1. Why is Java a platform independent language?

- Java language was developed so that it does not depend on any hardware or software because the compiler compiles the code and then converts it to platform-independent byte code which can be run on multiple systems.

2. What is JVM?

- JVM(Java Virtual Machine) acts as a run-time engine to run Java applications. JVM is the one that actually calls the main method present in a Java code. JVM is a part of JRE(Java Runtime Environment).

Java applications are called WORA (Write Once Run Anywhere). This means a programmer can develop Java code on one system and can expect it to run on any other Java-enabled system without any adjustment. This is all possible because of JVM.

When we compile a *.java* file, *.class* files(contains byte-code) with the same class names present in *.java* file are generated by the Java compiler. This *.class* file goes into various steps when we run it. These steps together describe the whole JVM.

JVM provides definitions for the:

- Memory area
- Class file format
- Register Set
- Garbage Collecting Heap
- Fatal Error Reporting

3. What are the memory storages available in JVM?

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

4. Difference between Heap and Stack Memory in java. And how java utilizes this.

- Stack memory is the portion of memory that was assigned to every individual program. And it was fixed. On the other hand, Heap memory is the portion that was not allocated to the java program but it will be available for use by the java program when it is required, mostly during the runtime of the program.

Java Utilizes this memory as -

1. When we write a java program then all the variables, methods, etc are stored in the stack memory.

2. And when we create any object in the java program then that object was created in the heap memory. And it was referenced from the stack memory.

5. Difference between JVM, JRE, and JDK.

JVM: JVM also known as Java Virtual Machine is a part of JRE. JVM is a type of interpreter responsible for converting bytecode into machine-readable code. JVM itself is platform dependent but it interprets the bytecode which is the platform-independent reason why Java is platform-independent.
JRE: JRE stands for Java Runtime Environment, it is an installation package that provides an environment to run the Java program or application on any machine.
JDK: JDK stands for Java Development Kit which provides the environment to develop and execute Java programs. JDK is a package that includes two things Development Tools to provide an environment to develop your Java programs and, JRE to execute Java programs or applications.

6. How is Java different from C++?

- C++ is only a compiled language, whereas Java is compiled as well as an interpreted language.
- Java programs are machine-independent whereas a c++ program can run only in the machine in which it is compiled.
- C++ allows users to use pointers in the program. Whereas java doesn't allow it. Java internally uses pointers.

- C++ supports the concept of Multiple inheritances whereas Java doesn't support this. And it is due to avoiding the complexity of name ambiguity that causes the diamond problem.

7. What are Packages in Java? Its uses and advantages.

- Packages in Java can be defined as the grouping of related types of classes, interfaces, etc providing access to protection and namespace management. Packages are used in Java in order to prevent naming conflicts, control access, and make searching/locating and usage of classes, interfaces, etc easier. There are various advantages of defining packages in Java.

  - Packages avoid name clashes.
  - The Package provides easier access control.
  - We can also have the hidden classes that are not visible outside and are used by the package.
  - It is easier to locate the related classes.

8. What are the different types of variables?

- There are three different types of variables in Java, we have listed them as follows:

- Instance variables: are those variables that are accessible by all the methods in the class. They are declared outside the methods and inside the class. These variables describe the properties of an object and remain bound to it at any cost.
- Local variables: are those variables present within a block, function, or constructor and can be accessed only inside them. The utilization of the variable is restricted to the block scope. Whenever a local variable is declared inside a method, the other class methods don’t have any knowledge about the local variable.
- Static variables: A variable that is declared as static is called a static variable. It cannot be local. You can create a single copy of the static variable and share it among all the instances of the class. Memory allocation for static variables happens only once when the class is loaded in the memory.

9. Explain different types of data types

There are 2 types of data types in Java as mentioned below:

1. Primitive Data Type
2. Non-Primitive Data Type or Object Data type

Primitive Data Type: Primitive data types specify the size and type of variable values. They are the building blocks of data manipulation and cannot be further divided into simpler data types. There are 8 primitive data types:

- boolean: stores value true or false
- byte: stores an 8-bit signed two's complement integer
- char: stores a single 16-bit Unicode character
- short: stores a 16-bit signed two's complement integer
- int: stores a 32-bit signed two's complement integer
- long: stores a 64-bit two's complement integer
- float: stores a single-precision 32-bit IEEE 754 floating-point
- double: stores a double-precision 64-bit IEEE 754 floating-point

