Java Day1

📘 Day 1: Java Basics

🔷 Introduction to Java

Java is one of the most popular and widely used programming languages in the world. It was developed by James Gosling at Sun Microsystems in 1995 and later acquired by Oracle Corporation.

Key Features of Java

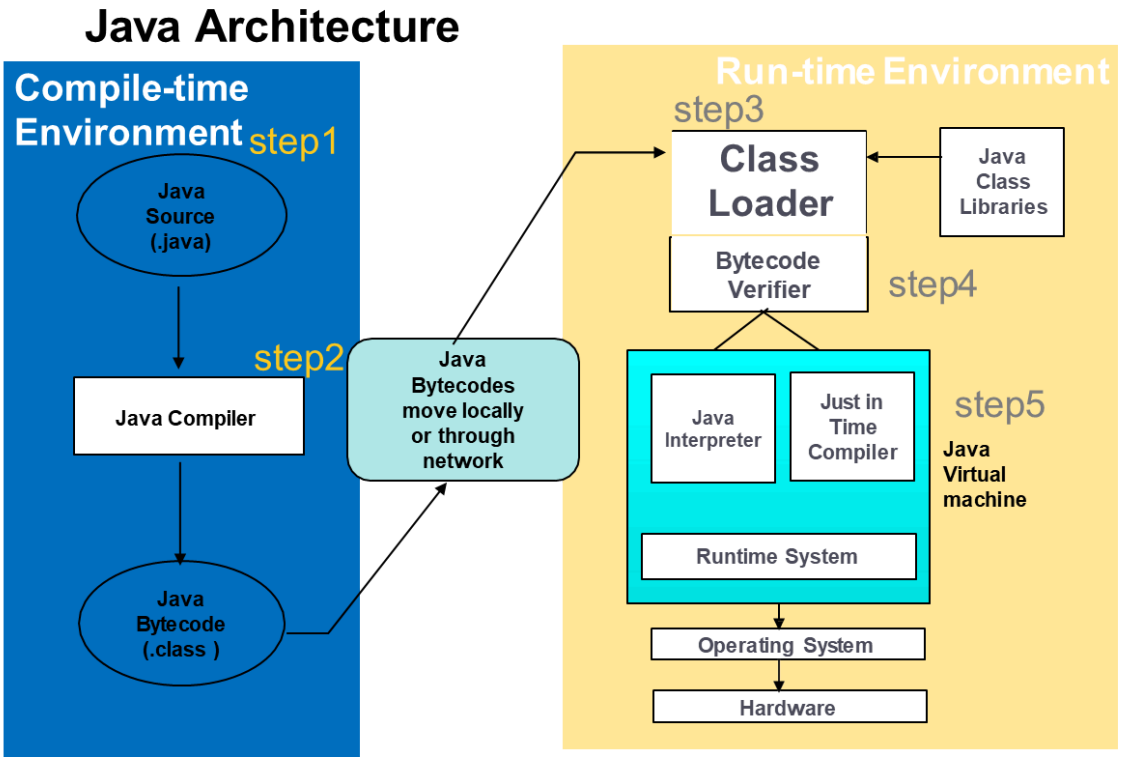
- **Simple** – Easy to learn, syntax similar to C/C++
- **Object-Oriented** – Everything is treated as an object
- **Platform Independent** – Runs on any system with JVM

Key Founders

- Java was the brainchild of:
  - James Gosling
  - Patrick Naughton
  - Chris Warth
  - Ed Frank &
  - Frank Sheridan
- The origin of Java can be traced back to the fall of 1992, and was initially called **Oak**
- Oak was renamed as **Java** in **1995**

Design Goal

- Java was originally meant to be a platform-neutral language for embedded software in devices
- The goal was to move away from platform and OS-specific compilers that would compile source for a particular target platform to a language that would be portable, and platform-independent
- The language could be used to produce platform-neutral code



Java Architecture (Contd.).

Step1:

Create a java source code with .java extension

Step2:

Compile the source code using java compiler, which will create bytecode file with .class extension

Step3:

Class loader reads both the user defined and library classes into the memory for execution

Java Architecture (Contd.).

