1. What is Java?

- Java is a **programming language** and a **platform**.
- Java is a high level, robust, object-oriented and secure programming language.
- Java was developed by *Sun Microsystems* (which is now the subsidiary of Oracle) in the year 1995.
- *James Gosling* is known as the father of Java.
- Before Java, its name was *Oak*. Since Oak was already a registered company, so James Gosling and his team changed the name from Oak to Java.

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?

JDK

- JDK is an acronym for Java Development Kit. The Java Development Kit (JDK) is a software development environment which is used to develop Java applications and applets. It physically exists. It contains JRE + development tools.
- The JDK contains a private Java Virtual Machine (JVM) and a few other resources such as an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc), etc. to complete the development of a Java Application.

JRE

- JRE is an acronym for Java Runtime Environment. It is also written as Java RTE.
- The Java Runtime Environment is a set of software tools which are used for developing Java applications.
- It is used to provide the runtime environment.
- It is the implementation of JVM. It physically exists. It contains a set of libraries + other files that JVM uses at runtime.

JVM

- JVM (Java Virtual Machine) is an abstract machine. It is called a virtual machine because it doesn't physically exist. It is a specification that provides a runtime environment in which Java bytecode can be executed. It can also run those programs which are written in other languages and compiled to Java bytecode. JVMs are available for many hardware and software platforms. JVM, JRE, and JDK are platform dependent because the configuration of each OS is different from each other. However, Java is platform independent.
- The JVM performs the following main tasks:

    - Loads code
    - Verifies code
    - Executes code
    - Provides runtime environment

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

1. Public: It is an *Access modifier*, which specifies from where and who can access the method.

2. Static: It is a *keyword* that is when associated with a method, making it a class-related method. The *main()* method is static so that JVM can invoke it without instantiating the class.

3. Void: It is a keyword and is used to specify that a method doesn't return anything. As the *main()* method doesn't return anything, its return type is *void*.

4. main: It is the name of the Java main method. It is the identifier that the JVM looks for as the starting point of the java program. It's not a keyword.

5. 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 the user can use any name in place of it.

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

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.	Java was designed and created as an interpreter for printing systems but later extended as a support network computing.
Goto	C++ supports the goto 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.
Operator Overloading	C++ supports operator overloading.	Java doesn't support operator overloading.
Pointers	C++ supports 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.

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.	Java has built-in thread 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 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 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.

6. Why Java is platform independent?

- Java is platform independent because it is different from other languages like C, C++, 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.
- 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

7. Why Java is platform independent?

- Java is platform independent because it is different from other languages like C, C++, 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.
- 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

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.

9. Why pointers are not used in Java?

- Java doesn’t use pointers because they are unsafe and increases the complexity of the program.
- Since, Java is known for its simplicity of code, adding the concept of pointers will be contradicting.
- Moreover, since JVM is responsible for implicit memory allocation, thus in order to avoid direct access to memory by the user, pointers are discouraged in Java.

10. List some features of Java?

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

1. Simple
2. Object-Oriented
3. Portable
4. Platform independent
5. Secured
6. Robust
7. Architecture neutral
8. Interpreted
9. High Performance
10. Multithreaded
11. Distributed
12. Dynamic

11. Why is Java Architectural 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.

12. How Java enabled 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.

13. Why Java is considered 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++.
- Java supports dynamic compilation and automatic memory management (garbage collection).

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?

- **Eclipse,**
- **NetBeans,**
- **IntelliJ IDEA**

16. Why Java is called as "Platform"?

17. Is Java Pure-Object oriented Language?

Java language is not a Pure Object Oriented Language as it contain these properties:

- **Primitive Data Type ex. int, long, bool, float, char, etc as Objects**
- **The static keyword**
- **Wrapper Class**

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

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

- 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).

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?

- 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.

22. What is the difference between JDK and JRE?

KEY	JDK	JRE
DEFINITION	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.
FUNCTIONALITY	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.
DEPENDENCY OF PLATFORM	It is platform-dependent.	It is also platform-dependent like JDK.
TYPE OF TOOLS	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.
IMPLEMENTATION OF JDK AND JRE	JDK = JRE + other development tools.	JRE = JVM + other class libraries.

