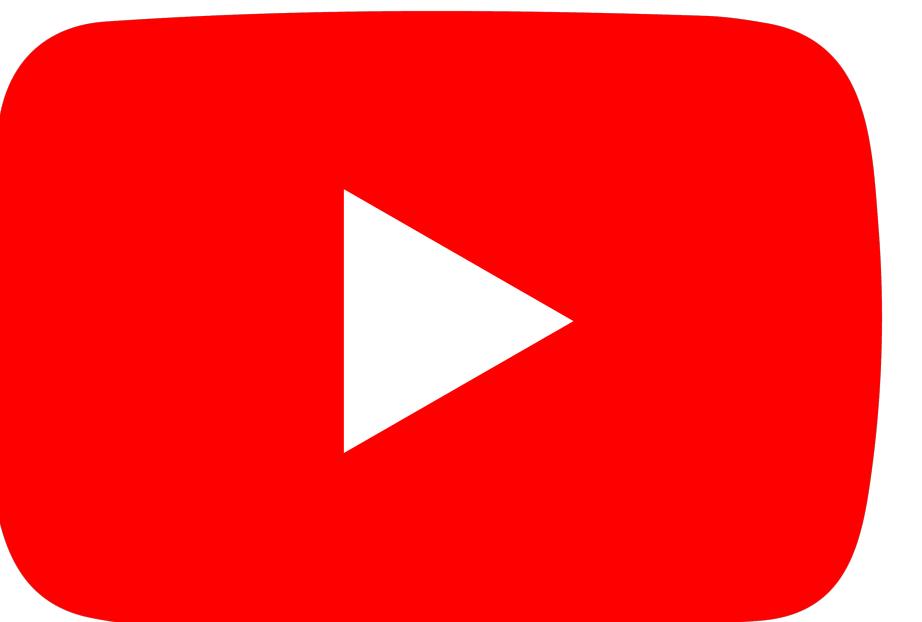




# About Me

Nelson  
Software Engineer

AMIGOS  
CODE



BRIGHT  
NETWORK

# Java Bootcamp Curriculum

- Intro Java
- Basics
- Object Oriented Programming - OOP
- Data structures
- Testing
- SQL
- Spring Boot
- Connecting to Databases
- Full Stack with React.js

# What is Java

- Object Oriented Programming Language
- Created by James Gosling in 1995
- 3 billion devices run Java programs
- Similar to C++ but easier
- Very popular

# Versions

| Version           | Release date   | End of Free Public Updates <sup>[1][6][7][8]</sup>  | Extended Support Until   |
|-------------------|----------------|---|--|
| JDK Beta          | 1995           | ?   | ?  |
| JDK 1.0           | January 1996   | ?   | ?  |
| JDK 1.1           | February 1997  | ?   | ?  |
| J2SE 1.2          | December 1998  | ?   | ?  |
| J2SE 1.3          | May 2000       | ?   | ?  |
| J2SE 1.4          | February 2002  | October 2008  | February 2013  |
| J2SE 5.0          | September 2004 | November 2009   | April 2015   |
| Java SE 6         | December 2006  | April 2013  | December 2018<br>December 2026, paid support for Azul Platform Core <sup>[9]</sup> |
| Java SE 7         | July 2011      | April 2015  | July 2022  |
| Java SE 8 (LTS)   | March 2014     | <b>January 2019 for Oracle (commercial)</b><br>December 2030 for Oracle (non-commercial)<br>December 2030 for Azul<br>At least May 2026 for AdoptOpenJDK<br>At least May 2026 for Amazon Corretto | December 2030  |
| Java SE 9         | September 2017 | March 2018 for OpenJDK  | N/A  |
| Java SE 10        | March 2018     | September 2018 for OpenJDK  | N/A  |
| Java SE 11 (LTS)  | September 2018 | September 2026 for Azul<br>At least October 2024 for AdoptOpenJDK<br>At least September 2027 for Amazon Corretto<br>At least October 2024 for Microsoft <sup>[10][11]</sup>                       | September 2026<br>September 2028 for Azul <sup>[9]</sup>                           |
| Java SE 12        | March 2019     | September 2019 for OpenJDK  | N/A  |
| Java SE 13        | September 2019 | March 2020 for OpenJDK  | N/A  |
| Java SE 14        | March 2020     | September 2020 for OpenJDK  | N/A  |
| Java SE 15        | September 2020 | March 2021 for OpenJDK<br>March 2023 for Azul <sup>[9]</sup>  | N/A  |
| <b>Java SE 16</b> | March 2021     | September 2021 for OpenJDK  | N/A  |
| Java SE 17 (LTS)  | September 2021 | September 2029 for Azul<br>At least September 2027 for Microsoft  | September 2029 or later<br>September 2031 for Azul Platform Prime                  |
| Java SE 18        | March 2022     | September 2022 for OpenJDK  | N/A  |

Legend:  Old version  Older version, still maintained  Latest version  Future release

