# ASSIGNMENT NO :-2

## Q1:-Difference between JDK, JRE, and JVM ?

### JVM

JVM (Java Virtual Machine) is an abstract machine. It is called a virtual machine because it doesn't physically exist. It is a specification that provides a runtime environment in which Java bytecode can be executed. It can also run those programs which are written in other languages and compiled to Java bytecode.

JVMs are available for many hardware and software platforms. JVM, JRE, and JDK are platform dependent because the configuration of each OS is different from each other. However, Java is platform independent. There are three notions of the JVM: *specification*, *implementation*, and *instanc*

The JVM performs the following main tasks:

Loads code

Verifies code

Executes code

Provides runtime environment

### JRE

JRE is an acronym for Java Runtime Environment. It is also written as Java RTE. The Java Runtime Environment is a set of software tools which are used for developing Java applications. It is used to provide the runtime environment. It is the implementation of JVM. It physically exists. It contains a set of libraries + other files that JVM uses at runtime.

The implementation of JVM is also actively released by other companies besides Sun Micro Systems.

### JDK

JDK is an acronym for Java Development Kit. The Java Development Kit (JDK) is a software development environment which is used to develop Java applications and applets. It physically exists. It contains JRE + development tools.

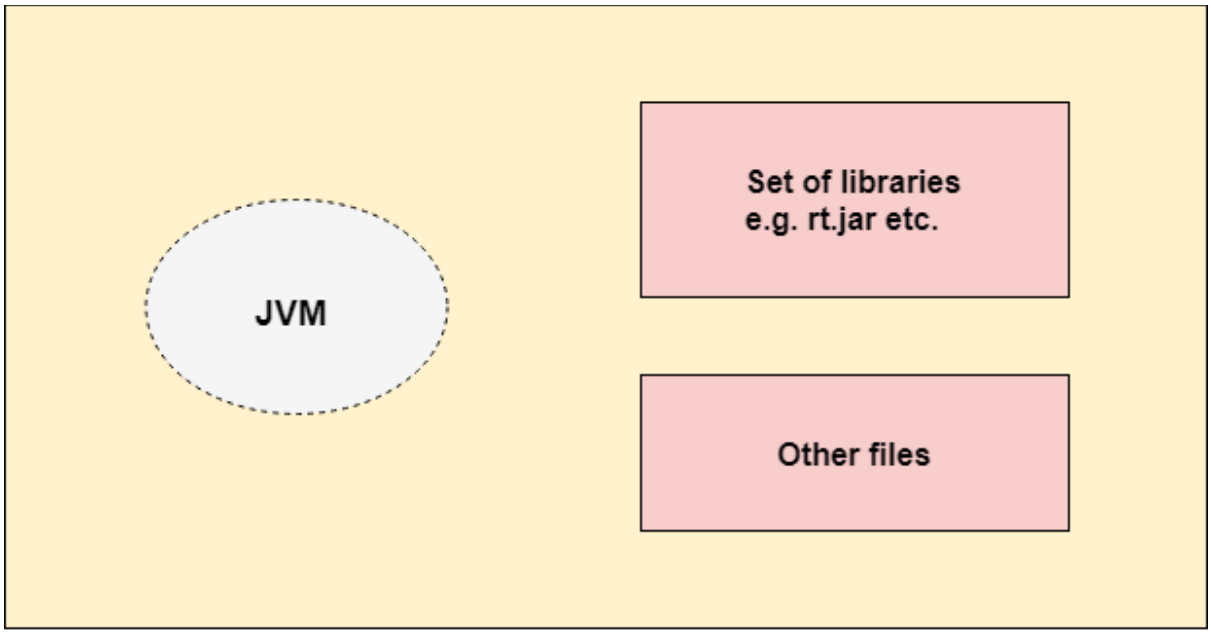
JDK is an implementation of any one of the below given Java Platforms released by Oracle Corporation:

Standard Edition Java Platform

Enterprise Edition Java Platform

Micro Edition Java Platform

The JDK contains a private Java Virtual Machine (JVM) and a few other resources such as an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc), etc. to complete the development of a Java Application.