Step4:

Bytecode verifier validates all the bytecodes are valid and do not violate Java's security restrictions

Step5:

JVM reads bytecodes and translates into machine code for execution. While execution of the program the code will interact to the operating system and hardware

The 5 phases of Java Programs

Java programs can typically be developed in five stages:

1. Edit

   Use an editor to type Java program (**Welcome.java**)
2. Compile
   - Use a compiler to translate Java program into an intermediate language called bytecodes, understood by Java interpreter (**javac Welcome.java)**
   - Use a compiler to create **.class** file, containing bytecodes (**Welcome.class**)
3. Loading

   Use a class loader to read bytecodes from **.class** file into memory

The 5 phases of Java Programs (Contd.).

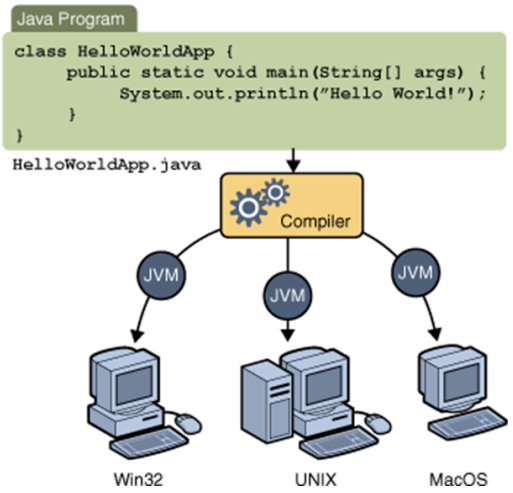
4. Verify

   Use a Bytecode verifier to make sure bytecodes are valid and do not violate security restrictions

5. Execute

   - Java Virtual Machine (JVM) uses a combination of interpretation and just-in-time compilation to translate bytecodes into machine language
   - Applications are run on user's machine, i.e. executed by interpreter with java command (java Welcome)

Java Virtual Machine



- The output of the compiler is bytecode
- The bytecodes are executed by JVM
- It is an interpreter which converts the byte code to machine specific instructions
  and executes
- JVM is platform specific

Components of Java Architecture

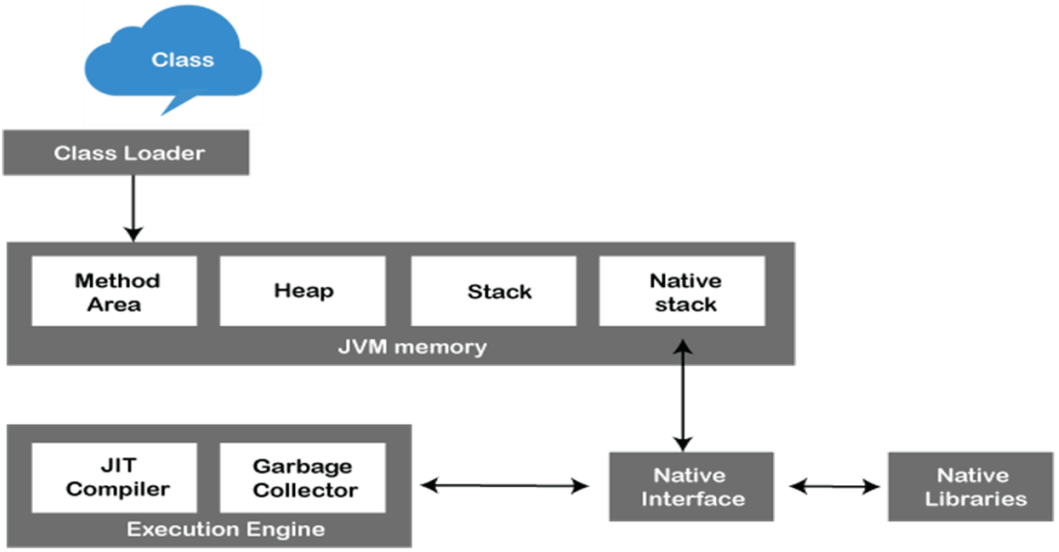
The Java architecture includes the three main components:

