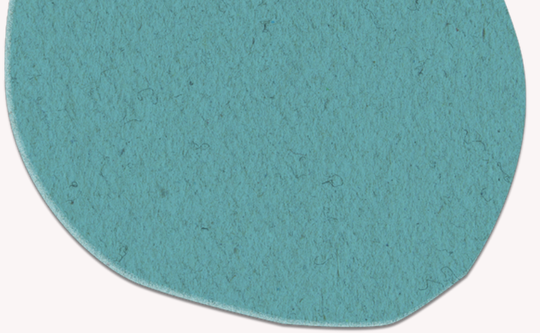


LETS LEARN JAVA

Understanding this timeless language driving global impact .


Java

!=


JS

Java is to JavaScript

is what

Car is to Carpet

WHY JAVA ?

here are some insights for you :

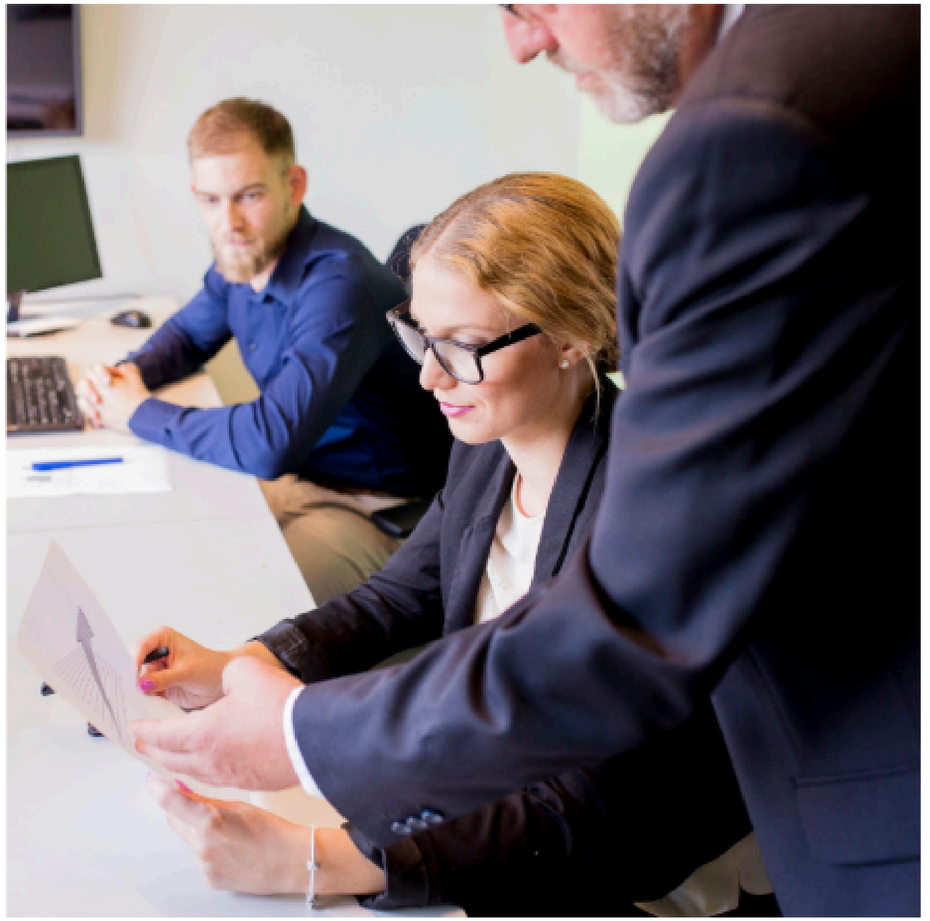
- one of the top prog. lang. globally.
- versatility (OOP , functional programming)
- a highly performant language.
- 90% of Fortune 500 companies rely on Java .
- companies like Amazon , Netflix , eBay, and Alibaba use Java for their backend systems.