## Q2:- WHAT IS JIT COMPILER?

When we write a program in any programming language it requires converting that code in the machine-understandable form because the machine only understands the binary language. According to the programming languages, compiler differs. The compiler is a program that converts the high-level language to machine level code. The Java programming language uses the compiler named javac. It converts the high-level language code into machine code (bytecode). JIT is a part of the JVM that optimizes the performance of the application. JIT stands for Java-In-Time Compiler. The JIT compilation is also known as dynamic compilation. In this section, we will learn what is JIT in Java, its working, and the phases of the JIT compiler.

### What is JIT in Java?

**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

## Q3:- WHAT IS Class Loader ?

Java ClassLoader is an abstract class. It belongs to a **java.lang** package. It loads classes from different resources. Java ClassLoader is used to load the classes at run time. In other words, JVM performs the linking process at runtime. Classes are loaded into the JVM according to need. If a loaded class depends on another class, that class is loaded as well. When we request to load a class, it delegates the class to its parent. In this way, uniqueness is maintained in the runtime environment. It is essential to execute a Java program.

Java ClassLoader is based on three principles: **Delegation**, **Visibility**, and **Uniqueness**.

**Delegation principle:** It forwards the request for class loading to parent class loader. It only loads the class if the parent does not find or load the class.

**Visibility principle:** It allows child class loader to see all the classes loaded by parent ClassLoader. But the parent class loader cannot see classes loaded by the child class loader.

**Uniqueness principle:** It allows to load a class once. It is achieved by delegation principle. It ensures that child ClassLoader doesn't reload the class, which is already loaded by the parent.

Types of ClassLoader

In Java, every ClassLoader has a predefined location from where they load class files. There are following types of ClassLoader in Java:

Bootstrap Class Loader: It loads standard JDK class files from rt.jar and other core classes. It is a parent of all class loaders. It doesn't have any parent. When we call String.class.getClassLoader() it returns null, and any code based on it throws NullPointerException. It is also called Primordial ClassLoader. It loads class files from jre/lib/rt.jar. For example, java.lang package class.

**Extensions Class Loader:** It delegates class loading request to its parent. If the loading of a class is unsuccessful, it loads classes from jre/lib/ext directory or any other directory as java.ext.dirs. It is implemented by sun.misc.Launcher$ExtClassLoader in JVM.

**System Class Loader:** It loads application specific classes from the CLASSPATH environment variable. It can be set while invoking program using -cp or classpath command line options. It is a child of Extension ClassLoader. It is implemented by sun.misc.Launcher$AppClassLoader class. All Java ClassLoader implements java.lang.ClassLoader.

# Q4:- Explain various memory logical partitions?

**Memory partitioning** means dividing the main memory into chunks of the same or different sizes so that they can be assigned to processes in the main memory.

There are two types of memory partitioning techniques:

**Fixed-sized memory partitioning**

**Variable-sized memory partitioning**

## Fixed-sized memory partitioning

In fixed-sized memory partitioning, the main memory is divided into blocks of the same or different sizes. Fixed-size memory partitioning can take place before executing any processes or during the configuration of the system.

Consider an example of a main memory of 80MB. Let's suppose that the main memory is divided into blocks of 10MB as shown below:

The operating system is assigned the first block in the memory.

Let's say we bring 3 processes, *process 1*, *process 2*, and *process 3*, from hard disk to main memory. *Process 1*, *process 2*, and *process 3* are of sizes 6MB, 8MB, and 5MB respectively.

To assign the processes in the main memory, the operating system will try to find a single block that is large enough to store the process. In other words, the operating system will find a block whose size is greater than or equal to the size of the process. After loading the three processes into the memory, the memory will look like this:

## Dynamic partitioning

To overcome difficulties with fixed partitioning, partitioning may be done dynamically, called dynamic partitioning.With it, the main memory portion for user applications is initially a single contiguous block.

When a new process is created, the exact amount of memory space is allocated to the process. Similarly when no enough space is available, a process may be swapped out temporarily to release space for a new process .

As time goes on, there will appear many small holes in the main memory, which is referred to as external fragmentation.

