150+ Most Asked Java Interview Question & Answers

Q1. What are the main features of Java?

A good way to answer this question would be to list the features and provide a brief explanation and example for each one.

For example: "Java has several key features that makes it a powerful and versatile programming language:

- **Simple:** Java is designed to be easy to learn and use. It removes many of the complexities of C++ and provides an automatic garbage collection feature. For example, you don't have to worry about memory management in Java; it's taken care of by the garbage collector.
- **Object-Oriented:** Everything in Java is an object, which means it can be easily extended. For example, in a car object, the data might be its make, model, color, and the behavior might be its ability to start, stop, accelerate, etc.
- **Platform Independent:** Once you've written a Java program, it can run on any device that has a Java runtime environment. This write-once, run-anywhere principle is one of the most significant advantages of Java.
- **Secure:** Java runs programs inside a sandbox to prevent any activities from untrusted sources, enabling the development of virus-free, tamper-free systems.
- **Robust:** Java emphasizes on compile-time error checking and runtime checking, and has a strong memory management system, making it a robust language.
- **Multithreaded:** Java supports multithreading, which allows multiple tasks to be performed simultaneously. For example, you can listen to music while typing a document in a multithreaded environment.
- **Interpreted:** Java code is translated into bytecode which is then interpreted by the JVM, allowing for increased flexibility and portability.
- **High Performance:** Java uses bytecode, which is close to native code in terms of speed, making it faster than traditional interpreted languages.
- **Distributed:** Java is designed to make distributed computing easy with the networking capability integrated into it. Writing network programs in Java is like sending and receiving data to and from a file.
- **Dynamic:** Java is capable of dynamically linking in new class libraries, methods, and objects, making it more flexible to use.

Q2. What are the major differences between JDK, JRE, and JVM?

JVM, JRE, and JDK are three core components of Java.

- **JVM (Java Virtual Machine):** This is like the engine of a car. It's the part that actually runs your Java programs. It does this by converting the Java bytecode into machine language that your computer can understand and execute.
- **JRE (Java Runtime Environment):** The JRE is like the entire car. It includes the JVM (the engine) plus the Java Class Libraries (all the parts and tools that your Java program needs to run). So, if you just want to drive the car (run a Java program), you need the JRE.
- **JDK (Java Development Kit):** The JDK is like the car factory. It includes everything in the JRE (the car), plus additional tools that you need to build and create new Java programs (like a compiler, debugger, and other tools). So, if you want to build a new car (write a new Java program), you need the JDK.

Q3. Explain the concept of Object-Oriented Programming (OOP).

Object-Oriented Programming (OOP) is a programming paradigm that organizes data into objects and functionality into methods. This approach makes it easier to design, implement, and manage complex systems. Here are the four main principles of OOP:

- **Encapsulation:** This is the principle of hiding the internal details of how an object works and exposing only what's necessary. It's like a car – you just need to know how to operate the pedals and steering wheel, not how the engine works.
- **Inheritance:** This is the principle that allows one class to inherit properties and methods from another class. It's like how a child inherits characteristics from their parents.
- **Polymorphism:** This principle allows one interface to be used for a general class of actions. It's like a remote control – it can be used to control any device that supports its interface, like a TV or a stereo.
- **Abstraction:** This principle involves simplifying complex systems by breaking them down into smaller, more manageable parts. It's like a map – it abstracts the complexity of the geography it represents, making it easier to understand.

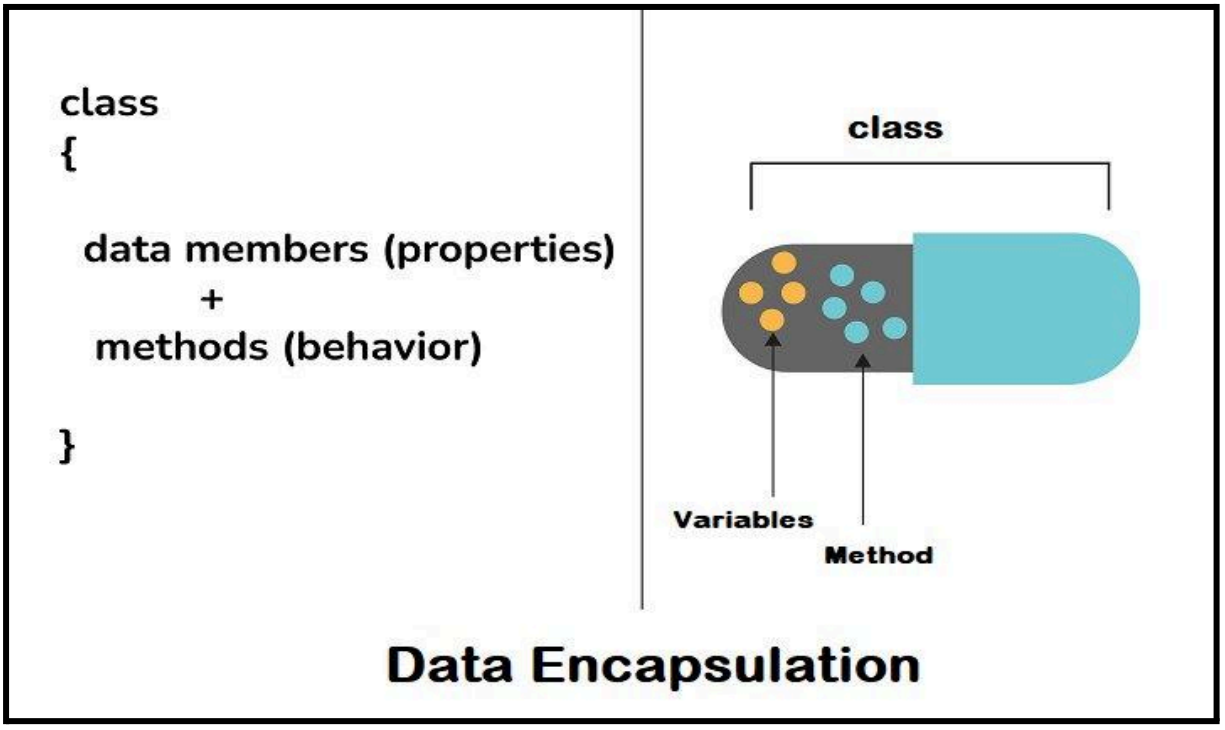
Q4. What is Data Encapsulation? What are the advantages of Encapsulation?

Data Encapsulation, also known as data hiding, is one of the fundamental principles of Object-Oriented Programming (OOP) in Java. It's the technique of making the fields in a class private and providing access to the fields via public methods. If a field is declared private, it cannot be accessed by anyone outside the class, thereby hiding the fields within the class. For this reason, encapsulation is also referred to as data hiding.

The main advantages of data encapsulation are:

- **Control over the data:** We can hide the data from outside access, and the only way to access the data is through the methods we provide, giving us more control over what can be done with our data.
- **Increased Flexibility and Maintainability:** Encapsulation makes our code more flexible and easy to change with new requirements. For instance, if we want to change the data type of a field or enforce new rules on a field, we can do so without affecting the classes that use it.
- **Enhanced Security:** By hiding the complexity of the data, we can protect the data from being altered by external classes.

Think of an analogy as a capsule. Much like a capsule holds multiple medicines together, a Java class encapsulates fields and methods. The outside world is aware only of the methods accessible through the class (similar to the capsule), while remaining unaware of the internal workings (the medicines inside).



Q5. Explain the difference between == and equals?

In Java, == and equals() are both used for comparison, but they work differently.

The == operator compares the references, not the values. It checks to see if two references (or pointers) refer to the exact same object in memory. So, when you compare two objects using ==, it will return true only if they both point to the same object.

On the other hand, equals() is a method that can be overridden in a class to check if two objects are meaningfully equivalent, even if they are not the same object in memory. By default, the equals() method in the Object class behaves the same as the == operator, but classes can override this method to perform a more complex comparison.

For example, when comparing two String objects, the equals() method will return true if the strings have the same sequence of characters, even if they are different objects in memory. The == operator, in this case, would only return true if both references point to the exact same String object.

Q6. What are Constructors in Java?

In Java, a constructor is a special method that is used to initialize an object. It's called when an instance of an object is created, and it has the same name as the class.

There are two types of constructors in Java:

- **Default constructor:** If you don't define a constructor in your class, Java creates one for you. This is the default constructor, and it doesn't take any parameters.
- **Parameterized constructor:** This is a constructor that you define in your class, and it takes one or more parameters. This allows you to initialize your objects with specific values at the time of creation.

