**What is Java?**

**Java** is a general-purpose, class-based, object-oriented programming language designed for having lesser implementation dependencies. It is a computing platform for application development. Java is fast, secure, and reliable, therefore. It is widely used for developing Java applications in laptops, data centers, game consoles, scientific supercomputers, cell phones, etc.

**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.

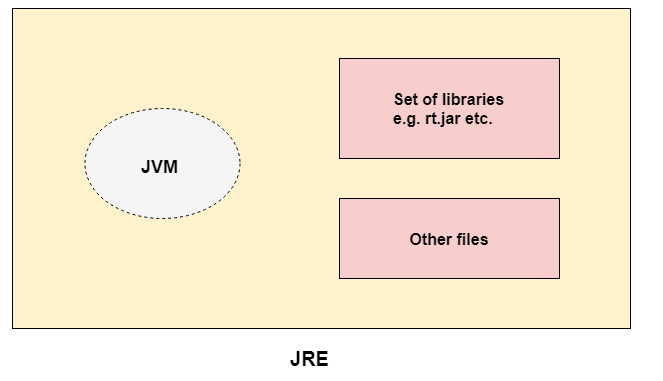
There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.

**Advantages**

* Java **package** is used to categorize by the classes and interfaces.
* It is easy to maintained.
* Java **package** is provide as access protection.
* It may remove naming collision.
* This **packages** can provide reusability of code.
* We can create our own **package** or extend already available **package**.

**3. Explain JDK, JRE and JVM?.**

**JDK** is a software development kit whereas **JRE** is a software bundle that allows **Java** program to run, whereas **JVM** is an environment for executing bytecode. The full form of **JDK** is **Java** Development Kit, while the full form of **JRE** is **Java** Runtime Environment, while the full form of **JVM** is **Java Virtual Machine**



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

Public- it is access specifier from anywhere we can access it

 Static- it is access modifier we can call the methods directly by class name without creating its objects

 Void- It is a keyword and used to specify that a method doesn’t return anything

 Main- it is a method name It is the identifier that the JVM looks for as the starting point of the java program. It’s not a keyword.

String[]args- It stores Java *command line arguments* and is an array of type *java.lang.String* class. Here, the name of the String array is *args*but it is not fixed and user can use any name in place of it.

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

* C++ uses only compiler, whereas Java uses compiler and interpreter both.
* C++ supports both operator overloading & method overloading whereas Java only supports method overloading.
* C++ supports manual object management with the help of new and delete keywords whereas Java has built-in automatic garbage collection.
* C++ supports structures whereas Java doesn’t supports structures.
* C++ supports unions while Java doesn’t support unions.

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

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. ... Different JVM is designed for different OS and byte code is able to run on different OS

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

Wrapper classes provide a way to use primitive data types (int, boolean, etc..) as objects.

The wrapper class in Java provides the mechanism *to convert primitive into object and object into primitive*.

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

Memory access via pointer arithmetic - this is fundamentally unsafe. Java has a robust security model and disallows pointer arithmetic for this reason. Java does have reference to objects, it just doesn't call them pointers. Any normal object reference works like a pointer would.

Java is platform independent. So if we use pointers the address of the variable varies in different machines. Thus using pointers becomes invalid and provides different result in different machine. Also java provides security without the use of pointers.

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

**Object Oriented**

In Java, everything is an Object. Java can be easily extended since it is based on the Object model.

**Platform Independent**

Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform-independent byte code. This byte code is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.

**Simple**

Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it would be easy to master.

**Secure**

With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.

Architecture-neutral Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.

Portable. Java makes an effort to eliminate error-prone situations by emphasizing mainly on compile time error checking and runtime checking.

**Multithreaded**

With Java's multithreaded feature it is possible to write programs that can perform many tasks simultaneously. This design feature allows the developers to construct interactive applications that can run smoothly.

**High Performance**

With the use of Just-In-Time compilers, Java enables high performance.

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

Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform-independent byte code. This byte code is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.

