# JVM Ecosystem Report 2020

snyk

# Table of contents

## About your JDK

## About your application

16. Do you use the Spring Framework?
17. What Spring version do you use for your main application?
18. Do you use Enterprise Java? (J2EE, Java EE, Jakarta EE)
19. What Java EE version do you use for your main application?
20. What was your reaction to Oracle and the Eclipse foundation not agreeing on continued usage of the javax namespace?
21. Would you consider switching to another framework/technology in order to avoid migrating to a newer Jakarta EE version, due to the javax namespace changes?
22. What other languages does your application use?
23. Which client-side web frameworks do you use?
24. Which server-side web frameworks do you use?

## About your tools

25. Which is the main Integrated Development Environment (IDE) you are using?
26. Which build tool do you use for your main application?
27. Which CI server do you use?
28. Which code repository do you use for your main application?
29. When do you scan your dependencies for known vulnerabilities?

## About you

30. Where are you from?
31. What is your current role?
32. What is the size of your company?

# Introduction

Welcome to our annual JVM ecosystem report! This report presents the results of the largest annual survey on the JVM ecosystem. The survey was conducted in the second half of 2019 gathering the responses of over 2000 participants. We would like to thank everyone who participated and offered their insights on Java and JVM-related topics.

For this survey, we teamed up with conferences and communities across the JVM ecosystem to reach as many developers as possible. Big shout out to Devoxx, DevNexus, Jfokus, JCrete, Adopt OpenJDK, VirtualJUG and other Java communities for their invaluable help. As a result of this massive effort, an impressive number of developers participated in the survey, giving great insight into the current state of the JVM ecosystem.

Find all demographic information at the end of this report.

Happy reading!

A big thank you to JFokus, Devoxx, JCrete, Java Specialists newsletter, The Developer's Conference, DevNexus, Virtual JUG, Transylvania JUG, Beirut JUG, Manchester Java Community, London Java Community, Timisoara JUG, Utrecht JUG and other user groups!

# TL;DR: report highlights

Before we start, here's a TL;DR overview of the main highlights in this report.

**1 in 3 developers** use the Oracle JDK in production

OpenJDK

**50% of developers** use OpenJDK distributions in production

Java 8

**2 in 3 developers** use Java SE 8 in production

Java 11

**1 in 4 developers** use Java SE 11 in production

Less than **1 in 10 developers** pay for commercial Java support

Less than **1 in 10 developers** is using JPMS today in their application

Kotlin

**1 in 20 developers** use Kotlin in their main application, making it the second most popular language on the JVM

spring

**50% of developers** use Spring Boot

Maven™

**2 in 3 developers** use Maven to build their main project

Jenkins

Almost **6 in 10 developers** use Jenkins in CI

IntelliJ IDEA

Almost **2 in 3 developers** use IntelliJ IDEA

GitLab

**1 in 3 developers** use GitLab, making it the most popular SCM tool among JVM devs

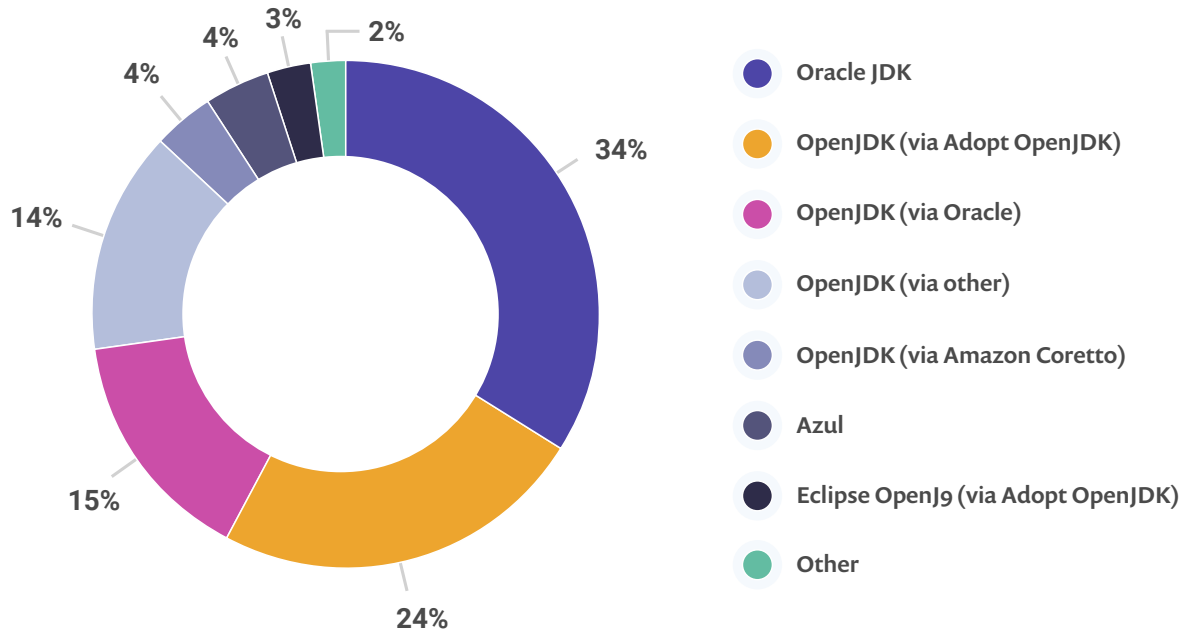
# About your JDK

# 01. Which Java vendor's JDK do you currently use in production for your main application?

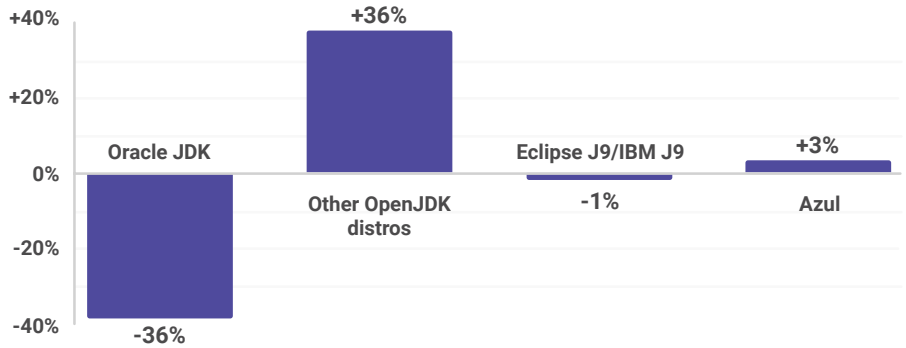
With Oracle changing the licensing on their JDK versions, we kickoff this report with the answer to an important question — which JDK are developers using for their main application?

According to the respondents, although Oracle JDK is still dominant with 34%, there is a huge shift towards other OpenJDK providers. 1 in 4 developers chooses the Adopt OpenJDK distribution. It's also interesting to note that, the Oracle JDK still uses the OpenJDK under the covers, despite carrying a commercial license.

Comparing this to the results from last year, where Oracle JDK accounted for 70% and OpenJDK for 21% of the prefered JDK distribution, we notice a major shift, with a 72% swing from Oracle JDK to alternate OpenJDK providers.

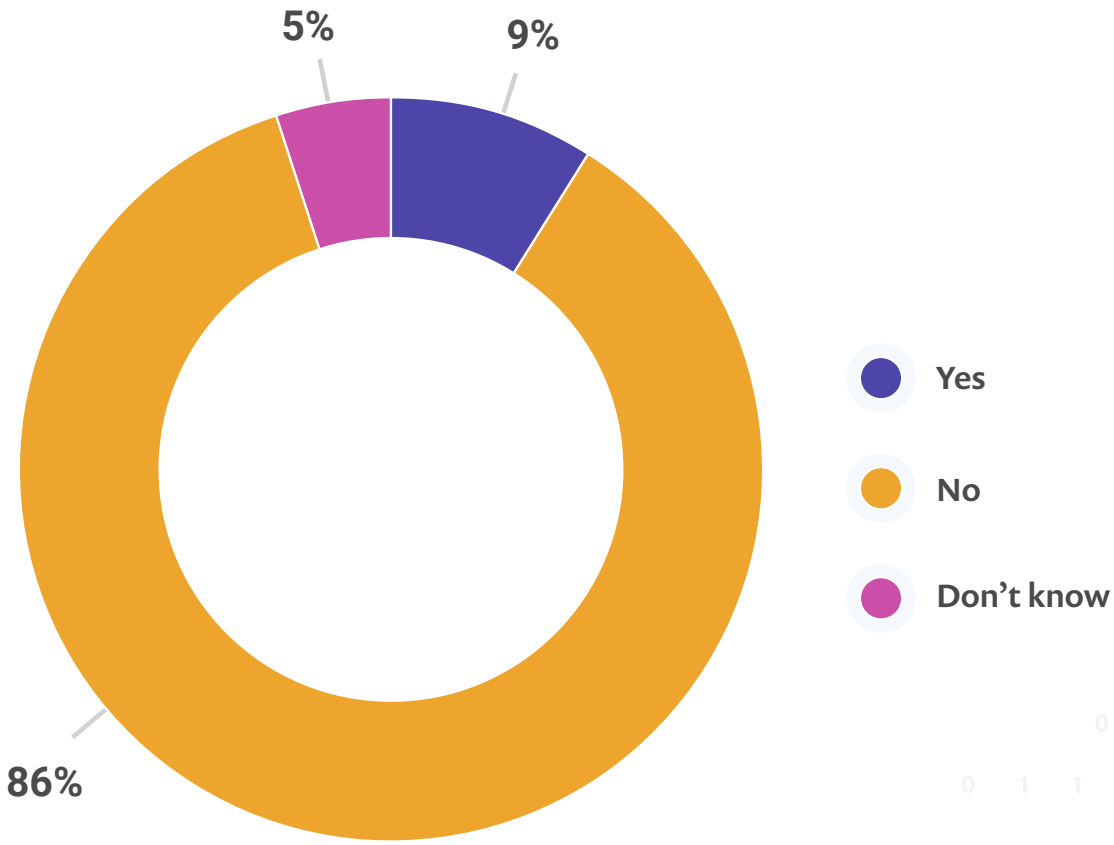



**Difference in share from 2018 to 2019**



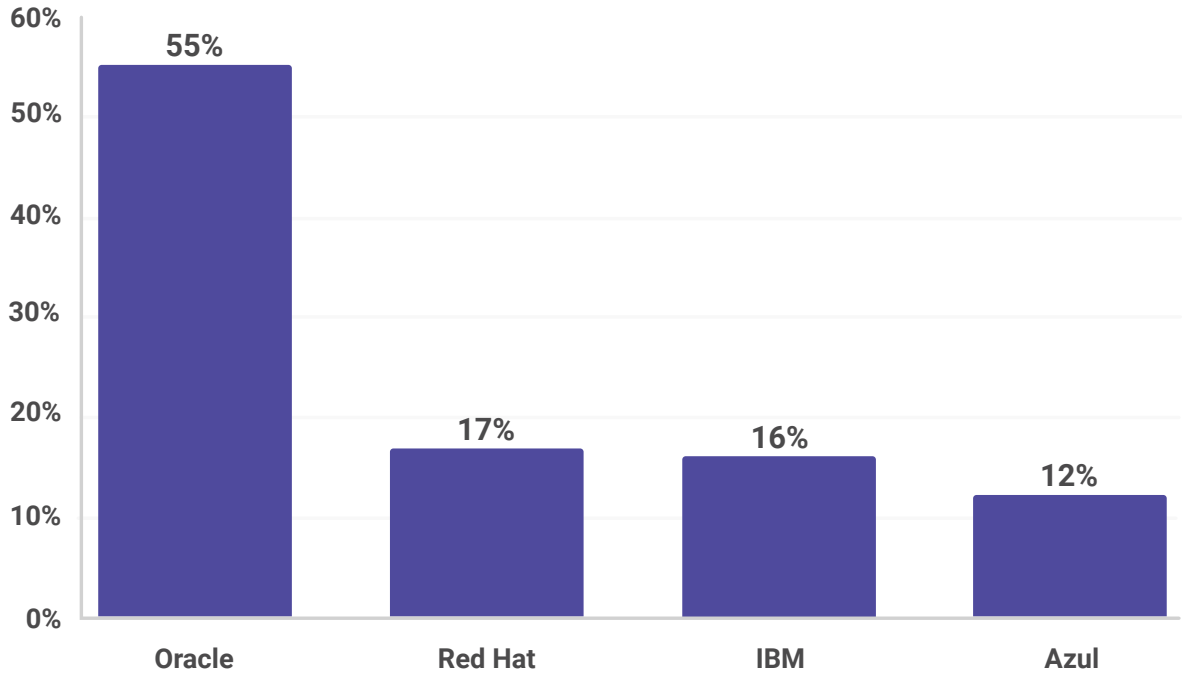
# 02. Are you currently paying a vendor for JDK support?

When we look at the responses to this question, the shift we saw earlier, from Oracle JDK to OpenJDK, makes more sense since the large majority of the participants (86%) do not wish to pay for JDK support. In fact, only 9% currently pay for support.



