No. 18-956

IN THE

Supreme Court of the United States

GOOGLE LLC,

Petitioner,

v.

ORACLE AMERICA, INC.,

Respondent.

On Petition for a Writ of Certiorari
to the United States Court of Appeals
for the Federal Circuit

MOTION FOR LEAVE TO FILE BRIEF OF 78 AMICI CURIAE AND BRIEF OF 78 AMICI CURIAE COMPUTER SCIENTISTS IN SUPPORT OF PETITIONER

Phillip R. Malone
Counsel of Record
JUELSGAARD INTELLECTUAL
PROPERTY AND
INNOVATION CLINIC
MILLS LEGAL CLINIC AT
STANFORD LAW SCHOOL
559 Nathan Abbott Way
Stanford, CA 94305
(650) 725-6369
pmalone@law.stanford.edu

Counsel for Amici Curiae

MOTION FOR LEAVE TO FILE BRIEF OF AMICI CURIAE COMPUTER SCIENTISTS

Amici are 78 computer scientists, engineers, and professors who are pioneering and influential figures in the computer industry.[1] Amici respectfully move, pursuant to Supreme Court Rule 37.2, for leave to file the attached brief of amici curiae in support of the petition for a writ of certiorari. Based on their extensive knowledge of and experience with software development and the software industry, amici are uniquely positioned to bring to the Court's attention important information that will not otherwise be provided by the parties or other amici. Amici's expert historical, technical, and industry knowledge may be of considerable help to the Court in this case.

Amici's experience is exceptionally broad and deep. Amici include the architects of iconic computers from the mainframe era to the microcomputer era, the most widely used programming languages, and operating systems such as MS-DOS and Unix. They are responsible for key advances in the field, such as computer graphics, cloud computing, public key cryptography, object-oriented programming, virtual reality, and the Internet itself. They wrote the standard college textbooks in areas including artificial intelligence, algorithms, computer architecture, computer graphics, computer security, data structures, functional programming, Java programming, operating systems, software engineering, and the theory of programming languages.

As some of the world's foremost experts in their field, amici are well suited to provide

[1] Amici's biographies are attached as Appendix A to the proposed brief.

the Court with an authoritative explanation of computer interfaces, particularly the Java Application Programming Interface (API) at issue in this case; of the Federal Circuit's fundamental misunderstanding of how interfaces differ from programs; and of the exceptional importance of this case. Moreover, as computer scientists, amici have themselves relied on reimplementing software interfaces to create and operate new software. Amici bring an unparalleled perspective to assist the Court in understanding this case's potential to upset the long-standing practices and expectations of the software industry and the innovation that has flourished under those practices and expectations.

Many of the computer scientists listed on the proposed brief were amici in four previous briefs in the course of this litigation, including two briefs in support of Petitioner on the merits at the Federal Circuit, one brief in support of Petitioner's petition for rehearing en banc at the Federal Circuit, and one brief in support of Petitioner's first petition for a writ of certiorari to this Court.

Pursuant to Rule 37.2(a), amici provided notice of intent to file the attached brief at least ten days before its due date. Petitioner provided blanket consent for the filing of amicus briefs. Respondent declined to consent.[2] Because amici represent a cross section of the world's most distinguished computer scientists and engineers, among the 78 amici are a small number (five) who are current employees of Petitioner

[2] Respondent indicated it would not consent if the brief included any amici who were current Google employees, independent contractors, or consultants, or who had served as witnesses or retained experts in this case.

(indicated by * next to their names in Appendix A); two who receive some support from Google (indicated by **); two who testified as unpaid fact witnesses at trial in this case (indicated by †); and one who was retained as an expert by Petitioner but did not testify (indicated by ‡).

Each of those amici signs this brief based on their personal experience and beliefs as individual computer scientists whose work in the field long preceded their affiliation with Petitioner or their participation in this case. They do not sign on behalf of Petitioner or at its request. Their interest in this case, like that of all of the other amici, is solely to help explain to the Court the background, nature, and history of APIs and the critical importance of software interface reimplementation for continued innovation across the computer industry.

Amici thus respectfully request leave to file the attached amici curiae brief to aid the Court in understanding the technical details of this case and its exceptional importance.

Respectfully submitted,

Phillip R. Malone
Counsel of Record
JUELSGAARD INTELLECTUAL PROPERTY AND INNOVATION CLINIC
MILLS LEGAL CLINIC AT STANFORD LAW SCHOOL
559 Nathan Abbott Way
Stanford, CA 94305
(650) 725-6369
pmalone@law.stanford.edu

February 25, 2019

TABLE OF CONTENTS

TABLE OF AUTHORITIES.................................. ii

INTEREST OF AMICI CURIAE............................ 1

SUMMARY OF ARGUMENT................................ 2

ARGUMENT... 4

I. The Decisions Below Reflect the Federal Circuit's Fundamental Misunderstanding of How Interfaces Differ from Programs............. 4

A. Software Interfaces Specify What a Program Does, Not How It Does So.......... 5

B. Google Reimplemented the Java API to Promote Interoperability and Transformed Java to Run on Smartphones....................12

II. This Case Is Exceptionally Important Because the Decisions Below Upend Decades of Settled Expectations and Threaten Software Innovation..16

A. The Computer Industry Has Long Relied on Freely Reimplementing Software Interfaces to Foster Innovation and Competition...17

B. Restricting the Reimplementation of Software Interfaces Will Stifle Competition by Increasing Entry Barriers....................22

C. Restricting the Reimplementation of Software Interfaces Will Exacerbate Lock-In Effects and Create an "Orphan Software" Problem...................................24

CONCLUSION...25

APPENDIX A — LIST OF AMICI CURIAE...........1a

TABLE OF AUTHORITIES

Cases **Page(s)**

Oracle Am., Inc. v. Google Inc. (Copyright I), 872 F. Supp. 2d 974 (N.D. Cal. 2012) ... *passim*

Oracle Am., Inc. v. Google Inc. (Copyright II), 750 F.3d 1339 (Fed. Cir. 2014) 4, 13, 14, 15

Oracle Am., Inc. v. Google Inc. (Fair Use I), 2016 WL 3181206 (N.D. Cal. 2016) 14, 16

Oracle Am., Inc. v. Google LLC (Fair Use II), 886 F.3d 1179 (Fed. Cir. 2018) 5, 13, 15

Constitutions

U.S. Const. art. I, § 8, cl. 8 25

Statutes

17 U.S.C. § 101 6

17 U.S.C. § 102(b) 5

Other Authorities

Cloud Storage Interoperability, Google Cloud (last updated Oct. 23, 2018), https://cloud.google.com/storage/docs/interoperability 21

TIOBE Index for December 2018, TIOBE (last visited Dec. 5, 2018), https://www.tiobe.com/tiobe-index 9

Trail: Essential Classes (The Java^{TM} Tutorials), Oracle (last visited Dec. 10, 2018), https://docs.oracle.com/javase/tutorial/essential/index.html 16

Usage Statistics and Market Share of Unix for Websites, W3Techs (Nov. 30, 2018), https://w3techs.com/technologies/details/os-unix/all/all 19

Dave Burke, *Android: Celebrating a Big Milestone Together with You*, Google (May 17, 2017), https://blog.google/products/android/2bn-milestone 15, 19

Fred von Lohmann, *The New Wave: Copyright and Software Interfaces in the Wake of* Oracle v. Google, 31 Harv. J.L. & Tech. 517 (2018) 23

Ira W. Cotton & Frank S. Greatorex, Jr., Data Structures and Techniques for Remote Computer Graphics, Am. Fed'n Info. Processing Soc'ys Fall Joint Computer Conf. 533 (1968) 17

Jay Greene & Laura Stevens, *"You're Stupid If You Don't Get Scared": When Amazon Goes from Partner to Rival*, Wall St. J. (June 1, 2018, 5:30 AM ET), https://www.wsj.com/articles/how-amazon-wins-1527845402 23

Maurice V. Wilkes, David J. Wheeler & Stanley Gill, *The Preparation of Programs for an Electronic Digital Computer* (1951) 17

Rita Zhang, *Access Azure Blob Storage from Your Apps Using S3 Java API*, Microsoft (May 22, 2016), https://www.microsoft.com/developerblog/2016/05/22/access-azure-blob-storage-from-your-apps-using-s3-api 21

Steven J. Vaughan-Nichols, *Linux Totally Dominates Supercomputers*, ZDNet (Nov. 14, 2017, 12:04 PM PST), http://www.zdnet.com/article/linux-totally-dominates-supercomputers 19

INTEREST OF AMICI CURIAE

Amici are 78 computer scientists, engineers, and professors who are pioneering and influential figures in the computer industry.[1] Amici include the architects of iconic computers from the mainframe era to the microcomputer era, including the IBM S/360 and the Apple II; languages such as AppleScript, AWK, C, C#, C++, Delphi, Go, Haskell, PL/I, Python, RenderMan, Scala, Scheme, Standard ML, Smalltalk, and TypeScript; and operating systems such as MS-DOS and Unix.[2] Amici are responsible for key advances in the field, including in computer graphics, computer animation, computer system architecture, cloud computing, algorithms, public key cryptography, the theory of computation, object-oriented programming,

[1] Pursuant to Rule 37.2(a), counsel for both parties received notice of intent to file this brief at least ten days before its due date. Petitioner consented to the filing; Respondent declined to consent. No counsel for a party authored this brief in whole or in part, and no party or counsel for a party made a monetary contribution intended to fund its preparation or submission. No person other than the amici or their counsel made a monetary contribution to the preparation or submission of this brief.

[2] Amici's biographies are attached as Appendix A. Amici sign this brief on their own behalf and not on behalf of the companies or organizations with which they are affiliated; those affiliations are for identification purposes only. Amici represent a cross section of the world's most distinguished computer scientists and engineers. As such, the 78 amici include five who are presently Google employees (indicated by * next to their names); two who receive some support from Google (indicated by **); two who testified as unpaid fact witnesses at trial in this case (indicated by †); and one who was retained as an expert by Google but did not testify (indicated by ‡). Each of these amici signs this brief based on their personal experience and beliefs as individual computer scientists whose work in the field long preceded their affiliation with Google or their participation in this case. None sign on behalf of Google or at Google's request.

relational databases, design patterns, virtual reality, the spreadsheet, and the Internet. Amici wrote the standard college textbooks in areas including artificial intelligence, algorithms, computer architecture, computer graphics, computer security, data structures, functional programming, Java programming, operating systems, software engineering, and the theory of programming languages.

Amici have been widely recognized for their achievements. They include at least 12 Association for Computing Machinery (ACM) Turing Award winners (computer science's most prestigious award); 24 ACM Fellows; 11 Institute of Electrical and Electronics Engineers (IEEE) Fellows; 14 American Academy of Arts and Sciences (AAAS) Fellows; 6 National Academy of Sciences Members; 24 National Academy of Engineering Members; 5 National Medal of Technology recipients; and numerous professors at many of the world's leading universities.

As computer scientists, amici have relied on reimplementing interfaces to create fundamental software. They join this brief because they believe, based on their extensive experience with and knowledge of computer software and programming, that the decisions below threaten to upend decades of settled expectations across the computer industry and chill continued innovation in the field.

SUMMARY OF ARGUMENT

The Court should grant the petition for a writ of certiorari. Review of the decisions below is exceptionally important because they undermine a fundamental process—software interface reimplementation—that has spurred historic innovation across the software industry for decades.

Software *interfaces*, including those embodied in the Java Application Programming Interface (API) at issue here, are purely functional systems or methods of operating a computer program or platform. They are not computer programs themselves. Interfaces merely describe *what* functional tasks a computer program will perform without specifying *how* it does so. The Java API's functional interfaces, called declarations, are written using the Java programming language, which mandates each declaration's precise form.

In contrast, *implementations* provide the actual step-by-step instructions to perform each task included in an interface. Sun implemented the Java API for desktop computers. Google reimplemented—or wrote its own original implementation of—the Java API when it created the Android platform for smartphones and tablets. Android was highly transformative: It enabled programs written in the Java programming language to successfully run on smartphones and tablets for the first time. Doing so required Google to make significant additions to the Java API to handle mobile-specific features, like touchscreen inputs.

Android also provided interoperability with Java: Programmers could use their preexisting knowledge to simultaneously write Java programs for both desktops and smartphones. Reimplementing the Java API was the only way to make Android interoperable with Java. Reimplementation requires duplicating an interface's declarations and organizational scheme—its structure, sequence, and organization (SSO). Had Android changed the Java API's declarations or SSO, programmers would have been forced to write different software for desktops and smartphones, eliminating one of Android's most significant benefits.

Google's reimplementation of an existing interface was not unusual. Reimplementing software interfaces is a long-standing, ubiquitous practice that has been essential to realizing fundamental advances in computing. It unleashed the personal computer revolution, created popular operating systems and programming languages, and established the foundation upon which the Internet and cloud computing depend. It also increases consumer choice, lowers prices, and fosters compatibility between programs. Free reimplementation of software interfaces was, and continues to be, essential for innovation and competition in software.

