Core Java Interview Questions

S.No	Topic Name	Sub - Topics
1	Java Introduction	Java History, Java Features, Java Editions, Difference between C, C++ and Java
		Types of Applications, JDK ,JRE, JVM, JIT,
2	JVM & First Java Program	JVM Architecture,Writing, Saving, Compilation and Execution of a Java Program
3	Understanding Object Class	Methods in Object Class
		Methods in Object Class
4	Naming Conventions, Java Method	Rules followed for a Java Identifier,Understanding Seperators
5	Returning a value from a method	Java Method Syntax, Zero Parameters Method, Methods with Parameters, Method Returns Void, Method Returns Value
6	Java main()	Understanding syntax of java main(),Command Line Arguments, Varargs
		Understanding syntax of java main()
7	Data Types	Primitive Data Types, Reference Types, Java Language Keywords.
8	Comments, Variables	Single Line Comments, Multiple Line Comments, Instance Variable, Static Variable, Local Variable.
9	Constructor	What is a Constructor, Types of Constructors
		Rules for a Constructor, Constructor Vs Method
		Coding programs on Constructors
10	Operators	Increment & Decrement Operators
		Arithmetic Operators, Relational Operators, Logical Operators, Negation Operator , Assignment Operator, Bitwise Operators
11	Understanding static Keyword	Static Variable, Static Method, Static Method Vs Instance Method, Static Block
12	Understanding Instance Members	Instance Variable Vs Static Variables, Instance Method, Instance Block Vs Static Block
13	static Keyword (Logical session)	Logical programs on static
14	Type Casting	What is Type Casting, Types of Type Casting, Implicit Type Casting, Explicit Type Casting,Upcasting, Downcasting
15	Wrapper Classes	What is the use of Wrapper Classes, List of Wrapper Classes, Wrapper Class Methods, Auto-Boxing, Auto-Unboxing
16	Control Statements	Selection Statements (if, if-else, if-else-if, switch)
		Logical programs on (if, if-else, if-else-if)
		Understanding switch case
		Iteration Statements (while, do-while, for, for-each)
		Understanding for loop
		Logical programs on for loop
		Understanding for-each loop & Jump Statements (break, continue, return)
17	Scanner Class	Understanding Scanner Class and its methods
18	Packages in Java	Need of Packages, Structure of a package, Built-in Packages, Package naming rules, How to access package from another package
19	Access Modifiers	Class Level Access Modifiers, Member Level Access Modifiers
20	Understanding 'this' Keyword	Uses of this keyword
21	Arrays	How to declare an Array, How to Instantiate an array, Rules for Array Instantiation,
		How to access an array element, length Vs length(), Multidimensional Array
		Logical Programming on Arrays
22	Strings	Why Strings are Immutable, SCP
		String Class Methods

		String Vs StringBuffer, StringBuffer Vs StringBuilder, String Vs StringBuffer Vs StringBuilder, String Buffer and String Builder Class Methods
		Logical Programming on Strings
23	Introduction To OOP's	Introduction, Procedure Oriented Vs Object Oriented Approach, Encapsulation, Inheritance, Polymorphism, Abstraction, Composition, Aggregation.
24	Encapsulation	What is Data Hiding, What is Encapsulation, Understanding Getter & Setters, What is the need for Encapsulation
25	Inheritance	What is Inheritance, Rules for Java Inheritance, Types Of Inheritance,
		Understanding super keyword, super Vs this, super() Vs this()
26	Polymorphism	What is Method Signature, What is polymorphism, Types of polymorphism, Method Overloading
		Method Overriding, Overloading Vs Overriding , Annotations
27	Abstraction	What is Abstraction, How can we achieve abstraction, what is abstract method, Abstract class, Rules for abstract class.
		What is interface, Interface Vs Abstract class, rules for java interface, How multiple Inheritance is achieved with interfaces,Functional Interface, Marker Interface.
28	Multhreading	Multitasking, What is Thread, Multithreading, Main Thread, Lifecycle of a Thread, implements Runnable Vs extends Thread
		Thread Priorities, Getting and Setting Name of a Thread, Performing multiple task by multiple thread
		Thread class methods
		Synchronization, Deadlocks
		InterThread Communications, Deamon Thread
		Thread Group & Thread Pool (Executor Framework)
29	Exception Handling	What is Exception, Exception Vs Error, Types of Errors and Exceptions, Hierarchy of Exception classes, Keywords used for Exception Handling
		Exception handling Rules, Methods To Display Exception Information,Finally Block (java -7 try with resources)
		Understanding throw & throws , Creating a user defined Exception Class
30	Understanding final Keyword And Garbage Collection	Uses of final Keyword, How Automatic Garbage Collection Works, When an Object is eligible for Garbage Collection, finalize() of java.lang.Object, Usage of equals and hash code method, System.gc(), Runnable.gc()
31	IO Streams	Streams in Java, Byte Streams
		Character Streams,Buffered Streams
		Data Streams, Object Streams
32	Collection Framework	Limitations of Array, Introduction to Collections,Collection Interface, Overview of Set - List - Queue and Map, Retrieving Elements from Collections
		Understanding List Interface (ArrayList) Generics
		Understanding List Interface (Vector,Stack), Generics
		Understanding List Interface (LinkedList), Generics
		Understanding Set Interface (HashSet,LinkedHashSet,TreeSet,EnumSet)
		Understanding Queue Implementaion Classes (ArrayDeque, PriorityQueue)
		Understanding Map Interface (HashMap, LinkedHashMap)
		Understanding Map Interface (Hash table, TreeMap, WeakHashMap, IdentityHashMap, Properties Class)
		Implementing Comparable & Comparator Interfaces, Iterator, ListIterator, Spliterator, Enumeration
33	Java Utility Classes	StringTokenizer, Date, Calendar,Formatting Date and Time, Locale, Formatter, Random classes,Arrays class,Collections class
34	Java 8 Features	Lambda expressions, Functional interfaces, Default methods, Static methods, private methods in interface, Method references,
		Stream API, Collectors class, forEach() method,

		Optional class, Parallel array sorting, New datetime,
35	Inner Classes	Nested Inner Class, Static Inner Class, Method Local Inner Class, Anonymous Inner Class
36	Enum	Enum inside a Class, Enum in a Switch, Iteration Through an Enum,

1. JAVA INTRODUCTION

1. History of Java

1) Who developed Java, and in which year?
2) What was Java originally called, and why was it renamed?
3) Why was Java developed?
4) Which company currently owns Java?
5) In which year was Java first released publicly?
6) What is the significance of the Java 1.0 release?
7) How has Java evolved from JDK 1.0 to the latest version?
8) What is the difference between Java SE, EE, and ME in terms of release history?
9) Why is Java called a platform-independent language?
10) What made Java more popular than other languages during its early release?

2. Features of Java

1) List and explain the main features of Java.
2) What does “Platform Independent” mean in Java?
3) What is meant by “Object-Oriented” in Java?
4) What is the importance of “Robustness” in Java?
5) What makes Java “Secure”?
6) Explain the feature “Portable” in Java.
7) How does Java achieve “Multithreading”?
8) What is “Dynamic” in Java?
9) Explain “Distributed” and how Java supports distributed systems.
10) How does Java achieve “Architecture Neutrality”?
11) What does “Compiled and Interpreted” mean in Java?
12) How does automatic garbage collection contribute to reliability?
13) What is bytecode, and why is it important?
14) Why is Java often called “Write Once, Run Anywhere”?
15) How do features like JIT improve Java’s performance?

3. Java Editions

1) What are the different editions of Java?

2) What is Java SE used for?

3) What is Java EE (Jakarta EE), and what does it include?

4) What is Java ME used for?

5) What is JavaFX, and when is it used?

6) What is the difference between Java SE and Java EE?

7) Can you build web applications with Java SE?

8) Which edition would you choose for enterprise-level applications, and why?

9) What tools or APIs are included in Java EE?

10) What happened to Java EE when Oracle handed it over to Eclipse Foundation?

4. Difference Between C, C++, and Java

1) What are the key differences between C and Java?

2) What are the key differences between C++ and Java?

3) Is Java fully object-oriented? Explain.

4) How does Java handle memory differently from C++?

5) Why doesn't Java support pointers?

6) How is exception handling different in C++ and Java?

7) What is the difference between compiler behavior in C++ and Java?

8) How does Java achieve platform independence while C/C++ do not?

9) What are destructors in C++? How is it handled in Java?

10) How is multiple inheritance handled in Java?

11) What is the difference between structures in C and classes in Java?

12) Why does Java not support operator overloading?

13) Compare access specifiers in C++ and Java.

14) What is the difference in memory management between C++ and Java?

15) Can you directly manage memory in Java like in C/C++?

5. Types of Java Applications

1) What are the different types of applications that can be developed using Java?

2) What is a standalone application? Give examples.

3) What is a web application in Java?

4) What technologies are used to create web applications in Java?

5) What is an enterprise application?

6) Give examples of enterprise-level Java technologies.

7) What is a mobile application in Java?

8) How does Java support cloud or distributed applications?

9) Which Java edition is used for embedded systems?

6. JDK (Java Development Kit)

1) What is JDK?

2) What does JDK contain?

3) What is the difference between JDK, JRE, and JVM?

4) Which tools are part of JDK?

5) Can you run a Java program without JDK?

6) What are the common JDK commands and their purposes?

7) How do you check your JDK version?

8) What is the difference between Oracle JDK and OpenJDK?

9) Can multiple JDK versions be installed on the same system?

10) How does JDK help in development and compilation?

7. JRE (Java Runtime Environment)

1) What is JRE?

2) What is included in JRE?

3) Can you run Java code without JRE?

4) How is JRE different from JDK?

5) What happens if your JRE version doesn't match your JDK version?

6) What are the components of JRE?

7) Does JRE contain compiler tools?

8) How does JRE interact with JVM?

9) Can you have JRE without JDK?

10) Why is JRE required for end users?

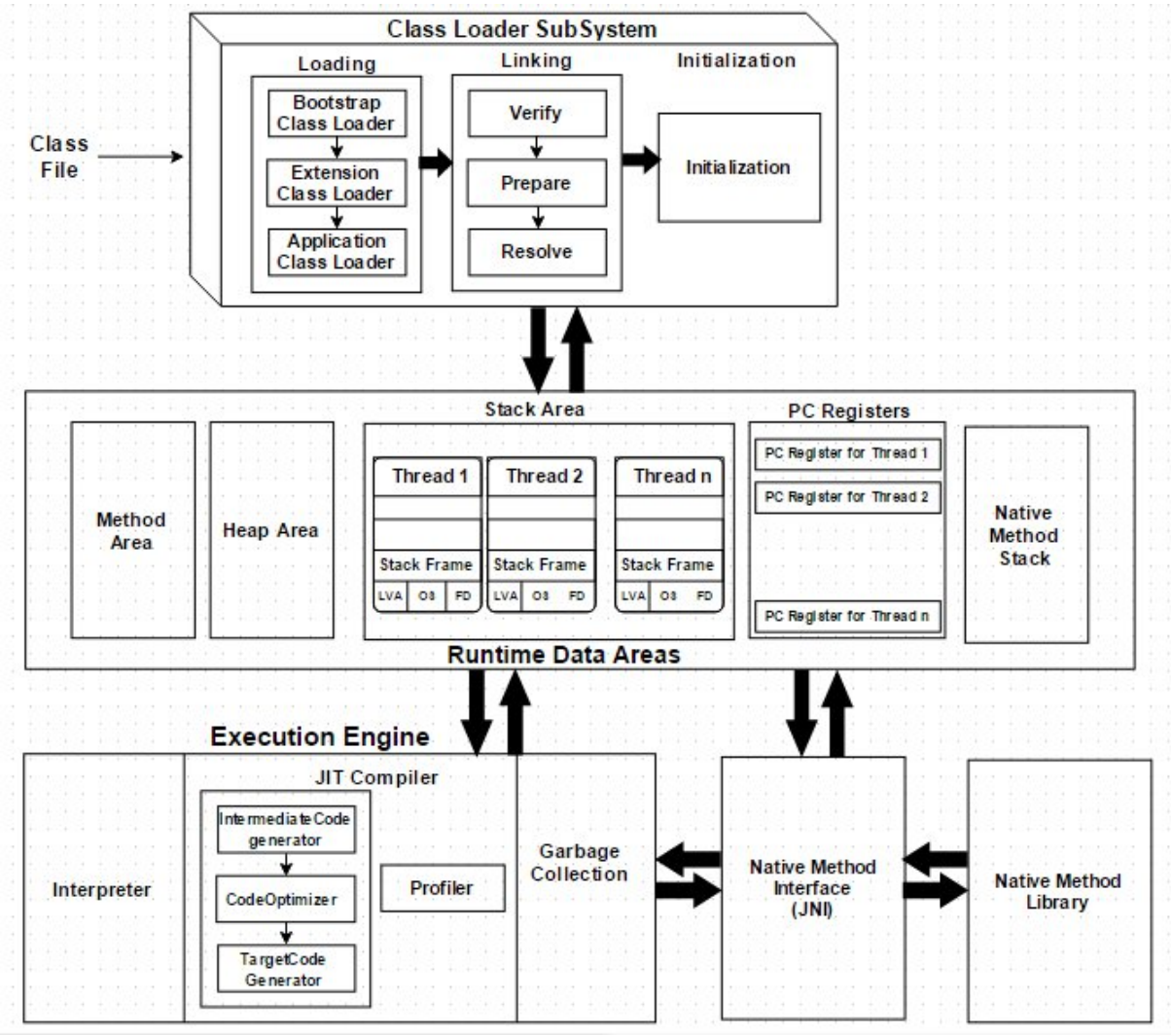
8. JVM (Java Virtual Machine)

1) What is the JVM?

