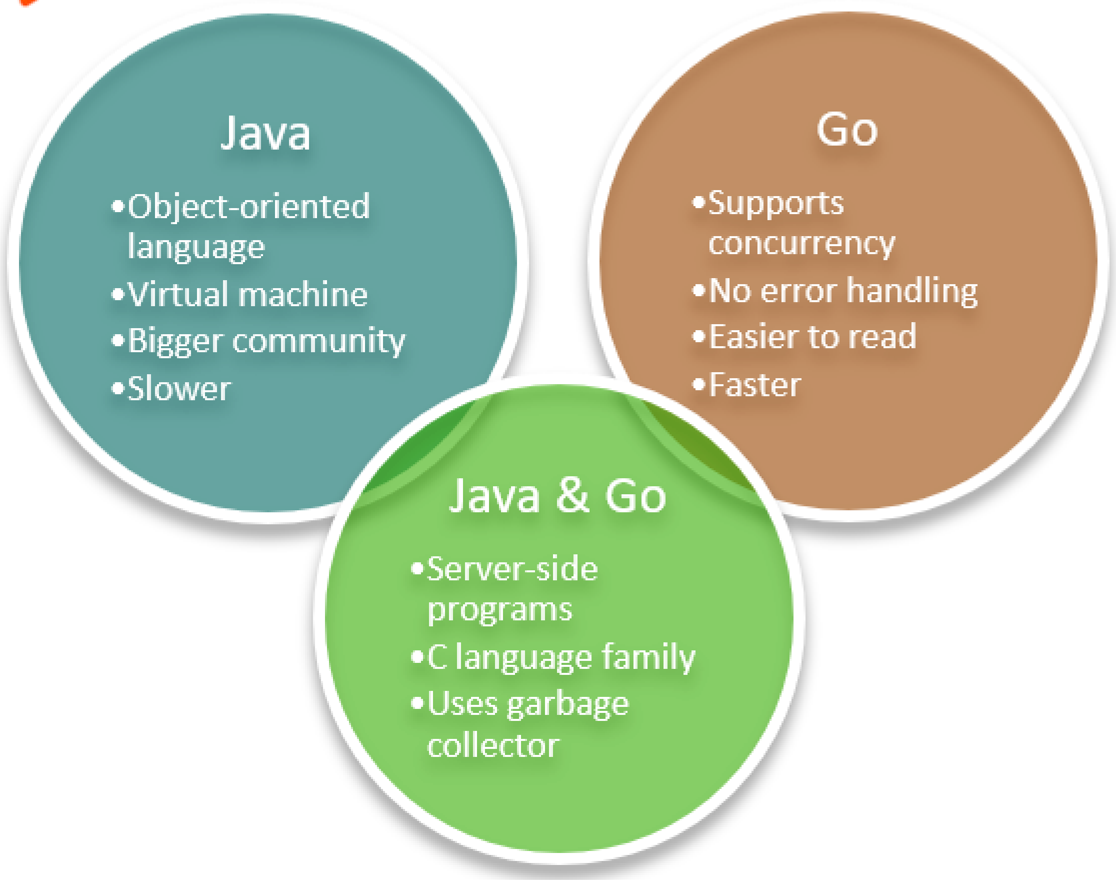
JAVA VS GO: WHAT'S THE DIFFERENCE?

Let's take a look at the differences and similarities in Java and Go, two globally popular programming languages.

Java vs Go: A quick glance

Java is the older and more widely used programming language. It is object-oriented, has a larger community—thus library, and relies on the Java virtual machine (JVM).

Go, or Golang, is newer, supports concurrency, is more readable, and is not object-oriented.



The Java programming language

Java is old. It was developed by James Gosling at Sun Microsystems and released in 1995 as a part of Sun's Java Platform.

Java basics

Java is a general-purpose programming language that utilizes classes and is object-oriented. Java is designed to run anywhere. It uses its Java Virtual Machine to interpret compiled code. The JVM acts as both an interpreter and an error detector.

