DZone

DZONE'S 2019 GUIDE TO

Java

NEW DEVELOPMENTS AND FEATURES

RESEARCH PARTNER SPOTLIGHT

WhiteSource

Key Research Findings

BY JORDAN BAKER
CONTENT COORDINATOR, DEVADA

Demographics

For this year's DZone Guide to Java survey, we received 804 total responses, with 492 complete responses. Below is the basic demographic breakdown of the respondents.

- Respondents have an average of 16 years of experience in IT.
- Respondents live in four main geographical areas:
  - 36% in Europe
  - 18% in South Central Asia
  - 15% in the USA
  - 12% in South America
- Survey-takers reported working in three main industry verticals:
  - 22% work for software vendors
  - 15% work in finance/banking
  - 12% work in consulting
- A majority of respondents work for enterprise-level organizations.
  - 27% for organizations sized 100-999
  - 21% for organizations sized 10,000+
  - 20% for organizations sized 1,000-9,999
- Three main roles were reported:
  - 42% work as developers/engineers
  - 24% work as developer team leads
  - 19% work as architects
- Respondents reported working on four main types of software projects:
  - 83% develop web applications/services
  - 49% work on enterprise business apps
  - 24% develop native mobile applications
  - 15% work on high-risk software

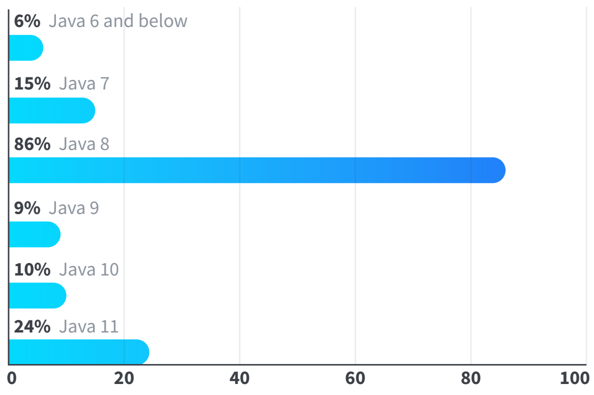
Java 8 Still the Favorite

Despite the fact that it's eight years old, Java 8 remains the most used and preferred version of the Java language. When we asked respondents what versions of Java are being used by their organization to work on existing applications, 83% reported using Java 8, while Java 7 came in a distant second with a 42% adoption rate. When we compare these statistics to our data on the two types of software most respondents develop (web applications and enterprise business applications) as delineated in the Demographics section, we find that Java 8, though dominant among both groups, proves slightly more popular among developers/organizations working on enterprise business apps. Among respondents who reported developing web apps, 82% said they use Java 8 on existing applications, while 87% of those who reported developing enterprise business apps told us they use Java 8 for existing applications.

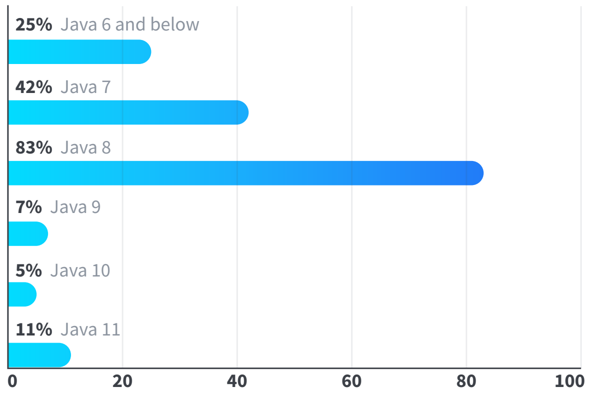
Given Java 8's age, its dominant position in existing applications is less surprising than its widespread use in building new applications. 86% of survey-takers reported using Java 8 to create new applications, while only 24% said they use Java 11, 10% use Java 10, and a mere 9% use Java 9. If we compare our data on Java 8 usage rates against our data on respondents developing

SURVEY RESPONSES

What versions of Java are being used at your organization when building new apps?



What versions of Java are being used in existing apps at your organization?



web and enterprise business applications, we see a similar trend to that reported above. Among respondents working to develop web applications, 87% use Java 8 to create their new apps, with 24% use Java 11, 10% use Java 10, and 10% use Java 9. For those respondents working to develop enterprise business apps, we found that 90% use Java 8 in the building of new applications, while, again, 24% use Java 11, 10% use Java 10, and only 7% use Java 9.