A Virtual Machine is a software implementation of a physical machine.

Java was developed with the concept of WORA (Write Once Run Anywhere), which runs on a VM.

The compiler compiles the Java file into a Java .class file, then that .class file is input into the JVM, which Loads and executes the class file. Below is a diagram of the Architecture of the JVM.



2) What is the role of JVM in Java execution?

3) Explain the lifecycle of a Java program in terms of JVM.

4) What are the main components of JVM?

5) What is the role of the Class Loader in JVM?

6) What is the role of the Execution Engine?

7) How does the JVM perform memory management?

8) What are the different memory areas in JVM?

9) What is method area and heap area?

10) How does JVM perform garbage collection?

11) Is JVM platform dependent or independent?

12) How does JVM handle bytecode?

13) What are native methods in JVM?

14) How does the JVM differ for different operating systems?

15) In which order does the JVM load classes (hierarchy of class loaders)?

16) What is the delegation model of class loading?

17) What kind of errors are detected during verification?

18) What is the Prepare phase and what happens during it?

19) What is the *Resolution* step in linking?

20) What is the main purpose of the initialization phase?

21) What happens when a StackOverflowError occurs?

22) What happens when an OutOfMemoryError occurs in the Heap?

23) What is the Native Method Stack, and why is it separate from JVM stack?

9. JIT (Just-In-Time Compiler)

1) What is JIT compiler in Java?

2) Why is JIT compiler used?

3) When does JIT compile code?

4) How does JIT improve performance?

5) What is the difference between interpreter and JIT?

6) What are the types of JIT optimizations?

7) How does JIT work with bytecode and native code?

8) Is JIT part of JVM or JRE?

9) What happens if JIT is disabled?

---

2. First Java Program

1. Basic Steps to Write and Run a Java Program

Step 1: Write the Code

- Create a simple Java program using any text editor or IDE (like VS Code, IntelliJ IDE, or Eclipse).

// File: HelloWorld.java

```
public class HelloWorld
{
      public static void main(String []  args)
      {
            System.out.println("Welcome to Java!");
       }
 }
```

Step 2: Save the File

- Save the file **with the same name as the class** that contains the main() method.

Example:

Class name → HelloWorld

File name → HelloWorld.java

Step 3: Compilation

- Compile the program using the javac command.

```
javac HelloWorld.java
```

- This generates a **bytecode file** named: HelloWorld.class

Step 4: Execution

- Run the program using the java command.

```
java HelloWorld
```

Output: Welcome to Java!

What Happens Internally (JVM Process)

- **Writing :** You write Java source code in .java file.
- **Compilation :** javac converts .java → .class (bytecode).
- **Class Loading :** JVM loads .class file into memory using ClassLoader.
- **Bytecode Verification :** Ensures code is safe and follows Java rules.
- **Execution :** JIT (Just-In-Time Compiler) converts bytecode → native machine code and executes.

Interview Questions

1) What is the extension of a Java source file?

2) What is the extension of the compiled bytecode file?

3) Which command is used to compile Java code?

4) Which command is used to run a Java program?

5) Why do we not use .class when running the program?

6) What is the entry point of a Java program?

7) What happens if the file name and class name differ?

8) What is bytecode?

9) Can we have multiple classes in one Java file?

10) What happens when we omit public keyword in class?

Coding Questions

1) Write a program to print "Hello, Java Developer!"

2) Write a program to add two numbers

3) Write a program to find the largest of two numbers

4) Write a program to check even or odd number

5) Write a program to print sum of first 10 natural numbers

6) Write a program to swap two numbers

7) Write a program to swap two numbers without using third variable

8) Write a program to print ASCII value of a character

9) Write a program to check if number is positive or negative

10) Write a program to print multiplication table

---

3. OBJECT CLASS

Interview Questions

1) What is Object Class in Java?

2) What is the superclass of all Java classes?

3) Is it necessary to extend Object class explicitly?

4) How many methods does the Object class have?

5) Can we override methods of the Object class?

6) Which methods are declared as final in the Object class?

7) What is the difference between == and equals()?

8) Why should we override hashCode() when we override equals()?

9) What is the default implementation of toString()?

10) What is the purpose of clone() method?

11) What is shallow copy?

12) What is deep copy?

13) What is the purpose of finalize() method?

14) What is the use of getClass()?

15) Why are wait(), notify(), and notifyAll() methods defined in Object class and not in Thread class?

16) What happens if we don't override toString()?

17) Can we override finalize()?

18) Can we call clone() without implementing Cloneable?

19) What is the difference between shallow copy and deep copy?

20) Why is finalize() deprecated in Java 9 and later?

Coding Questions

1) Write a Java program to display object details using toString() method.

2) Write a Java program to compare two objects using the equals() method.

3) Write a program to demonstrate the difference between == operator and equals() method.

4) Write a Java program to find whether two employee objects are equal by overriding equals() and hashCode().

5) Write a Java program to print the runtime class name using the getClass() method.

6) Write a program to clone an object using the clone() method.

7) Write a Java program to demonstrate deep cloning using the clone() method.

8) Write a Java program to override the hashCode() method.

9) Write a program to check the hash codes of two equal objects.

10) Write a Java program to call the finalize() method and see when it executes.

11) Write a Java program to demonstrate shallow copy?

12) Write a Java program to demonstrate deep copy?

---

4. IDENTIFIERS

Interview Questions

1) What is an identifier in Java? Why do we use identifiers in Java?

2) What are the rules for creating identifiers in Java?

3) Are Java identifiers case-sensitive?

4) Can we use special characters like @, #, $, % in identifiers?

5) Can we use keywords as identifiers?

6) What are reserved keywords? How are they different from identifiers?

7) Can method name and variable name be the same identifier?

8) What is the difference between an identifier and a literal?

9) What is the difference between identifier and variable?

10) What are legal and illegal identifier examples?

keywords

Interview Questions

1) What is a keyword in Java?

2) How many keywords are available in Java?

3) Are Java keywords case-sensitive?

4) Why are keywords reserved in Java?

5) Can keywords be used as identifiers? Why not?

6) What is the use of static keyword?

7) What is the use of final keyword? What happen when we declare a variable, method, class as final?

8) What does the super keyword do in Java?

9) What is the role of this keyword?

10) What is the difference between this and super?

11) What is the purpose of the new keyword?

12) Why do we use the return keyword?

13) What are break and continue keywords?

14) What is the use of void keyword?

15) Explain public, private, protected keywords.

16) What is the difference between abstract and final keywords?

17) What is the role of the extends keyword?

18) What is the use of implements keyword?

19) What is the use of package keyword?

20) What is the difference between import and package?

21) What is the purpose of instanceof keyword?

22) Explain the use of interface keyword.

23) What is the use of throws and throw keywords?

24) What does the try, catch, and finally block mean?

25) What is the use of enum keyword?

26) What does the transient keyword do?

27) What is the role of volatile keyword?

28) What is the use of synchronized keyword?

29) What is the meaning of strictfp keyword?

30) Where do we use native keyword?

31) What is the difference between const and final in Java?

32) Is goto keyword supported in Java?

33) Describe real-time uses of transient and volatile?

---

5. RETURNING A VALUE FROM A METHOD

Java Method Syntax

Interview Questions

1.What is the syntax of a method in Java, and what are its main components?

2.What is the difference between instance methods and static methods in Java?

3.What is the difference between a method declaration and a method definition?

4.Can a Java method have multiple return statements? Explain with an example.

5.What are access modifiers in Java methods, and how do they affect visibility?

Programs

1) Write a Java method with no parameters that returns an integer value.

2) Write a static method that takes two numbers as parameters and returns their sum.

3) Write a public method that returns a string message.

4) Write a method that prints a message to the console but does not return any value.

5) Write a method that accepts parameters, performs a calculation, and returns the result.

Non-Parameterized Methods

Interview Questions

1) Explain the purpose of non-parameterized methods in Java. When would you prefer using them?

2) Can a non-parameterized method return different data types? Demonstrate with examples.

3) Can a non-parameterized method access instance variables and static variables? Explain.

4) Can a non-parameterized method call another method that has parameters? Explain with an example.

5) What happens if you call a non-parameterized instance method before creating an object?

Programs

1) Write a non-parameterized method that returns your name.

2) Write a non-parameterized method that returns the current year.

3) Write a non-parameterized method that calculates and returns the square of 5.

4) Write a non-parameterized method that prints a welcome message.

5) Write a non-parameterized method that returns a greeting message based on a predefined variable.

Methods with Parameters

Interview Questions

1) What are parameters in a Java method, and why are they used?

2) What is the difference between formal parameters and actual parameters in Java?

3) Is Java pass-by-value or pass-by-reference? Explain with an example.

4) Can you pass an array or object as a parameter to a method in Java? Demonstrate with an example.

5) Can a method have multiple parameters? Explain with an example.

Programs

1) Write a method that accepts two integers and returns their sum.

2) Write a method that accepts a string parameter and returns its length.

3) Write a method that accepts an array of integers and returns the maximum element.

4) Write a method that accepts two numbers and returns the larger one.

5) Write a method that accepts multiple parameters, performs a calculation, and returns the result.

Method Returns Void

Interview Questions

1) What does the void return type signify in a Java method?

2) Can a void method include a return statement? Explain with an example.

3) When should you use a void method instead of a method that returns a value?

4) Can a void method call another method that returns a value? Explain how.

5) What are the advantages and limitations of using void methods in Java?

Programs

1) Write a void method that prints “Hello, World!” to the console.

2) Write a void method that accepts a name as a parameter and prints a personalized greeting message.

3) Write a void method that prints numbers from 1 to 10 using a loop.

4) Write a void method that accepts an array of integers and prints all its elements.

5) Write a void method that performs a calculation but does not return the result.

Method Returns Value

Interview Questions:

1) What does it mean when a method returns a value in Java? Explain with an example.

2) How is the return type of a method declared, and why must it match the returned value?

3) Can a method return different types of values based on conditions? Explain with example.

4) What happens if the return statement in a method is missing or mismatched with the declared return type?

5) What are covariant return types in Java, and how are they used in method overriding?

Programs

1) Write a method that returns the factorial of a given number.

2) Write a method that checks if a number is prime and returns a boolean value.

3) Write a method that takes a string and returns it in reverse order.

4) Write a method that returns the area of a circle given its radius.

5) Write a method that returns the average of two numbers.

Factory Method – Return Type in Java

Interview Questions:

1) What is a Factory Method in Java, and how does it use return types to create objects?

2) What types of values can a method return in Java, and how are they used in factory methods?

3) What happens if a non-void method does not return a value? Explain with an example.

4) What is the difference between returning a primitive type and returning an object in Java?

5) What are covariant return types in Java, and how are they applied in factory method implementations?

6) How do return types enable polymorphism in Java, particularly in the context of factory methods?

Programs

1) Write a factory method that returns an object based on the input type (e.g., Car, Bike).

2) Write a factory method that creates and returns different shapes (Circle, Rectangle) based on user input.

3) Write a factory method that returns an instance of a subclass depending on a condition.

4) Write a program demonstrating covariant return types using a base class and a subclass.

5) Write a factory method that returns null when no matching object type is found.

Code Snippet Interview Questions

1. Basic Method Syntax

Question:

```
public class Test
{
  public static void main(String[] args)
  {
     greet();
  }
  public static void greet() {
     System.out.println("Hello from Java!");
  }
}
```

Q: What is the output of this program?

Q: Why is the greet() method declared as static?

2. Zero Parameter Method

Question:

```
public class Example
{
public static void main(String[] args)
{
     displayMessage();
  }
  public static void displayMessage()
{
     System.out.println("This is a zero-parameter method.");
```

```
  }
}
```

Q: What will happen if you remove () from displayMessage while calling it?

Q: Can you overload the displayMessage() method?

3. Methods with Parameters

Question:

```
public class Demo
{
  public static void main(String[] args)
{
    int result = multiply(5, 3);
    System.out.println(result);
  }
  public static int multiply(int a, int b)
{
    return a * b;
  }
}
```

Q: What will be printed?

Q: What happens if you change the method's return type to void but still use return a * b;?

4. Method Returns Void

Question:

```
public class VoidExample
{
  public static void main(String[] args)
{
    printSum(10, 5);
  }
  public static void printSum(int x, int y)
{
    int sum = x + y;
```

```
    System.out.println("Sum = " + sum);
  }
}
```

Q: What happens if you try to store the result like int s = printSum(10, 5);?

5.Method Returns Value

Question:

```
public class ReturnExample
{
  public static void main(String[] args)
{
    int value = getSquare(4);
    System.out.println("Square = " + value);
  }
  public static int getSquare(int num)
{
    return num * num;
  }
}
```

Q: What will be the output?

Q: What happens if you forget to include the return statement?

6. Method Overloading Example

Question:

```
public class OverloadTest
{
  public static void main(String[] args)
{
    System.out.println(add(5, 10));
    System.out.println(add(2.5, 3.5));
  }
  public static int add(int a, int b)
{
```

```
    return a + b;
  }
  public static double add(double a, double b)
{
    return a + b;
  }
}
```

Q: What will be the output?

Q: How does Java determine which add() method to call?

7. Missing Return Type

Question:

```
public class ErrorExample {
  public static add(int a, int b) {
    return a + b;
  }
}
```

Q: What happens when you try to compile this code?

Q: How can you fix this error?

8. Return vs Print

Question:

```
public class Difference
{
  public static void main(String[] args)
{
    System.out.println(printMessage());
  }
  public static String printMessage()
{
    System.out.println("Inside Method");
    return "Returned Value";
  }
```

```
}
```

Q: What will be the output?

Expected Output:

Inside Method

Returned Value

Q: Why are there two lines printed?