An analogy for constructors could be the blueprint of a house. When a house is built, the blueprint defines the structure and initial state of the house – how many rooms it has, the layout, etc. Similarly, a constructor in Java defines how an object should be created and what initial state it should have.

For example, consider a 'Book' class. You might have a constructor that takes a title and an author as parameters. When you create a new Book object, you'd use the constructor to specify the title and author:

```
public class Book {
    String title;
    String author;
   // This is the constructor
    public Book(String title, String author) {
        this.title = title;
        this.author = author;
    }
}
// Creating a new Book object
Book myBook = new Book("1984", "George Orwell");
```

In this example, the constructor is used to initialize the new Book object with the title “1984” and the author “George Orwell”.

Q7. What are the differences between C++ and Java?

Here are the key differences:

- 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.

Q8. What do you understand about JVM?

The Java Virtual Machine (JVM) is a crucial component of the Java platform. It’s an abstract computing machine that enables a computer to run a Java program.

There are three key functionalities of the JVM:

- **Loading of code:** The JVM loads the bytecode of the Java program into its environment to execute it.
- **Verification of code:** The JVM verifies the bytecode to ensure it’s correct and adheres to Java’s syntactic and structural requirements.
- **Execution of code:** The JVM interprets the bytecode and executes the program.

One of the main advantages of JVM is that it allows Java to be platform-independent. The JVM interprets the bytecode into machine code, and this interpretation is specific to the particular operating system and hardware, allowing the same Java program to run on any device that has a JVM.

An analogy for JVM could be a movie projector. Just like a movie projector can take a film reel (the Java bytecode) and project it onto a screen (the computer) so that you can watch a movie, the JVM takes Java bytecode and interprets it into machine code so that your computer can run the program.

For example, when you write and compile a Java program, you get a .class file that contains bytecode. This bytecode is not understandable by your machine directly. Here, JVM comes into play. It interprets this bytecode into machine code, which is then executed by your machine.

This process is the same on every device that has a JVM, which is why you can write a Java program once and run it anywhere.

Q9. What will happen if we write static public void instead of the public static void?

In Java, the order of modifiers like 'public' and 'static' doesn't matter. So, writing 'static public void' instead of 'public static void' won't cause any issues or changes in your program. Both declarations will work the same way.

This is because Java follows a specific set of rules for the order of modifiers, but 'public' and 'static' can appear in any order. The key point is that they should precede the return type of the method, which in this case is 'void'.

An analogy could be putting on a hat and a pair of sunglasses. It doesn't matter whether you put on the hat first or the sunglasses first. The end result is the same: you're wearing both a hat and sunglasses.

For example, consider these two method declarations:

```
public static void myMethod() {
    // method body
}

static public void myMethod() {
    // method body
}
```

Both methods are exactly the same from the Java compiler's perspective.

Q10. Is the constructor inherited in Java?

Ans. Each class in Java has its own constructor, and it's used to create an instance of that specific class. If you create a subclass, it won't inherit the constructor of its superclass. Instead, the subclass will have its own constructor.

If you don't explicitly define a constructor in the subclass, Java will automatically provide a default constructor for you. However, this default constructor won't call the constructor of the superclass unless you explicitly use the super() keyword.

So, while you can call a superclass's constructor from a subclass using super(), the constructor itself is not technically inherited.

Let's say we have a superclass Animal and a subclass Dog.

```
public class Animal {
```

```
    String name;

    // Constructor of Animal class
    public Animal(String name) {
        this.name = name;
    }
}

public class Dog extends Animal {
    String breed;

    // Constructor of Dog class
    public Dog(String name, String breed) {
        super(name);  // Calling the constructor of the
superclass
        this.breed = breed;
    }
}
```

In this example, Dog is a subclass of Animal. The Dog class has its own constructor that takes two parameters: name and breed.

Notice the use of super(name) in the Dog constructor. This is how we call the Animal class's constructor from the Dog class. The super(name) statement must be the first statement in the Dog constructor.

Even though we're calling the Animal constructor from the Dog constructor, the Animal constructor is not technically inherited by the Dog class. Instead, we're just making use of it to initialize the Animal part of each Dog object.

If we create a Dog object like this:

```
Dog myDog = new Dog("Rex", "Labrador");
```

The Dog constructor will be called with "Rex" and "Labrador" as arguments. This constructor then calls the Animal constructor with "Rex" as the argument, setting the name field of the Animal part of the object. Then it sets the breed field of the Dog part of the object.

Q11. Is it possible to make a constructor final in Java?

Ans. No, we cannot make the constructor final in java.

Q12. What are the different types of memory areas allocated by JVM?

Ans. Different types of memory areas allocated by JVM are:

- Stack

- Class
- Program counter register
- Native method stack
- Heap

Q13. Is the JVM platform independent?

Ans. No, JVM is not platform-independent as it is not written in Java.

Q14. What is a Class in Java?

Ans. In Java, a class represents a defined common set of properties and methods to all objects of the same type. Generally, a class includes components like a modifier, class name, superclass, interface, and body. For real-time java applications, several types of classes are used.

Few ways to create a Class in Java:

- New keywords
- forName method
- Clone () method

Q15. What are wrapper classes?

Ans. Wrapper classes are used to convert or wrap Java primitives into reference objects.

Features of java wrapper classes:

- Wrapper classes convert numeric strings into numeric values.
- They are used to store primitive data into the object.
- All wrapper classes use typeValue() method. It returns the object value and its primitive type.
- The wrapper classes use the valueOf() method.

Q16. What is a pointer? Does Java support pointer?

Ans. A pointer helps to directly access the memory location using the address. Java does not have the concept of a pointer because improper handling of pointers results in memory leaks and other related problems. This makes Java a more powerful language than C or C++.

Q17. What is an immutable object?

Ans. An immutable object cannot be modified once created. Software developers rely on immutable objects for creating simple and reliable codes.

Q18. Why is it not possible to override the static method?

Ans. The reason why we cannot override static methods in Java is that overriding is based on dynamic binding at runtime, while the static methods are bonded at compile time using static binding.

Q19. Can you declare the main() method as final?

Ans. Yes, we can declare the main() method as final in Java.

Q20. What are the local variables?

Ans. A variable declared within the body of a method is a local variable. These variables need to be initialized before use.

Syntax

```
// An immutable class

public final class Student
{
final String name;
final int regNo;
public Student(String name, int regNo)
{
this.name = name;
this.regNo = regNo;
}
public String getName()
{
return name;
}
public int getRegNo()
{
return regNo;
// Driver class
class Test
{
public static void main(String args[])
{
Student s = new Student("ABC", 101);
System.out.println(s.getName());
System.out.println(s.getRegNo());
// Uncommenting below line causes error
// s.regNo = 102;
}
}
```

Output:

```
ABC
101
```

Q21. What are instance variables?

Ans. A variable declared inside a class, but outside a method is called an instance variable. These variables don't need to be initialized before use; they are automatically initialized to their default values.

Key features of instance variables:

- They are declared in the class, i.e., outside the method.
- Instance variables are created when the object is created, and it also destroys the purpose.
- Access modifiers are used in the instance variables.
- They are visible to all the class's methods, constructors, and blocks.
- Default values are given to the instance variables. If the value is numeric, it will be '0', and if the value is boolean, it will be 'FALSE.'

Example:

```
class Taxes

{
int count; //Count is an Instance variable
/*…*/
}
```

Q22. What is object cloning?

Ans. Object cloning is a way to create an exact copy of an object. A new instance of the class of the current object is created. All its fields are initialized with exactly the contents of the corresponding fields of this object. Clone() is defined in the object class.

Q23. Can you make an array volatile?

Ans. Yes, you can make an array volatile but only the reference pointing to an array.

Q24. Which operator is considered to be with the highest precedence?

Ans. Postfix operators.

Q25. Can a source file have more than one class declaration?

Ans. Yes, a source file can have more than one class declaration.

Q26. What is the difference between throw and throws?

Ans. A throw triggers an exception, while throws are used in the exception declaration.

Q27. What is the JAR file?

Ans. A JAR (Java Archive) holds java classes in a library. It is a file format based on the ZIP file format. It is used for aggregating many files into one.