Thus when you write a piece of Java code in a particular platform and generated an executable code .class file. You can execute/run this .class file on any system the only condition is that the target system should have JVM (JRE) installed in it.

In short, Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.

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.

**11. 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

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

Java is considered as Dynamic because of Byte code [a class file]. A source code written in one platform, that can be executed in any platform. And it also loads the class files at run time. Anything that happens at run time is considered as Dynamic.

**14. List two Java IDE’s?**

1)Eclipse. 2)Kite.

**15. Why Java is called as “Platform” ?**

Platform is a software and hardware programs that runs. JAVA is platform independent because it having its own JVM so that it can run on any platform . java is platform independent , which means once written you can run it anywhere. The platform is a hardware or software used to run an application.

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

A fully object-oriented language needs to have all the 4 oops concepts. In addition to that, all predefined and, user-defined types must be objects and, all the operations should be performed only by calling the methods of a class.

Though java follows all the four object-oriented concepts,

Java has predefined primitive data types (which are not objects).You can access the members of a static class without creating an object of it.

Therefore, Java is not considered as fully object-oriented Technology.

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

The Java Compiler compiles a java program (.java file) and converts it into class files (.class) that contain bytecodes , which is the intermediate language between source code and machine code . These bytecodes are not platform specific, so with the help of JVM (Java virtual machine), the java program can run on wide variety of platforms. The JVM (Java virtual machine) is platform dependent i.e its implementation differs from platform to platform (like windows, linux atc.), but these all JVMs can execute the same java bytecode .

This is something which can be termed as 'write once and run anywhere' . So, this means Java can be developed on any device, compiled into a standard bytecode and be expected to run on any device equipped with a JVM (Java virtual machine).

**19. Difference between path and classpath.**

**PATH** − The path environment variable is used to specify the set of directories which contains executional programs. When you try to execute a program from command line, the operating system searches for the specified program in the current directly, if available, executes it. In case the programs are not available in the current directory, operating system verifies in the set of directories specified in the ‘PATH ’ environment variable.

**CLASSPATH** − The class path environment variable is used to specify the location of the classes and packages. When we try to import classes and packages other that those that are available with Java Standard Library. JVM verifies the current directly for them, if not available it verifies the set of directories specified in the ‘CLASSPATH’ environment variable

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

JDK is a software development kit whereas JRE is a software bundle that allows Java program to run, whereas JVM is an environment for executing bytecode.

The full form of JDK is Java Development Kit, while the full form of JRE is Java Runtime Environment, while the full form of JVM is Java Virtual Machine.

JDK is platform dependent, JRE is also platform dependent, but JVM is platform independent.

JDK contains tools for developing, debugging, etc. JRE contains class libraries and other supporting files, whereas software development tools are not included in JVM.

JDK comes with the installer, on the other hand, JRE only contains the environment to execute source code whereas JVM bundled in both software JDK and JRE.

The JRE is the Java Runtime Environment. It is a package of everything necessary to run a compiled Java program, including the Java Virtual Machine (JVM), the Java Class Library, the java command, and other infrastructure. However, it cannot be used to create new programs.

The JDK is the Java Development Kit, the full-featured SDK for Java. It has everything the JRE has, but also the compiler (javac) and tools (like javadoc and jdb). It is capable of creating and compiling programs. Usually, if you only care about running Java programs on computer you will only install the JRE. It's all you need. On the other hand, if you are planning to do some Java programming, you need to install the JDK instead

**22. What is JVM ? What it does?**

A Java virtual machine (JVM) is a virtual machine that enables a computer to run Java programs as well as programs written in other languages that are also compiled to Java bytecode

JVM is the engine that drives the Java code.

Mostly in other Programming Languages, compiler produce code for a particular system but Java compiler produce Bytecode for a Java Virtual Machine.

When we compile a Java program, then bytecode is generated. Bytecode is the source code that can be used to run on any platform.

