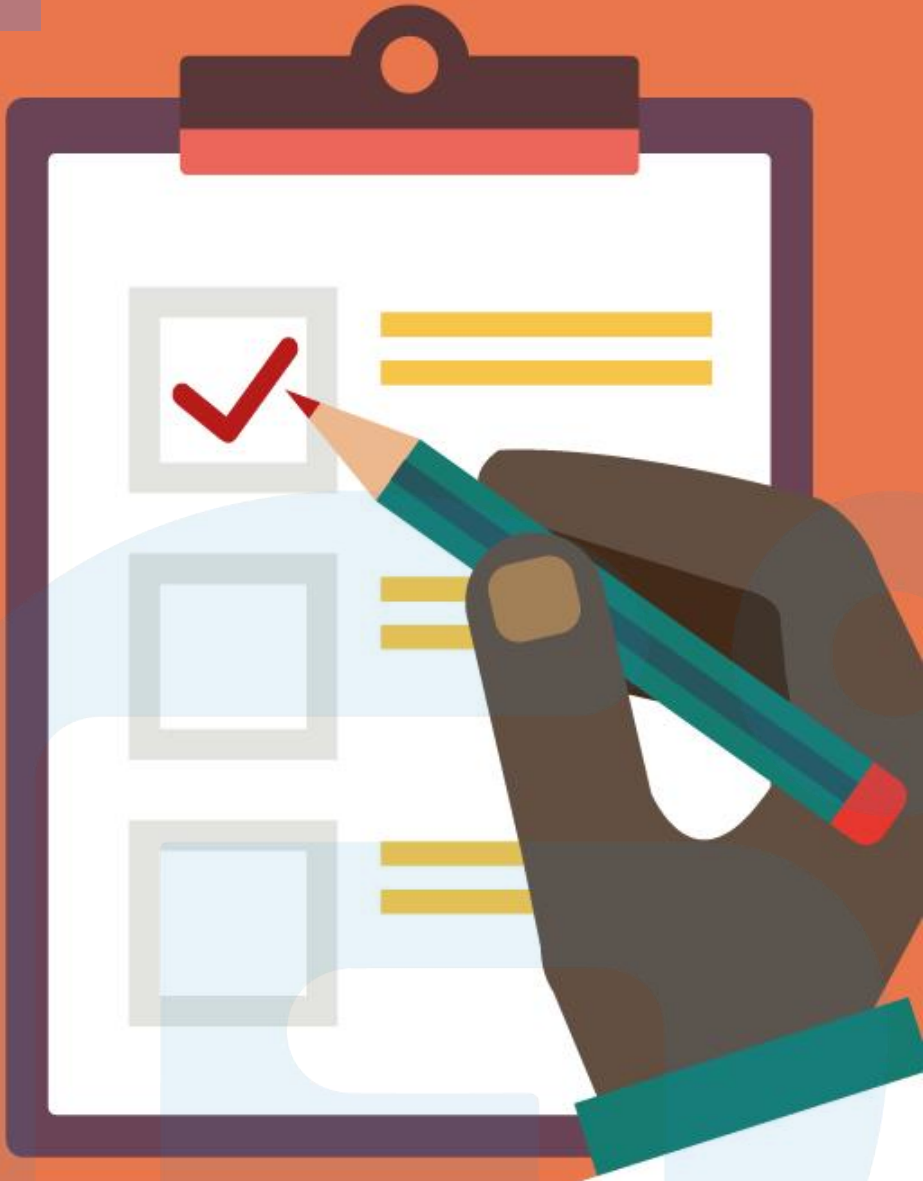




Lecture 1: Introduction to Java

CONTENTS OF THIS LESSON

- The History of Java Programming
 - Features of Java Language
 - Java Application Types
 - Java Virtual Machine
 - Installation of Java
-





The History of Java Programming



-
- Java is a programming language created by James Gosling from Sun Microsystems (Sun) in 1991. The target of Java is to write a program once and then run this program on multiple operating systems.
 - The first publicly available version of Java (Java 1.0) was released in 1995. Sun Microsystems was acquired by the Oracle Corporation in 2010.
 - Initially it was designed for small, embedded systems in electronic appliances like set-top boxes.
 - It was called "**Greentalk**" by James Gosling, and the file extension was .gt.
 - Later, it was renamed **Oak** and was developed as a part of the Green project.