Programming in Java can be easy because Java has many libraries built on top of it, making it easy to find code already written for a specific purpose.

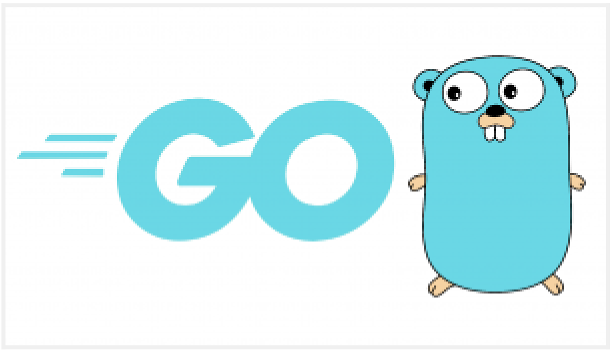
(Learn more about how Java works.)

Java popularity

With its ties to Sun Microsystems, Java was the most widely used server-side language. Though that is no longer the case, with Python surging in popularity. Still, Java reigned supreme for long enough, garnering a large community that, even today, continues to support it.

Go Programming

Go is new. It was designed at Google in 2007 by Robert Griesemer, Rob Pike, and Ken Thompson.



Go basics

Go is a statically typed, compiled programming language. It is an open-sourced language maintained by Google.

Like Java, Go is also a server-side programming language. It is a part of the C-Family programming languages, so it shares similar syntax. Similar to Java, it uses a garbage collector to handle memory leaks.

Go improves over Java

The Golang creators wanted to improve on what already existed. One of those elements was readability. To that end, there are a few changes Go makes:

- The Go library is smaller, making sifting through it easier.
- It does away with the colon at the ends of lines.
- It doesn't require the use of brackets and parentheses.
- Go has no error handling. (It is built for people who already know how to code.)

While easier to read and type, its simplicity comes at a cost of leaving greater room for errors in the code, meaning there will be more time spent debugging the code.

Speed vs platform dependency

Go is faster than Java on almost every benchmark. This is due to how it is compiled: Go doesn't rely on a virtual machine to compile its code. It gets compiled directly into a binary file.

On a benchmark test to calculate factorials, by Sunny Radadiya, Go performed better than Java.

Go speed:

```
Output:
Factorial   Time To calculate factorial
10000       0.03 seconds
50000       0.41 seconds
100000      2.252 seconds
500000      68.961 seconds
1000000     224.135 seconds
```

Java speed:

```
Output:
Factorial   Time To calculate factorial
10000       0.112 seconds
50000       1.185 seconds
100000      2.252 seconds
500000      89.500 seconds
1000000     385.868 seconds
```

Because Go does not have the VM, it is faster. But that that VM also helps Java work on more platforms. Aliaksei Novikau, Senior Software Engineer at Spiral Scout, considers:

Java less platform dependent than Go because Go requires you to create a binary file every time you compile code for a single platform.

Choosing Go or Java

Frankly, I am a fan of adopting new technologies. Where people think new technology lacks is generally where the new tech has improved over previous versions.

It is also good practice to make a habit of learning new things. The choice to adopt a new technology should occur because of merit and not an aversion to learning.

Really, though, Go is not that new of a language. It's not a new technology that's hot off the shelf, looking for prototype users. It works and works well. It is faster, sleeker, and supports concurrency.

If you're jumping into coding for the first, Go is the way to go. If you're just wanting to get something done, and it just has to work for the sake of working, then Java might be the solution. If you're going to spend day-in and day-out looking at one or the other, then it's time to learn Go.

Additional resources

For more on this topic, check out the BMC DevOps Blog and these articles:

- The State of Java Today
- Java Developer Roles and Responsibilities
- Would you Like an API with your Java?
- Top 17 Programming & Software Dev Conferences of 2020
- Top DevOps Books to Read in 2020