# 03. Who do you pay?

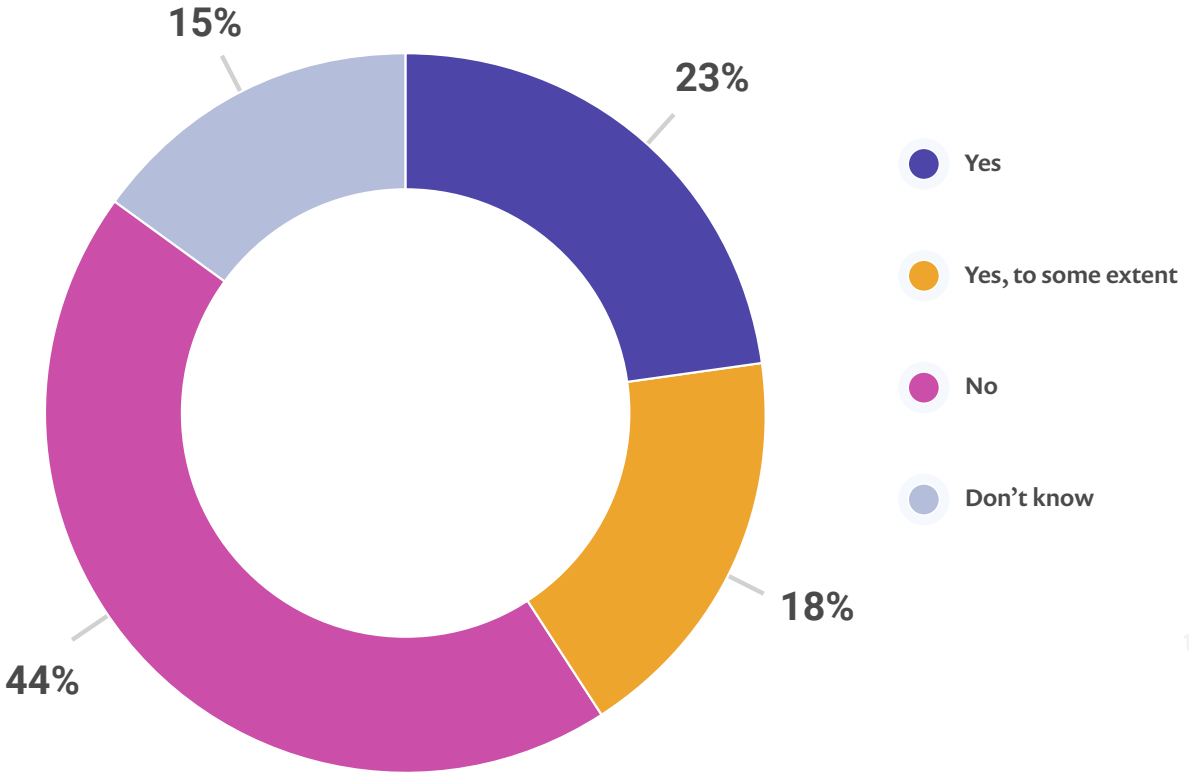
When developers do choose to pay for JDK support, Oracle is still the clear winner while the other three vendors share the remainder of the market rather equally. In retrospect, this means that, if only 9% of developers pay for JDK support (as seen in question 2), the total population of developers that pay Oracle, is 5% – or 1 in 20 developers.



# 04. Did the support and release cadence changes, since JDK 9, affect your decision to pay for support?

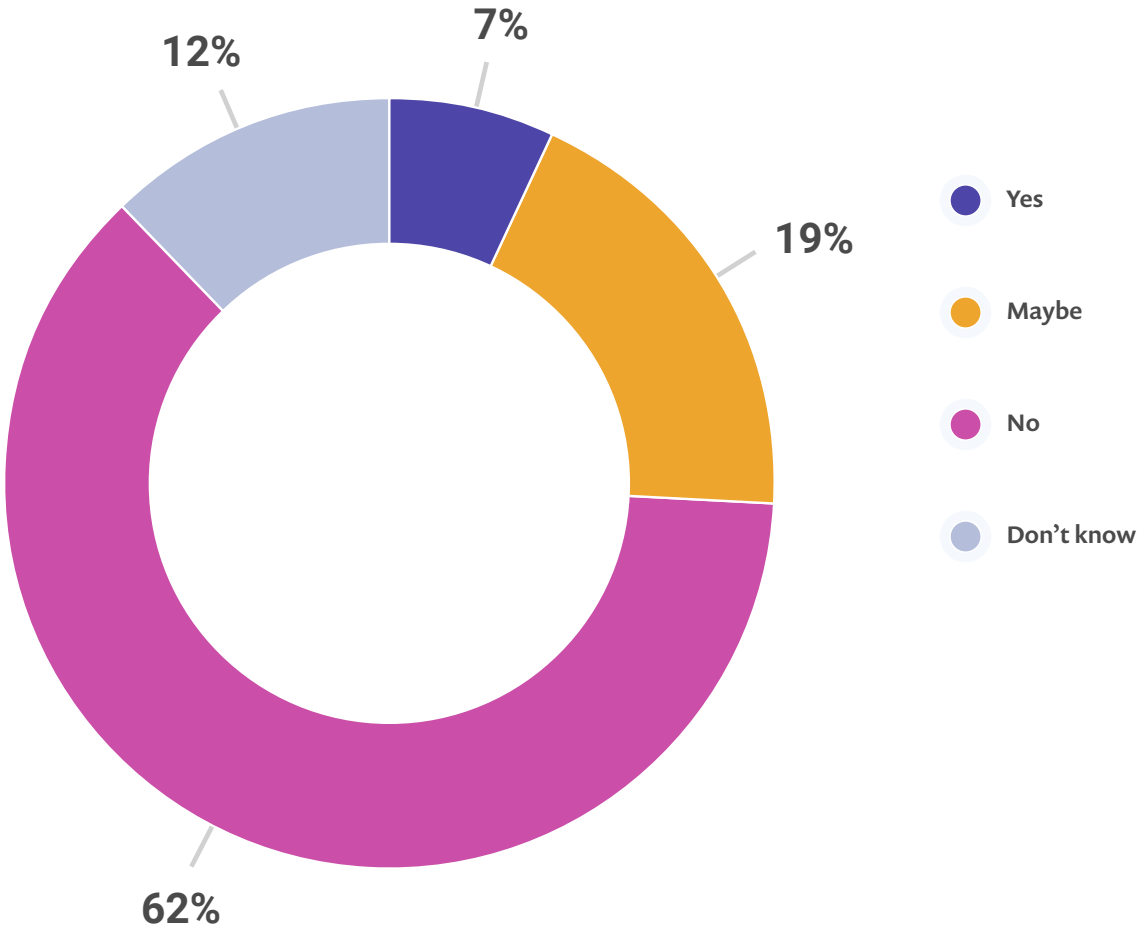
Starting with JDK 9, a new Java version is being released every March and September, which is a major change to the JDK release cadence. This impacts the update strategy for many users as this 6-month release cadence affects the support cycle as well. Moreover, this change has an impact on security too, with security patches not being backported to older versions.

According to our survey, for more than a third of the developers, the new cadence influenced their decision to pay for support. At least 41% of the respondents claim that the changes made to the release cadence and support played some role in that decision.



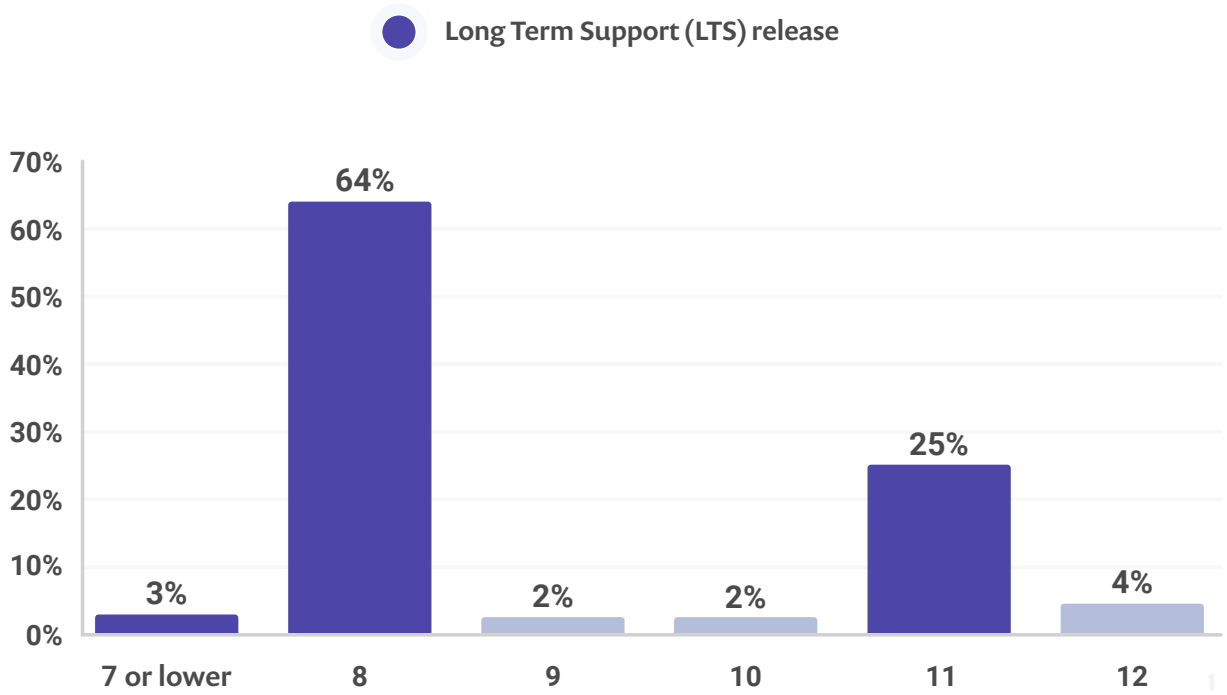
# 05. Will you consider paying for JDK support in the future, based on the latest release cadence changes?

The majority of the developers who participated in this survey, don't believe they will change their minds about paying JDK support in the future. As the community involvement is bound to grow in future JDK releases, it is possible that, in a year from now, we see a decrease in the number of developers who consider paying for JDK support. Although only 7% of the respondents are willing to pay for support in the future, a significant 19% is still considering the possibility.



# 06. Which Java SE version do you use in production for your main application?

The introduction to Java 9 brought significant structural changes to the JDK. Last year, we saw evidence that these changes were possibly holding people back from moving beyond Java 8. This year, the numbers are a little different. The number of people working with Java 8 in production is still very high; however, with the first Long Term Support (LTS) version of Java, Java 11, released in September 2018, the landscape is slowly changing. A quarter of the developers who participated in this survey, are now running Java 11 in production.

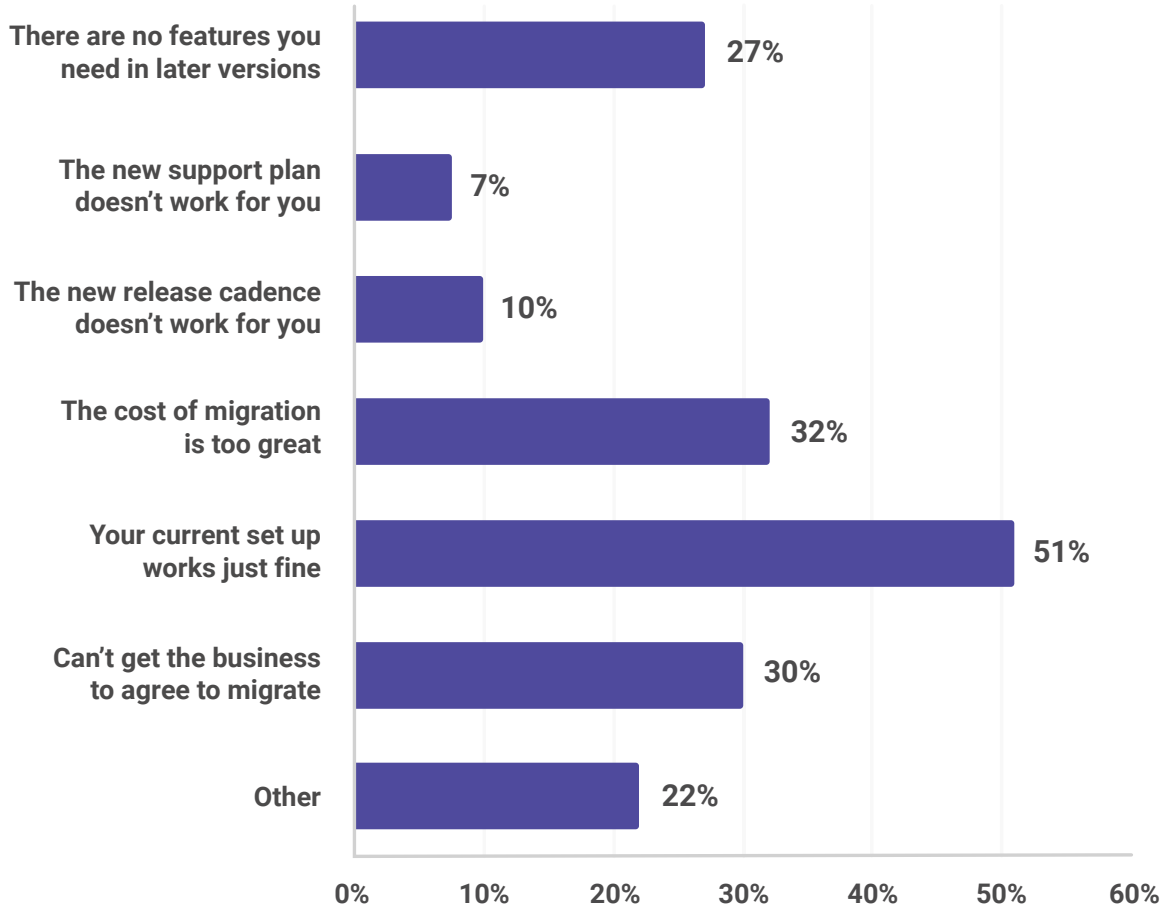


# 07. What are the reasons why you have not moved to a more recent version?

Although the new release cadence was introduced over 2 years ago we still do not see significant adoption. Many people are not able or willing to migrate every 6 months in order to stay up to date. 51% of respondents say that their current setup is working fine, so change is not needed. As the cost of migration seems to be too high, a lot of companies are reluctant to adopt changes so rapidly.

The biggest question is, are developers and companies to blame for staying put on an older version? In fact, migrating every six months, doesn't necessarily give you any real return on investment. Nonetheless, it is possible that we have to wait a bit longer to see the effects of users migrating to more recent versions.

Some of the reasons participants explained in the 'other' category include, application servers or libraries not supporting the latest versions of the JDK fast enough.



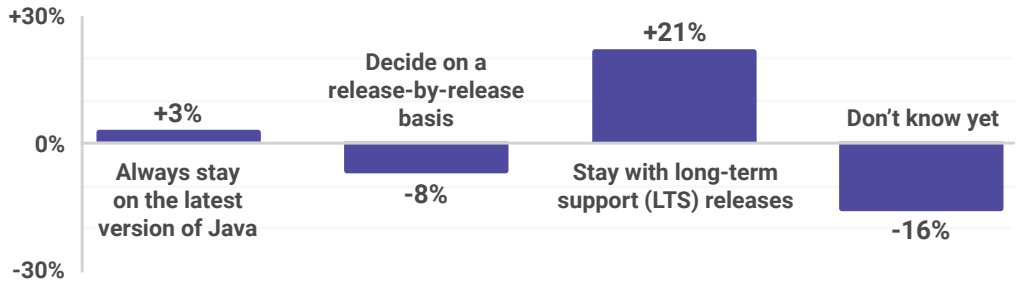

Up to 3 responses allowed.