The Court should grant the petition for a writ of certiorari to preserve software interfaces as uncopyrightable and prevent copyright from stifling innovation in software.

ARGUMENT

I. The Decisions Below Reflect the Federal Circuit's Fundamental Misunderstanding of How Interfaces Differ from Programs

The decisions below extend copyright protection to software interfaces—including the Java API—by erroneously equating them with computer programs. Asserting that software interfaces are simply a type of computer program, all of which are "by definition functional," the Federal Circuit misapplied general Ninth Circuit law recognizing computer programs as copyrightable. *See Oracle Am., Inc. v. Google Inc.* (Copyright II), 750 F.3d 1339, 1367 (Fed. Cir. 2014). But software interfaces are not computer programs, and no party argues that "one can copy line-for-line someone else's copyrighted computer program." *Oracle Am., Inc. v. Google Inc.* (Copyright I), 872 F. Supp. 2d 974, 987 (N.D. Cal. 2012).

The Federal Circuit's conclusory review fails to appreciate the district court's reasoned—and correct—recognition of software interfaces as uncopyrightable under 17 U.S.C. § 102(b) and the merger doctrine. *See Copyright I*, 872 F. Supp. 2d at 998-1000. The Federal Circuit compounded its error by overturning a jury finding of fair use and holding that Google's creation of Android was not fair use as a matter of law. *See Oracle Am., Inc. v. Google LLC* (Fair Use II), 886 F.3d 1179, 1185-86 (Fed. Cir. 2018).

Amici join Google's arguments that software interfaces cannot be copyrighted under either Section 102(b) or the merger doctrine, and that in any event, Google's creation of Android was fair use. Brief for Petitioner at 16-29, *Google LLC v. Oracle Am., Inc.*, No. 18-956 (Jan. 25, 2019). In support of those arguments, amici emphasize that software interfaces correspond to functional ideas, that Google had to duplicate the Java API's declarations exactly to provide interoperability between Android and Java, and that Android was a transformative achievement that successfully introduced Java to smartphones for the first time.

A. Software Interfaces Specify What a Program Does, Not How It Does So

A software interface specifies the set of commands used to operate a computer program or system. Each command defines one functional task a program must accomplish, such as finding the maximum of two numbers, sorting a list of numbers, or displaying text on the screen.

Each command in an interface includes its name, inputs, and outputs. Together, these comprise the command's "declaration." The declaration for a command to find the maximum of two numbers, for example, would include the name "max," two numbers as inputs, and one number—the maximum—as output. Declarations are purely functional: They specify *what* a computer program or system needs to do without specifying *how* it does so. By themselves, declarations do not instruct a computer to do anything.

In contrast, an interface's *implementation* is the actual "set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result," namely, carrying out the tasks specified by its declarations. 17 U.S.C. § 101 (defining "computer program"). The same declaration can be implemented in various ways to accomplish the same task. Some implementations prioritize speed, others memory use. So long as an implementation carries out the specified task, it is valid. While the "specification is the *idea*," the "implementation is the *expression*." *Copyright I*, 872 F. Supp. 2d at 998 (emphasis in original).

Because real-world software interfaces can include thousands of declarations, programmers group related declarations into their own "folders," just as everyday computer users group related files into folders on their desktop. The courts and parties have referred to this organizational scheme throughout this litigation as the interface's structure, sequence, and organization (SSO).

i. Declarations specify the individual tasks a program must perform

To better understand the relationship between an interface's declarations, implementations, and SSO, consider the `sort` declaration in the Java API.[3] In English, this declaration would read, "Given a list of numbers, sort them in ascending order." To express this functional requirement in terms a computer can understand, a programmer would write the following declaration in the Java language[4]:

```
public static void sort(int[] a)
```

Before explaining each component of this declaration, we emphasize that this line does not instruct the computer to do anything. If a programmer attempted to run this "program," nothing would happen because there are no instructions to run. The line simply indicates that this declaration's implementation will include a command, which Java calls a "method," for sorting numbers.

The Java language requires almost every word in this declaration. A programmer *must* type those words exactly as they appear above, including the same

[3] `Courier` denotes Java keywords and declarations.

[4] The Java language is one part of the Java platform (J2SE), which also includes the API and API implementations (the latter are also called "libraries"). While the boundary between the language and the API is indefinite, the language is generally responsible for defining the syntax and keywords programmers use to write software. Only the API is at issue here. *See Copyright I*, 872 F. Supp. 2d at 978.

capitalization, punctuation, and order. Otherwise, the declaration will cause an error or specify a method with different functionality, like sorting words instead of numbers. The word `public` is a Java language keyword that enables other programs to use `sort` once it has been implemented (other keywords, like `private`, restrict other programs' access to a method). Similarly, the Java language requires `static` for `sort` to work as expected.[5] The `void` keyword means that the method does not have any output; rather than output a sorted copy of the list, `sort` simply rearranges the given list of numbers. Finally, the parentheses enclose the method inputs. Here, the only input is the list of integers to be sorted—designated by the Java keyword `int[]`.

In contrast, only two words in the declaration leave the programmer any choice, and both are names. The first is `sort` itself. This word descriptively names the method based on the task its implementation will perform. While it would be possible to use a synonym—perhaps "arrange" or "order"—for the same method, few names are as intuitive as `sort` to describe the task this method's implementation will perform. Particularly short and intuitive names for common operations like `sort` even become customary terms of art used across interfaces.[6] Because the

[5] The Java language primarily views programs in terms of interactions among "objects" representing the program's data. Related objects are members of the same "class." Adding the `static` keyword to a method declaration allows that method to be called on all objects of a class even if the method could not be added to the class directly, as is the case here. Thus, for `sort` to work across any list of numbers, rather than one particular list, its declaration must include `static`.

[6] As of December 2018, seven of the top ten most used programming languages (Java, Python, C++, Visual Basic .NET,

programmers who design and use a software interface are typically different people, declaration names must be intuitive and succinctly describe their purpose to promote efficiency and minimize errors.

Similarly, `a` names the input "array," or list, of numbers to be sorted. Just as with `sort`, the programmer designing the interface chooses the input's name. Other options could be "array," "numbers," or "list." But just as with `sort`, the universe of potential names is small and further restricted by linguistic convention. While software interface designers have some choice for naming methods and inputs, the method's function, word length, and clarity constrain their choice. Particularly for programming language interfaces, which define the most basic commands used across programs, there are few practical options for naming declarations that satisfy these constraints.

ii. Implementations provide the step-by-step instructions to perform the tasks declarations specify

Once a software interface has been designed, programmers can supply implementations to carry out the tasks specified by its declarations. Google, for example, wrote its own implementations for the Java API's declarations. Implementations take the inputs listed in declarations and manipulate them to produce the correct output. While the syntax of the programming language dictates the form of each declaration, implementations are open-ended and can be thousands of lines long. Naïve

C#, JavaScript, and PHP) include a command called `sort` to arrange a list in ascending order. *See TIOBE Index for December 2018*, TIOBE (last visited Dec. 5, 2018), https://www.tiobe.com/tiobe-index.

implementations can be prohibitively slow or use excessive amounts of memory. In contrast, clever implementations can run quickly enough to make formerly unfeasible operations practical or conserve enough memory to allow programs to run on entirely new hardware—such as phones, tablets, televisions, or even home thermostats—that have far less memory available than desktop computers.

Computer scientists have evaluated dozens of implementations for `sort`. One of the simplest implementations is "selection sort." Given a list of numbers, a selection sort implementation starts at the beginning of the list and walks through number by number, keeping a running tally of the smallest number it has found. Once it reaches the end of the list, it swaps the smallest number with the number at the beginning of the list. Then, the program searches through the remainder of the list a second time, this time looking for the second smallest number to swap into the second position. This process repeats until the program has swapped every number into its correct position. Unfortunately, this implementation is prohibitively slow for large lists of numbers.

More sophisticated implementations for `sort`, like "quicksort" or "mergesort," can sort even large lists efficiently. With modern data sets comprising hundreds of millions or even billions of numbers, names, or images, inefficient sorting implementations like selection sort make entire categories of programs impossible to use. Because different devices have different constraints, software engineers devote considerable effort to choosing the best implementation to meet their specific needs. Their choice could mean the difference between the success of two competing pieces of software.

iii. SSOs establish how software interfaces group related declarations

Because interfaces can include tens of thousands of declarations, their designers organize related declarations in the same way users organize related files into folders on their desktop. In fact, Java's designers organized the Java API's files in exactly this way. *See* Figure 1.

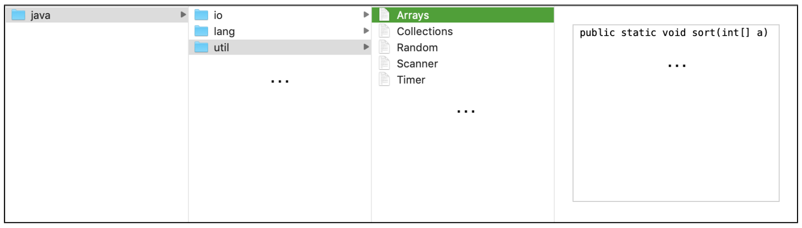

Figure 1

Java's API is organized in three tiers: packages, classes, and methods. Packages correspond to folders, classes to files, and declarations to individual lines in a file. The full file path for `sort`, for example, is `java.util.Arrays.sort`. The overall folder for the interface is named `java`, while `util`, short for utility, is the name of the package, or subfolder, containing the API's various general-purpose classes. One such class, `Arrays`, is a file that contains methods for manipulating lists of objects, like numbers. One of the lines in `Arrays` is the declaration given above for `sort`.

Programmers who reimplement, or provide their own implementation for, an interface must maintain its SSO. Just as users must know how to navigate to their saved documents, programmers using a software interface must specify the path for each declaration they use, like `sort`, so that the computer knows where to find the corresponding implementation. Telling a person to click on "My Documents," then on a folder called "Receipts," and finally on a file called "Sofa" to find how much their sofa cost is just like a program navigating through the Java API to a package called `util` and opening a class called `Arrays` to find the implementation for the `sort` method.

Changing this standard organizational scheme would prevent a person or a program from locating the file or implementation they need, rendering the interface specification incompatible. Thus, while interface designers have some choice in naming their method declarations and inputs, programmers who are reimplementing an existing interface, like Google did with the Java API, must use the same standard names and structure to achieve interoperability.

B. Google Wrote Its Own Implementation of the Java API to Promote Interoperability and Transform Java to Run on Smartphones

Google created the Android platform to promote interoperability and enable Java to run on an entirely new class of devices: smartphones. This required Google to reimplement the Java API: It duplicated the Java API's declarations and SSO but wrote its own implementations. *See Copyright I*, 872 F. Supp. 2d at 978. It would have been impossible for Google to make Android interoperable, or compatible, with Java without reimplementing the Java API.[7] In this context, making software interoperable *means* reimplementing a software interface.

[7] Though we follow convention in using interoperability and compatibility interchangeably, some computer scientists

In both of its opinions, the Federal Circuit questioned Google's claim that Android reimplemented the Java API to promote interoperability with Java because programs written for Android are not fully compatible with Java. *Fair Use II*, 886 F.3d at 1206 n.11 (finding evidence "unrebutted" that "Google designed Android to be incompatible with the Java platform"); *see also Copyright II*, 750 F.3d at 1371 (finding "Google's interoperability argument confusing"). But complete compatibility is not necessary, or even desirable, to promote interoperability in software development.

Because of its longevity, Java, and almost every other computer system, must remain backwards-compatible. Any program written in earlier versions of Java must also run on later versions, or programmers would be unable to make cumulative improvements and the software ecosystem would break down. However, this also means that inefficient or outdated software survives several generations of software development solely to maintain compatibility.

To avoid this problem, Google selectively reimplemented portions of the Java API for Android to eliminate functionality that was obsolete or inappropriate for smartphones, like using a mouse. *See Copyright I*, 872 F. Supp. 2d at 978. Rather than copy

---

distinguish compatibility as one-way functionality and interoperability as two-way functionality. For example, under this distinction, new versions of Microsoft Word are compatible with older versions because users can save documents in Word 2003 and open them in Word 2017. New versions of Word are not interoperable, however, because the converse is not true. In contrast, Oracle's requirement that companies obtain a Java Compatibility Kit (JCK) license to demonstrate "compatibility" is merely a licensing scheme, not a technical necessity.

Sun's implementations, Google was careful to write its own implementations to carry out the tasks the Java API's declarations specify. Google's decision empowered software developers to write Java programs that run equally well on both desktops and smartphones. *See Oracle Am., Inc. v. Google Inc.* (Fair Use I), 2016 WL 3181206, at *10 (N.D. Cal. 2016).

Android was highly transformative. Amici agree with the district court that "Sun and Oracle never successfully developed its own smartphone platform using Java technology." *Copyright I*, 872 F. Supp. 2d at 978.[8] Creating Android required Google to significantly expand Java's API in novel ways to account for external features and constraints unique to the smartphone context: built-in GPS tracking, limited battery life and memory, fluctuating network connections, and an entirely new user interface based on touchscreen gestures. *See Fair Use I*, 2016 WL 3181206, at *9.

