**INTERVIEW QUESTIONS**

**1. What is Java?**

-Java is a programming language and computing platform first released by **Sun Microsystems** in **1995**. It has evolved from humble beginnings to power a large share of today’s digital world, by providing the reliable platform upon which many services and applications are built. New, innovative products and digital services designed for the future continue to rely on Java, as well.

**-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.

**2. What is a package in Java? List down various advantages of packages.**

-A **java package** is a **group of similar types of classes**, **interfaces** and **sub-packages**.

-Package in java can be categorized in two form, built-in package and user-defined package.

-There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.

Advantage of Java Package

1) Java package is used to **categorize the classes and interfaces** so that they can be easily maintained.

2) Java package provides **access protection.**

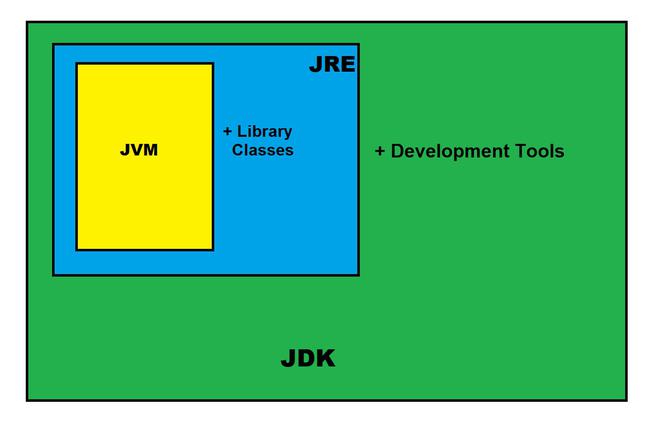
3) Java package **removes naming collision**.

**3. Explain JDK, JRE and JVM? (MODIFY)**

The **Java Development Kit** (JDK) is a cross-platformed software development environment that offers a collection of tools and libraries necessary for developing Java-based software applications and applets. It is a core package used in Java, along with the [**JVM (Java Virtual Machine)**](https://www.geeksforgeeks.org/jvm-works-jvm-architecture/) and the JRE (Java Runtime Environment).

Beginners often get confused with JRE and JDK, if you are only interested in running Java programs on your machine then you can easily do it using Java Runtime Environment. However, if you would like to develop a Java-based software application then along with JRE you may need some additional necessary tools, which is called JDK.

**JDK=JRE+Development Tools**



*JAVA Development Kit (JDK)*

#### 

#### The Java Development Kit  is an implementation of  one of the Java Platform:

* [Standard Edition](https://en.wikipedia.org/wiki/Java_Platform,_Standard_Edition) (Java SE),
* [Java Enterprise Edition (Java EE)](https://en.wikipedia.org/wiki/Jakarta_EE),
* [Micro Edition (Java ME),](https://en.wikipedia.org/wiki/Java_Platform,_Micro_Edition)

### Contents of JDK

The JDK has a private Java Virtual Machine (JVM) and a few other resources necessary for the development of a Java Application.

**JDK contains:**

* Java Runtime Environment (JRE),
* An interpreter/loader (Java),
* A compiler (javac),
* An archiver (jar) and many more.

The Java Runtime Environment in JDK is usually called Private Runtime because it is separated from the regular JRE and has extra content. The Private Runtime in JDK contains a JVM and all the class libraries present in the production environment, as well as additional libraries useful to developers, e.g, internationalization libraries and the IDL libraries.

### **Most Popular JDKs:**

* **Oracle JDK:** the most popular JDK and the main distributor of Java11,
* **OpenJDK:** Ready for use: JDK 15, JDK 14, and JMC,
* **Azul Systems Zing:** efficient and low latency JDK for Linux os,
* **Azul Systems:** based Zulu brand for Linux, Windows, Mac OS X,
* **IBM J9 JDK:** for AIX, Linux, Windows, and many other OS,
* **Amazon Corretto:** the newest option with the no-cost build of OpenJDK and long-term support.

### Set-Up:

Setting up JDK in your development environment is super easy, just follow the below simple steps.

**Installation of JDK**

* Go to this Oracle’s official Download Page through this [link](https://www.oracle.com/in/java/technologies/)
* Select the latest JDK version and click Download and add it to your classpath.
* Just check the JDK software is installed or not on your computer at the correct location, for example, at C:\Program Files\Java\jdk11.0.9.

**Set JAVA\_HOME for Windows:**

* Right-click My Computer and select Properties.
* Go to the Advanced tab and select Environment Variables, and then edit JAVA\_HOME to point to the exact location where your JDK software is stored, for example, C:\Program Files\Java\jdk11.0.9 is the default location in windows.

Java maintains backward compatibility, so don’t worry just download the latest release and you will get all the old and many new features. After Installing the JDK and JRE adds the java command to your command line. You can verify this through the command prompt by the **java -version** command. In some cases, you need to restart your system after installing the JDK.

You can use the JDK compiler to convert your Java text file into an executable program. Your Java text segment is converted into **bytecode** after compilation which carries the **.class** extension.

First, create a Java text file and save it using a name. Here we are saving the file as Hello.java.

