White Paper

Java Turns 25: Taking Stock of Java's Contributions to Development and Digital Transformation

Sponsored by: Oracle

Arnal Dayaratna
July 2020

IN THIS WHITE PAPER

Java celebrated its quarter century birthday on May 23, 2020. Started by James Gosling in 1995 while Gosling was employed by Sun Microsystems, Java was originally created for interactive television and devices such as cable boxes and televisions. Given that the television industry was too immature to support Java-based technologies in 1995, Sun Microsystems determined that Java's support for internet protocols such as HTTP rendered it more suitable for internet-based applications.

Java provides a framework for the development of distributed applications and can be used to create applications that run either on servers and clients within a network, or on a single computer. Today, Java is used to develop web applications, mobile applications, desktop applications embedded systems, web servers and application servers, enterprise applications and scientific applications.

Java is famous for enabling application portability via the Java Virtual Machine, which provides a runtime environment for executing Java bytecode and supports multiple hardware and software platforms. The "write once, run anywhere" attribute of Java means that organizations can run Java applications on heterogeneous infrastructures and subsequently reap the benefits of increased productivity and reduced costs.

While the write once, run anywhere attribute of the JVM was one of the main drivers for Java's widespread adoption, Java confronts the challenge of remaining relevant given recent innovations in modern application development such as increased use of containers, microservices, CI/CD tooling, automated testing and rapid application development methodologies. This document examines the current state of Java at its 25th birthday under the stewardship of Oracle and reflects on the relevance of Java within a contemporary development landscape characterized by ubiquitous digitization, rapid cloud adoption and organizational interest in high velocity development.

SITUATION OVERVIEW

Key Takeaways

- Java is the most widely utilized language in the industry today, a position that looks like to remain in place for years to come.
- Recent enhancements made to the Java ecosystem, such as GraalVM, makes the language more attractive for cloud native and serverless deployment environments.

- Java's availability as both a commercially supported solution and as an open source software solution broadens the appeal and the developer base that uses the language.

History

In 2006, Sun Microsystems was pivoting to embrace open source software and this included the open sourcing of Java. That led to the release of OpenJDK, an open source implementation of Java Standard Edition (Java SE). OpenJDK includes the virtual machine, the Java Class Library, and the Java Development Kit tools such as the Java compiler. While OpenJDK is the fundamental open source version of Java SE, it is complemented by a bevy of other open source implementations as well as commercial implementations of Java SE. Commercial implementations include Oracle JDK, IBM Java SDK, HP Java SDK, SAP JVM and Amazon Corretto. Oracle JDK remains the most commonly used commercial distribution of Java SE. As of JDK 11, Oracle JDK has converged with OpenJDK such that the two distributions are functionally equivalent.

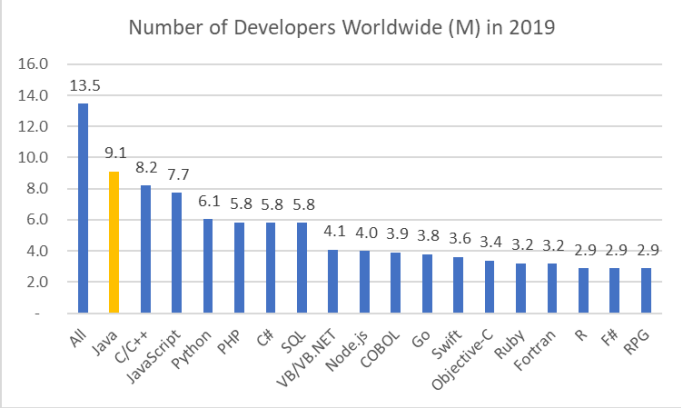
In 2018, Oracle made substantive changes to how it provided commercial license and support to the Oracle JDK by transitioning from a perpetual license to a subscription-based license, and committing to a time-based release schedule for subsequent versions of Java. (For more information on Oracle's current licensing program for Java, see *Understanding the New Java SE Release Cadence and Commercialization Models for Java SE, IDC #US44501118, December 2018.)*

The World's Most Popular Programming Language

Java is the world's most popular programming language amongst developers as shown in Figure 1 below:

FIGURE 1

Number of Developers Worldwide (M) in 2019



Source: IDC, 2020

The chart shows that 9.1 million developers, or 69% of the world's full-time developers use Java, in contrast to 60% for C/C++, which ranks second in popularity. What accounts for the enduring popularity of Java in 2020? Why do more than 9 million developers worldwide use Java?

