Java Bootcamp

BRIGHT NETWORK

About Me

Nelson
Software Engineer

Java Bootcamp Curriculum

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

BRIGHT NETWORK

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[1][6][7][8]	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 December 2026, paid support for Azul Platform Core[9]
Java SE 7	July 2011	April 2015	July 2022
Java SE 8 (LTS)	March 2014	**January 2019 for Oracle (commercial)** December 2030 for Oracle (non-commercial) December 2030 for Azul At least May 2026 for AdoptOpenJDK 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 At least October 2024 for AdoptOpenJDK At least September 2027 for Amazon Corretto At least October 2024 for Microsoft[10][11]	September 2026 September 2028 for Azul[9]
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 March 2023 for Azul[9]	N/A
Java SE 16	March 2021	September 2021 for OpenJDK	N/A
Java SE 17 (LTS)	September 2021	September 2029 for Azul At least September 2027 for Microsoft	September 2029 or later 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

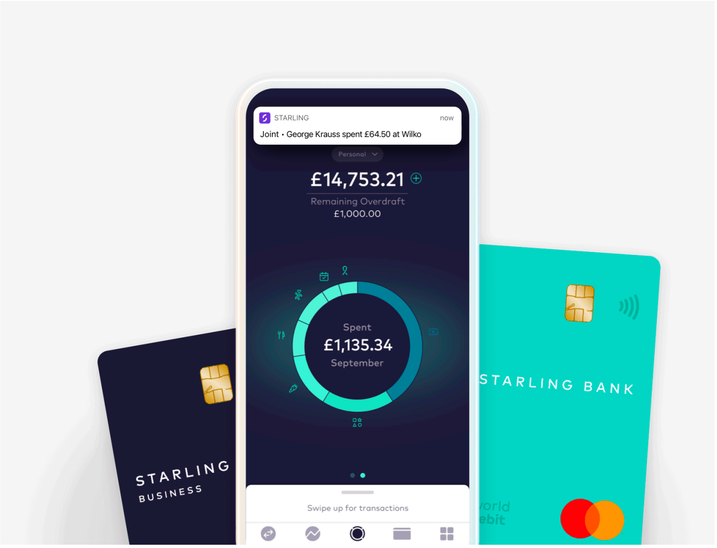
BRIGHT NETWORK

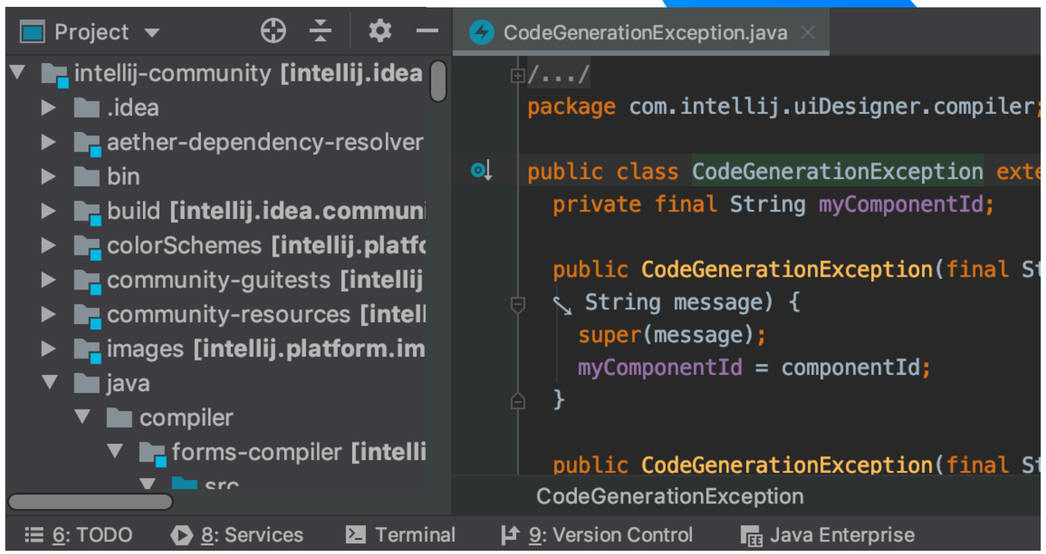
What can you build with Java

- Mobile apps for Android
- Desktop App using Java Swing, JavaFX
- Entreprise 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





IntelliJ IDEA
Capable and Ergonomic
Java IDE
JET BRAINS

BRIGHT NETWORK

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

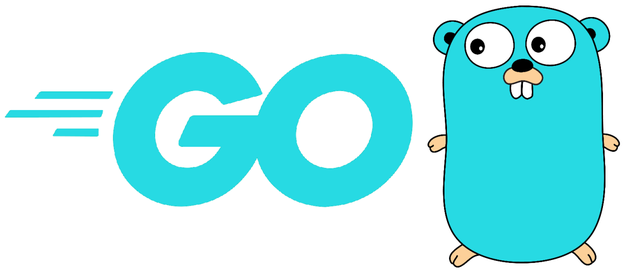
- The interpreter reads each statement of code and then converts or executes it directly

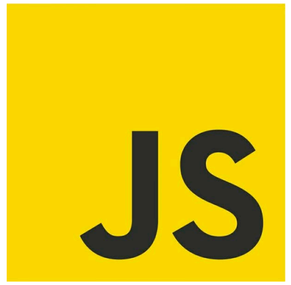
```
4 if (!isAdult) {
5     console.log("is not an adult :( ")
6 } else {
7     console.log("is adult :)")
8 }
```