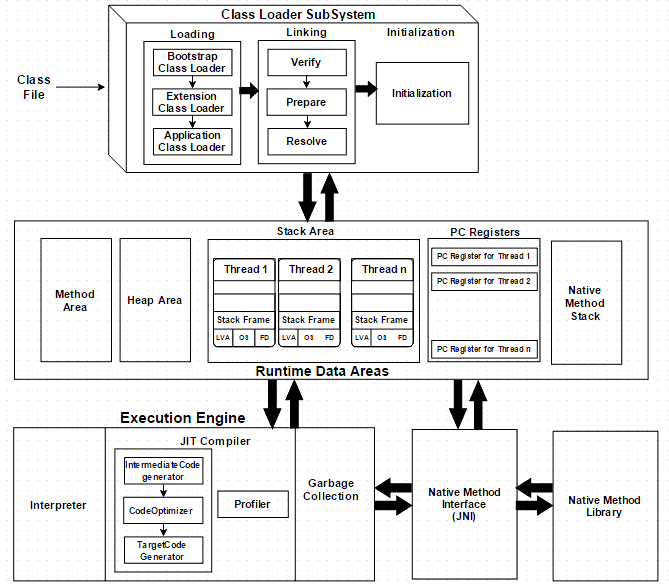
Bytecode is an intermediary language between Java source and the host system.

It is the medium which compiles Java code to bytecode which gets interpreted on a different machine and hence it makes it Platform/Operating system independent.

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

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. Therefore, the JVM must be implemented on a particular platform before compiled Java programs will run on that platform.

**24. What are the main components of JVM? Explain them. Or Explain JVM Architecture.**



**Class Loader Subsystem**

The class loader is a subsystem used for loading class files. It performs three major functions viz. Loading, Linking, and Initialization.

**2) Method Area** JVM Method Area stores class structures like metadata, the constant runtime pool, and the code for methods.

**3) Heap** All the Objects, their related instance variables, and arrays are stored in the heap. This memory is common and shared across multiple threads.

**4) JVM language Stacks** Java language Stacks store local variables, and it’s partial results. Each thread has its own JVM stack, created simultaneously as the thread is created. A new frame is created whenever a method is invoked, and it is deleted when method invocation process is complete.

**5) PC Registers** PC register store the address of the Java virtual machine instruction which is currently executing. In Java, each thread has its separate PC register.

**6) Native Method Stacks** It contains all the native methods used in the application. Native method stacks hold the instruction of native code depends on the native library. It is written in another language instead of Java.

**7) Execution Engine**

**It contains:**

**A virtual processor**

**Interpreter:** Read bytecode stream then execute the instructions.

**Just-In-Time(JIT) compiler:** It is used to improve the performance. JIT compiles parts of the byte code that have similar functionality at the same time, and hence reduces the amount of time needed for compilation. Here, the term "compiler" refers to a translator from the instruction set of a Java virtual machine (JVM) to the instruction set of a specific CPU. It is a type of software used to test hardware, software, or complete systems. The test execution engine never carries any information about the tested product.

**8) Native Method interface** Java Native Interface (JNI) is a framework which provides an interface to communicate with another application written in another language like C, C++, Assembly etc. Java uses JNI framework to send output to the Console or interact with OS librariesThe Native Method Interface is a programming framework. It allows Java code which is running in a JVM to call by libraries and native applications.

**9) Native Method Libraries** Native Libraries is a collection of the Native Libraries(C, C++) which are needed by the Execution Engine.

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

When **compiling** a **java** program, the static **compiler** that is run using the command **javac** converts the source code to byte code which are **in the** form of . class files. ... **JIT** compiles the code when it is needed but not before runtime.

26) Is Empty .java file name a valid source file name?

Answer :

You can have .java file withought any name . you have to compile it by

javac .java(it compile successfuly) and run it by java clasnname.

(so you must provide a class name)

class Test {

public static void main(String args[]) {

System.out.println("Hello World");

}

}

to compile: javac .java

to execute: java Test

27) Is JRE different for different Platforms ?

