Advanced Programming with Java

Prativa Nyaupane

What do we gain from this course?

- Gain in-depth understanding of advanced Java programming
- Gain insights into the object-oriented principles
- Develop expertise in creating graphical user interfaces
- Gain concepts of networking and distributed programming concepts
- Gain comprehensive knowledge of database connectivity with Java
- Gain insights into web development with servlets and JSP
- Gain exposure to more advanced Java topics

Applications built using Java

Android Apps	Financial Applications	Big Data Technologies	IDEs	Games
Instagram	PayPal	Apache Hadoop	Eclipse	Minecraft(Java Edition)
WhatsApp	Visa Online	Apache Spark	IntelliJ	Puzzle Pirates

Why was Java developed?

Java was developed by Sun microsystems(subsidiary of oracle) in the year 1995. James Gosling is known as the father of java. It was initially as oak.

Before Java, developers faced several challenges in software development, esp in the context of networked environments and platform independence.

Some of the key challenges include:

1. Platform Dependence	2. Memory management	3. Security Concerns
4. Distribution and Deployment	5. Networked Computing Challenges	6. Versioning and Compatibility

How java resolves this problem?

The challenges before Java were primarily related to platform dependence, memory management, security, distribution and deployment, networked computing and versioning and compatibility.

Java was developed to address these challenges by providing a platform-independent programming language and runtime environment with built-in support for networking, security, memory management and other key features.

It is simple, object-oriented nature, robustness, security and large community supports.

Platform Independence:

Java achieves platform independence through its compilation process and the Java Runtime Environment(JRE).

1. Compilation to bytecode - Compilation is the process of translating sources codes into machine-readable instructions or bytecode. Java source code is compiled into platform-independent bytecode, rather than machine code.

2. Java Virtual Machine - The bytecode is executed by the JVM, which is part of the JRE. JVM translates bytecode into native machine code at runtime. So, if JVM is installed in any system, then java programs can easily run.

Object-Oriented Programming(OOP):

OOP is a programming paradigm, where entities, which can represent real-world objects, such as cars or animals are represented using attributes(properties) and behaviors(actions or functions).

1. Create modular and reusable code so that it can be easily managed and maintained in large-scale applications.
2. Implement OOP concepts, such as, encapsulation, inheritance and polymorphism.

Robust and Secure:

Robustness refers to its ability to handle errors and exceptions gracefully, preventing crashes and ensure stability event in unexpected conditions. Security refers to the protection for various security threats, such as unauthorized access, code injection and malicious attacks.

1. Robustness is achieved through strong type checking, automatic memory management(garbage collection) and exception handling mechanisms.
2. Security is achieved through bytecode verification, access control mechanism and sandbox environment that restricts the actions of untrusted code.

Rich API and Large Community Support:

It is a key property due to which Java is successful and is adopted worldwide. API provides developers with a vast collection of pre-built libraries and frameworks for a wide range of functionalities, including networking, database access, GUI development and more.

Ready-to-use tools accelerates development and simplifies complex tasks.

Large amount of community support provides Java with large and active community of developers, enthusiasts and experts who contribute to its ecosystem through forums, online communities, open-source projects and collaborative development efforts.

Multithreading and Concurrency:

Multithreading is the process of doing multiple tasks at same time by a program within the same process. . Each task is a thread. Multiple things are happening at once the same program.

Concurrency is a broader concept. In this process, Java manages multiple tasks and make sure they progress smoothly together, even if they are not happening at the same time.

In GUI - multithreading allows the interface to remain responsive while performing tasks in the background.

Performance and Efficiency:

Java has a Just-In-Time(JIT) compiler that dynamically compiles Java bytecode into native machine code at runtime. This compilation process optimizes the performance of Java applications, making them faster and more efficient. Additionally, Java's garbage collector manages memory efficiently, automatically deallocating memory that is no longer in use.

Wide Range of Applications:

Java is a versatile language used for developing a wide range of applications, from web and enterprise applications to mobile apps, desktop software, scientific applications, and embedded systems. It is the language behind popular frameworks like Spring and technologies like Android development.