9. Method Returning Object

Question:

```
class Student
{
  String name;
  Student(String name)
{
    this.name = name;
  }
}
public class Main
{
  public static void main(String[] args)
{
    Student s = createStudent("Saraswati");
    System.out.println(s.name);
  }
  public static Student createStudent(String name)
{
    return new Student(name);
  }
}
```

Q: What will be the output?

Q: What is the return type of the createStudent() method?

10. Trick Question — Return Inside Void

Question:

```
public class ReturnInsideVoid
{
  public static void main(String[] args)
{
     show();
     System.out.println("Done");
  }
  public static void show()
{
     System.out.println("Before return");
     if (true)
        return;
     System.out.println("After return");
  }
}
```

Q: What will be the output?

Expected Output:

Before return

Done

Q: Why doesn't "After return" print?

11. Covariant Return Type Example

```
class Parent
{
   Parent show()
{
      System.out.println("Parent show()");
      return this;
   }
}

class Child extends Parent
```

```
{
   @Override
   Child show()
{
      System.out.println("Child show()");
      return this;
   }
}

public class Test
{
   public static void main(String[] args)
{
      Parent obj = new Child();
      obj.show();
   }
}
```

Q: What is the output of this program?

Q: What would happen if we change the return type in Child to Parent?

---

6. Java main()

Interview Questions

1. What is the signature of the main() method in Java?
2. Why is main() declared as public static void ?
3. Can we overload the main() method? If yes, how does Java determine which main() to call?
4. Can the main() method be private or protected?
5. Explain what happens when a Java program does not contain a main() method?
6. Explain why main() must be static?Can we change the order of keywords like static, public?
7. Can we have multiple main() methods in different classes?

8. Can main() throw exceptions? If yes, which types?
9. Can the main() method return a value?
10. What is the role of the String[] args parameter? What if we change the argument type? (Example: int[] instead of String[])?
11. Can the main() method be called recursively?
12. How jvm internally call main mathod?
13. Can we write main() inside an abstract class/interface?
14. What happens if main() is declared synchronized?
15. Can we use varargs (…) in place of array in main method?

Programs:

1. Write a program to demonstrate overloading of the main method.?
2. Write a program to call one class's main from another class.
3. Demonstrate main() calling itself recursively with a base case.
4. Write a program with multiple classes having main() — execute each.

Command Line Argument

Interview Questions

1. What are command-line arguments in Java and how are they passed to a program?
2. What happens if no arguments are passed to main() ?
3. Why String[] is used for command line arguments?
4. How to access a command line argument?
5. What is the difference between using the Scanner class and command-line arguments for input in Java?

Programs:

1. Write a program to find the maximum and minimum number from command-line arguments.
2. Implement a program that reverses the order of command-line arguments.
3. Write a program that counts the number of vowels in all command-line arguments.
4. Write a program to check if a specific argument exists among the command-line inputs.
5. Write a program to sort numbers passed as command-line arguments in ascending order.

Varargs in java

Interview Questions

1. What are varargs in Java?
2. How do varargs differ from arrays?

3. Can a method have multiple varargs parameters?
4. Where should varargs be placed in the method parameter list?
5. Can varargs accept zero arguments?
6. How does Java internally treat varargs?

Programs:

1. Write a method using varargs to calculate the sum of numbers.
2. Write a method to find the maximum value using varargs.
3. Write a method to concatenate multiple strings using varargs.
4. Write a method to calculate the average of double values using varargs.
5. Write a method to count the number of even numbers from varargs input.

---

7. DATATYPES

Interview Questions

1. What is a datatype in Java, and why do we need it for every variable?
2. What are the main types of datatypes in Java? Give examples.
3. What is a primitive datatype? Why do we use primitive datatypes?
4. List all primitive datatypes in Java. What kind of values do they store?
5. What is the size, default value, and range of each primitive datatype?
6. What is a reference datatype? Give examples.
7. Can primitive datatypes hold null? Can reference datatypes hold null?
8. What is the difference between primitive and reference datatypes?
9. Can a variable be declared without assigning a value? What happens if you try to use it?
10. Can primitive datatypes be declared inside constructors, static blocks, instance blocks, or non-static methods? Explain.
11. If you declare all primitive datatypes inside a method and try to print them without initialization, what happens?
12. What happens if you declare a primitive variable with a value larger than its range? Give an example.
13. If you declare a variable with a primitive types, what does JVM do internally? How much memory is allocated?

14. Where are primitive variables stored in memory?
15. Where are reference variables stored in memory?
16. What happens if you assign one reference variable to another?
17. Explain overflow and underflow with examples using byte, short, int, or long.
18. What happens if you call a method on a null reference variable?
19. Show an example where a local variable has the same name as a class-level variable (variable shadowing).
20. Can primitive and reference datatypes be mixed in expressions, like int + Integer? What happens?
21. Can a final primitive or reference variable be changed after initialization? What happens?

Code Snippets :

```
public class Test1
{
   public static void main(String[] args)
{
       byte b = 10;
       short s = 100;
       int i = 1000;
       long l = 10000L;
       float f = 10.5f;
       double d = 20.5;
       char c = 'A';
       boolean bool = true;
       System.out.println(b + " " + s + " " + i + " " + l + " " + f + " " + d + " " + c + " " + bool);
   }
}
```

```
public class Test2
{
   public static void main(String[] args)
{
       String s = null;
       System.out.println(s);
```

```
    }
}
```

```
public class Test3
{
   static int x;
   public static void main(String[] args)
{
      System.out.println(x);
   }
}
```

```
public class Test4
{
   static String str;
   public static void main(String[] args)
{
      System.out.println(str);
   }
}
```

```
public class Test5
{
   static int i = 10;
   public static void main(String[] args)
{
      int i = 50;
      System.out.println(i);
   }
}
```

```
public class Test6
```

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

```
public class Test7
{
   int a;
   {
      a = 10;
      System.out.println(a);
   }
   Test7()
{
      System.out.println(a);
   }
   public static void main(String[] args)
{
      new Test7();
   }
}
```

```
public class Test8
{
  static int i=10;
   public static void main(String[] args)
{
      int i;
```

```
        System.out.println(i);
    }
}
```

```
public class Test9
{
    public static void main(String[] args)
{
        byte b = 127;
        b += 1;
        System.out.println(b);
    }
}
```

```
public class Test10
{
    public static void main(String[] args)
{
        int i = 130;
        byte b = (byte)i;
        System.out.println(b);
    }
}
```

```
public class Test11
{
    public static void main(String[] args)
{
        byte b1 = 10;
        byte b2 = 20;
        int result = b1 + b2;
        System.out.println(result);
    }
```

```
}
---------------------------------------------------------------------------------------------------------------------------
public class Test12
{
   public static void main(String[] args)
{
       int i = 10;
       double d = 5.5;
       System.out.println(i + d);
   }
}
---------------------------------------------------------------------------------------------------------------------------
public class Test14
{
   public static void main(String[] args)
{
       String a = "Hello";
       String b = a;
       b = "World";
       System.out.println(b);
   }
}
---------------------------------------------------------------------------------------------------------------------------
class Person {
 String name;
}
public class Test16
{
   public static void main(String[] args)
{
       Person p1 = new Person();
       p1.name = "NIT";
       Person p2 = p1;
```

```
        p2.name = "TECH";
        System.out.println(p1.name);
    }
}
```

```
public class Test17 {
    public static void main(String[] args) {
        Integer a = 10;
        int b = 5;
        System.out.println(a + b);
    }
}
```

```
public class Test18
{
    public static void main(String[] args)
{
        char c = 'Z';
        int i = c;
        System.out.println(i);
    }
}
```

```
public class Test19
{
    public static void main(String[] args)
{
        Integer a = 100;
        Integer b = 100;
        Integer x = 200;
        Integer y = 200;
        System.out.println(a == b);
        System.out.println(x == y);
```

```
    }
}
```

```
public class Test20
{
   public static void main(String[] args)
{
      Integer a = null;
                  int b=a;
                  System.out.println(b);
   }
}
```

---

8. COMMENTS IN JAVA

Interview Questions

1. What are the types of comments in Java?
2. Difference between single-line and multi-line comments.
3. Can comments be nested?
4. Are comments compiled into bytecode?
5. What is the purpose of comments in coding standards or documentation?

Variables in Java (Instance, Static, Local)

Interview Questions

1. What are the types of variables in Java?
2. Difference between instance, static, and local variables.
3. Can we declare variables inside a method as static?
4. Scope and lifetime of instance variables, static variable, local variables?
5. How are static variables shared among objects?
6. Default values of instance, static, and local variables.?
7. Can a variable shadow another variable?
8. How do instance, static variables, differ from local variables in memory allocation?

9. What is the difference between class-level and object-level variables?
10. What happens if we try to access a local variable in Java without initializing it?

Programs:

1. Write a program to demonstrate the difference between instance and static variables.
2. Show how modifying a static variable affects all objects of a class.
3. Write a method to demonstrate local variable scope.
4. Create a program where variable shadowing occurs and explain the output.
5. Write a program to count the number of objects created using a static variable?
6. Predict the output: program using local variables inside loops and methods.
7. write a program demonstrating instance variable initialization and default values.
8. Create a program to show that local variables must be initialized before use.

---

9. CONSTRUCTOR

Interview Questions

1) What is constructor in java? What is the use of constructors?
2) What are the different types of constructors?
3) When a constructor is executed?
4) What happens when we did not provide any constructor? How object is created?
5) What is the use of the this() keyword in a constructor?
6) What is the use of the super() keyword in a constructor?
7) What is constructor overloading? can we overload a constructor?
8) Is it possible to override a constructor? What happens when we override a constructor?
9) What is constructor chaining?
10) Can a constructor be private? If yes, why do we use it?
11) Can an abstract class have a constructor?
12) Can an interface have a constructor?
13) Can a constructor be static or abstract? Why or why not?
14) Can a constructor be final?
15) What is a copy constructor in Java?
16) What is the difference between a method and a constructor?

17) Can we use final variables in constructors?

18) What is the use of exception handling inside constructors?

19) Why is super() called automatically if we don't write any constructor?

Programs:

Parameterized Constructor Example

```
class Student
{
   int id;
   String name;
   Student(int i, String n)
  {
      id = i;
      name = n;
   }
   void display()
  {
      System.out.println(id + " " + name);
   }
   public static void main(String[] args)
 {
      Student s1 = new Student(1, "Ram");
      Student s2 = new Student(2, "Krishna");
      s1.display();
      s2.display();
    }
 }
```

Constructor Overloading

```
class Demo {
   Demo() {
      System.out.println("No-Arg Constructor");
   }
   Demo(int x)
  {
      System.out.println("Parameterized Constructor: " + x);
```

```
    }
    Demo(int x, String y)
   {
        System.out.println("Two-Arg Constructor: " + x + ", " + y);
    }
    public static void main(String[] args)
   {
        new Demo();
        new Demo(10);
        new Demo(20, "Hello");
    }
}
```

Copy Constructor Example

```
class Employee
{
    String name;
    int salary;
    Employee(String n, int s)
{
        name = n;
        salary = s;
    }
    Employee(Employee e) { // Copy constructor
        name = e.name;
        salary = e.salary;
    }
    void show() {
        System.out.println(name + " " + salary);
    }
    public static void main(String[] args) {
        Employee e1 = new Employee("Raju", 25000);
        Employee e2 = new Employee(e1);
        e1.show();
        e2.show();
    }
```

```
}
```

Constructor Chaining using this()

```
class Car {
    Car() {
        this("BMW");  // calling parameterized constructor
        System.out.println("Default constructor");
    }
    Car(String model) {
        System.out.println("Car model: " + model);
    }
    public static void main(String[] args) {
        new Car();
    }
}
```

Inheritance Constructor Chaining using super()

```
class A {
    A() {
        System.out.println("Parent Constructor");
    }
}
class B extends A {
    B() {
        super(); // calls Parent constructor
        System.out.println("Child Constructor");
    }
    public static void main(String[] args)
 {
        new B();
    }
}
```

Private Constructor (Singleton Pattern)

```
class Singleton
{
```

```
    private static Singleton instance = new Singleton();
    private Singleton() {
        System.out.println("Private Constructor");
    }
   public static Singleton getInstance() {
        return instance;
    }
    public static void main(String[] args) {
        Singleton s = Singleton.getInstance();
    }
}
```

Constructor in Abstract Class

```
abstract class Animal {
    Animal() {
        System.out.println("Animal Constructor");
    }
}
class Dog extends Animal {
    Dog() {
        System.out.println("Dog Constructor");
    }
    public static void main(String[] args) {
        new Dog();
    }
}
```

Constructor +Non- Static Block Order Program

```
class Order {
    {
        System.out.println("Non-Static Block")
}
    Order() {
        System.out.println("Constructor");
    }
    public static void main(String[] args) {
```

```
        new Order();
        new Order();
    }
}
```

---

10. OPERATORS

1. INCREMENT & DECREMENT OPERATORS

Used to increase or decrease a variable's value by 1.

++ (increment), -- (decrement)

Interview Questions

1. What are increment and decrement operators in Java?
2. What is the difference between pre-increment and post-increment?
3. What happens if you use increment operators on a final variable?
4. Can you use increment operators on boolean values or reference types? Why or why not?

Programs

Predict the output:

```
int a = 5;

int b = ++a + a++;

System.out.println(b);
```

1. Write a program to demonstrate the difference between a++ and ++a
2. What is the output? Explain the order of increment/decrement.

```
int a = 5;

int b = a++ + ++a + --a + a--;

System.out.println(a + " " + b);
```

3. What will be printed? (Hint: post vs pre increment difference)

```
int x = 10;

System.out.println(x++ + ++x + x);
```

2. ARITHMETIC OPERATORS

Used for mathematical calculations: +, -, *, /, %

Interview Questions

---

1)What are arithmetic operators in Java?