"Java is not magical, it's a tool for people to build stuff. It's the hammer and nail. It's turns out to be easy to do things like AI and other fairly intelligent systems."

- James Gosling

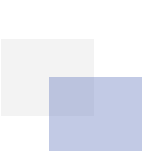
Quote.javaTpoint.com

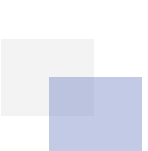





Features of Java Language

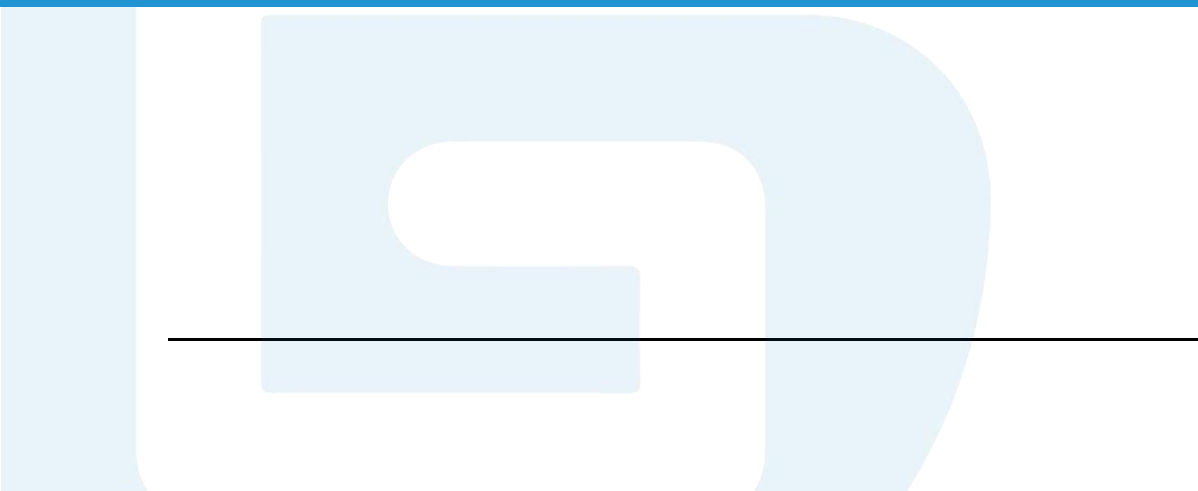


- 
-
- **Simple** - Java syntax is quite simple and easy to understand. Java is also easy to learn. Most of the features of C++ that were either difficult to understand or ambiguous are simplified in Java and some are omitted.
 - **Secure** - A Java program runs in Java Runtime Environment (JRE) that makes it secure. The JRE does not have interaction with system OS, so java programs do not interfere with the system. Additionally, the Java language has security features that allow us to develop virus-free, secure applications. This is why many banking applications are built on Java.
 - **Robust** - Java introduces automatic garbage collection and exception handling that helps to prevent any potential errors. Java also emphasizes on compile time and run time error checking. All these features make programs/applications developed in Java more robust.
 - **Portable** - The Java program compiled into Byte code can be run on any platform. It is implementation-independent and everything required to run the code like storage, data types, etc. are predefined.
 - **Object-oriented** - Java is a pure object-oriented programming language. Everything in Java is an object. The language features are easily extendible as there is a solid object-based model for programming. Java language supports all the major features of Object-oriented programming.
 - **Platform independent** - Java is a “write once, run anywhere” code. Unlike other programming languages that compile into machine-specific code, Java is compiled into a byte code that can run on any machine independent of the operating system. Any machine that runs Java Runtime Environment (JRE) can run this byte code.
-