Compiled vs Interpreted

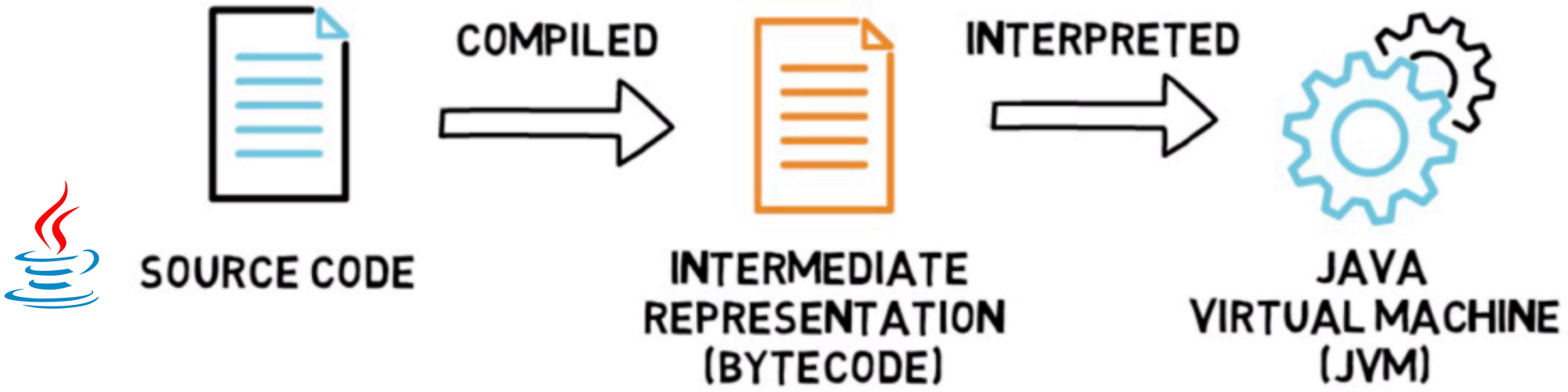
Compiled	Interpreted
Code can be executed directed by computer’s cpu	Another program executes code
Code must be transformed into machine readable instructions	No transformation needed
Runs faster	Slower
Better Performance	Faster for Development