2)What happens when we divide numbers of different types in Java (int and float)?

For example:

```
int / int
float / int
int / float
float / float
```

3)What happens when you divide by zero using integer division?

4)What is the difference between / and % operators?

5)Can arithmetic operators be overloaded in Java?

Programs

1. Write a program to perform all arithmetic operations on two numbers.

2. Predict the output:

```
int a = 10, b = 3;

System.out.println(a / b);

System.out.println(a % b);
```

1. What is the output of each line? Why?

```
int a = 10, b = 3;

System.out.println(a / b);

System.out.println(a % b);

System.out.println(a * b - a / b + b);
```

2. Predict output and explain operator precedence.

```
int x = 5;

x = x * (x + 2) / (x - 3);

System.out.println(x);
```

3. RELATIONAL OPERATORS

Used to compare values: >, <, >=, <=, ==, !=

Interview Questions

1) What are relational operators used for?
2) What is the difference between == and equals()?
3) Can relational operators be used with boolean values?
4) What will be the output of 5 == 5.0?
5) What is the return type of a relational expression?
6) What is the difference between = and == in java?

Programs

1. Write a program to compare two numbers and print the larger one.
2. Predict the output:

```
System.out.println('A' < 'a');
```

3. What values will be printed?

```
int a = 10, b = 20, c = 10;

System.out.println(a == b);

System.out.println(a <= c);
```

```
System.out.println(a != c);
```

4. Why is the comparison between char and double valid?

```
System.out.println('A' < 'a');

System.out.println(5 == 5.0);
```

4. LOGICAL OPERATORS

Used to combine conditions: &&, ||, !

Interview Questions

---

1. What are logical operators in Java?
2. What is short-circuit evaluation?
3. What is the difference between & and &&?
4. What is the result of combining true/false values using logical operators?
5. Can logical operators be applied to non-boolean operands?

Programs:

1. Write a program to check if a number is within a specific range.
2. Predict the output:

```
int a = 5, b = 10;
    System.out.println(a > 2 && b < 5);
```

3. Explain how short-circuiting affects evaluation.

```
int a = 10, b = 5, c = 20;

System.out.println(a > b && a < c);

System.out.println(a > b || a > c);

System.out.println(!(a < c));
```

4. What will be printed? What happens if you change && to &?

```
int x = 5;

if (x > 0 && ++x > 5)

   System.out.println(x);
```

5. NEGATION OPERATOR

Used to reverse a boolean value: !

Interview Questions:

1. What is the purpose of the negation operator !?
2. Can the negation operator be used with integers?
3. What is the output of !(true && false)?
4. How does negation work with multiple conditions?

Programs

1. Write a program to check if a user is *not eligible* for voting using !.
2. What do the two lines print?

```
boolean flag = false;

System.out.println(!flag);

System.out.println(!!flag);
```

3. What is the final output?

```
int a = 10, b = 20;

System.out.println(!(a > b));
```

6. ASSIGNMENT OPERATOR

Used to assign values: =, +=, -=, *=, /=, %=

Interview Questions

1. What is the assignment operator in Java?
2. What does x += y mean?
3. Can you chain assignment operators like a = b = c = 10?

Programs

1. Write a program using compound assignment operators to update a variable.
2. What is the output? Which operator executes first?

```
int x = 5;

x += 3 * 2;

System.out.println(x);
```

3. What will be printed and why?

```
int a, b, c;

a = b = c = 10;

System.out.println(a + b + c);
```

7. BITWISE OPERATORS

Used for bit-level operations: &, |, ^, ~, <<, >>, >>>

Interview Questions

---

1. What are bitwise operators used for?
2. What does >> and >>> mean?
3. What is the effect of the ~ operator?
4. Are bitwise operations faster than arithmetic operations?

Programs

1. Write a program to swap two numbers using bitwise XOR.
2. Predict the output:

```
int a = 5, b = 3;

System.out.println(a & b);

System.out.println(a | b);

System.out.println(a ^ b);
```

3. What is left shift and right shift doing here?

```
int x = 8;

System.out.println(x << 2);

System.out.println(x >> 1);
```

8. BOOLEAN OPERATOR

Interview Questions

1. What are boolean operators in Java?
2. What is the default value of a boolean variable?
3. Difference between Boolean (wrapper class) and boolean (primitive type)?
4. Can boolean be converted to int directly?

Programs

1. Write a program using boolean variables to check eligibility for driving.
2. What is the value of result?

```
Boolean x = true, y = false;

boolean result = x && !y;

System.out.println(result);
```

3. What will be printed?

```
Boolean a = true;

a = !a || false;

System.out.println(a);
```

9. TERNARY OPERATOR

Short form of if-else: condition ? value1 : value2

Interview Questions

1. What is the ternary operator in Java?
2. Can a ternary operator replace an if-else statement?
3. What is the syntax of a ternary operator?
4. Can we have nest ternary operators in java? Give example?

Programs

1. Write a program to find the maximum of two numbers using a ternary operator.
2. Write a program to check if a number is even or odd using ternary operator.

3. What is the output? Can you rewrite this using if-else?

```
Int a = 10, b = 20;

int max = (a > b) ? a : b;

System.out.println(max);
```

4. What will be printed for x = 5, x = 6, and x = 8?

```
Int x = 5;

String result = (x % 2 == 0) ? "Even" : (x > 5 ? "Greater" : "Odd");

System.out.println(result);
```

10. MEMBER OPERATOR (. OR DOT OPERATOR)

Used to access members (fields and methods) of a class or object.

Interview Questions

---

1. What is the dot operator used for?
2. Can the dot operator be used with static members?
3. What is the difference between object.method() and ClassName.method()?
4. What happens when you use a dot operator on a null reference?

Programs

1. Write a simple class and use the dot operator to access its members.
2. Which members are accessed using the dot operator?

```
Class Student {

    int id = 1;

    void show() {

       System.out.println("ID: " + id);

    }

}

public class Test {
```

```
    public static void main(String[] args) {
        Student s = new Student();
        s.show();
        System.out.println(s.id);
    }
}
```

11. NEW OPERATOR

Used to allocate memory for an object.

Interview Questions

---

1. What does the new operator do?
2. Can an object be created without the new operator?
3. What happens if memory allocation fails?
4. Can we overload the new operator in Java?

Programs

1. Create an object using new operator and call its methods.
2. What will the output be? What happens when you remove new?

```
class Demo {
    Demo() {
        System.out.println("Object Created");
    }
}
public class Main {
    public static void main(String[] args) {
        Demo obj = new Demo();
    }
```

```
}
```

12. INSTANCEOF OPERATOR

Used to test whether an object is an instance of a particular class or subclass.

Interview Questions

1. What is the purpose of the instanceof operator?
2. What is the return type of instanceof?
3. Can instanceof be used with interfaces?
4. What happens if you use instanceof with null?

Programs

1. Write a program to check whether an object belongs to a given class using instanceof.
2. What will be printed? Why is instanceof used?

```
class Animal {}

class Dog extends Animal {}

public class Test {

  public static void main(String[] args) {

     Dog d = new Dog();

     System.out.println(d instanceof Animal);

     System.out.println(d instanceof Dog);

  }

}
```

3. What is the output and why?

```
String s = null;

System.out.println(s instanceof String);
```

11. INSTANCE MEMBERS

1) non-static variable

Interview Questions

1) What is a non-static variable in Java?
2) Where are non-static variables stored in memory?
3) When are non-static variables created and destroyed?
4) Can non-static variables be accessed without creating an object? Why or why not?
5) Can we use ‘this’ keyword with static variables? Why or why not?
6) What is the default value of a non-static variable if not initialized?
7) If we modify a non-static variable using one object, will that affect another object?
8) What is java.lang.NullPointerException, and how can we handle it?
9) Can we print an object directly? Explain toString() method.
10) Difference between memory locations of static and non-static variables.

Programs :

1) Can a static method directly access a non-static variable? Explain with an example.
2) Create a class with one static and one non-static variable. Print their values after creating multiple objects to show the difference in behavior.
3) Demonstrate the use of the this keyword to initialize non-static variables through a constructor.
4) What happens if you try to use the this keyword inside a static method? Explain with code.
5) Create a program where both static and non-static variables are incremented in a method, and analyze the final values.

2) non-static Method

Interview Questions

1) What is a non-static method in Java, and how is it different from a static method?
2) When and how can a non-static method be invoked?
3) Can a non-static method access static variables and methods? Why or why not?
4) Can a static method call a non-static method directly? Explain with reason.
5) How does memory allocation differ for static and non-static methods in the JVM?
6) Can we use the this keyword inside a non-static method? Why?

7) When are non-static methods loaded into memory?

8) Can a non-static method be overridden in a subclass? Explain with an example.

9) What happens if we try to call a non-static method using a null object reference?

10) Can a non-static method be declared as final, abstract, or synchronized? What are the implications?

Programs :

1) Write a program to show that a static method cannot call a non-static method directly (and how to fix it).

2) Create a class with both static and non-static methods. Access both using an object reference and observe the behavior.

3) Write a program to demonstrate the use of the this keyword inside a non-static method.

4) Create a parent and child class where the child overrides a non-static method from the parent. Show method overriding in action.

5) Write a program that tries to call a non-static method using a null reference and observe what happens.

3) non-static Block

Interview Questions

1) What is a non-static (instance) block in Java, and when is it executed?

2) How is a non-static block different from a static block?

3) When is a non-static block executed in relation to the constructor?

4) Can we have multiple non-static blocks in the same class? If yes, in what order do they execute?

5) Can a non-static block access static and non-static variables? Explain.

6) Can a non-static block call methods (both static and non-static)?

7) What happens if both a parent and child class have non-static blocks? In which order are they executed during object creation?

8) Can we use the this keyword inside a non-static block? Why or why not?

9) Can a non-static block throw exceptions or use try–catch?

10) What is the real purpose of using non-static blocks in Java programs (practical use cases)?

Programs

1) Write a simple program with one non-static block and a constructor to show the order of execution.

2) Write a program that shows the difference between static and non-static blocks during class loading and object creation.

3) Create a non-static block that initializes instance variables before the constructor executes.

4) Demonstrate how non-static blocks are executed in an inheritance hierarchy (both parent and child having blocks).

5) Write a program combining static blocks, non-static blocks, constructors, and methods to show their complete execution flow.

12(A). UNDERSTANDING STATIC KEYWORD

1. Static Variable

Interview Questions

1. What is a static variable?
2. How does a static variable differ from an instance variable?
3. When and why do we use static variables?
4. How many copies of a static variable exist in memory?
5. Can a static variable be local to a method?
6. Can we access static variables without creating an object?
7. What are some real-world examples of using static variables?
8. What happens if two classes have static variables with the same name?
9. Can we change the value of a static variable in another class?
10. Is it possible to make a static variable final?

Programs

1. Count the number of objects created using a static variable
2. Demonstrate that static variables are shared between all objects.
3. Use a static variable for generating a unique ID for every object.

2. Static Method

Interview Questions

1. What is a static method in Java?

2. Can a static method access instance variables? Why or why not?
3. Can we call a static method using an object?
4. What happens if we declare a static method with the same signature in a subclass?
5. Why can't we use this keyword inside a static method?
6. Can we overload static methods?
7. Can we override static methods? Explain with example.
8. What is the difference between a static method and an instance method?
9. When should you make a method static?
10. Can a static method call another static method?

Programs

1. Create a static method to find factorial of a number.
2. Demonstrate static method calling another static method
3. Check whether a number is even or odd using static method
4. Program to access static variable inside a static method

3. Static Method VS Instance Method

Interview Questions

1. What is the main difference between static and instance methods?
2. Can a static method call an instance method directly?
3. Can an instance method call a static method?
4. How does memory allocation differ for static and instance methods?
5. Give a real-world example where you'd use a static vs an instance method.

Programs

1. Show difference between static and instance methods
2. Demonstrate calling instance method from static method using object
3. Show that static methods cannot directly access instance variable.

4. Static Block

Interview Questions

1. What is a static block in Java?
2. When is a static block executed?

3. How many times does a static block execute?
4. What is the purpose of a static block?
5. Can we have multiple static blocks in one class?
6. What is the order of execution of static block, constructor, and main method?
7. Can a static block throw an exception?
8. Can we use return statement inside a static block?
9. Can we initialize static variables inside a static block?
10. Is it possible to load a class without running its main method?

Programs

1. Demonstrate order of static block and main method.
2. Initialize static variables inside static block.
3. Demonstrate multiple static blocks.
4. Program showing static block executes even before object creation.

5. Common Combined Concept

Interview Questions:

---

1. What is static in Java?
2. Why do we use the static keyword in Java?
3. Where can we use the static keyword?
4. What is a static variable in Java?
5. What is a static method in Java?
6. What is a static block?
7. Can we override a static method?
8. Can we overload a static method?
9. Why is the Java main method static?
10. Can we declare multiple static blocks in our code?
11. Can we call a superclass static method in a subclass?
12. Can a constructor be static in Java?
13. Can we access non-static data members in a static method in Java?
14. Can we access static data members in a static method?
15. Can we use this and super in a static context?
16. Can we write static public void main(String args[])?

17. Can an abstract class have a static variable in it?
18. Can we execute a Java program without the main() method?
19. Can we apply the static keyword to an outer class?
20. Can we create a static inner class?
21. What is the advantage of using a static variable?
22. Can we declare a local variable as static?
23. Why is the combination of abstract and static not allowed for methods?
24. Explain *Illegal forward reference* error in a static block with an example.
25. What is the order of execution of static members in a single class?
26. What is the order of execution of static members in super and subclass?
27. Demonstrate how to declare a static variable.
28. How to define a static block and observe when the static block will execute?
29. How to define a static method and how to invoke it in the same class and different class?
30. How to differentiate static and non-static members?
31. Draw a memory diagram for static members showing where they get memory in the JVM.