# 08. What is your approach to adopting new JDK releases in production?

The plan for most people (55%) is to stick with long term releases. That is hardly surprising as these versions are released every 3 years, similar to the cadence people are already familiar with in the JDK ecosystem. Still, 22% of the respondents report that they plan to decide whether or not to upgrade on a release-by-release basis to see if the newly introduced features are important enough to justify migrating.


Comparing the numbers from 2018 to this year's numbers, it is interesting to see the shift in the approach of JDK adoption. In 2018 the number of indecisive developers was higher, as was the number of people who stated they would decide to upgrade on a release-by-release basis.
In contrast, with an increase of 21%, the majority of developers now prefer to adopt LTS releases only — this is a significant swing in just over one year, as shown in the bar graph below that compares the changes over time.

**Difference in share from 2018 to 2019**

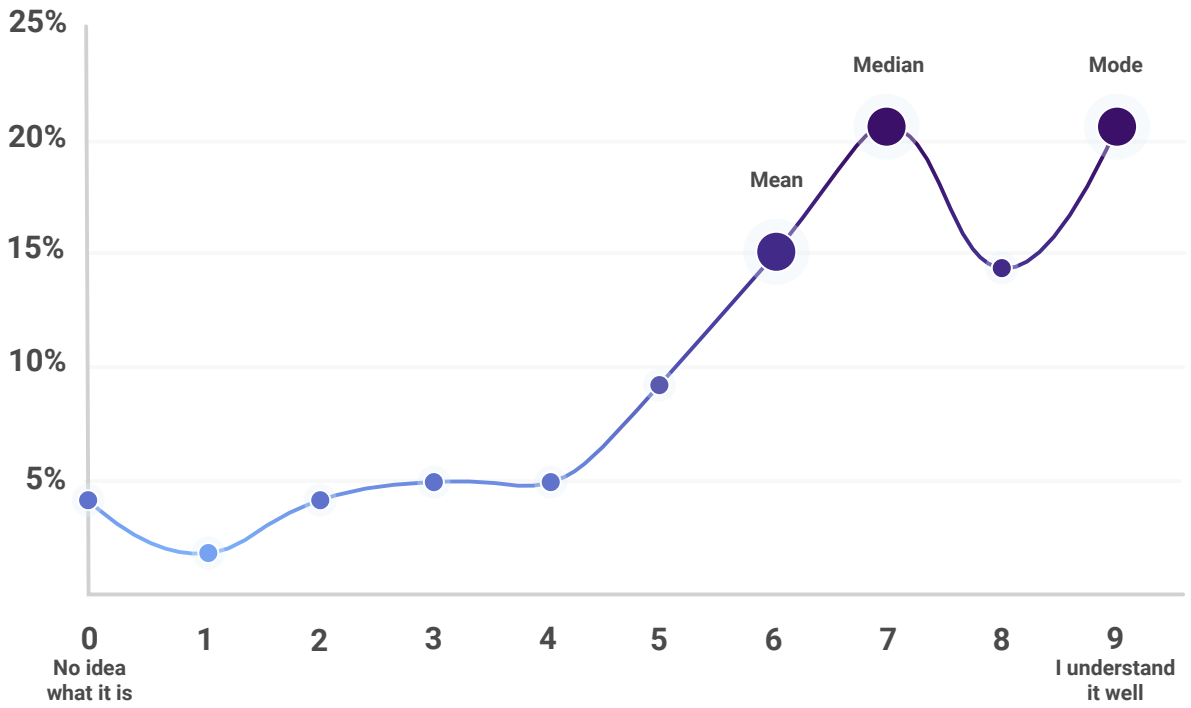


# 09. How well do you understand the new 6-month release model and the updated support statements made about each release?

With the new 6-month Java release model, features become fully or partially available to developers at a faster pace, between LTS releases. This means that the official support for a non-LTS release lasts only until the next version becomes available, that is every six months. Every three years or so — similarly to the old cadance — a version is marked as LTS and there are support options available for an extended period of time, until the next LTS version arrives.

It's also important to note that, according to the maintainers of Java, the non-LTS versions are not alpha or beta versions — they are fully supported, production-ready versions of Java.

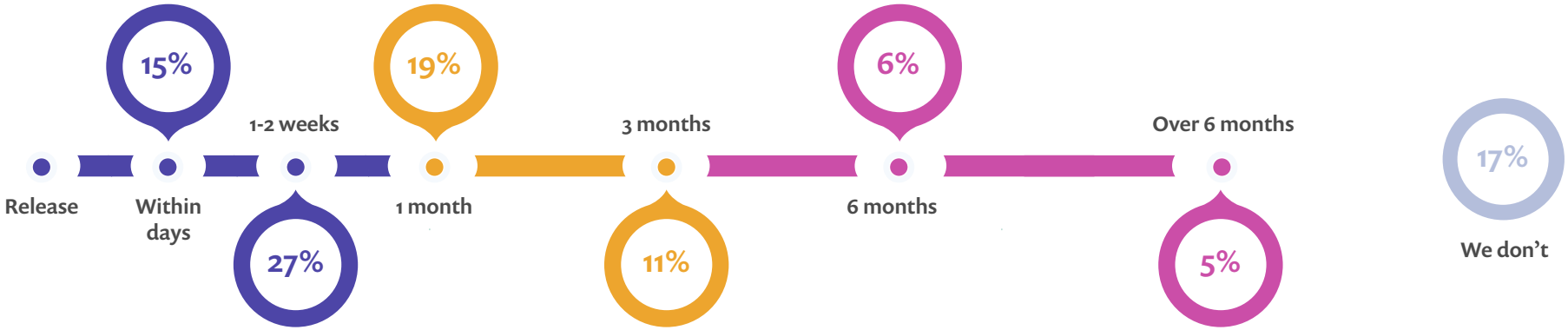
When we asked the community how well they understand this new release model, the responses were rather positive. Looking at the overall results, it's safe to say that the majority of Java users have a very good understanding of the new release cadence with more than half of them rating themselves with a score of 7 out of 10 or higher.



# 10. How quickly do you apply critical JDK security updates?

In order to keep a system healthy, applying critical JDK security updates is essential. However, 17% of respondents claim that they do not apply any security patches. Scary, isn't it? Nonetheless, it is a relief to see that the vast majority of developers (61%) do take security seriously and apply security updates within a month of release.

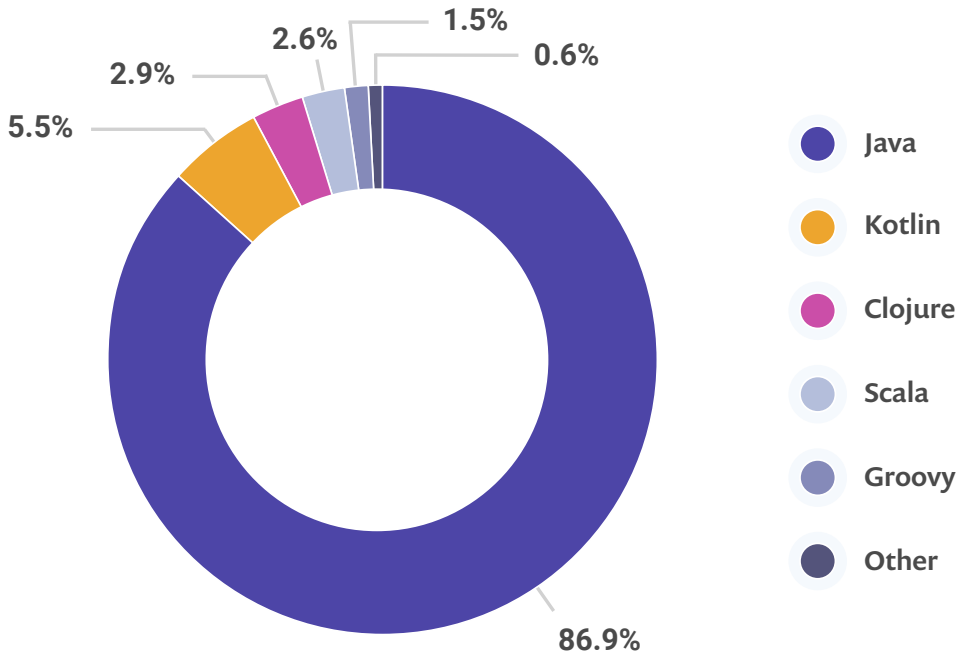
We also want to give a big shout-out to the 15% of the respondents who apply security updates almost immediately after release. Awesome job!



# 11. What is the main JVM language you use for your main application?

While the variety of JVM languages grew over the last couple of years, the large majority of JVM users — nearly 9 out of 10 — still use Java as their main language.

One of the reasons we see this consistently large percentage over the years, is the fact that Java is constantly changing. Many of the language constructs and paradigms that other JVM languages have been using to differentiate themselves from Java, have been introduced and implemented in Java, such as local type inference, and lambdas, to name just a couple. The new Java release cycle also makes these new features available to developers earlier than before.




However, despite the strong preference for Java, the use of other JVM languages grew as well. Particularly Kotlin, a language developed by JetBrains, gained a lot of popularity over the last couple of years. In fact, Kotlin grew from 2.4%, according to last year's report, to an impressive 5.5%.

The growth of Kotlin adoption, among JVM users, is not surprising considering how seamlessly it integrates with Java. Not to mention that, the adoption of Kotlin in frameworks like Spring Boot made it easier to create production-grade systems.

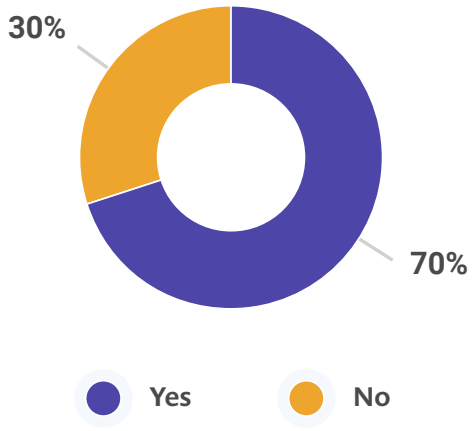
It is important to note that, the innovations Kotlin tries to achieve flow back into the development of Java. Newer Java versions try to integrate concepts that are popular and loved in languages like Kotlin. It is interesting to see how this influences the future adoption of different JVM languages.

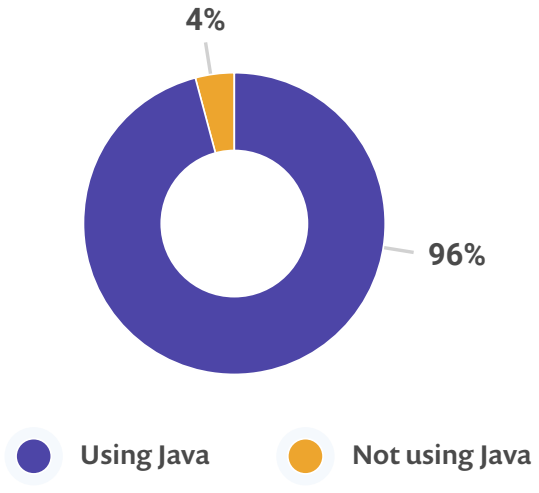
# 12. Do you write or maintain any Java applications?

## For those who don't use Java in their main application, do they use it at all?

Not every JVM developer uses Java as their main language. Based on the responses to the previous question, JVM developers who predominantly use other languages in their application, account for 13%. Out of this group of developers, the majority (70%) still uses Java in some capacity, during their regular work.

This means that 96% of overall respondents use Java, either as their main language or to a smaller extent, in their application. Only 4% of respondents run applications on the JVM that entirely use alternative JVM languages.



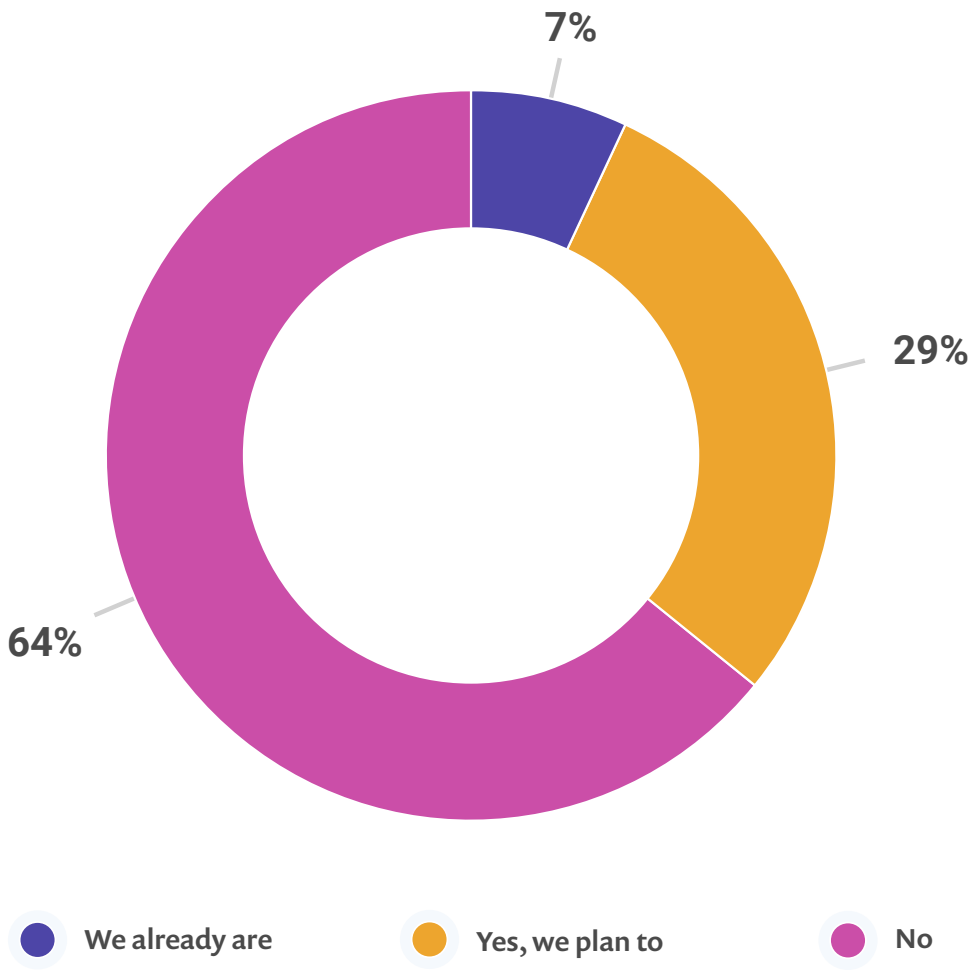



# 13. Are you using, or are you planning to use, Java modules in your Java applications?

The release of Java 9 introduced some major architectural changes. The most famous and impactful change was the introduction of the module system, formally known as the Java Platform Module System (JPMS). This new abstraction above packages make it possible to create a smaller, more fit-for-purpose JDK.

