b)に規定する方式または操作方法であって，著作権の対象ではないと判断した。

控訴審である CAFC[93]は原審を破棄し，Java API の宣言コードとその組織構造の両方が著作権で保護される可能性があると判決した[94]。すなわち，Google は独自の宣言コードを書くことが可能であったこと，SSO についても表現の方法は複数あることを主たる根拠に，宣言コード，SSO のいずれも著作物性を有すると判断した。フェア・ユース抗弁適用の可否については，先例である Accolade 事件と Connectix 事件のいずれも

がリバース・エンジニアリングの過程で著作物性のない機能的な部分へアクセスした事案であり，他方，本件は宣言コードと Java API パッケージの全てをコピーしており，事案が異なるとした。そして，フェア・ユースについては事実認定が十分ではないとして，さらなる審理を求めて事件を地裁に差し戻した[95]。

Java API の 37 パッケージに著作物性が認められるとの前提の下，地方裁判所は Google のコピー行為がフェア・ユースに当たるかどうかの判断を行うこととなった。地裁は，フェア・ユースの適用の可否を判断する 4 つの要素[96]のいずれについても Google 有利であり，Google はフェア・ユースを立証したとの評価を行った陪審員の判断を肯定し，フェア・ユースの適用を認めた[97]。

Oracle が控訴したため舞台は CAFC に戻った。CAFC は Java API の 37 パッケージを Google が使用したことはフェア・ユースに当たらないと判断し，再び地裁判決を破棄した上で，損害賠償額の算定を行うよう地裁に差し戻した[98]。

これに対し，Google が最高裁に対してサーシオレイライの申立書を提出し，その中で，Java API の著作物性およびフェア・ユースについての CAFC の決定を見直すよう求めた。最高裁はこの申立てを認めた[99]。

Ⅱ　フェア・ユースに関する本最高裁判決の判断

連邦最高裁判所は，Java API が著作物性を有すると仮定した上でフェア・ユースについての検討を行い，その結果，Google によるコピーはフェア・ユースを構成するとして，当該コピーは著作権法に違反しないと判示した。本章では，本最高裁判決のフェア・ユースに関する判断を紹介するが，その前に，フェア・ユース規定と，最高裁が頻繁に依拠する Campbell 事件最高裁判決[100]について，簡単に触れておく。

1．フェア・ユース概観[101]

フェア・ユースは米国著作権法における権利制限規定の 1 つである（107 条）。侵害の成否を「フェアか否か」という一般的な基準で判断するため，一般的制限規定と称され，わが国における個別制限規定と対照されることが多い。107 条は，「第 106 条および第 106A 条の規定にかかわらず，批評，解説，ニュース報道，教授（教室における使用のために複数のコピーを作成する行為を含む），研究または調査等を目的とする著作権のある著

作物のフェア・ユース（コピーまたはレコードへの複製その他第 106 条に定める手段による使用を含む）は，著作権の侵害とならない」と規定する。そして，フェア・ユースの成否を考慮すべき主たる要素として，①使用の目的および性質（使用が商業性を有するかまたは非営利的教育目的かを含む），②著作権のある著作物の性質，③著作権のある著作物全体との関連における使用された部分の量および実質性，および，④著作権のある著作物の潜在的市場または価値に対する使用の影響，を挙げる（以下，これらを「第 1 要素」などと呼ぶことがある）。

フェア・ユースに関する最も重要な，そして影響のある最高裁判決が，Campbell 事件最高裁判決である。

本件は映画「プリティ・ウーマン」の主題歌，「Oh, Pretty Woman」（「原曲」）のパロディの事例である。原曲の著作権は Acuff-Rose 社が有している。Campbell らをメンバーとするラップミュージックのグループ，2 Live Crew が原曲のパロディソング，「Pretty Woman」を作成したことが，フェア・ユースに該当するかが主たる争点となった。最高裁は，第 1 要素（使用の目的および性質）の検討において，「ここでの探求の中心的な目的は，新しく作られる作品が，単に原作品に取って代わる（supersede）だけなのか，あるいは，さらなる目的や別の性質を伴って，新しい表現や意味，メッセージなどを追加しているかどうかを判断することである。換言すれば，その新しい作品がトランスフォーマティブ（transformative）[102] なものかどうか，その程度はどのくらいなのかを問うことである」[103]，「新しい作品がトランスフォーマティブであればあるほど，商業性といったフェア・ユースの認定に不利となる他の要因の重要性は低くなる」[104] との見解を示した。

従来，第 1 要素の判断として，商業的な利用の場合にはフェア・ユースが否定される傾向にあったが，最高裁の述べたことから分かるように，そのことは必ずしも決め手になるわけではなく，トランスフォーマティブな使用かどうかが最も重要なメルクマールであることを明らかにしている。また，このトランスフォーマティブの基準は第 4 要素（市場に対する影響）の判断においても，フェア・ユースを肯定する方向に傾ける重大な要素として捉えられている。このような Campbell 事件最判のフェア・ユースに関する判断手法は，その後の裁判例に大きな影響を与えている。

2．フェア・ユースに関する本最高裁の判断

⑴ 基 本 方 針

Google のサーシオレイライの申立ては，Java API が著作権で保護されるかという問

題と，Google による Java API の利用がフェア・ユースであるかという問題の 2 つを提起するが，最高裁は，「急速に変化する技術，経済，ビジネスに関連する状況を考えると，当裁判所は，純粋に議論のために，Sun Java API 全体が著作権保護を認める定義の範囲内にあると仮定する」[105]と述べた上で，フェア・ユースの検討のみを行った。そして，Google による Sun Java API のコピー，特にその API の 37 個のパッケージのための宣言コードと組織構造の使用がフェア・ユースに該当するかを判断することとした。

（2） 著作権のある著作物の性質（第 2 要素）[106]

最高裁は，まず，「Sun Java API はユーザー・インターフェイスである」と指摘した上で，Lotus 判決を引き，「それはユーザー（ここではプログラマ）がタスクを実行するコンピュータ・プログラムを，一連のメニュー・コマンドを通じて操作および制御することを可能にする手段を提供するものである」と述べる。そして，Java API の 3 つの部分，すなわち，実行コードとメソッド・コール，および宣言コードの関係を述べ，「本件で問題となっている宣言コードはコンピュータ・プログラムの一部であるという点では他の著作物と類似している。議会はプログラムが著作権の対象であることを明らかにした」と認めた。他方で，宣言コードは，Oracle が著作物性を有するとは争っていないメソッド・コールやコピーされなかった実行コードと密接に結び付いている点を指摘する。

次に，コピーされなかった実行コードとコピーされた宣言コードとの違いを強調する。すなわち，「証人の陪審員に対する説明によれば，実行コードを書くためにはコンピュータがタスクを処理する速度やコンピュータのメモリの大きさといった要因をバランスさせることが求められ……その創作性を API によって実行される『魔法（magic）』と証言した者もおり……こうしたことは，ノートパソコンやデスクトップではなく，スマートフォンという全く異なる状況で使用される Android ソフトウェアを開発するための優れた創作性である」と述べたが，宣言コードについては「異なる種類の創作性を具現化している。たとえば，Sun Java の開発者は，直感的で覚えやすい宣言コード名を見つけようとした」などと指摘し，「ユーザー中心の宣言コードと革新的な実行コードとの間に，臨界線を引く証言が数多く見られた」とした。

そして，「〔宣言コードの〕こうした機能は，ユーザー・インターフェイスの一部として，宣言コードがコンピュータ・プログラムの実行とは，ある程度異なっていることを意味している。他のコンピュータ・プログラム同様，宣言コードは本質的には機能である。しかし，多くの他のプログラムとは異なり，その使用は著作物性を有さないアイデア（= タスクの一般的な区分と組織化）と新規で創作性のある表現（= Android の宣言コード）と

が本質的に結び付いている。他の多くのプログラムとは異なり，その価値の大部分は，著作権を有さない人々，すなわちコンピュータ・プログラマが当該 API システムに対して自分たちの時間と労力を投資するという価値に由来するものである。さらに，他の多くのプログラムと異なり，その価値は，プログラマが，Google がコピーしていない Sun 関連の実行コードを利用できる（そしてその後も使い続けることができる）ようにするために，プログラマに対して Android システムの学習と利用を奨励する努力に基づいている」との理解を示した。

最後に，Campbell 事件最判を引き，「著作権は多種多様な著作（writing）を保護するが，著作権の核心（core）に近い著作物もそうでない著作物もあることを認識する必要があると，当裁判所は強調してきた」と述べた上で，「当裁判所の見解では前述した理由から，宣言コードは完全に著作物性を有するとしても，ほとんどのプログラム（実行コードのような）と比較して，著作権の核心から遠い。この事実は……本件でフェア・ユースを適用することは，議会がプログラムに認めた著作権保護一般を，著しく損なうのではないかというおそれを減ずる。そして，このことは，本（第 2）要素がフェア・ユース支持の方向を指示することを意味する」との評価を下した。

(3) 使用の目的および性質（第 1 要素）[107)]