Even with the extreme popularity of Java 8, one would expect the newer versions of the language (i.e. Java 10 and 11 as well as the soon-to-be-released Java 12) to garner some excitement from a developer community that's so interested in Java. Yet, a majority of respondents seemed to be lukewarm in their opinions of the newer versions of Java and the features these versions brought with them. When we asked respondents what they thought constituted the most important new feature introduced in Java 10, well over half (70%) reported that they had no opinion on the matter. We also asked survey-takers which features of Java 11 they were using, and five features of Java 11 (Epislon, ZGC, Flight Recorder, `readString()` and `writeString()`, and the new HTTP client) all had non-adoption rates (i.e. are not being used) of 80% or higher. And finally, when asked about the new features in Java 12 for which they were excited, 53% of respondents reported "none of the above."

Despite this general lack of enthusiasm around the newer versions of Java, over 60% of survey-takers seem excited for the future of the language. 43% told us they feel fairly optimistic about Java's future, and another 29% feel very optimistic.

Development Practices and Coding Paradigms

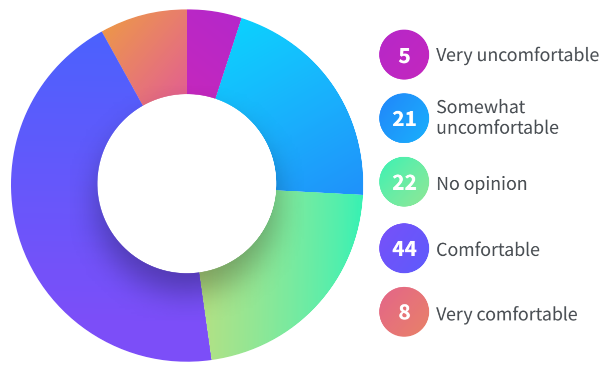
Object-oriented and functional programming represent two of the more popular coding paradigms in use today. This is reflected in that fact that 44% of respondents reported feeling "comfortable" using and mixing these two paradigms and another 8% asserted feeling "very comfortable" with this practice. Interestingly, it appears that respondents' comfort level in mixing object-oriented and functional programming varied by the types of applications they develop. Among those respondents working on web applications, 45% reported feeling comfortable mixing the object-oriented and functional paradigms, while 9% said they were very comfortable. Additionally, 21% of respondents working as web app developers told us there were "somewhat uncomfortable" mixing these two paradigms, while 4% reported being "very uncomfortable." But among those respondents developing enterprise business applications, 35% reported feeling comfortable mixing object-oriented and functional coding paradigms, while 5% reported feeling very comfortable. On the other side of the coin, 28% of these respondents said they feel somewhat uncomfortable in this practice, while 10% feel very uncomfortable. One reason for enterprise business app devs' reluctance to mix coding paradigms could come from the fact they work with massive code bases filled with legacy code that takes advantage of older coding paradigms. Thus, introducing object-oriented or functional paradigms (or a mix of the two) could result in a need for large refactoring projects.

When it comes to development practices specific to the Java language, however, a greater level of parity appears among the responses. When asked how they mix "old" and "new" style Java code, 33% reported using APIs for creating unmodifiable collections, another 33% said they never mix "old" and "new" Java, and 32% said they continue to use the `optional.orElseThrow()` method in their code. Additionally, 30% of respondents told us they mix "old" and "new" style Java while working on the removal of deprecated methods. This widespread mixing of "old" Java standards makes sense given the continued dominance of Java 8.

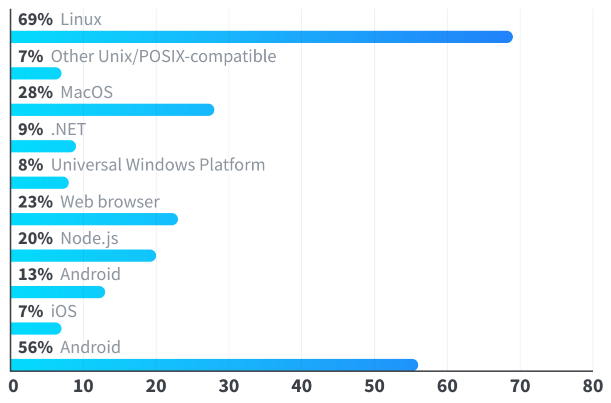
The JDK, no matter the provider, has become an integral part to any Java developer's work. It's interesting to see that the JDK developers use seems to be dependent on the version of Java they're currently working with. We asked respondents if they were

SURVEY RESPONSES

How comfortable are you and your team at mixing object-oriented and functional paradigms?



Which of the following platforms/runtimes/operating systems do you primarily develop on?



planning to stop using Oracle's JDK due to the change in Oracle's JDK long-term support and saw an interesting variance in answers. Among users of Java 10 and 11, the most popular response was, "I am using a different distribution of the JDK." Among respondents using all other versions of Java (i.e. Java 6-9), the most popular responses were, "I am planning to move to a different distribution of the JDK" and "I use Oracle's JDK and have no plans to switch." Thus, while users of every iteration of Java have some interest in exploring non-Oracle SDKs, it seems that those developers using the newer versions of the language have already jumped the preverbal Oracle SDK ship.