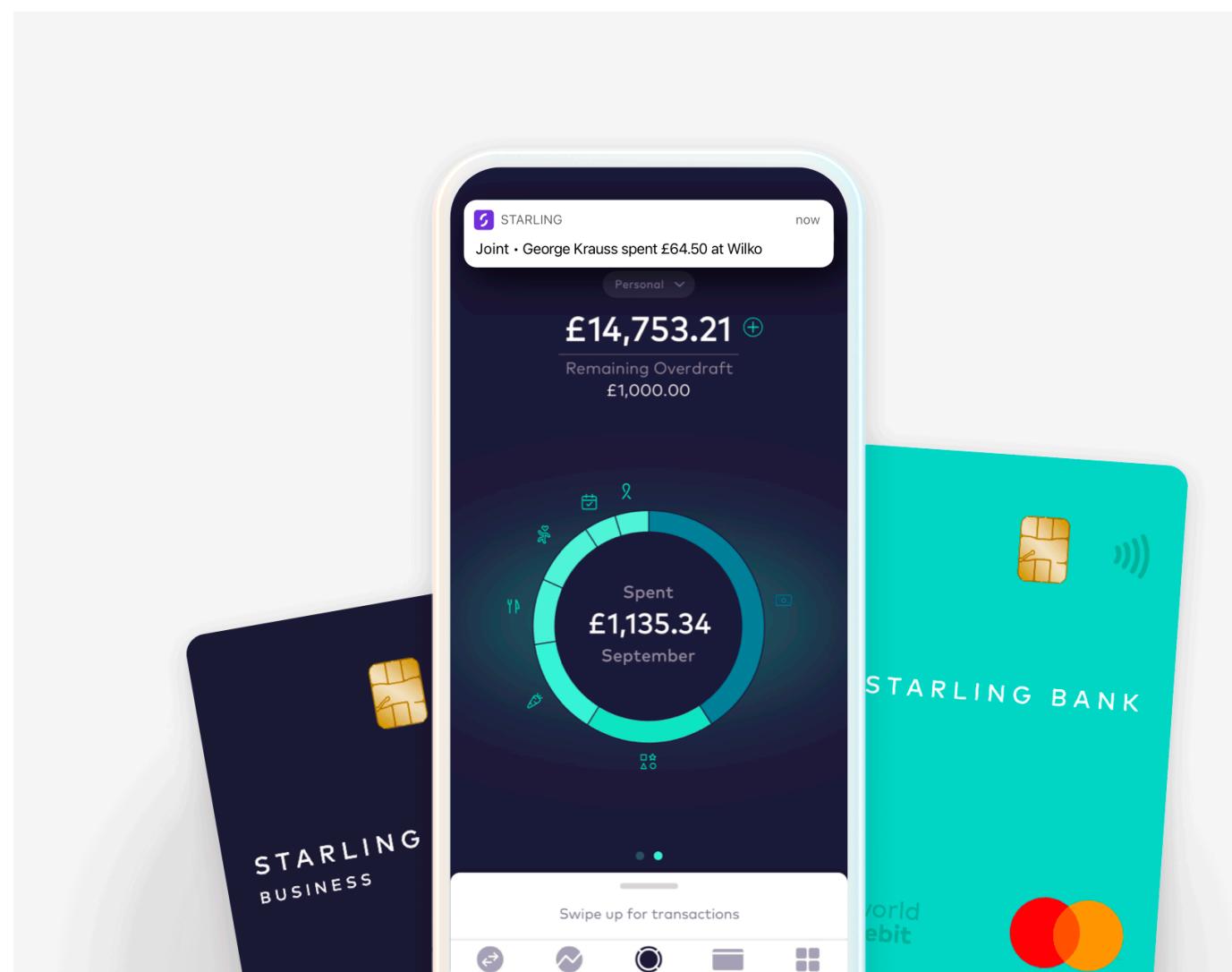
[https://en.wikipedia.org/wiki/Java\\_version\\_history](https://en.wikipedia.org/wiki/Java_version_history)

# What can you build with Java

- Mobile apps for Android
- Desktop App using Java Swing, JavaFX
- Enterprise Applications
- Games such as **Minecraft**
- Cloud based applications
- Web Applications
- and much more

# Who uses Java

- Big and Small companies
- Very popular amongst start ups

A screenshot of the IntelliJ IDEA Java IDE. The left side shows the project structure with a tree view of files and folders. The right side shows the code editor with Java code for a class named 'CodeGenException'. The code includes methods for constructor and message handling. The interface also features tabs for 'TODO', 'Services', 'Terminal', 'Version Control', and 'Java Enterprise'.

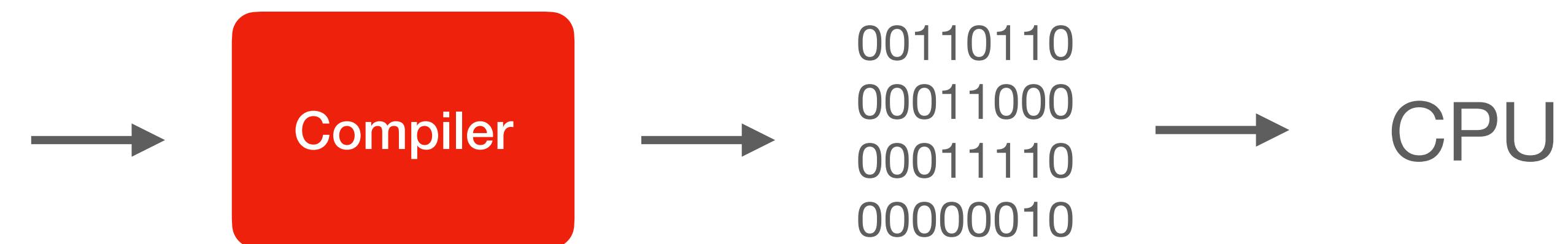
# Java is Not Javascript

- **Java** is a compiled/interpreted language
- Backend
- **Javascript** is an interpreted language
- Mostly used on web browsers and some backend with NodeJS

# Compiler

- A compiler is a special program that processes statements written in a particular programming language and turns them into machine language or "code" that a computer's processor uses.

```
4 if (!isAdult) {  
5     console.log("is not an adult :( ")  
6 } else {  
7     console.log("is adult :)")  
8 }
```



# Interpreter

- The interpreter reads each statement of code and then converts or executes it directly

```
4 if (!isAdult) {  
5     console.log("is not an adult :( ")  
6 } else {  
7     console.log("is adult :)")  
8 }
```