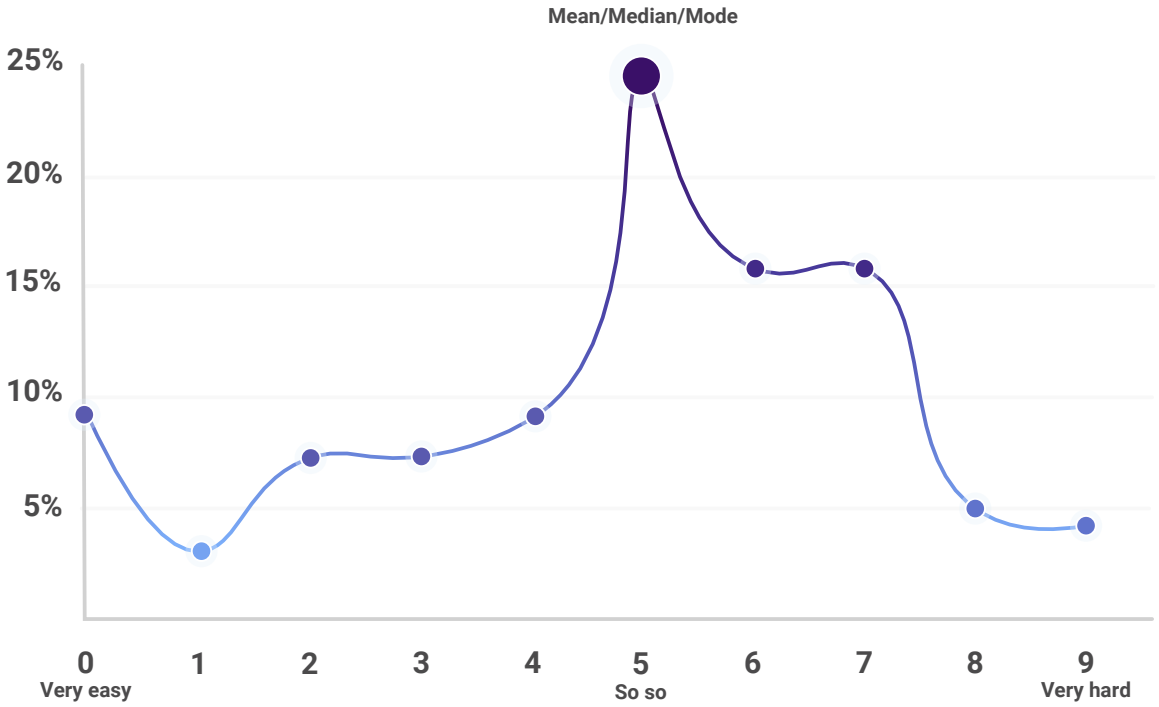
But architectural changes of this magnitude are bound to have a large impact on older or legacy systems, if they are to be migrated. It's important to note, however, that you are not forced to use the module system in Java 9 and higher. You are able to continue to use the classpath while the module system silently bundles all your jars into the unnamed module.

When we asked developers whether they are using or plan to use the module system, we got some interesting results with only 7% of the respondents stating that they are already using it. The vast majority of the participants — over 6 out of 10 — do not use the JPMS while 29% plan to use it in the future.



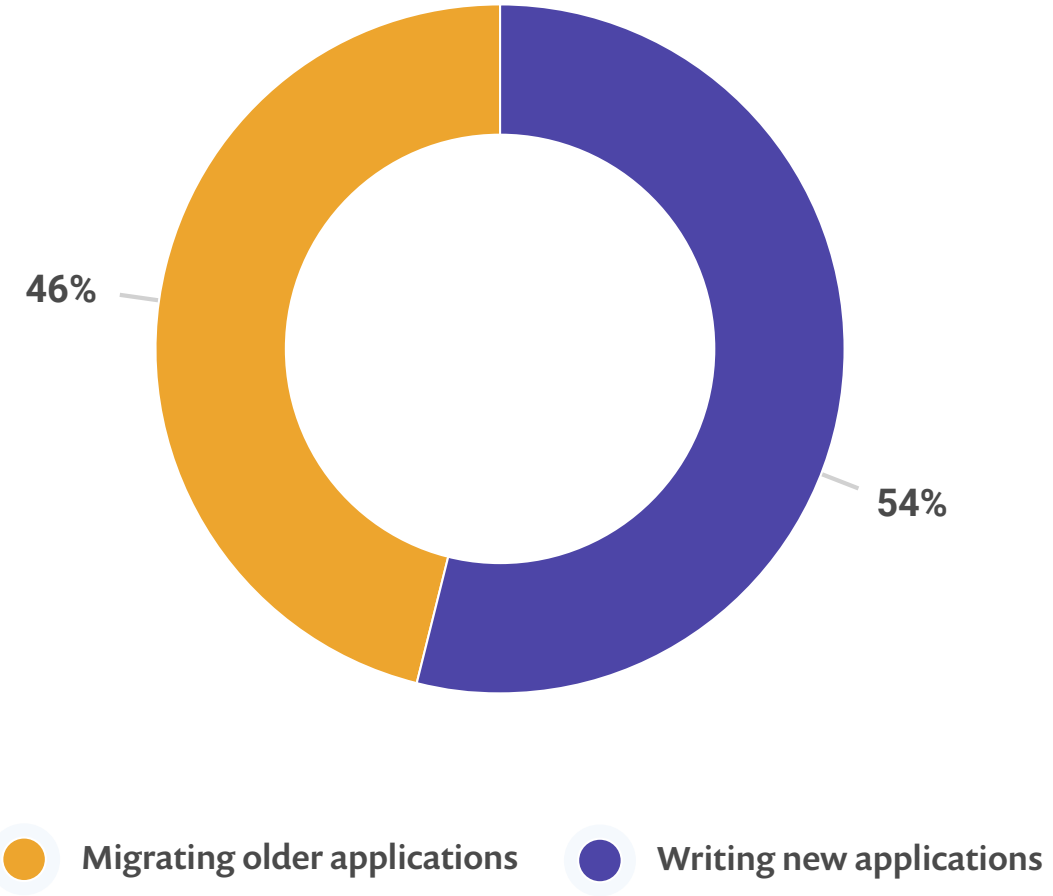
# 14. How easy was it to adopt Java modules?

When it comes to adopting Java modules, the opinions on the level of difficulty differs. While a quarter of the respondents found the adoption process to be neither hard or easy, a significant percentage (25%) found the JPMS adoption to be rather hard (7/10 or higher). The mean, median and modal averages were all 5/10. According to these responses, it is safe to conclude that, while adopting Java modules takes time and effort, it's not rocket science!



# 15. Did you adopt Java modules while writing new applications or while migrating older ones?

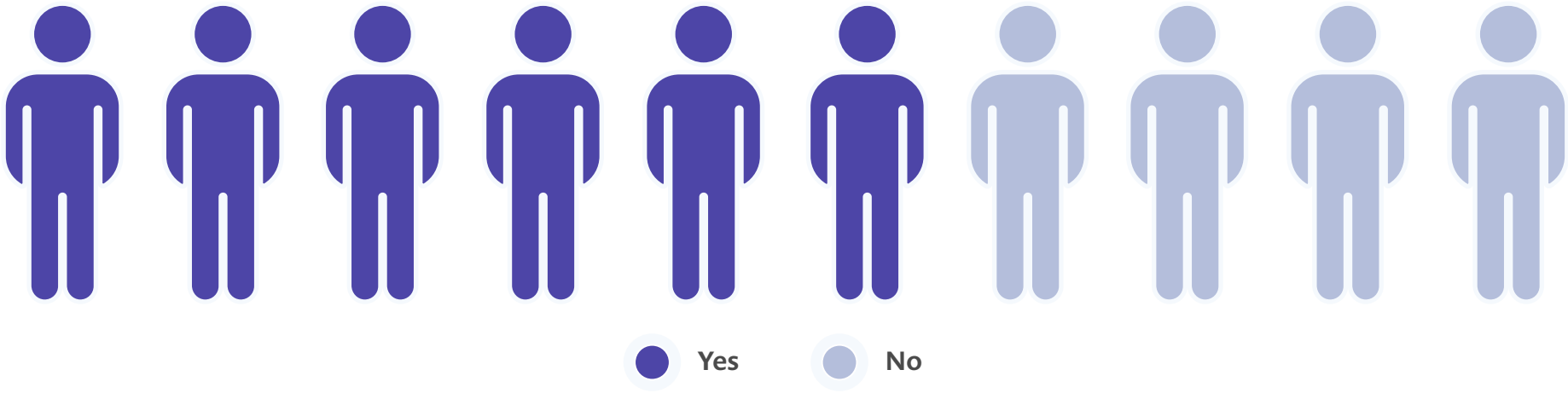
Adopting something new is considered easier when you work on a greenfield project. Creating an application from scratch, for the most part means that you do not have to deal with the complicated evolution of a software project. When it comes to adopting Java modules, though, this is not always the case. According to our survey, almost half of the respondents adopted Java modules while migrating older applications.



# About your application

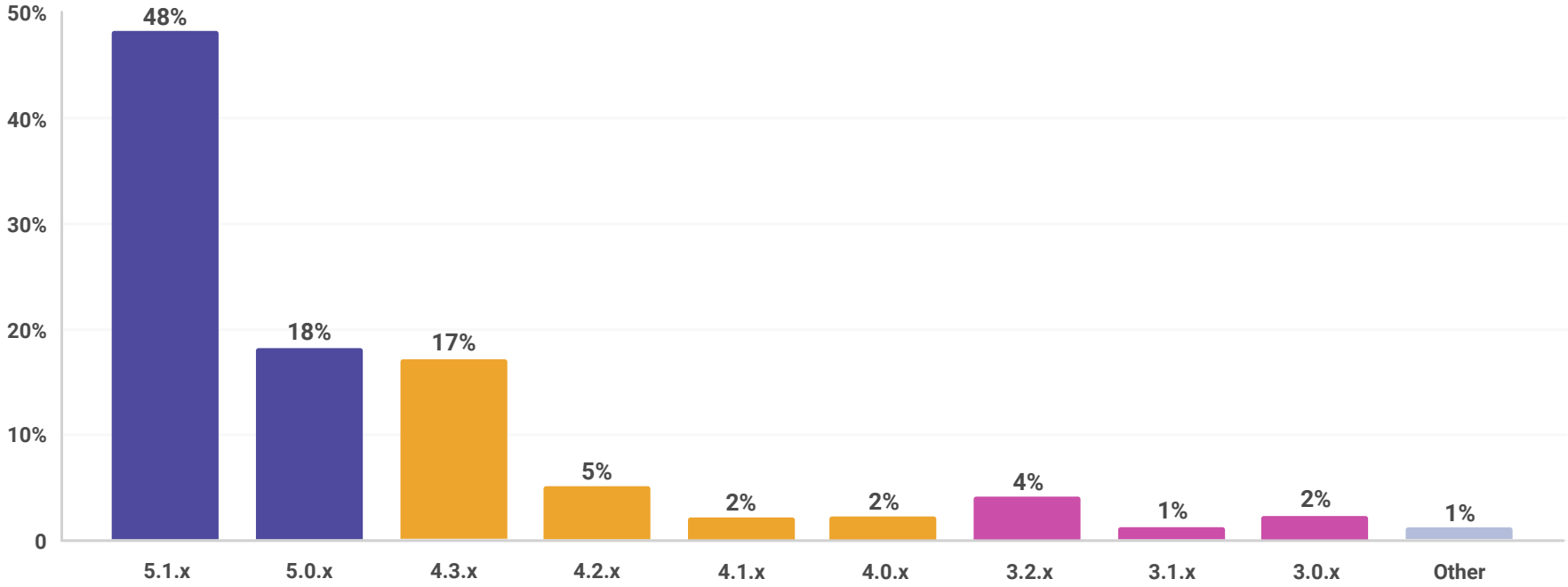
# 16. Do you use the Spring Framework?

Exactly 6 out of 10 people depend on the Spring Framework for the production of their application. That is a remarkably high market share for a third-party open source framework.



# 17. What Spring version do you use for your main application?

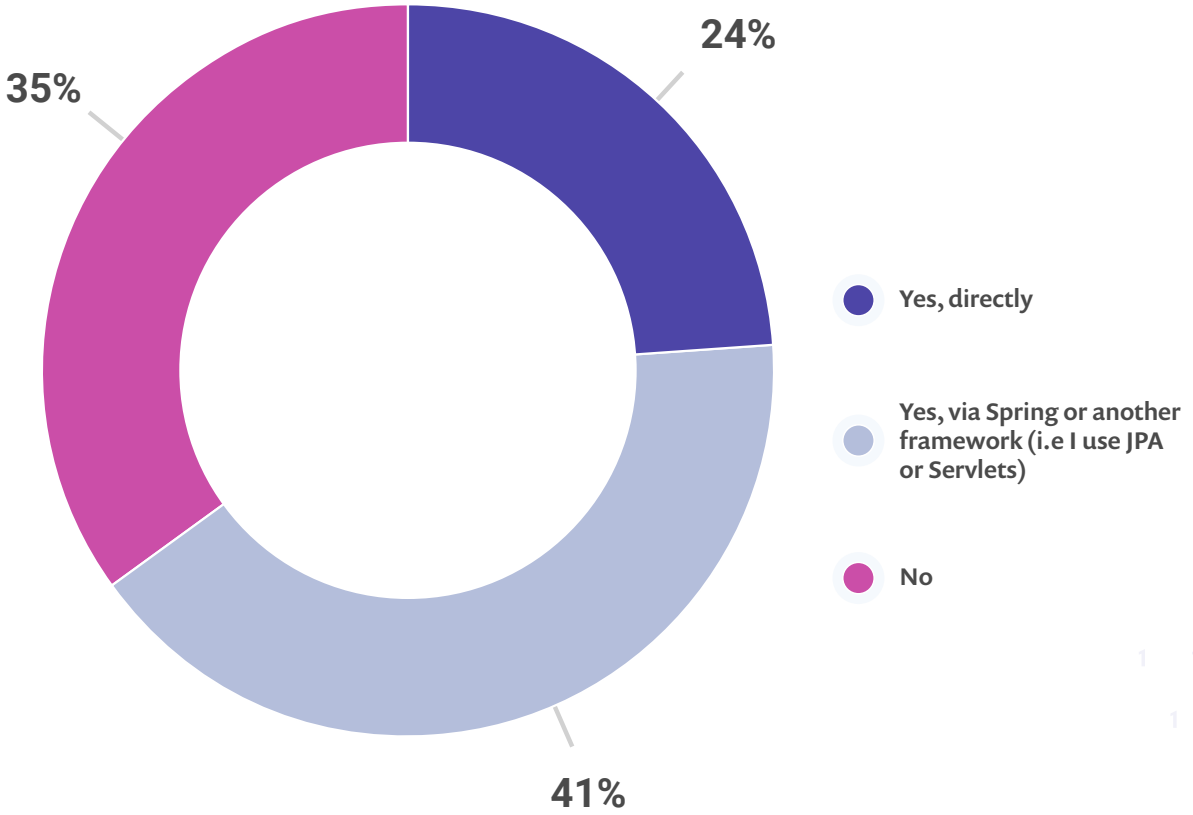
Spring has been around for a long time. By introducing significant changes and innovations, Spring has evolved into the most dominant framework in the Java ecosystem. With two thirds of Spring users working with Spring 5, there's a strong adoption rate of newer versions.