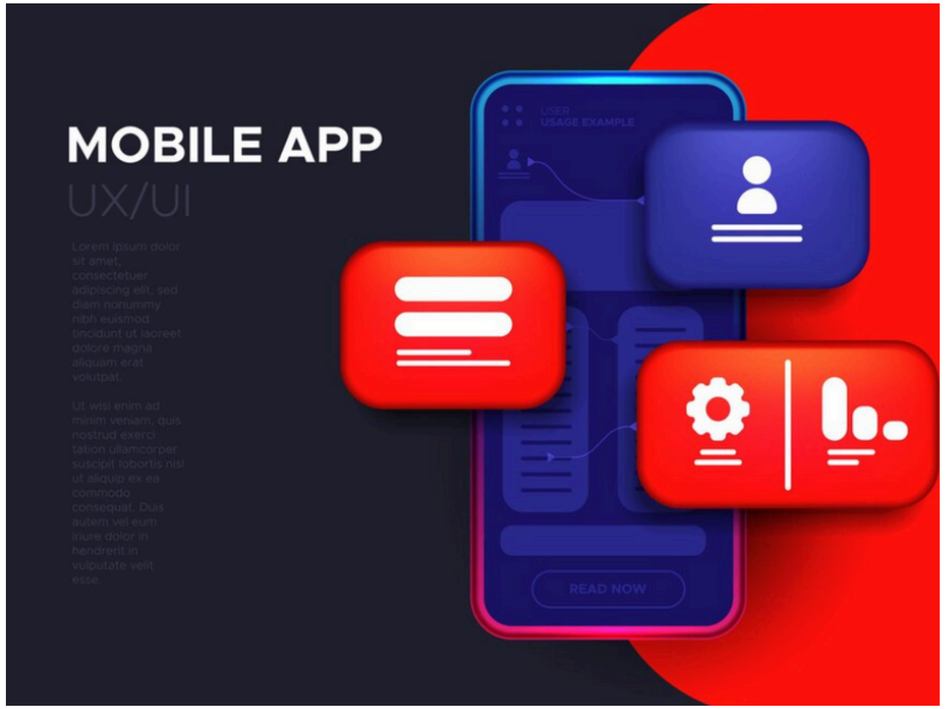
Introduction to Java's Impact

Java is a **versatile** and **powerful** programming language that has transformed the software industry. It enables developers to create **robust** applications across various platforms. This presentation explores the **significant** impact of Java in different sectors and its **global applications**.

Java in Enterprise Solutions

In the **enterprise** sector, Java is widely used for developing **scalable** and **secure** applications. Its features, such as **multithreading** and **cross-platform** capabilities, make it a preferred choice for large organizations aiming to enhance their **operational efficiency**.





Java for Mobile Apps

Java plays a crucial role in the development of mobile applications, particularly on the Android platform. Its robustness and rich API support allow developers to build interactive and user-friendly mobile experiences, driving the growth of the mobile app industry.

Java in Web Development

Java's **frameworks** like Spring and Hibernate have revolutionized **web development**. These frameworks enable developers to create **dynamic**, **secure**, and **scalable** web applications, catering to the increasing demand for online services and e-commerce solutions.

What is Spring Boot?

Spring Boot is an extension of the Spring framework that simplifies the setup and development of new applications. It offers a range of **features** like auto-configuration, embedded servers, and production-ready metrics, making it easier for developers to create stand-alone, production-grade applications with minimal fuss.

Object Oriented Programming

HISTORY OF JAVA

Java was created in 1995 by James Gosling at Sun Microsystems to simplify programming and make it more adaptable to emerging technologies.

It was designed with the vision of “Write Once, Run Anywhere” (WORA) to enable platform-independent development. Built on principles of simplicity, object-orientation, security, and platform independence

HOW TO GET STARTED ?

