**1. 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.

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

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

Java Virtual Machine (JVM) is a specification that provides runtime environment in which java bytecode (. class files) can be executed. ... JVM makes this possible because it is aware of the specific instruction lengths and other particularities of the platform (Operating System). The JVM is not platform independent.

**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.

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

jdk1.8.0\_281

**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

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

**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.

**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.

Just in time compiler coverts the byte code to a platform specific executable code that can be executed immediately. However the use of JIT is optional, but Sun Microsystems suggest that it is quite efficient to use JIT especially if there exists a repeated code in the program.

(jit) is platform-dependent and compiles some portions of the bytecode to machine-code in runtime!

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

Yes. An empty . java file is a perfectly valid source file.

Yes, save your java file by .java then compile it by javac .java and run by java yourclassname

**to compile**: javac .java

**to execute:** java Test

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

Like of JDK 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.**

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 ?**

The Java Virtual Machine then links the initial class, initializes it, and invokes the public class method void main(String[]). The invocation of this method drives all further execution. Execution of the Java Virtual Machine instructions constituting the main method may cause linking (and consequently creation) of additional classes and interfaces, as well as invocation of additional methods.

The main method is the entry point of the JVM when the class in launched. The JVM launchs the Java program by invoking the main method of the class identified in the command to start the program.

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

A Java class file is a file (with the . class filename extension) containing Java bytecode that can be executed on the Java Virtual Machine (JVM). A Java class file is usually produced by a Java compiler from Java programming language source files

**31. 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.

**java naming-conventions**. In JAVA, class name must always be the same as file name, but sometimes file contains multiple classes. Only single class(or interface) in file can be public, and it must have the same name as file

**32. What is JIT compiler?**

The Just-In-Time (JIT) compiler is a component of the runtime environment that improves the performance of Java™ applications by compiling bytecodes to native machine code at run time. ... The JIT compiler helps improve the performance of Java programs by compiling bytecodes into native machine code at run time.

**eg**.A just-in-time (JIT) inventory system is a management strategy that has a company receive goods as close as possible to when they are actually needed. So, if a car assembly plant needs to install airbags, it does not keep a stock of airbags on its shelves, but receives them as those cars come onto the assembly line.

**33. How many types of memory areas are allocated by JVM?**

The memory in the JVM divided into 5 different parts:

1) Class Mehod Area

2) Heap memory

3) Java Stack area

4) Program Counter Register

5) Native Method Stack

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

A variable that is declared within the method that is called local variables. It is defined in method or other statements, such as defined and used within the cache block, and outside the block or method, the variable cannot be used.Local variables are not assigned a default value, hence they need to be initialized.

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

In java we have four Access Specifiers and they are listed below.

1. public

2. private

3. protected

4. default(no specifier)



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

A variable that is declared within the method that is called local variables. It is defined in method or other statements, such as defined and used within the cache block, and outside the block or method, the variable cannot be used.Local variables are not assigned a default value, hence they need to be initialized.

**37. What is native code?**

Native code refers to programming code that is configured to run on a specific processor. Native code will generally not function if used on a processor other than the one it was specifically written for unless it is allowed to run over an emulator.

Because native code is tailor-made for a specific processor, programs written using such code should (theoretically) run optimally on the native processor. The downside is that such programs will usually be rendered useless when used on a different processor. This limitation has steadily decreased since manufacturers have started configuring their processors to work with the same kind of protocols and logic sequences.

Although a program written on native code can run on a processor it was not originally intended for by using emulation software (which mimics the configuration of the originally intended processor), the program’s performance will generally suffer.

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

In c it 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.

C needed sizeof because the size of ints and longs varied depending on the OS and compiler. In Java all sizes, bit configurations are better defined.

Because the size of primitive types is explicitly mandated by the Java language. There is no variance between JVM implementations.

Moreover, since allocation is done by the new operator depending on its argument there is no need to specify the amount of memory needed.

It would sure be convenient sometimes to know how much memory an object will take so you could estimate things like max heap size requirements.

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

Mobile Applications

Desktop GUI Applications

Web-based Applications

Enterprise Applications

Scientific Applications

Gaming Applications

Big Data technologies

Business Applications

Distributed Applications

Cloud-based Applications

**Mobile Applications**

Java is considered as the official programming language for mobile app development. It is compatible with software such as Android Studio and Kotlin. java runs on Java Virtual Machine(JVM), whereas Android uses DVK(Dalvik Virtual Machine) to execute class files. These files are further bundled as an Android application package(APK). With Java and its OOPs principles, it provides better security and ease of simplicity with Android.

**Web-based Applications**

Java is also used to develop web applications. It provides vast support for web applications through Servlets, Struts, or JSPs. With the help of these technologies, you can develop any kind of web application that you require. The easy coding and high security offered by this programming language allow the development of a large number of applications for health, social security, education, and insurance.

**Scientific Applications**

Sofware developers see Java is the weapon of choice when it comes to coding the scientific calculations and mathematical operations. These programs are designed to be highly secure and lighting fast. they support a higher degree of portability and offer low maintenance. Some of the most powerful applications like the MATLAB use Java for interacting user interface as well as part of the core system.

**Gaming Applications**

Java has the support of the open-source most powerful 3D-Engine, the jMonkeyEngine that has the unparalleled capability when it comes to the designing of 3D games.

**Big Data technologies**

Java is the reason why the leading Big Data technologies like Hadoop have become a reality and also the most powerful programming languages like Scala are existing. It is crystal clear that Java is the backbone when it comes to developing Big Data using Java.