Programs

1. Program to show the execution sequence of static block, main method, and constructor
2. Create a utility class using only static methods and variables.
3. Static method returning object of its class.
4. Count total login attempts using static variable and method.
5. Program to show static variable shared across objects but instance variable not.
6. Program to demonstrate static block execution order in inheritance.

12(B). STATIC KEYWORD (LOGICAL SESSION)

Logical Programs on static

---

1. What is the output of the program?

```
public class Test1
```

```
{
        public static void main(String[] args)
{
                int x = 20;
                System.out.println("Main method: "+x);
        }
        static
{
                int x=10;
                System.out.println("Static block: "+x);
        }
}
```

2. What is the output of the program?

```
public class Test2
{
        Static
{
                System.out.println("Test2 class Static Block");
        }
        public static void main(String[] args)
{
                System.out.println("Test2 class main method");
        }
}
```

3. What is the output of the program?

```
public class Test3 {

        public static void main(String[] args) {

                int a ;

                System.out.println("A :"+ a);

        }

}
```

4. Can we declare local variable as static if not what is the error?

```
public class Test4 {

        public static void main(String[] args) {

                static int x = 10;

                System.out.println(x);

        }

}
```

5. Define a class having one static variable say static int x=10 and invoke x variable in m1() and main(-) what is the output?

```
public class Test5

{

        static int x = 10;

        void m1()

    {

                System.out.println("m1 X :"+x);

        }

        public static void main(String[] args)

    {
```

```
            System.out.println("main X :"+x);
        }
    }
```

6. Is it possible to access non-static variable in static method directly if not what is the error?

```
public class Test6
{
        int a = 10;
        public static void main(String[] args)
       {
            System.out.println("A Value :"+a);
        }
}
```

7. What are the different ways are there to call static variables?

```
public class Test7
{
        //static variable a
        static int a = 20;
        public static void main(String[] args) {
            //write code here to access a variable with diff ways
        }
}
```

8. What are the different ways to call/invoke static methods from main method.

```
public class Test8
{
        static void m1()
```

```
        {
                System.out.println("Test8 : m1() called");
        }
        public static void main(String[] args)
      {
                //write code here
        }
}
```

9. What is the output of the below program?

```
public class Test9
{
        static void m1()
       {
                System.out.println("Test9 : m1()");
        }
        static void m2() {
                System.out.println("Test9 : m2()");
        }
       public static void main(String[] args) {
                m1();
                m2();
        }
}
```

10. What is the output of the below program?

```
public class Test10 {
```

```
	static {
		System.out.println("SB1");
	}
	static void m1() {
		System.out.println("m1()");
	}
	public static void main(String[] args) {

		Test10.m1();
	}
}
```

11. Example on execution order of static members in same class?

```
public class Test11 {
	static int x = 10;
	static int y = 20;
	static {
		System.out.println("---Test11 SB1 called---");
		System.out.println("X: "+x);
		System.out.println("Y: "+y);
	}
	static void m1() {
		System.out.println("---Test11 m1() called---");
		System.out.println("X: "+x);
		System.out.println("Y: "+y);
	}
```

```
        public static void main(String[] args) {
                m1();
        }
        static {
                System.out.println("---Test11 SB2 called---");
                System.out.println("X: "+x);
                System.out.println("Y: "+y);
        }
}
```

12. Can we call static method inside static block if yes, what is the output?

```
public class Test12 {
        static {
                System.out.println("Test12 : SB1 called");
                m1();
        }
        static void m1() {
                System.out.println("Test12 : m1() called");
        }
        public static void main(String[] args) {
                //no operation
        }
}
```

13. Is main method mandatory to execute java program if yes, what is the error?

```
public class Test13 {
        static {
```

```
            System.out.println("Test13 : SB1 called");
        }
        static void m1() {
            System.out.println("Test13 : m1() called");
        }
}
```

14. What is the output of the below program?

```
public class Test14 {
        static int x = 100;
        static {
            System.out.println("X :"+x);
            System.exit(0);
        }
}
```

15. What is the output of the below program?

```
class Test15Super{
        int x = 10;
        static void m1() {
            System.out.println("Test15Super : m1()");
        }
        static {
            System.out.println("Test15Super : SB1 called");
        }
}
public class Test15Sub extends Test15Super{
```

```
        static int x = 20;
        static {
                System.out.println("Test15Sub : SB1 called");
        }
        static void m2() {
                m1();
                System.out.println("Test15Sub : m2()");
        }
        public static void main(String[] args) {
                m2();
        }
}
```

16. What is the output of the below program?

```
public class Test16 {
        int x = 10;
        static int a = 20;
        //non-static block
        {
                System.out.println("Test16 : NSB1 called");
                System.out.println("X: "+x);
                System.out.println("A: "+a);
        }
        //static block
        static {
```

```
            System.out.println("Test16 : SB1 called");
            System.out.println("X: "+x);
            System.out.println("A: "+a);
        }
        //main method
        public static void main(String[] args) {
            //no operation
        }
}
```

17. What is the output of the below program?

```
class Test17Super{
        static void m2() {
            System.out.println("Test17Super : m2()");
        }
}
public class Test17Sub extends Test17Super{
        public static void m1() {
            System.out.println("Test17Sub : m1()");
        }
        public static void m2() {
            System.out.println("Test17Sub : m2()");
        }
        public static void main(String[] args) {
            Test17Sub.m1();
        }
```

```
}
```

18. What is the output of the below program?

```
public class Test18 {

        static int a = 20;

        public static void main(String[] args) {

                Test18 t1 = new Test18();

                Test18 t2 = new Test18();

                System.out.println("A: "+a);

                System.out.println("t1.A: "+t1.a);

                System.out.println("t2.A: "+t2.a);

                t1.a=30;

                System.out.println("t1.A: "+t1.a);

                System.out.println("t2.A: "+t2.a);

                t2.a=40;

                System.out.println("t1.A: "+t1.a);

                System.out.println("t2.A: "+t2.a);

        }

}
```

19. What is the output of the below program?

```
public class Test19 {

        int x = 10;

        {

                System.out.println("Test19:NSB1");

        }

        void m1() {
```

```
            System.out.println("Test19:m1()");
        }
        public static void main(String[] args) {
            //no operation
        }
}
```

20. What is the output of the below program?

```
public class Test20 {
        static int x = 100;
        static {
            x = 200;
            System.out.println("SB X: "+x);
        }
        static void m1() {
            x = 300;
            System.out.println("m1 X:"+x);
        }
        public static void main(String[] args) {
            System.out.println("mian X: "+x);
        }
}
```

21. What is the output of the below program?

```
public class Test21 {
        static int x = m1();
        static int m2() {
```

```
        System.out.println("Test21:m2() called");
        return 10;
    }
    static int m1() {
        System.out.println("Test21:m1() called");
        return 20;
    }
    public static void main(String[] args) {
        System.out.println("X: "+x);
    }
}
```

22. What is the output of the below program?

```
class Test22Super{
    static int x = m1();
    static int m2() {
        System.out.println("Test22:m2() called");
        return 10;
    }
}
public class Test22Sub extends Test22Super{
    static int x;
    static int m1() {
        System.out.println("Test22:m1() called");
        return 20;
    }
```

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

23. What is the output of the below program?

```
public class Test23 {
        static {
                System.out.println("SB1 X: "+x);
        }
        static int x = 10;
        static {
                x=20;
                System.out.println("SB2 X:"+x);
        }
        public static void main(String[] args) {
                // no-operation
        }
}
```

24. What is the output of the below program?

```
public class Test24 {
        static int a = 20;
        static {
                System.out.println("SB1: "+Test24.a);
                System.out.println("SB1: "+a);
        }
```

```
        static {
                System.out.println("SB2: "+Test24.b);
        }
        static int b = 30;
        public static void main(String[] args) {
                System.out.println("A: "+a);
                System.out.println("B: "+b);
        }
}
```

25. What is the output of the below program?

```
public class Test25 {
        public static void main(String[] args) {
                System.out.println("Welcome To Java");
        }
}
```

26. What is the output of the below program?

```
public class Test26 {
        //empty….
}
```

27. What is the output of the below program?

```
public class Test27 {
        public static void main(String[] args) {
                //empty…
        }
}
```

28. What is the output of the below program?

```
public class Test28 {

	static {

		System.out.println("Static Block....");

	}

	public static void main(String[] args) {

		System.out.println("Main Method....");

	}

}
```

29. What is the output of the below program?

```
public class Test29 {

	public static void main(String[] args) {

		System.out.println("Main Method....");

	}

	static {

		System.out.println("Static Block...");

	}

}
```

30. What is the output of the below program?

```
public class Test30 {

	static {

		System.out.println("Static Block1...");

	}

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

```
        System.out.println("Main Method....");
    }
    static {
        System.out.println("Static Block2...");
    }
}
```

31. What is the output of the below program?

```
public class Test31 {
    public static void main(String[] args) {
        System.out.println("Main Method...");
    }
    static void fun() {
        System.out.println("User Defined Method...");
    }
}
```

32. What is the output of the below program?

```
public class Test32 {
    public static void main(String[] args) {
        System.out.println("Main Method Starts...");
        Example8.fun();
        System.out.println("Main Method Ends...");
    }
    static void fun() {
        System.out.println("User Defined method...");
    }
```

```
}
```

33. What is the output of the below program?

```
public class Test33 {

	public static void main(String[] args) {

		System.out.println("Main Method starts...");

		Example9.m1();

		System.out.println("Main Method Ends...");

	}

	static void m1() {

		System.out.println("User Defined Method...");

	}

	static {

		System.out.println("Static Block Starts...");

		Example9.m1();

		System.out.println("Static Block Ends...");

	}

}
```

34. What is the output of the below program?

```
public class Test34 {

	static void m1() {

		System.out.println("Control in m1 method");

	}

	public static void main(String[] args) {

		System.out.println("Main Method Starts...");

		System.out.println("Callig m2 method");
```

```
		Example10.m2();
		System.out.println("Control Back to main from m2 method...");
		System.out.println("Main Method Ends...");
	}

	static void m2() {
		System.out.println("Control in m2 method");
		System.out.println("Calling m1 method");
		Example10.m1();
		System.out.println("Control Back to m2 from m1 method..");
		System.out.println("m2 method ends...");
	}
	static {
		System.out.println("Static Block Starts...");
		System.out.println("Calling m1 method");
		Example10.m1();
		System.out.println("Control Back to Static Block from m1");
		System.out.println("Static Block Ends...");
	}
}
```

35. What is the output of the below program?

```
public class Test35 {
	public static void main(String[] args) {
		int a;
		System.out.println(a);
```

```
        a=10;
        System.out.println(a);
    }
}
```

36. What is the output of the below program?

```
public class Test36 {
    static int x = 20;
    static {
        int x=10;
        System.out.println(x);
    }
    public static void main(String[] args) {

        System.out.println(x);
    }
}
```

37. What is the output of the below program?

```
public class Test37 {
    static {
        System.out.println("Static Block");
        System.out.println(Example13.a);
    }

    public static void main(String[] args) {
        System.out.println("Main Method");
```

```
            System.out.println(a);
        }
        static int a = 50;
}
```

38. What is the output of the below program?

```
public class Test38 {
        static int x = 100;
        public static void main(String[] args) {
            System.out.println(Example14.x);
            System.out.println(x);
        }
}
```

39. What is the output of the below program?

```
public class Test39{
        static int x = 100;

        public static void main(String[] args) {
            int x=200;
            System.out.println(Example15.x);
            System.out.println(x);
        }
}
```

40. What is the output of the below program?

```
public class Test40{
```

```
        static int x = 50;
        public static void main(String[] args) {
                int x = 60;
                Example16.x = Example16.x + x;
                System.out.println(Example16.x);
                System.out.println(x);
        }
}
```

41. What is the output of the below program?

```
public class Test41 {
        static {
                int a = 10;
                System.out.println("Class Level A: "+Example17.a);
                Example17.a = Example17.a + a;
        }
        static int a = 20;
        public static void main(String[] args) {
                System.out.println("Main Method: "+Example17.a);
        }
}
```

42. What is the output of the below program?

```
public class Test42 {
        static {
                int a = 50;
                Example18.a = Example18.a+a;
```

```
        a = a+Example18.a;
        Example18.a=a+a;
    }
    static int a = 60;
    public static void main(String[] args) {
        System.out.println("Class Level A: "+Example18.a);
    }
}
```

43. What is the output of the below program?

```
public class Test43 {
    static {
        int x=10;
        Example19.x = x + x;
    }
    static int x;
    public static void main(String[] args) {
        System.out.println("Class Level X: "+Example19.x);
    }
    static {
        x = x + Example19.x;
    }
}
```

44. What is the output of the below program?

```
public class Test44 {
    static int a;
```

```
        public static void main(String[] args) {
                System.out.println(Example20.a);
                Example20.a = Example20.initialize();
                System.out.println(Example20.a);
        }
        static int initialize() {
                Example20.a = 50;
                return 60;
        }
}
```

45. What is the output of the below program?

```
public class Test45 {
        static int x = Example21.m1();
        public static void main(String[] args) {
                System.out.println(Example21.x);
        }
        static int m1() {
                Example21.x=10;
                return Example21.m2();
        }
        static int m2() {
                System.out.println(Example21.x);
                return 20;
        }
}
```

46. What is the output of the below program?