Compiled vs Interpreted

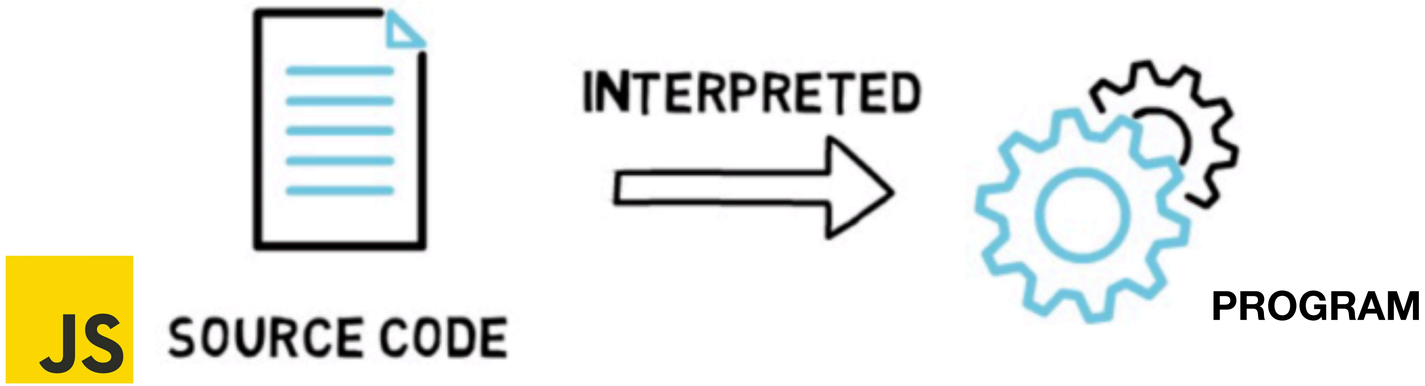




JAVA and JVM



JS and Interpeter



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

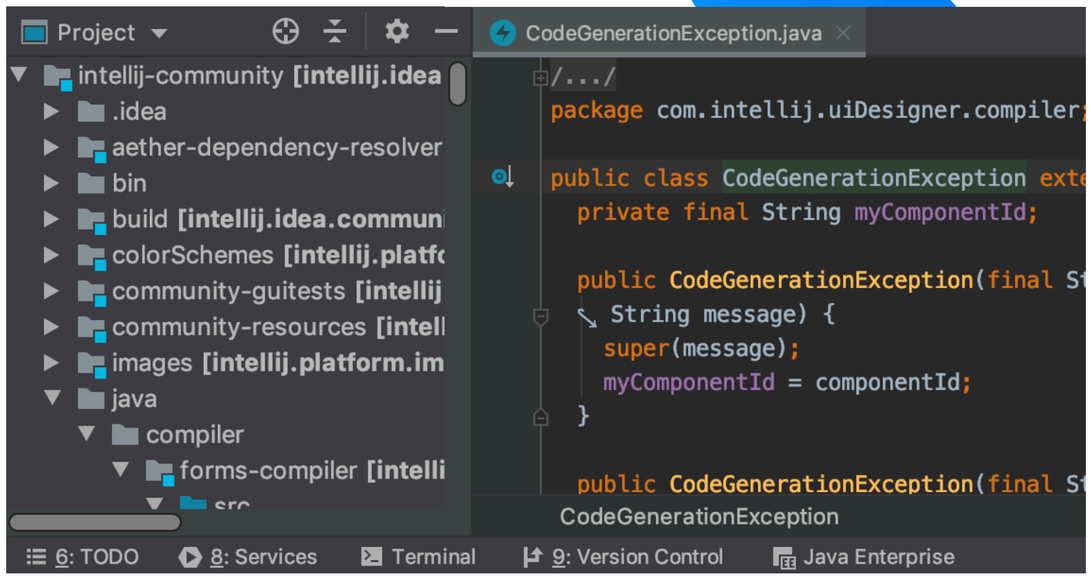
BRIGHT
NETWORK

Most popular Java IDE

- By JetBrains

2 Versions Available

- Community Edition
- Ultimate



BRIGHT NETWORK

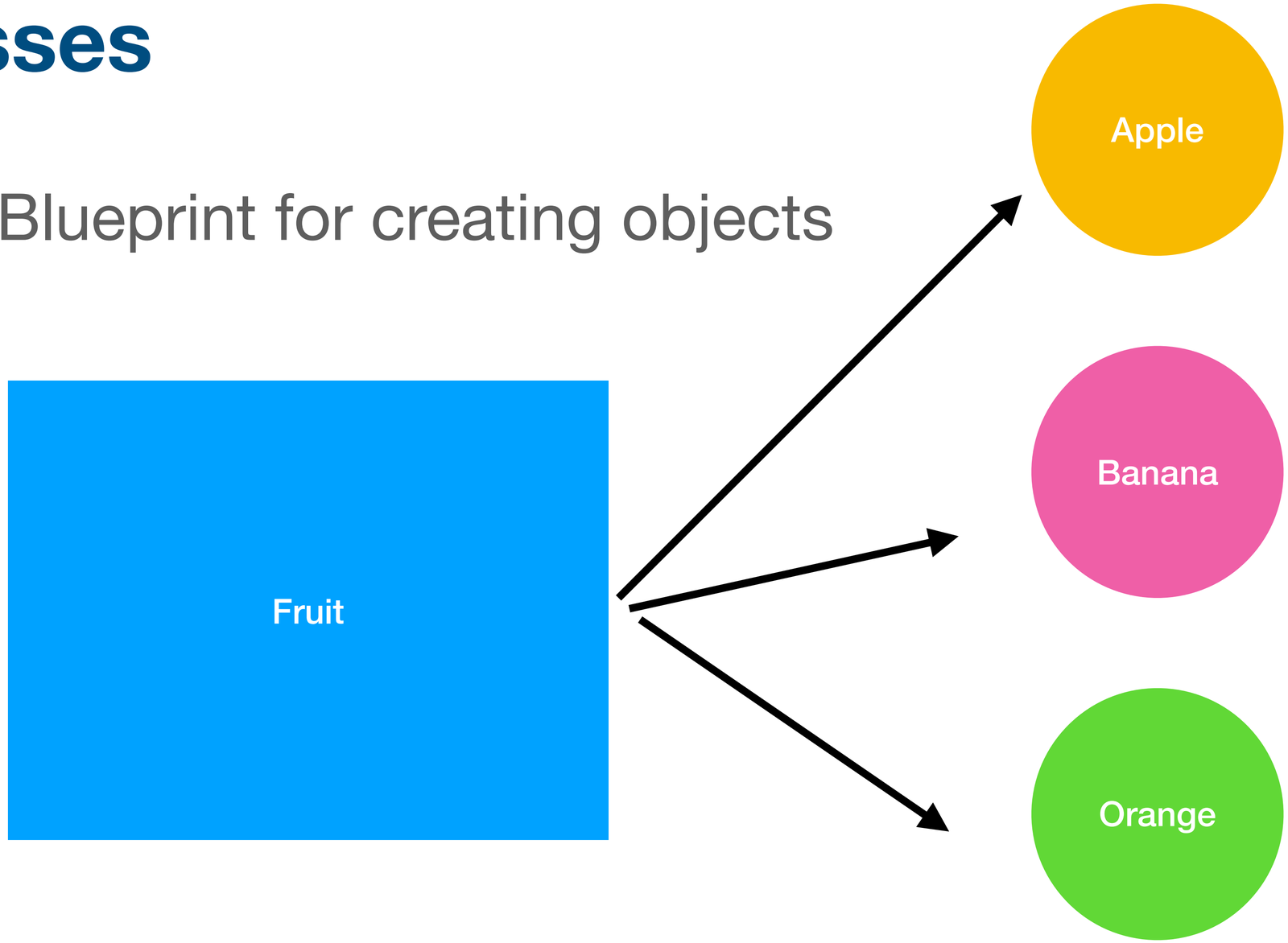
OOP