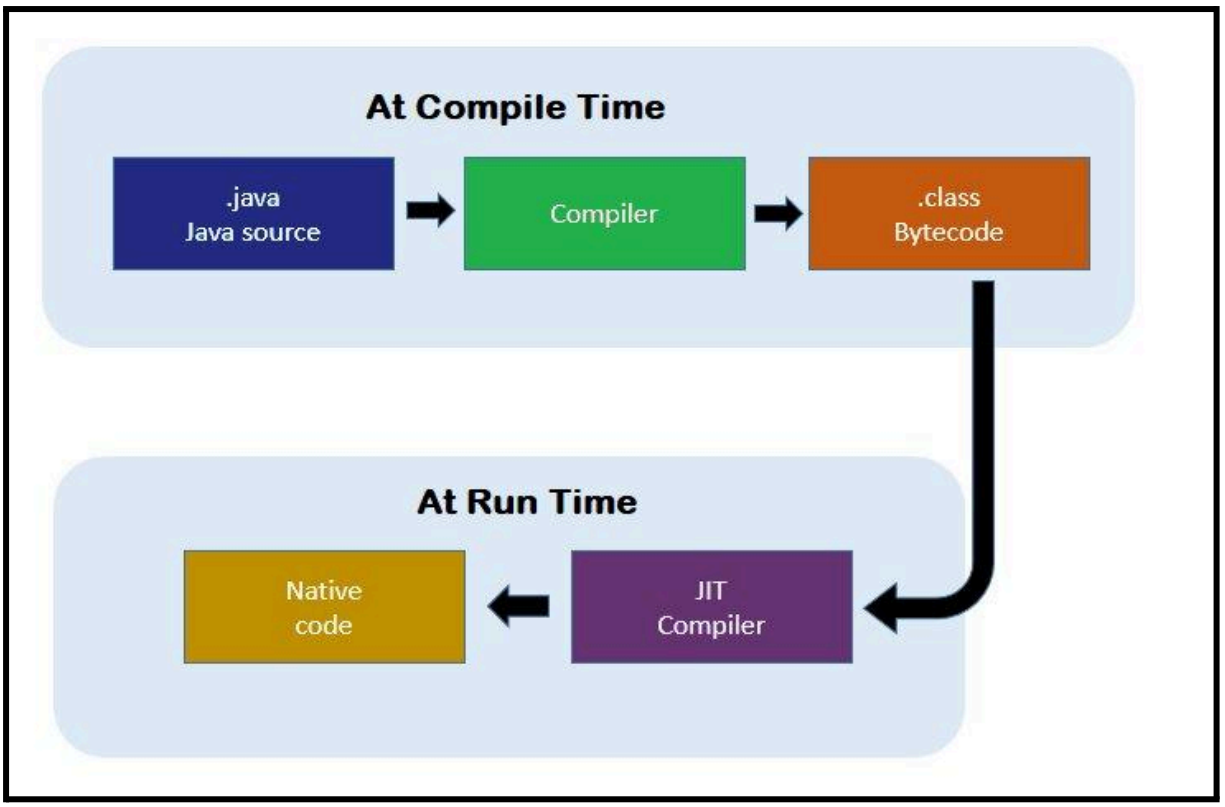
Q28. Define the JIT compiler?

Ans. JIT stands for Just-In-Time Compiler. It optimizes the performance of Java applications at run time or execution time. It does so by compiling bytecode into native machine code that can be sent directly to a computer's processor (CPU). It is enabled by default and is activated when a java method is called. JIT compiler is a part of JVM. When a method has been compiled, the JVM calls the compiled code of that method directly instead of interpreting it.

JIT compilation requires processor time and memory usage. The JIT compilation includes two approaches to translating code into machine code, namely AOT (Ahead-of-Time compilation) and interpretation.

How does JIT work?

- The Java source code (.java) converts to byte code (.class) occurs with the help of the javac compiler.
- The .class files are loaded at run time by JVM. These are converted to machine-understandable code with the help of an interpreter.
- Once the JIT compiler is enabled, the JVM analyzes the method calls in the .class files and compiles them to get more efficient and native code. It also ensures that the prioritized method calls are optimized.
- Now, the JVM executes the optimized code directly instead of interpreting the code again. This aids in improving the performance and speed of the execution.



Q29. Can you have virtual functions in Java?

Ans. By default, every function except static, private, or final is a virtual function in Java.

Q30. Is the Empty .java file name a valid source file name?

Ans. Yes, the Empty .java file name is a valid source file name as Java allows us to save java file by .java only. First, we need to compile it using javac .java and run using java classname.

Example:

```
//save by .java only

class Demo
{
public static void main(String args[])
{
System.out.println("Inforida Online");
}
}
//compile by javac.java
//run by     java Demo
compile it by javac .java
run it by java demo
```

Q31. What is the output of the following Java program?

```
class Naukri

{
public static void main (String args[])
{
System.out.println(20 + 20 + "Javatpoint");
System.out.println("Javatpoint" + 20 + 20);
}
}
```

Output:

```
40Javatpoint
Javatpoint2020
```

Q32. What is the output of the following Java program?

```
class NL
{
public static void main (String args[])
{
for(int i=0; 0; i++)
{
System.out.println("Hello Learners");}}}
```

Ans. Here, the above code will give the compile-time error because there is a need for integer value in the second part of for loop where we are putting '0'.

Q33. Give an example of a ternary operator?

```
Public class conditiontest
 {
Public static void main (string args [])
{
String status;
Int rank;
Status= (rank ==1) "Done": "Pending";
}}
```

Q34. Name the modifiers allowed for methods in an interface?

Ans. Public and abstract modifiers.

Q35. Define a constant variable.

Ans. A constant variable should be declared as final and static.

Example- Static final int MAX_LENGTH=50;

Q36. How can you implement the singleton pattern?

Ans. We can implement a singleton pattern by following common concepts:

- A private static variable of the same class, that is, the only instance of the class
- Private constructor to restrict instantiation of the class from other classes
- Public static method that returns the instance of the class

Q37. Give one difference between the continue and break statement?

Ans. The continue statement is used to end the current loop iteration while when the break statement is used inside a loop, the loop gets terminated and return at the next statement.

Q38. Is it possible to override the overloaded method?

Ans. Yes. The compiler considers the overloaded method as a different method. So, it is possible to override the overloaded method.

Q39. What is a reflection in Java programming? Why is it useful?

Ans. It is used to describe code that is used to inspect other code in the same system and make it dynamic and tied together at runtime.

Example: If you want to call a 'doSomething' method in java on an unknown type object. Java static system does not support this until the object confirms a known interface. If you are using reflection, then it takes your code to the object, and it will find the method 'doSomething.' After that, we can call the method whenever we want.

Method method_N = foo.getClass().getMethod("doSomething", null);

method_N .invoke(foo, null);

Q40. Is it possible to overload the methods by making them static?

Ans. No, it is not possible to overload the methods by just applying the static keyword to them (the number of parameters and types are the same).

Example:

```
public class Demo
{
void consume(int x)
{
System.out.println(x+" consumed!!");
}
static void consume(int x)
{
System.out.println("consumed static "+x);
}
public static void main (String args[])
{
Demo d = new Demo();
x.consume(10);
Demo.consume(20);
}
}
```

Output

```
Demo.java:7: error: method consume(int) is already defined in
class Demo
static void consume(int x)
^
Demo.java:15: error: non-static method consume(int) cannot be
referenced from a static context
Demomenu.consume(20);
^
2 error
```

Q41. What are the types of statements supported by JDBC?

Ans. The different types of statements supported by JDBC:

- Statement: Executes static queries and for general access to the database
- CallableStatement: Offers access to stored procedures and runtime parameters
- PreparedStatement: Provides input parameters to queries when they are executing

Q42. Mention what will be the initial value of an object reference, which is defined as an instance variable?

Ans. In Java, all object references are initialized to null.

Q43. What is the difference between StringBuilder and StringBuffer?

Ans. StringBuffer

- Synchronized means two threads can't call the methods of StringBuffer simultaneously.
- Less proficient than StringBuilder

StringBuilder

- Non-synchronized means two threads can call the threads of StringBuilder simultaneously
- More efficient than StringBuffer

Q44. Which are the four principle concepts upon which object-oriented design and programming rest?

Ans. Following are four principle concepts upon which object-oriented design and programming rest (A-PIE):

- Abstraction
- Polymorphism
- Inheritance
- Encapsulation

Q45. How is the queue implemented in Java?