本要素について最高裁は Campbell 最判に倣い，Google のコピー行為がトランスフォーマティブに当たるかという観点から主として検討を行った。そして，「Google による Java API の利用は，新しい製品の開発が目的である。Google は，Android ベースのスマートフォンの利用と有用性の拡大を目指している。Google の新製品はプログラマに対して，スマホ環境に関する非常に創造的かつ革新的なツールを提供する。プログラマが簡単に利用できるような新しいプラットフォームを構築する目的で，Google が Java API の一部を利用する範囲であれば，Google の利用は憲法が定める著作権の基本的な目的[108)]である創作的な『発展（progress）』と一致する[109)]」とまず述べた。

次に，Google のコピーの目的に関して，「Google は，スマホ向けのプログラムで使えるタスクを組み込むのに必要な範囲でのみ，（Sun がデスクトップ型またはノート型のパソコン向けに開発した）API をコピーした。そして，プログラマが使い慣れたプログラミング言語の一部を放棄して新しい言語を学習することなく，そうしたタスクをプログラマが呼び出すことを可能にするために必要な場合に限って，Google はコピーしたのである。繰り返すが，Google は Android を通じて，別個で異なるコンピューティング環境で動作するタスクの新しいコレクションを提供した。これらのタスクは，その新しい

環境下で動作するよう設計された（そして，Google が書いた）新規の実行コードの使用を通じて実行された。アミカス・ブリーフには Google の行為を『再実装（reinplementation）』と呼び，ある既存のシステムの『同じ文字と文法を再利用したシステムの構築』と定義づけるものがある。この場合，既存のシステムを学習したプログラマは，自分たちの基本的なスキルを新しいシステムにおいて利用できる」との評価を行った。そして，「異なるプログラム同士が互いに交信（speak）できるためには共有されたインターフェイスが必要であり[110]……プログラマが習得したスキルの利用を可能にするのであればインターフェイスの再実装が必要である[111]。API の再実装は，業界では一般的である。そして，Sun の幹部自身も，スマートフォンを含む Java 言語の広範な使用が，会社の利益をもたらすと考えていた」と述べた。判決は，証人が陪審員に対して説明した前記の点を，Google を擁護する幾つかのアミカスも以下のように要約しているとして，その要約を取り上げている。「Google が再実装した Java SE の部分は，より大きな Java 開発者コミュニティ内での使用の一貫性を維持することに役立つものである」（著作権法学者ら）。「機能的なコードの使用にフェア・ユースを合理的に認めることは，市場全体の成長にとって新たな機会を創造するというイノベーションを可能にする」（Microsoft 社）。「インターフェイスの再実装は，人気のあるプログラミング言語の広範な採用を促進した」（コンピュータ科学者）。「業界標準となったソフトウェアの主として機能的な要素についての著作権は，著作権者に反競争力を与える」（American Antitrust Institute）。

結論として，最高裁は，「こうした事実と関連する事実により，当裁判所は Google によるコピーの『目的と性質』がトランスフォーマティブであったと確信し，本（第 1）要素もまたフェア・ユース支持に傾く」と述べた。

(4) 使用された部分の量および実質性（第 3 要素）[112]

Google は Java API の 37 パッケージの宣言コードをコピーし，合計で 11,500 行のコードをコピーした。最高裁は，「これらコピーした行のコードは，数百の様々なタスクを呼び出すために必要な事実上すべての宣言コードをコピーした」と述べ，「宣言コード単独で考えるなら，その量は大きいものであった」と評価する。他方で，「Java API におけるソフトウェア素材の全一式を考えると，その量は小さく，実行コードを含む Java API コンピュータ・コードの一式の全体は 286 万行であり，そのうちの 11,500 行は 0.4 %に過ぎない」と述べた。このように，11, 500 行のコードを単独で考えるか全体の一部として捉えるかが問題となるが，最高裁はこの点に関して，「当裁判所は，少量のコピーであっても，コピーした抜粋部分が元の著作物の創作的表現の『核心（heart）』を構成

している場合には，フェア・ユースの範囲外であると判示している（1985 年の Harper & Row 最判（510 U.S.539））。他方で，比較的多くの量の素材をコピーしても，コピーした当該素材の創作的な表現の僅かしか捉えていない場合か，あるいは，コピーした者の正当な価値の本質である場合には，フェア・ユースの範囲内にある（Campbell 最判ほか）」との基本的な考え方を示した。

そして，「Google のコピーのいくつかの特徴は，Google がコピーしなかった数百万行を考慮した方が適切な数字であることを示唆する」と述べ，その特徴について以下のようにいう。「1 つには，Java API はタスクを実行する行と切り離せないほどに結び付いている。API の目的はそれらの行を呼び出すことである。別の特徴は，Google は行の創作性，行の美しさ，または（ある意味では）行が有する目的の故に行をコピーしたわけではない。Google がコピーした理由は，プログラマが Java API システムを動作させることを既に学習しており，それらの行なくしては，プログラマに対して Android スマートフォンシステムの構築に魅力を感じてもらうことは，おそらく桁外れに困難であったからである。さらに，Google の基本的な目的は，異なるコンピューティング環境（＝スマートフォン）向けに異なるタスク関連システムを構築し，その目的を達成し普及させる Android プラットフォームを構築することである」。そして，第 3 要素の解釈として，「〔第 3 要素の〕質問は，条文の第 1 要素に立ち返ることになる。というのは，許容されるコピーの程度は当該使用の目的と性質によって変わるからである」という Campbell 最判の説示を引用し，「『実質性』の要因は本件での事案のようにコピーの量が正当で，かつトランスフォーマティブな目的と結び付いていた場合には，フェア・ユースを支持する方に傾く」との考え方を明らかにした。

また，最高裁は，「Java 言語で書くために必要な 170 行をコピーするだけで，Java 互換の目的を達成することができたと」[113]との原判決である第二次 CAFC 判決には同意しないとして，「当裁判所の見解では，CAFC の結論は Google の正当な目的をあまりにも狭く捉えている。Google の基本的な目的は，単純に Android システム上で利用可能な Java プログラミング言語を開発することではなく，Android プラットフォームを利用してスマホ向けの新しいプログラムを書く際に，プログラマが Java API に用いた知識と経験を利用できるようにすることであった。原理的には，Google は独自の異なる宣言コードを書けたかもしれない。しかし陪審員は，独自の宣言コードを作成することでは，そうした Google の基本的な目的を達成できなかったと考えた。ある意味では，宣言コードは，プログラマの創作性のエネルギーを解き放つのに必要な鍵であった」との考えを示した。そして最後に，「当裁判所は，この『実質性』の要素は，フェア・ユース支持

に傾く」と述べた。

(5) 市場に対する影響（第4要素）[114]

最高裁は，本要素について，「〔著作権者の〕利益の損失が全てではない。金額だけではなく，損失の発生源も考慮する必要がある」との理解を示し，まずCampbell最判の指摘を引用して，「劇場に対する痛烈な批評のように，致命的なパロディはオリジナル作品の需要を消滅させるかもしれないが，この種の損害は，たとえ失われたドルに直接換算できるとしても，著作権法の下で認められるものではない」と述べた。また続けて，「公共の利益と著作権者の損失とを，本要素の下で衡量することが求められる」と判示したMCA判決[115]を引き，「さらに，複製がもたらす可能性のある公共の利益を考慮しなければならない。たとえば，これらの利益は新しい表現の創作的な生産に関する著作権上の懸念と関連しているか？損失の可能性のあるドルの金額と比較した場合（損失の源泉の性質も考慮した上で），それらの利益は相対的に重要か否か？」と述べた。そして，「これらの質問が……フェア・ユースの適用と常に関連しているとしても，また，裁判所が尋ねる唯一の質問ではない」と述べつつも，「しかしながら，Googleの再実装による市場への可能性のある影響を判断するのに役に立つという意味では，これらの利益は本事案の下で関連性があると考える」との理解を示した。

最高裁は，「可能性のある損失の量に関しては，AndroidがJava SEに対して現実のあるいは潜在的な市場に損害を与えていない，と陪審員は判断した。そして，Sun（現在のOracle）自体は，GoogleがAPIの一部をコピーしたかしなかったかに関係なく，そうした市場への参入に成功しなかったと判断した」と述べた。このことに関連して，最高裁は次のように述べている。「第一に，事実審理での証拠は，Androidスマホの技術とは関係なく，携帯電話市場で成功するためには厳しい立場にあったことを示している。Java SEの主たる市場はラップトップ型とデスクトップ型のPCであるという多くの証拠が存在した。また，Sunの携帯電話市場への進出に向けた多くの取り組みは成功しなかった。2006年まで遡ると，Androidのリリースに先立ち，Sunの幹部はスマートフォン技術の登場によって，携帯電話の収益が減少すると予測していた。Sunの前CEOは，Sunがスマートフォンを作らなかったのはGoogleのAndroid開発に起因しているかどうかを直接質問されると，そうではないと答えた。Sunが，携帯電話機の開発におけるビジネス上の課題に苦しんでいた証拠を考えると，陪審員がそうした評価に同意する資格がある。第二に，陪審員は，GoogleのAndroidプラットフォームを利用する機器は，Sunの技術をライセンスした機器とは種類が異なると繰り返し説明を受けた。