- 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

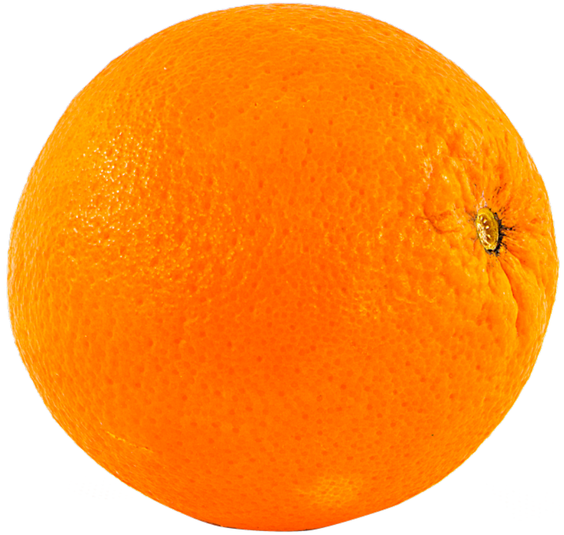


Classes

- 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 HashCode
- **Static**
- Pass by Value vs Reference
- **Inheritance**
- **Interfaces**
- **Polymorphism**

Day 4 - Data Structures and Streams

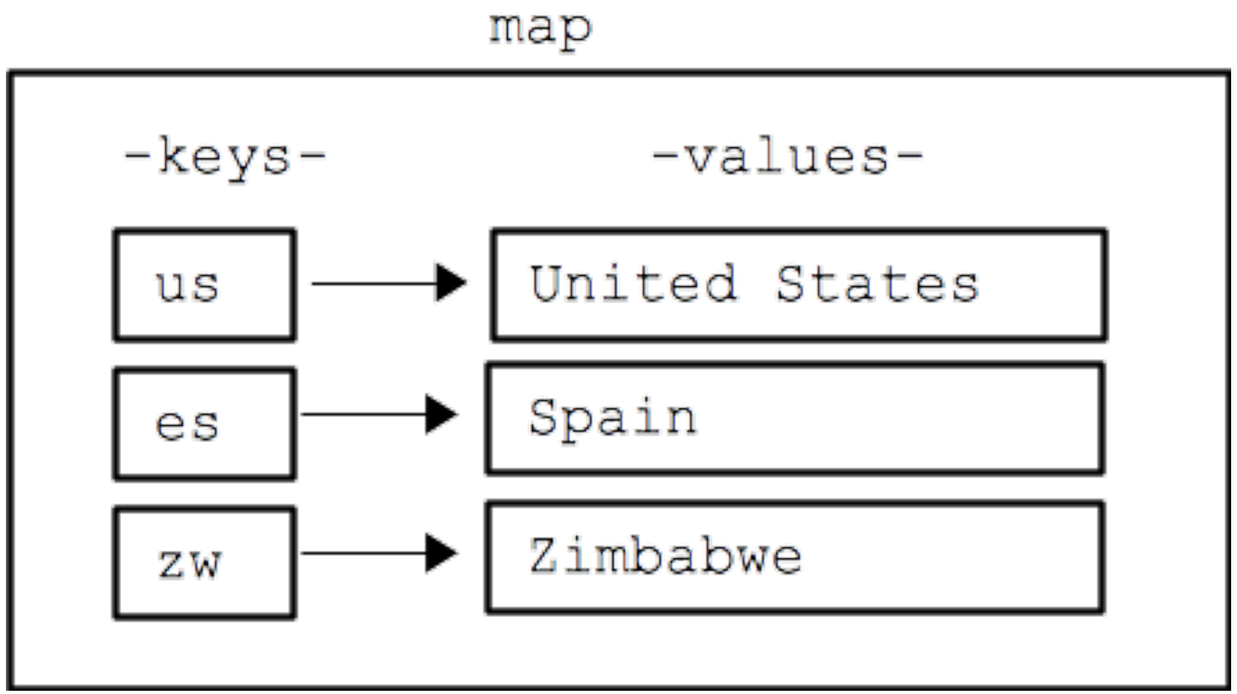
- Lists
- Maps
- Stacks
- Queues
- Imperative vs Declarative Approach
- Streams