JAVA INSTALLATION

1. Visit the link given to you , this will install JDK in your PC.

2. After installation , update environment variables.

3. Check through command prompt , if it successfully installed or not

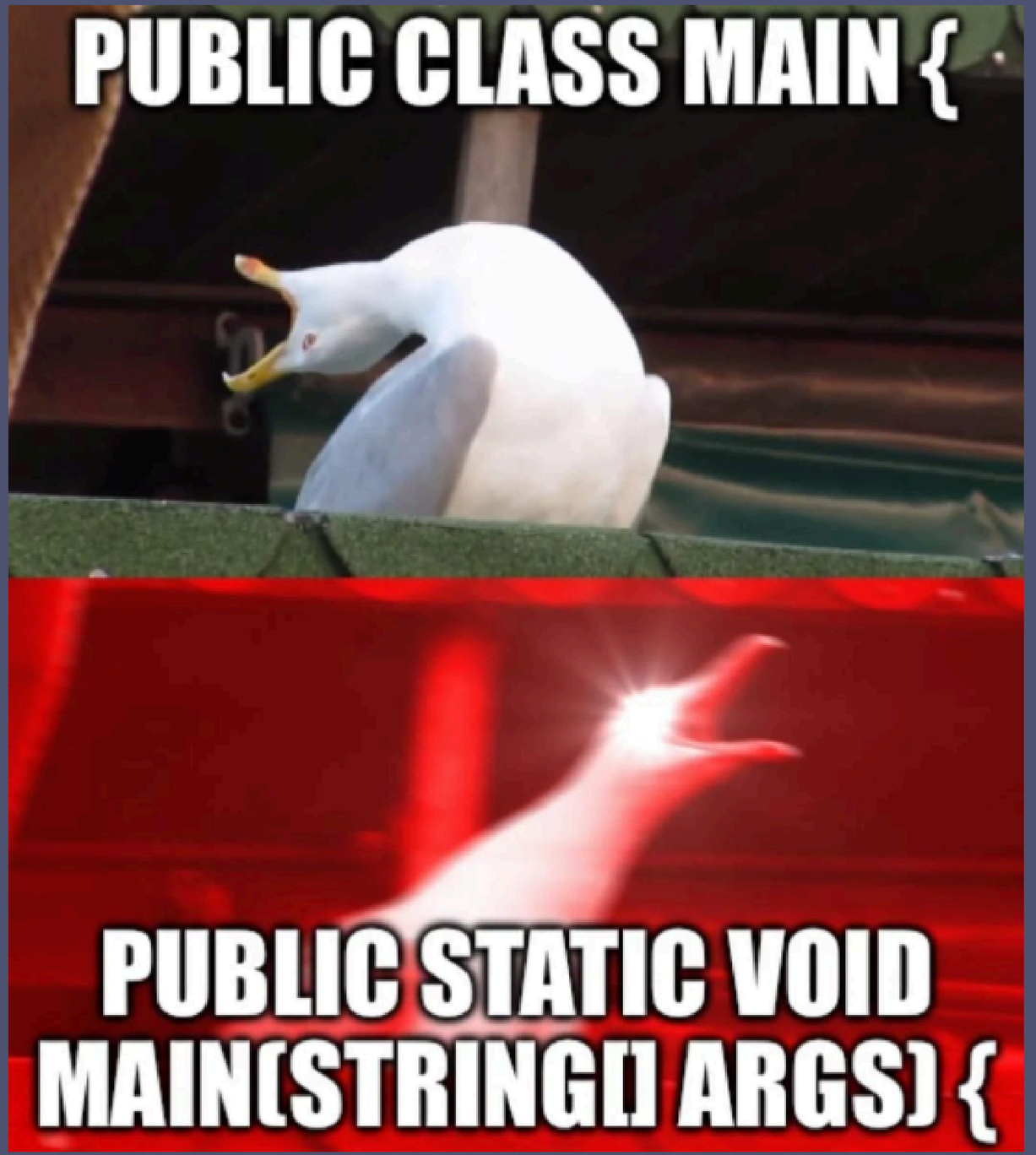
INSTALL THIS IDE


IJ
*Me when IntelliJ

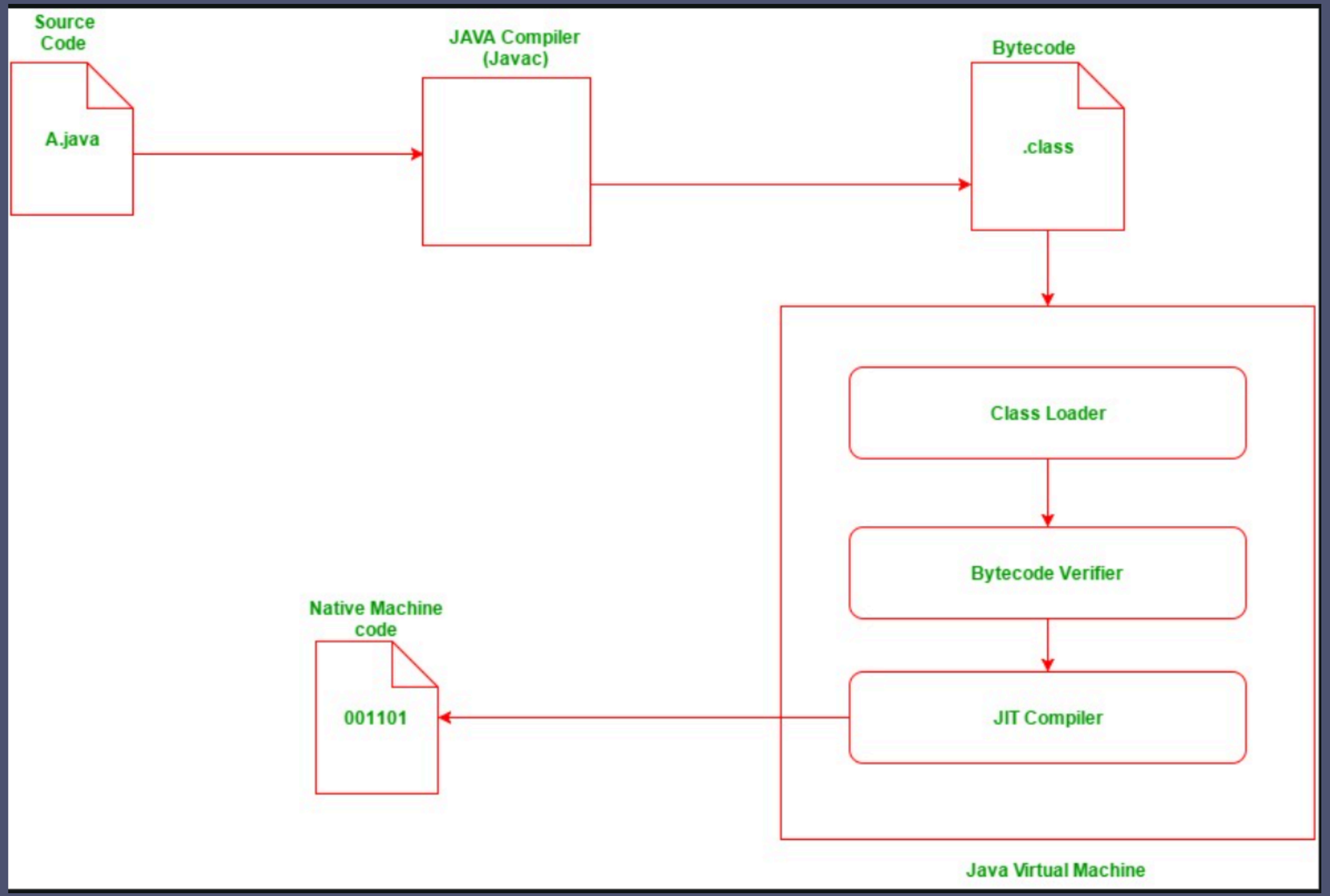
OUR FIRST JAVA PROGRAM

ABC.java

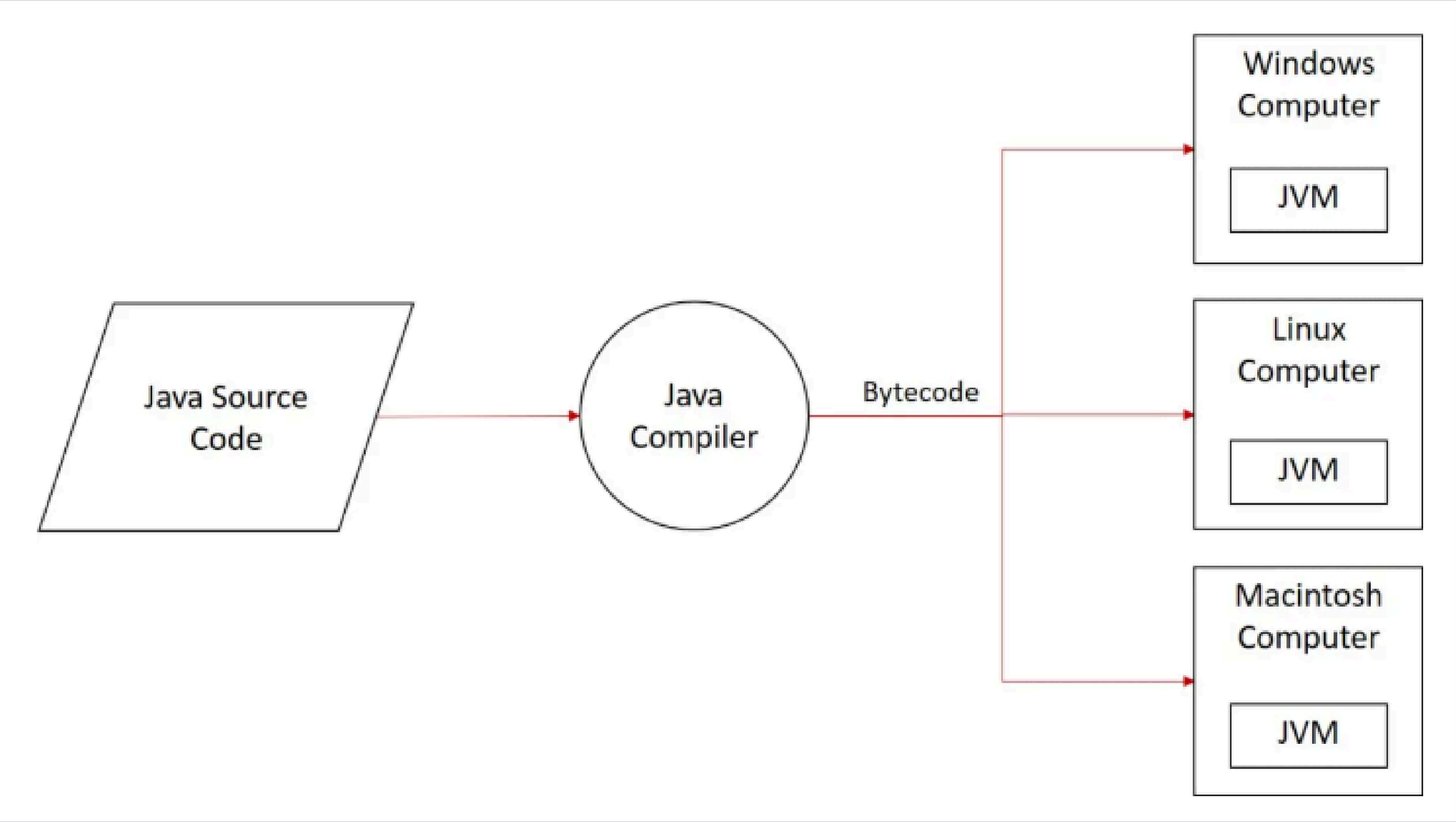
```
1  class ABC
2  {
3  public static void main(String[] args)
4  {
5  System.out.println("Hello Java");
6      }
7  }
```


PUBLIC CLASS MAIN {
PUBLIC STATIC VOID
MAIN(STRING[] ARGS) {

JAVA COMPILATION PROCESS



HOW PLATFROM INDEPENDENT?




JVM
JRE
Developers often install JDK

“that grimace shake be bussin bruh”

bytecode

machine code

OOPs in JAVA*

OOPs in other languages*

A LOW DOSE OF OOPS

Encapsulation:

Technique of bundling data (fields) and methods that operate on the data into a single unit (class) and restricting direct access to some of the object's components.