23. What is JVM? What it does?

What is JVM:

It is:

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.

What it does:

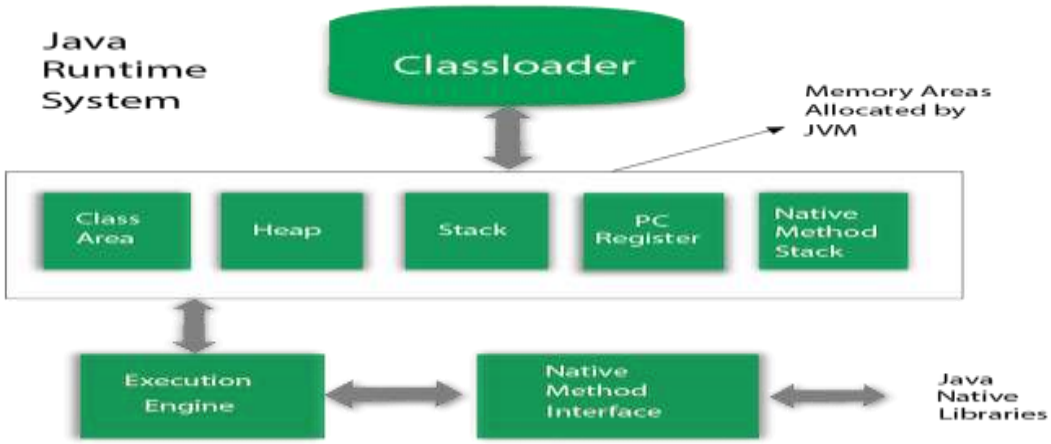
The JVM performs following operation:

- Loads code
- Verifies code
- Executes code
- Provides runtime environment

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.

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



1) Classloader

Classloader is a subsystem of JVM which is used to load class files. Whenever we run the java program, it is loaded first by the classloader. There are three built-in classloaders in Java.

1. **Bootstrap ClassLoader**
2. **Extension ClassLoader**
3. **System/Application ClassLoader**

2) Class(Method) Area
Class(Method) Area stores per-class structures such as the runtime constant pool, field and method data, the code for methods.

3) Heap
It is the runtime data area in which objects are allocated.

4) Stack
Java Stack stores frames. It holds local variables and partial results, and plays a part in method invocation and return.
Each thread has a private JVM stack, created at the same time as thread.
A new frame is created each time a method is invoked. A frame is destroyed when its method invocation completes.

5) Program Counter Register
PC (program counter) register contains the address of the Java virtual machine instruction currently being executed.

6) Native Method Stack
It contains all the native methods used in the application.

7) Execution Engine
It contains:

1. **A virtual processor**
2. **Interpreter:** Read bytecode stream then execute the instructions.
3. **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.

8) Java Native 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 libraries.

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

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

- True. An empty .java file name a valid source file name.

28. Is JRE different for different Platforms?

- Yes. JRE is platform dependent because the configuration of each OS is different from each other.

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

30. Who invokes main() function ?

- JVM invokes main() function.

31. What is .class file known as?

- .class file contains the Bytecode..

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

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

- Java allows us to create any number of classes in a program.
- But out of all the classes, at most, one of them can be declared as a public class.
- In simple words, the program can contain either zero public class, or if there is a public class present, it cannot be more than one.
- The myth about the file name and class name should be same only when the class is declared in **public**.

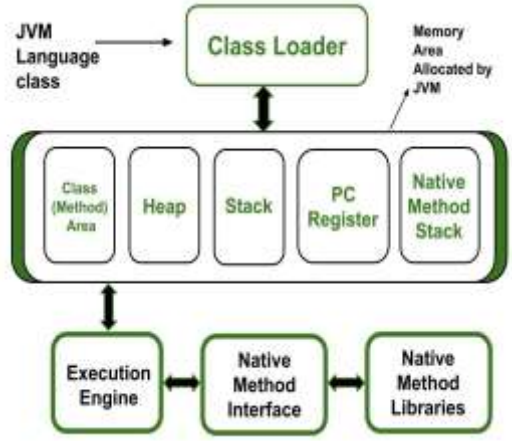
34. What is JIT compiler?