たとえば，複数の証人が，一般的な業界ではスマートフォンと比較的シンプルなフィーチャーフォンとを区別していると説明した……それ以外のモバイル機器でいえば，『キンドル（Kindle）』のような比較的シンプルな製品はJavaソフトウェアを使用するが，他方で，『キンドルファイア（Kindle Fire）』のようなより進化した技術はAndroid OS上で構築されていることが，証拠によって示された。こうした記録上の証拠は，より大きなコンピュータから小さなコンピュータにSunのコードを転用したのではなく，GoogleのAndroidプラットフォームはJavaソフトウェアと別個な（かつ，より発展した）市場の一部であることを示している」。そして，「これらの重要な違いに目を向けて，Googleの経済学の専門家は，AndroidはJavaソフトウェアを代替する市場ではないと陪審員に述べた。その専門家が説明したように，2つの製品は全く異なるデバイスに搭載されており，Androidプラットフォームは，単にアプリケーション・プログラミングのフレームワークであるJava SEとは全くタイプの異なる製品としてのモバイル操作全般のための技術である。まとめると，Sunの携帯電話ビジネスは縮小傾向にあったが，他方で，Sunが提供できなかったスマートフォン技術の新しい形式に対する市場の需要は増加したことを，証拠は示している」。

さらに最高裁は，「第4要素は，被告が行った種類の広範な行為が市場にどのような影響を与えるかを問うものである」[116]との説を引き，「陪審員は，Javaの訓練を受けたプログラマのネットワークをAndroidのような新しいプラットフォームがさらに拡大させるので，そのようなプラットフォームにおけるJavaプログラミング言語のより広範な利用からの利益をSunが予測したという証拠を聞かされた（APIの再実装が開始されると，APIが成功したことが分かる）。言い換えれば，陪審員はAndroidとJava SEが2つの異なる市場で稼働していると理解できた。そして，本件では2つの市場が存在するために，1つの市場（スマートフォン）で働くためにJava言語を学習したプログラマは，その後，そうした才能を，もう片方の市場（ラップトップ型PC）に持ち込むことができるのである」と述べた。

こうした理解に反対する証拠をOracleは提出したが，最高裁は次のような見解を示している。「確かに，SunがAndroid市場に参入しようとしたので，『市場に対する影響』という要素はフェア・ユース適用について部分的には不利に作用する，とCAFCは判断した[117]。しかし，そうした〔SunとGoogle間の〕ライセンス交渉は『Javaコードの実装』および両社間の『ブランディングと事業提携』といったトピックをカバーし，宣言コードの37パッケージよりもずっと多くのことに関係していた[118]。いずれにせよ，陪審員のフェア・ユースの〔成立を認める〕評決は，ライセンス獲得のためのSunの努

力も，Oracle の反対する証拠も――たとえ Google が Java API の一部を使用しなかったとしても―― Sun がスマートフォン市場へ参入することは困難であったことを示す証拠を覆すものではないことを意味している。一方，Google のコピーは，Android プラットフォームから莫大な金額を獲得することに寄与した。Java API の著作権の行使は，Oracle に対してこれらの資金のかなりの取り分を与えることになる。しかしながら，Oracle が，この金銭を受け取る権利についての理由と方法の両方を考えることが重要である。API やスプレッドシート・プログラムのような新しいインターフェイスが初めて市場に登場した場合，以前より見易い視覚画面といった表現力の豊かさという特性，あるいは，優れた機能性を理由に，新しいユーザーを引き付けることができる。しかし，時間が経過するにつれ，プログラマを含むユーザーがそれを単に使っているという別の理由によって価値がある可能性がある。その者たちは，すでにそれを動かす方法を学習しているのである[119)]。本件での記録は，この要因が Java API を利用したいという Google の強い欲求を説明しているという証拠に満ち溢れている。Android の収益力のこの源泉は，第三者（たとえばプログラマ）の Sun Java プログラムに対する投資と大いに関係がある。それに対し，Java API の開発に対する Sun の投資はあまり関係がない。著作権法は創作された著作物の操作方法の学習に対する第三者の投資の保護を目的としているとは信じられない[120)]」。

最後に最高裁は，「プログラマの Java API の学習に対する投資を考えると，Oracle による著作権の行使を本件で認めると，公衆に損害を与える危険性がある」との考え方を明らかにし，この点について，以下のように述べる。「プログラマにとって同じように魅力を持つ代替的な API を開発することのコストと困難性を考えるならば，本件で権利行使を認めると，Java API の宣言コードを新しいプログラムの将来の創作性を制限する錠前にしてしまう。Oracle だけが，その鍵を握ることになるだろう。この結果は，Oracle（または，コンピュータ・インターフェイスの著作権を有するその他の企業）にとって，非常に有益なものとなるであろう。しかしながら，これらの利益は，当該インターフェイスを用いて作業することを学習したユーザーによって開発された創作的な改良作品，新しいアプリケーション，および，新しい使用方法からもたらされた可能性がある。その程度において，前述した錠前は著作権法の基本的な創作目的を，促進するのではなく妨げることになる[121)]。結局，著作権はアイデアの創出と拡大という両方の経済的インセンティブをもたらすのであり，ユーザー・インターフェイスの再実装は新しいコンピュータ・コードが，より簡単に市場へ参入することを可能にする」。

最高裁は以上のように述べ，「Android の市場における Sun の競争能力の不確実性，

Sun の収益の損失源，および公衆に与える創作性に関する損害のリスク，これらを総合的に考慮すれば，本第 4 要素（市場への影響）も，フェア・ユース適用支持に傾く」と，締めくくった。

(6) 結　論

最高裁は，「Google がユーザー・インターフェイスを再実装し，ユーザーが蓄積した才能を新しいトランスフォーマティブなプログラムに生かすために必要なものだけを使用した場合，Google による Java API のコピーは，法律問題としてフェア・ユースであるという結論に達した」[122]として，原判決である第二次 CAFC 判決を破棄し，事件を差し戻した[123]。

なお，本判決は Breyer 判事が法廷意見を書き，これに Roberts，Sotomayor，Kagan，Gorsuch および Kavanaugh の各判事が参加する一方，Thomas 判事が反対意見を書いた（Alito 判事が参加）。この反対意見では，宣言コードも著作物性が認められること，また，フェア・ユースは認められないとの見解が示されている。

Ⅲ　検　討

1．本最高裁判決の意義と特徴

本稿の「はじめに」でも指摘したように，本最高裁判決の意義を大きな観点から捉えるならば，コンピュータ・プログラムの保護範囲を巡る議論に大きな足跡を残したことにある。以下では，本判決の判断の手法やその内容について，特に上記のような観点から検討を加えることにする。

連邦最高裁判所は，Java API の著作物性自体の判断を行わず，Google による Java API の 37 パッケージのコピーがフェア・ユースに当たるかどうかを検討し，そして，「当たる」との結論を導き出した。Java API の著作物性については第一次地裁判決が否定する判断を示したものの，第一次 CAFC 判決がこれを覆し，Java API の著作物性を肯定する判断を示して以降は，裁判所での議論の中心はフェア・ユース成立の可否に移っていた。本最高裁において改めて Java API の著作物について判断がなされることが期待されていたが[124]，最高裁はこの問題を回避した。この点についての詳しい検討は次の「2. 相互運用性の重要性」で行うこととし，最高裁が示したフェア・ユースの判断

について先に検討を行う。

最高裁のフェア・ユースについての判断には注目すべき点がいくつかある。最高裁の判断の順に従ってみていくと，まず第2要素（著作物の性質）に関しては[125]，Java APIをユーザー・インターフェイスであると指摘し，その性質を確認・決定している。そして，Sunが作成したJava APIを宣言コードと実行コードとに明確に区分し，前者は著作物であるにしても，実行コードのようなプログラムと比べると，著作権の核心から遠いという考え方を明らかにしている。著作物には美術芸術作品のようなものと，プログラムやデータベースのような機能が重視される著作物とが存在しており，両者に対する保護のあり方が違ってくることについては一般的なコンセンサスがあり，著作権の核心から遠い・近いという発想は目新しいものではない。注目すべきは，その「遠い」理由である。すなわち，本最高裁判決は，APIの価値がプログラマによる時間と労力の投資にあると述べており，さらには，プログラマにAndroidシステムの学習と利用を奨励するGoogleの努力に基づいているとも述べており，単純にAPIが機能であることに理由を求めていない。

第1要素（使用の目的および性質）の検討では，Campbell事件最判以降の多くのフェア・ユースの裁判例と同じく，トランスファーマティブの判断が中心となっている。トランスフォーマティブが何かについては，その実態あるいは範囲を正確に把握することは難しい。結局は，Campbell事件最判の示した「さらなる目的や別の性質を伴って，新しい何かを追加すること」の文言と，これまでの裁判例から浮き彫りにしていくしかない。何かを追加するとは，元の著作物に何かが追加され，その結果，変形する必要は必ずしもない。たとえば，Perfect 10事件[126]では，画像検索のためにサムネイル画像をそのまま利用する行為がトランスフォーマティブであると認められている。しかしながら，この事件では，画像検索という『さらなる目的や別の性質』を伴っている。これに対し，本件でのGoogleによる再実装はJava APIのそのままの利用であり，しかも，Java言語を用いたプログラムの作成という目的あるいは性質も変化がないようにも見える。しかしながら本件最高裁は，Androidプラットフォームでの利用を，「さらなる目的や別の性質」と捉えている。そのことによって，Androidスマホで動く新たなプログラムの創作が促されることを重視したのであろう。このようなトランスフォーマティブに対する本件最高裁の考え方は，業界慣行となっている再実装にお墨付きを与えたという意味で重要であろう[127]。また，第一開発者の立場から言えば，APIのインターフェイス部分については著作権の行使は許されないことになるが，このことの理由として，「業界標準となったソフトウェアの主として機能的な要素についての著作権は，著作権者に反競