Non-Primitive Data Type: Non-primitive data types or reference data types refer to instances or objects. They cannot store the value of a variable directly in memory. They store a memory address of the variable. Unlike primitive data types we define by Java, non-primitive data types are user-defined. Programmers create them and can be assigned with null. All non-primitive data types are of equal size. Types of Non-Primitive are mentioned below:

- Strings
- Array
- Class
- Interface
- Enum

10. How many ways you can take input from the console?

There are four methods to take input from the console in Java mentioned below:

1. Using Command line argument
2. Using Buffered Reader Class
3. Using Console Class
4. Using Scanner Class

11. What is the difference between float and double datatypes?

- The key difference between a float and double in Java is that a double can represent much larger numbers than a float. Both data types represent numbers with decimals, but a float is 32 bits in size while a double is 64 bits. A double is twice the size of a float — thus the term *double*.

12. How is the creation of a String using new() different from that of a literal?

- String using new() is different from the literal as when we declare string it stores the elements inside the stack memory whereas when it is declared using new() it allocates a dynamic memory in the heap memory. The object gets created in the heap memory even if the same content object is present.

13.What is array?

- An Array in Java is a data structure that is used to store a fixed-size sequence of elements of the same type. Elements of an array can be accessed by their index, which starts from 0 and goes up to a length of minus 1. Array declaration in Java is done with the help of square brackets and size is also specified during the declaration.

14. Explain super keyword in JAVA.

- The super keyword in Java is a reference variable which is used to refer immediate parent class object. Whenever you create the instance of subclass, an instance of parent class is created implicitly which is referred by super reference variable.

15. What is the use of final keyword?

- The final keyword can be applied with the variables, a final variable that have no value it is called blank final variable or uninitialized final variable. It can be initialized in the constructor only. The blank final variable can be static also which will be initialized in the static block only. We will have detailed learning of these. Let's first learn the basics of final keyword

16.What is the purpose of this keyword?

- In Java, ‘this’ is a reference variable that refers to the current object, or can be said “this” in Java is a keyword that refers to the current object instance. It can be used to call current class methods and fields, to pass an instance of the current class as a parameter, and to differentiate between the local and

instance variables. Using “this” reference can improve code readability and reduce naming conflicts.

17. Object: An entity that has state and behaviour is known as an object e.g., chair, bike, marker, pen, table, car, etc. It can be physical or logical (tangible and intangible). The example of an intangible object is the banking system.

18. Class: A class is a group of objects which have common properties. It is a template or blueprint from which objects are created. It is a logical entity. It can't be physical.

19. How many Characteristics an Object possess? What are they?

- variables, which hold values that can be changed;
- data structures, which are specialized formats used to organize and process data;
- functions, which are named procedures that perform a defined task; and
- methods, which are programmed procedures that are defined as components of a parent class and are included in any instance of that class.

20. What are the different ways of creating objects in Java?

1. Using new keyword
2. Using new instance
3. Using clone() method
4. Using deserialization
5. Using newInstance() method of Constructor class

21. What is a constructor?

- Constructor is a special method that is used to initialize objects. Constructor is called when a object is created. The name of constructor is same as of the class.

22. How many types of constructors are used in Java?

- There are two types of constructors in Java as mentioned below:
    1. Default Constructor: It is the type that does not accept any parameter value. It is used to set initial values for object attributes.
    2. Parameterized Constructor: It is the type of constructor that accepts parameters as arguments. These are used to assign values to instance variables during the initialization of objects.

23. What are the differences between the constructors and methods?

- Java constructors are used for initializing objects. During creation, constructors are called to set attributes for objects apart from these few basic differences between them are:
    1. Constructors are only called when the object is created but other methods can be called multiple times during the life of an object.
    2. Constructors do not return anything, whereas other methods can return anything.
    3. Constructors are used to setting up the initial state but methods are used to perform specific actions.

24. What is interface?

- An interface in Java is a collection of static final variables and abstract methods that define the contract or agreement for a set of linked classes. Any class that implements an interface is required to implement a specific set of methods. It specifies the behaviour that a class must exhibit but not the specifics of how it should be implemented.

25. What is Data Encapsulation?

- Data Encapsulation is the concept of OOPS properties and characteristics of the classes that The interface is binded together. Basically, it bundles data and methods that operate on that data within a single unit. Encapsulation is achieved by declaring the instance variables of a class as private, which means they can only be accessed within the class.