These significant augmentations to Java's API introduced Java to an entirely new Android platform that, with over two billion monthly active devices, now has the "largest reach of any computing platform of its kind." Dave Burke, *Android: Celebrating a Big Milestone Together with You*, Google (May

[8] While Sun did release Java ME to run Java on feature phones, these devices are far less sophisticated than modern smartphones. Moreover, Java ME did not support the entire Java language, omitting basic features like numbers with decimal points (known as floating point arithmetic). Finally, Java ME did not support key Java API features like the Java Collections Framework, which is part of `java.util`, a package necessary "to make any worthwhile use of the [Java] language." *Copyright II*, 750 F.3d at 1349. Thus, Java ME was far *less* compatible with standard Java than Android, and Java ME's failure to include such core functionality only underscores how transformative Android was.

17, 2017), https://blog.google/products/android/2bn-milestone. Programmers using only the reimplemented packages can write programs for desktops and smartphones using the same familiar instructions. Additionally, because Java and Android are both open source (meaning anyone can read their implementations), Google's focus on interoperability has enabled outside programmers, including many amici, to contribute improvements to both platforms simultaneously. Contrary to the Federal Circuit's assertion that there was no evidence of programs that rely only on Google's reimplemented packages, or that "[no] such program would be useful," *Copyright II*, 750 F.3d at 1371 n.15, Java and Android form parts of a broad and largely compatible ecosystem that drastically simplifies writing software for desktops and smartphones.

Android revitalized this ecosystem, inspiring renewed innovation and collaboration among programmers. Sun's CEO publicly congratulated Google upon Android's release on his official company blog and expressed support for Android. *See* Brief of Defendant-Appellee/Cross-Appellant Google Inc. at 17-18, *Fair Use II*, 886 F.3d 1179 (Docket No. 17-1118), 2017 WL 2305681. Sun's CEO also emailed Google's CEO directly to offer his congratulations on Android's success and to suggest further improvements. *See id.* at 18-19. After acquiring Sun, even Oracle initially praised Google for expanding Java to new devices. *See id.* at 19.

Sun had always promoted the Java API, along with the Java language, as free and open for all to use. *See id.* at 9-10. Many amici, along with instructors at high schools and colleges across the country, decided to teach Java in introductory programming courses precisely because of its free availability. Assertions that the Java API might be copyrightable only emerged after Oracle acquired Sun in 2010. While Oracle does not dispute that the Java language is free and open for all to use, it asserts a copyright interest in the Java API. *Copyright I*, 872 F. Supp. 2d at 978. Even then, Oracle concedes that at least sixty-two classes, spread across three Java API packages, are necessary for the Java language to work. *Fair Use I*, 2016 WL 3181206, at *5.

As professors, textbook authors, and industry leaders, amici have broad experience with both teaching and using the Java language and do not consider it to be fully separable from the Java API. Amici agree with the district court that "there is no bright line" between them. *Copyright I*, 872 F. Supp. 2d at 982. The API is part of what makes the Java language, Java. Indeed, Oracle's own online tutorials consider portions of the Java API—including packages like `java.util.regex` that it accuses Google of infringing—"essential to most programmers" for programming in Java. *Trail: Essential Classes (The Java^{TM} Tutorials)*, Oracle (last visited Dec. 10, 2018), https://docs.oracle.com/javase/tutorial/essential/index.html.

II. This Case Is Exceptionally Important Because the Decisions Below Upend Decades of Settled Expectations and Threaten Software Innovation

Software interfaces are essential to innovation. For decades, programmers have relied upon reimplementing interfaces to create fundamentally transformative technologies. Reimplementing software interfaces also promotes innovation by countering network effects and lock-in effects that inhibit competition. It is exceptionally important that the Court review the decisions below to preserve software interface reimplementation and the vitality of the software industry.

A. The Computer Industry Has Long Relied on Freely Reimplementing Software Interfaces to Foster Innovation and Competition

Oracle's attempt to assert copyright in the Java API is historically anomalous and jeopardizes the unparalleled innovation and competition that continue to flourish across the computer industry. The first practical description of an API appeared in 1951, *see generally* Maurice V. Wilkes, David J. Wheeler & Stanley Gill, *The Preparation of Programs for an Electronic Digital Computer* (1951), and the specific phrase "application programming interface" dates to at least 1968, *see* Ira W. Cotton & Frank S. Greatorex, Jr., Data Structures and Techniques for Remote Computer Graphics, Am. Fed'n Info. Processing Soc'ys Fall Joint Computer Conf. 533, 534-35 (1968). Programmers have freely reimplemented software interfaces throughout the ensuing decades. By creating standard specifications for computer programs to communicate with each other, uncopyrightable software interfaces have promoted competition in personal computing and led to the rise of popular operating systems, programming languages, the Internet, and cloud computing.

i. Software interfaces unleashed the personal computer revolution

Reimplementing software interfaces made personal computing commonplace. IBM released its first home computer in 1981. Software companies developed an ecosystem of products to run on IBM's machine, including the popular spreadsheet program Lotus 1-2-3. To run these programs, however, users had to purchase IBM's PC because the programs required full compatibility with IBM's basic input-output system (BIOS) responsible for starting the operating system and initializing the computer's hardware when turned on. To compete with IBM, other computer manufacturers, like Compaq, and software companies, like Phoenix, reimplemented the BIOS API, including its SSO, to enable users to run their favorite IBM-compatible software on competing machines.

Thus, reimplementing the BIOS API resulted in the manufacture and sale of faster, cheaper, and compatible alternatives to IBM's PC, including DOS, the operating system responsible for Microsoft's early success. If copyright had prevented competitors from reimplementing IBM's BIOS API and making IBM-compatible PCs, companies like Microsoft would never have been able to revolutionize personal computing.

ii. Software interfaces created the world's most ubiquitous operating systems

Operating systems, the fundamental programs responsible for managing all of a computer's hardware and software resources, depend on software interface reimplementation. The first modern operating system, Unix, was co-developed by amicus Ken Thompson at AT&T Bell Labs and released in 1969. AT&T licensed Unix's source code to academic institutions for a nominal fee, leading to widespread adoption. Because commercial licenses from AT&T were costly and restrictive, and because hardware evolutions outpaced AT&T's Unix API, programmers reimplemented the API themselves.

Today, nearly 70% of websites run on Unix-based operating systems, including the popular open source operating system Linux. *See Usage Statistics and Market Share of Unix for Websites*, W3Techs (Nov. 30, 2018), https://w3techs.com/technologies/details/os-unix/all/all. Linux alone runs nearly 40% of Internet servers and the 500 fastest supercomputers in the world. *See id*; Steven J. Vaughan-Nichols, *Linux Totally Dominates Supercomputers*, ZDNet (Nov. 14, 2017, 12:04 PM PST), http://www.zdnet.com/article/linux-totally-dominates-supercomputers. Android's operating system, the most popular in the world, *see* Burke, is itself built atop Linux. Apple, co-founded by amicus Steve Wozniak, also reimplemented the Unix API for its desktop OS X and mobile iOS operating systems. Programmers' ability to reimplement the Unix API established a standardized design for the fundamental program running on any computer: its operating system.

iii. Software interfaces fueled widespread adoption of popular programming languages

One of the most influential programming languages, C, became widespread due to the relative ease of reimplementing its API to enable C programs to run on different hardware. Open source enthusiasts reimplemented a version of C compatible with Unix, and industry leaders like Microsoft and Google reimplemented C for their own products. Other popular programming languages like C++, created by amicus Bjarne Stroustrup, also proliferated due in part to reimplementations of their APIs.

Similarly, Sun reimplemented existing APIs for Java. Java reimplemented C's math API, which includes methods for calculating a variety of mathematical functions. While at Sun, amicus Joshua Bloch oversaw Sun's reimplementation of the Perl programming language's regular expression API for Java, which allows sophisticated text searches and alterations. Oracle's attempt to copyright Java's API and hold Google liable for infringement of the resulting `java.util.regex` API ignores Java's own history of API reimplementation.

iv. Software interfaces enable computer networks, including the internet, to function

The Internet relies on programmers' ability to reimplement standardized interfaces to transmit data. Copyrighting those interfaces would defeat the Internet's goal of creating a global network of interconnected computers. In 1983, the Berkeley Systems Research Group released the Berkeley Systems Distribution (BSD) sockets API. Sockets control the endpoints for any communication over the Internet. Because the BSD sockets API was not copyrighted, it became widely adopted: Every major operating system reimplemented it to enable Internet communication. Thus, programmers can write standardized software compatible across computers to manage Internet connectivity.

v. Software interfaces are fundamental to cloud computing

Finally, reimplementing software interfaces has been, and continues to be, fundamental to cloud computing. With cloud computing, developers can rent powerful computer hardware to run resource-intensive computations, like machine-learning algorithms, without having to purchase and manage expensive hardware themselves. Amazon's Web Services (AWS) API serves as the de facto industry standard for cloud computing. AWS itself reimplemented IBM's BIOS API, enabling familiar BIOS commands to run on Amazon's servers. AWS therefore allows programmers to write programs as if they were running on a standard PC rather than learn commands unique to Amazon.

Major competitors, including Microsoft and Google, have in turn adopted AWS's API. *See* Rita Zhang, *Access Azure Blob Storage from Your Apps Using S3 Java API*, Microsoft (May 22, 2016), https://www.microsoft.com/developerblog/2016/05/22/access-azure-blob-storage-from-your-apps-using-s3-api; *Cloud Storage Interoperability*, Google Cloud (last updated Oct. 23, 2018), https://cloud.google.com/storage/docs/interoperability. Rather than compete on the API's design, cloud providers compete on business factors—like price and customer service—and on implementation factors—like latency, downtime, and redundancy. Software interface reimplementation therefore fosters competition in the cloud by allowing customers to transfer their data or programs to competing cloud providers that offer cheaper or better service without having to learn an entirely new interface or rewrite their software to conform to a new specification.

B. Restricting the Reimplementation of Software Interfaces Will Stifle Competition by Increasing Entry Barriers for Startups

The decisions below jeopardize the market for software. Reimplementing software interfaces enables startups to counter network effects and compete with established players. Network effects arise when a service's value increases along with its number of users. They make users unlikely to switch even to technically "better" competing software services that have not yet established a large userbase because much of a service's value comes from its community of users and its secondary market of compatible services. For example, a developer might choose not to learn a new programming language unless it is used by potential employers, even if that language is more intuitive than others and produces efficient results. On the other hand, an archaic language used by institutional employers is worth learning, regardless of its inefficiencies. Uncopyrightable software interfaces address network effect barriers by enabling startups to plug into existing systems and grow through cumulative improvements.

Just as the first car would look laughable today, the first word processing software would be a laughable replacement for modern applications. Yet a steering wheel, turn signals, and gas and brake pedals have been standard in cars for over a century. If Tesla had to re-invent the standard driving interface to make electric-powered cars, it would face high barriers in attracting new customers. *See* Fred von Lohmann, *The New Wave: Copyright and Software Interfaces in the Wake of* Oracle v. Google, 31 Harv.

J.L. & Tech. 517, 517 (2018). In software, treating interfaces as copyrightable would be like requiring car manufacturers to invent a substitute for the steering wheel. Startups would not risk manufacturing such a car, and even if they did, consumers likely would not purchase it.

Furthermore, extending copyright to software interfaces would enable companies, like Sun, to monopolize standard interfaces. Companies could initially make their interfaces freely available to lure developers to their platform, and then, after attracting a significant number of developers, demand a licensing fee for further use. These fees would be passed on to consumers, making software more expensive. Copyrightable interfaces could also curtail employee mobility because different employers would use competing proprietary APIs, and employees with expertise in one proprietary API would be less desirable to employers using another. Innovation could stagnate.

Amazon, for example, could follow Oracle's lead and use the decisions below to force every company that has reimplemented its cloud storage APIs to pay a licensing fee, stifling competition in a vibrant market valued at $42 billion in 2017 and projected to reach $72 billion by 2019. *See* Jay Greene & Laura Stevens, *"You're Stupid If You Don't Get Scared": When Amazon Goes from Partner to Rival*, Wall St. J. (June 1, 2018, 5:30 AM ET), https://www.wsj.com/articles/how-amazon-wins-1527845402. Amazon could gain a monopoly over cloud storage until its competitors redesigned their systems from scratch to avoid infringing on Amazon's APIs. The decisions below will make copyright an additional tool for incumbents to stave off competition.

Relying on fair use is no answer. A fair use standard creates uncertainty because it depends on fact-intensive, case-by-case determinations which can result, as demonstrated by this case, in lengthy and prohibitively expensive litigation. Rather than risk crippling lawsuits, startups will choose not to enter the market at all or will undertake inefficient workarounds. Conditioning API reimplementation on fair use would impede innovation and competition almost as much as denying reimplementation outright: Users will suffer from fewer product choices, higher prices, and incompatible software.

