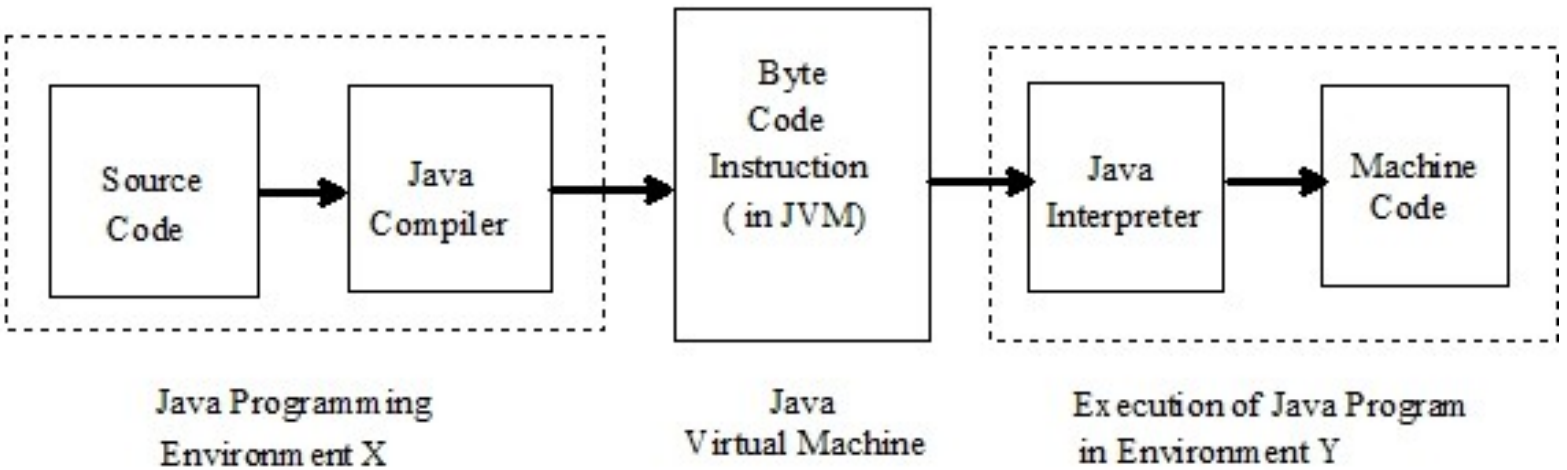
Introduction to Java

Introduction to Java

- Java is a widely used, versatile programming language known for its portability, performance, and security.



Introduction to Java

- Key Features
  - Write Once, Run Anywhere (WORA):
    - Java's platform independence allows code to run on any device with a Java Virtual Machine (JVM).
  - Object-Oriented:
    - Java follows the Object-Oriented Programming (OOP) paradigm, promoting modularity and reusability.
  - Platform Independence:
    - Java applications can run on various platforms without modification, thanks to the JVM.
  - Rich Standard Library:
    - Java provides a comprehensive standard library that simplifies development and enhances productivity.

Introduction to Java

- Application Domains
  - Web Development:
    - Java is widely used in server-side web development, with frameworks like Spring and JavaServer Faces (JSF).
  - Mobile Development:
    - Android applications are predominantly developed using Java.
  - Enterprise Applications:
    - Java is a popular choice for building scalable and robust enterprise-level applications.
  - Big Data and Analytics:
    - Hadoop and Apache Spark, two prominent tools for big data processing, are written in Java.

Introduction to Java

- Application Domains
  - Web Development:
    - Java is widely used in server-side web development, with frameworks like Spring and JavaServer Faces (JSF).
  - Mobile Development:
    - Android applications are predominantly developed using Java.
  - Enterprise Applications:
    - Java is a popular choice for building scalable and robust enterprise-level applications.
  - Big Data and Analytics:
    - Hadoop and Apache Spark, two prominent tools for big data processing, are written in Java.