Interpreter

# Compiled vs Interpreted

## Compiled

Code can be executed directly by computer's CPU

Code must be transformed into machine readable instructions

Runs faster

Better Performance

## Interpreted

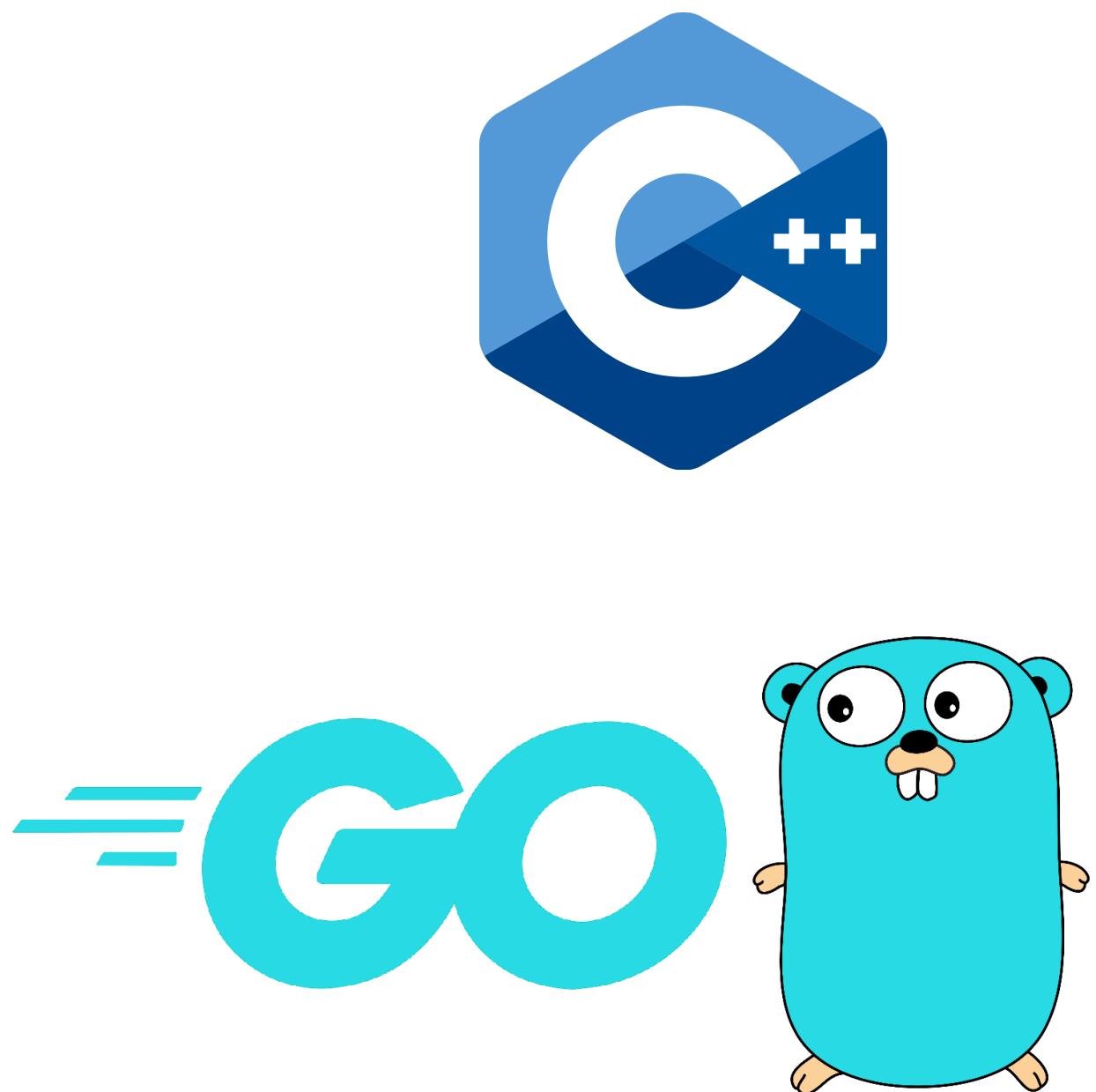
Another program executes code

No transformation needed

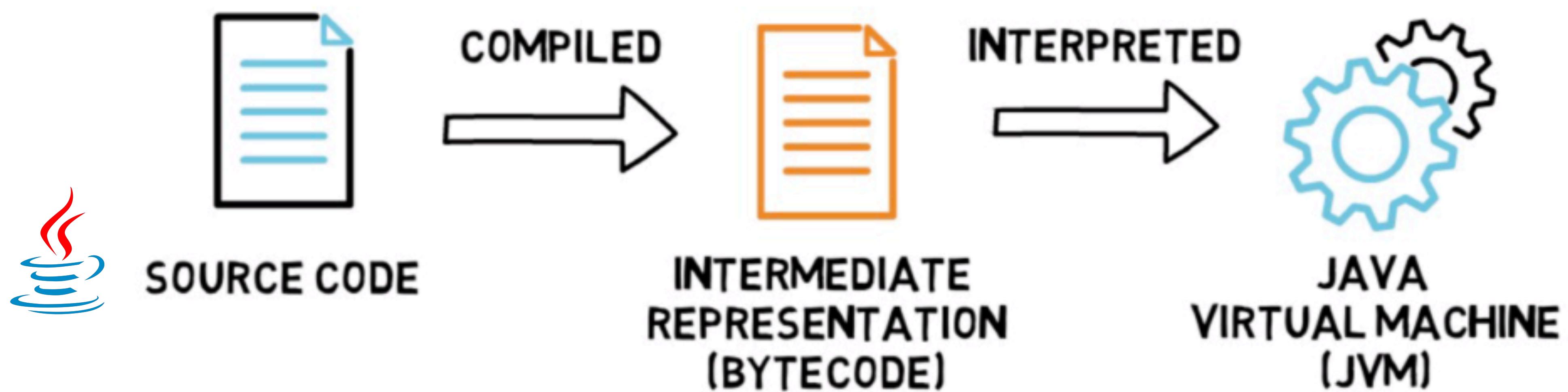
Slower

Faster for Development

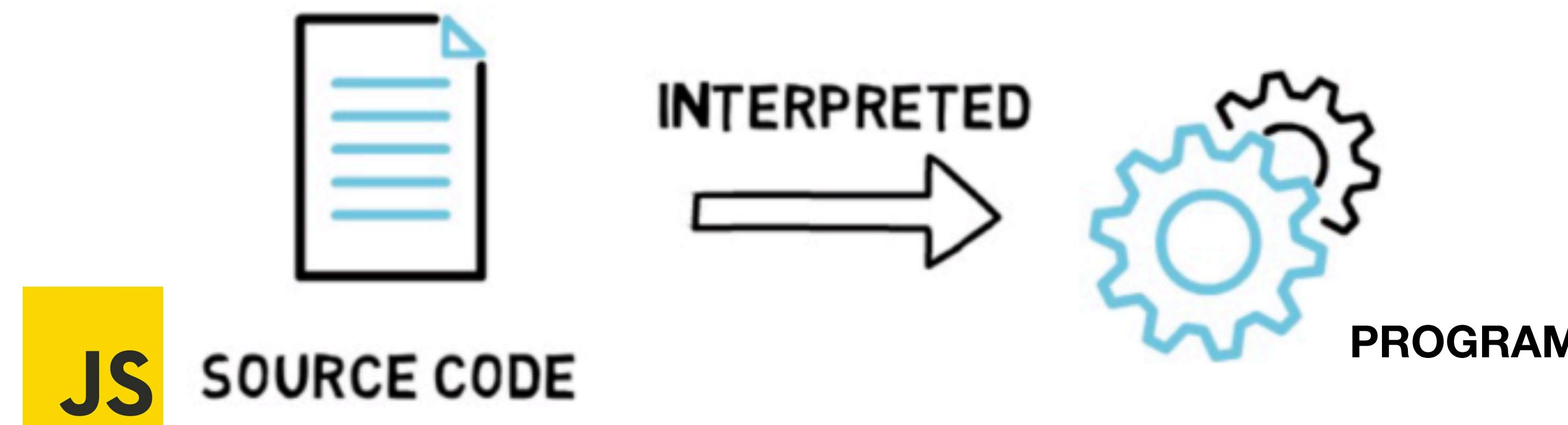
# Compiled vs Interpreted



# JAVA and JVM



# JS and Interpreter



# Static vs Dynamic Type checking

- Type - AKA data type, is a classification identifying one of various **types of data**. Strings, Integers, Objects, Arrays, Maps ...