争力を与える」とのアミカス・ブリーフを肯定的に引用している点も注目に値する[128]。

第3の要素（使用された部分の量および実質性）については，「量」よりも「実質性」が相対的には重要であり，そして，この実質性の判断には第1要素の内容である「目的と性質」を勘案するとの考え方が表明されている。より具体的には，実質性がトラスフォーマティブな目的と結び付いており，量もその目的を達成する範囲内であれば正当とみなされ，本要素の基準を満たすというものである。結局，第3要素についての判断の特徴は，最高裁自身が述べるように第1要素に立ち返ることになるのであって，Google のコピーの目的が Java API を学習済みのプログラマに対して Android スマホ向けのシステム構築（＝再実装）に関心を持ってもらうという事実を重視していることにあるといえる。

最高裁は再実装を非常に重要なことと把握しているため，Google が別の宣言コードを書くことができたとしても，Java API の宣言コードをコピーすることは許容されるとの考え方を示した点も重要であろう。

最後に，第4要素（市場に対する影響）についても，注目すべき考え方を最高裁はいくつか明らかにしている。まず，影響を受ける市場には著作権者にとって現在の市場と将来の市場（潜在的市場）が基本となる[129]。本件の場合は，Oracle（あるいは Sun）にとっての潜在的市場の範囲が問題となっているが，Android スマホ関連の市場への参入について，単なる可能性ではなく，参入に向けての努力，さらにはそれが奏功しているか否かということまで考慮して判断している。この点と関連して，Java 言語が元々用いられているパソコンと Android スマホとは別の市場と捉えていることも特徴的である。

次に，Google が Android プラットフォーム市場から獲得した利益の帰属について，Oracle に認めるべきではないという考慮を，最高裁が行っていることも注目に値する。具体的には，Google がこの利益を受け取る正当性の根拠として，プログラマによる Java API の学習に対する投資を指摘し，こうした投資は著作権法上，保護に値しないと説示している。

さらに，Oracle の著作権行使を認めてしまうと Java API を Oracle に独占させることになり，新しいプログラムの創作に対して悪影響を及ぼすという「公衆の利益に対する損害」を判断材料の1つに入れていることも興味深い。

2．相互運用性の重要性

コンピュータは装置（ハードウェア）として単体で利用されるものではなく，プログラム（ソフトウェア）との連携を前提としている。また，プログラムも大きく OS（オペレー

ティング・システム）とアプリケーション・プログラムとに大別できる。そして，このようなハードとハード，ソフトとソフト，あるいはハードとソフトの間で相互運用性や互換性を確保することがコンピュータの世界では必須である。

プログラム著作物に関連する問題の１つとして相互運用性は古くから認識されてきた[130]。また，相互運用性の確保はネットワーク化が進展する今日にあっては，ますますその重要性を高めている。第Ⅰ章１で紹介・確認したように，米国判例法においては，プログラムの相互運用性は強く達成されるべきであって，それゆえ，著作権による保護を認めないとする立場を複数の巡回区が明確にしてきた。本最高裁判決を以上のような状況に照らして評価するならば，相互運用性の実現に大きく関連している Java API について，その著作権法上の保護範囲を適切に画定あるいは限定したといえる。

もっとも，その手立てとして，最高裁はJava APIの著作物性を否定するのではなく，フェア・ユース規定を用いた。著作物性の判断を行わなかった理由について最高裁は，「急速に変化する技術，経済，ビジネスに関連する状況」を指摘する。本件に対して提出されたアミカス・ブリーフにおいては多種多様な主張がなされている。たとえば，既に紹介したように 72 人の知財法学者のアミカスでは，「互換性を可能にするインターフェイスは，著作権の制約から自由であるべきである」[131]との見解が示されている。また，「コンピュータ・システムやデバイスにとって API という機能的に不可欠な要素は……実行コードのマージしていない（non-merged）側面は保護可能であったとしても，著作権保護の範囲外である」[132]と主張する著名学者らのアミカスもある。他方で，別の学者であり元 CONTU 委員は，「CONTU は，ソフトウェアを他の全ての言語著作物と全く同列に扱うべきであると勧告し，議会もこれに同意したのである。機能性を有していることも大衆に受け容れられていることも，創作的な著作物を著作権保護から除外したり，侵害者の行為を許すものではない」[133]と主張していた。さらには，最高裁判事の中でも見解が異なっていたこと[134]に鑑みれば，最高裁の判断手法も致し方なかったのかもしれない[135]。しかしながら，著作物性の判断を行わなかった最高裁の前記理由は，ICT 業界に対する一般的観測としては妥当なものだとしても，だからこそ，API の著作物性を判断する基準を明確にすることは，当業界の発展という意味で重要であり，最高裁はその機会を放棄したともいえ[136]，より踏み込んだ見解を提示してもよかったと思われる。また，フェア・ユースの成否の判断は著作物性の判断と比べて相対的に複雑で困難であるため[137]，予測可能性の点でも，著作物性について判断を行うべきであったと考える[138]。

ところで，本最高裁判決は Java API の著作権保護の範囲を限定して捉えたものの，

その根拠は相互運用性を主たる根拠にはしていないように見える。すなわち，判決文には相互運用性（interoperability）や互換性（compatibility）の語句は殆ど現れていない。この理由として1つは，「最高裁の意見としては，相互運用性それ自体は重要ではなかった。すなわち，Google の利用によって，第三者的な立場にいるプログラマが，自分の知っているプログラミング言語を使い続けることができたかどうかが，重要なのであった」[139]と考えることができる。最高裁のこのような理解を推し進めると，Google による利用の評価に相互運用性の達成は無関係であり，互換性がなくとも許容されることになる[140]。もう1つの読み方は，「最高裁が本件を厳密な意味で互換性の訴訟とは考えていなかった」[141]というものである。確かに，本件は Altai 事件や Accolade 事件のように互換機の製作が問題とはなっているわけではない。しかしながら，完全な互換性でなく部分的な互換性であっても，新しいプログラムの作成，一般ユーザーとプログラマの利便，技術革新や関連業界の発展などに鑑みれば，十分な意味があると思われる[142]。

3．わが国への示唆

本件のような事案がわが国で争われた場合，どのように処理され，あるいは処理すべきであろうか。本件は米国での事例であるが，Java 言語は日本でも人気が高く，これを利用するプログラマの数は多い。また，API の再実装は ICT 関連業界において一般的に行われている慣行であること，さらには情報ネットワーク化の進展は世界的な潮流であり，わが国もその例外でないことに鑑みれば，そのような検討は意義深いと思われる。

まず，基本的視座として，プログラム著作物に関しては，相互運用性を確保するために必要となる API[143]は，自由利用を広く認めるべきであろう[144]。次に，その手立てであるが，これまで議論してきたように，API の相互運用性を確保するための機能的要素の著作物性を否定するという手法と，著作物性を肯定した上で（あるいはこの判断を保留した上で），何らかの形で権利（の行使）の制限を行うという手法の2つがあり得る。

(1) 著作物性の否定

最初の著作物性を否定する手法には，① 10 条 3 項 2 号の適用と②創作性の否定が考えられる。著作権法 10 条 3 項 2 号はプログラム著作物に対する保護が，「規約」には及ばない旨を明示するものである。同号は「規約」を，「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」と規定している。本号の立法趣旨は，「プログラムを作成する場合には，プログラム言語といういわば一般的なルールの他に，特

定のプログラムと連結して使うためにそれが用いている一定のルールに則らなければならないということがあります。例えば同一電子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がありますし，また，通信回線で結ばれた複数の電子計算機においてデータをやりとりして一連の処理をするために約束事を共通にする必要があるといった具合です。このような場合の特別の約束が規約でございます」[145]と説明されている。また，「規約とは……具体的には，インターフェイスやプロトコールのことである。〔中略〕プログラムの場合は……ネットワーク化のため，あるいは互換性を保つために，著作権保護を与えないということには格別の重要な意味を有する」[146]との理解もある。こうした考え方を踏まえた上で，API の特性および役割を考慮するならば，API が 10 条 3 項 2 号にいう「規約」に当たるとの解釈は合理的であろう[147]。

次に，創作性を否定するとの手法であるが，具体的には，マージ理論の適用が可能であると思われる。第Ⅰ章 1⑴および⑶で紹介したように，同理論は Baker 事件最判に起源を有するものであり，Altai 事件では AFC テストの濾過の段階においてプログラム著作物についても応用されている米国判例法由来のものであるが，わが国でもマージ理論は，以下の記述が明示するように，少なくともその趣旨については学説・判例とも広く受け入れている状況にある。「ある思想の表現方法が一つに，あるいは相当程度に限定されている場合には，著作物性を認めるべきではないという結論には，判例・学説上コンセンサスがあり，混同（マージャー merger）理論とも呼ばれている。思想と表現が混同するような場合には，表現の選択の幅がない，あるいは限定されているので創作性欠如により著作物とはならないと理論構成することが可能であろう」[148]。また，前述した「規約」を説明する箇所においてであるが，「通常の著作物と比し，インターフェイスやプロトコールのプログラムは，あるルールを前提とする以上，その記述の自由度は極めて狭い。……そのようなものに著作物性を認めると，著作権は形式的には表現の保護であっても実質的にはアイディアの保護となってしまうおそれがある」[149]との理解が示されている。以上のことから，相互運用性や互換性を実現するために利用しなければならない API の要素については，自由度あるいは選択の余地が極めて限られており，創作性が生じる余地がないことを理由に，当該 API の著作物性を認めないとの解釈を導き出すことは可能であろう。