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

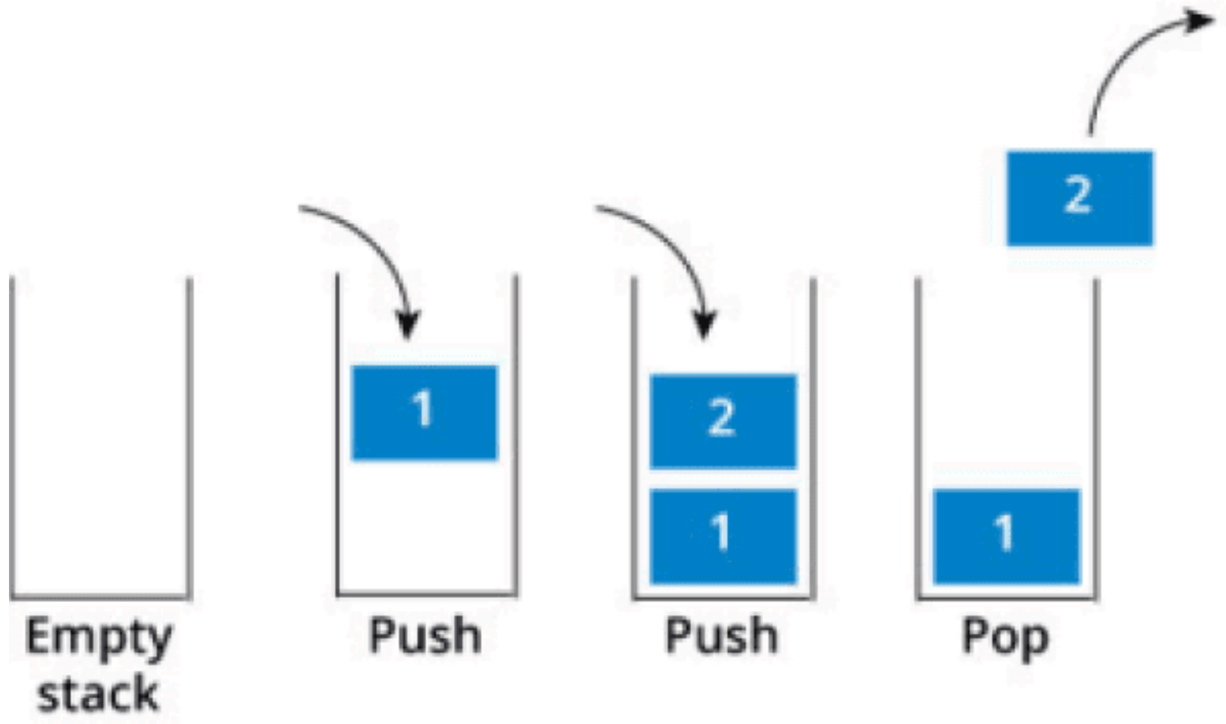
Maps

- Key Value Pair



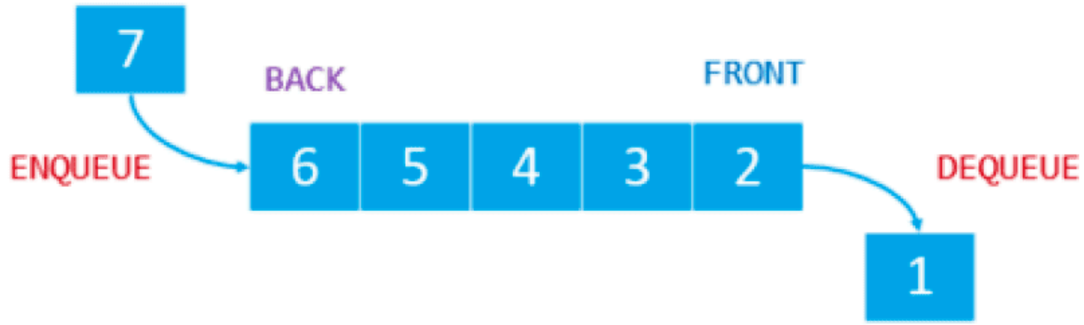
Stack

- LiFO Datastructures
- Push
- Pop
- Peek



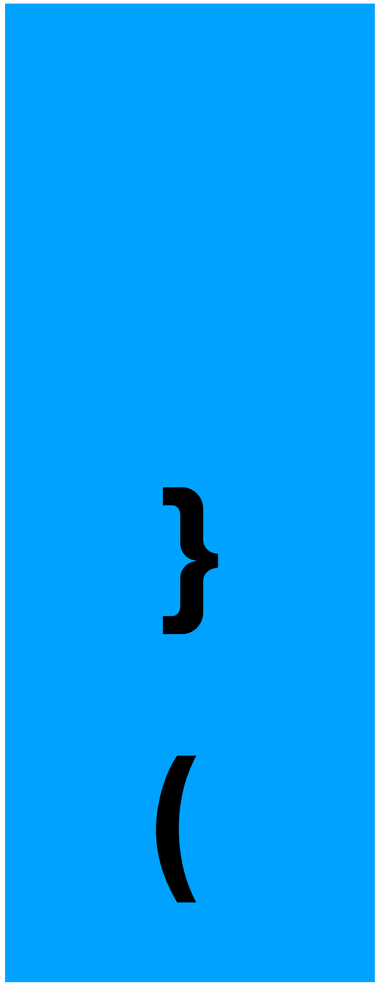
Queue

- FiFO Datastructures
- Push
- Pop
- Peek



```
(){

({}){}
```



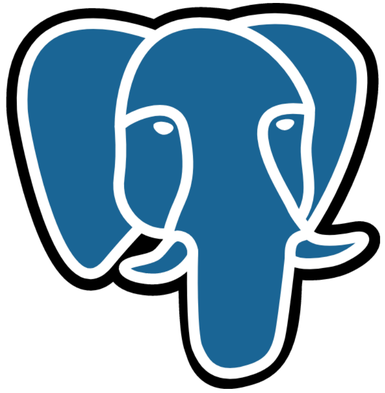
SQL

- Structured Query Language

Database

- Computer Server to store information

Relational Database Management System



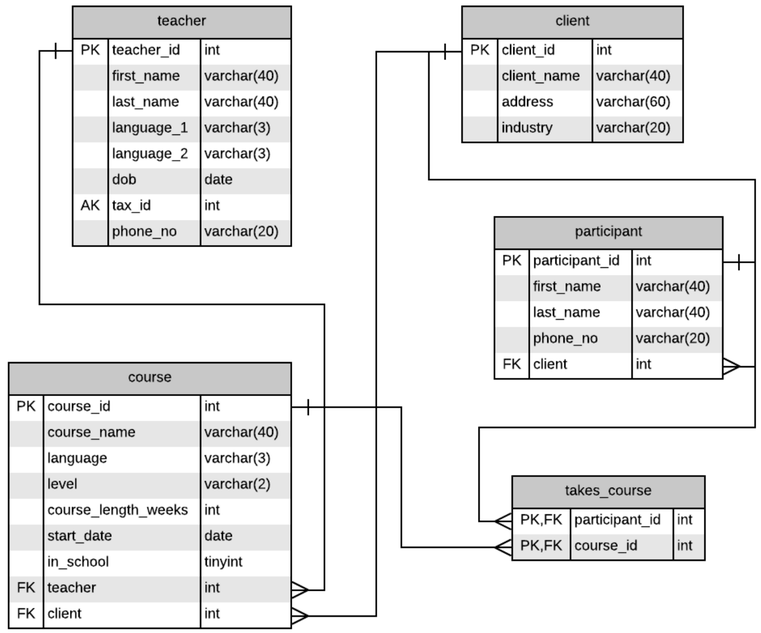
MySQL®

Microsoft® SQL Server®

BRIGHT NETWORK

Relational Database

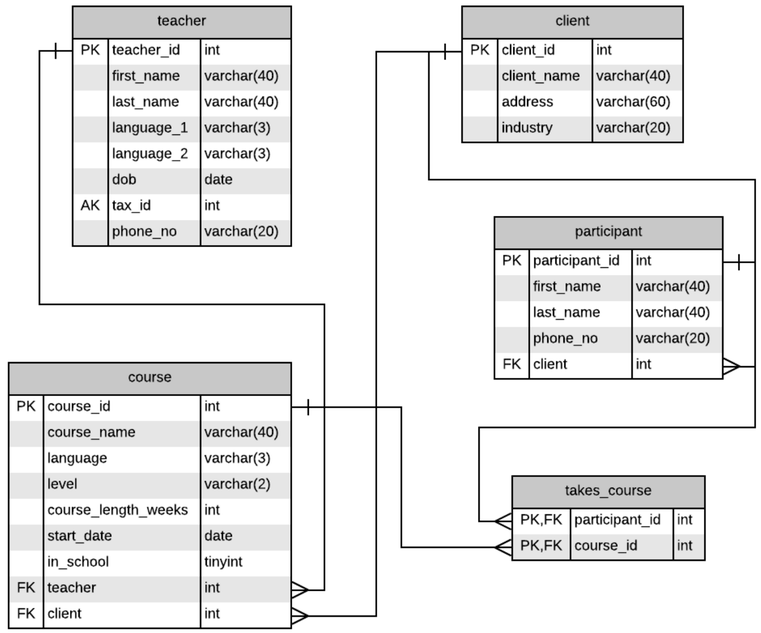
- Tables - Columns and Rows



BRIGHT NETWORK

Relational Database

- Tables - Columns and Rows

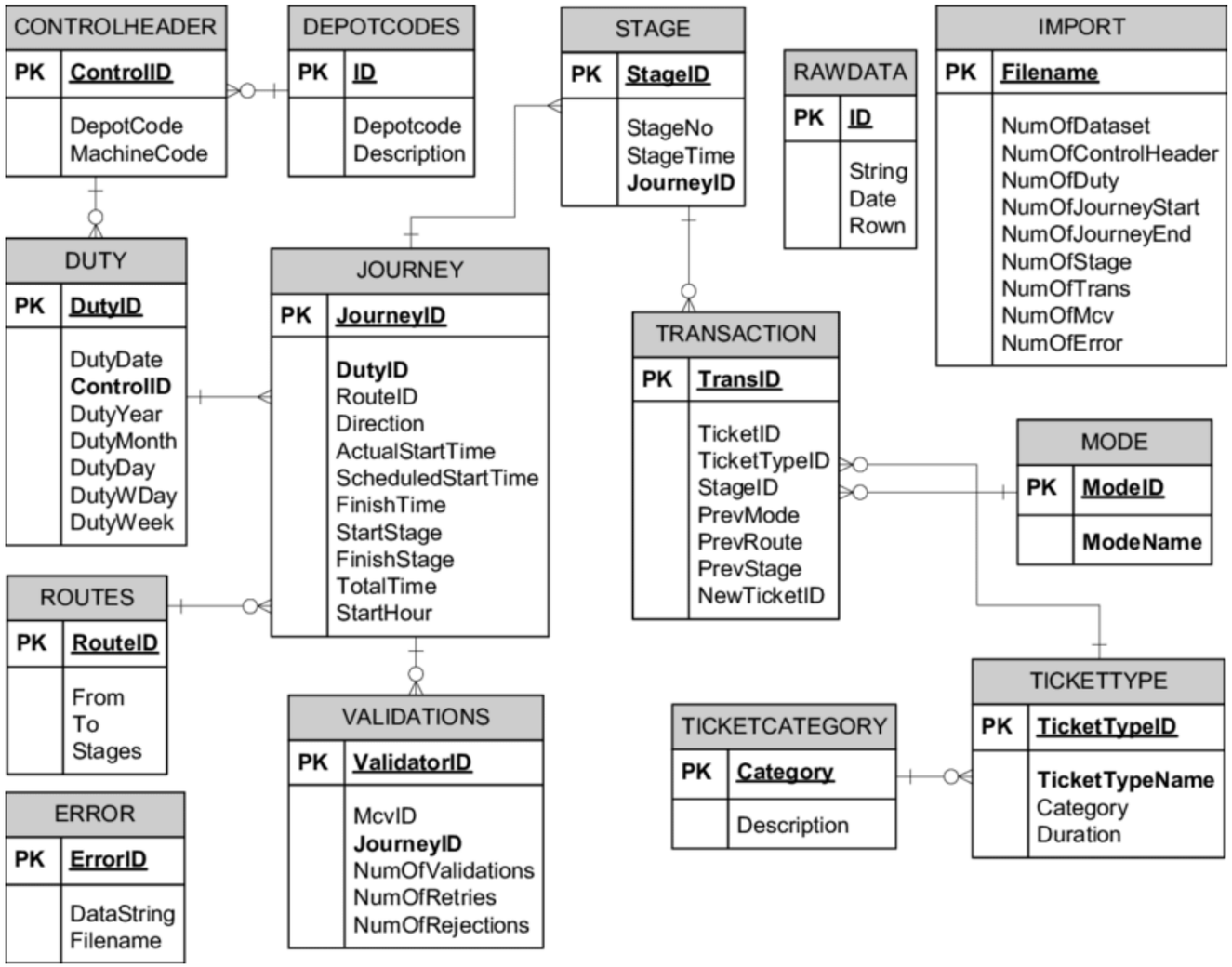


BRIGHT NETWORK

ERD Diagram

- An entity–relationship model describes interrelated things of interest in a specific domain of knowledge. A basic ER model is composed of entity types and specifies relationships that can exist between entities.

ERD Diagram



BRIGHT NETWORK

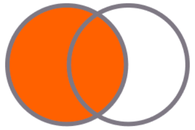
SQL Statements

- Create table
- Insert into
- Select from
- Delete from
- Join

Joins

- Create table
- Insert into
- Select from
- Delete from
- Join - used to **combine** rows from two or more tables

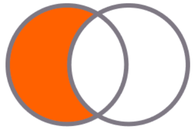
LEFT JOIN



Everything on the left
+
anything on the right that matches

```
SELECT *
FROM TABLE_1
LEFT JOIN TABLE_2
ON TABLE_1.KEY = TABLE_2.KEY
```

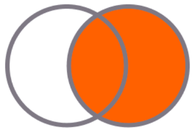
ANTI LEFT JOIN



Everything on the left
that is NOT on the right

```
SELECT *
FROM TABLE_1
LEFT JOIN TABLE_2