A Simple Java Program

```
public class HelloWorld {

    public static void main(String[] args) {

        System.out.println("Hello, World!");

    }

}
```

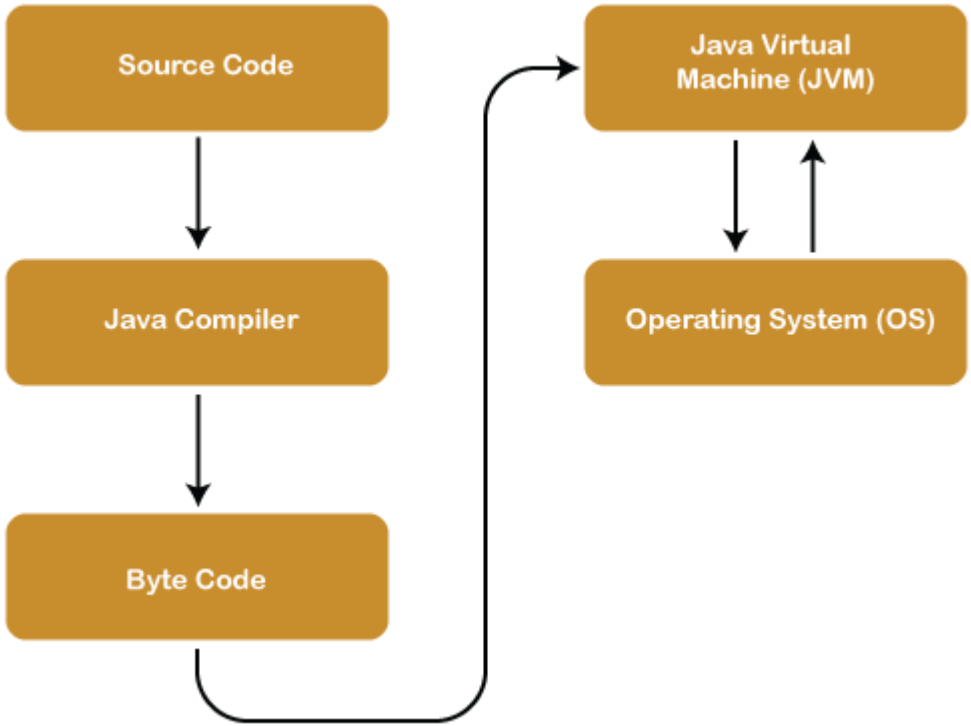
In this code:

- We define a class named **HelloWorld**.
- Inside the class, we have a **main** method, which is the entry point of the program in Java.
- Within the main method, we use **System.out.println()** to print the string **"Hello, World!"** to the console.

Java Architecture

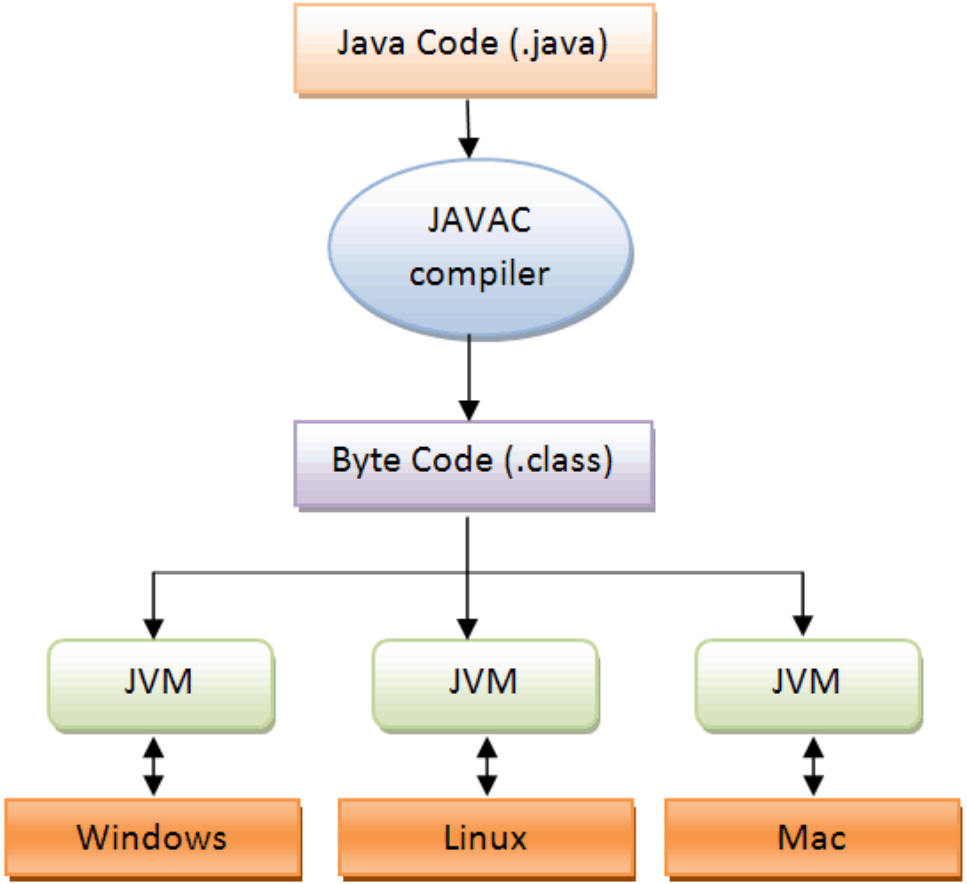
- Architecture is a collection of components, i.e., JVM, JRE, and JDK.
- It integrates the process of interpretation and compilation
- It defines all the processes involved in creating a Java program
- Java Architecture explains each and every step of how a program is compiled and executed.