⑵ 権利の制限

周知のとおり，わが国には米国のフェア・ユースのような一般的な権利制限規定は存

在しない。したがって，APIの著作物性が肯定された場合，Googleが行ったようなコピー行為を許容するには，著作権法30条以下に列挙されている個別の制限規定のいずれかを活用することが，順序としては最初に考えることになる。現行の規定で最も可能性の高いのは30条の4であろう。同条は，「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」については，いずれの方法によるかを問わず，利用できる旨を定めている（柱書）。そして，その具体的場面の1つとして，「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に供する場合」（3号）を例示している。本3号については，「プログラムの調査解析を目的とするプログラム著作物の利用（いわゆる「リバース・エンジニアリング」）」が該当すると説明されており[150]，米国のSega対Accolade事件およびSony対Connectix事件においてリバース・エンジニアリングをフェア・ユース規定の適用によって合法化していることも考えると，両者の類似性も窺い知れ，一見すると可能性がありそうである。しかしながら，柱書にある「表現された思想又は感情の享受」の文言について，「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」[151]と指摘されている。この指摘を踏まえれば，「本件におけるGoogleによる宣言コードの利用は，その機能を享受する利用に他ならないので，同条の適用の余地はない」[152]という保守的な解釈に落ち着き，結局，本条の適用は困難ということになる[153]。

ところで，第Ⅱ章2.⑸で紹介したように，本最高裁判決はフェア・ユースの第4要素（市場に対する影響）の検討において，Java ソフトウェアの市場とAndroidスマホに関連する市場は別の市場と捉え，後者の市場から得られる利益をOracleに与えることは適切ではないとの理解を示している。その根拠として，Java APIの価値が，プログラマによるJava言語の学習に対する投資にある点を重視している。また，Java APIの独占を認めることは新たなプログラムの創作活動を阻害することとなり，また，Sega事件を引用して，他者が競争できないようにして市場を独占する試みは，創作活動の促進という制定法上の目的に反すると述べている。以上のような最高裁の考え方は，Oracleによる著作権の行使は著作権法が権利者に認めた独占の範囲を不当に拡大し，また，競争上の弊害を引き起こすものであり，従って，こうした著作権の行使は制限されるべきである，と整理できよう。そして，このように整理した最高裁の考え方は，わが国の民法が定める権利濫用（1条3項）の趣旨と共通性を有し，重なるところが多であると評価できる。

いずれも特許法の分野の裁判例であるが，FRAND（公正，合理的かつ無差別な）条件で

ライセンスする約束をした標準必須特許に基づく差止請求を権利濫用に基づき否定したアップル対サムスン特許権濫用事件（知財高決平成26年5月16日判時2224号89頁），トナーカートリッジの再生品の製造販売を妨げる措置が独禁法違反（不公正な取引方法・取引妨害（一般指定14項））に抵触し，特許権に基づく差止請求が権利濫用（民1条3項）に当たるとされたリサイクルトナーカートリッジ事件（東京地判令和2年7月22日・平成29年（ワ）第40337号）がある。前者は特許権者の信義則違反を，後者は独禁法違反を権利濫用の根拠としているが，これらの裁判例と同様に，相互運用性を確保するために行うAPIのコピー行為に対して著作権を主張することは，法が認める著作権独占の不当な拡大であって新たな著作物の創作を通じて文化の発展を図るという著作権法の趣旨に反すること，あるいは，別の市場の独占を図る行為や競争を不当に妨げる行為であって独禁法に違反（具体的には私的独占，不公正な取引方法（取引拒絶や取引妨害など））することを根拠にして，権利濫用に該当するとの評価を下し，そのような著作権の主張を斥けることは十分可能であると思われる。

おわりに

コンピュータの世界では相互運用性あるいは互換性の達成が極めて重要である。このことにより，様々なハードウェアとソフトウェアが連動し，多種多様で複雑なタスクの処理が実現可能となる。また，こうした環境が確保されているからこそ，新たなプログラムの創作が促される。情報のネットワーク化が進展する今日にあっては，このような視点の意義はますます高まるであろう。プログラム著作物の保護範囲については市場独占の弊害に注意を払い，新たな創作活動の促進という点を考慮して画定することが求められる。本最高裁判決はこのことを再確認させるものである。わが国においても本最高裁判決を契機として，今後，議論を大いに活発化させることが求められる。

〈追記〉校正の段階で，小林和人＝齋藤歩記「Java API複製にフェアユース適用を認めた米国連邦最高裁判決―― Google LLC v. Oracle America, Inc., 141 S.Ct. 1183 (2021)」パテント74巻13号65頁（2021年）に接した。同稿は，①フェア・ユースの第1要件の検討の際に，Java APIをAndroidベースのスマホへ適用することがトラスフォーマティブであると判断したこと，②判決の結果，GoogleはJava APIを無償で利用することが出来ることになり，FRAND（合理的かつ被差別的な）条件でのロイヤリティ支払といった解決があり得たのではないか，という

点で，最高裁の判断に疑問を呈している。

〈付記〉本研究は，科研費 JSPS 科研費（課題番号：JP20K01437）の助成を受けたものである。

注

1） Google LLC v. Oracle America, Inc., 141 S.Ct. 1183（2021）.

2） 本最高裁判決をインターフェイス（interface）と相互運用性（interoperability）をキーワードに分析するものとして，Mark A. Lemley & Pamela Samuelson, *Interfaces and Interoperability After Google v. Oracle,*100 TEXAS L. REV.（2021）. また，本稿では，水越尚子「Google 社 v. Oracle 社米国最高裁判決― Google 社のフェアユースを認めた最高裁判決の意義及び日本法下での解釈について」コピライト 725 号 47 頁（2021 年），奥邨弘司「Google v. Oracle 事件合衆国最高裁判決― Java API を実現するプログラムのフェア・ユースについて」NBL 1202 号 20 頁（2021 年）も参考とした。

3） 米国の改正の経緯については，中山信弘「アメリカ著作権法とコンピュータ・プログラム」『法学協会百周年記念論文集〔第 3 巻〕』（有斐閣，1983 年）507 頁参照。

4） 米国著作権法第 101 条は，「『コンピュータ・プログラム』とは，一定の結果を得るためにコンピュータで直接または間接に使用される，文または命令の集合をいう。」と規定する。なお，米国著作権法の邦訳について本稿では，山本隆司訳『外国著作権法令集（56）―アメリカ編―』（著作権情報センター，2018 年）に拠った。

5） 米国議会技術評価局（Office of Technology Assessment；OTA ）の 1986 年報告書（Intellectual Property Rights in an Age of Electronics and Information）65 頁は，著作物を芸術的著作物（Works of art），事実的著作物（Works of fact），および機能的著作物（Works of function）に 3 分類し，機能的著作物の代表例としてプログラムを挙げている。

6） 複数の異なるハードウェアやソフトウェアを組み合わせたときに，全体として動作することを一般に，相互運用性（interoperability）と呼んでいる。他方，互換性（compatibility）とは，例えば，A というコンピュータ（ハード）が X という OS で動作しているときに，X で動く別のコンピュータ B を開発した場合，B は A と互換性があるとか，B は A の互換機であるなどという。したがって，厳密な意味としては，互換性は相互運用性とは異なる。しかし，B も X と連携して利用できるという意味では相互運用性の問題といえる。本稿では，基本的には両者を区別せず同義として用いている。

7） プログラムの実質的な保護範囲は，争いとなっているプログラムが著作物であるか否か（著作物性の有無）のみならず，権利侵害が否定されるフェア・ユース規定等の制限規定に該当するかといった議論を通じても画定される。

8） 1985 年頃までの判決例を概観するものとして，D.S. カージャラ＝椙山敬士『〔日本アメリカ〕コンピュータ・著作権法』（日本評論社，1989 年）159 頁以下〔椙山執筆〕，それ以降 1999 年頃までの状況については，椙山敬士『ソフトウェアの著作権・特許権』（日本評論社，1999 年）14 頁以下をそれぞれ参照。

9） Lemley = Samuelson, *supra* note 2, at 33 n.197 によれば，アミカス・ブリーフの内訳は，Oracle を支持するもの 31 通，Google を支持するもの 23 通である。これらアミカス・ブリーフは以下の合衆国最高裁判所のリンク先から入手可能である。https://www.supremecourt.gov/search.aspx ?filename = /docket/docketfiles/html/public/18-956.html（last visited Dec. 15, 2021）.

10） 本最高裁判決が出される以前の段階において，アミカス・ブリーフの様々な見解を紹介・検討するものとして，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」Nextcom42 号 4 頁（2020 年）がある。

11） Brief of 72 Intellectual Property Scholars as Amici Curiae in Support of Petitioner.