# 18. Do you use Enterprise Java? (J2EE, Java EE, Jakarta EE)

The question of whether Java developers use the Enterprise Edition (EE) of Java is something we ask every year. Only this year, we slightly changed the question. We added the option "Yes, via Spring or another framework (e.g. JPA or Servlets) to ensure that people who use EE indirectly do not choose the "No" option.

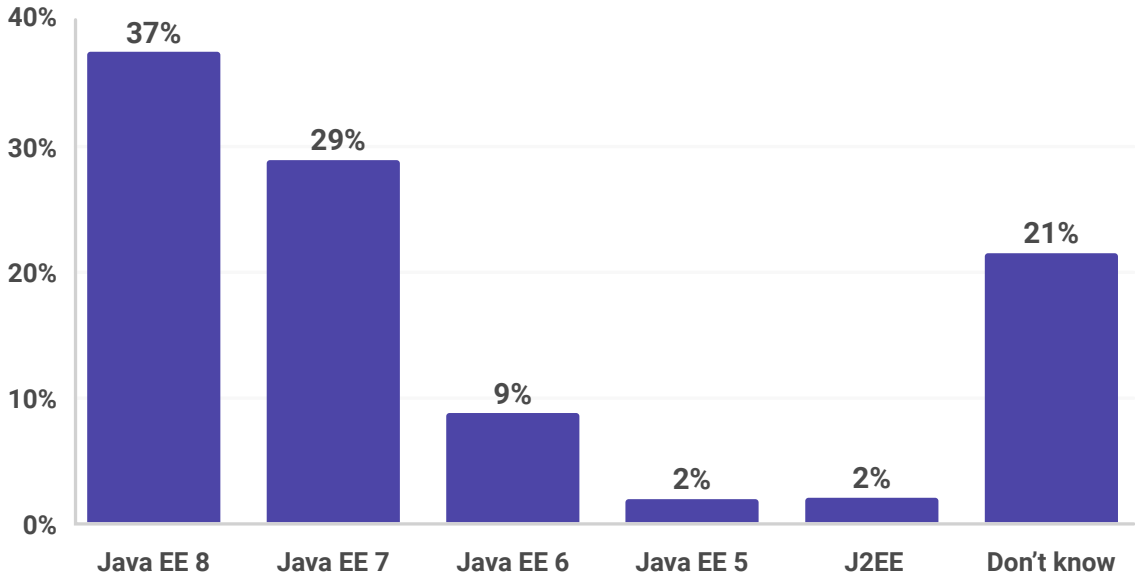
With 35% of developers reporting that they don't use Java EE, the landscape hasn't changed much since last year (38%). It is important to point out, however, that 4 out of 10 developers are using Enterprise Java indirectly. This does raise some concerns over Java EE's popularity.



# 19. What Java EE version do you use for your main application?

Almost 4 out of 10 people use the latest version of Java EE while Java EE 7 still remains quite popular. What's more, 2% of developers reported that they still use J2EE and, even though this seems like a very small percentage, it is a significant number as it is almost equal to the number of people that use Scala as their main application language!

It's also important to mention that 21% of the respondents do not know the exact version of Java EE they're using. By cross-referencing the answers to this question with the previous question, we found out that 95% of developers who are not aware of their exact Java EE version, use Java EE indirectly, namely through the Spring Framework.

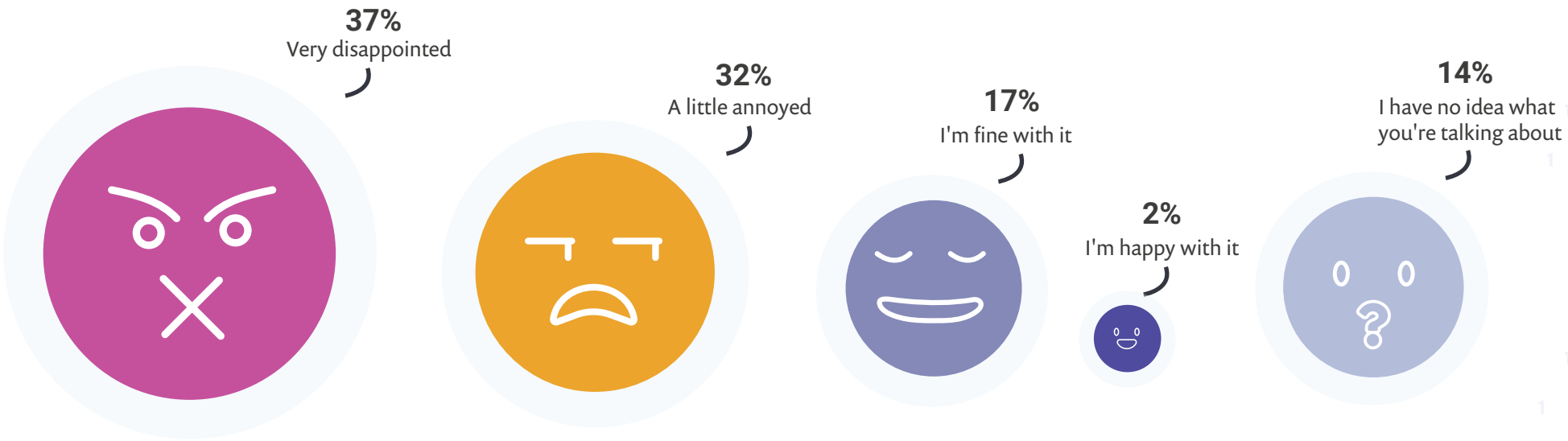


# 20. What was your reaction to Oracle and the Eclipse foundation not agreeing on continued usage of the javax namespace?

After many months of negotiations, Oracle and the Eclipse foundation weren't able to come to an agreement over the usage of the javax package namespace by the Eclipse Foundation Community. The javax namespace falls under trademark by Oracle which means that, moving forward, all improvements made to Jakarta EE by the Eclipse Foundation, have to use a different package name. As a result, changes to Jakarta EE are also accompanied with migration of library code.

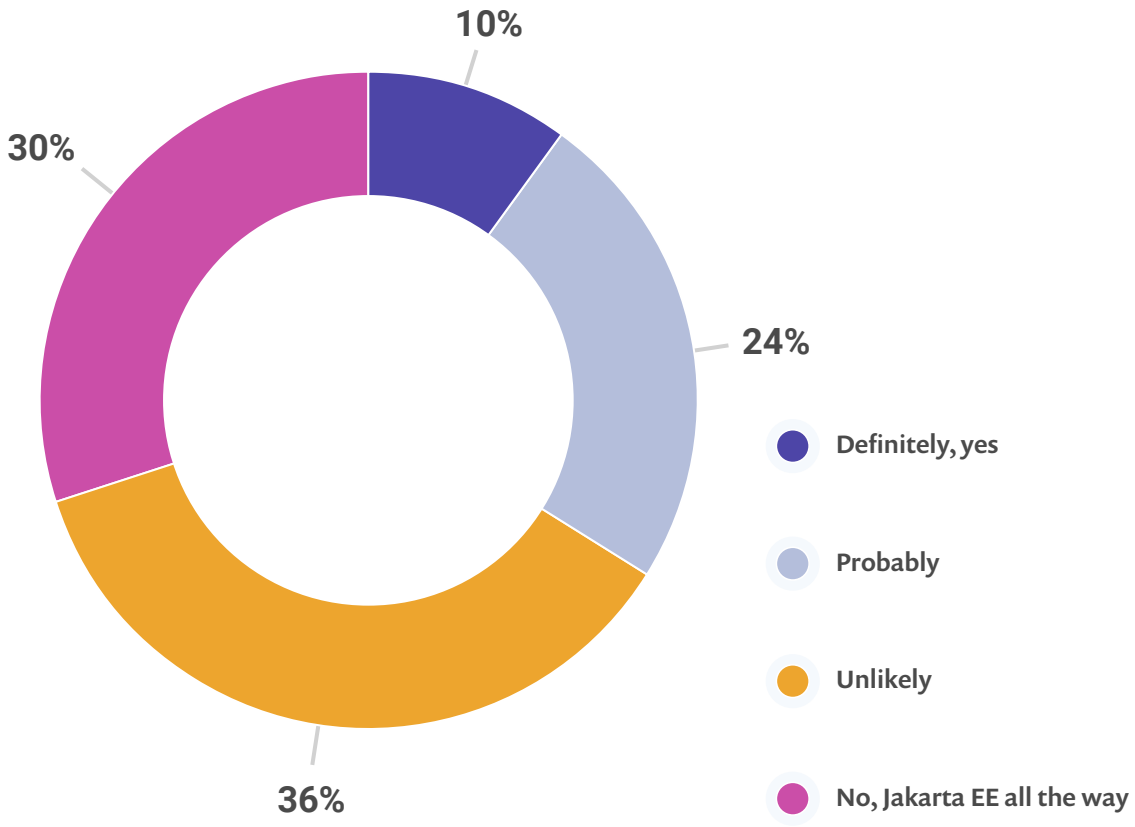
Although the change in package name clearly marks the ownership of that package — Oracle up to Java EE 8 and the Eclipse Foundation from Jakarta EE 8 onwards — it affects every API in the Enterprise Edition as they all begin with javax.

When asked about this development, the vast majority of participants feel annoyed by Oracle's position on the matter with 2 out of 3 JVM developers stating their disappointment of the negotiated outcomes. In fact, the responses to this question raise some concerns for Oracle. What if this outcome ultimately harms Oracle's stewardship of Java?



# 21. Would you consider switching to another framework/technology in order to avoid migrating to a newer Jakarta EE version, due to the javax namespace changes?

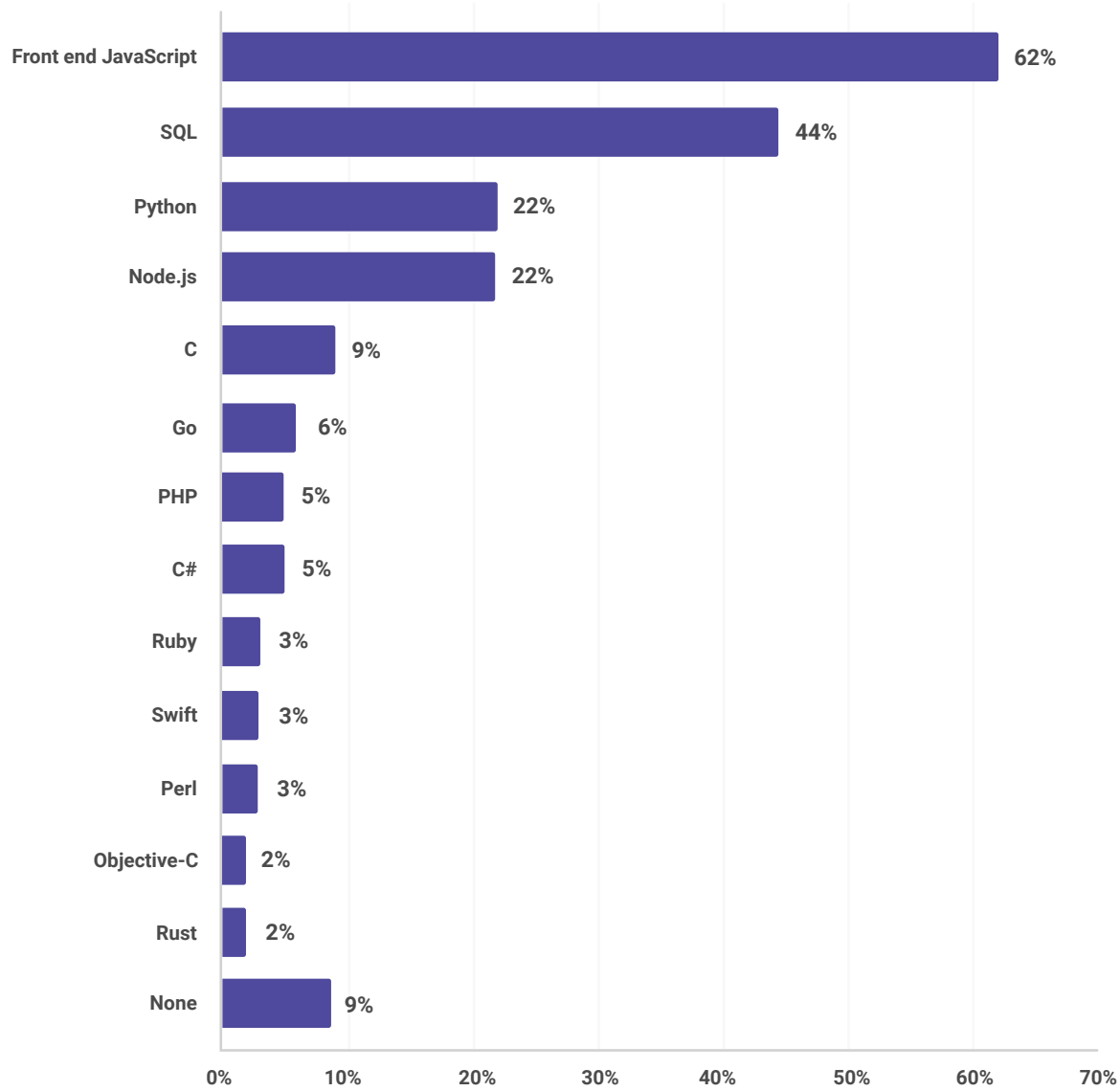
Despite the majority of the respondents being rather disappointed with the javax namespace changes, only 1 out of 10 developers would switch to another framework. According to our survey, 66% of developers are probably or definitely staying with Jakarta EE despite the namespace changes. It is possible that developers believe that these changes will not affect them, since the majority of them use the EE version indirectly, via frameworks like Spring. This points to the namespace change being more of a disappointing annoyance rather than anything developers really take action over.



