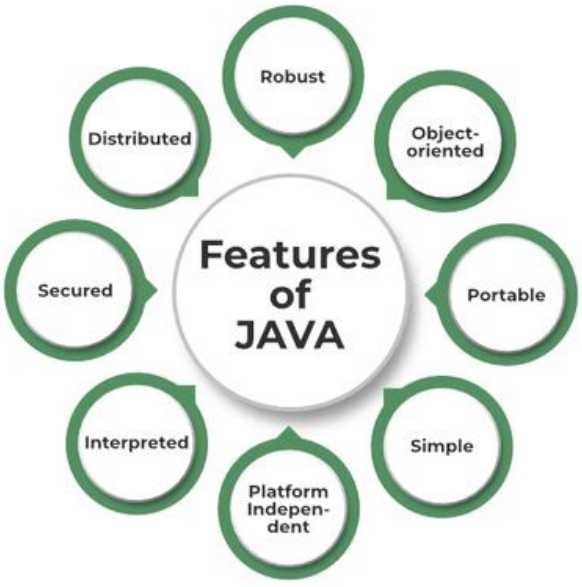
Java Interview Questions and Answers

1. Is Java Platform Independent if then how?

Yes, Java is a Platform Independent language. Unlike many programming languages javac compiler compiles the program to form a bytecode or .class file. This file is independent of the software or hardware running but needs a JVM(Java Virtual Machine) file preinstalled in the operating system for further execution of the bytecode. Although **JVM is platform dependent**, the bytecode can be created on any System and can be executed in any other system despite hardware or software being used which makes Java platform independent.

2. What are the top Java Features?

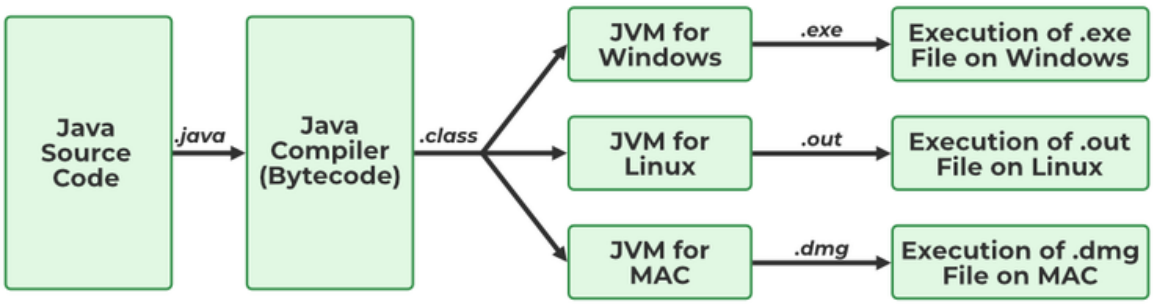
Java is one the most famous and most used language in the real world, there are many features in Java that makes it better than any other language some of them are mentioned below:



- **Simple**: Java is quite simple to understand and the syntax
- **Platform Independent:** Java is platform independent means we can run the same program in any software and hardware and will get the same result.
- **Interpreted**: Java is interpreted as well as a compiler-based language.

- **Robust**: features like Garbage collection, exception handling, etc that make the language robust.
- **Object-Oriented**: Java is an object-oriented language that supports the concepts of class, objects, four pillars of OOPS, etc.
- **Secured**: As we can directly share an application with the user without sharing the actual program makes Java a secure language.
- **High Performance:** faster than other traditional interpreted programming languages.
- **Dynamic**: supports dynamic loading of classes and interfaces. Distributed: feature of Java makes us able to access files by calling the methods from any machine connected.
- **Multithreaded**: deal with multiple tasks at once by defining multiple threads
- **Architecture Neutral**: it is not dependent on the architecture.

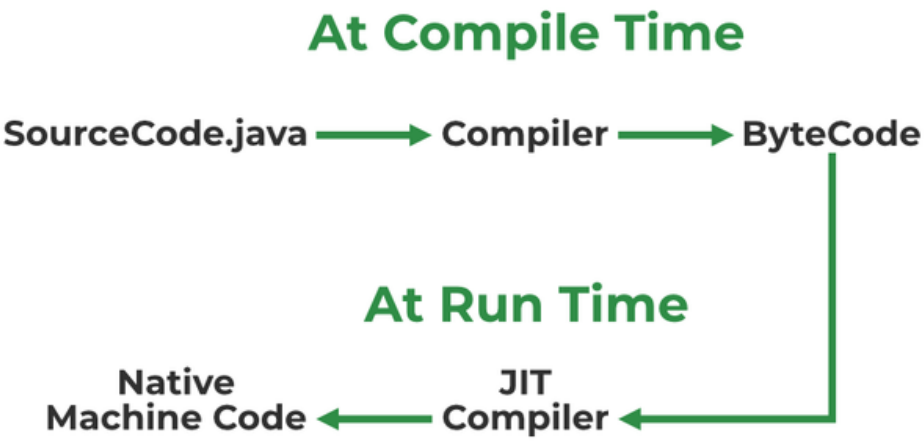
3. What is JVM?