12） Oracle America, Inc. v. Google Inc., 750 F.3d 1339 (Fed. Cir. 2014)（以下，「第一次CAFC判決」ともいう）。

13） この改正に関する当時の状況について詳しくは，中山信弘『ソフトウェアの法的保護〔新版〕』（有斐閣，1988 年）9 頁，半田正夫＝紋谷暢男編『著作権のノウハウ〔第 6 版〕』（有斐閣，2002 年）35 頁〔清水幸雄執筆〕など参照。なお，「発明」（特許 2 条 1 項）の定義に該当すればプログラムは特許法でも保護されるのであり，両者の保護が相互に排他的なわけではない。

14） わが国の判例の状況については，高部眞規子『実務詳説　著作権訴訟〔第 2 版〕』（金融財政事情研究会，2019 年）349 頁以下参照。

15） E.g., Lemley = Samuelson, *supra* note 2. また，同論文の執筆者の一人である Pamela Samuelson 教授は，72 人の知財法学者が名を連ねるアミカス・ブリーフ・前掲注（11）を，Catherine Crump 教授と共に執筆しており，当該アミカス・ブリーフも，Java API の著作物性を認めることは，相互運用性を実現するためのインターフェイスに著作権保護を認めてこなかった従来の判例法理に反する旨を主張している。

16） Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240（3d Cir. 1983）は人間が読むことができないオブジェクト・コードの著作物性を否定していた原判決（Apple Computer, Inc. v. Franklin Computer Corp., 545 F.Supp. 812（E.D.Pa. 1982））の判断を覆した。その際，1980 年著作権法改正を勧告した「著作権のある著作物の新技術による使用に関する国家委員会（National Commission on New Technological Uses of Copyrighted Works；以下，「CONTU」）の 1987 年最終報告書などに言及して，著作権法第 102 条（a）は，「コンピュータ・プログラムを著作物として明確に例示しているわけではないが，立法経緯は，プログラムが言語著作物として著作権により保護可能であると考えていたことを示唆する」（***Franklin,*** 714 F.2d at 1247）と判示している。

17） Baker v. Selden, 101 U.S. 99 (1879). 日本語での紹介として，白鳥綱重『アメリカ著作権法入門』（信山社，2004 年）77 頁以下，カージャラ＝椙山・前掲注（8）231 頁〔椙山執筆〕など。

18） ***Baker,*** 101 U.S. at 103.

19） ***Id.*** at 107.

20） 白鳥・前掲注（17）79 頁，山本隆司『アメリカ著作権法の基礎知識〔第 2 版〕』（太田出版，2008 年）57 頁など参照。

21） Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc., 609 F.Supp. 1307 (E.D.Penn. 1985). 本地裁判決の評釈として，椙山敬士「ウェーラン対ジャスロー判決の射程―アイデアと表現の二分法」法とコンピュータ 6 号 97 頁（1988 年）。

22） Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc., 797 F.2d 1222 (3d Cir. 1986). 本控訴審判決の評釈として，佐野稔「プログラムの『構造』も著作権で保護されるべきか」日経コンピュータ 1987 年 6 月 8 日号 61 頁。また，カージャラ＝椙山・前掲注（8）118 頁以下〔カージャラ執筆〕および 168 頁以下〔椙山執筆〕は 1 審，控訴審を含めて Whelan 事件を詳細に分析・検討する。

23） ***Whelan,*** 797 F.2d at 1248.

24） 構造（structure），順序（sequence）および組織（organization）の各用語は，控訴裁判所は同じ意味で用いるとしている（***Whelan,*** 797 F.2d at 1224 n.1）。

25） ***Whelan,*** 797 F.2d at 1236.

26） ***Id.*** at 1236.

27） ***Id.*** at 1238.

28） ***Id.*** at 1239（citing ***Whelan,*** 609 F.Supp. at 1320）.

29） Computer Associates Int'l, Inc. v. Altai, Inc., 982 F.2d 693（2d Cir. 1992）. Altai 事件 1 審判決の紹介と解説につき，「ウェラン判決の SSO 保護を否定した判決」SLN（ソフトウェア情報センター）

33 号（1992 年），控訴審判決につき，「ウェラン判決の SSO 保護を否定した地裁判決を支持」SLN38 号（1992 年）。また，白鳥・前掲注（17）114 頁も参照。

30） このプログラムの主な機能は，コンピュータが様々なタスクをいつ処理すべきかを定めたスケジュールを作成し，その後，そのスケジュールを実行するようにコンピュータを管理することである（*Altai,* 982 F.2d at 698）。

31） Computer Associates Int'l., Inc. v. Altai, Inc., 775 F.Supp. 544（E.D.N.Y. 1991）.

32） 控訴審は著作権侵害に関する地裁の判断をほぼ踏襲し，その判決に誤りはないとして支持している。それゆえ，以下で述べる「控訴審判決の考え方」のかなりの部分は，地裁判決の考え方でもある。

33） *Altai,* 982 F.2d at 704-705（citing *Baker,* 101 U.S. at 104）.

34） *Altai,* 982 F.2d at 705（citing 3 Melville B. Nimmer & David Nimmer, NIMMER ON COPYRIGHT § 13.03（F）, at 1362.34（1991））.

35） Altai 事件控訴審判決が提唱した AFC テストに関する詳しい紹介として，山本・前掲注（20）202 頁以下。

36） Nichols v. Universal Pictures Co., 45 F.2d 119（2d Cir. 1930）.

37） 抽象化の詳しい説明として，カージャラ＝椙山・前掲注（8）232 頁以下。

38） *Altai,* 982 F.2d at 707.

39） *Id.*

40） *Id.* at 710.

41） Lemley = Samuelson, *supra* note 2 at 11.

42） *Altai,* 982 F.2d. at 709-710（citing 3 Nimmer, *supra* note *34*, § 13.03［F］［3］, at 13-65-71）.

43） *Altai,* 982 F.2d. at 714-715（citing *Altai,* 775 F.Supp. at 562）.

44） *Altai,* 982 F.2d at 715（citing *Altai,* 775 F.Supp. at 562）.

45） 控訴審判決におけるこの部分の理解と関連して，Lemley = Samuelson, *supra* note 2 at 14 は，「互換性は著作権で保護できないプログラミングの現実とビジネス上の必要性である，と判決は判断した」との理解を示している。

46） Lotus Development Corp. v. Borland Int'l., Inc., 49 F.3d 807（1st Cir. 1995）.

47） Lotus Development Corp. v. Borland Int'l., Inc., 831 F.Supp. 223（D.Mss. 1993）.

48） 本件の 1 審についての紹介として「マクロ変換のためのメニュー構造の利用であっても著作権侵害―ロータス対ボーランド事件」SLN54 号（1994 年），控訴審判決の紹介として「ロータス 1-2-3 のメニュー・コマンド階層は，著作権で保護されない―ロータス対ボーランド事件，第一巡回区控訴裁判所判決」SLN62 号（1995 年）および白鳥・前掲注（17）118 頁。また，有用性や機能性の観点に着目して本件を分析するものとして，島並良「ロータス対ボーランド事件が残したもの―機能的表現の著作物性について」知財研フォーラム 34 号 36 頁（1998 年）。

49） キー・リーダーは Borland 社の Quattro Pro に追加された機能であり，そのファイルは Lotus1-2-3 のメニュー・ツリー構造と事実上同一である。キー・リーダーをオンにすると，Borland 社のプログラムが Lotus1-2-3 のマクロを理解して実行可能となる。キー・リーダーをオンの状態で使用すると，マクロ（Lotus1-2-3 マクロの開始キー）にある「/」（スラッシュ）キーを検出した場合を除き，表示，対話およびマクロ実行に関して Quattro Pro のメニューを使用する。この場合，Borland 社のプログラムは Lotus1-2-3 用に書かれたものとしてマクロを解釈する。したがって，Lotus1-2-3 の操作時間を短縮するためにマクロを書いたり購入したりするユーザーも，Borland 社のプログラムにおいてそれらのマクロを引き続き利用できる（*Lotus,* 49 F.3d at 811-812）。

50） *Id.* at 814.

51） *Id.* at 815.

52） 米国著作権法 102 条（b）が，「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」と定めていることについては既に述べた。

53） *Lotus,* 49 F.3d at 815.

54） *Id.* at 817–818.

55） *Id.* at 818.

56） Sega Enterprises Ltd. v. Accolade, Inc., 977 F.2d 1510（9th Cir. 1992）. 本判決の紹介として，白鳥・前掲注（17）229 頁以下，「リバース・エンジニアリングがフェア・ユースにあたるとした判例―第 9 巡回区連邦高裁がアコレードを逆転勝訴に」SLN 42 号（1992 年），山本隆司＝奥邨弘司『フェア・ユースの考え方』（太田出版，2010 年）200 頁以下。

57） Sony Computer Entertainment Inc., v. Connectix Corp., 203 F.3d 596（9th Cir. 2000. 本件を紹介するものとして，「エミュレータ作成目的のリバース・エンジニアリングをフェアユースと判断―PS エミュレータ事件で第 9 巡回区連邦控訴裁判所はソニーを逆転敗訴に」SLN83 号（2000 年），山本＝奥邨・同上 205 頁以下。

58） Bateman v. Mnemonics, Inc., 79 F.3d 1532（11th Cir. 1996）.