Some of the reasons why developers love Java include the following:

Platform Independence

The write once, run anywhere attribute of Java undergirded the meteoric adoption of Java in the first two decades of the 21st century. Enterprises discovered that they need not invest in specific hardware to deploy Java applications and hence transitioned entire technology stacks to Java to accommodate heterogeneous IT infrastructures. Meanwhile, developers appreciated Java because they could deploy Java applications on any device as long as it had a Java Runtime Environment installed.

Maturity

Java is known for reliability with respect to the quality of updates to the JDK and the rigor of review and testing that it has received over the years. Java has been refined and improved by developers, architects and DevOps engineers over a period spanning 25 years. The 25 year lifespan of Java marked decades of intense and rapid digitization that included meteoric adoption of internet technologies, the birth of cloud computing and contemporary digital transformation initiatives. Tested and true, Java is associated with enterprise-grade readiness that is derived from being battle-tested by millions of developers over the course of its history.

Performance

Developers associate Java with its performance and ability to scale. The Just in Time (JIT) compiler that is part of the JVM leverages a multitude of optimizations to improve the performance of Java applications. Because the Just in Time compiler compiles an application at runtime, it has the ability to dynamically optimize both the code itself as well as the intersection of the code with the specific hardware on which the application is run.

Object Oriented

As an object-oriented programming language, Java applications are built around the concept of objects. One of the benefits of object-oriented programming languages is that the code is modular and can be reused. In addition, object-oriented programming provides developers with clear guidance about how objects can interact with one another, thereby improving the productivity of developers.

Community

Java boasts one of the most vibrant developer communities in the world. Java Champions are technical experts with strong opinions about the future of Java. Meanwhile, developers can attend virtual or in-person Java User Groups to learn more about how Java is used by their peers. One of the differentiating attributes of the Java community consists in its attention to issues related to the progress of Java and the vociferousness and transparency of the associated discussions about the future of Java. With the robust community, Java has spawned a multitude of enterprise-grade frameworks that streamline and simplify Java development. For example, Spring Boot specializes in the development of microservices architectures while Struts is a framework geared toward web development.

Class Library (Java API)

The Java API is a list of all of the classes that are part of the JDK. These classes provide developers with pre-written functionality that can be reused in the process of development. Instead of manually recreating the wheel, developers can harness the power of Java's massive class library to reuse code and accelerate development.

Time-based release cadence and Transition to Subscription-based Model

In September 2017, Oracle and its ecosystem of Java community partners announced a new, time-based release cadence for Java marked by biannual releases as opposed to periods of two or more years between releases. Java Standard Edition (SE) 10 was the first version of Java characterized by the time-based release cadence in contrast to the feature-based release cadence responsible for Java SE 6, 7, 8, and 9.

One important attribute of the time-based release cadence is the designation of specific releases as long-term support releases. Every three years, Oracle designates a release as a long-term support release, which is a release that receives updates for a longer period of time than the six-month time frame specific to its duration. The first long-term support release was Java SE 11. After the release of Java SE 12, for example, customers who are using Java SE 11 can continue to receive updates by means of a commercial offering called the Oracle Java SE Subscription. The subscription offers organizations the flexibility to move at a pace commensurate with their business needs while protecting their Java estate.

Advantages of a time-based release cadence:

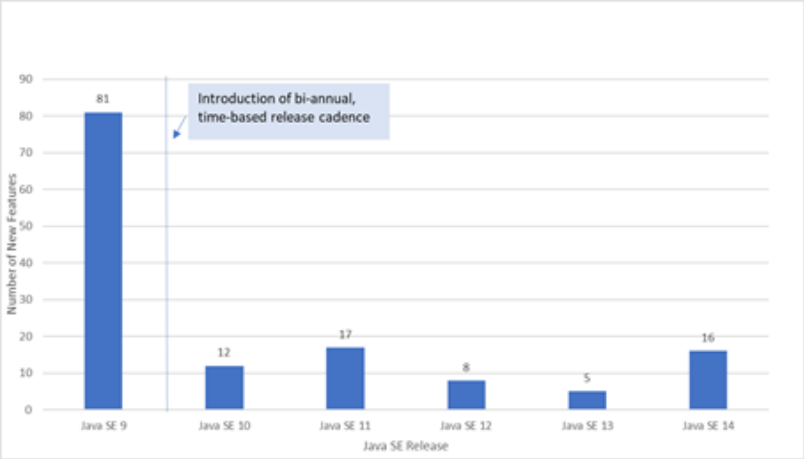
- Accelerated delivery of new features means that developers obtain access to Java SE innovations faster
- Simplified migration from one version of Java SE to another because the difference between consecutive, bi-annual releases is smaller than it was for feature-based releases that were separated by a period of several years
- The decomposition of features into smaller units, as needed, to remain compliant with the bi-annual release cadence. This practice of splitting features into a multitude of smaller features makes the delivery pipeline structured and predictable.
- A time-based release cadence means that customers can reliably expect upgrades of Java SE without encountering delays that arose from the community's commitment to releasing a specific feature with a designated version of Java SE.