- Java Virtual Machine (JVM)
- Java Runtime Environment (JRE)
- Java Development Kit (JDK)

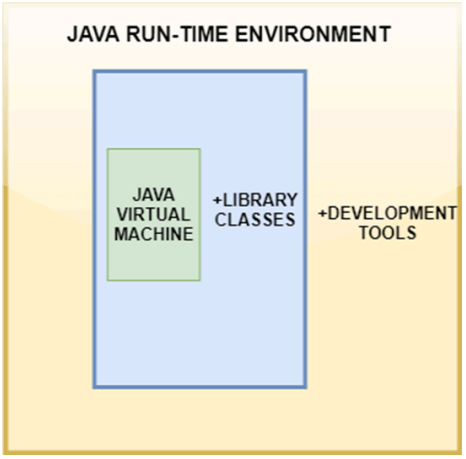
Java Virtual Machine(JVM)

- The main feature of Java is **WORA**. **WORA** stands for **Write Once Run Anywhere**.
- JVM's main task is to convert byte code into machine code.
- JVM, first of all, loads the code into memory and verifies it. After that, it executes the code and provides a runtime environment.

JVM Architecture



JRE Architecture(Java Runtime Environment)



A Simple Java Program

Our first Java Program:

```
public class Welcome {
    public static void main(String args[]) {
        System.out.println("Welcome..!");
    }
}
```

This program displays the output "Welcome..!" on the console

Create source file : Welcome.java

Compile : javac Welcome.java

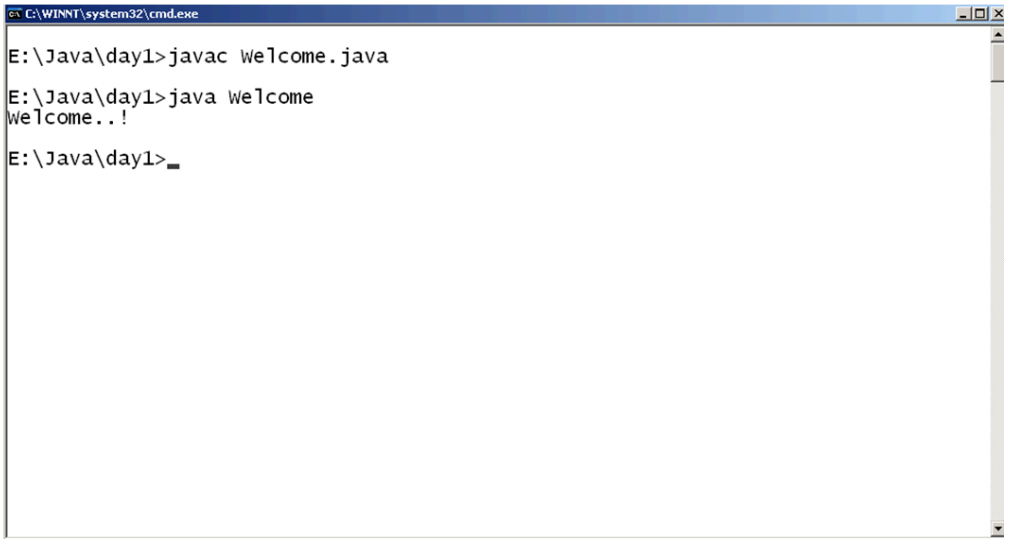
Execute : java Welcome

Executing your first Java Program

- Before executing the program, just check whether the PATH and the CLASSPATH parameters are properly set, by typing in the commands as shown in the screen below:

Executing your first Java Program (Contd.).

- Now compile and execute your program as given below :



🧠 Introduction to Git & GitHub

🔧 What is Git?