Thus although much space is still available, it cannot be allocated to new processes. A method for overcoming external fragmentation is compaction.

## Q5:-what is java its “write once and run anywhere nature”?

Java is **a widely-used programming language for coding web applications**. It has been a popular choice among developers for over two decades, with millions of Java applications in use today. Java is a multi-platform, object-oriented, and network-centric language that can be used as a platform in itself.

Java gets its WORA nature from its bytecode. JAVA codes or programs are typed by the programmer in high-level user-friendly language and they are converted into a class file (also known as bytecode), an intermediate language before being converted into machine code. The point that is being emphasized upon here is that you can write JAVA code on any device, on any machine or platform and the class file that will be created remains the same all throughout.

This means that the same JAVA code can be run on any platform. So you don't have to write the code separately for a Linux device if you've written a certain JAVA code on a Windows device. In this way, WORA is achieved.

This is also one of the top java interview questions and to understand such concepts much more clearly.you can take up some basic java tutorials.

## Q6:-Explain History of java. who invented java?

The history of Java is very interesting. Java was originally designed for interactive television, but it was too advanced technology for the digital cable television industry at the time. The history of Java starts with the Green Team. Java team members (also known as Green Team), initiated this project to develop a language for digital devices such as set-top boxes, televisions, etc. However, it was best suited for internet programming. Later, Java technology was incorporated by Netscape.

The principles for creating Java programming were "Simple, Robust, Portable, Platform-independent, Secured, High Performance, Multithreaded, Architecture Neutral, Object-Oriented, Interpreted, and Dynamic". Java was developed by James Gosling, who is known as the father of Java, in 1995. James Gosling and his team members started the project in the early '90s.

Currently, Java is used in internet programming, mobile devices, games, e-business solutions, etc. Following are given significant points that describe the history of Java.

1) **James Gosling**, Mike Sheridan, and Patrick Naughton initiated the Java language project in June 1991. The small team of sun engineers called Green Team.

2) Initially it was designed for small, embedded systems in electronic appliances like set-top boxes.

3) Firstly, it was called "Greentalk" by James Gosling, and the file extension was .gt.

4) After that, it was called Oak and was developed as a part of the Green project.

5) Why Oak? Oak is a symbol of strength and chosen as a national tree of many countries like the U.S.A., France, Germany, Romania, etc.

6) In 1995, Oak was renamed as "Java" because it was already a trademark by Oak Technologies.

## Q7:- What is original name of java ? why it was renamed ?

Initially, when James Gosling developed the language he named it **Oak** while he was staring at an **oak tree** outside his office window (in 1991).

The whole team behind the development of Java was searching for another good name because it was discovered that “Oak” was the name of the programming language of Sun Microsystems that Gosling had created initially for their set-top box project, which created **copyright** issues. But they couldn’t find one.

Then one day, after several hours of brainstorming, they got an inspiration when they took a trip to a local coffee shop with Gosling. Finally, the name **Java** came from sGosling, **A**rthur **V**an Hoff and **A**ndy Bechtolsheim - YES, THAT’S WHERE **JAVA** CAME FROM.

Later, the **logo** of Java was designed to depict “**coffee**”, as it is till now.

## Q8:- List features of java.

A list of the most important features of the Java language is given below.

