1. What is the difference between JDK and JRE?

JDK stands for Java Development Kit. It contains the tools and

libraries for development of Java programs. It also contains

compilers and debuggers needed to compile Java program,

JRE stands for Java Runtime Environment. This is included in JDK.

JRE provides libraries and JVM that is required to run a Java

program.

1. What is Java Virtual Machine (JVM)?

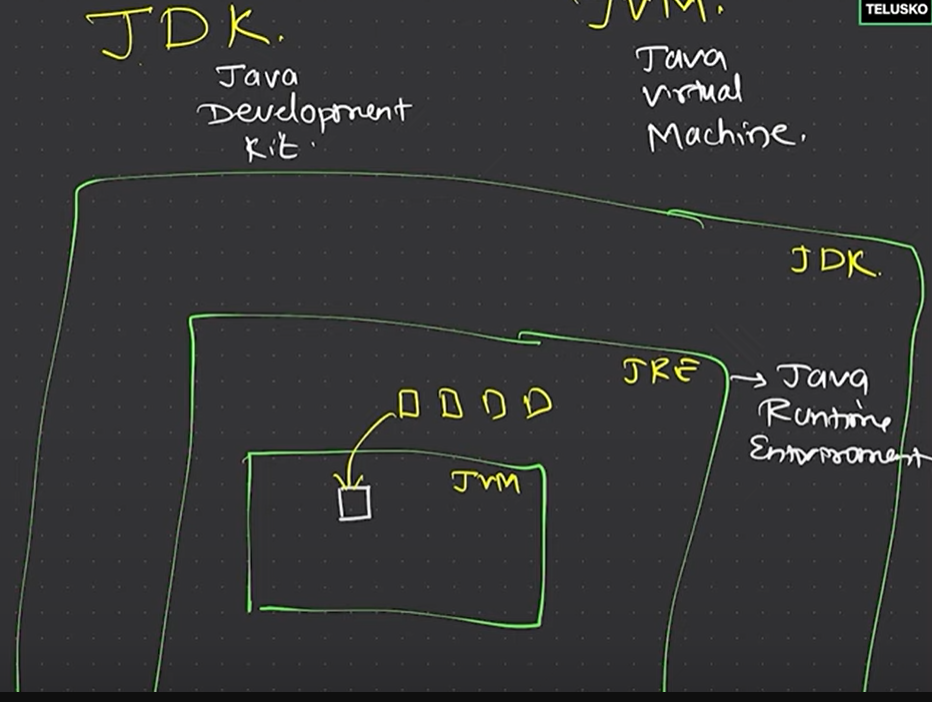
Java Virtual Machine (JVM) is an abstract machine that executes

Java Bytecode. There are different JVM for different hardware and

software platforms. So JVM is platform dependent. JVM is

responsible for loading, verifying and executing the Bytecode on a

platform.



1. What are the different types ofmemory areas allocated by JVM?

In java, JVM allocates memory to different processes, methods and

objects. Some of the memory areas allocated by JVM are:

1. ClassLoader: It is a component of JVM used to load class

files.

2. Class (Method) Area: It stores per-class structures such as

the runtime constant pool, field and method data, and the

code for methods.

3. Heap: Heap is created a runtime and it contains the runtime

data area in which objects are allocated.

4. Stack: Stack stores local variables and partial results at

runtime. It also helps in method invocation and return

value. Each thread creates a private JVM stack at the time

of thread creation.

5. Program Counter Register: This memory area contains the

address of the Java virtual machine instruction that is

currently being executed.

6. Native Method Stack: This area is reserved for all the

native methods used in the application.

1. What is JIT compiler?

Just In Time compiler also known as JIT compiler is used for

performance improvement in Java. It is enabled by default. It is

compilation done at execution time rather earlier.

Java has popularized the use of JIT compiler by including it in

JVM.

observing in runtime and optimizing the code, lets call it a dynamic compiler

1. How Java platform is different from other platforms?

Java is a platform independent language. Java compiler converts

Java code in to byte code that can be interpreted by JVM. There are

JVM written for almost all the popular platforms in the world.

Java byte code can run on any supported platform in same way.

Where as other languages require libraries compiled for a specific

platform to run.

1. Why people say that Java is 'write once and run anywhere' language?

You can write Java code on Windows and compile it in Windows

platform. The class and jar files that you get from Windows

platform can run as it is on Unix environment. So it is a truly

platform independent language.

Behind all this portability is Java byte code. Byte code generated by

Java compiler can be interpreted by any JVM. So it becomes much

easier to write programs in Java and expect those to run on any

platform.

Java compiler javac compiles java code and JVM java runs that

code.

1. How does ClassLoader work in Java?

In Java, ClassLoader is a class that is used to load files in JVM.

ClassLoader loads files from their physical file locations e.g.

Filesystem, Network location etc.

There are three main types of ClassLoaders in Java.

1. Bootstrap ClassLoader: This is the first ClassLoader. It

loads classes from rt.jar file.

2. Extension ClassLoader: It loads class files from jre/lib/ext

location.

3. Application ClassLoader: This ClassLoader depends on

CLASSPATH to find the location of class files. If you

specify your jars

8. Do you think ‘main’ used for main method is a keyword in Java?

No, main is just a name of method. There can be multiple methods

with same name main in a class file. It is not a keyword in Java.

9. Can we write main method as public void static instead of public

static void?

No, you cannot write it like this. Any method has to first specify the

modifiers and then the return value. The order of modifiers can

change.

We can write static public void main() instead of public static void

main().

10. In Java, if we do not specify any value for local variables, then what

will be the default value of the local variables?

In Java, local variables must be initialized before they can be used.

Unlike instance and class variables, local variables do not have default values.

This means that if you declare a local variable without initializing it,

you will get a compile-time error.

11. Let say, we run a java class without passing any arguments. What will be the value of String array of arguments in Main method?

As we all know array size should be define after creating and declaring it..

int[] x; //no error

int x[]=new int[]; error

int x[]=new int[1] ; //no error

int x[]=new int[0] //no error ..array size cn be zero

Therefore in string type array args if you are not passing anything and compile the code

It will be set to zero if don't pass any command line argument while running the class fine in cmd

Ex:

Public class Test{ public static void main(String[] args){sopln(args[0])}

}

Javac Test.Java

Java Test

Output zero