Inheritance :

Inheritance in Java is a mechanism where one class acquires the properties and behaviors of another class, promoting code reuse and establishing a hierarchical relationship.

Polymorphism :

allows objects of different classes to be treated as objects of a common superclass, enabling methods to behave differently based on the object's actual class.

Abstraction :

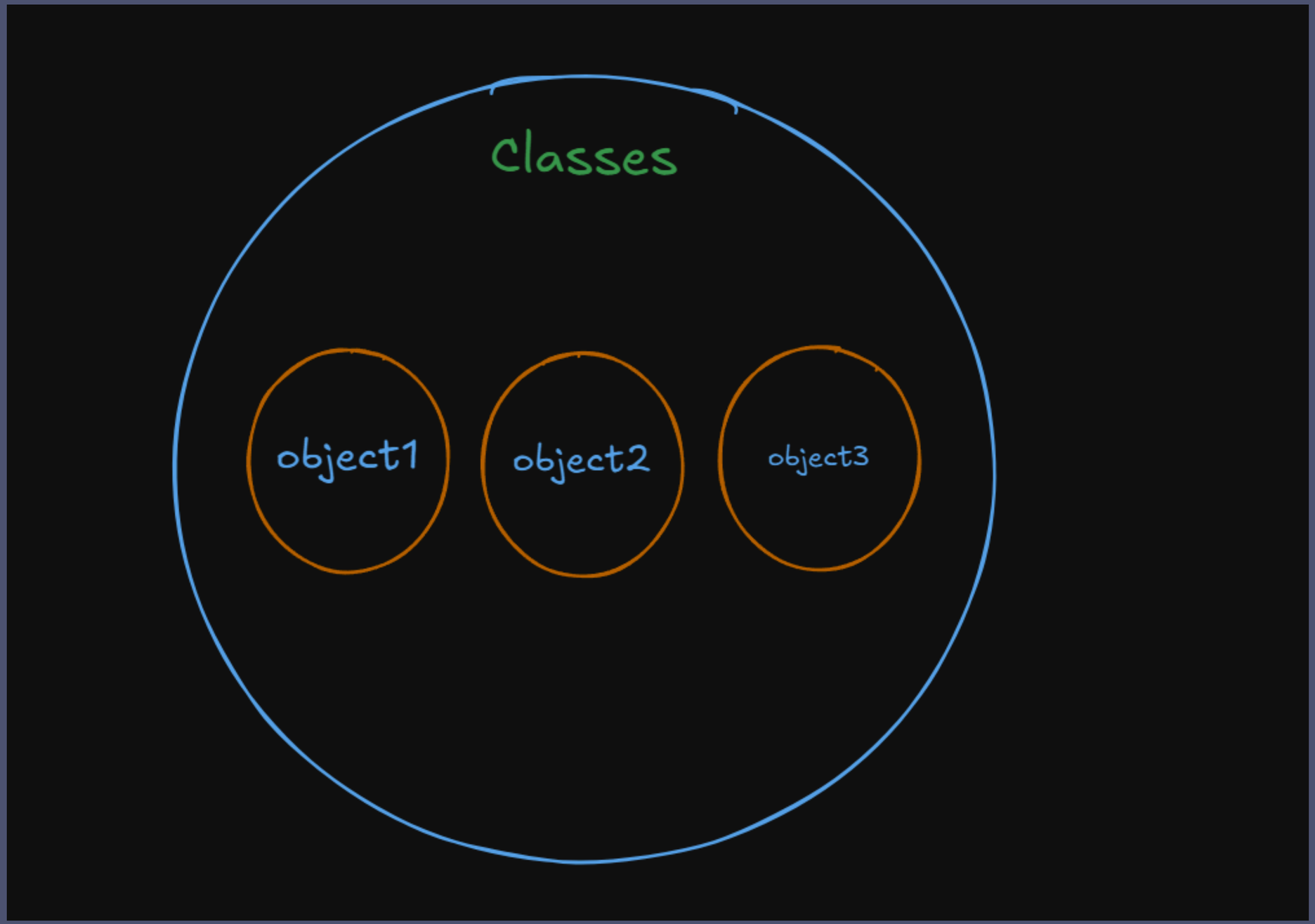
Concept of hiding complex implementation details and exposing only the essential features of an object, typically through abstract classes or interfaces.