simple

Object-Oriented

Portable

Platform independent

Secured

Robust

Architecture neutral

Interpreted

High Performance

Multithreaded

Distributed

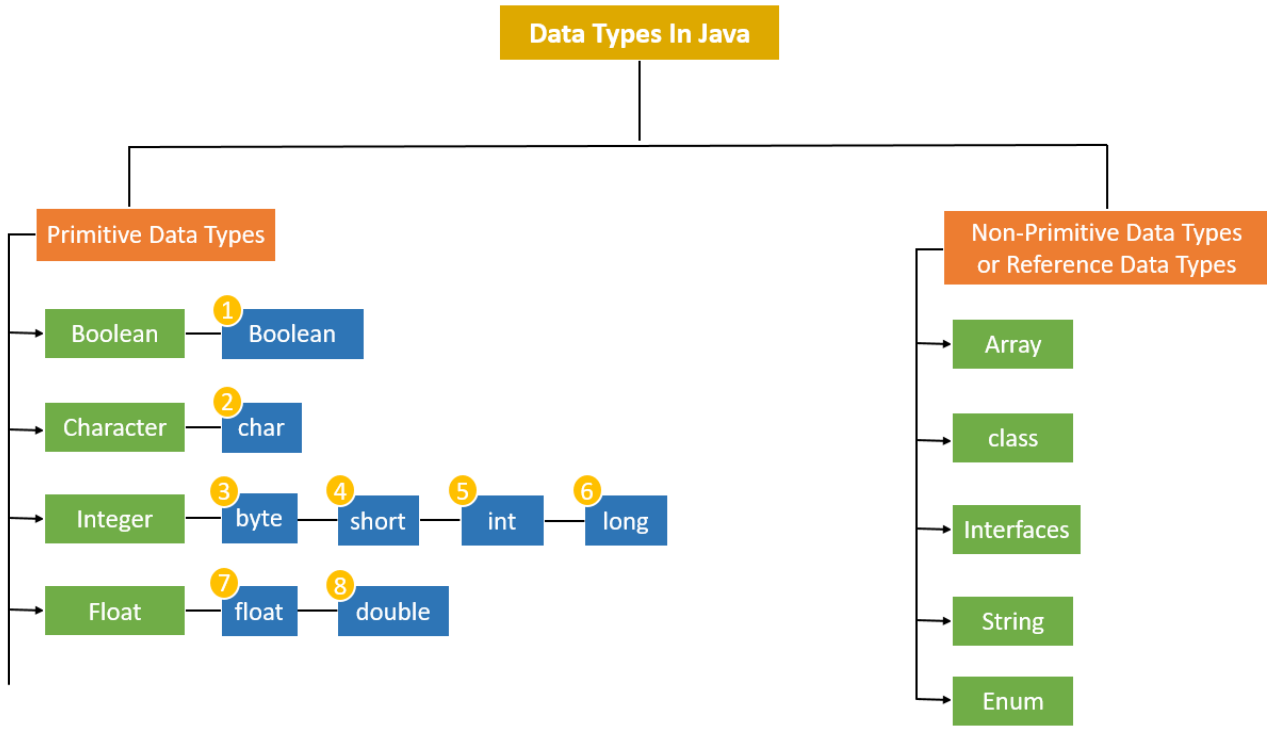
Dynamic

# Q9:- Data Types in Java.

Data types in Java are divided into 2 categories:

Primitive Data Types

Non-Primitive Data Types




# Q10:- What is difference between

## System.in

## System.out

## System.err.

## System.in

`System.in` is an InputStream which is typically connected to keyboard input of console programs. In other words, if you start a Java application from the command line, and you type something on the keyboard while the CLI console (or terminal) has focus, the keyboard input can typically be read via System.in from inside that Java application. However, it is only keyboard input directed to that Java application (the console / terminnal that started the application) which can be read via System.in. Keyboard input for other applications cannot be read via System.in .

System.in is not used as often since data is commonly passed to a command line Java application via command line arguments, files, or possibly via network connections if the

application is designed for that. In applications with GUI the input to the application is given via the GUI. This is a separate input mechanism from System.in.

### System.out

`System.out` is a **PrintStream** to which you can write characters. System.out normally outputs the data you write to it to the CLI console / terminal. System.out is often used from console-only programs like command line tools as a way to display the result of their execution to the user. This is also often used to print debug statements of from a program (though it may arguably not be the best way to get debug info out of a program).

System.err.

`System.err` is a `PrintStream`. `System.err` works like `System.out` except it is normally only used to output error texts. Some programs (like Eclipse) will show the output to `System.err` in red text, to make it more obvious that it is error text.

## Q11:- How java is platform independent?

Java is platform-independent because it uses a virtual machine. The Java programming language and all APIs are compiled into bytecodes. Bytecodes are effectively platform-independent. The virtual machine takes care of the differences between the bytecodes for the different platforms. The run-time requirements for Java are therefore very small. The Java virtual machine takes care of all hardware-related issues so that no code has to be compiled for different hardware.