The advantages of Encapsulation in Java are mentioned below:

1. Data Hiding: it is a way of restricting the access of our data members by hiding the implementation details. Encapsulation also provides a

way for data hiding. The user will have no idea about the inner implementation of the class.
2. Increased Flexibility: We can make the variables of the class read-only or write-only depending on our requirements.
3. Reusability: Encapsulation also improves the re-usability and is easy to change with new requirements.
4. Testing code is easy: Code is made easy to test for unit testing.

26. Define Inheritance.

- When an object that belongs to a subclass acquires all the properties and behaviour of a parent object that is from the superclass, it is known as inheritance. A class within a class is called the subclass and the latter is referred to as the superclass. Sub class or the child class is said to be specific whereas the superclass or the parent class is generic. Inheritance provides code reusability.

Inheritance is the method by which the Child class can inherit the features of the Super or Parent class. In Java, Inheritance is of four types:

- Single Inheritance: When a child or subclass extends only one superclass, it is known to be single inheritance. Single-parent class properties are passed down to the child class.
- Multilevel Inheritance: When a child or subclass extends any other subclass a hierarchy of inheritance is created which is known as multilevel inheritance. In other words, one subclass becomes the parent class of another.
- Hierarchical Inheritance: When multiple subclasses derive from the same parent class is known as Hierarchical Inheritance. In other words, a class that has a single parent has many subclasses.
- Multiple Inheritance: When a child class inherits from multiple parent classes is known as Multiple Inheritance. In Java, it only supports multiple inheritance of interfaces, not classes.

27. What is Polymorphism?

- Polymorphism is defined as the ability to take more than one form It is of two types namely,

- Compile time polymorphism or method overloading- Method overriding is a feature that allows a child class to provide a specific implementation of a method that is already provided by one of its parent classes. When a method in a child class has the same name, the same parameters or signature, and the same return type (or sub-type) as a method in its parent class, then the method in the subclass is said to override the method in the superclass
- Run time polymorphism or method overriding- Method overriding, also known as run time polymorphism is one where the child class contains the same method as the parent class.

28. What is Data Abstraction?

- Abstraction refers to the act of representing essential features without including background details. The detailed information or the implementation is hidden. The most common example of abstraction is a car, we know how to turn on the engine, accelerate and move, however, the way engine works, and its internal components are complex logic hidden from the general users. This is usually done to handle the complexity.

29. What is abstract class?

- A class declared as abstract, cannot be instantiated i.e., the object cannot be created. It may or may not contain abstract methods but if a class has at least one abstract method, it must be declared abstract.

30. Difference between Abstract class and Interface.

Abstract class	Interface
1) Abstract class can **have abstract and non-abstract** methods.	Interface can have **only abstract** methods. Since Java 8, it can have **default and static methods** also.
2) Abstract class **doesn't support multiple inheritance**.	Interface **supports multiple inheritance**.
3) Abstract class **can have final, non-final, static and non-static variables**.	Interface has **only static and final variables**.
4) Abstract class **can provide the implementation of interface**.	Interface **can't provide the implementation of abstract class**.
5) The **abstract keyword** is used to declare abstract class.	The **interface keyword** is used to declare interface.
6) An **abstract class** can extend another Java class and implement multiple Java interfaces.	An **interface** can extend another Java interface only.
7) An **abstract class** can be extended using keyword "extends".	An **interface** can be implemented using keyword "implements".
8) A Java **abstract class** can have class members like private, protected, etc.	Members of a Java interface are public by default.

31. What is Wrapper Class?

- Wrapper classes provide a way to use primitive data types (int, boolean, etc..) as objects. There are 8 types of Wrapper class in java: -

- Boolean- boolean
- Integer- int
- Character- char
- Byte- byte
- Double- double
- Float- float

- Short- short
- Long- long

32. What is typecasting in JAVA?

- In Java, type casting is a method or process that converts a data type into another data type in both ways manually and automatically. The automatic conversion is done by the compiler and manual conversion performed by the programmer.

33. What is access modifiers?

- The access modifiers in Java specifies the accessibility or scope of a field, method, constructor, or class. We can change the access level of fields, constructors, methods, and class by applying the access modifier on it.