JVM stands for Java Virtual Machine it is a Java interpreter. It is responsible for loading, verifying, and executing the bytecode created in Java. Although it is platform dependent which means the software of JVM is different for different Operating Systems it plays a vital role in making Java platform Independent.

To know more about the topic refer to JVM in Java.

4. What is JIT?

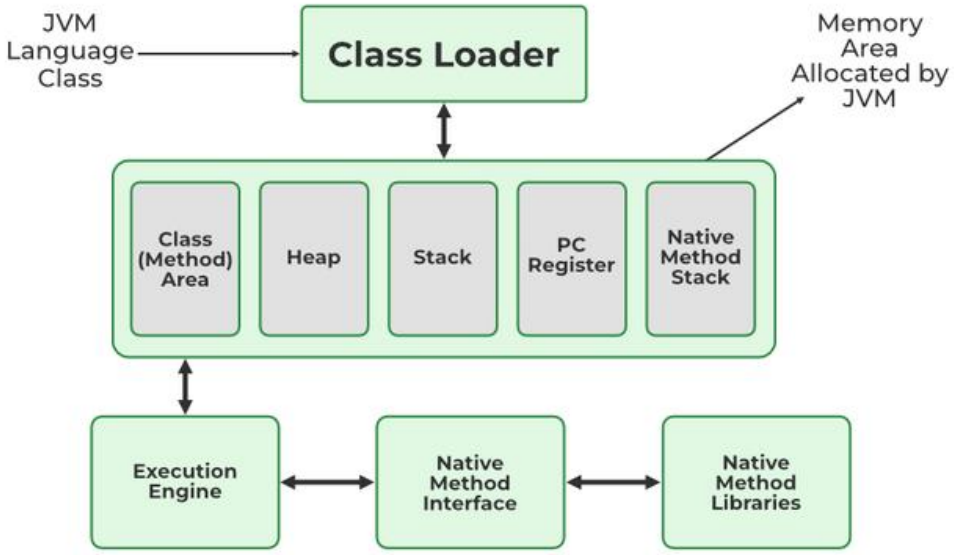


JIT stands for (Just-in-Time) compiler is a part of JRE(Java Runtime Environment), it is used for better performance of the Java applications during run-time. The use of JIT is mentioned in step by step process mentioned below:

1. Source code is compiled with **javac** compiler to form bytecode
2. Bytecode is further passed on to JVM
3. JIT is a part of JVM, JIT is responsible for compiling bytecode into native machine code at run time.
4. The JIT compiler is enabled throughout, while it gets activated when a method is invoked. For a compiled method, the JVM directly calls the compiled code, instead of interpreting it.
5. As JVM calls the compiled code that increases the performance and speed of the execution.

To know more about the topic refer to JIT in Java.

5. What are Memory storages available with JVM?



JVM consists of a few memory storages as mentioned below:

1. Class(Method) Area: stores class-level data of every class such as the runtime constant pool, field, and method data, and the code for methods.
2. Heap: Objects are created or objects are stored. It is used to allocate memory to objects during run time.
3. Stack: stores data and partial results which will be needed while returning value for method and performing dynamic linking
4. Program Counter Register: stores the address of the Java virtual machine instruction currently being executed.
5. Native Method Stack: stores all the native methods used in the application.

To know more about the topic refer to JVM Memory Storages.

6. What is a classloader?

Classloader is the part of JRE(Java Runtime Environment), during the execution of the bytecode or created .class file classloader is responsible for dynamically loading the java classes and interfaces to JVM(Java Virtual Machine). Because of classloaders Java run time system does not need to know about files and file systems.

To know more about the topic refer to ClassLoader in Java.

7. Difference between JVM, JRE, and JDK.

JVM: JVM also known as Java Virtual Machine is a part of JRE. JVM is a type of interpreter responsible for converting bytecode into machine-readable