## Q12:- What is bytecode? how is different from machine code in java?

Both of these are codes that act as a set of instructions that help machines/ devices behave in a specified manner or perform certain operations/ tasks. The primary difference between byte code and machine code is that bytecode is an intermediate code while the machine code is the final code that the CPU processes.

### Difference between Byte Code and Machine Code:

| S.NO. | Byte Code | Machine Code |
|---|---|---|

| | | |
|---|---|---|
| 01. | Byte Code consisting of binary, hexadecimal, macro instructions like (new, add, swap, etc) and it is not directly understandable by the CPU. It is designed for efficient execution by software such as a virtual machine.intermediate-level | Machine code consisting of binary instructions that are directly understandable by the CPU. |
| 02. | Byte code is considered as the intermediate-level code. | Machine Code is considered as the low-level code. |
| 03. | Byte code is a non-runnable code generated after compilation of source code and it relies on an interpreter to get executed. | Machine code is a set of instructions in machine language or in binary format and it is directly executed by CPU. |
| 04. | Byte code is executed by the virtual machine then the Central Processing Unit. | Machine code is not executed by a virtual machine it is directly executed by CPU. |
| 05. | Byte code is less specific towards machine than the machine code. | Machine code is more specific towards machine than the byte code. |
| 06. | It is platform-independent as it is dependent on the virtual machine and the system having a virtual machine can be executed irrespective of the platform. | It is not platform independent because the object code of one platform can not be run on the same Operating System. Object varies depending upon system architecture and native instructions associated with the machine. |
| 07. | All the source code need not be converted into byte code for execution by CPU. Some source code written by any specific high-level language is | All the source code must be converted into machine code before it is executed by the CPU. |

| | | |
|---|---|---|
| | converted into byte code then byte code to object code for execution by CPU. | |

## Q13:- What is the difference between jar file and runnable jar file?

**I. Runnable JAR File**

Click the "File" button in the upper left corner of Eclipse and select "Export".

Select "Runnable JAR File" under Java and click "Next".

Direct "Finish".

**II. JAR File**

File>export.

Select "JAR File" under Java and click "Next".

Click "Next".

Click "Next".

Select the file where the main function is located and click "Finish".

How to execute the jar.
The command line can be run via Java-jar < filename >.jar;
If there is no graphical interface, it can only be run in the command line mode;
If you do a GUI, you can run it directly by double-clicking.

What happens if the export jar file does not select the main function.

### the difference between jar file and runnable jar file.

The runnable jar contains a MANIFEST. MF file, which defines the Main class to being executed when the jar is run. The "Main-class" must be defined and the Java runtime knows which Class to call when the jar is "run."
If a jar does not include a manifest with the "Main-class:" It's not considered a "runnable jar"-it's just a library O F Java code.

## Q14:- what is difference between runnable jar file & Executable jar file?

**Jar file** is the combination of compiled java classes.
**Executable jar file** is also be combination of compiled java classes with Main class.

## Q15:- How is c platform dependent language?

Whenever we install C software, depending on the operating system we need to download and installed it. Let say we want to install C on Windows and Mac operating system. Windows understands .exe and MAC understands .dmg file. We also know that every application is a standalone application including programming languages. So, all the programming languages are stand-alone applications only. So, we need to download and installed it based on the operating system.

Whenever we install C, Compiler (to compile the application and generate machine code) + Library (to develop application) will be installed in the system. The Windows compiler will work for the Windows operating system only and the MAC compiler will work only for MAC operating system.

## Q16:- what is the difference between path and class path?

differences are like...

1).**Path** is an environment variable which is used by the operating system to find the executables.

**Classpath** is an environment variable which is used by the Java compiler to find the path, of classes.ie in J2EE we give the path of jar files.

2).**PATH** is nothing but setting up an environment for operating system. Operating System will look in this PATH for executables.

**Classpath** is nothing but setting up the environment for Java. Java will use to find compiled classes

3).**Path** refers to the system while **classpath** refers to the Developing Envornment.

In **path** we set the path of executables while in
**classpath** we set path of jars for compiling classes.