C. Restricting the Reimplementation of Software Interfaces Will Exacerbate Lock-In Effects and Create an "Orphan Software" Problem

Reimplementing software interfaces protects consumers from lock-in effects by promoting interoperability among operating systems, programs, and Internet browsers. Consumers depend on operating systems that run on their hardware, programs that run across operating systems, and Internet applications that run across browsers. Under the decisions below, software interfaces enabling interoperability might require expensive licenses, and their owners could significantly restrict their use. Consumers will face higher prices and fewer choices. Software will become harder to use because switching to a competing service will require users to learn an unfamiliar interface. Rather than switch to more innovative software, users will remain locked in to outdated systems.

If software interfaces are copyrightable, it will become economically infeasible to continue using orphan software, i.e., software no longer supported or updated by its creator. Previously, when copyrighted software became unsupported, developers could reimplement its interface to allow it to run on new systems. When NASA needed to refurbish old manufacturing robots for a project, for example, it contracted with a company to reimplement the interface necessary for integrating newly manufactured memory chips with the old robot hardware. Had the interface been copyrighted, NASA would have needed to purchase new robots at a significantly higher cost.

Restricting the reimplementation of software interfaces could make generations of software unusable by the people and organizations who paid for them, hindering, rather than promoting, "the Progress of Science and useful Arts." U.S. Const. art. I, § 8, cl. 8. Copyrightable interfaces would particularly harm public, nonprofit, and research-based entities because of their limited resources, undermining crucial services for public health and safety, national defense, and access to justice.

CONCLUSION

The Court should grant the petition for a writ of certiorari to review the decisions below and ensure continued innovation and competition in the software industry.

Respectfully submitted,

Phillip R. Malone
Counsel of Record
JUELSGAARD INTELLECTUAL PROPERTY AND INNOVATION CLINIC
MILLS LEGAL CLINIC AT STANFORD LAW SCHOOL
559 Nathan Abbott Way
Stanford, CA 94305
(650) 725-6369
pmalone@law.stanford.edu

February 25, 2019

APPENDIX A — LIST OF AMICI CURIAE

(In alphabetical order)

Amici sign this brief on their own behalf and not on behalf of the companies or organizations with which they are affiliated; those affiliations are for identification purposes only.[1]

1. **Harold Abelson.**** Dr. Harold "Hal" Abelson is a Professor of Electrical Engineering and Computer Science at MIT, a fellow of the IEEE, and a founding director of both Creative Commons and Public Knowledge. He directed the first implementation of the Logo computing language for the Apple II, which made the language widely available on personal computers beginning in 1981, and published a popular book on Logo in 1982. Abelson co-developed MIT's introductory computer science subject, which included innovative advances in curricula designed for students pursuing different kinds of computing expertise. These curricula had a worldwide impact on university computer science education. Notable awards include the Bose Award (MIT School of Engineering, 1992), the Taylor L. Booth Education Award (IEEE-CS, 1995), and the SIGCSE 2012 Outstanding Contribution to Computer Science

[1] The * indicates five amici who are current Google employees, ** indicates two amici who receive some support from Google, † indicates two amici who testified as unpaid fact witnesses at trial in this case, and ‡ indicates one amicus who was retained as an expert by Google but did not testify at trial. Each of these amici sign this brief based on their personal experience and beliefs as individual computer scientists whose work in the field long preceded their affiliation with Google or their participation in this case.

Education (ACM, 2012). Abelson holds an A.B. from Princeton University and a Ph.D. in mathematics from MIT.

2. **Brian Behlendorf.** Brian Behlendorf is Executive Director of Hyperledger, an open source blockchain technology collaborative based at the Linux Foundation. He also serves as Chairman of the Board of the Electronic Frontier Foundation, and a member of the boards of the Mozilla Foundation and Benetech. He also co-founded the Apache Software Foundation, has worked as CTO for the World Economic Forum, advised and served the White House on open data and open source software issues, and co-founded a string of successful startups.

3. **Jon Bentley.** Jon Bentley's research interests include programming techniques, algorithm design, and the design of software tools and interfaces. He has written three books on programming and over a hundred articles on a variety of topics, ranging from the theory of algorithms to software engineering. He received a B.S. from Stanford in 1974 and an M.S. and Ph.D. from the University of North Carolina in 1976, then taught Computer Science at Carnegie Mellon for six years. He joined Bell Labs Research in 1982, where he became a Distinguished Member of Technical Staff. He left Bell Labs in 2001 to join Avaya Labs Research, from which he retired in 2013. He has been a visiting faculty member at West Point and Princeton, and has been a member of teams that have shipped library functions, software tools, telephone switches, telephones and web services. He holds over 40 US Patents. In March 2000 he received the Dr. Dobb's

Excellence in Programming Award for "advancing the craft of computer programming."

4. **Matthew Bishop.** Matthew Bishop received his Ph.D. in computer science from Purdue University, where he specialized in computer security, in 1984. He is on the faculty at the Department of Computer Science at the University of California at Davis. His main research area is the analysis of vulnerabilities in computer systems, including modeling, detecting, and analyzing them. Currently, he has research projects involving data sanitization, modeling election processes, and analyzing attacks. He is co-leading an education project aimed at improving the practice of programming using a "secure programming clinic" to help students improve the robustness and security of their programs. He has been active in the area of UNIX security since 1979, and has presented tutorials at SANS, USENIX, and other conferences. He also has done work on electronic voting, and was one of the two principle investigators of the California Top-to-Bottom Review, which performed a technical review of all electronic voting systems certified for use in the State of California. His textbook, Computer Security: Art and Science (Addison-Wesley, 2002), is used at many academic institutions throughout the world.

5. **Joshua Bloch.**† Joshua Bloch is an expert on API design, with over a quarter century of experience. He is a Professor of Computer Science at Carnegie Mellon University. Previously, he was Chief Java Architect at Google, a Distinguished Engineer at Sun Microsystems, and a Senior Systems Designer at Transarc Corporation. He led the design

and implementation of numerous Java APIs and language features, including the award-winning Java Collections Framework. He is the author of several books, including the bestselling, Jolt Award-winning Effective Java (Addison-Wesley, 2001; Second Edition, 2008), the de facto standard guide to Java best practices. He served on the National Academies CSTB Certifiably Dependable Software Committee. He holds a B.S. from Columbia and a Ph.D. in Computer Science from Carnegie Mellon University.

6. **Gilad Bracha.** Gilad Bracha is the creator of the Newspeak programming language and a well known researcher in the area of object-oriented programming languages. He was awarded the senior Dahl-Nygaard prize in 2017. Previously, he has worked at Google, as a VP at SAP Labs in Palo Alto, a Distinguished Engineer at Cadence, and a Computational Theologist and Distinguished Engineer at Sun. He has authored or co-authored several books including the Java Language and Virtual Machine Specifications, and the Dart Programming Language. Prior to joining Sun, he worked on Strongtalk, the Animorphic Smalltalk System. He received his B.Sc.in Mathematics and Computer Science from Ben Gurion University in Israel and a Ph.D. in Computer Science from the University of Utah.

7. **Daniel Bricklin.** Daniel "Dan" Bricklin is best known for conceiving and co-developing VisiCalc, the pioneering electronic spreadsheet, while he was a student at the Harvard Business School. VisiCalc, released in 1979, is widely credited for fueling the rapid growth of the personal computer industry.

There is a direct line of upward-compatibility for data files from VisiCalc to Lotus 1-2-3 to Microsoft Excel and then to Google Docs. Prior to VisiCalc, he worked at Digital Equipment Corporation developing early screen-based word processing software. He is the author of a popular app for the Apple iPad, and is currently Chief Technology Officer of Alpha Software Corporation. Mr. Bricklin is a member of the National Academy of Engineering, a Fellow of the Computer History Museum, an Association for Computing Machinery (ACM) Fellow, and a recipient of the ACM Software System Award, the ACM Grace Murray Hopper Award, and the Western Society of Engineers Washington Award. Mr. Bricklin holds a BS in Electrical Engineering/Computer Science from MIT and an MBA from the Harvard Graduate School of Business Administration.

8. **Frederick Brooks.** Frederick Brooks is the Kenan Professor of Computer Science (Emeritus) at University of Northern Carolina at Chapel Hill. As Corporate Project Manager for IBM's System/360 (mainframe) computer family hardware and the Operating System/360 software, he in 1964 switched the standard computer byte size from 6 to 8 bits. He was an architect of the Stretch and Harvest supercomputers. He founded UNC's Computer Science Department. He's researched computer architecture, software engineering, the design process, and graphics virtual environments. He wrote The Mythical Man-Month, The Design of Design, and with G.A. Blaauw, Computer Architecture. Honors include the National Medal of Technology, the ACM Turing award, the National

Academies of Engineering and Sciences, and British and Dutch academies.

9. **Edwin Catmull.** Edwin "Ed" Catmull is co-founder of Pixar Animation Studios and was president of Pixar Animation and Disney Animation, vice president of Lucasfilm Computer Division, and head of the Computer Graphics Laboratory at the New York Institute of Technology. He is an architect of the RenderMan rendering software, which was used in 44 of the last 47 films nominated for an Academy Award in the Visual Effects category. Catmull was honored with five Academy Awards including the Gordon E. Sawyer Award for lifetime achievement in the field of computer graphics. He also holds the IEEE John von Neumann Medal for pioneering contributions to the field of computer graphics, the ACM SIGGRAPH Steven Anson Coons Award for Outstanding Creative Contributions to Computer Graphics, the Carnegie Mellon University Entertainment Technology Center Randy Pausch Award, the Progress Medal and the Fuji Gold Medal awards from the Society of Motion Picture and Television Engineers, and the animation industry's Ub Iwerks Award for technical advancements in the art or industry of animation. Catmull is a Fellow of the Association for Computing Machinery, the Computer History Museum, and the Visual Effects Society. He received his Ph.D. in computer science from the University of Utah.

10. **Rick Cattell.**[‡] R. G. G. "Rick" Cattell is an independent consultant in database systems. He previously worked as a Distinguished Engineer at Sun Microsystems. Dr. Cattell served for 20 years at Sun Microsystems in management and senior

technical roles, and for 10 years in research at Xerox PARC and Carnegie Mellon University. He is best known for his contributions in database systems and middleware, including database scalability, Enterprise Java, object/relational mapping, object-oriented databases, and database interfaces. At Sun he instigated Enterprise Java, JDBC, Java DB, and Java Blend, and contributed to many Java APIs and products. He previously developed Xerox PARC's Cedar DBMS, Sun's Simplify database GUI, and SunSoft's CORBA-database integration. He is a co-founder of SQL Access (predecessor to ODBC), founder and chair of the Object Data Management Group (ODMG), author of the world's first monograph on object/relational and object databases, recipient of the ACM Outstanding Ph.D. Dissertation Award, and an ACM Fellow.

11. **Vinton G. Cerf.*** Vinton G. "Vint" Cerf is an Internet pioneer, and VP and Chief Internet Evangelist for Google, where he contributes to global policy development and the continued spread of the Internet. Widely known as one of the "Fathers of the Internet," Cerf is the co-designer of the TCP/IP protocols and the architecture of the Internet. He has served in executive positions at MCI, the Corporation for National Research Initiatives, the Defense Advanced Research Projects Agency, and on the faculty of Stanford University. Cerf served as chairman of the board of the Internet Corporation for Assigned Names and Numbers (ICANN) from 2000-2007. Cerf is a Fellow of the IEEE, ACM, and the American Academy of Arts and Sciences, the International Engineering Consortium, the Computer History Museum, and is

a member of the National Academy of Engineering. He is a former President of the ACM and Founding President of the Internet Society. President Obama appointed him to the National Science Board in 2012. Cerf is a recipient of numerous awards and commendations in connection with his work on the Internet, including the US Presidential Medal of Freedom, US National Medal of Technology, the Queen Elizabeth Prize for Engineering, the ACM Turing Award, Officer of the Legion d'Honneur and 29 honorary degrees. In December 1994, People magazine identified Cerf as one of that year's "25 Most Intriguing People." Cerf holds a B.S. from Stanford, and an M.S. and Ph.D. from UCLA.

12. **David Clark.** David Clark received his PhD from MIT in 1973. He is a Senior Research Scientist at the MIT Computer Science and Artificial Intelligence Laboratory. He is technical director of the MIT Internet Policy Research Initiative. Since the mid-70s, he has been leading the development of the Internet; from 1981-1989 he acted as Chief Protocol Architect in this development, and chaired the Internet Activities Board. He is past chairman of the Computer Science and Telecommunications Board of the National Academies, and has contributed to a number of studies on the societal and policy impact of computer communications. He is a member of the National Academy of Engineering and the American Academy of Arts and Sciences, and serves as a member of the Academy Council. His recently published book, Designing an Internet, discusses the key role that modularity and open interfaces played in the success of the Internet.