Ans. A queue is known as the linear data structure similar to the stack data structure. It is an interface available in java.util package and used to store the elements with insertion and deletion operations. In this process, the first element is inserted from the one end called REAR (Tail), and the existing element is deleted from the other end called FRONT (Head). The

whole operation of queue implementation is known as FIFO (First in first out).

Q46. Name the two environment variables that must be set to run any Java program.

Ans. PATH variable and CLASSPATH variable are the two environment variables that must be set to run any Java program.

- The PATH environment variable is used to specify the set of directories that contains execution programs.
- The CLASSPATH environment variable specifies the location of the classes and packages.

Q47. Explain the difference between the Runnable and Callable interface in Java.

Ans. Following are the differences between the Runnable and Callable interface in Java:

- **Runnable** exists since JDK 1.0; **Callable** was added in Java 5.
- **Runnable**'s run() cannot return a value; **Callable**'s call() returns a result.
- **Runnable**'s run() cannot throw checked exceptions; **Callable**'s call() can.
- **Runnable** is executed via Thread or ExecutorService; **Callable** requires ExecutorService.
- **Runnable** is for fire-and-forget tasks; **Callable** is for result-bearing tasks.
- **Runnable** has run(); **Callable** has call().

Example:

```
Runnable r = () -> System.out.println("No return");
Callable<String> c = () -> "Return me";
```

Q48. Explain the difference between the poll() and remove() method of a Queue interface in Java?

Ans. Following are the difference between the poll() and remove() method:

- ➢ **Availability**: Both poll() and remove() are part of java.util.Queue interface.
- ➢ **Function**:
  - poll(): Retrieves *and removes* the head of the queue.
  - remove(): Removes the head of the queue.
- ➢ **Empty Queue Behavior**:
  - poll(): Returns null if queue is empty (safe operation).

- remove(): Throws NoSuchElementException if queue is empty.

➢ **Return Value**:
- poll(): Returns the removed element (type E).
- remove(): In Queue interface, returns the removed element (type E); in Collection interface, returns boolean.

➢ **Use Case**:
- Use poll() when you want to safely handle empty queues.
- Use remove() when the queue should never be empty in normal operation.

Q49. How do you avoid deadlock in Java?

Ans. We can achieve the deadlock in Java by breaking the circular wait condition. For this, we need to arrange the code such that it imposes the ordering on acquisition and release of locks.

Q50. How can you implement to use an Object as a Key in HashMap?

Ans. To use any object as a Key in HashMap or Hashtable, it must implement equals and hashcode method in Java.

Q51. How do you convert bytes to a character in Java?

Ans. This is one of the most commonly asked interview questions on Java. Here is how you can frame the answer.

We can convert bytes to character or text data using character encoding. Incorrect choice of character encoding may change the meaning of the message as it interprets it differently.

Example:

```
public class Conv
{
public static void main(String args[])
{
byte b1 =70;
// char ch = b1;
char ch = (char) b1;
System.out.println("byte value: " + b1);      // prints 70
System.out.println("Converted char value: " + ch);    //
prints A (ASCII is 70 for F)
}
}
```

Output:

```
byte value: 70
Converted char value: F
```

Q52. How do you define Destructors in Java?

Ans. Generally, a destructor is a method that removes an object from the computer's memory. Java lacks a destructor element, and instead, they make use of a garbage collector for resource deallocation.

Q53. What is an Anonymous Class?

Ans. As the name suggests, an Anonymous Class is defined without a name in a single line of code using a new keyword. It is used when one needs to create a class that will be instantiated only once.

Q54. What is the difference between Stack and Queue?

Ans. The stack is based on the Last in First out (LIFO) principle

A queue is based on FIFO (First In, First Out) principle.

Q55. What is a Package?

Ans. A package is a namespace of related classes and interfaces. They are similar to different folders on your computer.

Q56. Which package is imported by default?

Ans. Java.lang package is imported by default.

Q57. How to access Java package from another package?

Ans. We can access the java package from outside the package via three ways –

- import package.*;
- import package.classname;
- fully qualified name

Example:

```
//save by X.java

package p1;

public class X{

public void msg(){System.out.println("Hey!");}
```

```
}
//save by Y.java
package myp1;
class Y{
public static void main(String args[]){
p1.X obj = new p1.X();
obj.msg();
}
}
```

Output:

```
Hey!
```

Q58. What is the default value of the local variable?

Ans. The class variables have default values. Local variables don't have any default value.

Q59. Can a program be executed without main () method?

Ans. One can execute a program without a main method by using a static block.

Q60. What are the various access modifiers for Java sessions

Ans. Access modifiers are the keywords that specify the accessibility of a field, class, method, constructor, and other members.

There are four access modifiers in Java:

- Public: The class, method, or other members defined as Public can be accessed by any class or method.
- Protected: The access level is within the package and outside the package through child class.
- Default: The access level is only within the package.
- Private: This can be accessed within the class only.

Q61. How does Java handle integer overflows and underflows?

Ans. In java , overflow and underflow are handled by using low order bytes that can fit into the size of the type provided by the operation.

Q62. What is the output of the following Java program?

```
class Unit {
```

```
public static void main(String args[]){
final int i;
i = 100;
System.out.println(i);
}
}
```

Output

```
100
```

Q63. What is the difference between an object-based programming language and an object-oriented programming language?

Ans. Object-based programming language doesn't support the features of OOPs except for inheritance. Object-oriented languages are java, C, and C++.

Q64. What is the name of the package used for matching with regular expressions?

Ans. Java.util.regex package

Q65. Name frequently used Java Tools.

Ans. Web developers know that Java is one of the most used commercial grade languages. The frequently used Java tools are:

- JDK (Java Development Kit)
- Eclipse IDE
- NetBeans
- JRat
- Junit

Q66. Explain the implementation of Binary Tree in Java.

Ans. A binary tree is a linear data structure similar to stack, queue, and lists. It consists of two children in a parent node.

```
class Node {

int val;
Node left;
```

```
Node right;
Node(int val) {
this.val = val;
right = null;
left = null;
}
}
```

By the below code, we can add the starting node (Root) of the binary tree:

```
public class BinaryTree {

Node root;
// …
}
```

Q67. What is the Executor Framework in Java? How is it different from the Fork Join Framework?

Ans. The Executor framework is used to manage various threads with the help of a group of components. It is used to run the runnable objects without building new threads and use the existing threads.

Example:

```
public class Test implements Runnable

{
private String message;
public Test(String message)
{
this.message = message;
}
@Override public String run() throws Exception
{
return "Hey " + message + "!";
```

```
}
}
```

Executor Framework is different from the Fork Join Framework:

Fork Join Framework is created to execute ForkJoinTask. It is the lighter version of FutureTask, whereas Executor Framework is created to offer a thread pool that executes the offered task with the use of multiple pooled threads.

There are some ThreadPoolExecutor accessible by JDK API, which is a tread pool of the background thread. Executors.newCachedThreadPool is an unbounded thread used to spawn new threads and reclaim old threads.

Q68. What advantage does the Java layout manager offer over the traditional windowing system?

Ans. Java uses a layout manager to layout components across all windowing platform. They do not have consistent sizing and positioning so that they can provide platform-specific differences among the windowing system.

Q69. Explain map interface in Java programming?

Ans. A map is an object used for mapping between a key and a value. It does not contain a duplicate key, and each key can map to one value. A map interface is not a subtype of the collection interface. So it acts differently from other collection types.

Q70. Which Classes Of Exceptions may be caught by a Catch Clause In Java Programming?

Ans. A catch clause can catch any exception that is assigned to the throwable type, including the error and exception types.

Q71. What do three dots in the method parameters mean? What do the three dots in the following method mean?

```
public void Method_N(String… strings){

// method body
}
```

Ans. The given code describes that it can receive multiple String arguments.

Syntax:

```
Method_N("foo", "bar");

Method_N("foo", "bar", "baz");
Method_N(new String[]{"foo", "var", "baz"});
We can use the String var as an array:
public void Method_N(String… strings){
for(String whatever: strings){
// do whatever you want
}
// the code above is is equivalent to
for( int j = 0; i < strings.length; j++){
// classical for. In this case, you use strings[i]
}
}
```

Q72. Name the Container Method used to cause Container to be laid out and redisplayed in Java Programming?

Ans. 'validate()' Container method is used to cause a container to be laid out and redisplayed in Java Programming.

Q73. What is a classloader in Java?