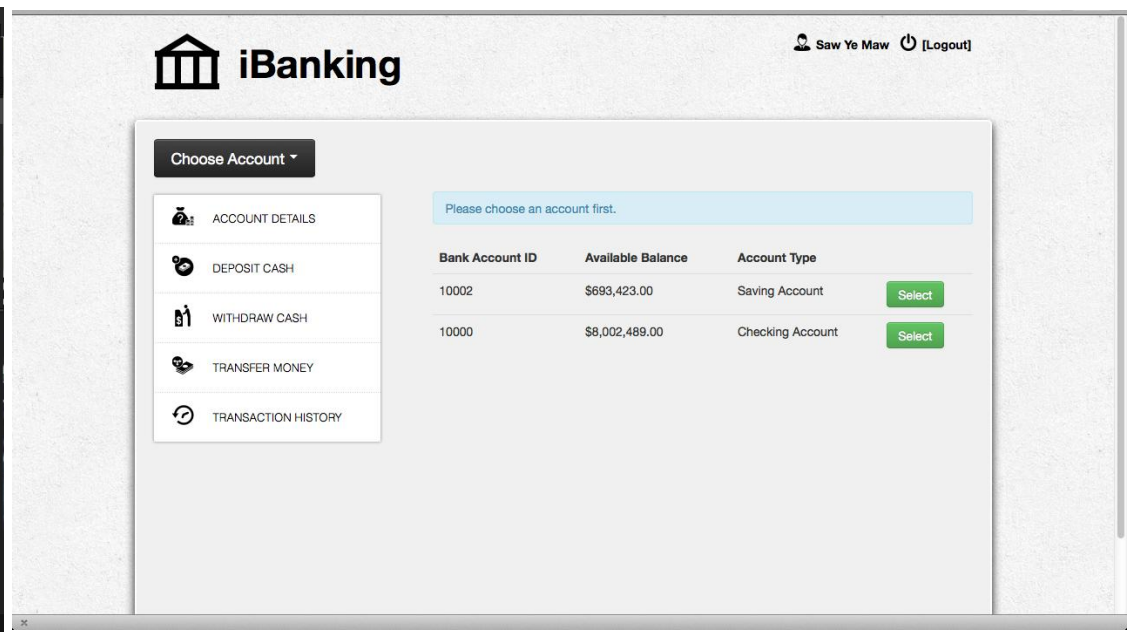
- 
-
- **Multithreaded** - Java's multithreading feature allows Java applications to perform multiple tasks simultaneously. Moreover, multiple threads utilize the same memory and other resources and carry out tasks simultaneously.
 - **Distributed** - Using Java we can develop distributed applications using advanced Java concepts like Remote Method Invocation (RMI) and Enterprise Java Beans (EJB).
-



Java Application Types

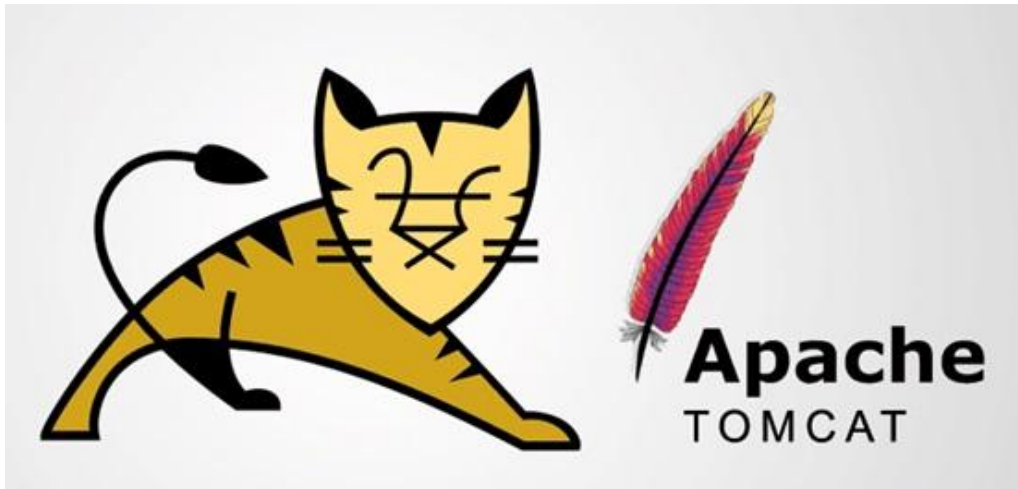


- **Standalone Applications** - These applications are usually independent and installable software on the desk top, such as media player, antivirus software, Java's multithreading feature allows Java applications to perform multiple tasks simultaneously. Moreover, multiple threads utilize the same memory and other resources and carry out tasks simultaneously.
- **Enterprise Applications** – These applications are distributed applications and have features, such as high security, load balancing, and clustering. The most popular enterprise applications are banking applications.