- The process of verifying and enforcing the constraints of types—*type checking*—may occur either at **compile time** (a static check) or at **run-time** (dynamic check).

# Static Type checking

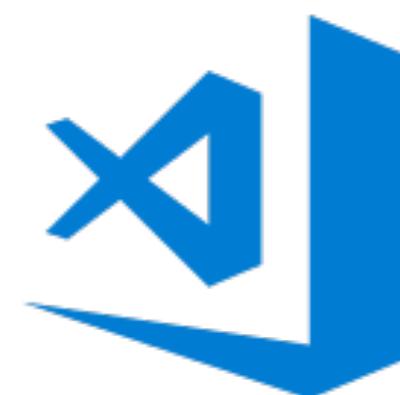
- A language is statically-typed if the type of a variable is known at **compile time** instead of at runtime
- Allows many type errors to be caught early in the development cycle.

# Dynamic Type checking

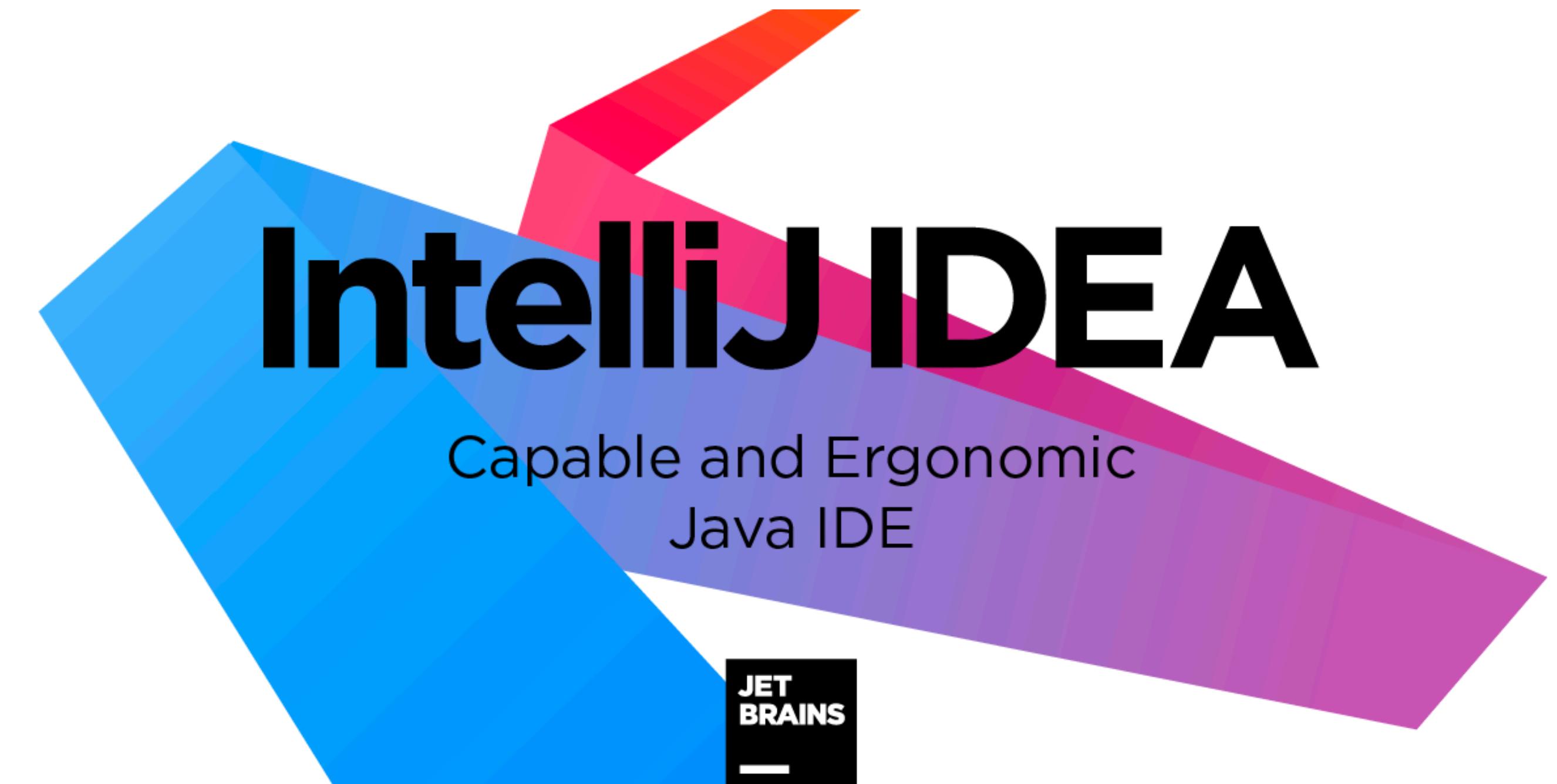
- is the process of verifying the type safety of a program at **runtime**. Common dynamically-typed languages include Groovy, JavaScript, Lisp, Lua, Objective-C, PHP, Prolog, Python, Ruby, Smalltalk and Tcl.