# 22. What other languages does your application use?

Not many people use a single language for their application anymore. It's safe to say that the vast majority of developers nowadays need to be polyglot, fullstack or multi-lingual. As languages in many cases serve a specific goal it is obvious that developers use other languages alongside their main JVM language. This doesn't mean you have to like the language. Some languages are considered ~~a necessary evil~~ fit-for-purpose.

It is important to mention that multiple answers were allowed when answering this question. The results, however, are not that surprising. JavaScript is the most popular language for front end development with 62%, SQL with 44% is the most popular for querying databases, while the most popular choice for data science and machine learning applications is Python with 22%



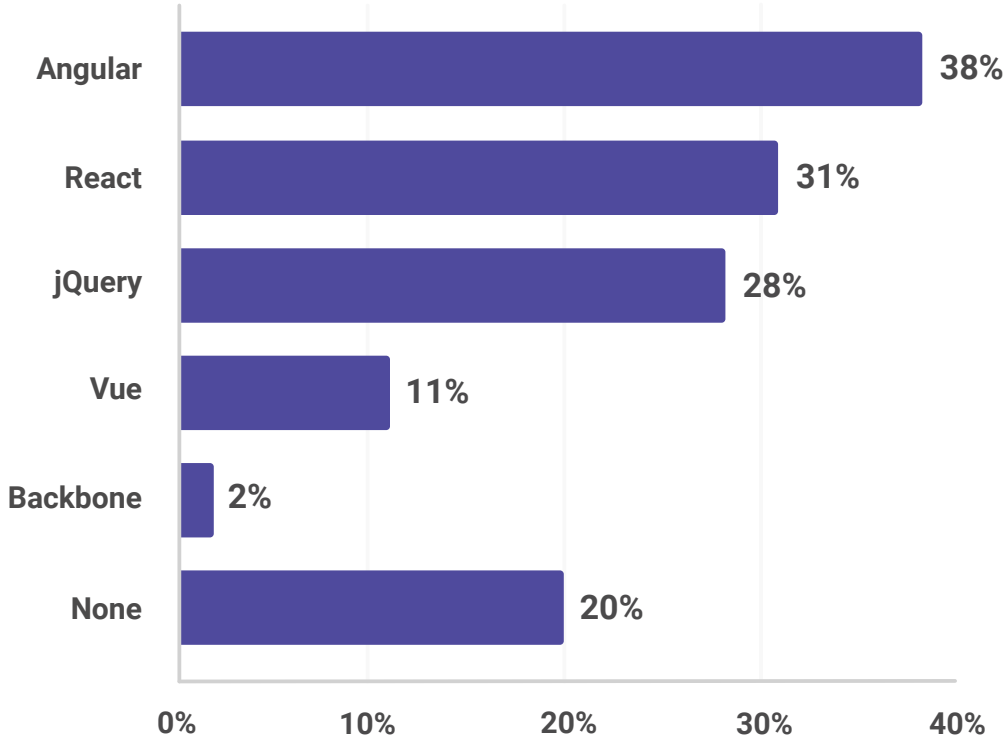

Multiple responses allowed.

# 23. Which client-side web frameworks do you use?

Participants were able to select multiple options for this question — why choose one framework when you can use them all, right?

Looking at the responses, Angular looks like the clear winner with 38%. However, with so many Angular versions available, we are not certain whether newer or older versions of Angular are the winners here. React is the runner-up with 31%, closely followed by jQuery with 28%.

It's also interesting to point out that, according to our survey, 2 out of 10 developers don't use any frameworks. Let's see if this list changes significantly over time.

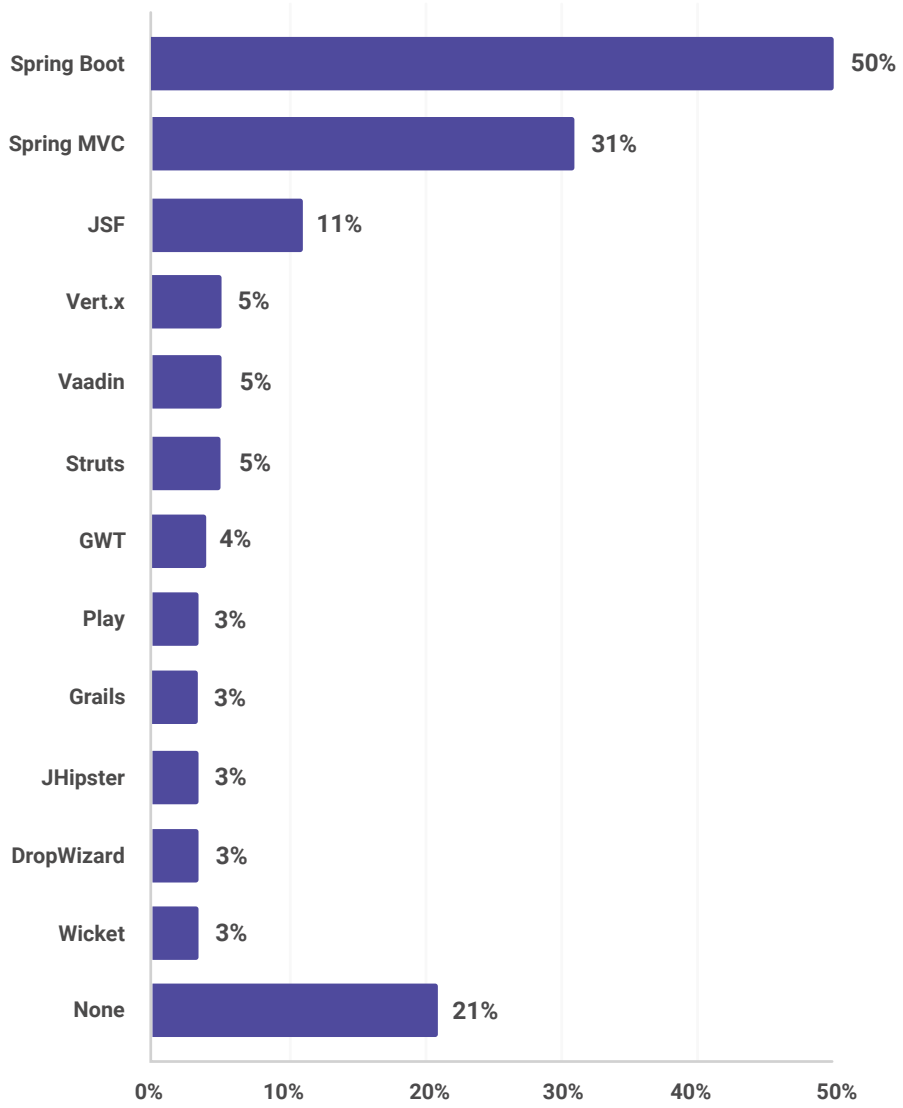



Multiple responses allowed.

# 24. Which server-side web frameworks do you use?

The server-side is still a Spring-dominated world, with half of the market using Spring Boot and almost a third using Spring MVC.

Frameworks like Micronaut and Quarkus probably have what it takes to compete against Spring. Nonetheless, let's wait until next year's report before we draw any conclusions. JHipster does not look as popular as one would expect from all the conference talks. It's also interesting to see that JSF is still alive.




Multiple responses allowed.

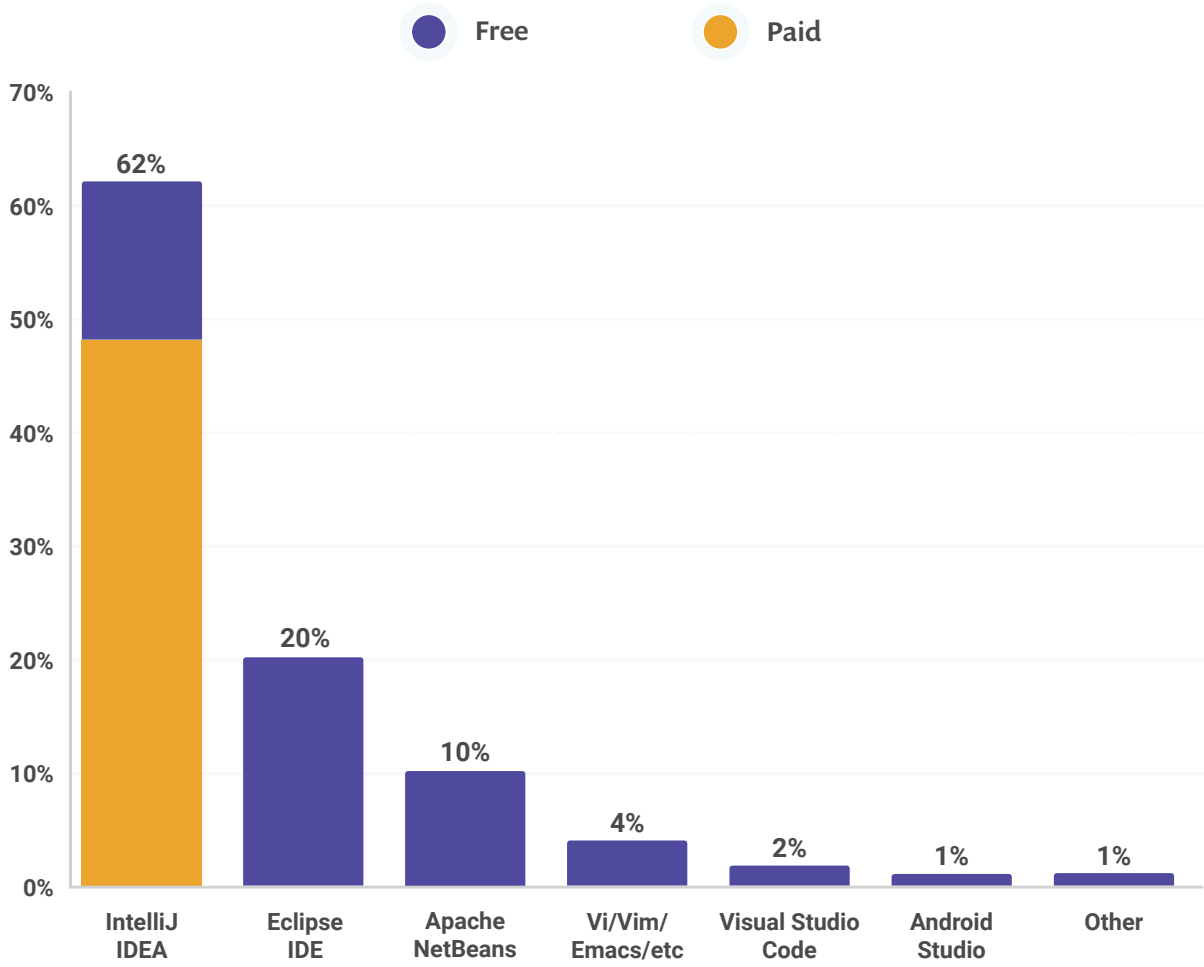
# About your tools

# 25. Which is the main Integrated Development Environment (IDE) you are using?

The results we see in the graph below are consistent with other recent surveys — IntelliJ IDEA is the most widely used IDE within the JVM community. According to our survey, 62% of developers use the Community and Ultimate versions of IntelliJ IDEA, making it today's dominant IDE among developers on the JVM.

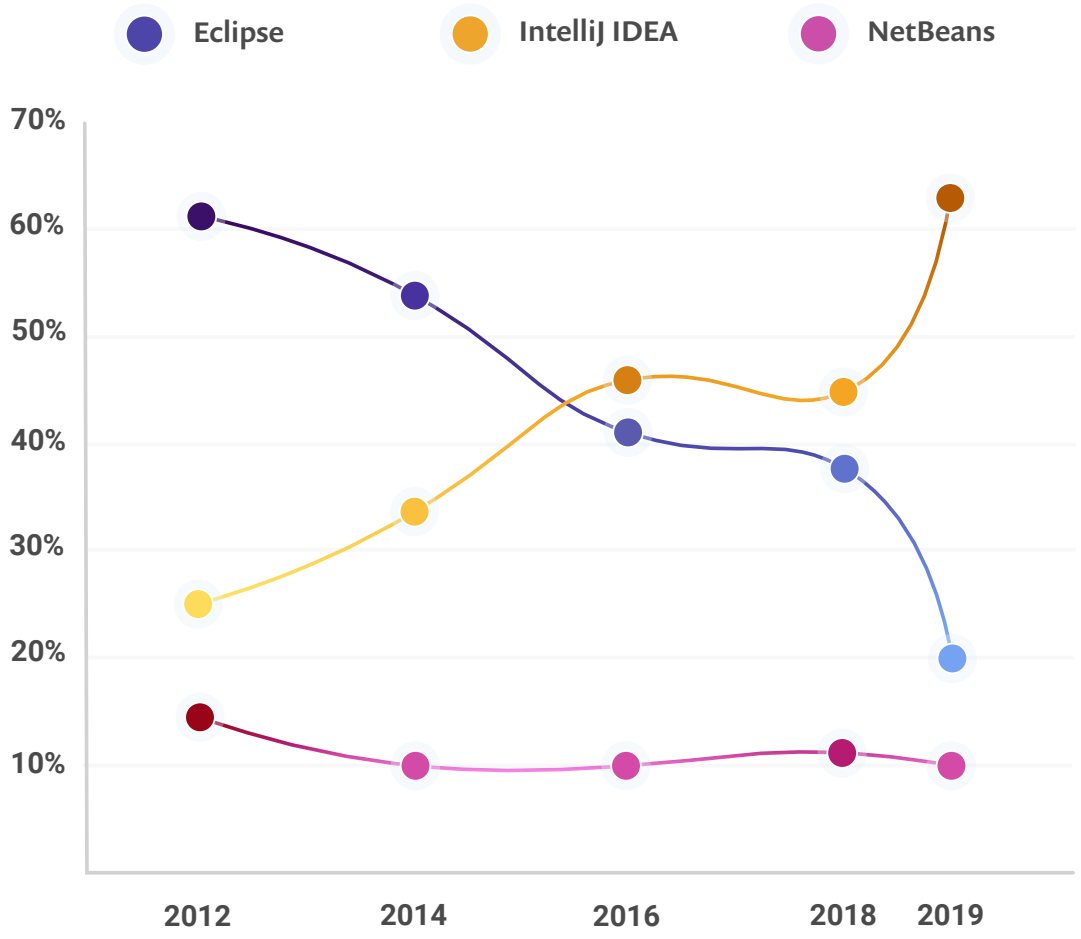
Apache NetBeans remains steady in 3th place with 10% of the market —roughly the same numbers as last year. However, when we look further down the list, it is surprising to see that the VS Code adoption barely grew, compared to last year. Despite being considered as one of the favoured IDEs in other ecosystems, it seems that VS Code does not share the same popularity among JVM developers.