Introduction to Java

- **History of Java**
  - Origins: Developed by James Gosling and Mike Sheridan at Sun Microsystems in the early 1990s.
  - Name Change: Originally named "Oak," renamed to Java in 1995.
  - WORA Principle: "Write Once, Run Anywhere" - Java aimed for platform independence with the introduction of the Java Virtual Machine (JVM).
  - Major Releases:
  - Java 1.0 (1996): Initial release with applets and Swing GUI toolkit.
  - J2SE 1.2 (1998): Introduced Swing components, inner classes, and the Collections Framework.
  - J2EE (Late 1990s): Focused on enterprise applications.
  - …….
  - Java 17 (Java SE 17 - September 14, 2021): A long-term support (LTS) release with features like pattern matching for switch, Sealed Classes, and more.
- Modern Era: Continual evolution under Oracle, regular releases, strong community involvement, and Java remains a widely-used and versatile programming language.

Introduction to Java

- **Java Buzzwords**
  - Java buzzwords refer to the set of fundamental characteristics and features that define the design and philosophy of the Java programming language.
  - These buzzwords capture the key principles and goals that Java aimed to achieve when it was created.

Introduction to Java

- **Java Buzzwords**
  - Simple:
    - Java was designed to be simple and easy to learn. It avoids complex features found in other languages like C++.
  - Object-Oriented:
    - Java is based on the Object-Oriented Programming (OOP) paradigm, encouraging the use of classes and objects for building modular and reusable code.
  - Platform-Independent:
    - "Write Once, Run Anywhere" (WORA) is a key concept. Java code can run on any device with a Java Virtual Machine (JVM), providing platform independence

Introduction to Java

- **Java Buzzwords**
  - Distributed:
    - Java has built-in support for creating distributed applications, enabling seamless communication between devices over a network.
  - Multithreaded:
    - Java supports concurrent execution of multiple threads within a program. This allows developers to create efficient, concurrent applications.
  - Secure:
    - Security is a crucial aspect of Java. The language includes features like bytecode verification and runtime checks to provide a secure execution environment.