Java Adjacent

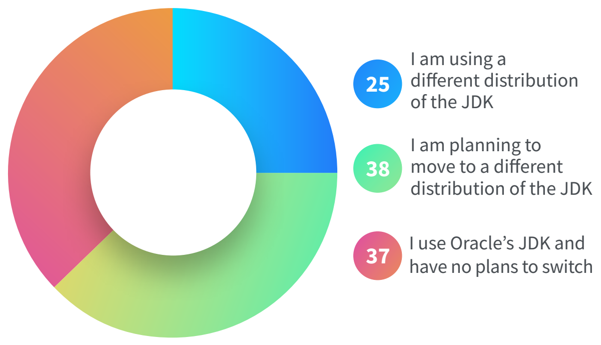
Despite all the features we've discussed so far, Java and the applications built upon it cannot operate in a vacuum. Thus, in this section, we'll focus on how developers use Java alongside other important technologies. To start off, we found that respondents had two main platforms/runtimes/operating systems on which they primarily develop. 69% of respondents told us they typically develop on Linux-based operating systems, while 56% reported using Win32/64. When we compare this data to our data on web application and enterprise business app developers, we find that these two systems remained, far and away, the most popular platforms/runtimes/operating systems to develop on. 72% of web application developers reported using Linux and 54% use Win32/64, while 69% of enterprise business app developers develop on Linux and 63% use Win32/64. Interestingly, when we asked respondents which of these platforms/runtimes/operating systems they target, we saw a bigger discrepancy between the answers of web app and enterprise business app developers. Among the general survey population, 84% of respondents target Linux and 42% target Win32/64. When we narrow this down to respondents working on web applications, we find that 86% target Linux and 38% target Win32/64; and among respondents working on enterprise business applications, 82% target Linux and 49% target Win32/64.

Not all technologies in the survey witnessed such variance, however. Indeed, when we asked which API quality attributes are most important, we saw almost mirrored results across three respondent groups. Among the general survey population, 74% told us they care about simplicity, 60% said consistency, 60% reported performance/robustness, 58% feel productivity is important, and 54% care about learnability. When we again pare the data down to reflect those respondents working on web applications and enterprise business applications, we see almost the exact same results. Among web developers, the most important API quality attributes were simplicity (74%), consistency (64%), performance/robustness (62%), productivity (59%), and learnability (54%). Among enterprise business developers, the most important API qualities reported were simplicity (75%), performance/robustness (66%), productivity (61%), consistency (59%), and learnability (53%). The only notable variance among all these factors is how performance/robustness appears more important for respondents working on enterprise-level applications.

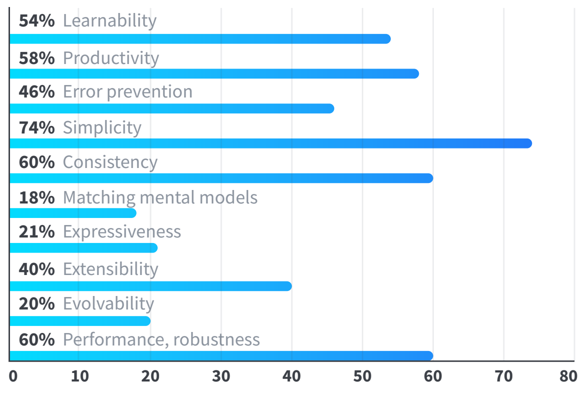
To round out this report, let's examine the use of microservices among Java developers. Of those surveyed, 58% reported using microservices in some capacity during development, while another 18% said they do not, but are planning on adopting microservices. Of note, microservices proved more popular among those respondents working on web applications than those working on enterprise business apps. Of those respondents currently developing web apps, 63% told us they are currently using microservices in some manner, while 18% said they are not using microservices but are planning to in the future. Among respondents currently working on enterprise business applications, we found that 54% use microservices in their development efforts, and 22% are planning on adopting microservices.

SURVEY RESPONSES

Are you planning to stop using Oracle's JDK due to the change in Oracle's JDK long-term support?



Which of the following API quality attributes are most important for you (for any API, not just web APIs)?



Diving Deeper Into Java

Zones

Java dzone.com/java

The largest, most active Java developer community on the web. The Java Zone provides news and tutorials on Java tools, performance tricks, and new standards and strategies that keep your skills razor-sharp.

Microservices dzone.com/microservices

The Microservices Zone will take you through breaking down the monolith step-by-step and designing microservices architecture from scratch. It covers everything from scalability to patterns and anti-patterns. It digs deeper than just containers to give you practical applications and business use cases.