# How do we write Java code

- With any text editor



Visual Studio Code

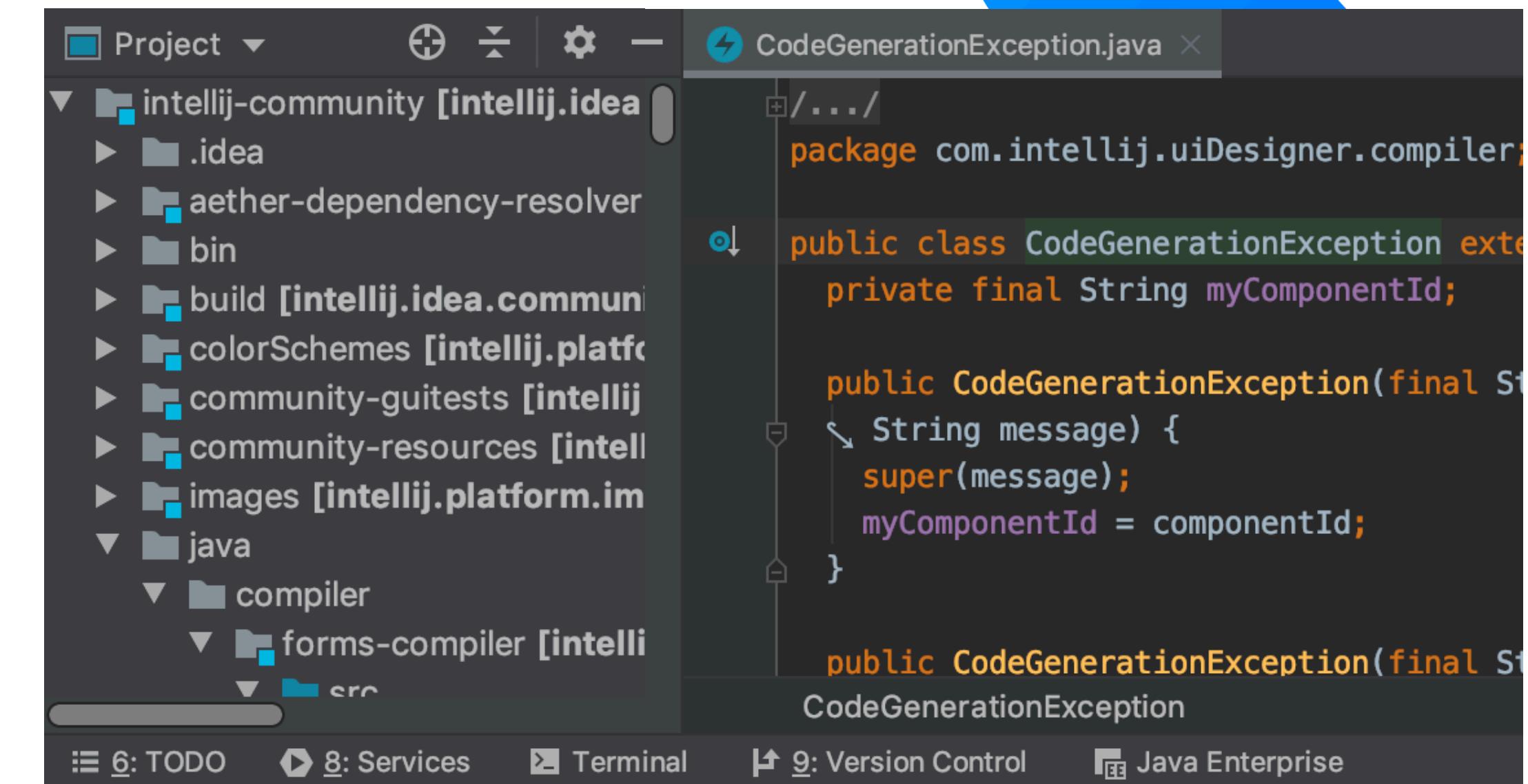


# Most popular Java IDE

- By JetBrains

**2 Versions Available**

- Community Edition
- Ultimate

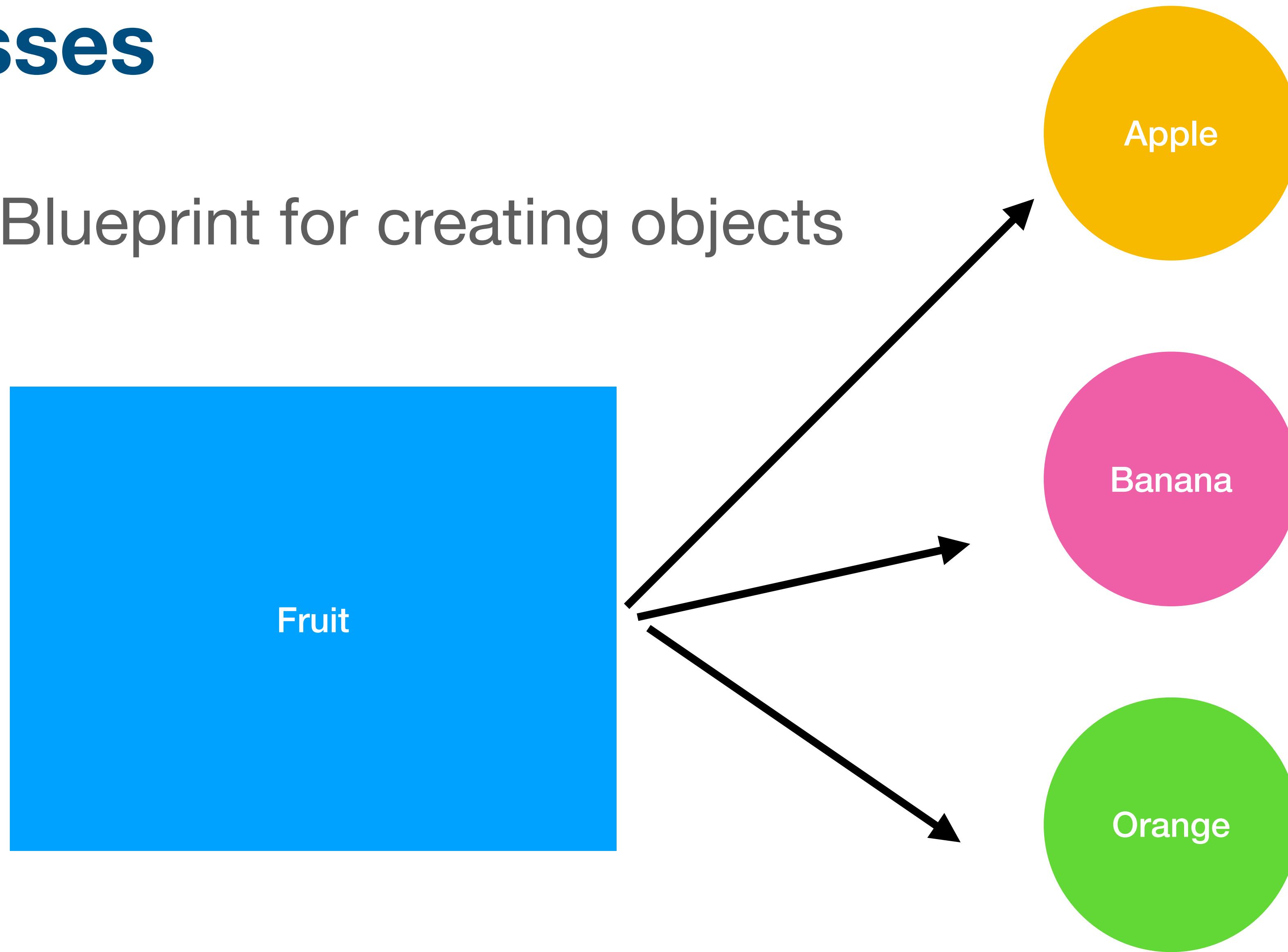


# OOP

- is about creating objects that contain both data and methods
  - OOP is faster and easier to execute
  - OOP provides a clear structure for the programs
  - OOP helps to keep the Java code DRY "Don't Repeat Yourself", and makes the code easier to maintain, modify and debug
  - OOP makes it possible to create full reusable applications with less code and shorter development time