ON TABLE_1.KEY = TABLE_2.KEY
WHERE TABLE_2.KEY IS NULL
```

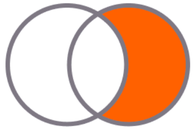
RIGHT JOIN



Everything on the right
+
anything on the left that matches

```
SELECT *
FROM TABLE_1
RIGHT JOIN TABLE_2
ON TABLE_1.KEY = TABLE_2.KEY
```

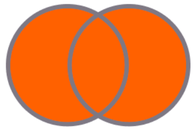
ANTI RIGHT JOIN



Everything on the right
that is NOT on the left

```
SELECT *
FROM TABLE_1
RIGHT JOIN TABLE_2
ON TABLE_1.KEY = TABLE_2.KEY
WHERE TABLE_1.KEY IS NULL
```

OUTER JOIN



Everything on the right
+
Everything on the left

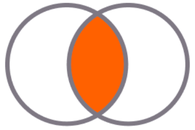
```
SELECT *
FROM TABLE_1
OUTER JOIN TABLE_2
ON TABLE_1.KEY = TABLE_2.KEY
```

ANTI OUTER JOIN

Everything on the left and right
that is unique to each side

```
SELECT *
FROM TABLE_1
OUTER JOIN TABLE_2
ON TABLE_1.KEY = TABLE_2.KEY
WHERE TABLE_1.KEY IS NULL
OR TABLE_2.KEY IS NULL
```

INNER JOIN



Only the things that match on the left AND the right

```
SELECT *
FROM TABLE_1
INNER JOIN TABLE_2
ON TABLE_1.KEY = TABLE_2.KEY
```

CROSS JOIN

All combination of rows from the right and the left (cartesean product)

```
SELECT *
FROM TABLE_1
CROSS JOIN TABLE_2
```

Day 5 - Team Project

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

Spring Boot

- Java Framework For Building Backend and Full Stack Applications
- Lots of helper projects to get you up and running quick
- Most popular

WebServer

- A web server is computer software and underlying hardware that accepts requests via **HTTP**
- **HTTP** is a protocol used for sharing between web browsers and web servers

Request methods

- Used indicate the desired action to be performed on the identified resource.