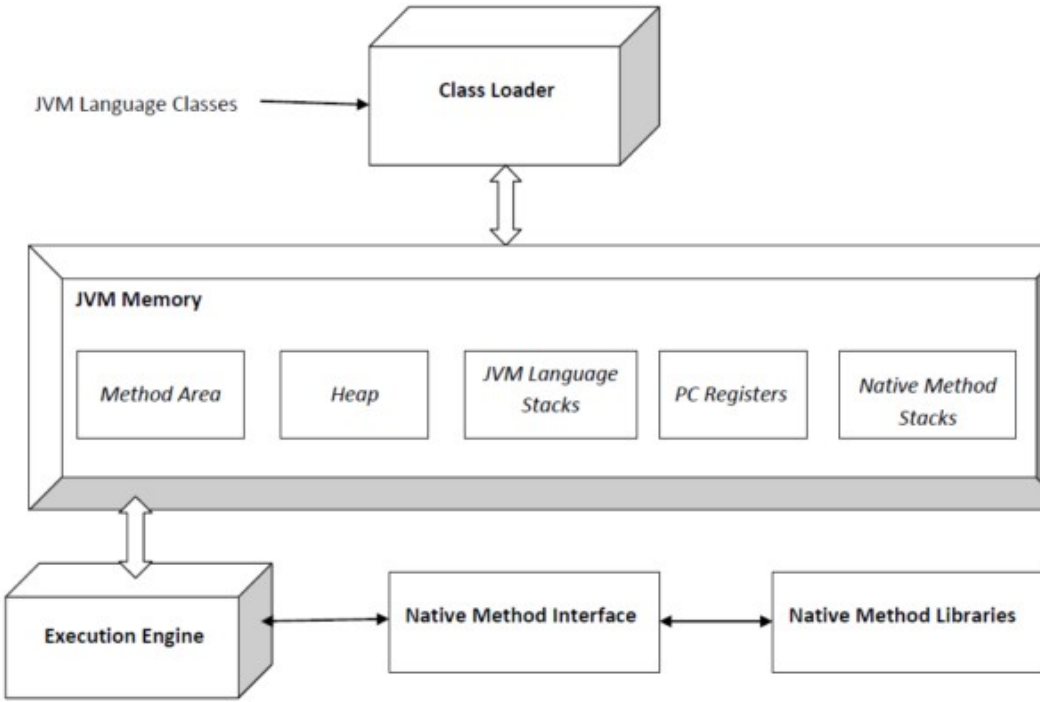
Introduction to Java

- **Java Buzzwords**
  - Dynamic:
    - Java supports dynamic memory allocation during runtime. It also allows dynamic linking and loading of classes, enhancing flexibility.
  - High Performance:
    - Java applications are compiled to bytecode, which is then interpreted by the JVM. Just-In-Time (JIT) compilation contributes to high-performance execution.
  - Robust:
    - Java emphasizes robustness through features like strong memory management, automatic garbage collection, and built-in exception handling.

Introduction to Java

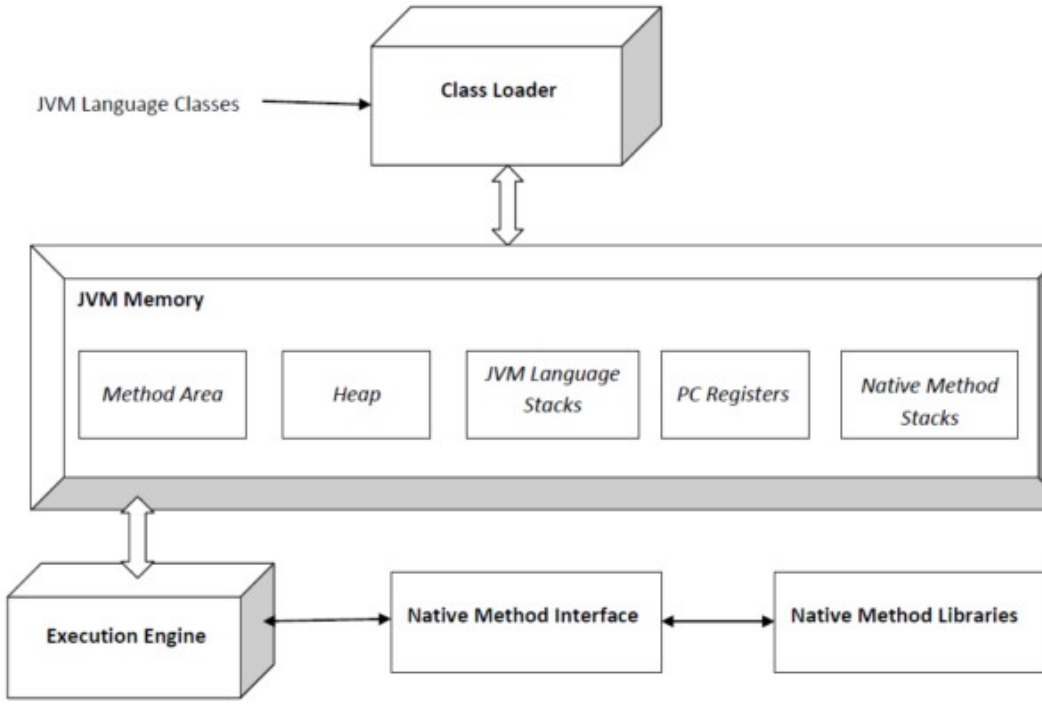
- **Java Buzzwords**
  - Architectural-Neutral:
    - Java applications are designed to be architecture-neutral, meaning they are not tied to any specific hardware or operating system.
  - Portable:
    - Code portability is a key goal, allowing Java applications to run on diverse platforms without modification.
  - Interpreted and Compiled:
    - Java uses a combination of interpreted and compiled approaches. Code is initially compiled to bytecode and then interpreted by the JVM.

- **Java Virtual Machine**
  - JVM is specifically responsible for converting bytecode to machine-specific code and is necessary in both JDK and JRE.
  - It is also platform-dependent and performs many functions, including memory management and security.
  - In addition, JVM can run programs written in other programming languages that have been translated to Java bytecode.