# Classes

- Blueprint for creating objects

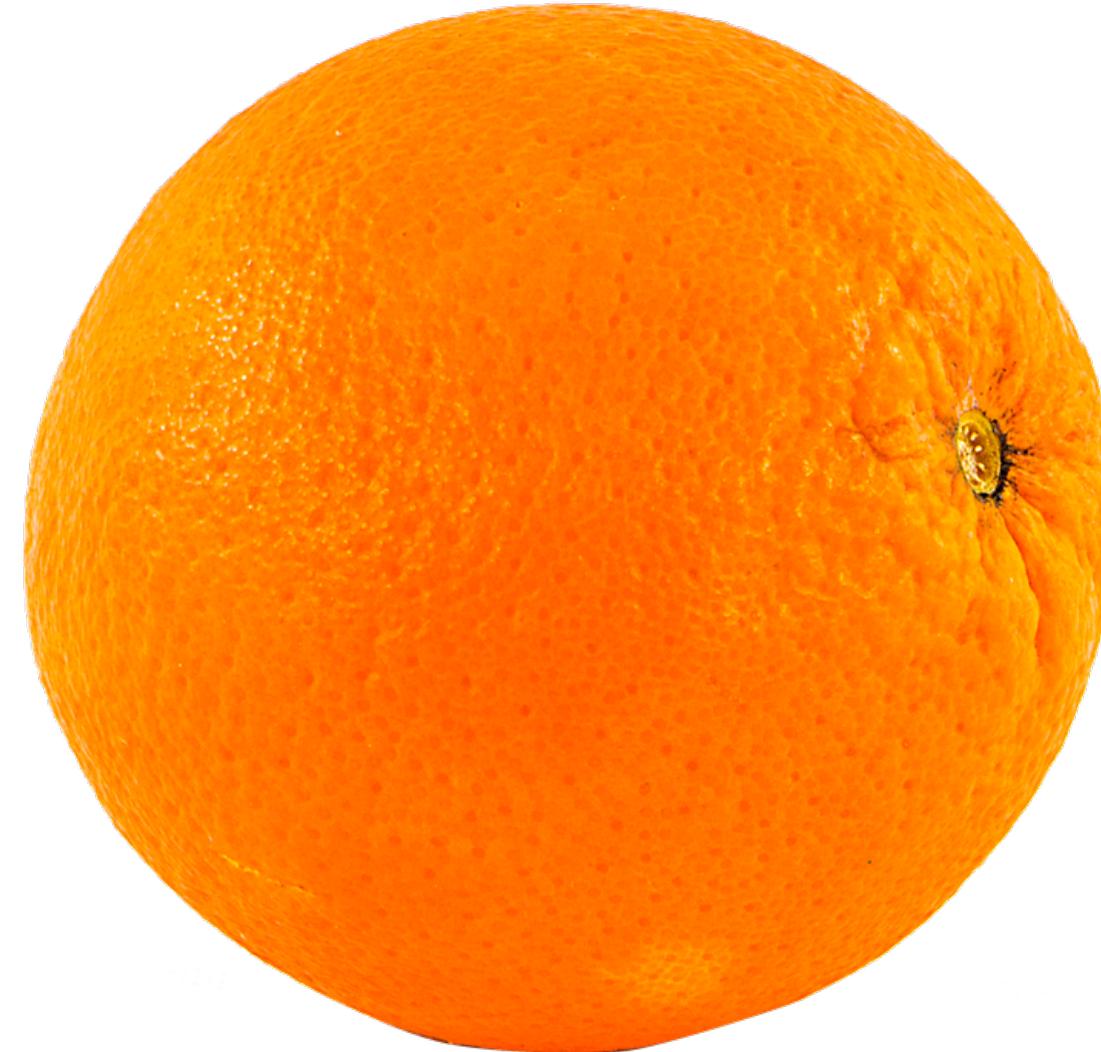


# Classes

- Properties and behaviours

# Objects

- The real thing



# Day 1 - Getting Started

- Get IntelliJ and our first Java code
- Javac and bytecode
- Understanding the syntax & Reserved Keywords
- Comments
- Variables
- Primitive and Reference Types
- Data Types
- String class
- Arrays
- `++` & `--`
- Loops
- Break and Continue
- Packages and Import Statement
- Comparison Operators
- Logical Operators
- If statement & Ternary
- Switch Statement
