Core Specific- Java Assignment 1

Q1. What is the difference between Compiler and Interpreter?

Ans: Difference Between Compiler and Interpreter:

Compiler	**Interpreter**
Steps of Programming: • Program Creation. • Analysis of language by the compiler and throws errors in case of any incorrect statement. • In case of no error, the Compiler converts the source code to Machine Code. • Linking of various code files into a runnable program. • Finally runs a Program.	**Steps of Programming:** • Program Creation. • Linking of files or generation of Machine Code is not required by Interpreter. • Execution of source statements one by one.
The compiler saves the Machine Language in form of Machine Code on disks.	The Interpreter does not save the Machine Language.
Compiled codes run faster than Interpreter.	Interpreted codes run slower than Compiler.
Linking-Loading Model is the basic working model of the Compiler.	The Interpretation Model is the basic working model of the Interpreter.
The compiler generates an output in the form of (.exe).	The interpreter does not generate any output.
Any change in the source program after the compilation requires recompiling the entire code.	Any change in the source program during the translation does not require retranslation of the entire code.
Errors are displayed in Compiler after Compiling together at the current time.	Errors are displayed in every single line.

Core Specific- Java Assignment 1

Q2. What is the difference between JDK, JRE, and JVM?

Ans:

1. **JDK** (Java Development Kit) is a Kit that provides the environment to **develop and execute(run)** the Java program. JDK is a kit(or package) that includes two things

- Development Tools(to provide an environment to develop your java programs)
- JRE (to execute your java program).

2. **JRE** (Java Runtime Environment) is an installation package that provides an environment to **only run(not develop)** the java program(or application)onto your machine. JRE is only used by those who only want to run Java programs that are end-users of your system.

3. **JVM (Java Virtual Machine)** is a very important part of both JDK and JRE because it is contained or inbuilt in both. Whatever Java program you run using JRE or JDK goes into JVM and JVM is responsible for executing the java program line by line, hence it is also known as an ***interpreter***.

Now let us discuss the components of JRE in order to understand its importance of it and perceive how it actually works. For this let us discuss components.

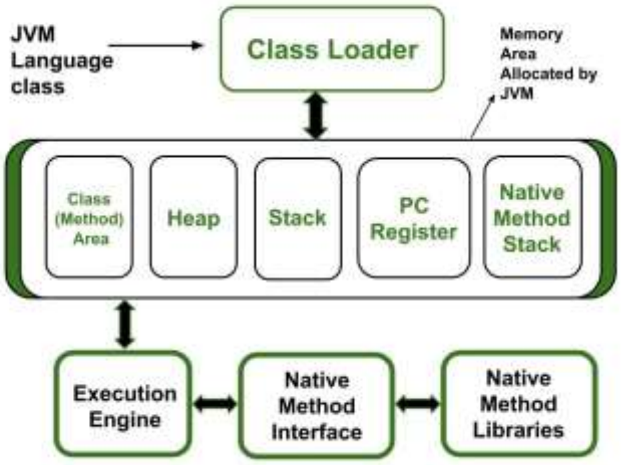
The components of JRE are as follows:

1. **Deployment technologies**, including deployment, Java Web Start, and Java Plug-in.
2. **User interface toolkits**, including *Abstract Window Toolkit (AWT), Swing, Java 2D, Accessibility, Image I/O, Print Service, Sound, drag, and drop (DnD)*, and *input methods*.
3. **Integration libraries**, including *Interface Definition Language (IDL), Java Database Connectivity (JDBC), Java Naming and Directory Interface (JNDI), Remote Method Invocation (RMI), Remote Method Invocation Over Internet Inter-Orb Protocol (RMI-IIOP)*, and *scripting*.
4. **Other base libraries**, including *international support, input/output (I/O), extension mechanism, Beans, Java Management Extensions (JMX), Java Native Interface (JNI), Math, Networking, Override Mechanism, Security, Serialization*, and *Java for XML Processing (XML JAXP)*.
5. **Lang and util base libraries**, including *lang and util, management, versioning, zip, instrument, reflection, Collections, Concurrency Utilities, Java Archive (JAR), Logging, Preferences API, Ref Objects*, and *Regular Expressions*.
6. **Java Virtual Machine (JVM)**, including *Java HotSpot Client* and *Server Virtual Machines*.

Q3. How many types of memory areas are allocated by JVM?

Ans:

Memory in the JVM is divided into 5 different parts:

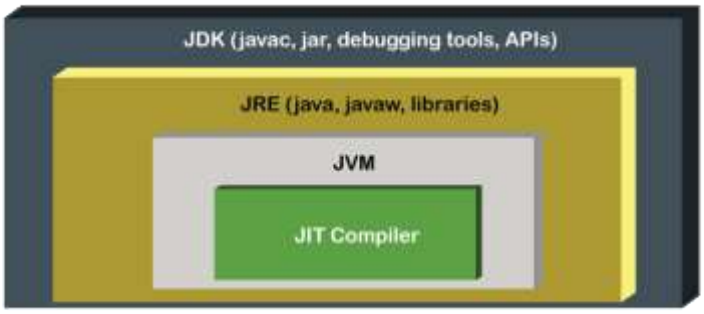


1. Class(Method) Area
2. Heap
3. Stack
4. Program Counter Register
5. Native Method Stack

Q4. What is JIT compiler?

Ans:

JIT in Java is an integral part of the **JVM**. It accelerates execution performance many times over the previous level. In other words, it is a long-running, computer-intensive program that provides the best performance environment. It optimizes the performance of the Java application at compile or run time.