13. **William Cook.** William Cook is an Associate Professor in the Department of Computer Sciences at the University of Texas at Austin. His research is focused on object-oriented programming, programming languages, modeling languages, and the interface between programming languages and databases. Prior to joining UT in 2003, Dr. Cook was Chief Technology Officer and co-founder of Allegis Corporation. He was chief architect for several award-winning products, including the eBusiness Suite at Allegis, the Writer's Solution for Prentice Hall, and the AppleScript language at Apple Computer. At HP Labs his research focused on the foundations of object-oriented languages, including formal models of mixins, inheritance, and typed models of object-oriented languages. He completed his Ph.D. in Computer Science at Brown University in 1989. He recieved the Dahl-Nygaard Senior Prize in 2014 for his contributions to the theory and practice of object-oriented programming.

14. **Miguel de Icaza.** Miguel de Icaza is currently a Distinguished Engineer at Microsoft and was an early contributor to Linux projects. In 1997, he co-founded the GNOME project, with the goal to create a completely free desktop environment. In 2001, he co-founded and directed the Mono Project, with the goal to reimplement Microsoft's .NET development platform on Linux. He has started two companies: Ximian in 1999, which focused on the Linux desktop and was sold to Novell in 2003; and Xamarin which was founded in 2011 to build mobile development tools and was sold to Microsoft in 2016. He has received numerous awards and recognitions including: the Free Software Foundation Free Software Award,

the MIT Technology Review Innovator of the Year Award, and was named one of Time Magazine's 100 innovators for the new century.

15. **Jeffrey Dean.*** Jeff Dean joined Google in 1999 and is currently one of two Senior Fellows in the company, where he leads Google's Research, AI and Health efforts. He has co-designed/implemented five generations of Google's crawling, indexing, and query serving systems, and co-designed/implemented major pieces of Google's initial advertising and AdSense for Content systems. He is also a co-designer and co-implementor of Google's distributed computing infrastructure, including the MapReduce, BigTable and Spanner systems, protocol buffers, LevelDB, systems infrastructure for statistical machine translation, the TensorFlow open-source machine learning system, and a variety of internal and external libraries and developer tools. Prior to joining Google, Jeff did computer systems research at Digital Equipment Corporation's Western Research Lab. Jeff has also worked for both the Centers for Disease Control and the World Health Organization, designing computer software for epidemiology and for statistical analysis of the HIV/AIDS pandemic. He is a Fellow of the ACM and the AAAS, a member of the U.S. National Academy of Engineering, and a recipient of the Mark Weiser Award and the ACM-Infosys Foundation Award in the Computing Sciences. Jeff holds a B.S., summa cum laude, in computer science and economics from the University of Minnesota, and a M.S. and Ph.D. in computer science from the University of Washington.

16. **L. Peter Deutsch.** Dr. L. Peter Deutsch received a Ph.D. in Computer Science from U.C. Berkeley in 1973. Subsequently at Xerox PARC, he helped develop the Interlisp-D, Cedar Mesa, and Smalltalk-80 programming systems. Deutsch's work on Smalltalk implementation, among other innovations, was an important contributor to the just-in-time compilation technology now used widely to dramatically improve the performance of Java and JavaScript implementations. He is also the author of a number of RFCs and of the The Eight Fallacies of Distributed Computing, and originated the Deutsch limit adage about visual programming languages. From 1986 to 1991, as Chief Scientist at ParcPlace Systems, he developed cross-platform JIT technology. From 1986 to 2003, dba Aladdin Enterprises, he was the creator of Ghostscript, an Open Source implementation of the PostScript language. In 1993, he was a co-recipient of the ACM Software System Award, and was also named a Distinguished Alumnus of the U.C. Berkeley Computer Science program; he was named an ACM Fellow in 1994. In 1994, he founded Artifex Software to license Ghostscript commercially while continuing its development and its release as Open Source; Artifex today is a multi-million-dollar business. In 1999-2000, he served on the board of the Open Source Initiative. He is a co-inventor on two patents.

17. **Whitfield Diffie.** Dr. Whitfield Diffie serves as advisor to a variety of startups, primarily in the field of security. He is best known for discovering the concept of public key cryptography, which underlies the security of internet commerce and all modern secure communication systems. Diffie's

two principal positions after leaving Stanford University in the late 1970s were Manager of Secure Systems Research for Bell-Northern Research, the laboratory of the Canadian telephone system, and Chief Security Officer at Sun Microsystems. Diffie received the 2015 Turing Award and in 2017 was elected to both the National Academy of Engineering and the Royal Society.

18. **David L. Dill.** David Dill is The Donald E. Knuth Professor, Emeritus, in the School of Engineering at Stanford University. Prof. Dill's Ph.D. thesis, "Trace Theory for Automatic Hierarchical Verification of Speed Independent Circuits" was named as a Distinguished Dissertation by the Association for Computing Machinery (ACM), and published as such by M.I.T. Press in 1988. He was named a Fellow of the Institute of Electrical and Electronics Engineers (IEEE) in 2001 for his contributions to verification of circuits and systems, and a Fellow of the ACM in 2005 for contributions to system verification and for leadership in the development of verifiable voting systems. In 2008, he received the first "Computer-Aided Verification" award for fundamental contributions to the theory of real-time systems verification. In 2013, he was elected to the National Academy of Engineering and the American Academy of Arts and Sciences. In 2016, he received the Alonzo Church Award for Outstanding Contributions to Logic and Computation.

19. **Lester Earnest.** Lester Earnest is a widely-recognized computer scientist, best known for his deep involvement with the Advanced Research Project Agency Network (ARPAnet) startup committee, which led to his invention of

the Finger social networking protocol. He served as a US Navy Aviation Electronics Officer and Digital Computer Project Officer at the Naval Air Development Center, and later joined MIT to help design the Semi-Automatic Ground Environment air defense system. Later, he innovated numerous early features in the nascent field of word processing, including the first spell-checker, search engine, self-driving vehicle, robotic hand-eye assembler that took verbal instructions, online restaurant reviews, online news service, and a number of other successful innovations.

20. **Stu Feldman.** Dr. Stuart Feldman is a well known computer scientist and specialist in programming languages. Feldman is currently Chief Scientist of Schmidt Futures where he is responsible for the Scientific Knowledge programs, including creating fellowship programs, supporting nascent innovative research projects, and driving new platforms and larger research projects that aim to change the way scientific research is done and the way universities support it. Feldman did his academic work in astrophysics and mathematics and earned his AB at Princeton and his PhD at MIT. He has been awarded an honorary Doctor of Mathematics by the University of Waterloo and an Honorary Doctorate from the Technion. He is former President of ACM (Association for Computing Machinery, the largest professional society in computing) and former member of the board of directors of the AACSB (Association to Advance Collegiate Schools of Business). Feldman is best known for writing "Make" and other essential tools. He received the 2003 ACM Software System Award. He is a Fellow of the IEEE, ACM, and AAAS. He is Board Chair

of the Center for the Minorities and Disabled in IT, serves on a number of university advisory boards and National Academy panels, and has served on a wide variety of government advisory committees.

21. **Martin Fowler.** Martin Fowler is an author and educator on software development. He is currently chief scientist at ThoughtWorks, a global system delivery and consulting firm. Mr. Fowler concentrates on the design of enterprise software: what makes a good design and what practices are needed to enhance it. He is the author of seven books on software development, which have over a million copies in print in over a dozen languages. He is the editor of a book series with Addison-Wesley on software design. His website, http://martinfowler.com, is a wide-ranging resource of software development techniques attracting around 150,000 visitors per month.

22. **Bob Frankston.** Bob Frankston co-founded Software Arts with Dan Bricklin, where he implemented VisiCalc. Prior to VisiCalc, Bob worked to develop the first online financial language. He worked at Lotus Development where he created Lotus Express, and at Microsoft, where he championed home networking which extended the Internet into people's homes. He is a fellow of the IEEE, the ACM and the Computer History Museum. Bob is a recipient of the the ACM Software System Award, the MIT William L. Stewart Award, the MIT LCS Industrial Achievement Award, and the Western Society of Engineers Washington Award. He holds Math and EECS BS Degrees, and Masters and Engineers degrees in EECS from MIT. He is

on the Board of Governors of the IEEE Consumer Electronics Society.

23. **Neal Gafter.** Neal Gafter is a Principal Engineer at Microsoft, where he is a technical lead for the Roslyn Project (Microsoft's implementation for the C# and Visual Basic programming languages). Previously he was a software engineer and Java Evangelist at Google, where he designed and implemented the distributed storage architecture for Google Calendar, and a Senior Staff Engineer at Sun Microsystems, where he led the development of the Java compiler and implemented the Java language features in releases 1.4 through 5.0. Neal was a member of the C++ Standards Committee and led the development of C and C++ compilers at Sun Microsystems, Microtec Research, and Texas Instruments. He holds a B.S. in computer engineering from Case Western Reserve University and a Ph.D. in computer science from the University of Rochester.

24. **Erich Gamma.** Erich Gamma is a Microsoft Technical Fellow. He is co-author of "Design Patterns: elements of reusable object-oriented software" which is a highly influential book in the software industry. It has become a classic in the literature of object-oriented development, offering timeless and elegant solutions to common problems in software design. The book has sold over 1/2 million copies and was awarded the Association of Computing Machinery ("ACM") Programming Language Award. Prior to joining Microsoft, Gamma was a Distinguished Engineer at IBM. He was one of the leaders of the Eclipse Project and was responsible for its Java Development Tools. For the

contribution to Eclipse he was awarded the ACM Software System Award.

25. **Allan Gottlieb.** Allan Gottlieb is a professor of computer science at New York University where he has been teaching for 39 years and is a Fellow of the Association of Computing Machinery. He has co-authored a book (with George Almasi) on parallel computing. The "Ultracomputer" group he led introduced the fetch-and-add coordination instruction still in use today.

26. **Anoop Gupta.** Anoop Gupta is co-founder and CEO of SeekOut, a startup providing recruiting and engagement tools for hard-to-find and diverse technology talent. Prior to SeekOut, Anoop spent 18 years at Microsoft. He was Distinguished Scientist at Microsoft Research leading Telepresence and Natural-User-Interface efforts; he directly reported to Bill Gates as his technology assistant; he headed all enterprise communications products (Exchange, Skype) as Corporate Vice President of Unified Communications; and he was Corporate Vice President for Global Technology Policy for Microsoft. Prior to Microsoft, Anoop was a tenured professor of Computer Science at Stanford, where he founded one of the earliest companies in the streaming media space that was acquired by Microsoft in 1997. He has a Ph.D. in Computer Science from Carnegie-Mellon University. He holds over 85 issued US patents and is author 100+ research papers.

27. **Robert Harper.** Robert Harper is a professor in the computer science department at Carnegie Mellon University. He holds a Ph.D. in computer

science from Cornell University. His main research interest is in the application of type theory to the design and implementation of programming languages and to the mechanization of their meta-theory. Harper made major contributions to the design of the Standard ML programming language and the LF logical framework. Harper is a recipient of the Allen Newell Medal for Research Excellence and the Herbert A. Simon Award for Teaching Excellence, and is an Association for Computing Machinery Fellow.

28. **Anders Hejlsberg.** Anders Hejlsberg is a Technical Fellow in the Cloud and Enterprise group at Microsoft Corporation and has been designing and implementing programming languages and development tools for over 35 years. Anders is the lead architect of the TypeScript open-source project and the original designer of the C# programming language. Before joining Microsoft in 1996, Anders was a Principal Engineer at Borland International. As one of the first employees of Borland, he was the original author of Turbo Pascal and later worked as the Chief Architect of the Delphi product line. Anders studied Engineering at the Technical University of Denmark.

29. **Martin Hellman.** Martin Hellman is co-inventor, with Whitfield Diffie and Ralph Merkle, of public key cryptography, the technology that secures e-commerce and protects trillions of dollars per day in financial transactions. His many honors include the top prize in computer science (the million dollar ACM Turing Award) and election to the National Academy of Engineering. Hellman has a deep interest in the ethics of technological development

and serves on Advisory Boards at the Federation of American Scientists and Verified Voting. He was on the faculty at MIT (1969-1971) and Stanford University (1971-1996), where he is now Professor Emeritus of Electrical Engineering.

30. **Maurice Herlihy.** Maurice Herlihy has an A.B. in Mathematics from Harvard University, and a Ph.D. in Computer Science from M.I.T. He has served on the faculty of Carnegie Mellon University and the staff of DEC Cambridge Research Lab. He is the recipient of the 2003 Dijkstra Prize in Distributed Computing, the 2004 Gödel Prize in theoretical computer science, the 2008 ISCA influential paper award, the 2012 Edsger W. Dijkstra Prize, and the 2013 Wallace McDowell award. He received a 2012 Fulbright Distinguished Chair in the Natural Sciences and Engineering Lecturing Fellowship, and he is fellow of the ACM, a fellow of the National Academy of Inventors, the National Academy of Engineering, and the American Academy of Arts and Sciences.

31. **Tom Jennings.** Tom Jennings has specialized in computers, software, and electronics since 1977; computer networking since 1984; and the Internet since 1992. Jennings was on the team that wrote the interface specification (API in today's parlance) for Phoenix Software's IBM compatible ROM BIOS. Jennings is the creator of FidoNet, the first and most influential message and file networking system protocol for networking computer bulletin boards. Jennings built Wired magazine's first internet presence as its first webmaster and ran an early regional internet service provider, TLGnet.