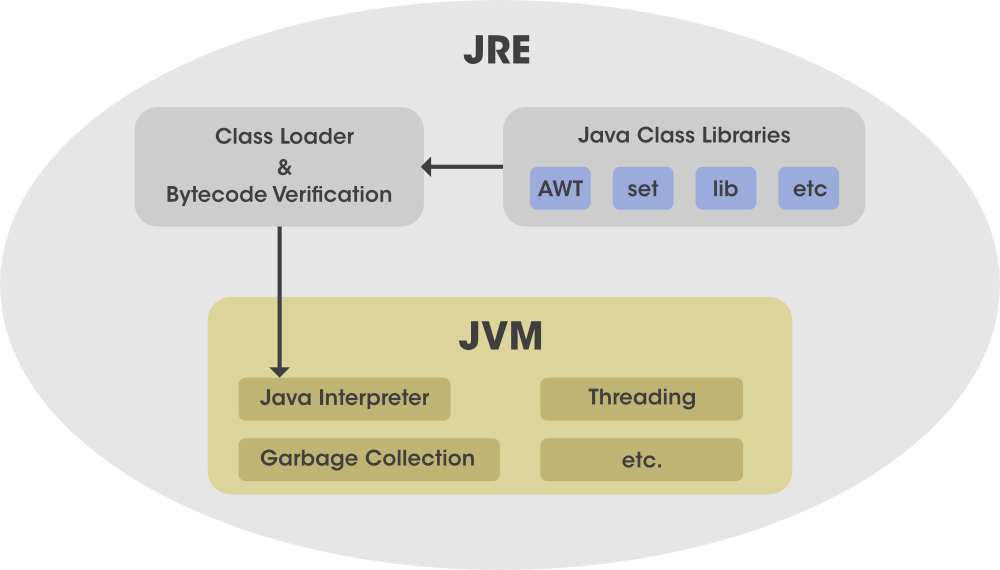
# After that just simply use the**javac** command, which is used for the compilation purpose in Java. Please don’t forget to provide the full path of your java text file to the command line else you will get an error as “The system cannot find the path specified”,

# JRE in Java

**Java Runtime Environment** (JRE) is an open-access software distribution that has a Java class library, specific tools, and a separate JVM. JRE is one of the interrelated components in the Java Development Kit (JDK). It is the most common environment available on devices for running Java programs. Java source code is compiled and converted to Java bytecode. If you want to run this bytecode on any platform, you need JRE. The JRE loads classes check memory access and get system resources. JRE acts as a software layer on top of the operating system.

### Components of JRE

* Integration libraries include Java Database Connectivity (JDBC)
* Java Naming, Interface Definition Language (IDL)
* Directory Interface (JNDI)
* Remote Method Invocation Over Internet Inter-Orb Protocol (RMI-IIOP)
* Remote Method Invocation (RMI)
* Scripting



Java Virtual Machine (JVM) consists of Java HotSpot Client and Server Virtual Machine.

* User interface libraries include Swing, Java 2D, Abstract Window Toolkit (AWT), Accessibility, Image I/O, Print Service, Sound, drag, and drop (DnD), and input methods.
* Lang and util base libraries, which include lang and util, zip, Collections, Concurrency Utilities, management, Java Archive (JAR), instrument, reflection,  versioning, Preferences API, Ref Objects,  Logging,  and Regular Expressions.
* Other base libraries, including Java Management Extensions (JMX), Java Native Interface (JNI), Math, Networking, international support, input/output (I/O), Beans, Java Override Mechanism, Security, Serialization, extension mechanism, and Java for XML Processing (XML JAXP).
* Deployment technologies such as Java Web Start, deployment, and Java plug-in.

### **Working of JRE**

Java Development Kit (JDK) and Java Runtime Environment (JRE)  both interact with each other to create a sustainable runtime environment that enables Java-based applications to run seamlessly on any operating system. The JRE runtime architecture consists of the following elements as listed:

1. [ClassLoader](https://www.geeksforgeeks.org/classloader-in-java/)
2. ByteCode verifier
3. [Interpreter](https://www.geeksforgeeks.org/compiler-vs-interpreter-2/)

Now let us brief about them as follows:

* **ClassLoader**: Java ClassLoader dynamically loads all the classes necessary to run a Java program. Because classes are only loaded into memory whenever they are needed, the JRE uses ClassLoader will automate this process when needed.
* **Bytecode Verifier:**The bytecode checker ensures the format and precision of Java code before passing it to the interpreter. If the code violates system integrity or access rights, the class is considered corrupt and will not load.
* **Interpreter:**After loading the byte code successfully, the Java interpreter creates an object of the Java virtual machine that allows the Java program to run natively on the underlying machine.

# JVM (Java Virtual Machine) Architecture

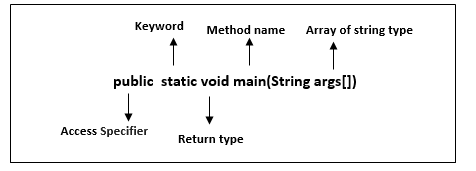
JVM (Java Virtual Machine) is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed.

JVMs are available for many hardware and software platforms (i.e. JVM is platform dependent).

1. **A specification** where working of Java Virtual Machine is specified. But implementation provider is independent to choose the algorithm. Its implementation has been provided by Oracle and other companies.
2. **An implementation** Its implementation is known as JRE (Java Runtime Environment).
3. **Runtime Instance** Whenever you write java command on the command prompt to run the java class, an instance of JVM is created.