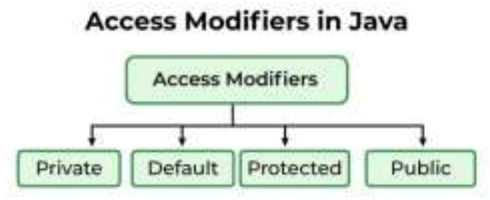

JDK (javac, jar, debugging tools, APIs)
JRE (java, javaw, libraries)
JVM
JIT Compiler

Q5. What are the various access specifiers in Java?

Ans:

There are four types of access modifiers available in Java:

1. Default – No keyword required
2. Private
3. Protected
4. Public



Q6. What is a compiler in Java?

Ans:

A compiler is a translator that converts the high-level language into the machine language. High-level language is written by a developer and machine language can be understood by the processor. Compiler is used to show errors to the programmer. The main purpose of compiler is to change the code written in one language without changing the meaning of the program. When you execute a program which is written in HLL programming language then it executes into two parts.

In the first part, the source program compiled and translated into the object program (low level language).

In the second part, object program translated into the target program through the assembler.

Q7. Explain the types of variables in Java?

Ans: There are three types of variables in Java:

- local variable
- instance variable
- static variable

Core Specific- Java Assignment 1

1) Local Variable

A variable declared inside the body of the method is called local variable. You can use this variable only within that method and the other methods in the class aren't even aware that the variable exists.

A local variable cannot be defined with "static" keyword.

2) Instance Variable

A variable declared inside the class but outside the body of the method, is called an instance variable. It is not declared as static.

It is called an instance variable because its value is instance-specific and is not shared among instances.

3) Static variable

A variable that is declared as static is called a static variable. It cannot be local. You can create a single copy of the static variable and share it among all the instances of the class. Memory allocation for static variables happens only once when the class is loaded in the memory.

Q8. What are the Datatypes in Java?

Ans:

Data types are divided into two groups:

- Primitive data types - includes byte, short, int, long, float, double, boolean and char
- Non-primitive data types - such as String, Arrays and Classes (you will learn more about these in a later chapter)

Q9. What are the identifiers in java?

Ans:

All Java **variables** must be **identified** with **unique names.**

These unique names are called **identifiers**.

Identifiers can be short names (like x and y) or more descriptive names (age, sum, totalVolume).

Note: It is recommended to use descriptive names in order to create understandable and maintainable code:

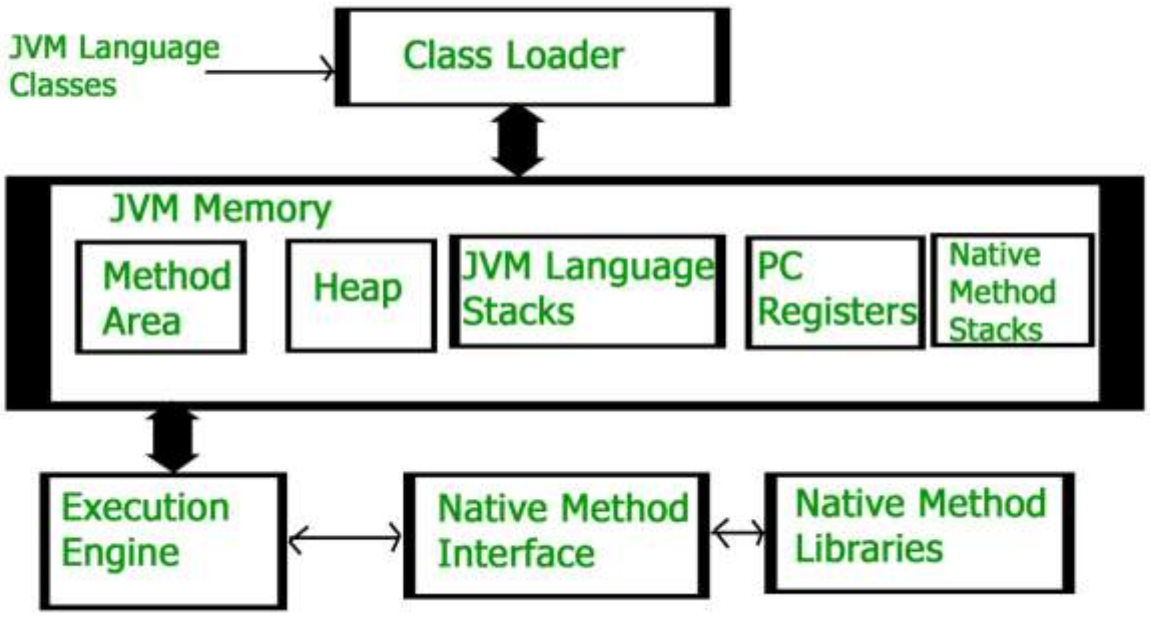
Q10. Explain the architecture of JVM .

Ans:

JVM(Java Virtual Machine) acts as a run-time engine to run Java applications. JVM is the one that actually calls the **main** method present in a java code. JVM is a part of JRE(Java Runtime Environment).

Java applications are called WORA (Write Once Run Anywhere). This means a programmer can develop Java code on one system and can expect it to run on any other Java-enabled system without any adjustment. This is all possible because of JVM.

When we compile a *.java* file, *.class* files(contains byte-code) with the same class names present in *.java* file are generated by the Java compiler. This *.class* file goes into various steps when we run it. These steps together describe the whole JVM.


JVM Language Classes
Class Loader
JVM Memory
Method Area
Heap
JVM Language Stacks
PC Registers
Native Method Stacks
Execution Engine
Native Method Interface
Native Method Libraries