Java SE Innovation

The bi-annual, time-based release cadence means that Java developers obtain access to new features and functionality more frequently than they did in comparison to the feature-based releases that transpired roughly every 2-3 years. This enhancement of access to new features in Java SE gives developers the opportunity to test new features and provide feedback about the value to the Java community. In addition, a faster release cadence improves developer engagement with Java and fosters a participatory culture whereby developers are more likely to request additional features and functionality. Meanwhile, the steady release of new features fosters the perception of Java as a cutting-edge, modern development language that has kept pace with the evolution of software development as opposed to a legacy, development language.

Figure 2 below illustrates how the time-based release cadence has transformed the frequency and number of updates to Java SE:

FIGURE 2



Source: Oracle, 2020

Java 9 was the last feature-oriented release and contains over 80 discrete new features.

The main features in Java 9 were from Project Jigsaw, the central component of which was the Java Platform Module System which added the concept of a module to the JVM. A module is a container of packages whereas a package is a container of classes and interfaces. Project Jigsaw enables developers to apply access control to packages within modules such that they cannot be seen by other modules. More generally, Project Jigsaw enhances the ability of Java SE to use a modular architecture to manage a large codebase. The enhanced ability to modularize Java SE renders it easier to create libraries, implement security, improve application performance and update and modernize legacy applications.

Since Java 9, there have been five Java SE releases that have collectively delivered 58 JEPs. Notable enhancements include HTTP Client (JDK 11), Flight Recorder (JDK 11), Text Blocks (JDK 13), Pattern Matching (JDK 13) and Switch Expressions (JDK 14).

Meanwhile, some of the focused initiatives aimed at enhancing Java SE include:

- **Project Valhalla** focuses on reducing the memory and computational overhead associated with storing objects representing small, immutable, identity-less types in memory. This focus on reducing the operational overhead as it relates to these kinds of "inline-types" is particularly important for optimizing Java for modern hardware.
- **Project Loom** innovates with respect to Java's handling of concurrent programming. Using lightweight threads called Fibers, the project aims to make it possible to write scalable, synchronous blocking code.
- **Project Panama** seeks to improve performance and make it easier to develop I/O intensive applications through Java-native platform enhancements.
- **Project Amber** seeks to improve the productivity of Java developers by means of enhancements such as pattern matching, switch expressions and text blocks.

GraalVM

GraalVM, initiated in Oracle Labs, is a project that optimizes Java performance and renders Java far more suitable for cloud native development. The GraalVM infrastructure delivers enhanced performance and operational efficiency to Java applications as well as applications developed in other languages. Built in Java, GraalVM's compiler can be used in Just in Time (JIT) and Ahead of Time (AOT) mode. GraalVM uses Just in Time compilation to generate native code for target deployment environments from platform independent Java Bytecode. Meanwhile, GraalVM uses Ahead of Time compilation to transform JVM-based applications into native executables, which allows the application to be executed without a JVM.

Depending on the application, there are benefits and trade-offs associated with both just in time and ahead of time compilation. Ahead of time compilation typically enables faster startup time, reduced memory footprint and increased deployment density that optimizes the performance of applications for containers and serverless deployments. Meanwhile, GraalVM's just in time compiler boasts a multitude of optimizations that enable faster, more intelligent compilation.

Another important attribute of GraalVM is that it supports the execution of a multitude of languages on the JVM which are compiled to highly performant machine code just like Java. Other languages supported by GraalVM includes JavaScript, Ruby, Python, R, WebAssembly, C/C++ via LLVM bitcode. The shared runtime environment delivered by GraalVM empowers developers to create polyglot applications that leverage the strengths of discrete languages.