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

The main() is the starting point for JVM to start execution of a Java program. Without the main() method, JVM will not execute the program. The syntax of the main() method is:



**public:** It is an access specifier. We should use a public keyword before the main() method so that JVM can identify the execution point of the program. If we use private, protected, and default before the main() method, it will not be visible to JVM.

**static:** You can make a method static by using the keyword static. We should call the main() method without creating an object. Static methods are the method which

invokes without creating the objects, so we do not need any object to call the main() method.

**void:** In Java, every method has the return type. Void keyword acknowledges the compiler that main() method does not return any value.

**main():** It is a default signature which is predefined in the JVM. It is called by JVM to execute a program line by line and end the execution after completion of this method. We can also overload the main() method.

**String args[]:** The main() method also accepts some data from the user. It accepts a group of strings, which is called a string array. It is used to hold the command line arguments in the form of string values.

## What happens if the main() method is written without String args[]?

The program will compile, but not run, because JVM will not recognize the main() method. Remember JVM always looks for the main() method with a string type array as a parameter.

**5. What are the differences between C++ and Java?**

There are many differences and similarities between the [C++ programming](https://www.javatpoint.com/cpp-tutorial) language and [Java](https://www.javatpoint.com/java-tutorial). A list of top differences between C++ and Java are given below:

|  |  |  |
| --- | --- | --- |
| **Comparison Index** | **C++** | **Java** |
| **Platform-independent** | C++ is platform-dependent. | Java is platform-independent. |
| **Mainly used for** | C++ is mainly used for system programming. | Java is mainly used for application programming. It is widely used in Windows-based, web-based, enterprise, and mobile applications. |
| **Design Goal** | C++ was designed for systems and applications programming. It was an extension of the [C programming language](https://www.javatpoint.com/c-programming-language-tutorial). | Java was designed and created as an interpreter for printing systems but later extended as a support network computing. It was designed to be easy to use and accessible to a broader audience. |
| **Goto** | C++ supports the [goto](https://www.javatpoint.com/cpp-goto-statement) statement. | Java doesn't support the goto statement. |
| **Multiple inheritance** | C++ supports multiple inheritance. | Java doesn't support multiple inheritance through class. It can be achieved by using [interfaces in java](https://www.javatpoint.com/interface-in-java). |
| **Operator Overloading** | C++ supports [operator overloading](https://www.javatpoint.com/cpp-overloading). | Java doesn't support operator overloading. |
| **Pointers** | C++ supports [pointers](https://www.javatpoint.com/cpp-pointers). You can write a pointer program in C++. | Java supports pointer internally. However, you can't write the pointer program in java. It means java has restricted pointer support in java. |
| **Compiler and Interpreter** | C++ uses compiler only. C++ is compiled and run using the compiler which converts source code into machine code so, C++ is platform dependent. | Java uses both compiler and interpreter. Java source code is converted into bytecode at compilation time. The interpreter executes this bytecode at runtime and produces output. Java is interpreted that is why it is platform-independent. |
| **Call by Value and Call by reference** | C++ supports both call by value and call by reference. | Java supports call by value only. There is no call by reference in java. |
| **Structure and Union** | C++ supports structures and unions. | Java doesn't support structures and unions. |
| **Thread Support** | C++ doesn't have built-in support for threads. It relies on third-party libraries for thread support. | Java has built-in [thread](https://www.javatpoint.com/multithreading-in-java) support. |
| **Documentation comment** | C++ doesn't support documentation comments. | Java supports documentation comment (/\*\* ... \*/) to create documentation for java source code. |
| **Virtual Keyword** | C++ supports virtual keyword so that we can decide whether or not to override a function. | Java has no virtual keyword. We can override all non-static methods by default. In other words, non-static methods are virtual by default. |
| **unsigned right shift >>>** | C++ doesn't support >>> operator. | Java supports unsigned right shift >>> operator that fills zero at the top for the negative numbers. For positive numbers, it works same like >> operator. |
| **Inheritance Tree** | C++ always creates a new inheritance tree. | Java always uses a single inheritance tree because all classes are the child of the Object class in Java. The Object class is the root of the [inheritance](https://www.javatpoint.com/inheritance-in-java) tree in java. |
| **Hardware** | C++ is nearer to hardware. | Java is not so interactive with hardware. |
| **Object-oriented** | C++ is an object-oriented language. However, in the C language, a single root hierarchy is not possible. | Java is also an [object-oriented](https://www.javatpoint.com/java-oops-concepts) language. However, everything (except fundamental types) is an object in Java. It is a single root hierarchy as everything gets derived from java.lang.Object. |

**Note**

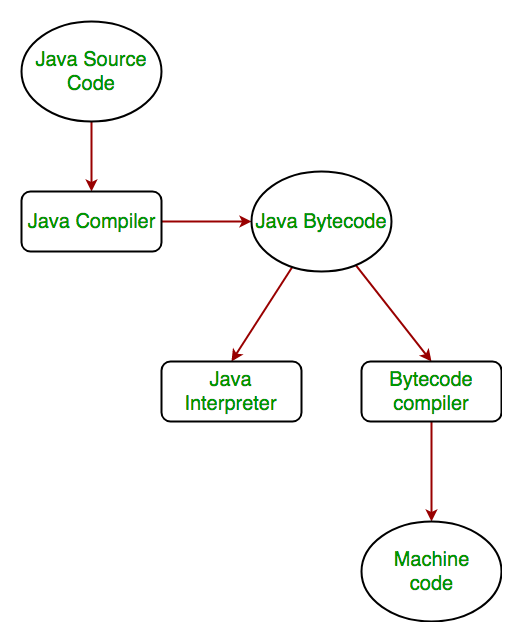
* Java doesn't support default arguments like C++.
* Java does not support header files like C++. Java uses the import keyword to include different classes and methods.