- **JIT** stands for **Java-In-Time Compiler**.
- **JIT** in Java is an integral part of the **JVM**.
- It accelerates execution performance many times over the previous level.
- In other words, it is a long-running, computer-intensive program that provides the best performance environment.
- It optimizes the performance of the Java application at compile or run time.

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

- 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



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

- A variable declared inside the body of the method is called local variable.
- You can use this variable only within that method and the other methods in the class aren't even aware that the variable exists.
- A local variable cannot be defined with "static" keyword.

37. What are the various access specifiers in Java?

- Java has the following access modifiers:

1. Private: The scope of the private data member and member function is limited and can access them only within the same class in which they declare.
2. Protected: We can access the protected data members and member functions of a class within the same package or the subclasses in different packages.
3. public: It is one of the access modifiers which have the widest scope. We can access the methods and variables of a class from anywhere in the program, which is declared as public.
4. default: When we define class, data members, and member function without specifying any access modifier, it has a **default** access modifier by default.

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

- A variable declared inside the body of the method is called local variable.
- You can use this variable only within that method and the other methods in the class aren't even aware that the variable exists.
- A local variable cannot be defined with "static" keyword.

39. What is native code?

40. Why there is no sizeof operator in java?

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

- We can use Java technology to develop the following applications:

  - Mobile App Development
  - Desktop GUI Applications
  - Web-based Applications
  - Gaming Applications
  - Big Data Technologies
  - Distributed Applications
  - Cloud-based Applications
  - IoT Applications

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

43. What is the job done by classloader?

- Java ClassLoader is an abstract class.
- It belongs to a **java.lang** package.
- It loads classes from different resources.
- Java ClassLoader is used to load the classes at run time.
- In other words, JVM performs the linking process at runtime.
- Classes are loaded into the JVM according to need. If a loaded class depends on another class, that class is loaded as well.
- When we request to load a class, it delegates the class to its parent.
- In this way, uniqueness is maintained in the runtime environment.
- It is essential to execute a Java program.

44. Explain the hierarchy of classloaders in java.

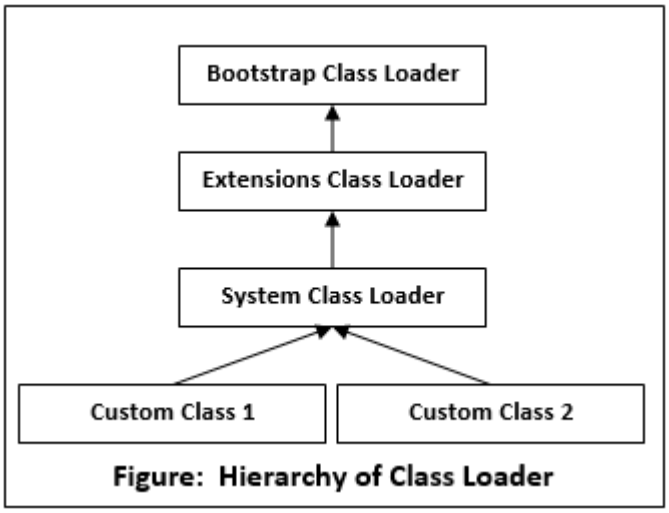


Figure: Hierarchy of Class Loader

45. What is the role played by Bytecode Verifier?

46. What are the memory areas allocated by JVM?

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

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

- Mobile App Development
- Desktop GUI Applications
- Web-based Applications
- Gaming Applications
- Big Data Technologies
- Distributed Applications
- Cloud-based Applications
- IoT Applications

48. When parseInt() method can be used?

- The method **parseInt()** belongs to the Integer class which is under **java.lang package**.
- It is used to parse the string value as a signed decimal value.
- It is used in Java for converting a string value to an integer by using the method parseInt().

49. What is finalized() 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.

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