In fact, even the VI/Vim/Emacs adoption is bigger than that of VS Code. These results bring to the surface a group of developers who, apparently, don't like IDEs. Are these real die-hard coders or do they feel smarter typing everything manually? In either case, we are not judging! :)



The support for a long list of out-of-the-box features, as well as the native support for Kotlin, have contributed to IntelliJ IDEA's rising popularity, With the Eclipse IDE dropping from 38% last year to only 20% this year, the gap between IntelliJ IDEA and Eclipse IDE is getting larger. Taking into account that, prior to 2016 (results kindly used from RebelLabs reports), Eclipse was the most used IDE, it becomes evident that folks over at JetBrains did a good job improving their software to fit the needs of JVM developers.

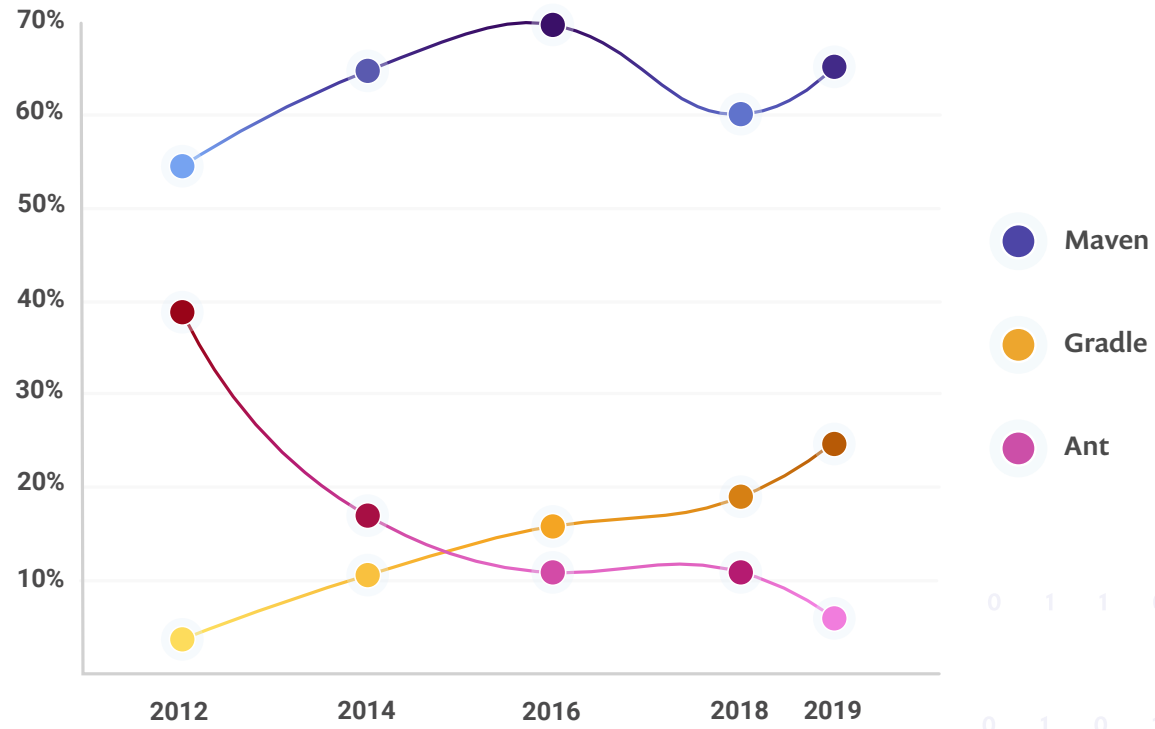
## IDE usage since 2012



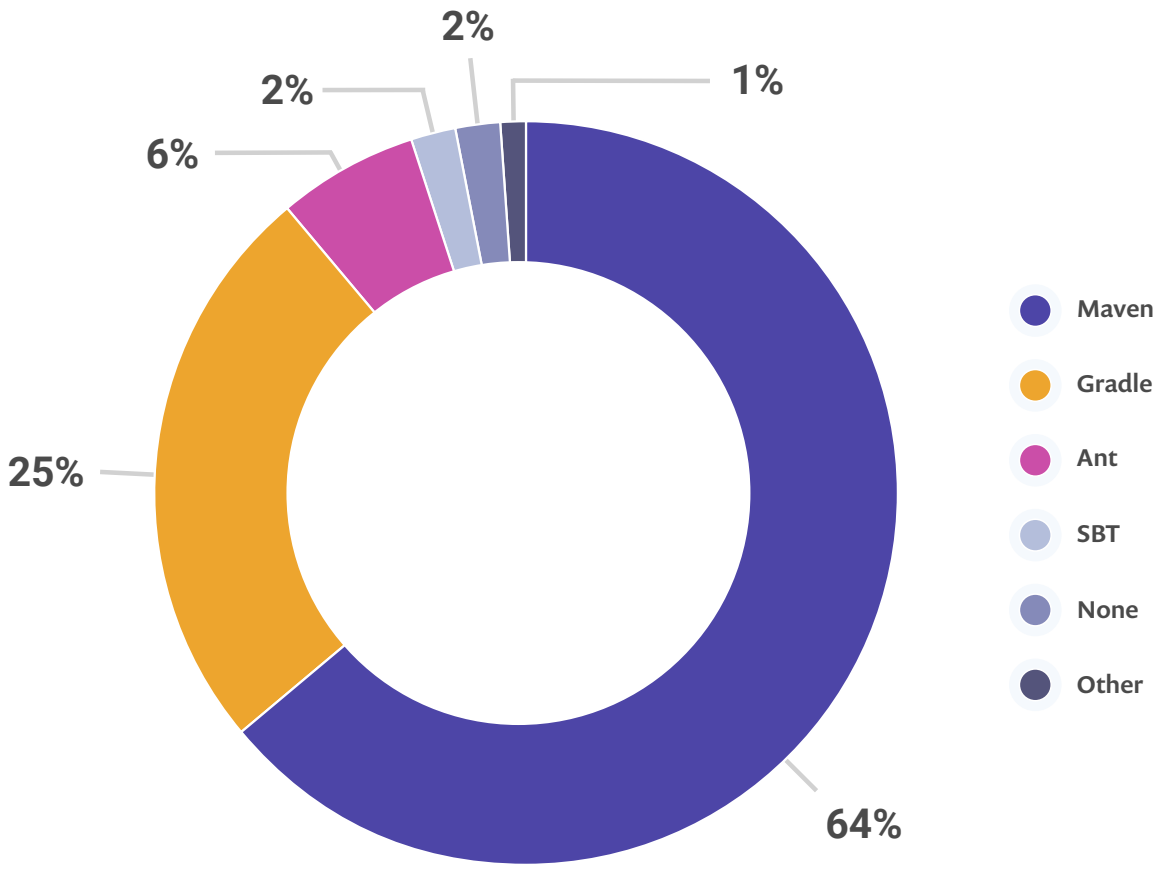
# 26. Which build tool do you use for your main application?

It is possible that teams depend on multiple build systems for different projects. So, for this question, we allowed only one answer, since we wanted to have a look at the one build tool developers use the most for their main application and compare it to the historical data (again, used from previous RebelLabs and Snyk reports) in order to reveal any trends.

## Build tool usage since 2012



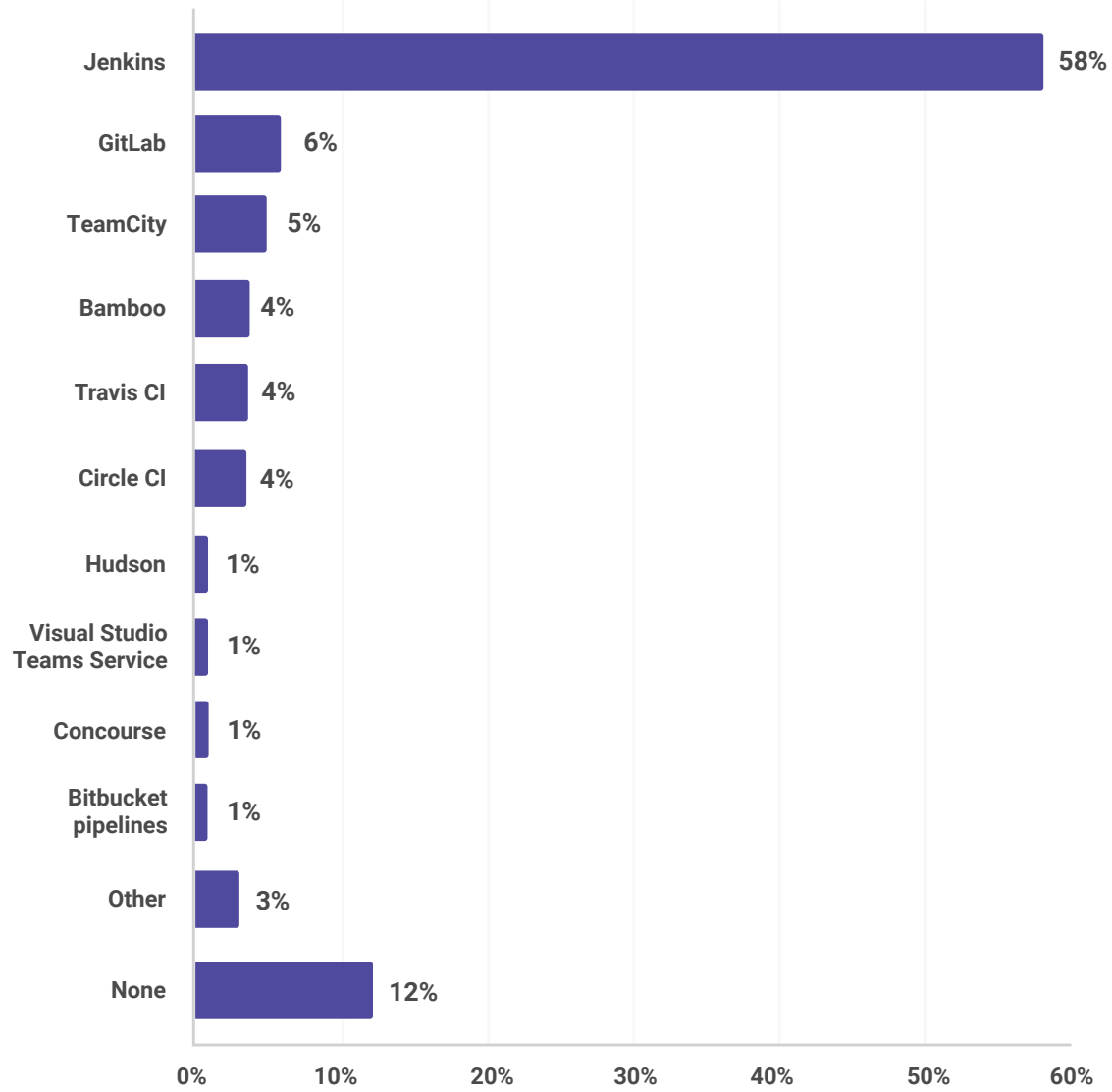
Maven is still number one, with two thirds of the share and a slight increase since last year. The runner-up, Gradle, shows the same rate of growth as its competition, Maven. So, is the "war" between build systems over or are we just taking a break?



# 27. Which CI server do you use?

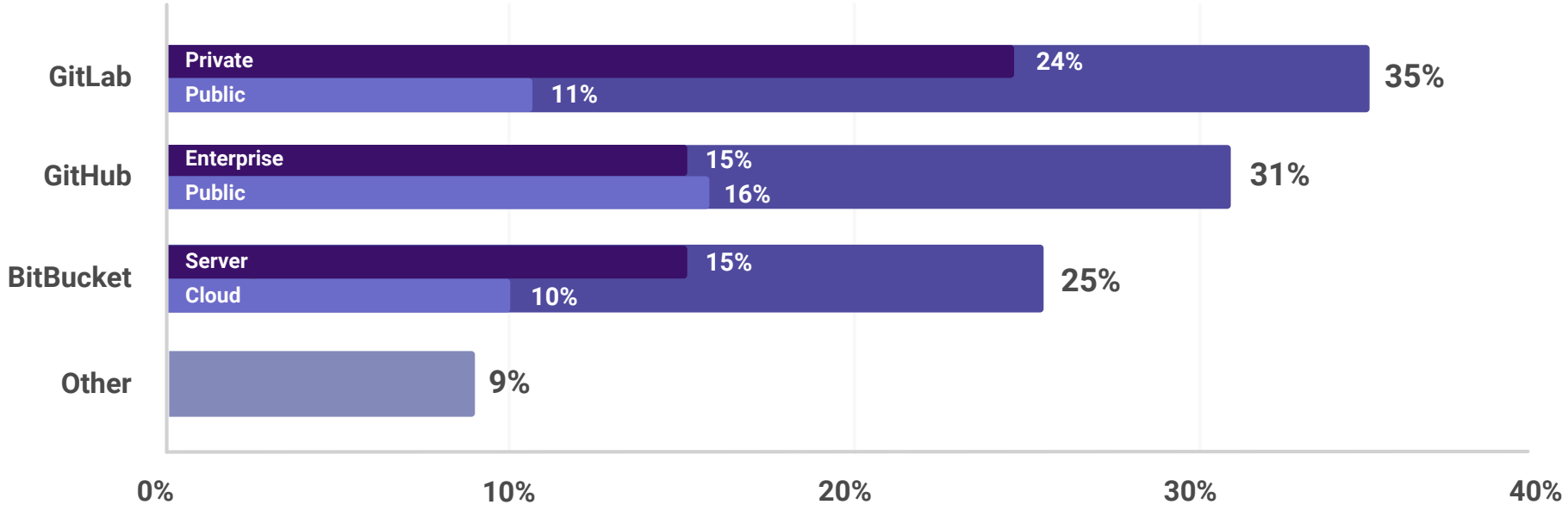
As most Java developers would expect, Jenkins wins the CI server race with a whopping 58% market share. Between Jenkins and the second most selected option, which was “none”, we see a huge gap. Although the amount of people who don't use any CI server is much smaller compared to last year, it is still surprisingly high. But why do people choose to not use CI servers? That's an interesting question to ask developers in future surveys!