**6. Why Java is platform independent?**

The meaning of platform-independent is that the **java compiled code(byte code) can run on all operating systems.**  
A program is written in a language that is a human-readable language. It may contain words, phrases, etc which the machine does not understand. For the source code to be understood by the machine, it needs to be in a language understood by machines, typically a machine-level language. So, here comes the role of a compiler. The compiler converts the high-level language (human language) into a format understood by the machines. Therefore, a compiler is a program that translates the source code for another program from a programming language into executable code.  
This executable code may be a sequence of machine instructions that can be executed by the CPU directly, or it may be an intermediate representation that is interpreted by a virtual machine. This intermediate representation in Java is the **Java Byte Code.**

**Step by step Execution of Java Program:**

* Whenever, a program is written in JAVA, the javac compiles it.
* The result of the JAVA compiler is the **.class file or the bytecode** and not the machine native code (unlike C compiler).
* The bytecode generated is a non-executable code and needs an interpreter to execute on a machine. This interpreter is the JVM and thus the Bytecode is executed by the JVM.
* And finally program runs to give the desired output.

.

In case of C or C++ (language that are not platform independent), the compiler generates an .exe file which is OS dependent. When we try to run this .exe file on another OS it does not run, since it is OS dependent and hence is not compatible with the other OS. In Java, the main point here is that the JVM depends on the operating system – so if you are running Mac OS X you will have a different JVM than if you are running Windows or some other operating system. This fact can be verified by trying to download the JVM for your particular machine – when trying to download it, you will be given a list of JVMs corresponding to different operating systems, and you will obviously pick whichever JVM is targeted for the operating system that you are running. So we can conclude that JVM is platform-dependent and it is the reason why Java is able to become “Platform Independent”.  
**Important Points:**

* In the case of Java, **it is the magic of Bytecode that makes it platform independent**.
* This adds to an important feature in the JAVA language termed as **portability**. Every system has its own JVM which gets installed automatically when the jdk software is installed. For every operating system separate JVM is available which is capable to read the .class file or byte code.
* An important point to be noted is that while **JAVA is platform-independent language, the JVM is platform-dependent.** Different JVM is designed for different OS and byte code is able to run on different OS.

**8. What are wrapper classes in Java?**

A Wrapper class in Java is a class **whose object wraps or contains primitive data types**. When we create an object to a wrapper class, it contains a field and in this field, we can store primitive data types. In other words, we can wrap a primitive value into a wrapper class object. Let’s check on the wrapper classes in java with examples:

## ****Need of Wrapper Classes****

1. They convert primitive data types into objects. Objects are needed if we wish to modify the arguments passed into a method (because primitive types are passed by value).
2. The classes in java.util package handles only objects and hence wrapper classes help in this case also.
3. Data structures in the Collection framework, such as [ArrayList](https://www.geeksforgeeks.org/arraylist-in-java/) and [Vector](https://www.geeksforgeeks.org/vector-vs-arraylist-java/), store only objects (reference types) and not primitive types.

**9. Why pointers are not used in Java?**

Java do not use pointers because using pointer the memory area can be directly accessed, which is a security issue. pointers need some memory spaces at the runtime. **to reduce the usage of memory spaces** java does not support pointers. and also pointers take more time at the run time

**10. List some features of Java?**

Ans: The primary objective of [Java programming](https://www.javatpoint.com/java-tutorial) language creation was to make it portable, simple and secure programming language. Apart from this, there are also some excellent features which play an important role in the popularity of this language. The features of Java are also known as Java buzzwords.

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

1. [Simple](https://www.javatpoint.com/features-of-java#Simple)
2. [Object-Oriented](https://www.javatpoint.com/features-of-java#Object-Oriented)
3. [Portable](https://www.javatpoint.com/features-of-java#Portable)
4. [Platform independent](https://www.javatpoint.com/features-of-java#Platform-independent)
5. [Secured](https://www.javatpoint.com/features-of-java#Secured)
6. [Robust](https://www.javatpoint.com/features-of-java#Robust)
7. [Architecture neutral](https://www.javatpoint.com/features-of-java#Architecture-neutral)
8. [Interpreted](https://www.javatpoint.com/features-of-java#Interpreted)
9. [High Performance](https://www.javatpoint.com/features-of-java#High-Performance)
10. [Multithreaded](https://www.javatpoint.com/features-of-java#Multithreaded)
11. [Distributed](https://www.javatpoint.com/features-of-java#Distributed)
12. [Dynamic](https://www.javatpoint.com/features-of-java#Dynamic)

Simple

Java is very easy to learn, and its syntax is simple, clean and easy to understand. According to Sun Microsystem, Java language is a simple programming language because:

* Java syntax is based on C++ (so easier for programmers to learn it after C++).
* Java has removed many complicated and rarely-used features, for example, explicit pointers, operator overloading, etc.
* There is no need to remove unreferenced objects because there is an Automatic Garbage Collection in Java.