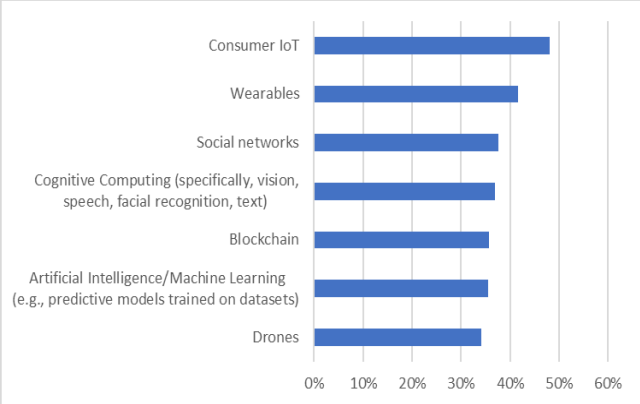
IDC believes that GraalVM's ability to enable code to run faster and more efficiently, in addition to its unique capabilities to facilitate the development of polyglot applications, represent a dramatically important innovation for Java. GraalVM enhances the efficiency with which applications are run and renders developers more productive as a result. Moreover, its ability to optimize applications for containers and microservices positions it to become central to cloud native development practices and methodologies.

Use of Java for Emerging Technologies

While Java has a venerable history of developing web, mobile and desktop applications, it also stands at the forefront of innovation related to emerging technologies for use cases such as AI/ML, IoT, AR/VR and blockchain. In a worldwide survey of 2500 developers, IDC asked about the top five use cases for which they have used Java for emerging technologies as illustrated below (Figure 3):

FIGURE 3

Percent of Full-time Developers Who Use Java



Source: IDC, 2020

Roughly 50% of full-time developers cited IoT as a use case for which they used Java. Meanwhile, wearables, social networks, cognitive computing, blockchain and AI/ML were also use cases for which developers had used Java to create applications over the last 12 months. This data illustrates how Java is used to develop applications for disruptive, cutting-edge use cases that include AI/ML and IoT. Even though Java remains relevant for the development of web and mobile applications, for example, it has also diversified to embrace a broader range of use cases and problems. As such, the chart shows how Java remains poised at the cusp of software-driven innovation and is a key driver of digital transformation initiatives in the enterprise.

ORACLE'S STEWARDSHIP OF JAVA

Oracle had been a key part of the Java ecosystem since its inception in the 1990's. In 2008, with the acquisition of BEA, Oracle even had its own Java Runtime (JRockit). Since the acquisition of Sun Microsystems in 2010, Oracle has led the systematic introduction of innovation to Java as demonstrated by increased support for Lambda functions and serverless computing, enhanced functionality for mobile application development, increased modularity and improved support for microservices and containers.

The most important dimension of Oracle's stewardship of Java SE is its sustained contribution to rendering the OpenJDK ready for production-grade applications. When Oracle acquired SUN, the OpenJDK was not ready to support production-grade, mission critical applications. Oracle led the process of ensuring that the OpenJDK was battle-tested and capable of supporting the needs of enterprises by doubling down on its efforts to strengthen the product.

Oracle reinvigorated OpenJDK starting in 2011 by making the Java 7 Reference Implementation (and subsequent releases) fully open source and 100% buildable from OpenJDK sources. Oracle continues to invest in the stewardship of OpenJDK providing the vast majority of source code for current releases. Oracle also leads collaboration with hundreds of contributors on various ports and projects.

One of the biggest ways in which Oracle has increased innovation for Java SE involves the time-based release cadence that it introduced in 2017. By transitioning to a bi-annual release cadence, Oracle has

energized Java-related development by giving developers a steady stream of enhancements that they can use or evaluate. Some of the features and functionality that are released bi-annually are introduced in preview format, thereby allowing developers to engage with projects that are incubating and provide Oracle with feedback that they can use before the feature is released for general usage.

In addition, Oracle's stewardship of Java has expanded community participation with Java by democratizing the ability of users to submit Java Specification Requests (JSRs) and participate in the Java Community Process (JCP). This means that startups, SMBs and individuals have increased opportunities to submit JSRs alongside contributions made by technology companies and large enterprises. Oracle's stewardship of Java has also democratized the governance framework and process for the Java Community Process by increasing the transparency associated with the review of JCP in addition to the diversity of the reviewers.