- Take user input

# Day 2 - Basics of Java

- The Java Syntax
- Access Modifiers
- Methods
- Enums
- Working with Dates
- Handling Error
- Working with Files
- Optionals

# Day 3 - OOP Object Oriented Programming

- Classes and Objects
- ToString
- Equals vs ==
- Equals and GetHashCode
- **Static**
- Pass by Value vs Reference
- **Inheritance**
- **Interfaces**
- **Polymorphism**

# Day 4 - Data Structures and Streams

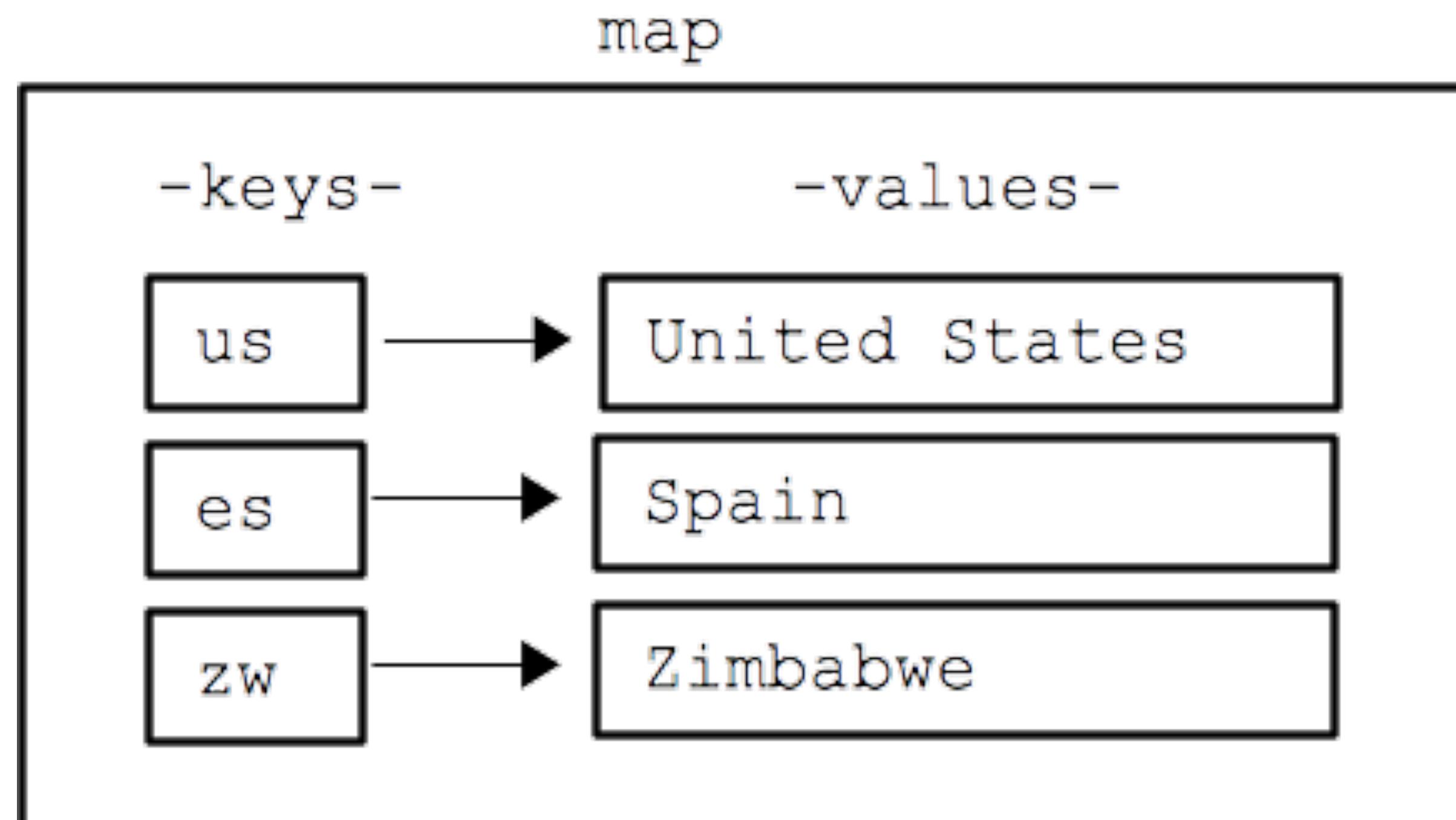
- Lists
- Maps
- Stacks
- Queues
- Imperative vs Declarative Approach
- Streams