**Cloud-Based Applications:**

Cloud computing means on-demand delivery of IT resources via the internet with pay-as-you-go pricing. It provides a solution for IT infrastructure at a low cost. Java provides you with features that can help you build applications meaning that it can be used in the SaaS, IaaS and PaaS development. It can serve the companies to build their applications remotely or help companies share data with others, whatever the requirement.

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

**Null**, The default value of a reference type variable is null when they are not initialized. Null means not refering to any object.

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

The Java ClassLoader is a part of the Java Runtime Environment that dynamically loads Java classes into the Java Virtual Machine. The Java run time system does not need to know about files and file systems because of classloaders. Java classes aren't loaded into memory all at once, but when required by an application.

Built-in ClassLoader Types

**There are three types of built-in ClassLoader in Java.**

**Bootstrap Class Loader –** It loads JDK internal classes. It loads rt.jar and other core classes for example java.lang.\* package classes.

**Extensions Class Loader –** It loads classes from the JDK extensions directory, usually $JAVA\_HOME/lib/ext directory.

**System Class Loader –** This classloader loads classes from the current classpath. We can set classpath while invoking a program using -cp or -classpath command line option.



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

The class loaders in Java are organized in a tree. By request a class loader determines if the class has already been loaded in the past, looking up in its own cache. If the class is present in the cache the CL returns the class, if not, it delegates the request to the parent. If the parent is not set (is Null) or can not load the class and throws a ClassNotFoundException the classloader tries to load the class itself and searches its own path for the class file. If the class can be loaded it is returned, otherwise a ClassNotFoundException is thrown. The cache lookup goes on recursively from child to parent, until the tree root is reached or a class is found in cache. If the root is reached the class loaders try to load the class and unfold the recursion from parent to child. Summarizing that we have following order:

Cache

Parent

Self

This mechanism ensures that classes tending to be loaded by class loaders nearest to the root. Remember, that parent class loader is always has the opportunity to load a class first. It is important to ensure that core Java classes are loaded by the bootstrap loader, which guarantees that the correct versions of classes such as java.lang.Object are loaded. Furthermore it ensures, that one class loader sees only classes loaded by itself or its parent (or further ancestors) and it cannot see classes loaded by its children or siblings!

The picture illustrates the hierarchy of class loaders. Root loader is bootstrap class loader which has native implementation and cannot be instantiated by Java code.



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

**Verify** - the bytecode verifier checks if the bytecode loaded are valid and do not breach java's security restrictions and checks that the instructions cannot perform actions that are damaging. All classes except for system classes are verified**.**

**Some of the checks that verifier performs:**

Uninitialized Variables

Access rules for private data and methods are not violated.

Method calls match the object Reference.

There are no operand stack overflows or underflows.

The arguments to all the Java Virtual Machine instructions are of valid types.

Ensuring that final classes are not subclassed and that final methods are not overridden

Checking that all field references and method references have valid names, valid classes, and a valid type descriptor

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

The memory in the JVM divided into 5 different parts:

Class(Method) Area

Heap

Stack

Program Counter Register

Native Method Stack

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

Mobile Applications

Desktop GUI Applications

Web-based Applications

Enterprise Applications

Scientific Applications

Gaming Applications

Big Data technologies

Business Applications

Distributed Applications

Cloud-based Applications

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

While operating upon strings, there are times when we need to convert a number represented as a string into an integer type. The method generally used to convert String to Integer in Java is parseInt().

**47. What is finalize () method?**

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

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

**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.

**50. What are the expressions allowed in switch block of java ?**

A **switch statement** allows the application to have **multiple possible execution paths** based on the value of a given expression in runtime.

The evaluated expression is called the **selector expression** which must be of type char, byte, short, int, Character, Byte, Short, Integer, String, or an enum

It has the support of **multiple case labels** and using **yield** to return value in place of old return keyword.

* It also support returning value via label rules (arrow operator similar to lambda).
* If we use arraw (->) operator, we can skip yield keyword as shown in *isWeekDayV1\_1()*.
* If we use colon (:) operator, we need to use yield keyword as shown in *isWeekDayV1\_2()*.
* In case of multiple statements, use **curly braces** along with yield keyword as shown in *isWeekDayV2()*.
* In case of [enum](https://howtodoinjava.com/java/enum/enum-tutorial/), we can skip the **default case**. If there is any missing value not handled in cases, compiler will complain. In all other expression types (int, strings etc), we must provide default case as well.

public class SwitchExpressions

{

public static void main(String[] argv)

{

System.out.println(isWeekDayV1\_1(Day.MON)); //true

System.out.println(isWeekDayV1\_2(Day.MON)); //true

System.out.println(isWeekDayV2(Day.MON)); //true

}

//1 - Return value directly

enum Day {

MON, TUE, WED, THUR, FRI, SAT, SUN

};

public static Boolean isWeekDayV1\_1 (Day day)

{

Boolean result = switch(day) {

case MON, TUE, WED, THUR, FRI -> true;

case SAT, SUN -> false;

};

return result;

}

public static Boolean isWeekDayV1\_2 (Day day)

{

Boolean result = switch(day) {

case MON, TUE, WED, THUR, FRI : yield true;

case SAT, SUN : yield false;

};

return result;

}

//2 - Multiple statements

public static Boolean isWeekDayV2 (Day day)

{

Boolean result = switch(day) {

case MON, TUE, WED, THUR, FRI ->

{

System.out.println("It is WeekDay");

yield true;

}

case SAT, SUN ->

{

System.out.println("It is Weekend");

yield false;

}

};

return result;

}

}