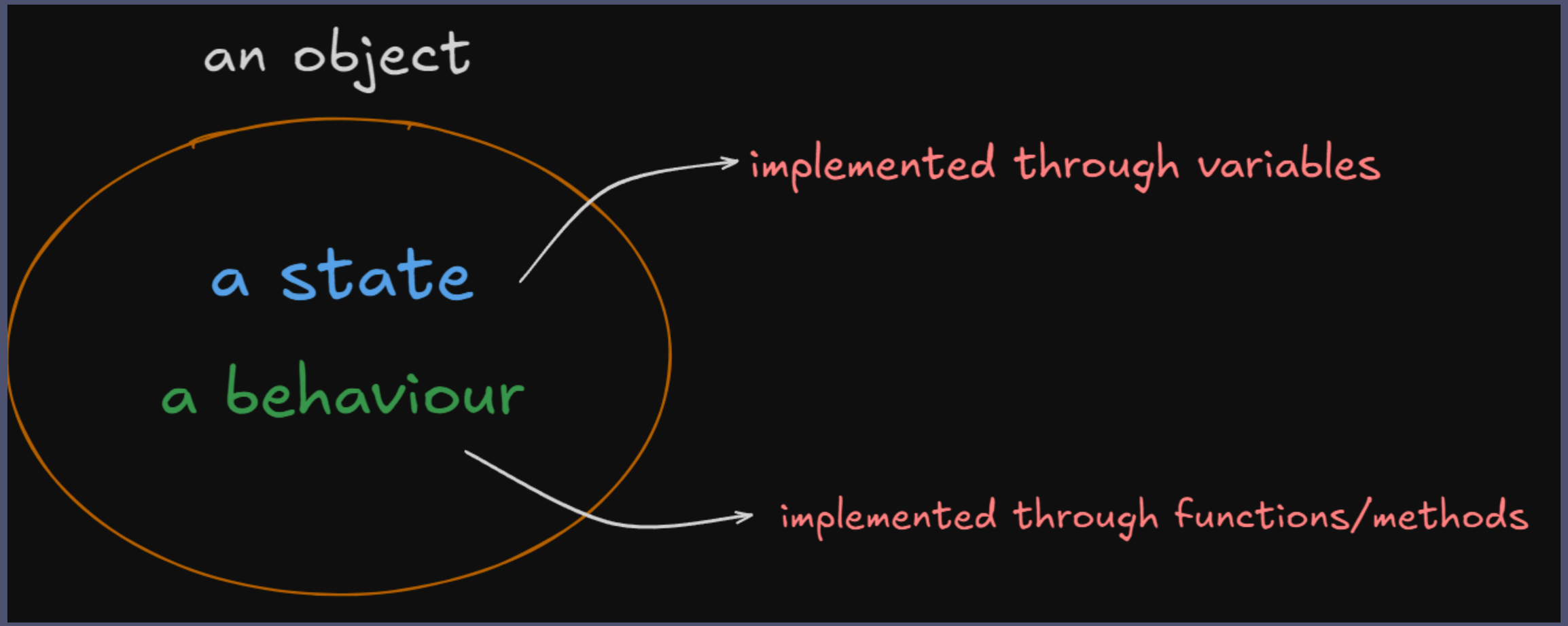
CLASSES AND OBJECTS

In Java, a class is a blueprint or template that defines the properties (fields) and behaviors (methods) that objects created from the class will have. It serves as a structure for creating objects.

An object is an instance of a class, created using the new keyword. It represents a real-world entity with specific values for the class's fields and the ability to invoke its methods to perform actions or modify its state. Each object has its own unique identity, state, and behavior.

After studying OOPs for 8 minutes


Classes
object1
object2
object3


an object
a state
a behaviour
implemented through variables
implemented through functions/methods

THE DOT AND NEW OPERATOR

The dot operator (.) in Java is used to access members of a class or an object, such as fields (variables), methods, or nested classes. It allows you to specify the relationship between an object or class and its components.

The new keyword in Java is used to create new objects or allocate memory for objects dynamically. It initializes the object by calling its constructor and returns a reference to the object.

```java
class Car {
    String model = "Toyota";
}

public class Main {
    public static void main(String[] args) {
        Car car = new Car();
        System.out.println(car.model);
    }
}
```

here is the 'new' keyword allocating memory to this created object

```java
class Person {
    String name = "John";

    void greet() {
        System.out.println("Hello, " + name);
    }
}

public class Main {
    public static void main(String[] args) {
        Person person = new Person();
        System.out.println(person.name);
        person.greet();
    }
}
```

Static and Non-Static

In Java, static and non-static methods differ in how they are accessed and associated with a class or its objects.

Static Methods :

> Definition: Methods declared with the static keyword.
> Association: Belong to the class rather than any specific object.
> Access: Can be called directly using the class name, without creating an object.
> Limitations: Cannot access non-static fields or methods directly because they do not depend on an instance.

Non-Static Methods

Definition: Methods without the static keyword.
Association: Belong to an object and require an instance of the class to be called.
Access: Can access both static and non-static fields or methods of the class.

```java
class Example {
    void nonStaticMethod() {
        System.out.println("This is a non-static method.");
    }
}

public class Main {
    public static void main(String[] args) {
        Example obj = new Example(); // Create an object
        obj.nonStaticMethod(); // Called using the object
    }
}
```

```java
class Example {
    static void staticMethod() {
        System.out.println("This is a static method.");
    }
}

public class Main {
    public static void main(String[] args) {
        Example.staticMethod(); // Called using the class name
    }
}
```