The most advanced Java Media Player: <https://github.com/goxr3plus/XR3Player>

- **Web Applications** - These applications run on the server-side and create dynamic web pages. These applications need to be highly dynamic and should be able to develop web pages on the go. Java features like JSP, servlets, struts, spring, hibernate, etc. are used for developing web applications.
- **Mobile Applications** – A mobile application is an application developed to run on mobile. Currently, Java ME and Android are used for developing mobile applications.



Apache Tomcat is a free and open-source implementation of the Jakarta Servlet, Jakarta Expression Language, and WebSocket technologies.. Based on Java, dynamic websites and application can be built and maintained on Tomcat Server.



Face Mask Detection Android App: <https://apkpure.com/face-mask-detector/com.larynx.facemaskdetector>



Java Virtual Machine



- **Java Virtual Machine (JVM)** – JVM is an abstract computing machine that enables a computer to run a Java program. The JVM architecture contains three major components: class loader, run time data area, and execution engine.

❖ **Class Loader**

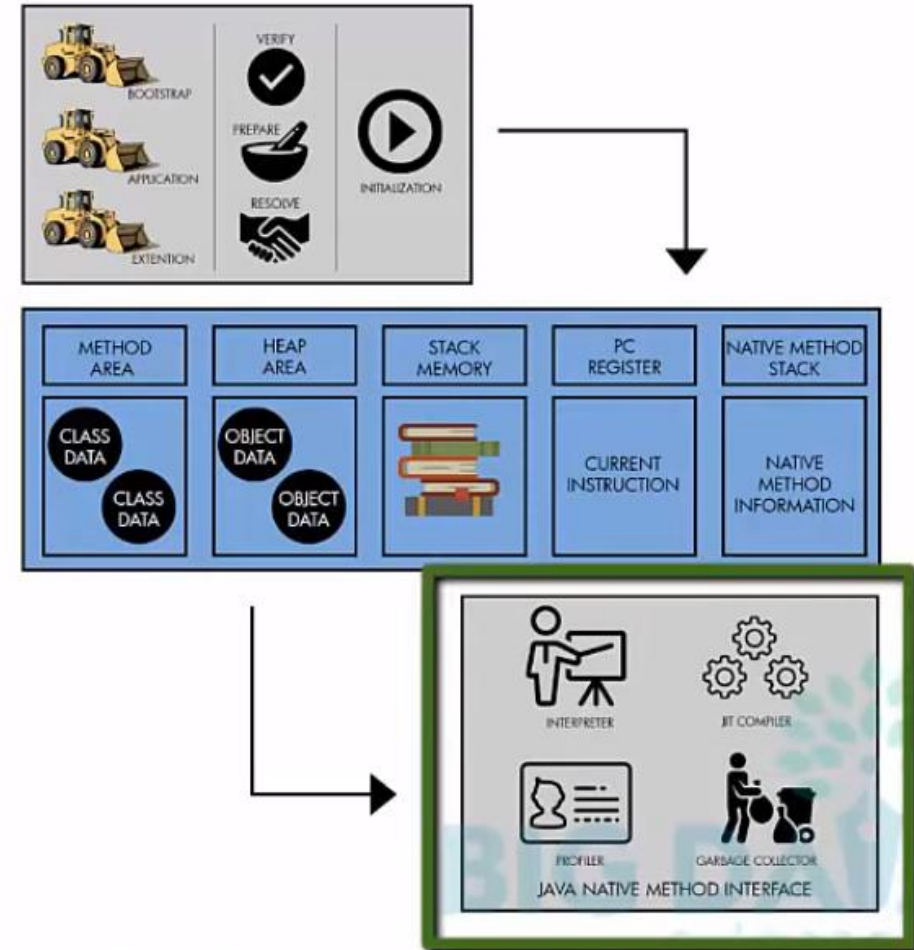
- Loading phase
- Linking phase
- Initialization phase