Currently, Jennings is on the faculty at Calarts Art+Technology program.

32. **Mitchell Kapor.** Mitchell Kapor founded Lotus Development Corp. in 1982 and co-created Lotus 1-2-3. He served as the President (later Chairman) and Chief Executive Officer of Lotus from 1982 to 1986, and as a Director until 1987. In 1990, Mr. Kapor co-founded the Electronic Frontier Foundation, and served as its chairman until 1994. From 1999 to 2001, he was a partner at Accel Partners, a venture capital firm based in Palo Alto, California. In 2003, Mr. Kapor became the founding Chair of the Mozilla Foundation, which is dedicated to the development and promulgation of standards-compliant, open source web browser software. From 1994-1996, he served as Adjunct Professor at the Massachusetts Institute of Technology's Media Lab, where he taught courses on software design, democracy and the Internet, and digital community. From 2005 to 2009, Mr. Kapor was on the faculty of the Information School at the University of California, Berkeley, as Lecturer (2005-2006) and Adjunct Professor (2006-2009), where he co-taught "Open Source Development and Distribution of Information." He is currently a Partner at Kapor Capital, which invests in seed-stage, social-impact tech startups; and Co-Chair of the Kapor Center for Social Impact, which pursues creative strategies to leverage tech for positive, progressive change. Mr. Kapor received a B.A. from Yale College in 1971 and studied psychology, linguistics, and computer science as part of an interdisciplinary major in Cybernetics.

33. **Alan Kay.** Alan Kay is one of the pioneers of object-oriented programming, personal computing, and graphical user interfaces. For this work, Dr. Kay has received the Draper Prize from the National Academy of Engineering, the ACM Turing Award, and the Kyoto Prize from the Inamori Foundation. Alan has been elected a fellow of the American Academy of Arts and Sciences, the National Academy of Engineering, the Royal Society of Arts, and the Computer History Museum. Alan has held fellow positions at HP, Disney, Apple, and Xerox, and has served as the chief scientist at Atari. While at Xerox PARC, he was one of the key members there to develop prototypes of networked workstations using the programming language Smalltalk. He is an adjunct professor of computer science at UCLA and an advisor to One Laptop per Child. At Viewpoints Research, Alan also continues his work with "powerful ideas education" for the world's children, as well as the development of advanced personal computers and networking systems.

34. **Brian Kernighan.**** Brian Kernighan is a professor in the Computer Science Department of Princeton University. He worked at Bell Labs alongside Unix creators Ken Thompson and Dennis Ritchie and contributed to the development of Unix. He co-authored a number of Unix programs, including widely used document preparation tools. He is also the author or co-author of eleven books on computing, including the first book on the C programming language with Dennis Ritchie; these books have been translated into more than two dozen languages. He is also a co-creator of the AWK and AMPL programming languages. In

collaboration with Shen Lin he devised well-known heuristics for two fundamental NP-complete optimization problems: graph partitioning and the traveling salesman problem. Kernighan received a Bachelor's degree in engineering physics from the University of Toronto, and his Ph.D. in electrical engineering from Princeton University. He is a member of the National Academy of Engineering.

35. **David Klausner.** David Klausner has over fifty years of software/hardware development and consulting experience in the computer and software industry. He has written software for commercial products as an engineer, developer, supervisor, project manager, department manager, middle manager and company executive. He has worked in forensic investigation and in reverse engineering. He has been employed in various capacities for various companies, such as Microsoft, AT&T, Cisco, IBM, Hewlett Packard, and Intel Corporation. He holds a Bachelors of Arts degree in Mathematics, and a Master of Science degree in Electrical Engineering. He has taught programming, public speaking, has guest lectured at Stanford University, and been an invited speaker by IBM, AT&T, and others. His technical opinions have been confirmed in several cases by the United States Court of Appeals for the Federal Circuit.

36. **Kin Lane.** Kin is a computer scientist and API Evangelist working to understand the technology, business and politics of APIs and help share this insight with the world. He is the author of the book, Business of APIs, and is behind the popular API Evangelist blog. He has over twenty years of experience as a

programmer, database administrator, architect, product developer, manager, and executive in the API space.

37. **Ed Lazowska.** Ed Lazowska holds the Bill & Melinda Gates Chair in the Paul G. School of Computer Science & Engineering at the University of Washington. His research concerns the design, implementation, and analysis of high performance computing and communication systems, and more recently, the techniques and technologies of data-intensive discovery. He co-chaired (with Marc Benioff) the President's Information Technology Advisory Committee from 2003-05, and (with David E. Shaw) the Working Group of the President's Council of Advisors on Science and Technology to review the Federal Networking and Information Technology Research and Development Program in 2010. He is a Member of the National Academy of Engineering and a Fellow of the American Academy of Arts and Sciences.

38. **Doug Lea.** Doug Lea is a Professor of Computer Science at the State University of New York at Oswego. He is an author of books, articles, reports, and standardization efforts on object oriented software development including those on specification, design and implementation techniques, distributed, concurrent, and parallel object systems, and software re-usability; he has served as chair, organizer, or program committee member for many conferences and workshops in these areas. He is the primary author of several widely used software packages and components.

39. **Bob Lee.**† Bob Lee is CEO of Present Company, makers of Present, a new location-based social

network. Prior to that, as Square's CTO, Bob built Square's core products, scaled the team from 12 to 1200 people, and created Square Cash. Before Square, Bob worked at Google where he created Guice and was the core library lead for Android.

40. **Harry Lewis.** Harry Lewis is Gordon McKay Professor of Computer Science, where he has taught since 1974, and where he is also a Faculty Fellow of the Berkman Klein Center for Internet and Society. His students have included Bill Gates, Mark Zuckerberg, and computer science professors at Harvard, Yale, Brown, Stanford, MIT, and other leading universities. He has served both as dean of Harvard College and as interim dean of Harvard's School of Engineering and Applied Sciences. Lewis is a member of the board of directors of EPIC, the Electronic Privacy Information Center. He is a co-author of Blown to Bits: Your Life, Liberty, and Happiness after the Digital Explosion, and of several other books on computer science, mathematics, and higher education. He holds Bachelor's, Master's, and PhD degrees in Applied Mathematics from Harvard University.

41. **Sheng Liang.** Sheng Liang is a software entrepreneur. He is cofounder and CEO of Rancher Labs, an enterprise software company. He was CTO of the Cloud Platform group at Citrix Systems after their acquisition of Cloud.com, where he was co-founder and CEO. Sheng was co-founder and CTO of Teros, a provider of perimeter and network security solutions for enterprises and service providers, acquired by Citrix Systems in 2005. He also served as VP of Engineering at SEVEN Networks, and Director of Software

Engineering at Openwave Systems. He was a Staff Engineer in Java Software at Sun Microsystems, where he designed the Java Native Interface (JNI) and led the Java Virtual Machine (JVM) development for the Java 2 platform. He has a B.S. from the University of Science and Technology of China and a Ph.D. from Yale University.

42. **Barbara Liskov.** Barbara Liskov is one of the world's leading authorities on computer language and system design. Liskov joined MIT in 1972 as a member of the Department of Electrical Engineering and computer Science. She is also a member of the MIT laboratory for Computer Science and Artificial Intelligence and heads the programming methodology group. Her research interests lie in programming methodology, programming languages and systems, and distributed computing. Major projects include: the design and implementation of CLU, the first programming language to support data abstraction; the design and implementation of Argus, the first high-level language to support implementation of distributed programs; and the Thor object-oriented database system, which provides transactional access to persistent, highly available objects in wide-scale distributed environments. Liskov is a fellow of the American Academy of Arts and Sciences, the National Academy of Inventors, the Association for Computing Machinery, and the Massachusetts Academy of Science. She is a member of the National Academy of Sciences and the National Academy of Engineering. In 2009, she received the A.M. Turing Award from the ACM. Other honors include the Society of Women Engineers' Achievement Award, the IEEE von

Neumann medal, the ACM SIGPLAN Programming Languages Achievement Award, the University of Pennsylvania Harold Pender Award, the ACM SIGOPS Hall of Fame Award, the CMU and Tokyo University of Technology Katayanagi Award for Research Excellence, the ACM SIGOPS Lifetime Achievement Award, and five honorary doctorates. She holds a B.A. from UC Berkeley and a Ph.D. from Stanford.

43. **Paul Menchini.** Paul Menchini is the Chief Information Security Officer at the North Carolina School of Science and Mathematics. Previously, he held technical positions at HP, Intel, GE Microelectronics, CLSI and OrCAD. As a member of the "Woods Hole Summer Study on Hardware Description Languages," he contributed to the specifications for VHDL; subsequently, he edited two revisions of IEEE Std 1076 VHDL and developed the first commercially successful VHDL compiler. As part of the compiler project, he developed an API for a VHDL intermediate form, which was subsequently standardized by the IEEE. He holds a Masters in Computer Engineering from Stanford University and is the recipient of numerous technical awards, including charter membership in the "IEEE Golden Core."

44. **Douglas McIlroy.** Douglas McIlroy is an adjunct professor of computer science at Dartmouth College, to which he retired from Bell Laboratories. At Bell he headed a research department that originated the Unix operating system [and counts seven members of the National Academy of Engineering among its alumni]. He pioneered macroprocessors, was a designer of the PL/I programming language,

and provided PL/I for writing the Multics operating system. He has done basic work in speech synthesis, text processing, and graphics algorithms. In connection with Unix, he conceived "pipes" that enable one to combine concurrent computer processes, originated standard Unix programming tools, and edited the manual, some editions of which were trade books. He received awards for lifetime achievement and programming tools from the Usenix association and is a fellow of the American Association for the Advancement of Science and a member of the National Academy of Engineering.

45. **James H. Morris.** Dr. James H. Morris is a Professor Emeritus of Computer Science at Carnegie Mellon University, where he served as Dean of the Silicon Valley Campus, Dean of the School of Computer Science, Head of the Computer Science Department, and Director of the Information Technology Center, a joint project with IBM that developed a prototype university computing system. He founded Carnegie Mellon's Human Computer Interaction Institute, Robot Hall of Fame, and Silicon Valley Campus. He was an Associate Professor at UC Berkeley, where he developed two fundamental principles of programming languages: inter-module protection and lazy evaluation. He was co-discoverer of the Knuth-Morris-Pratt string-searching algorithm. He was Principal Scientist and Research Fellow at Xerox PARC, where he was part of the team that developed the Alto, a precursor to today's personal computers. He is a founder of MAYA Design Group and an ACM Fellow. He holds a B.S. from CMU and an M.S. and Ph.D. from MIT.

46. **Peter Norvig.*** Peter Norvig is a Director of Research at Google; previously he directed Google's core search algorithms group. He is co-author of Artificial Intelligence: A Modern Approach, the leading textbook in the field, and co-teacher of an Artificial Intelligence class that signed up 160,000 students, helping to kick off the current round of massive open online classes (MOOCs). He is a fellow of the AAAI, ACM, California Academy of Science and American Academy of Arts and Sciences.

47. **Martin Odersky.** Martin is a professor at EPFL in Lausanne, Switzerland. He is best known as the creator and principal designer of the Scala programming language. Prior to that, he made several contributions to the development of Java. He created the Pizza and GJ languages, designed the original version of generics for Java, and wrote the javac reference compiler for Java. He is a fellow of the ACM.

48. **John Ousterhout.** John Ousterhout is the Bosack Lerner Professor of Computer Science at Stanford University. His current research focuses on new software stack layers to allow data center applications to take advantage of communication and storage technologies with microsecond-scale latencies. Ousterhout's prior positions include fourteen years in industry, where he founded two companies (Scriptics and Electric Cloud), preceded by fourteen years as Professor of Computer Science at U.C. Berkeley. He is the creator of the Tcl scripting language and is also well known for his work in distributed operating systems and storage systems. Ousterhout received a BS degree in Physics from Yale University and a PhD in

Computer Science from Carnegie Mellon University. He is a member of the National Academy of Engineering and has received numerous awards, including the ACM Software System Award, the ACM Grace Murray Hopper Award, the National Science Foundation Presidential Young Investigator Award, and the U.C. Berkeley Distinguished Teaching Award.

49. **Tim Paterson.** Tim began his career designing one of the first 16-bit microcomputer systems at Seattle Computer Products. He then wrote an operating system for that computer, which was later sold to Microsoft and became widely used as MS-DOS. He went on to found his own company, Falcon Technology, whose primary products were hard disk systems for personal computers. He moved on to Microsoft where he was a software engineer for many years, working on such products as QuickBASIC, Visual Basic, VBScript, and Visual J++ (Java). After his retirement in the late '90s he has continued developing software and microcontroller-based hardware projects as a hobby and part-time small business. He has been granted three U.S. patents on software methods.