Ans. Classloader is a class in java that is used to load other class files from the network, file system, and other sources. Java code in classes is compiled by the javac compiler and executed by JVM. Java stores three built-in classloaders:

- Bootstrap ClassLoader
- Extension ClassLoader
- System/Application ClassLoader

Q74. What is runtime polymorphism or dynamic method dispatch?

Ans. Runtime polymorphism or dynamic method dispatch is the process in which the overridden methods are resolved at runtime, not at compile time.

```
class Car
{
```

```
 void run()
{
System.out.println("car is running");
} }
class Audi extends Car {
void run()
{
System.out.prinltn("Audi is running safely with 100km");
}
public static void main(String args[])
{
Car b= new Audi(); //upcasting
b.run();
} }
```

Q75. Does a class inherit constructors of its superclass in Java programming?

Ans. No, A class does not inherit constructs of its superclass.

Q76. How are the elements of a Gridbaglayout arranged in Java programming?

Ans. In Gridbaglayout, elements are arranged in the form of a grid, whereas elements are present in different sizes and occupy the space according to their sizes. It can cover one or more rows and columns of various sizes.

Q77. What are the common data structures and algorithms used in Java?

Ans. The following most common data structures and algorithms used in Java:

Linear Data Structures

- Arrays
- Linked List
- Stacks
- Queues

Hierarchical Data Structures

- Binary Trees
- Heaps
- Hash Tables

Q78. What do you mean by loops? What are the three types of loops in Java?

Ans. In Java, Loops execute a set of statements or a block repeatedly until a particular condition is satisfied. The three types of Loops in Java are:

For Loops

For loops execute statements repeatedly for a given number of times. These are used when the number of times to execute the statements is known to the programmer

```
for (initialization; testing condition; increment/decrement)
{
statement(s)
}
```

While Loops

It is used when certain statements need to be executed repeatedly until a condition is fulfilled. The condition is checked first before the execution of statements.

```
while (boolean condition)

{
loop statements…
}
```

Do While Loops

It is the same as the While loop with a difference that condition is checked after execution of a block of statements. Thus, statements are executed at least once.

```
do

{
statements.
}
```

```
while (condition);
```

Q79. How to generate random numbers in Java?

Ans. We can generate random numbers in Java using the following:

- random method
- util.Random class
- ThreadLocalRandom class

Q80. What is the significance of Java packages?

Ans. A Java package is a set of classes and interfaces that are bundled together in a way that they are related to each other. Java Packages provide multiple benefits, such as protection against name collisions; help organize source code; hide implementation for multiple classes; make searching/locating and usage of classes, control access, and annotations easy.

Q81. What is an infinite loop?

Ans. The infinite loop is an instruction in which the loop has no exit function, so it repeats endlessly. The infinite loop will terminate automatically when the application ends.

Example:

```
public class InfiniteLoop

{
public static void main(String[] arg) {
for(;;)
System.out.println("Hello Guys");
}
}
```

Q82. What is the output of the following java code?

```
public

class Demo {
public static void main(String[] args)
```

```
{
int a = 10;
if (a) {
System.out.println("HELLO STUDENTS");
} else {
System.out.println("BYE");
}}}
```

Output: `Compile time error`

If statement should have the argument of boolean type

Q83. How can you prevent a method from being overridden?

Ans. Use the final modifier on the method declaration to prevent a method from being overridden.

```
Public final void examplemethod ()
{
// method statements
}
```

Q84. Give an example of a class variable declaration?

Ans. Class variables are declared with static modifiers.

```
Public class product
{
Public static int Barcode;
}
```

Q85. What environment variables are required to run java programs?

Ans. PATH and CLASSPATH, JAVA_HOME, and JRE_HOME are needed to run a simple java application and used to find JDK binaries that are used to run java programs on platforms like Windows and Linux.

Environment variables are used in the programs to know which directory is used to install files and where to store installed files.

Q86. What does JAXB stand for?

Ans. JAXB means Java API for XML binding. It is a fast and suitable way to bind Java representation to incorporate XML data in Java applications.

Q87. What are the new features for Java 8?

Ans. Java 8 is full of really interesting features at both the language level and the JVM level. Some of the features that are an absolute must to know about are:-

- Parallel operations
- Concurrent accumulators
- Lambda Expressions
- Generic type changes and improvements
- Functional interfaces

Q88. What type of variables can a class consist of?

Ans. A class comprises of an instance variable, class variable, and local variable.

Q89. What are the differences between Heap and Stack Memory in Java?

Ans. The differences between heap and stack memory:

Stack Memory

- Linear data structure (LIFO - Last In, First Out).
- Used by **only one thread** of execution.
- **Faster access** (memory allocation is quick).
- Stores **local variables**, method calls, and references.
- **Fixed size** (depends on OS, can lead to StackOverflowError if exceeded).
- **Variables cannot be resized**.
- Memory is allocated in a **contiguous block**.
- Automatically freed when method execution completes (LIFO-based deallocation).

Heap Memory

- Hierarchical (tree-based) data structure.
- Shared by **all parts** of the application.
- **Slower access** compared to stack.
- Stores **objects and global variables**.
- **No fixed size limit** (limited by system memory, can lead to OutOfMemoryError).
- **Variables can be resized** (dynamic memory allocation).
- Memory is allocated in **random order**.

- Managed by **Garbage Collector (GC)** (objects removed when no longer referenced).

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 will be available for use by the java program when it is required, mostly during the program's runtime. **Java Utilizes this memory as –**

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

Example- Consider the below java program:

```
class Main {
public void printArray(int[] array){
for(int i : array)
System.out.println(i);
}
public static void main(String args[]) {
int[] array = new int[10];
printArray(array);
}
}
```

Q90. Explain the final keyword in Java.

Ans. final is used as a non-access modifier to restrict the user. A final variable can be used in several contexts, such as variable, method, and class.

➢ final variable

When the final keyword is used with a variable then its value cannot be changed once assigned. It will be constant.

➢ final method

When a method is declared final, then it cannot be overridden or hidden by subclasses.

➢ final class

When a class is declared as final, it cannot be extended (inherited).

OOPs – Constructor Interview Questions

Q91. What is a constructor?

Ans. A constructor is a block of code that enables you to create an object of the class. It is called automatically when a new instance of an object is created.

Q92. What is the use of the default constructor?

Ans. A constructor is used to initialize the object. E.g.:

```
Class Bike1

{
Bike1()
}
System.out.println("Bike is created");
}
Public static void main(string args[])
{
Bike1 b= new Bike1();
}
}
```

Q93. Name some methods of an object class.

Ans. Methods of an object class are:

- clone(): Create and returns a copy of an object
- equals(): Compares the equality of two objects
- hashCode(): Returns a hash code value for the object
- getClass(): Returns the runtime class of the object
- finalize(): It is called by the garbage collector on the object
- toString(): Returns a string representation of the object
- notify(), notifyAll(), and wait(): Synchronize the activities of the independently running threads in a program

Q94. Why do we use the default constructor?

Ans. The default constructor is used to assign the default value to the objects. If there is no constructor in the class then the java compiler will automatically create a default constructor.

Example 1:

```
class Employee

{
int id;
String name;
void display(){System.out.println(id+" "+name);}
public static void main(String args[])
Employee e1=new Employee();
Employee e2=new Employee();
e1.display();
e2.display();
}
}
```

Output:

```
0 null
0 null
```

Reason: In the given code there is no constructor, so the compiler will automatically provide a default constructor. Where the default constructor provides the null values.

Example 2:

```
class Learner
{
int id;
String name;
void display(){System.out.println(id+" "+name);}
public static void main(String args[])
{
Learner l1=new Learner();
Learner l2=new Learner();
l1.display();
l2.display();
}
}
```

Output:

```
0 null
0 null
```

Q95. What is the output of the following Java program?

```
public class Test


{  Test(int x, int y)  {  System.out.println("x = "+x+" y =
"+y);  }  Test(int x, float y)  {  System.out.println("x =
"+x+" y = "+y);  }  public static void main (String args[])

{

byte x = 10;

byte y = 15;

Test test = new Test(x,y);

}

}
```

Output:

```
X = 10 y = 15
```

In this program, the data type of the variables x and y, i.e., byte gets promoted to int, and the first parameterized constructor with the two integer parameters is called.

Q96. What is the output of the following Java program?

```
class Naukri
{

int a;

}

class Main

{

public static void main (String args[])

{

Naukri naukri = new Naukri();

System.out.println(naukri.a);

}
}
```

Output: 0

Here, the variable a is initialized to 0 internally where a default constructor is implied in the class, the variable i is initialized to 0 since there is no constructor in the class.