The nearest competitors to Jenkins are GitLab with 6% and TeamCity with 5%.



# 28. Which code repository do you use for your main application?

It is possibly surprising for people to hear that GitLab is the winner of this battle. With 35% in total market share, it has a small advantage over GitHub in second place with 31%. We also notice that the public use of GitLab is lower, mostly because they have been offering private repositories for a long time. On top of that, GitLab offers a lot more than just a repository, including a CI pipeline. However, taking into account the responses to the previous question, this is unlikely to be the reason for using GitLab over GitHub.



# 29. When do you scan your dependencies for known vulnerabilities?

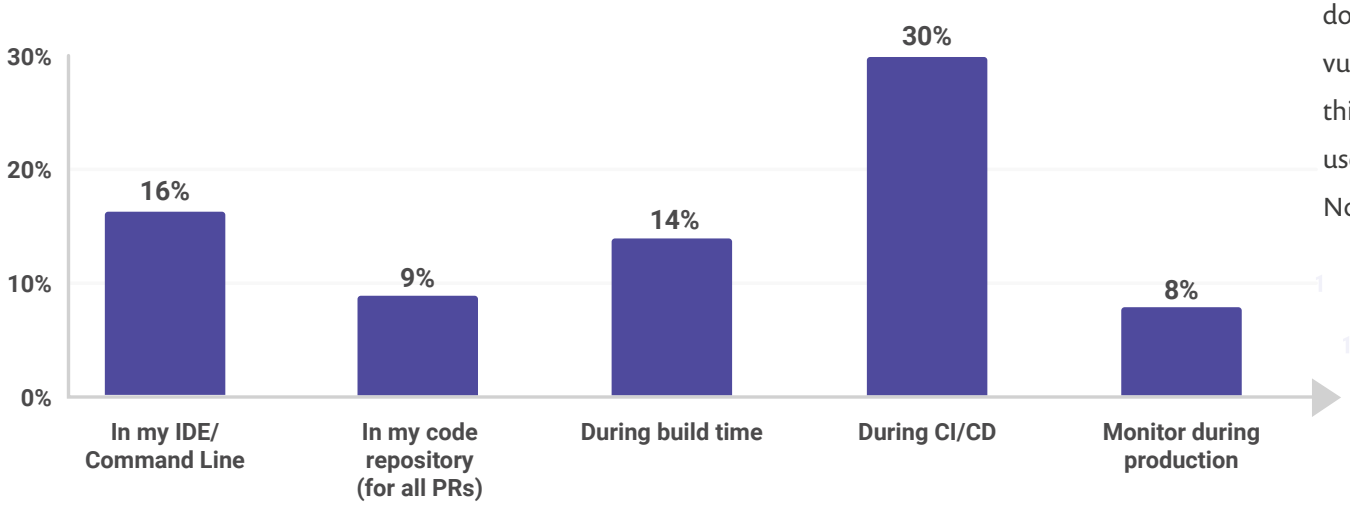
Scanning your dependencies for known vulnerabilities is the wisest thing to do! It is crucial to know if the code someone else produced is safe to use. Once a vulnerability is discovered, the list of potential victims is extensive, depending on how widely that particular package is used. If a vulnerability is already disclosed, there's a good chance that a fix is already available in a newer version of the package. However, if a developer is still using an older version, unaware of the existing security issue or its fix, they are vulnerable without knowing it.

According to our survey, 30% of the respondents, scan their dependencies for known vulnerabilities as part of the CI/CD pipeline. Using these scans as a gatekeeper before production deployment, is a good start.

However, scanning in multiple places during development, for example, on your local machine (16%) or when a PR is published (9%), helps to identify problems earlier.

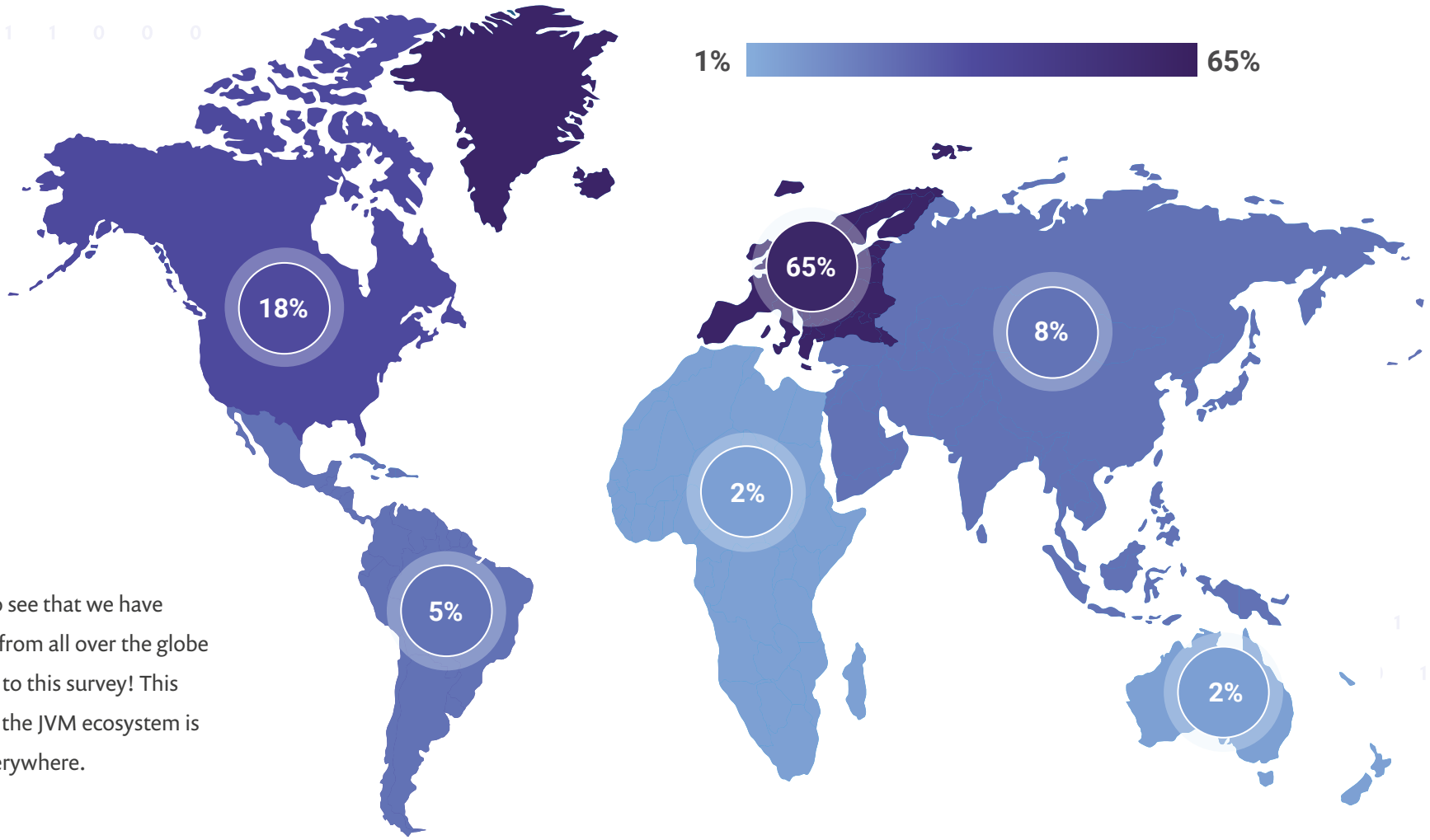
Discovering issues later in the software development life cycle (SDLC), often means that there's a significant amount of rework to be done in order to fix it.

Having said that, it is surprising to see that only 8% of the respondents monitor their applications during production. Discovering vulnerabilities happens over time, therefore, monitoring a production snapshot on a regular basis is the wise thing to do. Something even scarier to witness is that 28% of the participants do not scan their dependencies for known vulnerabilities. Hopefully, the explanation behind this percentage is that, these developers do not use any dependencies in their current application. Nobody wants to be the next Equifax, right?



# About you

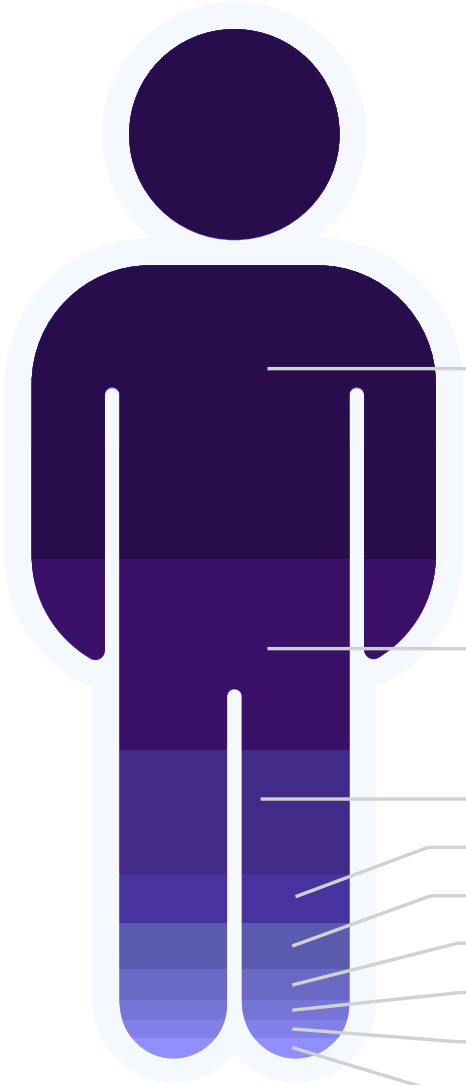
# 30. Where are you from?




It is great to see that we have developers from all over the globe responding to this survey! This shows how the JVM ecosystem is thriving everywhere.

# 31. What is your current role?

The vast majority of respondents come from a technical background with 89% of them being either developers, team leaders, or architects. What's more, an important number of C-level employees took time out of their busy schedule to participate in our survey.



54% Software Developer

21% Architect

14% Team Leader

3% C-Level

3% Consultant

2% Project Manager

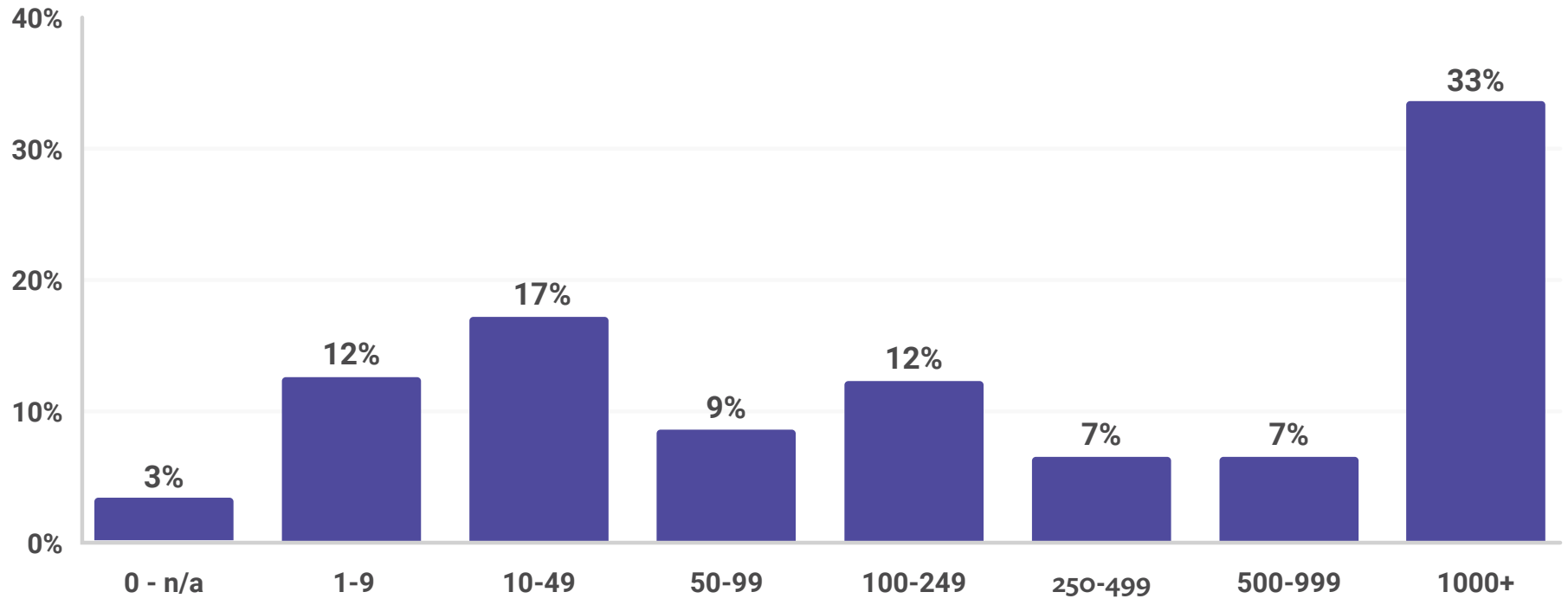
1% Developer Relations

1% Hobbyist

1% Other

# 32. What is the size of your company?

According to the demographics, the JVM ecosystem has a place in enterprises as well as startups! With 41% of the respondents working for a company that has less than 100 employees, it is safe to say that Java has a role to play everywhere.



snyk

**Develop fast. Stay secure.**


Twitter: @snyksec

Web: https://snyk.io


## Office info


### London

1 Mark Square

London EC2A 4EG

### Tel Aviv

40 Yavne st., first floor

### Boston

200 Berkeley, 24th floor

Boston, MA 02116



### Report author

Brian Vermeer (@BrianVerm)

### Reviewers

Simon Maple (@sjmaple)

Eirini-Eleni Papadopoulou (@Esk_Dhg)

### Report design

Growth Labs (@GrowthLabsMKTG)