### **Object-oriented**

Java is an [object-oriented](https://www.javatpoint.com/java-oops-concepts) programming language. Everything in Java is an object. Object-oriented means we organize our software as a combination of different types of objects that incorporate both data and behaviour. Object-oriented programming (OOPs) is a methodology that simplifies software development and maintenance by providing some rules.

### **Platform Independent**



Java is platform independent because it is different from other languages like [C](https://www.javatpoint.com/c-programming-language-tutorial), [C++](https://www.javatpoint.com/cpp-tutorial), etc. which are compiled into platform specific machines while Java is a write once, run anywhere language. A platform is the hardware or software environment in which a program runs.

There are two types of platforms software-based and hardware-based. Java provides a software-based platform.

The Java platform differs from most other platforms in the sense that it is a software-based platform that runs on top of other hardware-based platforms. It has two components:

1. Runtime Environment
2. API(Application Programming Interface)

Java code can be executed on multiple platforms, for example, Windows, Linux, Sun Solaris, Mac/OS, etc. Java code is compiled by the compiler and converted into bytecode. This bytecode is a platform-independent code because it can be run on multiple platforms, i.e., Write Once and Run Anywhere (WORA).

Secured

Java is best known for its security. With Java, we can develop virus-free systems. Java is secured because:

* **No explicit pointer**
* **Java Programs run inside a virtual machine sandbox**

Robust

The English meaning of Robust is **strong.** Java is robust because:

* It uses strong memory management.
* There is a lack of pointers that avoids security problems.
* Java provides automatic garbage collection which runs on the Java Virtual Machine to get rid of objects which are not being used by a Java application anymore.
* There are exception handling and the type checking mechanism in Java. All these points make Java robust.

### **Architecture-neutral**

Java is architecture neutral because **there are no implementation dependent features**, for example, the size of primitive types is fixed.

In C programming, int data type occupies 2 bytes of memory for 32-bit architecture and 4 bytes of memory for 64-bit architecture. However, it occupies 4 bytes of memory for both 32 and 64-bit architectures in Java.

### **Portable**

Java is portable because it facilitates you to carry the Java bytecode to any platform. It doesn't require any implementation.

### **High-performance**

Java is faster than other traditional interpreted programming languages because Java bytecode is "close" to native code. It is still a little bit slower than a compiled language (e.g., C++). Java is an interpreted language that is why it is slower than compiled languages, e.g., C, C++, etc.

### **Distributed**

Java is distributed because it facilitates users to create distributed applications in Java. RMI and EJB are used for creating distributed applications. This feature of Java makes us able to access files by calling the methods from any machine on the internet.

### **Multi-threaded**

A thread is like a separate program, executing concurrently. We can write Java programs that deal with many tasks at once by defining multiple threads. The main advantage of multi-threading is that it doesn't occupy memory for each thread. It shares a common memory area. Threads are important for multi-media, Web applications, etc.

### **Dynamic**

Java is a dynamic language. It supports the dynamic loading of classes. It means classes are loaded on demand. It also supports functions from its native languages, i.e., C and C++.

**11. Why is Java Architectural Neutral?**

Java was designed with architecture neutrality in mind, and a lot of the language features were added to ensure this feature. One of the main reasons why Java is considered to be architecture neutral is because **it doesn't have any architecture-specific instructions or features** that store or refer to physical addresses, and instead abstracts the underlying architecture and hardware from the code.

When running a program, the Java process first translates the code into a\* Bytecode\*, which is a set of instructions that are understood by the Java Virtual Machine (JVM). The JVM is then responsible for converting this code into instructions that the underlying computer architecture can understand and execute. This means that regardless of the underlying system architecture, the application will still run the same.

Another contributing factor to the architecture neutrality of Java is the fact that it does not depend on a particular operating system. Java applications can be run on Windows, Macs and Linux systems with no changes to the code. This ensures that developers can write code that is platform-independent, thus ensuring the application can run on multiple prevailing operating systems.

**12. How Java enabled High Performance?**

Java enabled High performance by introducing **JIT- Just In Time compiler** , JIT helps the compiler to compile the code On demand basis i.e which ever method is called only that method block will get compiled making compilation fast n time-efficient. This makes the java delivering high performance.

**13. Why Java is considered dynamic?**

Java is considered to be more dynamic than C or C++ since it is **designed to adapt to an evolving environment.** Java programs can carry an extensive amount of run-time information that can be used to verify and resolve accesses to objects at run-time.

**14. What is Java Virtual Machine and how it is considered in context of Java’s platform independent feature?**

Java is called platform independent because of Java Virtual Machine. As different computers with the different operating system have their JVM, when we submit a . class file to any operating system, JVM interprets the bytecode into machine level language.