Q97. Is it possible to initialize the final blank variable?

Ans. Yes, it is possible to initialize the final blank variable in the constructor, if it is not static. If it is a static blank final variable, it can be initialized only in the static block.

Q98. What is Polymorphism?

Ans. Polymorphism, an ability to perform single tasks in multiple ways, in Java is divided into two parts – Compile-time and Runtime polymorphism.

- Compile-time also called a static method is a process wherein the call during the overloading method is resolved at the same time of compilation.
- Runtime polymorphism also known as dynamic dispatch has a provision to support the overriding of methods. This is done during the runtime.

Q99. Mention the differences between the constructors and methods?

Ans. Following are the difference between constructors and methods:

Constructors

- Used to **initialize an object** in the class.
- **No return type** (not even void).
- Java compiler provides a **default constructor** if none is defined.
- **Name must match the class name**.
- Cannot be **static**, **final**, or **abstract**.
- Automatically called when an object is created (new keyword).

Methods

- Used to **expose the behavior** (actions) of an object.
- **Must have a return type** (void if no return value).
- **Not provided by the compiler**—must be explicitly defined.
- Name **can differ** from the class name.
- Can be **static**, **final**, or **abstract**.
- Called explicitly using object reference (e.g., obj.method()).

Q100. What is the output of the following Java program?

```
class Men
{
public Men()
{
System.out.println("Men class constructor called");
}
}
public class Child extends Men
{
public Child()
{
System.out.println("Child class constructor called");
}
public static void main (String args[])
{
Child c = new Child();
}
}
```

Output:

```
Men class constructor called
Child class constructor called
```

Q101. What is a Java copy constructor?

Ans. Java has a special type of constructor called the Copy Constructor in Java that is used for delivering a copy of specified objects. This is mostly used when a coder wants to copy something heavy to instantiate. Also, it is recommended that to detach both objects, use the deep copy. Further, it also offers full control over object creation.

Q102. Define the advantage of Copy constructor over Object.clone().

Ans. Java experts pick copy constructor over the Object.clone() because the copy constructor does not push to implement any specific interface but one can implement it as and when required. Similarly, it also allows modification in final fields.

Spring Interview Questions

Q103. Please introduce Java Spring Framework.

Ans. Frameworks is a large body of pre-existing codes. It is fast, efficient, and light-weighted. It is used to solve the coder's specific problems very quickly. Java being a fast-paced programming language has widely used frameworks like JSF, Struts, and play!.

Among these, the Spring framework is one of the most powerful and light-weighted frameworks used for Enterprise Java (JEE). Spring is widely used for developing Java-based applications.

Features of Java Spring Framework

- Extensible XML configuration
- IoC container containing assembler code handling configuration management
- Spring MVC framework
- Support various dynamic languages
- Supports recognized system modules
- Compatible with various versions of Java
- Simple to use, testable,
- Loose coupling because of concepts like AOP

Q104. Define the Spring annotations that you use in projects.

Ans. Widely used Spring Annotations:

- @Controller
- @RequestMapping
- @Qualifier
- @Scope

Servlet Interview Questions

Q105. What is a Java servlet?

Ans. Java developers use Servlets when they need to outspread the capabilities of a server. Servlets are used to extend the applications that are hosted by a web server and it is deployed to design a dynamic web page as well by using Java. These servlets run on JVM and resist attacks. Unlike CGI (Common Gateway Interface) the servlets are portable.

Q106. When is the servlet object is created?

Ans. The Servlet object is created at the time of the first request.

Q107. Define the steps of creating a servlet in Java.

Ans. Servlet follows a lifecycle having four stages:

- Loading
- Initializing
- Request Handling
- Servlet destroying

To go ahead with the process, the first and foremost thing is to create a servlet. Below is the process:

- Develop a structured directory
- Create the servlet
- Compile
- Add mappings into the web.xml file
- Deploy the project by initiating server
- Access it

Q108. What is a JSP in Java and are they better than servlets?

Ans. Java Server Pages (JSP) is a server-side technology that is used to create applications, dynamic web content, and independent web pages.

Yes, JSP is a better technology than servlets as they are very easy to maintain. Also, it doesn't require any sort of recompilation or redeployment. JSP, being an extension to servlets, majorly covers most of its features.

Q109. For multiple requests, how many objects of a servlet are created?

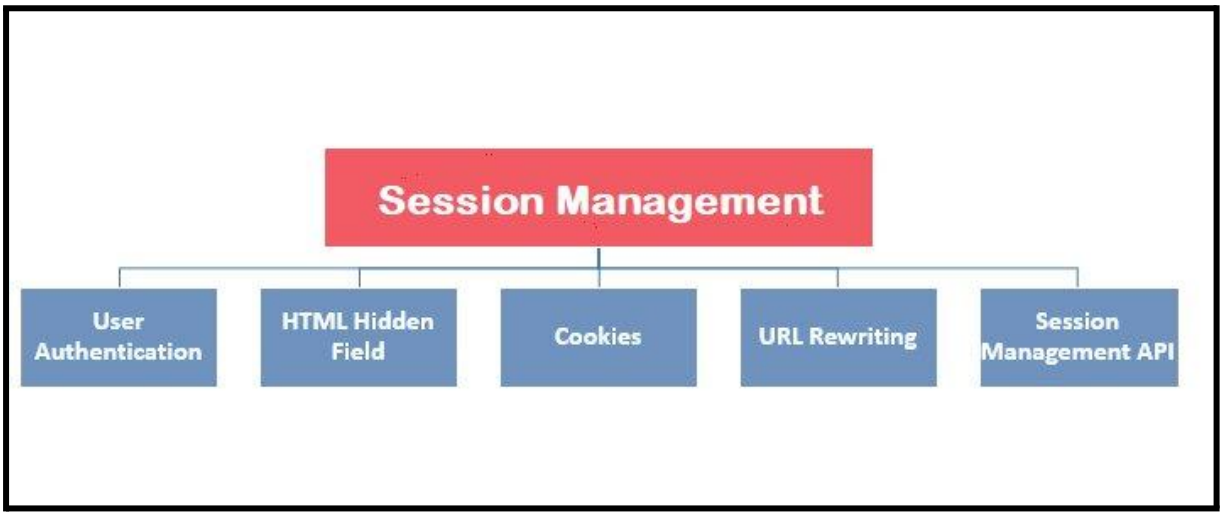
Ans. The Servlet creates only one object at the time of the first request, irrespective of how many incoming requests it receives.

Q110. What are the different methods of session management in Servlets?

Ans. There are many ways to manage sessions in Java web applications written using Servlets. Some of the common ways are:

- **User Authentication –** In this, a user can provide authentication credentials from the login page. Then, we can pass the authentication information between server and client to maintain the session.
- **HTML Hidden Field –** This allows us to create a unique hidden field in the HTML and when the user starts navigating. We can set its value unique to the user and keep track of the session.

- **Cookies –** Cookies are small pieces of information sent by a web server in response header and it gets stored in the browser cookies. When a further request is made by the client, it adds the cookie to the request header and it can be utilized to keep track of the session.
- **URL Rewriting –** In this, we can append a session identifier parameter with every request and response to keep track of the session.
- **Session Management API –** It is built on top of the above methods for session tracking.



Java Collections Interview Questions

Q111. Explain the Collection Framework in Java.

Ans. The Java Collection framework provides an architecture to store and manage a group of objects. It defines several algorithms that can be applied to collections and maps. It enables the developers to access prepackaged data structures as well as algorithms to manipulate data.

The Java Collection Framework includes:

- Interfaces
- Classes
- Algorithm

Q112. What are the benefits of the Java Collection Framework?

Ans. The following are the benefits of the Java Collection Framework:

- Provides useful data structures and algorithms to reduce programming effort

- Increases the speed and quality of the program
- Promotes interoperability among unrelated APIs
- Reduces the effort required to learn, use, and design new APIs
- Encourages reusability of software

Q113. What is a Linked List? What are its different types?

Ans. A linked list is a linear data structure in which each element is considered as a separate object or entity in itself. Every element within a list consists of the data and the reference to the next node.

The different types of Linked Lists are:

- Singly Linked List: Each node stores two pieces of information – the address of the next node and the data.
- Doubly Linked List: Each node has two references – reference to the next node and the previous node.
- Circular Linked List: All nodes are connected to each other. It can be singly circular or doubly circular.