50. **David Patterson.*** David Patterson joined UC Berkeley in 1977. He has been Director of the Par Lab, Chair of UC Berkeley's CS Division, Chair of the Computing Research Association, and President of the Association for Computing Machinery. His most successful projects have been Reduced Instruction Set Computers (RISC), Redundant Arrays of Inexpensive Disks (RAID), and Network of Workstations. All helped lead to multibillion-dollar industries. This research led

to many papers, six books, and about 35 honors, including election to the National Academy of Engineering, the National Academy of Sciences, the Silicon Valley Engineering Hall of Fame, and Fellow of the Computer History Museum. He shared the IEEE von Neumann Medal, the NEC C&C Prize,and the ACM Turing Award (the highest prize in computing) with John Hennessy, former President of Stanford University and co-author of two of his books.

51. **Alex Payne.** Alex Payne consults, advises, and invests in early-stage technology startups. As Platform Lead at Twitter he managed one of the web's most popular APIs. He was subsequently co-founder and Chief Technology Officer of online banking service Simple, acquired by BBVA in 2014. Alex organizes an annual conference showcasing advances in programming languages and has co-authored a book on the Scala programming language (O'Reilly, 2009). He is a regular speaker at technology and business conferences worldwide and has lectured on API design at Stanford.

52. **Tim Peierls.** Since receiving a BS in Computer Science from Yale in 1983 and an MS in CS from Cornell in 1986, Tim has continuously worked in the software industry, first at Bell Labs (airline crew scheduling), then co-founding the Lightstone Group in 1990 (aircraft scheduling, delivery vehicle routing and scheduling, sold to Descartes Systems Group in 1998) and Seat Yourself in 2002 (online ticketing for school performing arts groups). For the last fifteen years, almost all of his programming work has been in Java. He has served on the Expert Groups of several Java Specification Requests (166, 201, 330,

334) and on the SE/EE Executive Committee of the Java Community Process; he co-authored a book, Java Concurrency in Practice; and he contributes code, support, and advice to various open source projects, including Restlet, Hazelcast, and JClouds.

53. **Ronald L. Rivest.** Ronald L. Rivest is an MIT Institute Professor in the Electrical Engineering and Computer Science Department. He is well-known as a co-inventor of the RSA public-key cryptosystem, for which he received the ACM Turing Award. He is a co-author of the widely-used textbook Introduction to Algorithms. His current research interest is voting systems and election integrity.

54. **Aviel D. Rubin.** Dr. Aviel "Avi" D. Rubin is Professor of Computer Science and Technical Director of the Information Security Institute at Johns Hopkins University. He is also the Director of the JHU Health and Medical Security Lab. Prior to joining Hopkins, Rubin was a research scientist at AT&T Labs. His is also the founder of Harbor Labs, a cybersecurity company. Rubin testified about information security before the U.S. House and Senate on multiple occasions, and he is the author of several books about computer security. Rubin is a frequent keynote speaker at industry and academic conferences, and he delivered a widely viewed TED talk in 2011 and another TED talk in September, 2015. He also testified in federal court as an expert witness on numerous occasions in matters relating to high tech litigation. Rubin served as Associate Editor of IEEE Transactions on Information Forensics and Security, Associate Editor of Communications of the ACM (CACM), and an Advisory Board member of Springer's

Information Security and Cryptography Book Series. In 2010-2011 Rubin was a Fulbright Scholar at Tel Aviv University. In January, 2004 Baltimore Magazine named Rubin a Baltimorean of the Year for his work in safeguarding the integrity of our election process, and he is also the recipient of the 2004 Electronic Frontiers Foundation Pioneer Award. Rubin has a B.S, ('89), M.S.E ('91), and Ph.D. ('94) from the University of Michigan.

55. **Curtis Schroeder.** Curtis is a Hardware-in-the-Loop Simulation Engineer at Draper. He served as the Drafting Group Editor for the Simulation Interoperability Standards Organization (SISO) Common Image Generator Interface (CIGI) 4.0 international standard. The success of SISO international standards depends upon implementation of said copyrighted standards by numerous simulation vendors and end-users, including NATO. Previously, Curtis has worked for Antycip Simulation in the UK and the Lockheed Martin Aeronautics Company, where he utilized a number of open standards in projects he was involved in. He earned B.S. & M.S. Computer Science degrees at the Missouri University of Science & Technology.

56. **Robert Sedgewick.** Robert Sedgewick is the founding chair and the William O. Baker Professor in the Department of Computer Science at Princeton and served for many years as a member of the board of directors of Adobe Systems. He has over fifty years of experience working with software systems. He has held visiting research positions at Xerox PARC, Palo Alto, CA; Institute for Defense Analyses, Princeton, NJ; and INRIA, Rocquencourt,

France. He regularly serves on journal editorial boards and organizing program committees of conferences and workshops on data structures and the analysis of algorithms held throughout the world. Professor Sedgewick's research interests include analytic combinatorics, algorithm design, the scientific analysis of algorithms, curriculum development, and innovations in the dissemination of knowledge. He has published widely in these areas and is the author of twenty books, including a series of books on algorithms that have been bestsellers for four decades and have sold nearly one million copies. He has also published extensive online content (including studio-produced video lectures) on analysis of algorithms and analytic combinatorics and (with Kevin Wayne) algorithms and computer science. Their massive open online course (MOOC) on algorithms has been named one of the "top 10 MOOCs of all time."

57. **Mary Shaw.** Mary Shaw is the Alan J. Perlis University Professor of Computer Science in the Institute for Software Research at Carnegie Mellon University. Her research focuses on software engineering and software design, particularly software architecture and design of systems used by real people. She has made fundamental and significant contributions to an engineering discipline for software through developing data abstraction with verification, establishing software architecture as a major branch of software engineering, designing influential and innovative curricula in software engineering and computer science supported by two influential textbooks, and helping to found the Software Engineering Institute at Carnegie Mellon. She has received

the United States' National Medal of Technology and Innovation, the George R. Stibitz Computer & Communications Pioneer Award, the ACM SIGSOFT Outstanding Research Award, the IEEE Computer Society TCSE's Distinguished Educator and Distinguished Women in Software Engineering Awards, and CSEE&T's Nancy Mead Award for Excellence in Software Engineering Education. She is an elected Life Fellow of the ACM and the IEEE and an elected Fellow of the AAAS. She holds a BA cum laude from Rice and a Ph.D. from Carnegie Mellon.

58. **Barbara Simons.** Barbara Simons is a former President of the Association for Computing Machinery (ACM), the nation's oldest and largest educational and scientific society for computing professionals. She is the only woman to have received the Distinguished Engineering Alumni Award from the College of Engineering of U.C. Berkeley, where she earned her Ph.D. in computer science. A fellow of ACM and the American Association for the Advancement of Science, she also received the Computing Research Association Distinguished Service Award and the Electronic Frontier Foundation Pioneer Award. She has published Broken Ballots: Will Your Vote Count?, a book on voting machines co-authored with Douglas Jones. She has been on the Board of Advisors of the U.S. Election Assistance Commission since 2008, and she co-authored the report that led to the cancellation of Department of Defense's Internet voting project (SERVE) in 2004 because of security concerns. She was a member of the National Workshop on Internet Voting, convened by President Clinton, which conducted one of the first studies of

Internet Voting and produced a report in 2001. She is Board Chair of Verified Voting and is retired from IBM Research.

59. **Daniel Sleator.** Daniel Sleator is a professor of computer science at Carnegie Mellon university. Prior to joining CMU, Sleator was a member of technical staff at AT&T Bell Laboratories. He is the joint winner (with Bob Tarjan) of the 1999 ACM Paris Kanellakis Theory and Practice Award. This was awarded for the invention of the splay tree. His primary area of research is algorithm and data structure design and analysis. He has also built systems for parsing English, and doing automated music analysis. In 1995 he founded the Internet Chess Club.

60. **Alfred Z. Spector.** Alfred Spector is Chief Technology Officer and Head of Engineering at Two Sigma, a firm dedicated to using information to optimize diverse economic challenges. Prior to joining Two Sigma, Dr. Spector spent nearly eight years as Vice President of Research and Special Initiatives, at Google, where his teams delivered a range of successful technologies including machine learning, speech recognition, and translation. Prior to Google, Dr. Spector held various senior-level positions at IBM, including Vice President of Strategy and Technology (or CTO) for IBM Software and Vice President of Services and Software research across the company. He previously founded and served as CEO of Transarc Corporation, a pioneer in distributed transaction processing and wide-area file systems, and he was a professor of computer science at Carnegie Mellon University. Dr. Spector received a bachelor's degree in Applied

Mathematics from Harvard University and a Ph.D. in computer science from Stanford University. He is a Fellow of both the Association for Computing Machinery and the IEEE. He is an active member of the National Academy of Engineering and the American Academy of Arts and Sciences, where he serves on the Council. Dr. Spector won the IEEE Kanai Award for Distributed Computing in 2001 and the ACM Software Systems Award for his work on the Andrew File System (AFS) in 2016.

61. **Michael Stonebraker.** Dr. Stonebraker has been a pioneer of data base research and technology for more than 40 years. He was the main architect of the INGRES relational DBMS, and the object-relational DBMS, POSTGRES. These prototypes were developed at the University of California at Berkeley where Stonebraker was a Professor of Computer Science for twenty-five years. More recently at M.I.T. he was a co-architect of the Aurora/Borealis stream processing engine, the C-Store column-oriented DBMS, the H-Store transaction processing engine, the SciDB array DBMS, and the Data Tamer data curation system. Presently he serves as Chief Technology Officer of Paradigm4 and Tamr, Inc. Professor Stonebraker was awarded the ACM System Software Award in 1992 for his work on INGRES. Additionally, he was awarded the first annual SIGMOD Innovation award in 1994, and was elected to the National Academy of Engineering in 1997. He was awarded the IEEE John Von Neumann award in 2005 and the 2014 Turing Award, and is presently an Adjunct Professor of Computer Science at M.I.T, where he is co-director of the Intel Science and Technology Center focused on big data.

62. **Bjarne Stroustrup.** Bjarne Stroustrup is the inventor of the C++ programming language. He wrote the standard textbook on the language and its original implementation, The C++ Programming Language, and many other academic and popular books and articles. He has served on the ISO Standards committee since its creation in 1989. He is a fellow of the ACM, the IEEE and the CHM. He was elected member of the National Academy of Engineering and is a recipient of the NAE's highest award, The Charles S. Draper Prize. He holds a masters degree in mathematics and computer science from Aarhus University, in Denmark, and a Ph.D. in computer science from the University of Cambridge, where he is an honorary fellow of Churchill College.

63. **Gerald Jay Sussman.** Gerald Jay Sussman is the Panasonic (formerly Matsushita) Professor of Electrical Engineering at the Massachusetts Institute of Technology. He has been involved in artificial intelligence research at M.I.T. since 1964. His research has centered on understanding the problem-solving strategies used by scientists and engineers, with the goals of automating parts of the process and formalizing it to provide more effective methods of science and engineering education. Sussman has also worked in computer languages, in computer architecture, and in VLSI design. Sussman is a coauthor of the introductory computer science textbook that included innovative advances in curricula designed for students pursuing different kinds of computing expertise, which has had a worldwide impact on university computer-science education. Sussman has received numerous awards and recognitions including: the ACM's Karl

Karlstrom Outstanding Educator Award, the Amar G. Bose award for teaching, a fellow of the Institute of Electrical and Electronics Engineers, a fellow of the American Academy of Arts and Sciences, a member of the National Academy of Engineering, and a fellow of the American Association for the Advancement of Science. He received the S.B. and the Ph.D. degrees in mathematics from the Massachusetts Institute of Technology in 1968 and 1973.

64. **Ivan E. Sutherland.** Ivan E. Sutherland received his B.S. degree from the Carnegie Institute of Technology, his M.S. degree from the California Institute of Technology, and his Ph.D. degree from the Massachusetts Institute of Technology, all in electrical engineering. He holds honorary degrees from Harvard University, the University of North Carolina, and the University of Utah. He joined Sun in 1990 as a Sun Fellow, Sun's highest technical rank. He joined Portland State University in 2009 to found the Asynchronous Research Center. He leads a small group working on self-timed VLSI systems; his group develops self-timed circuit methodologies and design techniques for fast CMOS circuits and applies them to new hardware architectures. His book, Logical Effort (1999) with joint authors Sproull and Harris, describes the mathematics of designing fast circuits. His 1963 MIT Ph.D., Sketchpad, is widely known, and he has been called the "father of computer graphics." He is author of more than seventy patents, as well as numerous publications and lectures. Dr. Sutherland holds the 1988 ACM Turing Award, the 2012 Kyoto Prize and the IEEE Von Neumann Award. He is a Fellow of the ACM and a member of both the National

Academy of Engineering and the National Academy of Sciences.