**15. List two Java IDE’s? ( MODIFY ANS )**

Java IDE (**Integrated Development Environment**) is a software application that enables users to **write** and **debug** Java programs more easily. Most IDEs have features such as syntax highlighting and code completion that helps users to code more easily. Usually, Java IDEs include a **code editor, a compiler, a debugger**, and **an interpreter** that the developer may access via a single graphical user interface. Java IDEs also provide language-specific elements such as **Maven, Ant building tools, Junit**, and **TestNG for testing**.

### It is a Java-based open-source platform that enables us to create highly customized IDEs from [Eclipse](https://www.javatpoint.com/javafx-how-to-install-eclipse) member's plug-in components. This platform is also suitable for beginners to create user-friendly and more sophisticated applications. It contains many plugins that enable developers to develop and test code written in different languages.

### **NetBeans**

NetBeans is a Java-based IDE and basic application platform framework. Besides Java, [JavaScript](https://www.javatpoint.com/javascript-tutorial) and [JavaFX](https://www.javatpoint.com/javafx-tutorial), NetBeans supports [PHP](https://www.javatpoint.com/php-tutorial), [C](https://www.javatpoint.com/c-programming-language-tutorial)/[C++](https://www.javatpoint.com/cpp-tutorial), [Groovy](https://www.javatpoint.com/groovy), and [HTML5](https://www.javatpoint.com/html5-tutorial) languages. It is a free and open-source Java IDE that enables Java developers to develop various applications using different module sets. Some of the following features of [NetBeans](https://www.javatpoint.com/how-to-install-netbeans-on-centos) are as follows:

1. NetBeans is available for various operating systems, such as Linux, MacOS, Windows, Solaris, etc.
2. Although NetBeans is mainly a Java IDE, it has extensions to operate in many other programming languages, such as C, PHP, C++, JavaScript, HTML5, etc.
3. NetBeans may be used on different systems such as MacOS, Windows, Solaris and Linux.

**16. Why Java is called as “Platform” ? (MODIFY)**

Most platforms can be described as a combination of the operating system and underlying hardware. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware-based platforms. The Java platform has two components: The Java Virtual Machine.

**17. Is Java Pure-Object oriented Language ?**

Despite the fact that it supports the four pillars of OOPs, **Java is not 100% object oriented** due to the following reasons: **Existence of Primitive data types: It's data that's not an object and has no properties or any methods**

**18. Which version of java have u learned? Name some of the new features added to it.**

There are many new features that have been added in java. There are major enhancement made in Java5, Java6, Java7 and Java8 like **auto-boxing**, **generics**, **var-args**, **java annotations**, **enum**, **premain method** , **lambda expressions**, **functional interface**, **method references** etc. INCOMPLETE ADD MORE

**19. What gives Java its 'write once and run anywhere' nature?**

**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.

**20. Difference between path and classpath.**

| **S. No.** | **PATH** | **CLASSPATH** |
| --- | --- | --- |
| 1. | An environment variable is used by the operating system to find **the executable files.** | An environment variable is used by the Java compiler to find the **path of classes.** |
| 2. | PATH setting up an environment for the operating system. Operating System will look in this PATH for executables. | Classpath setting up the environment for Java. Java will use to **find compiled classes**. |
| 3. | Refers to the **operating system**. | Refers to the **Developing Environment.** |
| 4. | In path variable, we must place .\bin folder **path** | In classpath, we must place **.\lib\jar file** or directory path in which .java file is available. |
| 5. | PATH is used by **CMD prompt** to **find binary files.** | CLASSPATH is used by the compiler and **JVM** to **find library files.** |

**21. What is the signature of main function in java ?**

**22. What is the difference between JDK and JRE?**

| **JDK** | **JRE** |
| --- | --- |
| JDK(Java Development Kit) is used to **develop Java applications**. JDK also contains numerous development tools like compilers, debuggers, etc. | JRE(Java Runtime Environment) is the **implementation of JVM**(Java Virtual Machine) and it is specially designed to **execute Java programs.** |
| It is mainly used for the **execution of code** and its main functionality is development. | It is mainly used for **creating an environment for code execution.** |
| It is platform-dependent. | It is also platform-dependent like JDK. |
| Since JDK is responsible for the development purpose, therefore it **contains tools** which are required for development and **debugging purpose**. | On the other hand, JRE is not responsible for development purposes so it doesn’t contain such tools as the compiler, debugger, etc. Instead, it **contains class libraries** and **supporting files** required for the purpose of execution of the program. |
| **JDK = JRE + other development tools.** | **JRE = JVM + other class libraries.** |

**23. What is JVM ? What it does? (MODIFY)**

A **Java virtual machine** (**JVM**) is a [virtual machine](https://en.wikipedia.org/wiki/Virtual_machine) that enables a computer to run [Java](https://en.wikipedia.org/wiki/Java_(software_platform)) programs as well as programs written in [other languages](https://en.wikipedia.org/wiki/List_of_JVM_languages) that are also compiled to [Java bytecode](https://en.wikipedia.org/wiki/Java_bytecode). The JVM is detailed by a [specification](https://en.wikipedia.org/wiki/Specification_(technical_standard)) that formally describes what is required in a JVM implementation. Having a specification ensures interoperability of Java programs across different implementations so that program authors using the [Java Development Kit](https://en.wikipedia.org/wiki/Java_Development_Kit) (JDK) need not worry about idiosyncrasies of the underlying hardware platform.