Q114. How will you convert an ArrayList to Array and an Array to ArrayList?

Ans. We can convert an Array into an ArrayList by using the asList() method.

Syntax:

```
Arrays.asList(item)
```

An ArrayList can be converted into Array using the toArray() method.

Syntax:

```
List_object.toArray(new String[List_object.size()])
```

Q115. What is the emptySet() method in the Java Collections framework?

Ans. The Collections.emptySet() returns the empty (immutable) Set whenever the programmer attempts to remove null elements. It returns a serializable Set. The syntax of emptySet() is as follows:

Syntax:

```
public static final <T> Set<T> emptySet()
```

Q116. What is the difference between Concurrent Collections and Synchronized Collections?

Ans. Concurrent collection and synchronized collection are used for thread safety. The difference between them is how they achieve thread-safety by their scalability, performance. The performance of concurrent collection is better than the synchronized collection as it holds the single portion of the map to attain concurrency.

Q117. What is CopyOnWriteArrayList? How is it different from ArrayList and Vector?

Ans. CopyOnWriteArrayList is used to implement a list interface in the array list. It is a thread-safe variant implemented for creating a new copy of the existing arrays.

ArrayList has a synchronized collection, whereas CopyOnWriteArrayList has a concurrent collection

Java Interview Questions on Static Keyword

Q118. What is a static keyword?

Ans. A Static Keyword is used for memory management to refer to the common property of all objects. If we apply the java static keyword with any method, it is called the static method. Static can be used to class nested with another class, initialization block, process, and variable.

Example:

```
class Demo

{
static int x = 10;
static int y;
// static block
static {
System.out.println("Static block initialized.");
y = x * 4;
}
public static void main(String[] args)
{
System.out.println("from main");
```

```
System.out.println("Value of x : "+x);
System.out.println("Value of y : "+y);
}
}
```

Output:

```
Static block initialized.
from main
Value of x: 10
Value of y: 40
```

Q119. Define the static method?

Ans. Following are the pointers that define the static method:

- The static method is used to access and change the value of the static variable.
- No object is required to call the static methods.
- A static method is a part of a class, not an object.

Example:

```
class Demo

{
// static variable
static int x = 10;
static int y;
// static block
static {
System.out.println("Static block initialized.");
y = x * 4;
}
public static void main(String[] args)
{
System.out.println("from main");
System.out.println("Value of x : "+x);
```

```
System.out.println("Value of y : "+y);
}
}
```

Output:

```
Static block initialized.
from main
Value of x: 10
Value of y: 40
```

Q120. Can the non-static variable be accessed in a static context?

Ans. No, the non-static variable cannot be accessed in a static context.

Q121. Can you override the static method?

Ans. No, a static method cannot be overridden because they are resolved in compile-time and not runtime.

Q122. Explain what the static block is?

Ans. A static block is used to initialize the static data member in the program. It is executed before the main method.

```
class Test
{
static{System.out.println("static block is invoked");
}
public static void main(String args[])
{
System.out.println("Hello Learners");
}
}
```

Output:

```
static block is invoked
Hello Learners
```

Q123. Is it possible to execute a program without main() method?

Ans. Yes, it is possible to execute a program without main() method where one can execute the program using a static block.

Q124. What if the static modifier is removed from the signature of the main method?

Ans. If the static modifier is removed from the signature of the main method, then the program compiles. But, at runtime, it throws an error "NoSuchMethodError."

Q125. Is it possible to declare the static variables and methods in an abstract class?

Ans. Yes, It is possible to declare static variables and methods in an abstract method. There is no need to make the object access the static context. We can access the static context declared inside the abstract class by using the name of the abstract class.

Example:

```
abstract class Learning
{
static int i = 2020;
static void LearningMethod()
{
System.out.println("Naukri");
}
}
public class LearningClass extends Learning
{
public static void main (String args[])
{
Learning.LearningMethod();
System.out.println("i = "+Learning.i);
}
}
```

Output

```
Naukri
i = 2020
```

Q126. What is an abstract method?

Ans. It is a function that is declared without an implementation. It lies in the heart of inheritance when you define certain tasks functions that are dealt with within the child's classes differently.

Syntax

```
modifier abstract class demo {

//declare fields
//declare methods
abstract dataType methodName();
}
modifier class childClass extends demo {
dataType methodName(){}
}
```

Q127. What is the abstract class in Java? Is it similar or different when compared to C++?

Ans. The abstract class generally contains one or more than one abstract methods, and these abstract methods are declared, but they do not contain implementation. In Java, to make a class abstract, a separate keyword is required 'abstract.' But in C++, if any class comprises at least one pure vital function, then the class automatically becomes abstract.

Example:

```
//Declare class using abstract keyword

abstract class X{
//Define abstract method
abstract void myMed();
//This is concrete method with body
void anotherMed(){
//Does something
}
}
```

Q128. When should we use abstract classes?

Ans. We use interfaces when something in our design changes frequently.

Q129. What is the base class of all classes?

Ans. Java.lang.object

Java Inheritance Interview Questions

Q130. How "this" keyword is used in the class?

Ans. Following are the uses of this keyword in the class:

- Can be used as an argument in the method call.
- It is used as an argument in the constructor call.
- It is used to refer to the existing class instance variable.
- Can be used to assign current class methods.
- It can be used to return the existing class instance from the method.

Q131. Do we use this keyword to refer to static members?

Ans. Yes, we can use this keyword to refer to static members because this is used as a reference variable that refers to the current class object.

Example:

```
public class Demo
{
static int i = 50;
public Demo ()
{
System.out.println(this.i);
}
public static void main (String args[])
{
Demo d = new Demo();
}}
```

Output:

```
50
```

Q132. What do you mean by Super? What are the main uses of the super keyword?

Ans. The super keyword in Java is used to refer to the immediate parent class object as it is a reference variable. When there is any instance created of the subclass, then the instance of the parent class is created implicitly, which is referred to by a super reference variable.

Super () keyword is used to call the main methods and also access the superclass constructor. It is also used to differentiate between the methods with the same name of superclasses and subclasses. The following are the uses of the super keyword:

- The super keyword is used as the immediate parent class instance variable.
- It can be used to instance the immediate parent class method.
- Super() can be used to request a parent class constructor.

```
class Student
{
Student()
{
System.out.println("Student is created");}
}
class Jack extends Student{
Jack(){
System.out.println("Jack is created");
}
}
class TestSuper4{
public static void main(String args[]){
Jack j=new Jack();
}
}
```

Output:

```
Student is created
Jack is created
```

Q133. Name the class considered as the superclass for all the classes?

Ans. The object class is considered as the superclass of all other classes in Java.

Q134. Why Java doesn't support multiple inheritances yet multiple interface implementation is possible?

Ans. Multiple inheritances refer to the process by which one inherits the properties and behavior of multiple classes into one single class like in C++. But Java being a simple, robust, and secure language, it omits multiple inheritances usually called the Diamond Problem.

In Java 8, it supports default methods and that's how a class can go ahead with the implementation of two or more interfaces.

Q135. How can you do constructor chaining using this keyword?

Ans. Constructor chaining is used to call one constructor from another constructor of the class as per the current class object. This keyword is used to perform constructor chaining within the same class.

Example:

```
public class Student

{
int id,age;
String name, address;
public Student (int age)
{
this.age = age;
}
public Student(int id, int age)
{
this(age);
this.id = id;
}
public Student(int id, int age, String name, String address)
```

```
{
this(id, age);
this.name = name;
this.address = address;
}
public static void main (String args[])
{
Student stu = new Student (10121, 22, "Ron", "Boston");
System.out.println("ID: "+stu.id+" Name:"+stu.name+"
age:"+stu.age+" address: "+stu.address);
}
}
```

Output

```
ID: 10121 Name:Ron age:22 address: Boston
```

Q136. Is it possible to use this() and super() both in a constructor?

Ans. No, because this() and super() must be the first statement in the class constructor.

Example:

```
public class Demo
{
Demo()
{
super();
this();
System.out.println("Demo class object is created");
}
public static void main(String []args){
Demo d = new Demo();
}
}
```

Output:

```
Demo.java:5: error: the call to this must be the first
statement in the constructor
```

Q137. How to restrict inheritance for a class?

Ans. We can restrict inheritance for a class by:

- Using the final keyword
- Making a constructor private
- Using the Javadoc comment (" // ")
- Making all methods final, so that we cannot override them

Q138. Explain the difference between aggregation and composition in java?