65. **Andrew Tanenbaum.** Andrew S. Tanenbaum has an S.B. degree from M.I.T. and a Ph.D. from the University of California. He is a professor emeritus at the Vrije Universiteit in Amsterdam. Tanenbaum is the principal designer of three operating systems: TSS-11, Amoeba, and MINIX, as well as a considerable amount of other open-source software. In addition, Tanenbaum is the author or coauthor of five books, which together have been translated in more than 20 languages and over 175 editions. Tanenbaum has lectured on a variety of topics all over the world. He has been keynote speaker at 40 conferences and has given talks at over one hundred universities and companies in fifteen countries all over North America, Europe, Asia, and Australia. In 2004, Tanenbaum became an Academy Professor of the Royal Netherlands Academy of Arts and Sciences. In 2008, he received a prestigious European Research Council Advanced Grant. Tanenbaum is a Fellow of the ACM, a Fellow of the IEEE, and a member of the Royal Netherlands Academy of Arts and Sciences. In 1994 he was the recipient of the ACM Karl V. Karlstrom Outstanding Educator Award. In 1997 he won the ACM SIGCSE Award for Outstanding Contributions to Computer Science. In 2007 he won the IEEE James H. Mulligan, Jr., Education Medal. In 2008 he won the USENIX Lifetime Achievement Award and in 2015 he won the inaugural Eurosys Lifetime Achievement Award. He has also won numerous other awards, some of which are on his Wikipedia page. He has two honorary doctorates.

66. **Brad Templeton.** Brad Templeton, active in the computer network community since 1979, was founder and publisher at ClariNet Communications Corp., the electronic newspaper that was perhaps the earliest dot-com company. He participated in the building and growth of USENET from its earliest days, and in 1987 founded and edited rec.humor.funny, for many years the world's most widely read electronic publication. He was the first employee of Personal Software/Visicorp, the first major microcomputer applications software company. He later founded Looking Glass Software and over the years was author of a dozen packaged microcomputer software products, including VisiPlot for the IBM-PC, various games, popular tools and utilities for Commodore computers, special Pascal and Basic programming environments designed for education (ALICE), an add-in spreadsheet compiler for Lotus 1-2-3 (3-2-1 Blastoff), and various network related software tools. He currently is track chair for computing and networks at Singularity University, a consultant and speaker on self-driving cars, and is on the board of the Electronic Frontier Foundation and the Foresight Nanotech Institute. He is Chairman Emeritus of the Electronic Frontier Foundation.

67. **Ken Thompson.*** Ken Thompson spent much of his career at Bell Laboratories where he co-designed and implemented the original Unix operating system, invented the B programming language that was a precursor to the C programming language, invented the Bon programming language, co-developed the Plan 9 operating systems, developed the CTSS version of the editor QED, developed ed, which is the standard text editor on

Unix, and the definition of the UTF-8 encoding, which is used for more than half of all Web pages. Thompson also co-developed the software and hardware for Belle, which was the first computer built for the sole purpose of chess playing, and it officially became the first master-level machine in 1983. He is currently a Google Advisor and was formerly a Distinguished Engineer at Google, where he invented new programming languages (including the Go programming language as a co-inventor), among other projects. Thompson is a recipient of numerous awards and commendations in connection with his work on Unix, including the IEEE Emanuel R. Piore Award (1982), the Turing Award (1983), the IEEE Richard W. Hamming Medal (1990), the National Medal of Technology (1999), and the Japan Prize (2011). He is a member of the National Academy of Sciences and the National Academy of Engineering. Thompson holds a B.S. and an M.S., both in Electrical Engineering and Computer Science, from the University of California, Berkeley. He has been awarded two honorary Ph.D degrees.

68. **Jeffrey Ullman.** Jeffrey Ullman is the Stanford W. Ascherman Professor of Engineering (Emeritus) in the Department of Computer Science at Stanford and CEO of Gradiance Corp. He received a B.S. degree from Columbia University in 1963 and a Ph.D. from Princeton in 1966. Prior to his appointment at Stanford in 1979, he was a member of the technical staff of Bell Laboratories from 1966-1969, and on the faculty of Princeton University between 1969-1979. From 1990-1994, he was chair of the Stanford Computer Science Department. Ullman was elected to the National Academy of Engineering in 1989, the American

Academy of Arts and Sciences in 2012, and has held Guggenheim and Einstein Fellowships. He has received the Sigmod Contributions Award (1996), the ACM Karl V. Karlstrom Outstanding Educator Award (1998), the Knuth Prize (2000), the Sigmod E. F. Codd Innovations award (2006), the IEEE von Neumann medal (2010), and the NEC C&C Prize (2017). He is the author of 16 books, including books on database systems, compilers, automata theory, and algorithms.

69. **Leslie Valiant.** Leslie Valiant is the T. Jefferson Coolidge Professor of Computer Science and Applied Mathematics in the School of Engineering and Applied Sciences at Harvard University. He has been a founding contributor to the theory of machine learning, devised the bulk synchronous model of parallel computation, invented randomized communication methods for data centers, and developed fundamental theories of the inherent limits of computational feasibility. He is the recipient of the Nevanlinna Prize from the International Mathematical Union, and of the Turing Award from the ACM. He is a Fellow of the Royal Society and a member of the National Academy of Sciences.

70. **Andries van Dam.** Andries van Dam is a Professor of Computer Science at Brown University, and has served on Brown's Computer Science faculty since 1965. He was also Brown's first Vice President of Research from 2002 to 2006. He is the author of the widely used reference books Computer Graphics: Principles and Practice and Object-Oriented Programming Java: A Graphical Approach. In 1967, Andries co-founded ACM

SICGRAPH, the precursor to SIGGRAPH. Andries is an IEEE Fellow, an ACM Fellow, and is a member of the National Academy of Engineering and the American Academy of Arts and Sciences. Andries has won multiple awards, including the Information Display's Special Recognition Award (1974), the IEEE Centennial Medal (1984), the National Computer Graphics Association's Academic Award (1990), the ACM SIGGRAPH Steven A. Coons Award (1991), the L. Herbert Ballou University Professor Chair (1992), the ACM Karl V. Karlstrom Outstanding Educator Award (1994), the Thomas J. Watson, Jr. University Professor of Technology and Education Chair (1995), the IEEE James H. Mulligan, Jr. Education Medal (1999), and the ACM SIGCSE Award for Outstanding Contributions to Computer Science Education (2000). Andries received a B.S. with honors in Engineering Science form Swarthmore College, a M.S. and Ph.D. from the University of Pennsylvania, and holds honorary Ph.D. degrees from Darmstadt Technical University, University of Waterloo, ETH Zurich, and Swarthmore College.

71. **Guido van Rossum.** Guido van Rossum created the open-source programming language Python, and is its lead developer and thought leader. Python is widely used in industry, and is the most popular introductory teaching language at top U.S. universities. Guido developed the Python language while at CWI in Amsterdam. After moving to the United States, he worked as a guest researcher at NIST, at CNRI, and at several start-up companies. He became a Senior Staff Engineer at Google, and is currently a principal engineer at Dropbox. Guido is an ACM Distinguished Engineer, a Fellow of

the Computer History Museum, and a recipient of several awards including the USENIX STUG Award, the NLUUG Award, the Free Software Foundation Award, and the Dr. Dobb's Journal 1999 Excellence in Programming Award. In 2013, Python was awarded the Dutch National ICT COMMIT Award. Guido holds an M.S. in Mathematics and Computer Science from the University of Amsterdam.

72. **John Villasenor.** John Villasenor is on the faculty at UCLA, where he is a professor of electrical engineering, public policy, and management, as well as a visiting professor of law. He is also a nonresident senior fellow at the Brookings Institution and a visiting fellow at the Hoover Institution. Professor Villasenor's research considers communications and information technologies and their broader ramifications, and has addressed topics including cybersecurity, autonomous vehicles, and digital media policy. Professor Villasenor is a member of the Council on Foreign Relations and a former vice chair of the World Economic Forum's Global Agenda Council on the Intellectual Property System. He holds an M.S. and Ph.D. in electrical engineering from Stanford University, and a B.S. in electrical engineering from the University of Virginia. Professor Villasenor has previously served as, though is not currently serving as, a consultant to Google in relation to the *Oracle v. Google* matter.

73. **Jan Vitek.** Jan Vitek is a Professor of Computer Science at Northeastern University. He is the past Chair of the ACM Special Interest Group on Programming Languages (SIGPLAN), the vice chair of AITO and of the IFIP WG 2.4, and

is Chief Scientist at Fiji Systems. He holds a Ph.D. from the University of Geneva and an MSc from the University of Victoria. He works on various aspects of programming languages including virtual machines, compilers, software engineering, real-time and embedded computing, concurrency and information security. Professor Vitek led the Ovm project which resulted in the first successful flight test of real-time Java virtual machine. With Noble and Potter, Vitek proposed the notion of ownership for alias control, which became known as ownership types. He chaired PLDI, ISMM and LCTES and was program chair of ESOP, ECOOP, VEE, Coordination, and TOOLS.

74. **Philip Wadler.** Philip Wadler is a Professor of Theoretical Computer Science at the University of Edinburgh and Senior Research Fellow at IOHK. He is an ACM Fellow and a Fellow of the Royal Society of Edinburgh, past chair of ACM SIGPLAN, past holder of a Royal Society-Wolfson Research Merit Fellowship, winner of the SIGPLAN Distinguished Service Award, and a winner of the POPL Most Influential Paper Award. Previously, he worked or studied at Stanford, Xerox Parc, CMU, Oxford, Chalmers, Glasgow, Bell Labs, and Avaya Labs, and visited as a guest professor in Copenhagen, Sydney, and Paris. He has an h-index of 66 with more than 22,000 citations to his work, according to Google Scholar. He contributed to the designs of Haskell, Java, and XQuery, and is a co-author of Introduction to Functional Programming (Prentice Hall, 1988), XQuery from the Experts (Addison Wesley, 2004) and Generics and Collections in Java (O'Reilly, 2006). He has delivered invited talks in locations ranging from Aizu to Zurich.

75. **James H. Waldo.** James "Jim" Waldo is the Gordon McKay Professor of the Practice of Computer Science in the School of Engineering and Applied Sciences at Harvard, where he is also the Chief Technology Officer. Jim is also a professor of technology policy at the Harvard Kennedy School. Previously, Jim designed clouds at VMware, and was a Distinguished Engineer at Sun Microsystems, where he investigated next-generation large-scale distributed systems. He was the lead architect for Jini, a distributed programming system based on Java. Before joining Sun, Jim spent eight years at Apollo Computer and Hewlett Packard, working in the areas of distributed object systems, user interfaces, class libraries, text and internationalization. While at HP, he led the design and development of the first Object Request Broker, and was instrumental in getting that technology incorporated into the first OMG CORBA specification. Jim edited the book The Evolution of C++: Language Design in the Marketplace of Ideas (MIT Press), co-edited Engaging Privacy and Information Technology in a Digital Age (National Academies Press), and was one of the authors of The Jini Specification (Addison Wesley). More recently, he authored Java: The Good Parts. He is currently a member of the editorial boards of Queue magazine and Communications of the ACM. He holds over fifty patents. Jim received his Ph.D. from the University of Massachusetts (Amherst). He holds two M.A. degrees from the University of Utah.

76. **Dan Wallach.** Dan Wallach is a professor in the Department of Computer Science and a Rice Scholar at the Baker Institute for Public Policy at

Rice University in Houston, Texas. His research considers a variety of different computer security topics, ranging from web browsers and servers through Java security, electronic voting technologies, and smartphones. Wallach is a former member of the Air Force Science Advisory Board and a former member of the USENIX Association Board of Directors.

77. **Steve Wozniak.** Steve Wozniak co-founded Apple and invented the Apple I and Apple II computers. He holds a B.S. in Electrical Engineering and Computer Science from UC Berkeley, and honorary doctorates from twelve universities. Wozniak is Innovator in Residence at High Point University. He founded many companies including CL 9, which brought the first programmable universal remote control to market in 1987, Wheels of Zeus (WOZ), and Acquicor Technology. He was Chief Scientist at Fusion-io and at Primary Data. He designed calculators for Hewlett-Packard and taught computer science to elementary school students and their teachers. Wozniak won numerous awards including the ACM Grace Murray Hopper Award, the National Medal of Technology (with Steve Jobs), the IEEE Hoover Medal, the Heinz Award for Technology, the American Humanist Association Isaac Asimov Science Award, the Global Award of the President of Armenia for Outstanding Contribution to Humanity Through IT, the Young Presidents' Organization Lifetime Achievement Award, the Cal Alumni Association Alumnus of the Year Award, and the Legacy for Children Award from the Children's Discovery Museum in San Jose. He was named a Fellow of the Computer History Museum "for co-founding

Apple Computer and inventing the Apple I personal computer," and inducted into the National Inventors Hall of Fame, the Manufacturing Hall of Fame, and the Consumer Electronics Hall of Fame.

78. **Frank Yellin.** Frank Yellin spent over a decade working on runtime systems for interpreted and compiled languages. As a Staff Engineer in Embedded and Consumer at Sun Microsystems, he was an original member of the Java project. Yellin is co-author of The Java Virtual Machine Specification (Addison-Wesley, 1999), and coauthored the first version of the Java API specification. Yellin worked as a Staff Software Engineer at Google, where he specialized in automatic scalable security testing. Previously he worked at Lucid, where he focused on multitasking, garbage collection, interrupts, and the compilation of Common Lisp. He holds an A.B. in Applied Mathematics from Harvard and an M.S. in Computer Science from Stanford. He is the inventor or co-inventor of sixteen patents.