Web Dev dzone.com/webdev

The Web Dev Zone is devoted to all things web development—and that includes everything from front-end user experience to back-end optimization, JavaScript frameworks, and web design. Popular web technology news and releases will be covered alongside mainstay web languages.

Twitter

@trisha_gee

@rafabene

@brunoborges

@dianecraigdavis

@jessitron

@sugrue

@danielbryantuk

@mon_beck

@lunivore

@s1m0nw1

Refcardz

Java Application Vulnerabilities

Java applications, like any other, are susceptible to gaps in security. This Refcard focuses on the top vulnerabilities that can affect Java applications and how to combat them.

Java Containerization

This Refcard focuses on the design, deployment, service discovery, and management of Java applications on the open-source project called Docker so that you can get your Java application up and running inside a Docker-deployed Linux container.

Java API Best Practices

In this Refcard, you'll learn how to make well-documented, consistent, evolvable APIs to help your users make the most of your Java applications.

Books

Effective Java

Learn about the latest language and library features that Java has to offer, get insight into Java subtleties, and see how modern Java supports multiple paradigms.

Java Concurrency in Practice

Dive into basic concepts of concurrency and thread safety, techniques for building and composing thread-safe classes, using concurrency building blocks, and more.

Modern Java in Action

Explore the new features of Java 9 like support for reactive programming and learn how to build on your existing Java skills with the newest techniques.

Podcasts

Java Pub House

Learn about real issues that developers face when programming Java, like O/R setups, threading, troubleshooting, and more.

The Changelog

Through conversations with hackers, leaders, and innovators, learn about OSS code, service meshes, cloud-enabled apps, and more.

Software Defined Talk

Dive into various topics of interest to Java developers, like Kubernetes, serverless, cloud, DevOps, and coding.

Find and Fix Java Open Source Vulnerabilities

Java is consistently one of the most popular languages in use today, providing us with beloved projects including Android SDK, the Spring Framework, and more.

However, despite its continued popularity, Java is witnessing a spike in the number of published vulnerabilities in open source components. WhiteSource's database shows that reported vulnerabilities for open source Java components more than doubled in 2018, rising from 187 in 2017 to 446 in 2018, and posing a higher risk to using them in our products.

The good news here is that the jump in the number of known vulnerabilities is due to the open source community doing a stellar job of finding and reporting these vulnerabilities, sharing their knowledge to help keep us safe.

On the flip side, we know that hackers love known vulnerabilities, scouring the NVD, security advisories, and other publicly available resources for intel on how to exploit these highly reusable open-source software components.

Your time is too valuable to be spent manually finding and fixing vulnerabilities, so why let vulnerability management get in the way? To solve this, there are developer-friendly tools that can integrate into your native coding environment and harness the widest range of security resources in order to keep you up to date on all vulnerabilities. This allows you to speed up remediations with suggested fixes, dramatically reducing the amount of work needed to stay secure, and — most importantly — allowing you to focus on coding.

WRITTEN BY RAMI SASS
CEO & CO-FOUNDER, WHITESOURCE

PARTNER SPOTLIGHT

WhiteSource

Helps you find and fix open source vulnerabilities in your code by fully integrating into your software development lifecycle.

Category Application Security | Software Composition Analysis | DevOps | Secure Coding

New Release Continuous

Open Source? Yes

Case Study Due to the sensitive regulatory environment Siemens Healthineers work in, Siemens H. needed assurance that they are remaining compliant and secure in their use of open source components.

"With open source software, usually the source code is available for all to see, including hackers," explained Code Clinic Lead, Neil Langmead. In order to avoid costly mistakes that can result from vulnerable or risky open source components being added to their products, Siemens Healthineers turned to WhiteSource.

"We chose WhiteSource because of its ease of use, its excellent data, and for the in-depth security vulnerability information that comes with the reporting engine." With WhiteSource, Siemens was offered the widest coverage of plugins and languages that they needed, as well as the continuous monitoring and policy enforcement safeguards they required in order to allow their team to code with confidence. Read more here

Strengths

- **Largest Vulnerabilities Database:** Continuously aggregates information from multiple sources
- **Comprehensive Coverage:** Supports 200+ programming languages and 20+ environments (incl. containers)
- **Pinpoint Accuracy:** Proprietary algorithms guarantee no false positives
- **Automated Workflow:** Enforce policies automatically at all stages of the SDLC
- **Easy Remediation:** Pinpoints vulnerable methods affecting your products

Notable Customers

- Microsoft
- Comcast
- EllieMae
- IGT
- Spotify

Website
whitesourcesoftware.com

Twitter
@WhiteSourceSoft

Blog
resources.whitesourcesoftware.com

WhiteSource

Reported Java Open Source Vulnerabilities More Than Doubled in 2018

Find & Fix

Vulnerabilities in Your Native Environment With **WhiteSource Bolt.**

Get Free Tool