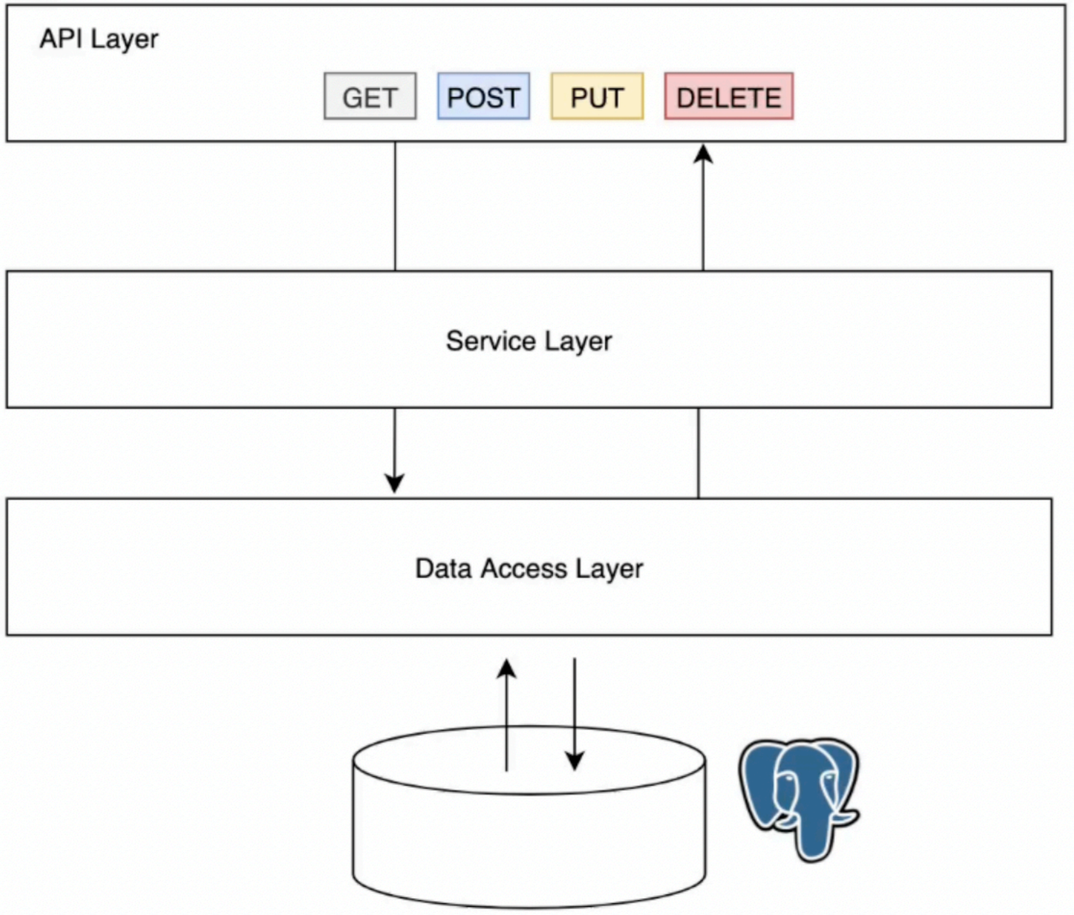
- **GET** - get existing resource
- **POST** - add new resource
- **PUT** - update existing new resource
- **DELETE** - update existing new resource

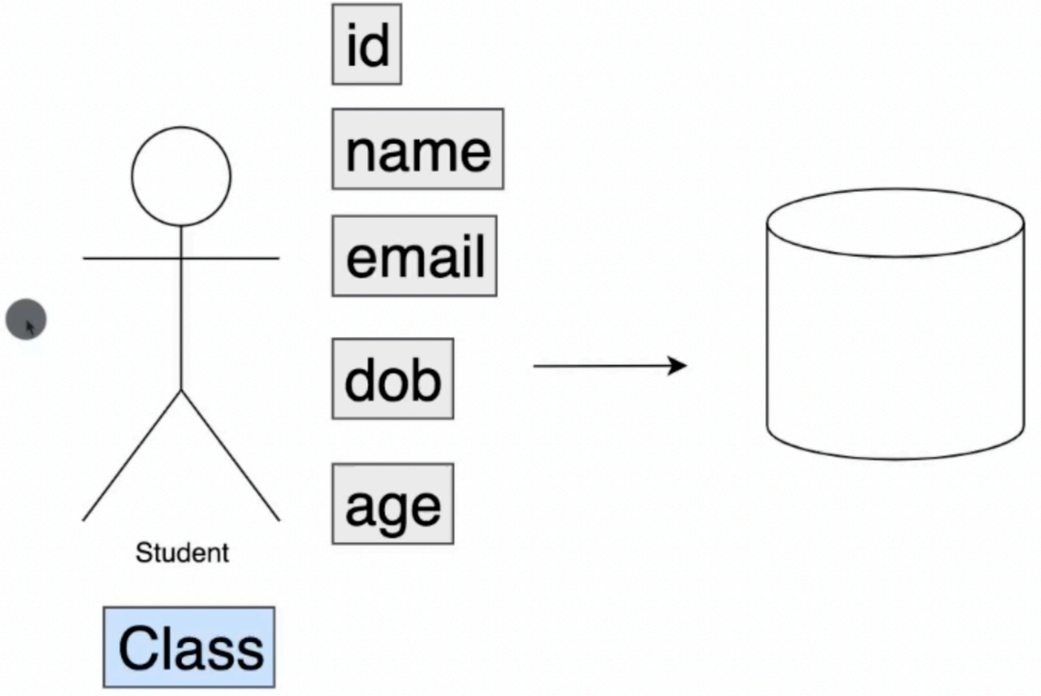
API

- An application programming interface is a connection between computers or between computer programs. It is a type of software interface, offering a service to other pieces of software