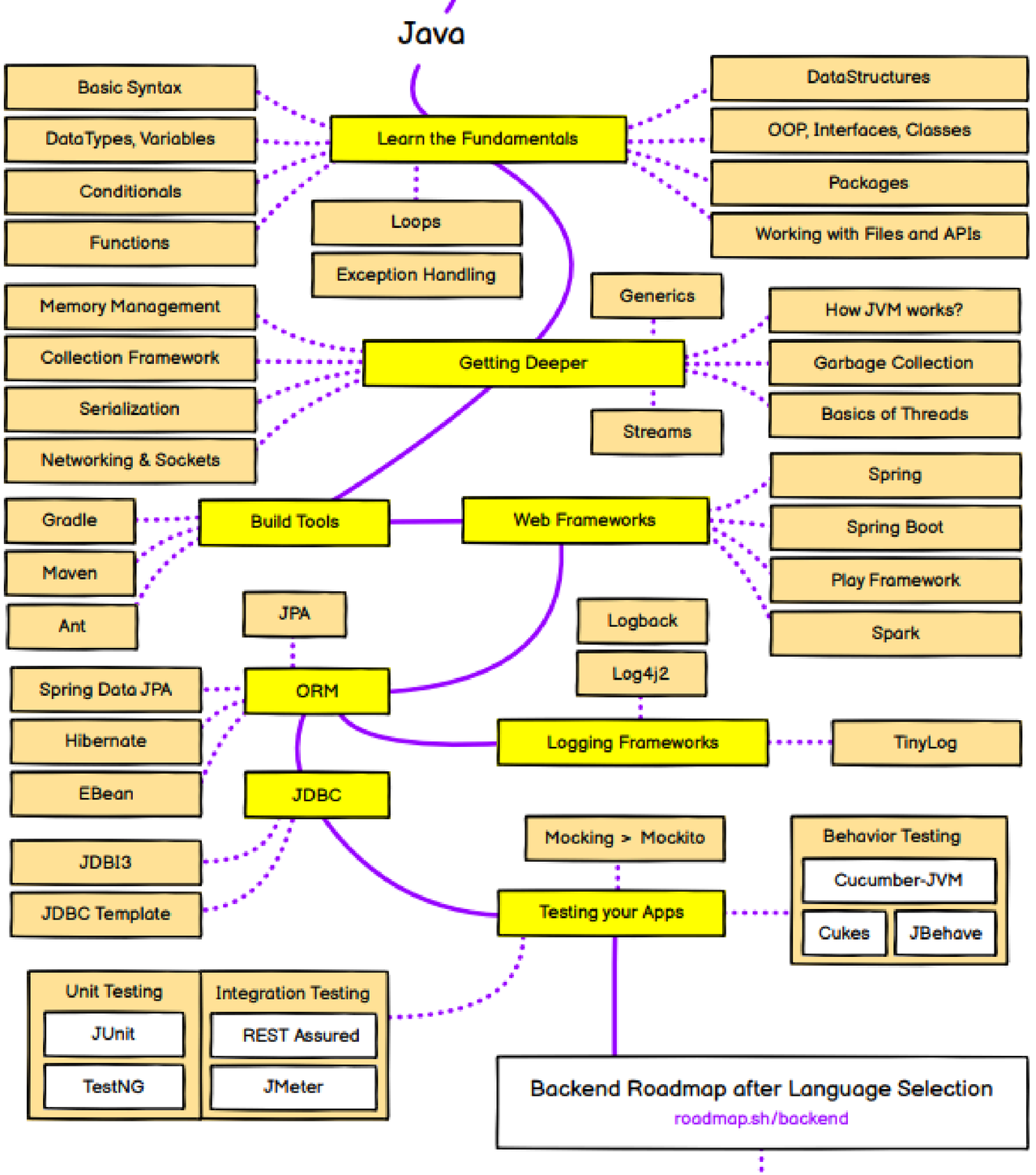
A small catch !