Another important dimension of Oracle's stewardship of Java is its support for local and regional Java user groups. These communities provide a physical and digital infrastructure that empowers Java developers to collaborate and continue upskilling themselves. Oracle's leadership has also led to the development of a community that includes Java evangelists and champions that variously lead conversations about the current and future state of Java.

CHALLENGES/OPPORTUNITIES

While Java is the most popular development language as of 2019, it faces strong competition from JavaScript because of the increased prioritization of rich user experience and front-end development on the part of contemporary users. Additionally, Python has been experiencing robust growth because of its versatility, easy to learn syntax and specialization in data science. Strong competition from other languages means that Oracle and the larger Java community would do well to continue delivering rich batches of bi-annual enhancements as a part of its time-based release cadence. Oracle has an opportunity to respond to the challenges posed by other languages by showcasing Java-related innovation thematically in the context of use cases such as cloud native development, application modernization, AI/ML, front-end development, rich user experiences and personalization.

In addition, Oracle would do well to continue expanding its developer community by exploring opportunities to convert non-Java application development to Java-based workloads, where appropriate. To expand its developer community further, Oracle would also benefit from a more granular understanding of how developers think of Java with respect to its strengths and weaknesses for specific use cases. Meanwhile, the rapid growth of developers in emerging markets such as Brazil, Turkey, Indonesia, Kenya, Tanzania and South Africa provides yet another opportunity for Oracle to expand its community of Java developers. IDC research shows that the growth of developer populations worldwide is highest in emerging market geographies and, as such, Oracle has an opportunity to consolidate the leadership of Java by means of targeted developer outreach and evangelism initiatives in these geographies.

CONCLUSION

As Java turns 25, it can justifiably claim to be one of the most influential languages of the last quarter century. Java led the explosive proliferation of web-based applications upon its inception and has since diversified to embrace embedded systems, mobile applications, desktop applications as well as application development for emerging technologies such as AI/ML, IoT, AR/VR and applications for

social networks and blockchain. In recent years, with Oracle’s stewardship, Java has spawned a multitude of innovative technologies that include GraalVM, a project that optimizes Java performance and renders Java more suitable for cloud-native development.

GraalVM is aptly illustrative of Java's ability to inspire new projects that address contemporary business needs and challenges. Over its quarter century history, Java has demonstrated resilience to multiple layers of business and technology-related innovation such as the growth of the internet, adoption of cloud computing, and the use of containers, container orchestration frameworks and microservices. IDC expects Java to continue innovating and adapting to the needs of organizations and developers as the business and technology landscape continues to change.

Synopsis

This IDC whitepaper considers the history and the probable future for the Java language and related/supporting technologies including GraalVM. IDC research confirms that Java is the most widely used language in the industry today, a position that is in the hands of Oracle and the Java community to maintain.

"As Java turns 25, Oracle can proudly reflect on how Java has become the world's most popular programming language," said Arnal Dayaratna, research director, IDC, Software Development. "IDC data reveals that Java remains vibrant and used widely across a multitude of industry verticals and use cases that include AI/ML, IoT, AR/VR and data-driven analytics. In recent years, the continued popularity of Java has been bolstered by Java SE's transition to a bi-annual release cadence that provides a structured timeline for the delivery of new features and innovation. Meanwhile, Oracle's GraalVM initiative has made progress with respect to improving Java performance, rendering Java more suitable for cloud native development and supporting polyglot development. The future remains bright for Java as long as Oracle continues to programmatically improve its relationship with developers and cultivate developer mindshare for projects such as GraalVM."

About IDC

International Data Corporation (IDC) is the premier global provider of market intelligence, advisory services, and events for the information technology, telecommunications and consumer technology markets. IDC helps IT professionals, business executives, and the investment community make fact-based decisions on technology purchases and business strategy. More than 1,100 IDC analysts provide global, regional, and local expertise on technology and industry opportunities and trends in over 110 countries worldwide. For 50 years, IDC has provided strategic insights to help our clients achieve their key business objectives. IDC is a subsidiary of IDG, the world's leading technology media, research, and events company.

Global Headquarters

5 Speen Street
Framingham, MA 01701
USA
508.872.8200
Twitter: @IDC
idc-community.com
www.idc.com