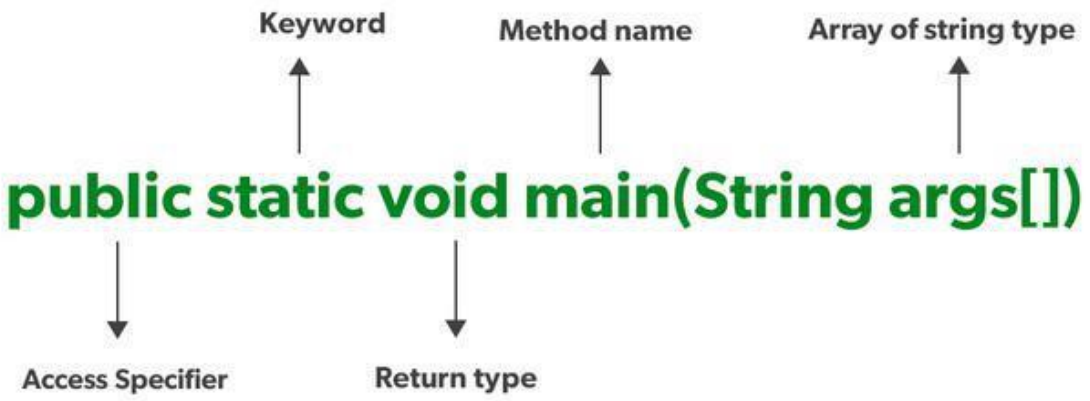
code. JVM itself is platform dependent but it interprets the bytecode which is the platform-independent reason why Java is platform-independent.
JRE: JRE stands for Java Runtime Environment, it is an installation package that provides an environment to run the Java program or application on any machine.
JDK: JDK stands for Java Development Kit which provides the environment to develop and execute Java programs. JDK is a package that includes two things Development Tools to provide an environment to develop your Java programs and, JRE to execute Java programs or applications.
To know more about the topic refer to the Differences between JVM, JRE, and JDK.

8. What are the differences between Java and C++?

Basis	C++	Java
Platform	C++ is Platform Dependent	Java is Platform Independent
Application	C++ is mainly used for System Programming	Java is Mainly used for Application Programming
Hardware	C++ is nearer to hardware	Java is not so interactive with hardware
Global Scope	C++ supports global and namespace scope.	Java doesn't support global scope.
Not Supporting	Functionality supported in Java but not in C++ are: • thread support • documentation comment	Functionality supported in C++ but not in Java are: • goto • Pointers • Call by reference

Basis	C++	Java
	• unsigned right shift(>>>)	• Structures and Unions • Multiple Inheritance • Virtual Functions
OOPS	C++ is an object-oriented language. It is not a single root hierarchy .	Java is also an object-oriented language. It is a single root hierarchy as everything gets derived from a single class (java.lang.Object).
Inheritance Tree	C++ always creates a new inheritance tree.	Java uses a Single inheritance tree as classes in Java are the child of object classes in Java.

9. Explain public static void main(String args[]) in Java.



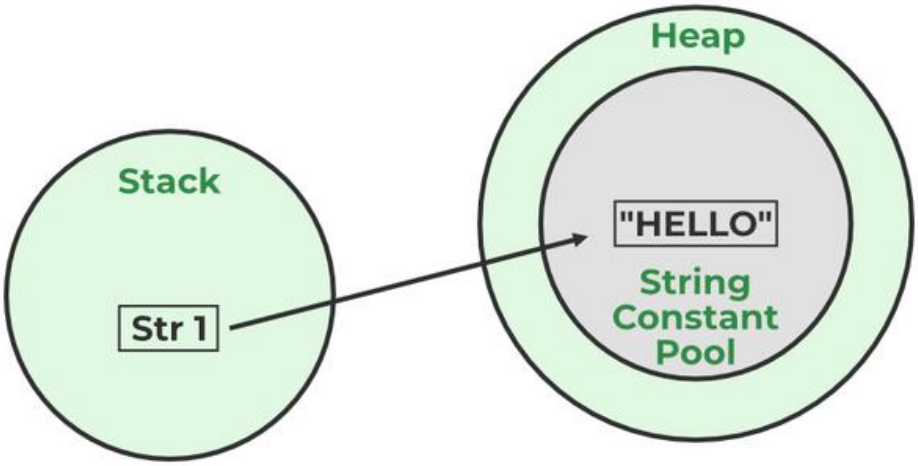
Unlike any other programming language like C, C++, etc. In Java, we declared the main function as a public static void main (String args[]). The meanings of the terms are mentioned below:

1. **public**: the public is the access modifier responsible for mentioning who can access the element or the method and what is the limit. It is responsible for making the main function globally available. It is made public so that JVM can invoke it from outside the class as it is not present in the current class.
2. **static**: static is a keyword used so that we can use the element without initiating the class so to avoid the unnecessary allocation of the memory.
3. **void**: void is a keyword and is used to specify that a method doesn't return anything. As the main function doesn't return anything we use void.
4. **main**: main represents that the function declared is the main function. It helps JVM to identify that the declared function is the main function.
5. **String args[]**: It stores Java command-line arguments and is an array of type java.lang.String class.

10. What is Java String Pool?

A Java String Pool is a place in heap memory where all the strings defined in the program are stored. A separate place in a stack is there where the variable storing the string is stored. Whenever we create a new string object, JVM checks for the presence of the object in the String pool, If String

is available in the pool, the same object reference is shared with the variable, else a new object is created.



Example:

```
String str1="Hello";
// "Hello" will be stored in String Pool
// str1 will be stored in stack memory
```