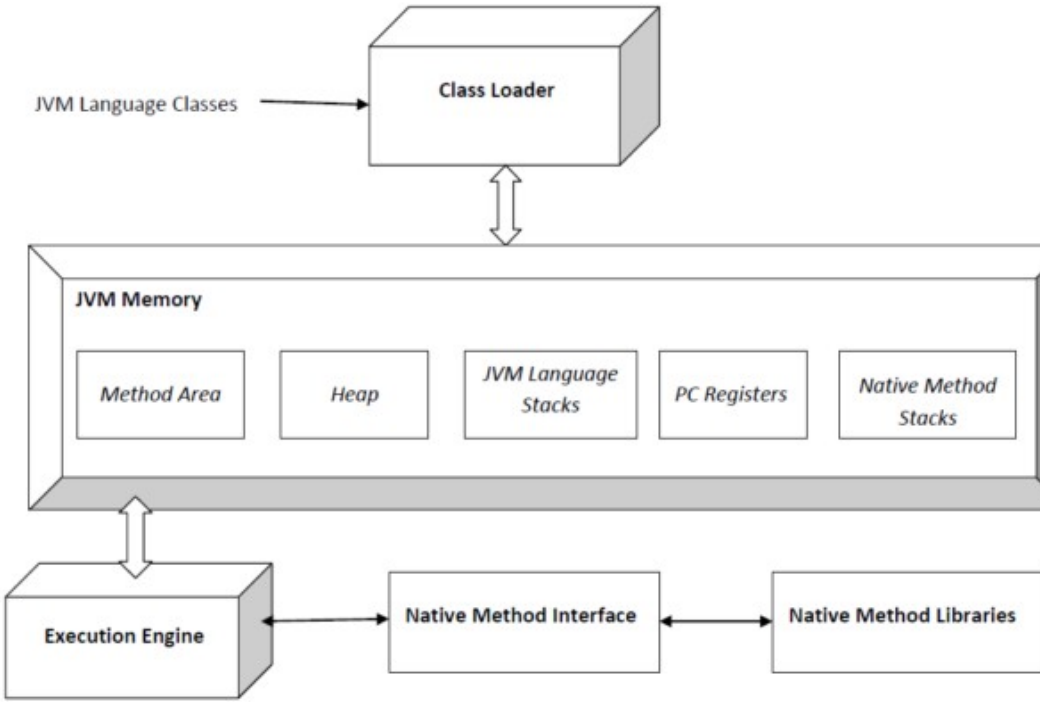


Introduction to Java

- **Java Virtual Machine**
  - JVM consists of three main components or subsystems:
  - **Class Loader Subsystem** is responsible for loading, linking and initializing a Java class file (i.e., “Java file”), otherwise known as dynamic class loading.
  - **Runtime Data Areas** contain method areas, PC registers, stack areas and threads.
  - **Execution Engine** contains an interpreter, compiler and garbage collection area.



- **Java Virtual Machine**
  - How Does the Java Virtual Machine Work?
    - Loading the bytecode
    - Verifying the bytecode
    - Preparing memory resources
    - Interpreting the Java bytecode
    - Just-in-time Compilation
    - Garbage Collection



- **JDK vs. JRE vs. JVM: Key differences**
  - JDK is the development platform, while JRE is for execution.
  - JVM is the foundation, or the heart of Java programming language, and ensures the program's Java source code will be platform-agnostic.
  - JVM is included in both JDK and JRE – Java programs won't run without it.

Assignments

- Define the concept of "Write Once, Run Anywhere" in the context of Java. How does the Java Virtual Machine (JVM) contribute to achieving this principle?
- Discuss the key motivations behind the creation of Java. What were the initial goals, and how did Java evolve over its early releases?
- Explain two of the Java buzzwords (e.g., simplicity, platform independence). How do these characteristics contribute to the appeal and success of Java as a programming language?
- What is the role of the Java Virtual Machine (JVM) in the execution of Java programs? How does it contribute to Java's platform independence?
- Compare and contrast the roles and components of Java Virtual Machine (JVM), Java Runtime Environment (JRE), and Java Development Kit (JDK). Highlight the specific functionalities, target users, and scenarios in which each plays a crucial role in the Java ecosystem.