Java Architecture(contd…)



Java Virtual Machine(JVM)

JVM is an abstract machine that provides the environment in which Java bytecode is executed.



Java Runtime Environment(JRE)

The JRE creates the JVM and ensures dependencies are available to your Java programs.

- It provides an environment in which Java programs are executed
- JRE takes our Java code, integrates it with the required libraries, and then starts the JVM to execute it

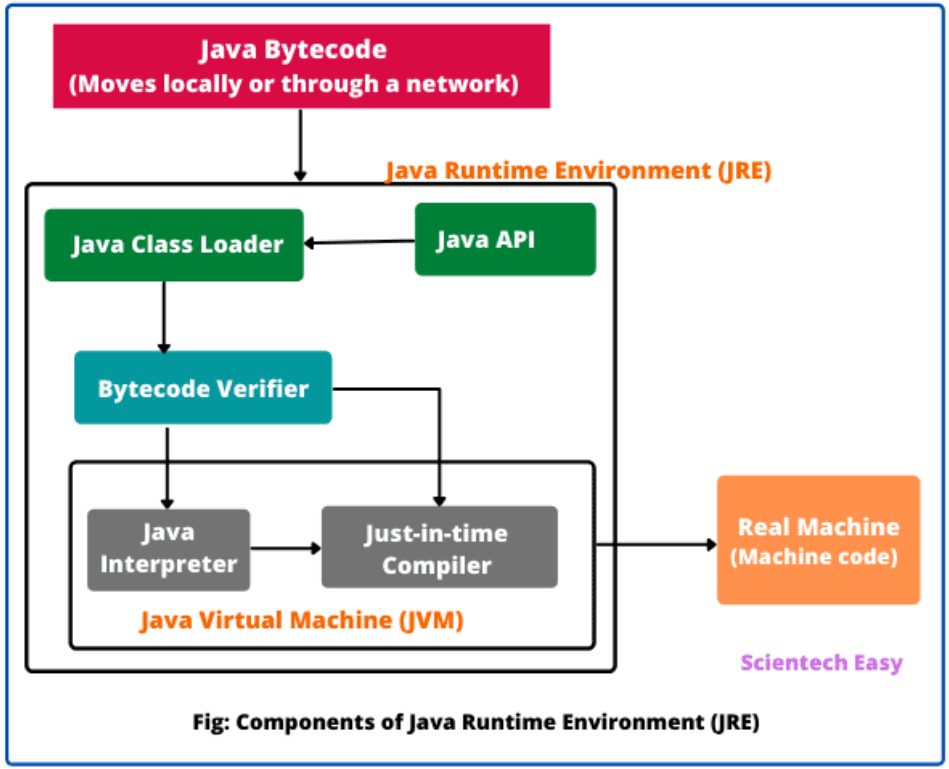


Fig: Components of Java Runtime Environment (JRE)

Java Development Kit

- It is a software development environment used in the development of Java applications
- Java Development Kit holds JRE, a compiler, an interpreter or loader, and several development tools in it

JRE = JVM + library classes.

JDK = JRE + Developer tools.