Backend

Full Stack

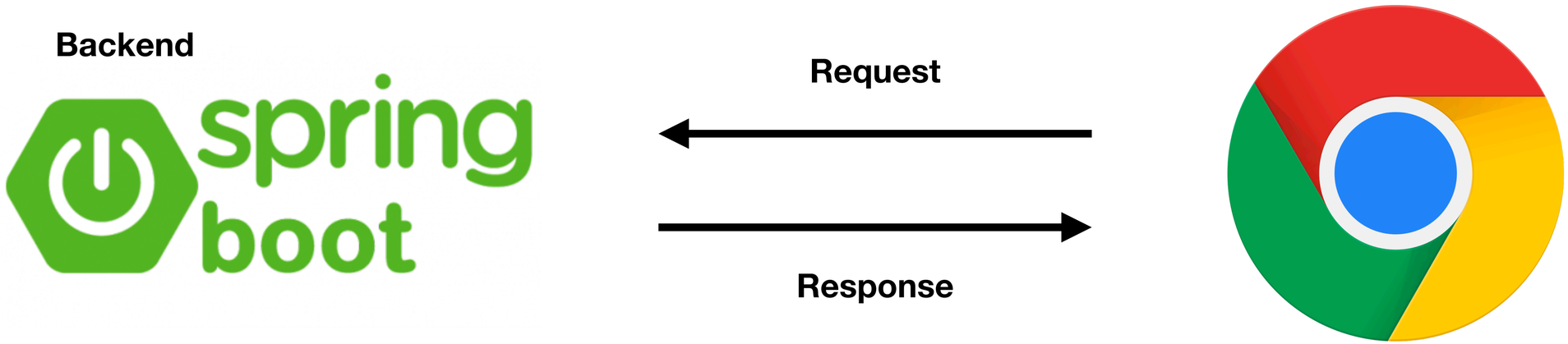




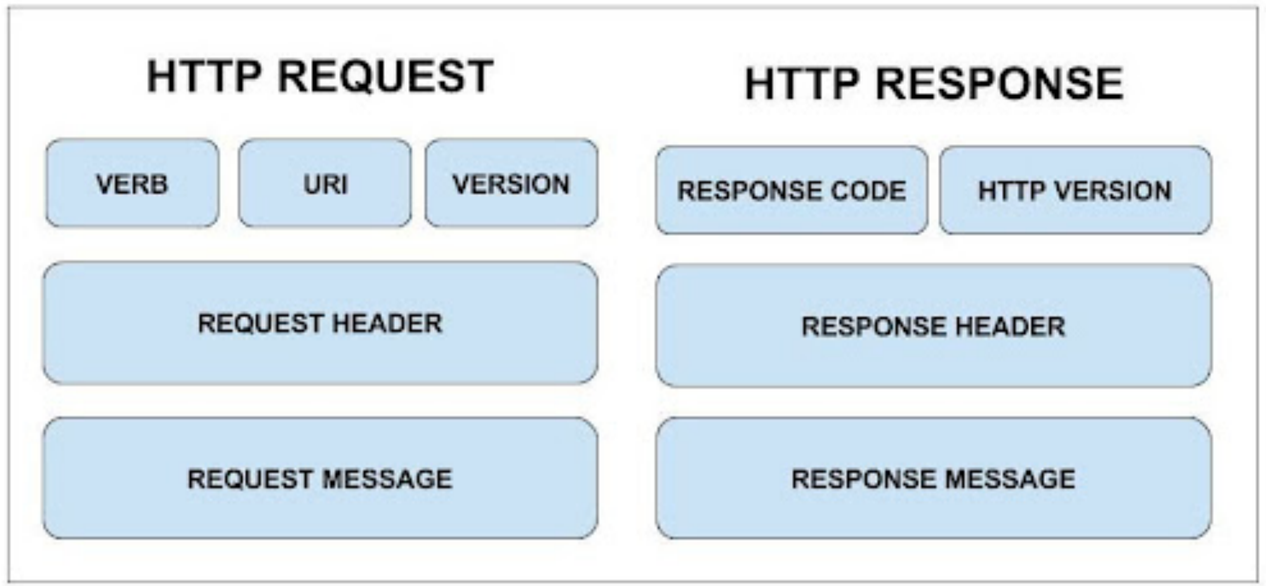
Client and Server

- Client a web app, mobile app or a backend app
- Server is the brains and stores the data
- Communication happens through **HTTP (Hypertext Transfer Protocol)**

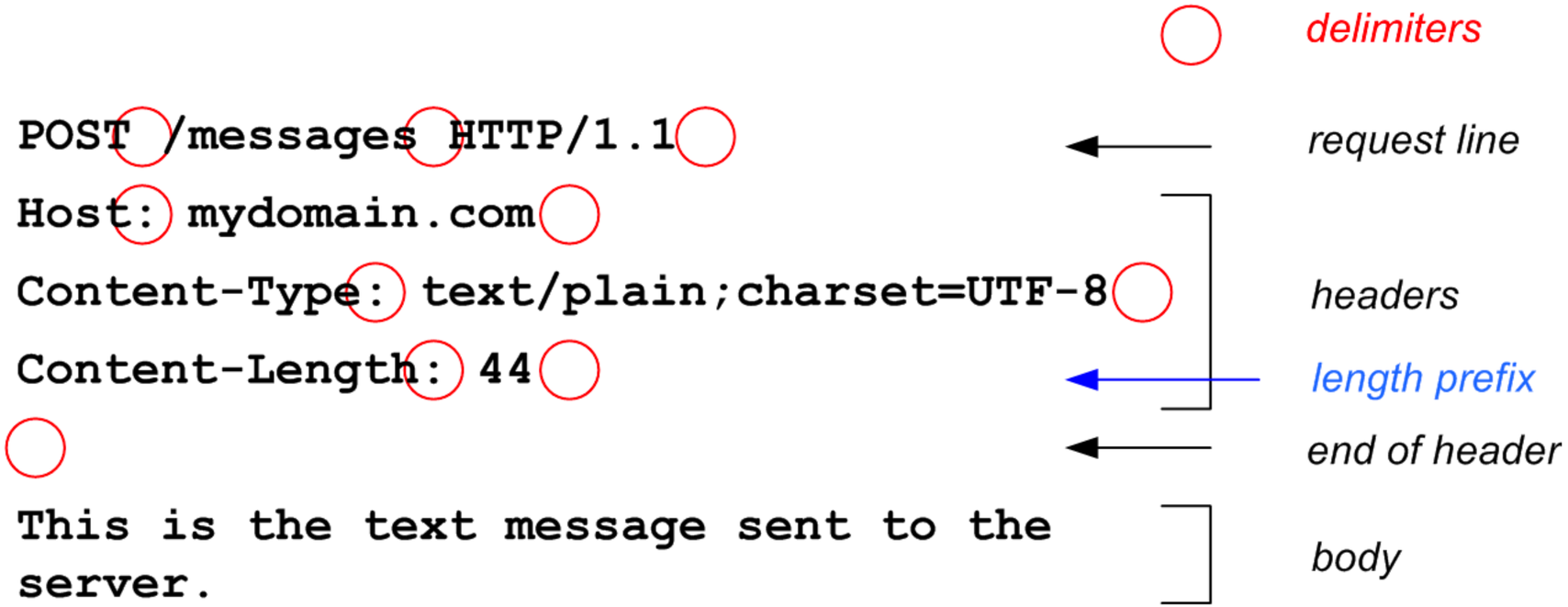
Client and Server



HTTP Message



HTTP Message



HTTP response status codes

HTTP response status codes indicate whether a specific HTTP request has been successfully completed. Responses are grouped in five classes:

1. Informational responses (`100–199`)
2. Successful responses (`200–299`)
3. Redirects (`300–399`)
4. Client errors (`400–499`)
5. Server errors (`500–599`)

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