Answer:

Whenever we try to run the code, JVM requires some library set and files for code execution and these files are presented in JRE.

JRE = JVM + set of libraries.

So to run the JVM based programs, the JRE is a minimum requirement.

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

28) Difference between C++ and java in terms of object creation.

Answer:

In C++ objects are managed manually. The creation and destruction of objects are carried out manually

using the new and delete operators respectively.

We also use constructors and destructors for class objects.

Java does not support destructors though it supports constructors.

Java is also heavily dependent on automatic garbage collection for collecting and destroying objects.

29) Who invokes main() function ?

Answer:

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.

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.

30. What is .class file known as ?

Answer:

Class file is a binary file which contains bytecode and Java compiler is used to crate Class file in Java.

Class file in Java is compiled form of Java source file.

When we compile Java program which is written in Java source file ended with .java extension,

it produces one more class files depending upon how many classes are declared and defined in that Java source file.

One Java source file can only contain one public class and its name must match with name of file

e.g. HelloWorld.java file can contain a public class whose name should be HelloWorld, then after compilation of this source file it will creat

one class file as HelloWorld.class

Q36 What is the rule for local member in java.

Local variale- Variables declared inside a method and cant be access outside the method

public class Test

{

void method1()

{

int x=1; // Local variable

}

}

Local variables don’t exist after method’s execution is over.

Local variables are not assigned a default value, hence they need to be initialized

Access modifiers cannot be used for local variables

Final is the Only Non Access Modifier that can be applied to a local variable.

Q37-What is native code?

Native code is computer programming (code) that is compiled to run with a particular processor (such as an Intel x86-class processor) and its set of instructions.

Native code can also be distinguished from byte code.

Byte code- A form of code that can be run in a virtual machine(eg.JVM).

The virtual machine is a program that converts the byte code into the native code that will be run in a specific processor

Machine code also called as native code. it is a program which is written in machine language.

machine code considered as lowest level code for computer .

It is written in binary but also written in octal and hexadecimal to make it easier to handle.

38.Why there is no sizeof operator in java ?

In c there is size of operator is used.

In c is useful only because you have to manually allocate and free memory. However, since in java there is automatic garbage collection, this is not necessary.

In java, you don't work directly with memory, so sizeof is usually not needed

39.What kinds of programs u can develop using Java ?

Java is currently the most popular Programming lanuage.

Following are tha applications of java programming.

Application serve.rs

Web applications.

Scientific Applicatons.

Mobile applications.

Desktop applications.

Enterprise applications.

cloud based applications.

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

Q45 What kinds of programs u can develop using java?

* Java Mobile Applications. (android studio and java for apps)
* Java Desktop GUI(graphical user interface) Applications. (change color,size , visuals)
* Java Web-based Applications. (for dynamic website creation)
* Java Web Servers and Application Servers. (one of the big server of java -Apache Tomcat)
* Java Enterprise Applications. (The java enterprise  provides a platform for developers with enterprise  features such as distributed computing and web services)
* Java Scientific Applications. (MATLAB)
* Java Gaming Applications. (BOUNCE nokia)

Q46 When parseInt () method can be used?

-->Convert a string to an integer with the parseInt method of the Java Integer class. The parseInt method is to convert the String to an int and throws a Number Format Exception if the string cannot be converted to an int type.

Q47 What is finalized () method?

-->Finalize () is the method of Object class. This method is called just before a0n object is garbage collected. Finalize () method overrides to dispose system resources, perform clean-up activities and minimize memory leaks.

Q48.Difference between C++ pointer and Java reference.

-->**:** A reference is a variable that refers to something else and can be used as an alias for that something else.  
**Pointer:** A pointer is a variable that stores a memory address, for the purpose of acting as an alias to what is stored at that address.  
So, a pointer is a reference, but a reference is not necessarily a pointer. Pointers are a particular implementation of the concept of a reference, and the term tends to be used only for languages that give you direct access to the memory address. Q50.What are the expressions allowed in switch block of java?