❖ **Run time data area**

- Method area
- Heap memory
- Stack memory
- PC registers
- JNI

❖ **Execution engine**

- Interpreter
- JIT compiler
- JNI
- Garbage collector



Class Loader

Loading - Three types of ClassLoader:

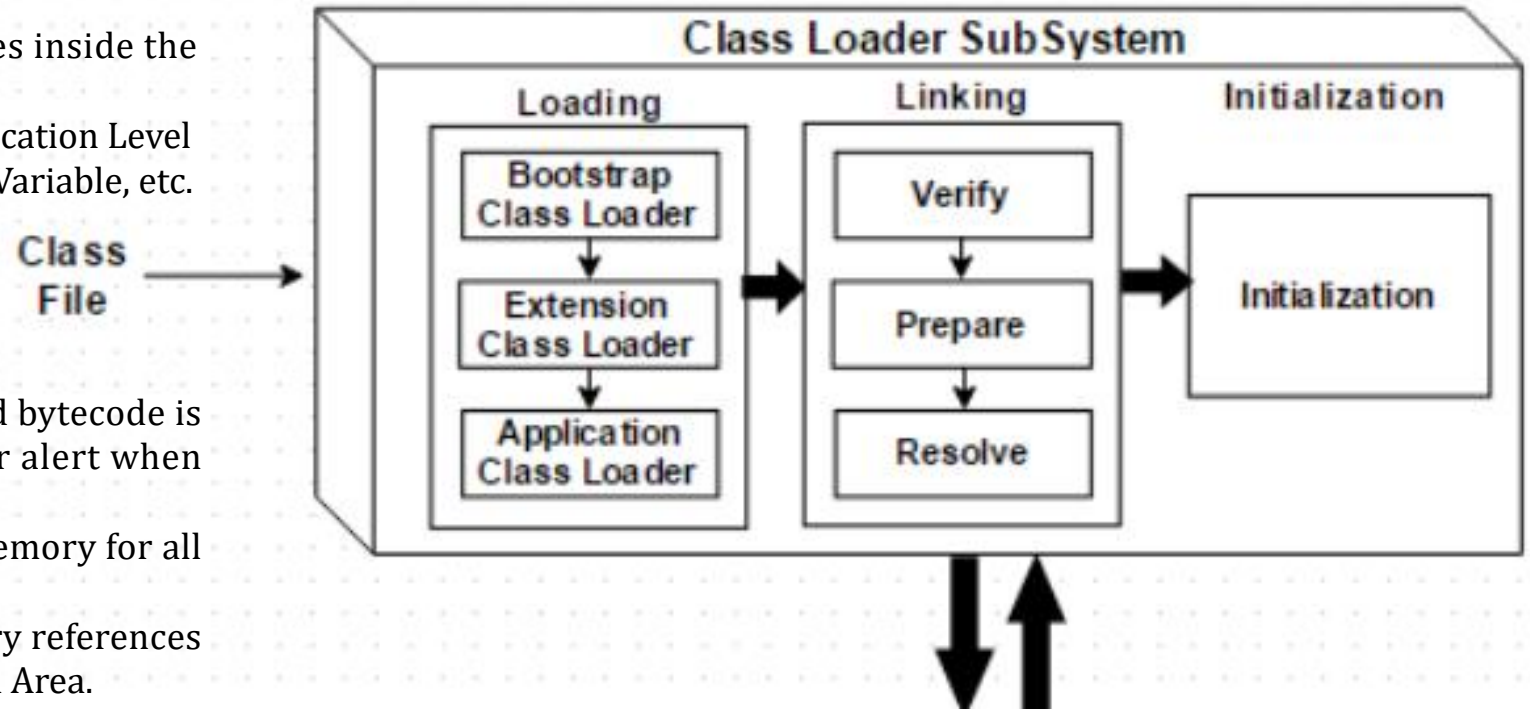
- Bootstrap ClassLoader**: loading classes from the bootstrap classpath `rt.jar`
- Extension ClassLoader**: loading classes inside the ext folder (`jre\lib`).
- Application ClassLoader**: loading Application Level Classpath, path mentioned Environment Variable, etc.