59） Lexmark International, Inc. v. Static Control Components, Inc., 387 F.3d 522（6th Cir. 2004）. 本件の紹介として，赤尾太郎「互換性に由来する制約から著作権により保護可能性を否定―レックスマーク事件控訴審判決」SLN102 号（2005 年）。また，本件を素材の 1 つにして，著作権によるアフターマーケット（本件ではプリンタのカートリッジ市場）の支配が許されないことを明らかにするものとして，泉克幸「アクセスコントロール技術による著作物の保護とアフターマーケット」根岸哲＝川濱昇＝泉水文雄編『ネットワーク市場における技術と競争のインターフェイス』（有斐閣，2007 年）289 頁。

60） Mitel, Inc. v. Iqtel, Inc., 124 F.3d 1366（10th Cir. 1997）.

61） Lemley = Samuelson, *supra* note 2, at 15–23. 同論文はこれらの判例を Altai 事件の「子孫（progeny）」と称している。

62） *Accolade,* 977 F.2d at 1525. なお，本件で AFC テストが用いられなかったのは，Accolade がコピーしたのは API ではなく，Sega の全てのコードであったからであると Lemley = Samuelson, *supra* note, at 18 は述べる。

63） 山本・前掲注（20）49 頁。

64） 白鳥・前掲注（17）101 頁。

65） *Altai,* 982 F.2d at 709. ただし，Lemley = Samuelson, *supra* note 2, at 14 は，「外的要因によって決定される要素」の「決定される（dictated）」の語句は，ありふれた場面の法理ではなくマージ理論での用法であると指摘する。

66） Lemley = Samuelson, *supra* note 2, at 26.

67） 以下の本文で述べる技術的な説明については，主として，立木秀樹＝有賀妙子『すべての人のための Java プログラミング〔第 3 版〕』（共立出版，2020 年）を参照した。

68） ライブラリとは，他のプログラムに特定の機能を提供するために作られたプログラムの商品群のこと（同上 2 頁・注 7）。

69） Oracle America, Inc. v. Google Inc., 872 F.Supp. 974, 977（N.D.Cal. 2012）（以下，「第一次地裁判決」ともいう）。

70） *Google,* 141 S.Ct. at 1191（citing *Oracle,* 750 F.3d at 1349）. なお，API 一般については次のような説明がある。「ライブラリ（命令や関数）の集合体，またはそれらを呼び出すための規約集のこと。ソフトウェアをモジュール化し，ライブラリを構成して，API によって呼び出す，という発想が，一度開発したソフトウェアの再利用を容易にし，生産性が大幅に向上することになった」（影島広泰編著『法律家・法務担当者のための IT 技術用語辞典〔第 2 版〕』（商事法務，2021 年）115 頁）。

71） *Google,* 141 S.Ct. at 1210.

72）実際の実行コードは何百行にも及ぶ可能性があるため，複雑なプログラムの作成には予め書かれたタスク実行プログラムの利用なくしては不可能であろうと，最高裁は指摘する（*Google,* 141 S.Ct. at 1191）。

73）したがって，Java 言語に慣れたプログラマは多くのメソッド・コールを知っていることになる（*id.* at 1192）。

74）*Id.*

75）*Id.*

76）*Id.*

77）一般に公開されている Java API の仕様書（サマリーの「メソッドと説明」）では，

「max（int a, int b）

2 つの int 値のうち大きいほうを返します。」

と説明されている（「int」は用いられるデータの種類（「型」と呼ばれる）が，普通の整数であることを示す）。なお，Java API の日本語の仕様書は，以下の Java SE（Java Platform Standard Edition）のドキュメントサイトから利用できる。

〈https://docs.oracle.com/javase/jp/8/docs/api/〉

78）クラスメソッドとはメソッドの種類の 1 つである。本判決の理解とは直接的には無関係なので，より詳しい説明は省く。

79）クラスの 1 つで，指数関数や平方根，三角関数といった数学的な計算を行う各種のクラスメソッドを提供する。

80）java.lang パッケージは，Java の基本的な機能を提供するクラス群である。

81）参考図の「Declaring Code」における「public class Math」の「public」とは，アクセス修飾子と呼ばれるものの 1 種であり，当該クラスにアクセスできる場所が限定されないという意味を示す。

82）*Google,* 141 S.C.t. at 1193.

83）*Id.*

84）ここでいうプラットフォームとは，新しくプログラムやアプリケーションを開発するためのインフラストラクチャーあるいは環境を指す。本最高裁判決は，自動車労働者やデザイナー，製造業者が集まる工場フロアに例えている（*id.* at 1190）。

85）Java プラットフォームのうち，通常のプログラミング用に，一般的な機能がライブラリで提供されているもの。

86）本最高裁判決によれば，当時，600 万人のプログラマが Java 言語の学習と使用にかなりの時間を費やしていた（*Google,* 141 S.Ct. at 1190）。

87）既に述べた「一度書けば，どこでも走る」というスローガンの意味するところである。

88）その理由は，Google は Android プラットフォーム上で書かれたプログラムの全てについて相互運用性を求めることはせず，制限を非常に少なくしたかったのに対し，Sun は相互運用性が基本ポリシーであったからであると判決は述べる（*Google,* 141 S.Ct. at 1191）。

89）特許権侵害の成否についても陪審員に委ねられたが，陪審員は侵害を否定した。Oracle はこれに対して不服を申し立てなかったため，その後，特許権侵害は争点となっていない。

90）それらの中には，本稿で紹介した Accolade 事件や Connectix 事件（以上，第 9 巡回区），Whelan 事件（第 3 巡回区），Altai 事件（第 2 巡回区），Lotus 事件（第 1 巡回区）が含まれている。

91）*Oracle,* 872 F.Supp. 2d at 1000.

92）*Id.*

93）1 審で特許権侵害の争点が含まれていたため，控訴審はカリフォルニア地区連邦地裁を管轄する第 9 巡回区連邦控訴裁判所ではなく，CAFC が受け持つこととなった。ただし，準拠する判例法は第 9 巡回区のものとなっている。

94）第一次 CAFC 判決・前掲注（12）。この判決の解説として，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」SLN138号（2014年）。

95）Googleはサーシオレイライ（certiorari；裁量上訴）を申し立てたが，最高裁は却下した（Google Inc. v. Oracle Am., Inc., 576 U.S. 1071（2015））。

96）4要素も含め，フェア・ユースについては後述（「第Ⅱ章1.」）する。

97）Oracle America, Inc. v. Google Inc., No. C 10-05361 WHA, 2016 WL 3181206（N.D.Cal. June 8, 2016）（「第二次地裁判決」）。

98）Oracle America, Inc. v. Google LLC, 886 F.3d 1179（Fed. Cir. 2018）（以下，「第二次CAFC判決」ともいう）。本判決の解説として，椙山敬士「ORACLE, 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」SLN159号（2018年）。

99）なお，サーシオレイライには，米民訴法50条に規定されている陪審評決破棄動議（Motion for Judgement as a Matter of Law：「JMOL動議」と呼ばれることがある）に関する手続き上の論点が含まれているが（詳しくは，石新智規「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」SLN165号（2020年）2頁），本稿では指摘にとどめる。

100）Campbell v. Acuff-Rose Music, Inc., 510 U.S. 569（1994）. 本件を解説するものとして，白鳥・前掲注（17）219頁，山本＝奥邨・前掲注（56）126頁など。

101）フェア・ユースを議論する文献は数多い。2010年頃までの判決例を利用の種類ごとに分類した上で，紹介・分析するものとして山本＝奥邨・前掲注（56），平成24年に行われたわが国の権利制限を主たる内容とする法改正との関係で論じるものとして，奥邨弘司「〔講演録〕フェア・ユース再考―平成24年改正を理解するために」コピライト629号（2013年）2頁など。

102）「transformative」は「変形的」，「変容的」などと訳されることもあるが，その意味するところを的確に表しているとは言えない。本稿では，そのまま「トランスフォーマティブ」としている。

103）*Campbell,* 510 U.S. at 579.

104）*Id.*

105）*Google,* 141 S.Ct. at 1197.

106）*Id.* at 1201-1202.

107）*Id.* at 1202-1204.

108）米合衆国憲法第1編第8条第8項は，「連邦議会は，著作者と発明者に対してそれぞれの著作と発明について…科学と有用な技芸の発展を促進させる権限を有する」（U.S. Const., Art I, §8, cl.8）と定める。

109）Feist事件最判（Feist Publication, Inc. v. Rual Telephon Service Co., 499 U.S. 340）の次の説示を引く。「著作権の主たる目的は著作者の労働に報酬を与えることではなく科学と有益な技芸（Useful Arts）の進歩を促進することにある」（*id.* at 349-350）。

110）具体例として，動画を表示させるためのアプリを，あるデバイスで実行させるケースを挙げる。

111）APIのラベルが変更されると，ソフトウェアがもはや動作を続けないことになるか，それとも，開発者が全く新たな言語を学ぶか，のいずれかになってしまうと述べる。

112）*Google,* 141 S.Ct. at 1204-1206.

113）*Oracle,* 886 F.3d at 1206.

114）*Google,* 141 S.Ct. at 1206-1208.

115）MCA, INC. v. Wilson, 677 F.2d 180（2d Cir. 1981）.

116）4 Nimmer, *supra* note 34, § 13.05［A］［4］（F）.