Ans. The aggregation offers a weak relationship in which both objects can exist independently whereas composition allows a strong relationship in which if an object owns another object and that object cannot exist without the owner object then it is said to be composition.

Example of Composition

```
public class Bike {

private final Engine engine;  //engine is a mandatory part of
the Bike
public Bike () {
engine = new Engine();
}
}
class Engine {}
```

Example of Aggregation

```
public class Group {

private List student;  //students can be 0 or more
public Bike () {
players = new ArrayList();
}
}
class Student {}
```

Method Overriding Interview Questions

Q139. What is the difference between Method Overloading and Overriding?

Ans. The differences between Method Overloading and Overriding are:

Method Overloading

✅ **Same method name**, different parameters
✅ **Compile-time polymorphism** (static binding)
✅ Works **within the same class**
✅ **Return type can differ**
✅ **Static methods can be overloaded**
✅ Improves **code readability**
✅ **No inheritance required**
✅ **Access modifiers can vary**

Method Overriding

✅ **Same method name & parameters** as superclass
✅ **Runtime polymorphism** (dynamic binding)
✅ Requires **inheritance** (subclass overrides superclass)
✅ **Return type must be same** (or covariant)
❌ **Static methods cannot be overridden** (only hidden)
✅ Enhances **code reusability**
❌ **Access modifier cannot be more restrictive**
✅ Used for **specific implementation**

Q140. Is it possible to override private methods?

Ans. No, it is not possible to override the private methods because the scope of private methods is limited to the class, and we cannot access these methods outside of the class.

Q141. Explain what is the covariant return type?

Ans. In Java, it is possible to override any method by altering the return type if the return type of the subclass overriding method is the subclass type, which is known as a covariant return type. The covariant return type defines that the return type may vary in the same direction as the subclass.

```
class Test
```

```
{
Test get(){return this;}
}
class Test1 extends Test{
Test1 get(){return this;}
void message(){System.out.println("Welcome to the covariant return type");}
public static void main(String args[]){
new Test1().get().message();
}
}
```

Output:

```
Welcome to the covariant return type
```

Q142. What do you mean by method overriding?

Ans. If the child class has the same method as declared in the parent class, this is called method overriding. It allows a subclass or child class to provide a specific implementation of a method that is already provided by one of its super-classes or parent classes.
Program to illustrate Java Method Overriding

```
class Vehicle {
   public void displayInfo() {
      System.out.println("It is a Vehicle.");
   }
}
class Car extends Vehicle {
   @Override
   public void displayInfo() {
      System.out.println("It is a car.");
   }
}
class Main {
   public static void main(String[] args) {
```

```
        Car c1 = new Car();
        c1.displayInfo();
    }
}
```

Output:

```
It is a car.
```

Exception and Thread Java Interview Questions

Q143. What do you understand by thread-safe?

Ans. It is simply the program that behaves correctly when multiple simultaneous threads are using a resource.

Q144. Which Swing methods are thread-safe?

Ans. The thread-safe Swing methods are:

- Repaint
- Revalidate
- Invalidate

Q145. Explain the different states of a thread.

Ans. The different states of a thread are:

- Ready State: A thread is created.
- Running State: Thread currently being executed.
- Waiting State: A thread waiting for another thread to free certain resources.
- Dead State: A thread that has gone dead after execution.

Q146. Explain multithreading in Java.

Ans. Multithreading is a programming feature that allows multiple tasks to run simultaneously within a single program. In simple terms, it enables concurrent execution of two or more parts of a program for maximum utilization of CPU. Each program is called a thread. In Java, threads are light-weight processes within a process. They run in parallel and help in the performance improvement of any program.

Q147. How can you implement multithreading in Java?

Ans. We can implement multithreading in Java by using the following:

- By implementing the Runnableinterface.
- By writing a class that extends Java.Lang.Thread class.

Q148. What is thread-safety in Java? How do you achieve it?

Ans. Java provides a multi-threaded environment, multiple threads are created from the same object share object variable, and it will cause data inconsistency when threads are used to update and read shared data. JVM helps in improving the performance of applications by running bytecode in different worker threads.

Different methodologies to achieve thread-safety:

- Stateless implements
- Synchronized collections
- Thread-Local Fields
- Atomic objects
- Concurrent collections
- Immutable implementations

Q149. What is the objective of the Wait(), Notify(), And Notifyall() methods in Java programming?

Ans. In Java programming, the wait(), notify(), and notifyAll() methods are used to provide a systematic way for threads to communicate with each other.

Q150. What is Livelock?

Ans. Livelock is a concurrency problem in Java in which the state of threads changes between one another without making any progress. The execution of threads stops because of the unavailability of resources.

Q151. What is synchronization in Java?

Ans. Synchronization is used to control the access of multiple threads in the shared resources to provide better results. It also helps to prevent thread interference and solve consistent problems.

There are two types of synchronization:

- Process Synchronization
- Thread Synchronization

Example:

```
public class Syx
{
private static int c1 = 0;
```

```
public static synchronized int getCount(){
return c1;
}
public synchoronized setCount(int c1){
this.c1 = c1;
}
}
```

Q152. When throws keyword can be used?

Ans. Providing information to the programmer that there may happen an exception so that the normal flow can be maintained, Throws keyword is used to declare an exception.

Syntax of java throws

```
return_type method_demo() throws exception_class_demo{

//method code
}
```

Q153. Explain NullPointerException.

Ans. NullPointerException in Java is a RuntimeException. A special null value can be assigned to an object reference in Java. NullPointerException is thrown when a program attempts an object reference with the null value.

Null Pointer Exception is thrown in specific scenarios like:

- Method invoked using a null object.
- Accessing or modifying a null object's field.
- When a null object is passed as an argument to a method
- Taking the length of a null array.
- Accessing or modifying the slots of a null array.
- Throwing a null object.
- Synchronizing a null object.

Q154. How can you avoid NullPointerException in Java?

Ans. We can avoid NullPointerException by the following:

- Use valueOf() over toString() where both return same result
- Avoid unwanted autoboxing and unboxing in the code
- Use null safe methods and libraries

- Avoid returning null from a method
- Use apache commons StringUtils for String operations
- Avoid passing of Null Parameters

Q155. How to avoid ConcurrentModificationException?

Ans. We can avoid the Exception by using Iterator and call remove():

Iterator<String> iter1 = myArrayList.iterator();

while (iter1.hasNext()) {

String str = iter1.next();

if (someCondition) iter1.remove();

}

Q156. What is the purpose of garbage collection in Java, and when is it used?

Ans. Garbage collection is an automatic process. The memory is allocated to all the java programs. At the run time, there are some objects created at the heap and take the memory space. The garbage collectors delete these unwanted objects.

Serialization Interview Questions

Q157. What is serialization in Java?

Ans. In this process, objects are converted into byte codes to store the object in memory space. The main task is to save the state of the object in order.

```
class X implements Serializable{

// Y also implements Serializable
// interface
Y ob=new Y();
}
```

Q158. When we should use serialization?

Ans. Serialization is used when data is transmitted over the network. Serialization enables us to save the object state and convert it into a byte stream that can be sent over the network. When the byte stream transfers over the network, the object is re-created at the destination.

Q159. When we use the Serializable vs. Externalization interface?

Ans. Following are the difference between Serializable and Externalization:

Serializable

- Marker interface (no methods).
- Default serialization (JVM handles everything).
- **Pros**: Easy to implement.
- **Cons**: Less control, slower for complex logic.
- **No constructor called** during deserialization.

Externalization

- Child of Serializable with **writeExternal()/readExternal()**.
- **Pros**: Faster (custom logic), full control over serialization.
- **Cons**: Manual implementation required.
- Requires **public no-arg constructor**.

Q160. Explain JSON.

Ans. JSON stands for JavaScript Object Notation. It is a lightweight data-interchange format that is used as an alternative to XML. It is language-independent and is easy to understand. JSON uses JavaScript syntax, but its format is text only. The text can be read and used as a data format by any programming language.

Q161. What are the advantages of JSON over XML?

Ans. The advantages of JSON are:

- JSON is lighter and faster. The XML software parsing process can take a long time.
- The structure of JSON is straightforward, readable, and easily understandable.
- JSON objects and code objects match, enabling us to quickly create domain objects in dynamic languages.
- JSON uses a map data structure unlike the tree structure in XML.
- While JSON supports multiple data types—string, number, array, or Boolean – XML data are all strings.