Linking - Three tasks:

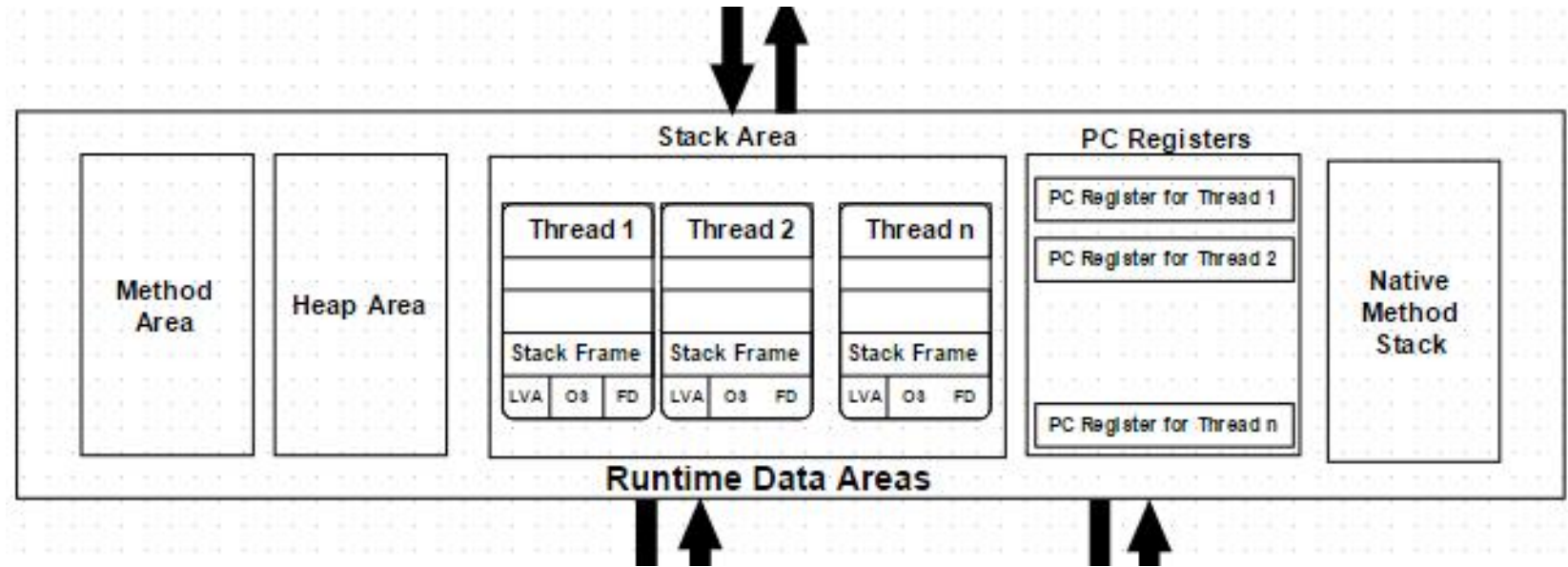
- Verify**: checking whether the generated bytecode is proper and giving the verification error alert when verification fails.
- Prepare**: assigning default values of memory for all static variables.
- Resolve**: replacing all symbolic memory references with the original references from Method Area.

Initialization:

Assigning all static variables with the original values and executing the static block.



Runtime Data Area



Method Area - All the class-level data is stored.

Heap Area - All the objects and their corresponding instance variables and arrays are stored.

Stack Area - A separate runtime stack will be created. For every method call, one entry will be made in Stack Frame. The Stack Frame is divided into three subentities: local variable array, operand stack, and frame data.

- **Local variable array** - Related to the method how many local variables are involved and the corresponding values will be stored.