# What is Data Structures

- Data Structures are a specialized means of organizing and storing data in computers in such a way that we can perform operations on the stored data more efficiently

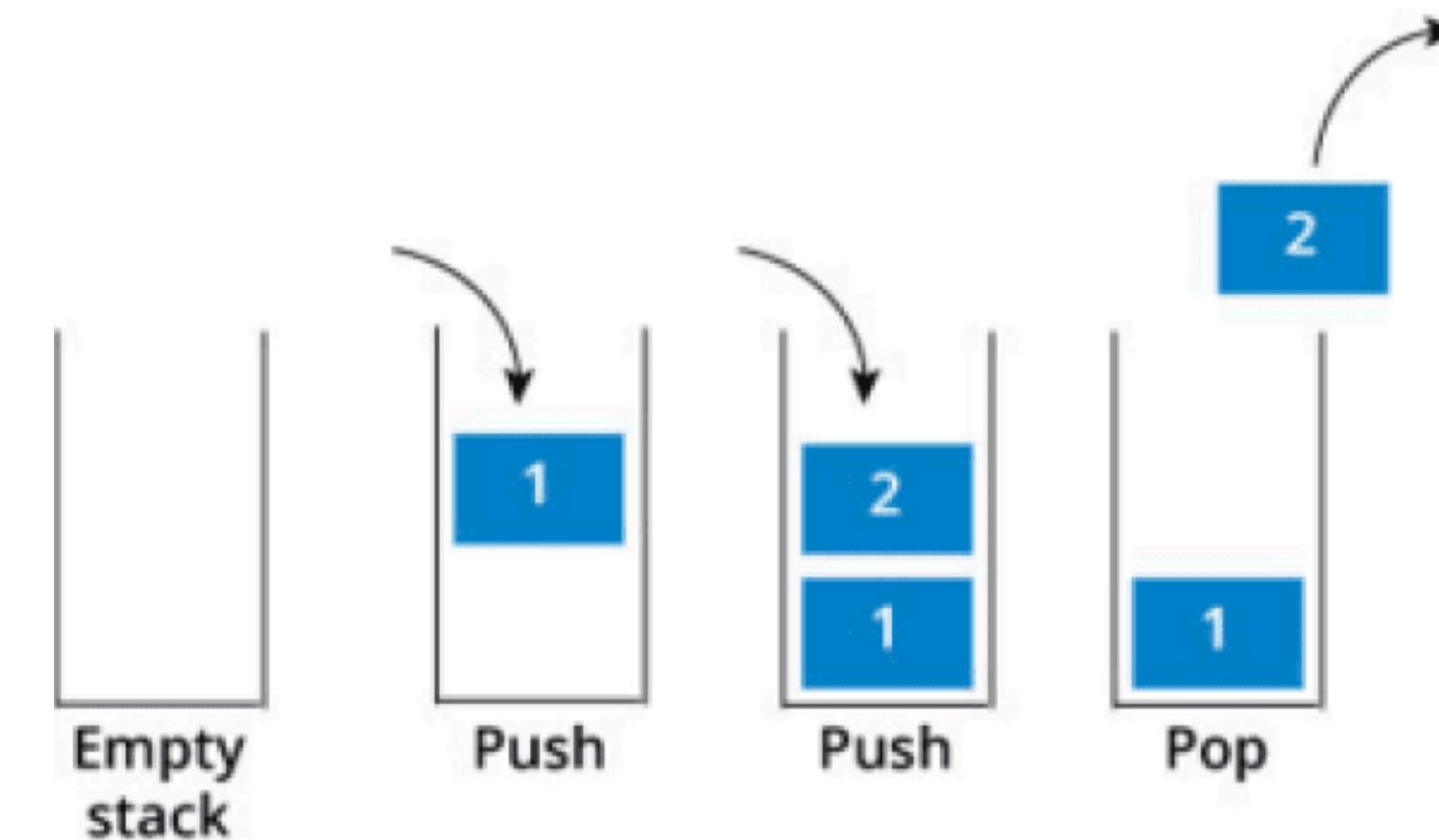
# Maps

- Key Value Pair



# Stack

- LIFO Datastructures
- Push
- Pop
- Peek



# Queue

- FiFO Datastructures
- Push
- Pop
- Peek



# Day 5 - Team Project

- Git and Github
- Implement a database management system for a car rental company

# Day 6 - Testing

- What is testing
- JUnit
- AssertJ
- Mocking
- Test Driven Development

# Day 7 - Databases and SQL

- Getting Started with DB
- Relational Databases
- Queries
- Joins
- Aggregate Functions

# Day 8 Spring Boot

- Getting Started with Spring
- Restful APIs

# Day 9 Spring Boot and DB

- Connect Spring Boot backend to database

# Day 10 Getting Started with React

- Create react app
- Build a simple app to connect to backend