The JVM [reference implementation](https://en.wikipedia.org/wiki/Reference_implementation) is developed by the [OpenJDK](https://en.wikipedia.org/wiki/OpenJDK" \o "OpenJDK) project as [open source](https://en.wikipedia.org/wiki/Open-source_software) code and includes a [JIT compiler](https://en.wikipedia.org/wiki/JIT_compiler) called [HotSpot](https://en.wikipedia.org/wiki/HotSpot_(virtual_machine)" \o "HotSpot (virtual machine)).

Java Virtual Machine, or JVM, loads, **verifies and executes Java bytecode**. It is known as the **interpreter or the core of Java programming language** because it executes Java programming.

**24. Why JVM is called as “virtual machine”?**

The Java Virtual Machine, or JVM, is an abstract computer that **runs compiled Java programs**. The JVM is "virtual" because it is generally implemented in software on top of a "real" hardware platform and operating system. All Java programs are compiled for the JVM.

**26. What is the difference between Java compiler ( javac ) and JIT ?**

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.**

**27. Is Empty .java file name a valid source file name?**

**Yes**, Java allows to save our java file by . java only, we need to compile it by javac .

**28. Is JRE different for different Platforms ?**

JRE = JVM + set of libraries.

JRE is also platform dependent. That means we have different JRE versions for different platforms.

**29. Difference between C++ and java in terms of object creation.**

I would answer: C++ allows creating an object everywhere: on the heap, stack, member. **Java forces you allocate objects on the heap, always.**

**30. Who invokes main() function ?**

In 'C', the "main" function is called by the operating system when the user runs the program and it is treated the same way as every function, it has a return type.

**31. What is .class file known as ?**

A **Java class file** is a file containing **Java bytecode** and having **.class extension** that can be executed by [JVM](https://www.geeksforgeeks.org/jvm-works-jvm-architecture/). A Java class file is created by a Java compiler from .java files as a result of successful compilation. As we know that a single Java programming language source file (or we can say .java file) may contain one class or more than one class. So if a .java file has more than one class then each class will compile into a separate class files.

**33. Can we define more than one public class in a java source code ? what is the rule of public class and file name . ?**

No, while defining multiple classes in a single Java file you need to make sure that **only one class among them is public**. If you have more than one public classes a single file a compile-time error will be generated.

**34. What is JIT compiler? (MODIFY)**

In order to improve performance, JIT compilers interact with the [Java Virtual Machine (JVM)](https://www.geeksforgeeks.org/jvm-works-jvm-architecture/) at run time and compile suitable bytecode sequences into native machine code. While using a JIT compiler, the hardware is able to execute the native code, as compared to having the JVM interpret the same sequence of bytecode repeatedly and incurring overhead for the translation process. This subsequently leads to performance gains in the execution speed, unless the compiled methods are executed less frequently.

The JIT compiler is able to perform certain simple optimizations while compiling a series of bytecode to native machine language. Some of these optimizations performed by JIT compilers are data analysis, reduction of memory accesses by register allocation, translation from stack operations to register operations, elimination of common sub-expressions, etc. The greater the degree of optimization done, the more time a JIT compiler spends in the execution stage. Therefore it cannot afford to do all the optimizations that a static compiler is capable of, because of the extra overhead added to the execution time and moreover its view of the program is also restricted.

**35. How many types of memory areas are allocated by JVM? (DETAIL ANS IN Q.NO 46)**

### -Types of Memory Areas Allocated By the JVM:

-All these functions take different forms of memory structure. The **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

**37. What are the various access specifiers in Java?**

The four access modifiers in Java are **public, protected, default, and private**.

**38. What is the rule for local member in java.**

Local variables **cannot use any of the access level** since their scope is only inside the method. **Final is the Only Non Access Modifier that can be applied to a local variable**. Local variables are not assigned a default value, hence they need to be initialized.

**39. What is native code?**

Native code is a form of [**computer programming**](https://www.easytechjunkie.com/what-is-computer-programming.htm)**code** that is configured to function with the use of a specified processor. The exact structure of the code is set up to respond to the instructions that are issued by the processor. All types of software function with native code and are written to function at optimal efficiency with a selected type of processor or with processors that are manufactured to mirror the configuration of the specified processor.

**40. Why there is no sizeof operator in java ?**

-Sizeof() is not needed because **Java basic types' sizes are fixed.** Yes, a Java int is 32 bits in all JVMs and on all platforms, but this is only a language specification requirement for the programmer-perceivable width of this data type.

**42. U have reference type as a member of class. What is the default value it gets?**

The default value of a reference type is **null** . It means that if a reference type is a static class member or an instance field and not assigned an initial value explicitly, it will be initialized automatically and assigned the value of null .

**43. What is the job done by classloader ?**

The **Java ClassLoader** is a part of the [**Java Runtime Environment**](https://www.geeksforgeeks.org/differences-jdk-jre-jvm/) that dynamically loads Java classes into the [**Java Virtual Machine**](https://www.geeksforgeeks.org/jvm-works-jvm-architecture/).The Java run time system does not need to know about files and file systems because of classloaders. [Java classes](https://www.geeksforgeeks.org/classes-objects-java/) aren’t loaded into memory all at once, but when required by an application. At this point, the **Java ClassLoader** is called by the **JRE** and these ClassLoaders load classes into memory dynamically.