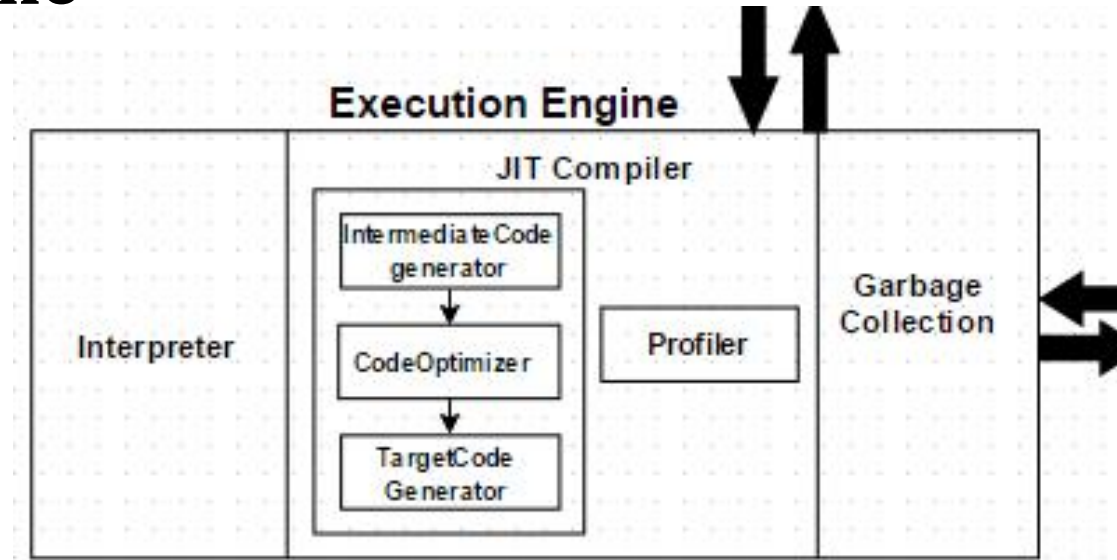
- **Operand stack** - If any intermediate operation is required to perform, operand stack acts as runtime workspace to perform the operation

- **Frame data** - All symbols corresponding to the method is stored.

PC Registers - Each thread will have separate PC Registers, to hold the address of current executing instruction once the instruction is executed the PC register will be updated with the next instruction.

Native Method stacks - Native Method Stack holds native method information. For every thread, a separate native method stack will be created.

Execution Engine



Interpreter – The interpreter interprets the bytecode faster but executes slowly. The disadvantage of the interpreter is that when one method is called multiple times, every time a new interpretation is required.

JIT Compiler - The JIT Compiler neutralizes the disadvantage of the interpreter. The Execution Engine will be using the help of the interpreter in converting byte code, but when it finds repeated code it uses the JIT compiler, which compiles the entire bytecode and changes it to native code. This native code will be used directly for repeated method calls, which improve the performance of the system. Four components are inside JIT Compiler: Intermediate Code Generator, Code Optimizer, Target Code Generator, Profiler.

- **Intermediate Code Generator** – Generating intermediate code.

- **Code Optimizer** – Optimizing the intermediate code.
- **Target Code Generator** – Generating Machine Code or Native Code.
- **Profiler** – Finding hotspots.

Garbage Collector - Collects and removes unreferenced objects. Garbage Collection can be triggered by calling **System.gc()**, but the execution is not guaranteed. Garbage collection of the JVM collects the objects that are created.



Installation of Java



Java Installation

Step 1: Check if Java is already installed in your device.

- Open a command prompt and run the following command:
> Java - version

```
C:\Users\boskom>java -version
'java' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\boskom>
```

If Java is already installed, the command will output the Java version in your device. If it isn't installed, it will be showed as above.

Step 2: Download the latest Java Development Kit installation file for your device, based on the operation system in your device (Windows, macOS, and Linux).

- Go to the Oracle Java Downloads page (<https://www.oracle.com/java/technologies/downloads/#jdk17-windows>)
- Download the corresponding version of Java for your device.

Step 3: Run the downloaded file.

Step 4: Set environmental variables in Java.

- Add Java to system variables.
- Add JAVA_HOME variable.

Please read [Java Installation Guidance](#) for more details!

Java Compiler

1. You can use free online Java Compiler to run your code.
onlinegdb:
https://www.onlinegdb.com/online_java_compiler
2. You can install Eclipse IDE
[Please read installation guidance for more details](#)





Thank you!
Any questions?