Git is a **version control system** that helps you track changes in your code. It lets you:

- Save different versions of your project
- Undo mistakes by reverting to earlier versions
- Collaborate with others without overwriting each other's work

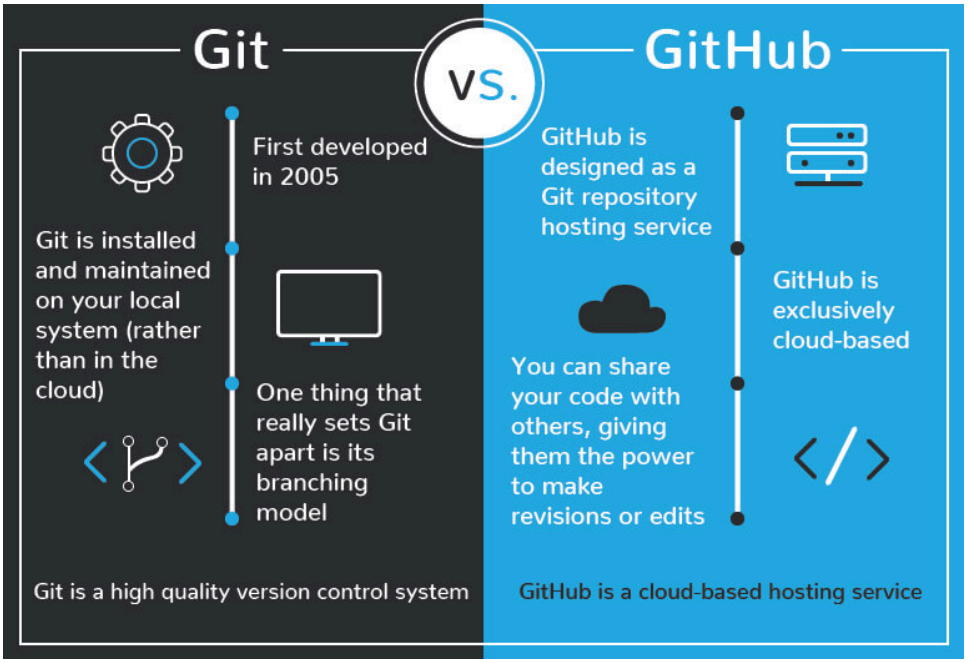
Think of Git as a **time machine for your code** — every commit is a snapshot you can revisit.

🌐 What is GitHub?

GitHub is a **web-based platform** that hosts Git repositories online. It allows you to:

- Store your code in the cloud
- Share projects with others
- Collaborate through pull requests, issues, and branches

Git is the tool, and GitHub is the place where you use that tool with others.



👨‍💻 Java Project Setup with Version Control

🔷 1. Create a GitHub Repository

- Go to GitHub
- Create a new repository named `OOP`

🔷 2. Prepare Local Workspace

- Create a folder named `Mohamed Sathak` on your desktop
- Open the folder in **IntelliJ IDEA**

🔷 3. Initialize Git Locally

Open IntelliJ terminal and run:

bash

```
git init
```

📝 Starts Git tracking in your local folder

🔷 4. Configure Git (first-time setup)

bash

```
git config --global user.name "Your Name"
git config --global user.email "your@email.com"
```

📝 Sets your identity for commits

🔷 5. Clone GitHub Repo into Local Folder

bash

```
git clone https://github.com/your-username/OOP.git
```

📝 Downloads the `OOP` repo into your local folder

🔷 6. Create Java Project

- In IntelliJ, create a new Java project named `JavaLearning`
- Inside `src`, create a package: `com.Day1`
- Add a class: `HelloWorld.java`

🔷 7. Navigate to Package Folder

bash

```
cd src/com/Day1
```

📝 Moves into the folder containing your Java file

🔷 8. Stage and Commit Your Code

bash

```
git add HelloWorld.java
git commit -m "feat: added a new java class file named HelloWorld.java"
```

📝 Adds and saves your changes with a message

🔷 9. Push Code to GitHub

bash

```
git push origin main
```

📝 Sends your local commits to GitHub

🔷 10. Edit File on GitHub

- Go to GitHub repo
- Open `HelloWorld.java`
- Add a new line and save the change

🔷 11. Pull Changes to Local

bash

```
git pull
```

🗒️ Updates your local file with changes from GitHub

🔷 12. Check File Status

bash

`git status`

🗒️ Shows which files are modified, staged, or untracked