```
public class Test46 {

        static int x = 10;

        public static void main(String[] args) {

                System.out.println(Example22.m1() + Example22.x);

        }

        static int m1() {

                Example22.x = Example22.x + 100;

                return Example22.x;

        }

}
```

47. What is the output of the below program?

```
public class Test47 {

        static int x = m1();

        public static void main(String[] args) {

                System.out.println(Example23.x);

        }

        static {

                System.out.println(x);

                Example23.x = x+20;

        }

        static int m1() {

                Example23.x = 50;

                return m2();

        }
```

```
        static int m2() {
                System.out.println(Example23.x);
                return 100;
        }
}
```

48. What is the output of the below program?

```
public class Test48 {
        static {
                Example24.a = m1();
        }
        static int a = 50;

        public static void main(String[] args) {
                System.out.println(a);
        }

        static {
                Example24.a = Example24.a + m1();
        }

        static int m1() {
                Example24.a = 30;
                return m2();
        }
```

```
        static int m2() {
                System.out.println(a);
                return Example24.a + 20;
        }
}
```

49. What is the output of the below program?

```
public class Test49 {
        static int a = 50;

        public static void main(String[] args) {
                int a = 60;
                a = a;
                System.out.println(a);
                System.out.println(Example25.a);
        }
}
```

50. What is the output of the below program?

```
public class Test50 {
        public static void main(String[] args) {
                int x=20;
                System.out.println(x);
        }
        static {
                int x=10;
                System.out.println(x+" ");
```

```
        }
}
```

51. What is the output of the below program?

```
public class Test51 {
        int x = 10;

        public static void main(String[] args) {
                System.out.println(x);
        }
        static {
                System.out.println(x+”  “);
        }
}
```

52. What is the output of the below program?

```
public class Test52 {
        int x = 10;

        public static void main(String[] args) {

                Test52 t1 = new Test52();
                System.out.println(t1.x);
        }
        static {
                int x=20;
                System.out.println(x+" ");
```

```
        }
}
```

53. What is the output of the below program?

```
public class Test53 {
        int x = 10;
        public static void main(String[] args) {
                System.out.println(Test53.x);
        }
        static {
                int x = 20;
                System.out.println(x+" ");
        }
}
```

54. What is the output of the below program?

```
public class Test54 {

        static int x = 10;
        public static void main(String[] args) {
                Test54 t1 = new Test54();
                Test54 t2 = new Test54();

                t1.x = 20;
                System.out.print(t1.x+" ");
                System.out.println(t2.x);
```

```
        }
}
```

55. What is the output of the below program?

```
public class Test55 {


        static int i = 1;
        public static void main(String[] args){
                for(int i=1; i<10; i++) {
                        i=i+2;
                        System.out.print(i+" ");
                }
        }
}
```

56. What is the output of the below program?

```
public class Test56 {
        static int i = 1;
        public static void main(String[] args) {
                int i = 1;
                for(Test56.i = 1; Test56.i<10; Test56.i++) {
                        i=i+2;
                        System.out.print(i+" ");
                }
        }
}
```

57. What is the output of the below program?

```
public class Test57 {

	static int i = 1;


	public static void main(String[] args) {

		static int i = 1;

		for(Test57.i = 1; Test57.i<10; Test57.i++) {

			i=i+2;

			System.out.print(i+" ");

		}

	}

}
```

58. What is the output of the below program?

```
public class Test58 {

	public static void main(String[] args) {

		int arr1[] = {11, 22, 33};

		int arr2[] = {11, 22, 33, 44, 55};

		int ptr[];


		ptr=arr1;

		arr1=arr2;

		arr2=ptr;

		System.out.print(arr1.length+" ");

		System.out.println(arr2.length);

	}

}
```

59. What is the output of the below program?

```
public class MyClass {

        static int x;

        public MyClass() {
                method();
                method2();
        }
        public void method() {
                x+=3;
                MyClass.x = MyClass.x-2;
        }

        public void method2() {
                x+=5;
                this.x = x*x+2;
        }
        public static void main(String[] args) {
                MyClass mc = new MyClass();
                MyClass mc2 = new MyClass();
                MyClass mc3 = new MyClass();
                System.out.println(mc2.x+"-"+mc3.x);
        }
}
```

60. What is the output of the below program?

```
public class StaticVariableExample {

	static int a = 10;

	public static void main(String[] args) {

		StaticVariableExample s1 = new StaticVariableExample();

		StaticVariableExample s2 = new StaticVariableExample();

		System.out.println("s1.a value: "+s1.a);

		System.out.println("s2.a value: "+s2.a);

		//Change s1 a value alone

		s1.a=20;

		System.out.println("s1.a value: "+s1.a);

		System.out.println("s2.a value: "+s2.a);

	}

}
```

61. What is the output of the below program?

```
public class Test {

	public static void main(String[] args) {

		welcome();

	}

	void welcome() {

			System.out.println("Welcome to java");

		}

	static void welcome() {

		System.out.println("Welcome to java");

	}
```

```
}
```

---

13. JAVA TYPECASTING

Interview Questions

1. What is typecasting in Java, and why is it used?
2. Explain the difference between widening (implicit) and narrowing (explicit) typecasting, with examples.
3. Can a double be stored in an int variable? How? What happens if the value exceeds the int range?
4. What is type promotion in Java, and how does it work when different data types are used together?
5. Can an int be cast to a byte? What happens if the value is too large?
6. What is upcasting in Java? Provide an example with parent and child classes.
7. What is downcasting in Java? Why is it considered risky? How can it be done safely?
8. What is a ClassCastException? Give an example scenario.
9. Can Object references be cast to their actual object type? Show an example.
10. Can interface references be upcast to their implementation class? Explain.
11. How does upcasting affect overridden methods in Java?
12. What happens when you cast a double to a float?
13. Can null references be safely downcast? Explain.
14. Can boolean be cast to int? Why or why not?
15. Can you assign a child class object to a parent class reference? What are the rules?
16. How does typecasting affect memory allocation and storage for primitives vs reference types?

Code Snippets

1)

```
public class Test1 {
   public static void main(String[] args) {
      int i = 100;
      double d = i;
      System.out.println(d);
   }
```

```
}
```

2)

```
public class Test2 {
   public static void main(String[] args) {
      double d = 99.99;
      int i = (int)d;
      System.out.println(i);
   }
}
```

3)

```
public class Test3 {
   public static void main(String[] args) {
      char c = 'A';
      int i = c;
      System.out.println(i);
   }
}
```

4)

```
public class Test4 {
   public static void main(String[] args) {
      int i = 130;
      byte b = (byte)i;
      System.out.println(b);
   }
}
```

5)

```
class Parent {}
class Child extends Parent {}
public class Test5 {
```

```
    public static void main(String[] args) {
        Child c = new Child();
        Parent p = c;
    }
}
```

6)

```
class Parent1 {}
class Child1 extends Parent1 {}
public class Test6 {
    public static void main(String[] args) {
        Parent1 p = new Child1();
        Child1 c = (Child1)p;
    }
}
```

7)

```
class Parent2 {}
class Child2 extends Parent2 {}
class Unrelated {}
public class Test7 {
    public static void main(String[] args) {
        Parent2 p = new Parent2();
        // Child2 c = (Child2)p; // ClassCastException
    }
}
```

8)

```
public class Test8 {
    public static void main(String[] args) {
        int a = 10;
        Integer b = 20;
        System.out.println(a + b);
```

```
    }
}
```

9)

```
public class Test10 {
    public static void main(String[] args) {
        byte b1 = 10;
        byte b2 = 20;
        byte sum = (byte)(b1 + b2);
        System.out.println(sum)
}
}
```

10)

```
public class Test15 {
    public static void main(String[] args) {
        Object obj = "Hello";
        String s = (String)obj;
        System.out.println(s);
    }
}
```

11)

```
public class Test16 {
    public static void main(String[] args) {
        Object[] objArr = new String[2];
        objArr[0] = "Java";
        // objArr[1] = 10; // ArrayStoreException
    }
}
```

12)

```
class A {}
```

```
class B extends A {}
public class Test19 {
   public static void main(String[] args) {
      A a = new A();
      // B b = (B)a; // ClassCastException
   }
}
```

---

14. JAVA WRAPPER CLASSES, AUTOBOXING & UNBOXING

Interview Questions

1) What is a wrapper class in Java, and why do we need it?
2) List all wrapper classes in Java for each primitive type.
3) How is a wrapper class different from a primitive type? Give examples.
4) Can a wrapper object store null? Give an example.
5) What is autoboxing / unboxing in Java? Diff b/w autoboxing and unboxing?
6) Can we perform arithmetic operations directly on wrapper objects? How does it work?
7) What is the difference between == and .equals() when comparing wrapper objects?
8) How does the caching mechanism work for Integer and Boolean?
9) What happens if you try to unbox a null wrapper object?
10) How does autoboxing affect performance in Java programs?
11) Can we declare wrapper objects as final? What happens if we try to change their value?
12) Can autoboxing occur when passing arguments to a method? Give an example.
13) What is the difference between Integer.parseInt() and Integer.valueOf()?
14) What is the behavior of null in wrapper comparisons?
15) Can wrapper objects be synchronized for thread safety?
16) Are wrapper objects immutable? How does Java handle immutability for wrappers?

Programs

```
public class Test2 {
    public static void main(String[] args) {
        Integer a = null;
        // int b = a;  // What happens here?
    }
}

public class Test3 {
    Integer a = 10;
    int b = 5;
    public void calculate() {
        System.out.println(a + b);
    }
}

public class Test4 {
    public static void main(String[] args) {
        Integer a = 100;
        Integer b = 100;
        System.out.println(a == b);
        System.out.println(a.equals(b));
    }
}

public class Test5 {
    public static void main(String[] args) {
        Integer a = 127;
        Integer b = 127;
        Integer x = 128;
        Integer y = 128;
        System.out.println(a == b);
        System.out.println(x == y);
    }
```

```
}

import java.util.ArrayList;
public class Test8 {
    public static void main(String[] args) {
        ArrayList<Integer> list = new ArrayList<>();
        list.add(10);
        list.add(20);
        System.out.println(list);
    }
}

public class Test9 {
    public static void main(String[] args) {
        Integer a = null;
        System.out.println(a == null);
    }
}

public class Test12 {
    public static void main(String[] args) {
        Object obj = Integer.valueOf(10);
        Integer a = (Integer)obj;
        System.out.println(a);
    }
}
public class Test16 {
    public static void main(String[] args) {
        final Integer a = 10;
        // a = 20; // Can you reassign? What happens?
    }
}
```