117）原審において，CAFCは「SunがGoogleとのライセンス契約を求めていた」ことを認定している（*Oracle,* 886 F.3d at 1209）。

118）この点に関して最高裁は，「実現しなかったライセンスの機会を考慮することで持ち出される『循環論法の危険』に陥らないよう注意すべきである。というのは，もし，潜在的市場の意味を，まさに争われている使用のライセンスに関する理論的市場と定義してしまうと，原告が潜在的市場

での損失を被ることが，いずれのフェア・ユースのケースにおいても当然認められてしまうことになるからである」との学説（4 Nimmer, *supra* note 34, § 13.05［A］［4］）を引用する。

119）いわゆる，「使い慣れ」のことを指摘していると思われる。

120）Campbell 最判において，「著作権法の下で認められる損害の性質を見極める必要性」を議論している（*Campbell,* 510 U.S. at 591–592）ことを引用している。

121）Connectix 事件を引用している（*Connectix,* 203 F.3d at 607）。また判決は，「他者が競争できないようにして市場を独占する試みは，創作的表現の促進という制定法上の目的に反する」との理解を示した Sega 事件（*Sega,* 977 F.2d at 1523–1524），「後続のユーザーが機能を促進するためにコンピュータ・プログラムをコピーした場合，著作物としてのプログラムの商業的価値を利用しなかった」と指摘する Lexmark 事件（*Lexmark,* 387 F.3d at 544）を紹介している。

122）*Google*, 141 S.Ct. at 1208.

123）差戻しを受けた CAFC は，2021 年 5 月 14 日，フェア・ユースに関する第二次 CAFC 判決を自ら取り消し，Google を支持した第二次地裁判決を維持する判決を行った（Oracle America, Inc. v. Google LLC, 847 F.Appx 931（Fed. Cir. May 14, 2021）（mem））。

124）最高裁は Java API を宣言コードと実行コードとに区別して理解しているが，後者については，Google は独自に作成しており，コピーしたのは宣言コードの部分であった。宣言コードにおける個別のタスク（メソッド）をクラスとパッケージによって階層化した構造の著作物性が第一義的な問題であるが，階層構造の著作物性あるいはインターフェイスの著作物性という点では Lotus 事件との共通性を見ることができる。Lotus 事件では，第 1 巡回区連邦控訴裁判所が，メニュー・コマンドの階層構造は「操作方法」（米著作 102 条（b））に当たるとして著作物性を否定した。同判決はサーシオレイライが認められたが，最高裁判事の意見は 4 対 4（Stevens 判事は回避）の同数となり，控訴裁判所の判決が支持されるにとどまった（116 S.Ct. 804（1996））。こうしたことからも，この種の問題に対する初の最高裁の判断が待たれていた。

125）本最高裁判決は第 2 要素から検討を始めている。107 条における 4 要素の検討順序に特に決まりがあるわけではない。ちなみに，プログラムの利用行為（いずれもがリバース・エンジニアリングのケースであるが），Accolade 事件では第 1，第 4，第 2，第 3 の順で，Connectix 事件では第 2，第 3，第 1，第 4 の順で各要素の検討が行われている。

126）Perfect 10, Inc. v. Amzon.com, Inc., 487 F.3d 701（9th Cir. 2007）.

127）奥邨・前掲注（2）28 頁も，「今後，API に限らず，各種インターフェイス，プロトコル，言語…さらにはより広い分野における再実装目的の利用が，フェアユースと評価される余地が生まれた」と評する。

128）本件最高裁判決の注目点の 1 つとして奥邨・前掲注（2）28 頁も，「再実装を広く認めることによって，アプリ・プログラマが Java 言語習得のために行った投資を人質にするような囲い込みを排した点」を指摘する。

129）たとえば，ファイル交換ソフトによる音楽ビジネスが問題となった Napster 事件（A & M Records, Inc. v. Napster, Inc., 239 F.3d 1004（9th Cir. 2001））では，音楽の CD 市場（現在の市場）とダウンロード市場（将来の市場）について第 4 要素が検討されている。

130）たとえば，カージャラ＝椙山・前掲注（8）238 頁。この論点につき，特にリバース・エンジニアリングに焦点を当て，米国と EU の状況を分析検討した論稿として，Ull-Maija Mylly, *An Evolutionary Economics Perspective on Computer Program Interoperability and Copyright,* 41 International Review of Intellectual and Competition Law 284（2010）がある（邦訳として，青柳由香訳「コンピュータプログラムの互換性と著作権に関する進化経済学的視点（1）（2・完）」知的財産法政策学研究 29 号 93 頁・30 号 71 頁（2010 年））。なお，同論文は競争法による規律に消極的であり，積極的に活用すべきと説く本稿の立場とは異なる。

131）Brief of 72 Intellectual Property Scholars as Amici Curiae, *supra* note 11, at 3.

132）Brief of Professors Peter S. Menell, David Nimmer, and Shyamkrishna Balganesh as Amici

Curiae in Support of Petitioner, at 3.

133) Brief of Amicus Curiae Professor and Former CONTU Member Arthur R. Miller in Support of Respondent, at 2.

134) 実際，Thomas 判事と Alito 判事の反対意見は，Java API の宣言コードについても著作物性を肯定している。また，石新・前掲注（99）は，2020 年 10 月 7 日に行われた口頭弁論の様子を明らかにするが，著作物性の有無を含め，8 人の最高裁判事の中でも意見が分かれていることが読み取れる。

135) この論点に関し，水越・前掲注（2）53-54 頁は「著作物性を認める範囲の基準を定め，最も適切な影響範囲で線引きすることが実際に困難だったのではないだろうか」「多数意見を構成する裁判官で合意することは困難であった」「フェアユースで多数意見をまとめることの方が，より広い支持に基づいた判決になるとの判断が考えられる」などと述べる。

136) Lemley = Samuelson, *supra* note 2, at 35 n.211 は，IBM のアミカスを引き合いに，著作物性を肯定した CAFC の判決を破棄するようソフトウェア業界は強く求めており，最高裁の示した理由は根拠が薄弱であると批判する。

137) 特に，決め手となるトランスフォーマティブの評価をプログラムの素人である陪審員が合理的に行うことはハードルが高いであろう。

138) Lemley = Samuelson, *supra* note 2, at 42-43 も，API の再実装を巡って紛争が繰り返される可能性，また，コマンド構造や入出力フォーマットといった API 以外のインターフェイスに対する判断の予測困難性，さらには，フェア・ユースは抗弁として被告側が立証義務を負うという問題（著作物性は原告が立証義務を負う）などを指摘し，最高裁のフェア・ユースによる問題解決の不十分さを指摘する。

139) Lemley=Samuelson, *supra* note 2, at 40.

140) *Id.*

141) *Id.*

142) *Id.* at 49-51.

143) ここでは，Java API のような個別のものではなく，API 一般を前提としている。このような API は，「OS やミドルウェアとその上で稼働するアプリケーションとのインターフェイス規定のこと。汎用性の高い API を利用すると，機種互換性，アプリケーション互換性の実現が容易になります」（秀和システム編集本部編著『最新　基本パソコン用語事典〔第 5 版〕』（秀和システム，2020 年）343 頁）と，説明されている。前掲注 70）における API の説明も参照。

144) もちろん，実行コードのデッドコピーのような利用まで，無条件で許容すべきという趣旨ではない。

145) 加戸守行『著作権法逐条講義〔六訂新版〕』（著作権情報センター，2013 年）129-130 頁。

146) 中山・前掲注（13）42-44 頁。

147) これに対し，水越・前掲注（2）55 頁は，「本件 Java API 全体としてみれば通信インターフェイス等特別な約束に関連するものではないから，本件 Java API は本号に該当しない」と述べる。なお，影島・前掲注 70）116 頁は，「API は，著作権法上，著作物とはならない。著作権法 10 条 3 項 2 号の『規約』…にあたるからである（もちろん，API が呼び出すプログラムそのものは著作物となり得る）」との理解を示す。

148) 中山信弘『著作権法〔第 3 版〕』（有斐閣，2020 年）79-80 頁。なお，文中の「判例」の具体的事例の 1 つとして，プログラムに関する東京高決平成元年 6 月 20 日判時 1322 号 138 頁〔システムサイエンス第 1 事件〕が挙げられている。

149) 中山・前掲（13）46 頁。

150) 文化庁著作権課「デジタル・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（著作権法第 30 条の 4，47 条の 4 及び第 47 条の 5 関係）」（2019 年 10 月 24 日）40 頁。

151) 同上。

152）奥邨・前掲注（2）29頁・注34。同論稿は，代わりに引用規定（32条）の適用可能性を論ずる（同上29–30頁）。

153）なお，奥邨・前掲注（101）23–24頁は，少なくとも互換機の開発を目的とするリバース・エンジニアリングに対して30条の4を適用することについて，慎重な意見が述べられている。

●Summary

This article discusses an appropriate range of computer program copyright protection from the perspective of interoperabiliity. In 2021, the U.S. Supreme Court held that Google's copying of Java API (application programming interface) for reimplementation was fair use. Part 1 of this article clarifies how the software interface that realizes interoperability has been judged in American case law, what API is, and the procedural history of this case. Part 2 details the Supreme Court's decision on fair use. Part 3 analyzes the decision and clarifies the importance of interoperability. The author concludes that allowing the use of APIs that achieve interoperability is also necessary in Japan, and he proposes specific methods for doing this.