There are four types of Java access modifiers:

1. Private: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.
2. Default: The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.
3. Protected: The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.
4. Public: The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

34. What is the difference between Error and Exception?

S.No	Errors	Exceptions
1.	Errors primarily arise due to the lack of system resources.	Exceptions may occur during both runtime and compile time.
2.	Recovery from an error is typically not possible.	Recovery from an exception is possible.
3.	All errors in Java are unchecked.	Exceptions in Java can be either checked or unchecked.
4.	The system executing the program is responsible for errors.	The program's code is responsible for exceptions.
5.	Errors are defined in the java.lang.Error package.	Exceptions are defined in the java.lang.Exception package.

35. Type of exceptions in JAVA.

- Built-in exceptions are the exceptions that are available in Java libraries. These exceptions are suitable to explain certain error situations. Below is the list of important built-in exceptions in Java.

1. **ArithmeticException:** It is thrown when an exceptional condition has occurred in an arithmetic operation.
2. **ArrayIndexOutOfBoundsException:** It is thrown to indicate that an array has been accessed with an illegal index. The index is either negative or greater than or equal to the size of the array.
3. **ClassNotFoundException:** This Exception is raised when we try to access a class whose definition is not found
4. **FileNotFoundException:** This Exception is raised when a file is not accessible or does not open.
5. **IOException:** It is thrown when an input-output operation failed or interrupted
6. **InterruptedException:** It is thrown when a thread is waiting, sleeping, or doing some processing, and it is interrupted.
7. **NoSuchFieldException:** It is thrown when a class does not contain the field (or variable) specified
8. **NoSuchMethodException:** It is thrown when accessing a method that is not found.
9. **NullPointerException:** This exception is raised when referring to the members of a null object. Null represents nothing

10. **+NumberFormatException:** This exception is raised when a method could not convert a string into a numeric format.
11. **RuntimeException:** This represents an exception that occurs during runtime.
12. **StringIndexOutOfBoundsException:** It is thrown by String class methods to indicate that an index is either negative or greater than the size of the string
13. **IllegalArgumentException :** This exception will throw the error or error statement when the method receives an argument which is not accurately fit to the given relation or condition. It comes under the unchecked exception.
14. **IllegalStateException :** This exception will throw an error or error message when the method is not accessed for the particular operation in the application. It comes under the unchecked exception.

36. Difference between throw and throws keyword.

Sr. no.	Basis of Differences	throw	throws
1.	Definition	Java throw keyword is used throw an exception explicitly in the code, inside the function or the block of code.	Java throws keyword is used in the method signature to declare an exception which might be thrown by the function while the execution of the code.
2.	Type of exception Using throw keyword, we can only propagate unchecked exception i.e., the checked exception cannot be propagated using throw only.	Using throws keyword, we can declare both checked and unchecked exceptions. However, the throws keyword can be used to propagate checked exceptions only.	
3.	Syntax	The throw keyword is followed by an instance	The throws keyword is followed by class names of Exceptions to be thrown.

		of Exception to be thrown.	
4.	Declaration	throw is used within the method.	throws is used with the method signature.
5.	Internal implementation	We are allowed to throw only one exception at a time i.e. we cannot throw multiple exceptions.	We can declare multiple exceptions using throws keyword that can be thrown by the method. For example, main() throws IOException, SQLException.

37. What is finally block in JAVA?

- The finally block in java is used to put important codes such as clean up code e.g. closing the file or closing the connection. The finally block executes whether exception rise or not and whether exception handled or not. A finally contains all the crucial statements regardless of the exception occurs or not.

38. What is Collection Interface?

- The **Collection** interface is a member of the Java Collections Framework. It is a part of **java.util** package. It is one of the root interfaces of the Collection Hierarchy. The Collection interface is not directly implemented by any class. However, it is implemented indirectly via its subtypes or subinterfaces like List, Queue, and Set.

39. Difference between Collection and Collections.

- Collection is called interface in java whereas Collections is called a utility class in java and both of them can be found in java.util.package.

- Collection is used to represent a single unit with a group of individual objects whereas collections is used to operate on collection with several utility methods.
- Since java 8, collection is an interface with static as well as abstract and default methods whereas collections operate only with static methods.

40. Path: PATH is an environment variable that is used to find and locate binary files like “java” and “javac” and to locate needed executables from the command line or Terminal window. To set the path, we’re supposed to include or mention JDK_HOME/bin directory in a PATH environment variable. The PATH can not be overridden by providing command and PATH is only used by the operation system(OS) to find binary files.