```
public class Test17 {
    public static void main(String[] args) {
        Long a = 1000;
        int b = a;
        System.out.println(b);
    }
}
public class Test19 {
    public static void print(int a) { System.out.println("Primitive: " + a); }
    public static void print(Integer a) { System.out.println("Wrapper: " + a); }
    public static void main(String[] args) {
        Integer num = 10;
        print(num);
    }
}
```

---

15. CONTROL STATEMENTS

1. Selection Statements (if, if-else, if-else-if, switch)

Interview Questions

1. What are selection statements in Java?
2. Difference between if and if-else.
3. Can we use multiple if statements without else?
4. What is an if-else-if ladder, and how does it work?
5. When should you use switch instead of if-else?
6. What happens if you forget to use a break in a switch case?
7. Can we use logical operators (&&, ||) in a switch expression?
8. Difference between switch expression (Java 12+) and old switch statement.
9. Can we have duplicate case values in a switch?

10. What happens if no case matches and no default is present?
11. Can we use a switch with wrapper classes like Integer?
12. Can a switch statement work on enum types?
13. What is the fall-through behavior in a switch?

2. Logical programs on (if, if-else, if-else-if)

Interview Questions

1) Write a program to find the greatest of two numbers using if-else.
2) Write a program to find the greatest of three numbers using if-else-if.
3) Check if a number is positive, negative, or zero.
4) Write a program to check if a given year is a leap year.
5) Check if a character is a vowel or consonant using switch.
6) Check if a number is even or odd using if-else.
7) Find the grade of a student using if-else-if (based on marks).
8) Write a program to find the largest of three numbers using nested if.
9) Program to check if a number is divisible by both 3 and 5.
10) Write a calculator program using switch (add, sub, mul, div).

3. Understanding switch case

Interview Questions

1) What is a switch statement in Java?
2) What is the syntax of a switch statement?
3) What is the purpose of the break keyword in a switch?
4) What happens if break is omitted in a switch-case?
5) Can we use multiple statements inside a case block?
6) What is the purpose of the default case? Is the default case mandatory in a switch?
7) Can default be written anywhere in the switch block?
8) How does Java decide which case to execute?
9) Can two case labels have the same constant value?
10) Which primitive data types are supported in switch?
11) How is a switch expression different from a switch statement?

4. Iteration Statements (while, do-while, for, for-each)

Interview Questions

1) What are iteration statements in Java?
2) Difference between for, while, and do-while loops.
3) Which loop is entry-controlled and which is exit-controlled?
4) Explain the syntax of a for, while ,do-while ,for -each loop in Java.
5) Can we have an infinite loop? How to create it by using for, while Give an example.
6) What is the difference between for and for-each loop?
7) When is a for loop preferred over a while loop?
8) When is a do-while loop used?
9) How do break and continue work in loops?
10) What is the difference between break and return inside a loop?
11) What happens if we put a continue statement in the last line of a loop?
12) Can loops be replaced with recursion? When is that useful?
13) When should we use a for loop, a while loop, or a for-each loop in Java?

5. Jumping Statements (break, continue, return)

Interview Questions

1) What are jump statements in Java?
2) What is the purpose of the break statement?
3) What is the purpose of the continue statement?
4) What is the purpose of the return statement?
5) Can we use break without a loop or switch?
6) Can we use continue in a switch statement? Why or why not?
7) What is the difference between break and continue / return?
8) Can break terminate a method in Java?
9) What will happen if you write multiple break statements in nested loops?

Programming Questions

1) Write a program to check Armstrong number (e.g., 153 → $1^3+5^3+3^3=153$).
2) How do you check if a number is a Strong Number using loops in Java?
3) How do you check if a number is a Perfect Number using loops in Java?
4) Write a program to find the factorial of a given number using a loop.?

5) Write a program to reverse a number.

6) Write a program to check if a number is palindrome or not.

7) Write a program to check if a number is prime or not.

8) Find the Nth prime number using nested loops.

9) Write a program to print Fibonacci series up to N terms.

10) Write a program to find LCM and GCD of two numbers using loops.

11) Write a program to find smallest and largest digit in a given number.

12) Check if the digits of a number are in increasing order?(1234 ✓, 1324 ✗)

13) Find the frequency of each digit in a number using loops?

14) Remove all zeros from a number (e.g., 10204 → 124)?

6. Pattern Printing (Nested Loop Programs)

Interview Questions

1) Print inverted right triangle pattern of *.
2) Print pyramid pattern of *.
3) Print inverted pyramid of *.
4) Print half pyramid with numbers.
5) Print Floyd's triangle.
6) Print Pascal's triangle.
7) Print diamond pattern of *.
8) Print hollow square pattern using *.
9) Print hollow pyramid pattern (outline only)
10) Print right-angled triangle pattern of *.

---

16. SCANNER CLASS

Interview Questions:

1. What is the Scanner class in Java and Which package contains the Scanner class?
2. How do you create a Scanner object for keyboard input?
3. How do you read a string using Scanner?

4. How do you read a complete line using Scanner?
5. Difference between next() and nextLine()?
6. How do you read integers, floats, doubles, and booleans?
7. How do you read a single character using Scanner?
8. What happens if user enters wrong data type?
9. What is InputMismatchException? How to avoid InputMismatchException?
10. What is the default delimiter of Scanner? How to change the delimiter in Scanner?
11. Why should Scanner be closed?
12. Can you reuse Scanner for another input source?
13. What is the purpose of System.in in Scanner?
14. Difference between Scanner and BufferedReader?
15. Why does Scanner skip input after reading numbers?
16. What happens if you try to read from a closed Scanner?
17. Can Scanner read binary data?

Programming Questions

1. Sum of N numbers – Read N integers and print their sum.
2. Maximum and Minimum number – Read 5 integers and print the max and min.
3. Average of floating-point numbers – Read N doubles in one line and calculate the average.
4. Reverse a string – Read a line of text and print it reversed.
5. Count words in a sentence – Read a line and count the number of words.

---

17. PACKAGES IN JAVA

Interview Questions

1) What is a package in Java, and why do we need it?
2) Explain the benefits of using packages in Java.
3) How do packages help in organizing large projects?

4) Can a class exist without a package? Explain.
5) Difference between `package` and `import` statements.
6) What is the difference between default package and user-defined packages?
7) How do packages improve code reusability and accessibility?
8) Explain real-time use cases where packages are essential.?

Structure of a Package

Interview Questions

1) What is the structure of a Java package?
2) Explain the difference between top-level and sub-packages.
3) What are naming conventions for packages?
4) How does the JVM locate classes inside a package?
5) Explain the difference between .class files in default and user-defined packages.

Built-in Packages

Interview Questions

1) What are built-in packages in Java? Give examples.
2) Explain the difference between java.lang and java.util packages.
3) Which built-in package is automatically imported in every Java program?
4) Can you use a class from a built-in package without import? Explain.

Package Naming Rules

Interview Questions

1) What are the naming rules for Java packages?
2) Can a package name start with a number or special character?
3) Why is lowercase recommended for package names?
4) Explain the use of reverse domain name convention in package naming.
5) Can two packages have the same name in Java? What happens?
6) What are the restrictions on special characters in package names?

How to Access Package from Another Package

Interview Questions

1) How do you access a class from one package in another package?

2) Difference between using import package.* and import package.ClassName.

3) Can private classes be accessed from another package?

---

18. ACCESS MODIFIERS

1. Class level access modifier

Interview Questions

1. What are access modifiers in Java, what are their types, and what is their purpose?
2. Which access modifiers are allowed for top-level classes, and why can't a top-level class be protected or private?
3. What does the public modifier mean for a class or interface, and how does it affect accessibility across packages?
4. What does it mean for a class to have default (package-private) access, and can such a class be accessed from another package?
5. How do access modifiers apply to constructors, and what impact do they have on object creation and inheritance?
6. How do access modifiers support encapsulation and data hiding in Java?
7. How do access modifiers differ between class level and member level?
8. Can we declare a private constructor inside an abstract class? Or can a class be both abstract and final at the same time?

2. Method level access modifier

Interview Questions

1. What is the default access modifier for a method, and how does it differ from public, private, and protected access?

2. Can a constructor or a method be declared private or protected? In what scenarios is this useful?

3. Why do we use private methods inside a class, and can a private method be accessed or overridden in subclasses?

4. Can a protected method be accessed or overridden in another package? How does inheritance affect this?
5. Can you reduce or increase the visibility of an overridden method (e.g., public → protected)? What happens if you try to?
6. Can a default/package-private, final, or static method be overridden or hidden in another package? Explain the difference.
7. Do access modifiers affect method overloading? Can you overload methods with different access levels?
8. Can a private static method be overloaded in the same class? How is it different from overriding?
9. Can an abstract class have a private abstract method? Why or why not? Can you declare a private synchronized method?
10. Can an interface's default method be private or protected?

3.Variable level access modifiers

Interview Questions

1. What are the types of variables in Java (local, instance, static)? What access modifiers can be applied to each, and what is the default access modifier for instance variables?
2. What's the difference between local, instance, and static variables in terms of scope, lifetime, and storage? Can local variables have access modifiers? Why or why not?
3. What is a final variable**?** Can it be static or transient**?** What happens if you try to reassign it?
4. What is a static variable**?** How does it differ from an instance variable? Can it be accessed using an object reference**,** and when/where is it initialized and stored?
5. Can a constructor modify a static variable? Can we use the this keyword with static variables? What happens if multiple objects change a static variable?
6. What is a blank final variable and a static blank final variable? Where should each be initialized?
7. What is a transient variable? What happens to transient variables during serialization?
8. What is a volatile variable? When should it be used, how does it differ from?

Programming Questions:

1. Write code to prove that a default class cannot be accessed outside its package.
2. Demonstrate that a private constructor restricts object creation outside its class.
3. Write code to show that final methods cannot be overridden.
4. Create a program to show visibility changes in method overriding (protected → public).
5. Demonstrate method hiding using a static method.
6. Create a Singleton class and test multiple instance creation.

7. Show that abstract + final combination on a class is invalid.
8. Write code to access protected members across packages using inheritance.
9. Show that default methods cannot be accessed outside package even with import.
10. Prove that private static methods are not accessible through object references

---

19. USES OF THIS KEYWORD

Interview Questions

1. What is the this keyword in java? What is the main purpose of this in a class?
2. How is this used to refer to current object?
3. How can this be used to differentiate instance variables from local variables?
4. How is this used to call another constructor in the same class? (this() constructor chaining)
5. Can this be used in a static method? Why or why not?
6. Can this be passed as an argument to a method or constructor? Give an example.
7. How does this help in method chaining?
8. Difference between this and current class name for object reference.

Use of Super keyword

Interview Questions

1. What is the super keyword in Java?
2. How is super used to refer to immediate parent class object?
3. How can super be used to call parent class methods?
4. How can super be used to access parent class variables when hidden by subclass?
5. How is super() used to call parent class constructors?
6. Can super() and this() be used together in a constructor? Why or why not?
7. Can super be used in a static context? Why or why not?
8. Difference between super and this in terms of constructor chaining.
9. How does super help in method overriding to call parent class version of method?

Programs

1. Write a program demonstrating this to resolve variable shadowing?

2. Write a program using this() to chain constructors?
3. 3.Write a program using super to call parent constructor.
4. Write a program using this in inner classes to access outer class instance.

---

20. ARRAYS

Declaring, Instantiating an Array

Interview Questions

1) How do you declare an array in Java? What are the three ways to declare an array?
2) How do you instantiate an array?
3) Can you declare and instantiate an array in a single line?
4) Can an array have zero size?
5) Can you declare an array without specifying size? and is it valid to instantiate an array with a negative size in Java?
6) How is an array object stored in memory?
7) What is the difference between stack and heap in context of arrays?
8) Can array size be changed after instantiation?
9) Difference between new int[5] and int[]arr= {1,2,3,4,5}
10) Can you store different data types in an array?
11) Difference between shallow copy and deep copy of arrays
12) How do you declare a 2D array?
13) How do you instantiate a 2D array?
14) How to access elements in a 2D array?
15) Difference between rectangular and jagged arrays
16) How to declare and instantiate, & access 3D array?