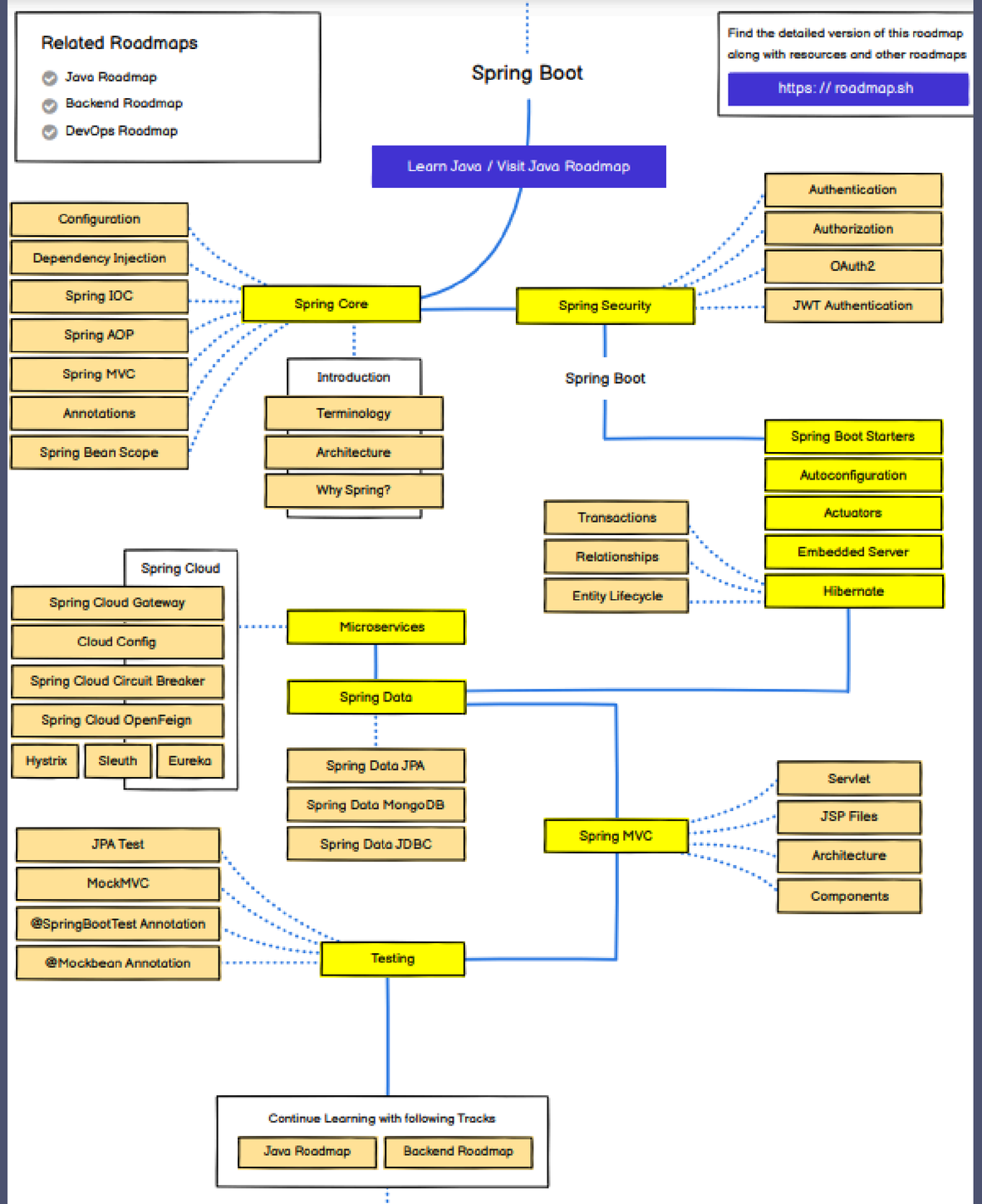
as we all know , JAVA is an Object Oriented language
i.e. everything in JAVA will be accessed by creation of objects/instances of classes .
but on taking a look at the 'main' method which is essential for starting the JAVA program , the class in which it is contained does not have any instance in the first place.
Then , how does it even start at the first place and how is the 'main' method called ?

well , if we examine it closely ...we will see the format :

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Number of arguments: " + args.length);
    }
}
```

if we notice...its a 'static' method
hence it does not require an object to be accessed so the JVM uses the class name directly to invoke this method and make our JAVA program run :)


Java
Basic Syntax
DataTypes, Variables
Conditionals
Functions
Learn the Fundamentals
Loops
Exception Handling
DataStructures
OOP, Interfaces, Classes
Packages
Working with Files and APIs
Memory Management
Collection Framework
Serialization
Networking & Sockets
Getting Deeper
Generics
Streams
How JVM works?
Garbage Collection
Basics of Threads
Gradle
Maven
Ant
Build Tools
Web Frameworks
Spring
Spring Boot
Play Framework
Spark
JPA
Spring Data JPA
Hibernate
EBean
ORM
Logback
Log4j2
Logging Frameworks
TinyLog
JDBC
JDBI3
JDBC Template
Mocking > Mockito
Testing your Apps
Behavior Testing
Cucumber-JVM
Cukes
JBehave
Unit Testing
JUnit
TestNG
Integration Testing
REST Assured
JMeter
Backend Roadmap after Language Selection
roadmap.sh/backend


Related Roadmaps
Java Roadmap
Backend Roadmap
DevOps Roadmap
Spring Boot
Find the detailed version of this roadmap along with resources and other roadmaps
https: // roadmap.sh
Learn Java / Visit Java Roadmap
Configuration
Dependency Injection
Spring IOC
Spring AOP
Spring MVC
Annotations
Spring Bean Scope
Spring Core
Introduction
Terminology
Architecture
Why Spring?
Spring Security
Authentication
Authorization
OAuth2
JWT Authentication
Spring Boot
Spring Boot Starters
Autoconfiguration
Actuators
Embedded Server
Hibernate
Transactions
Relationships
Entity Lifecycle
Spring Cloud
Spring Cloud Gateway
Cloud Config
Spring Cloud Circuit Breaker
Spring Cloud OpenFeign
Hystrix
Sleuth
Eureka
Microservices
Spring Data
Spring Data JPA
Spring Data MongoDB
Spring Data JDBC
Spring MVC
Servlet
JSP Files
Architecture
Components
JPA Test
MockMVC
@SpringBootTest Annotation
@Mockbean Annotation
Testing
Continue Learning with following Tracks
Java Roadmap
Backend Roadmap


THANK YOU FOR YOUR ATTENTION
ANY QUESTIONS?