41. CLassPath: Classpath is an environment variable that is used by the application ClassLoader or system to locate and load the compiled Java bytecodes stored in the .class file. To set CLASSPATH. the CLASSPATH can be overridden by adding classpath in the manifest file and by using a command like set -classpath. the CLASSPATH is only used by Java ClassLoaders to load class files.

42. Explain the method to convert ArrayList to Array and Array to ArrayList.

- Array to ArrayList:-

- Arrays.asList()
- Collections.addAll()
- add() method

ArrayList to Array

- Object[].toArray() - obj.toArray(new String[obj.size()])
- T[].toArray(T[]a)
- get() method

43. How does the size of ArrayList grow dynamically? And also state how it is implemented internally.

- Due to ArrayLists array-based nature, it grows dynamically in size ensuring that there is always enough room for elements. When an ArrayList element is first created, the default capacity is around 10-16 elements which basically depends on the Java version. ArrayList elements are copied over from the

original array to the new array when the capacity of the original array is full. As the ArrayList size increases dynamically, the class creates a new array of bigger sizes and it copies all the elements from the old array to the new array. Now, the reference of the new array is used internally. This process of dynamically growing an array is known as resizing.

44 How to make Java ArrayList Read-Only?

- An ArrayList can be made ready only using the method provided by Collections using the Collections.unmodifiableList() method.

45. What is the Stack class in Java and what are the various methods provided by it?

- A Stack class in Java is a LIFO data structure that implements the Last In First Out data structure. It is derived from a Vector class but has functions specific to stacks. The Stack class in java provides the following methods:

- **peek():** returns the top item from the stack without removing it
- **empty():** returns true if the stack is empty and false otherwise
- **push():** pushes an item onto the top of the stack
- **pop():** removes and returns the top item from the stack
- **search():** returns the 1, based position of the object from the top of the stack. If the object is not in the stack, it returns -1

46. What is Set in the Java Collections framework and list down its various implementations?

- Sets are collections that don't store duplicate elements. They don't keep any order of the elements. The Java Collections framework provides several implementations of the Set interface, including:

- **HashSet:** HashSet in Java, stores the elements in a has table which provides faster lookups and faster insertion. HashSet is not ordered.
- **LinkedHashSet:** LinkedHashSet is an implementation of HashSet which maintains the insertion order of the elements.
- **TreeSet:** TreeSet stores the elements in a sorted order that is determined by the natural ordering of the elements or by a custom comparator provided at the time of creation.

47. What is a Map interface in Java?

- The map interface is present in the Java collection and can be used with Java.util package. A map interface is used for mapping values in the form of a key-value form. The map contains all unique keys. Also, it provides methods associated with it like containsKey(), contains value (), etc.

There are multiple types of maps in the map interface as mentioned below:

1. SortedMap
2. TreeMap
3. HashMap
4. LinkedHashMap

48. Why is ArrayList is better than Array?

- Array is a fixed length data structure whereas ArrayList is a variable length Collection class. We cannot change length of array once created in Java but ArrayList can be changed.

We cannot store primitives in ArrayList, it can only store objects. But array can contain both primitives and objects in Java. Since Java 5, primitives are automatically converted in objects which is known as auto-boxing.

49. Iterator : The Iterator interface provides methods to iterate over any Collection in Java. Iterator is the replacement of Enumeration in the Java Collections Framework. It can get an iterator instance from a Collection using the _iterator()_ method. It also allows the caller to remove elements from the underlying collection during the iteration.

50. Enumeration: Enumeration is a user-defined data type. It is mainly used to assign names to integral constants, the names make a program easy to read and maintain. The main objective of the enum is to define user-defined data types.

51. What is difference between Iterator an Enumeration?

Iterator	Enumeration
Iterator is a universal cursor as it is applicable for all the collection classes.	Enumeration is not a universal cursor as it applies only to legacy classes.
Iterator has the remove() method.	Enumeration does not have the remove() method.
Iterator can do modifications (e.g using remove() method it removes the element from the Collection during traversal).	Enumeration interface acts as a read only interface, one can not do any modifications to Collection while traversing the elements of the Collection.
Iterator is not a legacy interface. Iterator can be used for the traversal of HashMap, LinkedList, ArrayList, HashSet, TreeMap, TreeSet .	Enumeration is a legacy interface which is used for traversing Vector, Hashtable.