ADD MORE

**44. Explain the hierarchy of classloaders in java.**

-ClassLoader is hierarchical in loading a class into memory. Whenever a request is raised to load a class, it delegates it to the parent classloader. This is how uniqueness is maintained in the runtime environment. If the parent class loader doesn't find the class then the class loader itself tries to load the class.

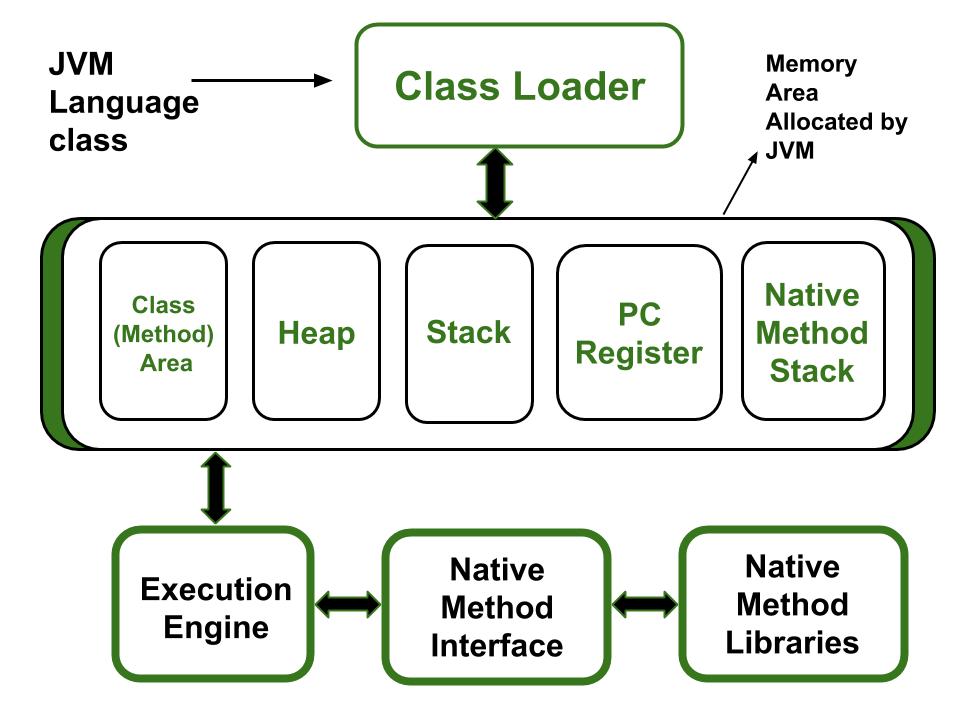
**45. What is the role played by Bytecode Verifier ?**

-The **bytecode verifier acts as a sort of gatekeeper**: it ensures that code passed to the Java interpreter is in a fit state to be executed and can run without fear of breaking the Java interpreter.

**46. What are the memory areas allocated by JVM ?**

### Types of Memory Areas Allocated By the JVM:

All these functions take different forms of memory structure. The **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

Let’s see about them in brief:

#### 1. Class (Method) Area

The class method area is the **memory block that stores the class code**, **variable code**(static variable, runtime constant), **method code**, and the **constructor** of a Java program. (Here method means the function which is written inside the class). It stores class-level data of every class such as the runtime constant pool, field and method data, the code for methods.

#### 2. Heap

The Heap area is the memory block where **objects are created or objects are stored**. Heap memory allocates memory for class interfaces and arrays (an array is an object). It is used to allocate memory to objects at run time

#### 3. Stack

Each thread has a private JVM stack, created at the same time as the thread. It is used to **store data and partial results** which will be needed while returning value for method and performing dynamic linking.

Java Stack stores frames and a new frame is created each time at every invocation of the method. A frame is destroyed when its method invocation completes

#### 4. Program Counter Register:

Each JVM thread that carries out the task of a specific method has a program counter register associated with it. The non-native method has a PC that stores the address of the available JVM instruction whereas, in a native method, the value of the program counter is undefined. PC register is capable of storing the return address or a native pointer on some specific platform.

#### 5. Native method Stacks:

Also called **C stacks**, native method stacks are not written in Java language. This memory is allocated for each thread when it’s created and it can be of a fixed or dynamic nature.

**47. What kinds of programs u can develop using Java**

**-Top 10 Applications of Java** in the real world:

* *Desktop GUI Applications*
* *Mobile Applications*
* *Artificial intelligence*
* *Web applications*
* *Big Data technology*
* *Gaming applications*
* *Business applications*
* *Embedded systems*
* *Cloud applications*
* *Scientific applications*

**48. When parseInt() method can be used?**

**-This method is used to get the primitive data type of a certain String.**

**49. What is finalized() method ?**

The Finalize method is used to **perform clean-up operations** on unmanaged resources held by the current object before the object is destroyed. The method is protected and therefore is accessible only through this class or through a derived class.

**50. Difference between C++ pointer and Java reference.**

-In C/C++, a pointer can be incremented/decremented to point to a new address but in Java, arithmetic operations on references are not allowed.

-No Pointer Manipulation in Java Although **a reference internally uses a pointer** but **Java does not allow any manipulation to an underlying pointer using a reference variable.**