2. Logical Programming on Arrays

Interview Questions

1. Find the Largest Element in an Array?

2. Find the Second Largest Element in an Array without sorting?
3. Given an array of n integers, return true if the array is sorted in non-decreasing order or else false?
4. Remove duplicates from sorted array?
5. Given an array of N integers, left rotate the array by one place.?
6. Given an array of N integers, left rotate the array by one place. Left rotate an array by D places?
7. You are given an array of integers, your task is to move all the zeros in the array to the end of the array and move non-negative integers to the front by maintaining their order?
8. Given an array, and an element num the task is to find if num is present in the given array or not. If present print the index of the element or print -1?
9. Given two sorted arrays, arr1, and arr2 of size n and m. Find the union of two sorted arrays?
10. Given an integer N and an array of size N-1 containing N-1 numbers between 1 to N. Find the number(*between 1 to N*), that is not present in the given array.
11. Given an array that contains only 1 and 0 return the count of maximum consecutive ones in the array?
12. Given a non-empty array of integers arr, every element appears twice except for one. Find that single one.
13. Given an array nums of size n and an integer k, find the length of the longest sub-array that sums to k. If no such sub-array exists, return 0?
14. Remove the duplicate elements from the array without taking the help of new array & collections?
15. Given a matrix if an element in the matrix is 0 then you will have to set its entire column and row to 0 and then return the matrix?
16. Given a matrix, your task is to rotate the matrix 90 degrees clockwise &anti-clockwise?
17. You have been given a 2-D array 'mat' of size 'N x M' where 'N' and 'M' denote the number of rows and columns, respectively. The elements of each row are sorted in non-decreasing order. Moreover, the first element of a row is greater than the last element of the previous row (if it exists). You are given an integer ‘target’, and your task is to find if it exists in the given 'mat' or not?
18. Find the transpose of a non-square matrix.
19. Convert a matrix to its lower triangular form.
20. Convert a matrix to its upper triangular form.
21. Iterate the 3D array by using loops?

---

21. STRINGS

1. Strings

Interview Questions

1. What is a String in Java?
2. How do you create a String object & how many way to create the Strings in java?
3. Difference between: String s1 = "Hello"; & String s2 = new String("Hello");
4. Why are String objects immutable in Java?
5. What is String Constant Pool (SCP)?
6. How does SCP help in memory optimization?
7. What happens when you create two String literals with the same value?
8. How does Java store Strings created using new keyword?
9. How does intern() method work in Strings?
10. Why did Java designers make String immutable? (Security, Caching, Thread-safety, Performance)?
11. What is the advantage of storing Strings in SCP?
12. What happens internally when you modify a String?
13. What is the difference between mutable and immutable objects?
14. What happens if you concatenate Strings using + operator?
15. What is the difference between == and equals() in Strings?
16. What is the difference between compareTo() and equals()?
17. Can String be extended (inheritance)? Why or why not?
18. Difference between substring() and subSequence()?
19. What is StringBuffer class in Java?
20. Is StringBuffer mutable or immutable? Is StringBuffer thread-safe? Why?
21. Difference between append() and insert() methods?
22. What is the default capacity of StringBuffer?
23. How does the capacity of StringBuffer increase dynamically?
24. What is StringBuilder in Java?
25. Difference between StringBuffer and Stringbuilder?
26. Is Stringbuilder thread-safe? Why or why not?
27. When should you use Stringbuilder over StringBuffer?
28. What are common methods in Stringbuilder and StringBuffer?
29. Which one is best for frequent string modification?

30. Which one is best for multithreaded environment and singlethread environment?
31. Difference between length() method and length property of arrays.
32. Difference between isEmpty() and isBlank().
33. Difference between trim() and strip() (introduced in Java 11).
34. Difference between strip(), stripLeading(), and stripTrailing().
35. Difference between indexOf() and lastIndexOf().
36. Difference between split() and join() (breaking vs combining).
37. Difference between valueOf() and toString().

2. Logical Programming on Strings

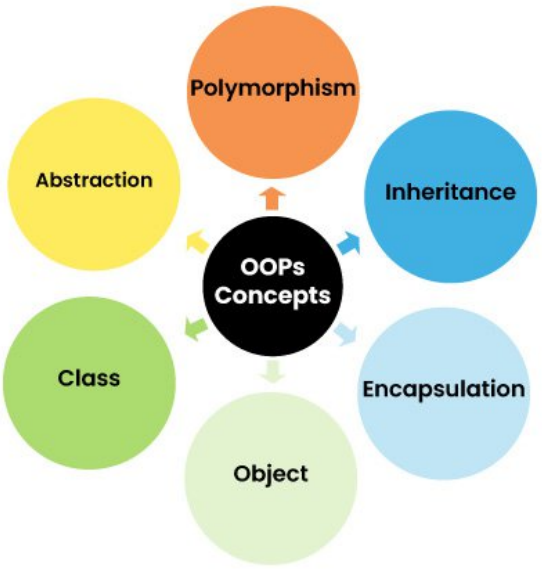
Interview Questions

22. Write a program to reverse a string.
23. Write a program to check if a string is a palindrome.
24. Count the number of vowels and consonants in a string.
25. Count the number of words in a string.
26. Convert a string to uppercase and lowercase.
27. Find the length of a string without using length().
28. Count the frequency of a given character in a string.
29. Check whether a string contains only digits.
30. Remove all white spaces from a string.
31. Compare two strings without using equals().
32. Find the first non-repeated character in a string.
33. Count the occurrences of each character in a string.
34. Check if two strings are anagrams of each other.
35. Remove duplicate characters from a string.
36. Swap case of each character (uppercase → lowercase, lowercase → uppercase).
37. Reverse each word in a sentence (not the full string).
38. Find the largest and smallest word in a sentence.
39. Count special characters, digits, and alphabets separately.
40. Check if a string is a pangram (contains all letters A–Z).
41. Given a string s, representing a large integer, the task is to return the largest-valued odd integer (as a string) that is a substring of the given string s.
    The number returned should not have leading zero's. But the given input string may have leading zero?

42. Write a function to find the longest common prefix string amongst an array of strings. If there is no common prefix, return an empty string ""?
43. Given two strings s and t, determine if they are isomorphic. Two strings s and t are isomorphic if the characters in s can be replaced to get t.
    All occurrences of a character must be replaced with another character while preserving the order of characters. No two characters may map to the same character, but a character may map to itself?
44. Write a program to check if one string can become another by shifting its characters to the right in Java?"
45. You are given a string s. Return the array of unique characters, sorted by highest to lowest occurring characters?
46. Write a program to Count Number of Substrings in a given String?
47. Given a string s, return the longest palindromic substring in s?
48. Find The beauty of a string is defined as the difference between the frequency of the most frequent character and the least frequent character (excluding characters that do not appear) in that string?
49. Validate email format using regex.
50. Validate mobile number using regex.
51. Write a Java program to check whether a given string containing parentheses (), curly braces {}, and square brackets [] is valid or not?
    ({[]})èvalid ([)]èinvalid

---

22. INTRODUCTION TO OOP'S



Interview Questions

1) What is Object-Oriented Programming (OOP)?
2) What are the main principles/features of OOP? Explain each briefly?
3) What is an Object in Java?
4) What is a Class in Java? How is it different from an Object?
5) What is the difference between Procedural (functional) Programming and Object-Oriented Programming?
6) Why is OOP better for real-world applications?
7) Why is Java considered a pure Object-Oriented language? Is it fully true?
8) How does OOP support security in applications?
9) What happens in memory when an object is created?
10) Can a class exist without objects? Explain with example?
11) What is an immutable object? How to create one in Java?
12) What are the advantages of using OOP?
13) What is the SOLID principle in OOP?

---

23. ENCAPSULATION

Interview Questions

1. What is encapsulation in Java?
2. How is encapsulation different from abstraction?

3. Why is encapsulation important in object-oriented programming?
4. How do you achieve encapsulation in Java?
5. What are the benefits of using private variables in a class?
6. Can you give a real time example of encapsulation in Java?
7. What is the role of getters and setters in encapsulation?
8. Is encapsulation possible without access modifiers?
9. What happens if you make all fields public? Is it still encapsulation?
10. How does encapsulation improve code maintainability?
11. Can you explain how encapsulation helps in securing data?
12. Why should we avoid exposing too many getters and setters?
13. How does encapsulation relate to the concept of immutability?
14. Can you encapsulate a class that has static fields?
15. What is the difference between encapsulation and information hiding?
16. Can you achieve encapsulation without using setter methods?
17. How does encapsulation interact with inheritance and polymorphism?
18. Can encapsulation be broken using reflection? How would you prevent it?
19. How would you design a class that exposes only read-only access to its internal state?
20. What are some anti-patterns that violate encapsulation?
21. Demonstrate encapsulation using interfaces and abstract classes.

Encapsulation Programming Problems

1) Create a Student class with private fields name, rollNumber, and age.
Provide public getters and setters for all fields.
Write a program to create a Student object, set its details, and print all information using getters

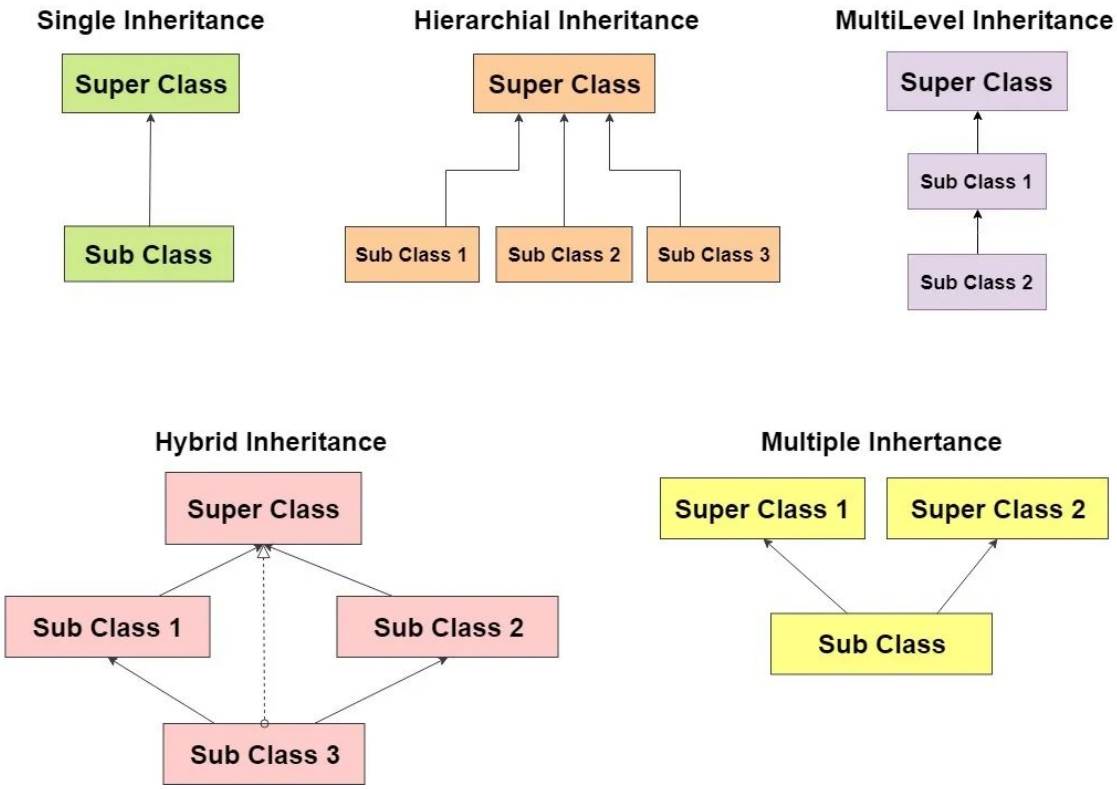
2) Create a BankAccount class with private fields accountNumber,
accountHolderName, and balance.
Provide methods to deposit() and withdraw() money.
Use getters to display the balance.
Ensure withdraw() does not allow balance to go negative.

3) Create a Product class with private fields productId, productName, and price.
Ensure price cannot be negative using the setter.
Write a program to create multiple products and display their details.

4) Create an Employee class with private fields name, baseSalary, and bonus.
Write a method calculateTotalSalary() that returns baseSalary + bonus.
Ensure bonus is non-negative using setter validation.
Print total salary of the employee using getter methods.

---

24. INHERITANCE



Interview Questions

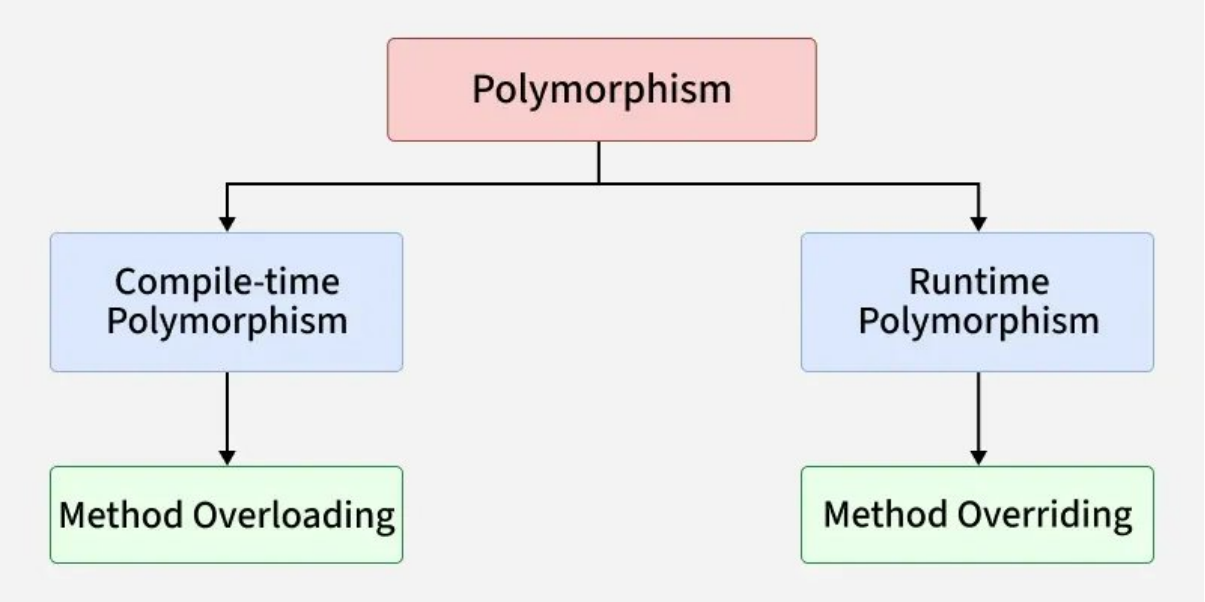
1) What is inheritance in Java? Why do we use inheritance?
2) Explain the concept of parent (super) class and child (sub) class in Java. How does inheritance establish a relationship between them?

3) What is single inheritance in Java? Explain with an example.
4) What is multilevel inheritance in Java? Explain with an example
5) Why is multiple inheritance not supported in Java?
6) What is hierarchical inheritance in Java? Provide an example.
7) What is hybrid inheritance in Java? Explain with an example.
8) What is the purpose of the extends and implements keywords in Java inheritance, and what are the key differences between them?
9) How does constructor chaining work in inheritance?
10) Can a subclass access private members of the superclass?
11) Can we override static methods? Why or why not?
12) What is HAS-A relationship and IS-A relationship in OOP?
13) What happens when a method is declared final in a superclass?
14) Can we override a private method? Why?
15) What is the role of access modifiers in inheritance?
16) What is the order of execution: static block, instance block, constructor in inheritance?
17) Can final classes be inherited?
18) How does multiple inheritance get achieved through interfaces?
19) What is the difference between composition and aggregation?

Programs

1) Write Java programs to demonstrate **single**,**multilevel**,**hierarchical** and **hybrid inheritance**.
2)Write a program to access parent class members using super?
3)Write a program to show constructor execution order in inheritance?
4) Write a program showing why multiple inheritance is not supported with classes?
5) Write a program using final class in inheritance?
6) Write a program demonstrating diamond problem using interfaces and show how Java resolves it?
7)Write a program demonstrating composition vs inheritance with a real-life example (e.g., Car HAS-A Engine vs Car IS-A Vehicle).
8)Write a program to show that final methods cannot be overridden by a subclass. Try to override and show compiler behavior.
9) Create an abstract class with both abstract and non-abstract methods. Extend it and implement the abstract methods in the subclass.

25. POLYMORPHISM



Interview Questions

1) What is polymorphism in Java?

2) What is the difference between compile-time/early binding/static polymorphism and runtime/late binding/dynamic polymorphism?

3) How does method overloading achieve compile-time polymorphism?

4) How does method overriding achieve runtime polymorphism?

5) What is dynamic method dispatch in Java, and how does the JVM resolve method calls at runtime?

6) What is method hiding in Java? Why static methods cannot participate in runtime polymorphism?

7) What is the role of @Override annotation?

8) What are the rules to override a method?

9) What is covariant return type?

10) Rules for method overloading in Java?

11) What is upcasting and downcasting? Why is downcasting risky?

12) Can final, static, or private methods be overridden in Java?

13) Can main() be overridden?

14) Why constructors cannot be overridden?

15) Can we override default methods in an interface?

16) Can overloading be based on access modifiers?

17) Can a subclass use a different access modifier when overriding a superclass method? Explain with an example.

Programs

1) Create a class to calculate the area of different shapes using
method overloading:
area(double radius) → Circle
area(double length, double width) → Rectangle
area(double side) → Square
Explain how overloading allows using the same method name for
different shapes.

2) Demonstrate an ambiguity problem with overloaded methods:
test(int a, double b)
test(double a, int b)
Call with (10, 10) and resolve ambiguity using type casting.
Explain why the compiler throws an error and how explicit casting fixes it.

3) Create classes Animal and Dog. Override sound() in Dog to print "Bark".
Use Animal a = new Dog() and call sound().
Explain how runtime polymorphism works here.

4) Create an abstract class Shape with method draw().
Implement Circle, Triangle, and Rectangle classes overriding draw().
Use a Shape[] array to call draw() for all objects.
Explain dynamic method dispatch.

5)Create a class Bank with method rateOfInterest().

Subclasses: SBI, HDFC, ICICI → override the method with different rates.

Use a parent reference to call the method for each bank.

Explain runtime polymorphism in real-world context.

6)Online Order Example:

Parent class OnlineStore with placeOrder()

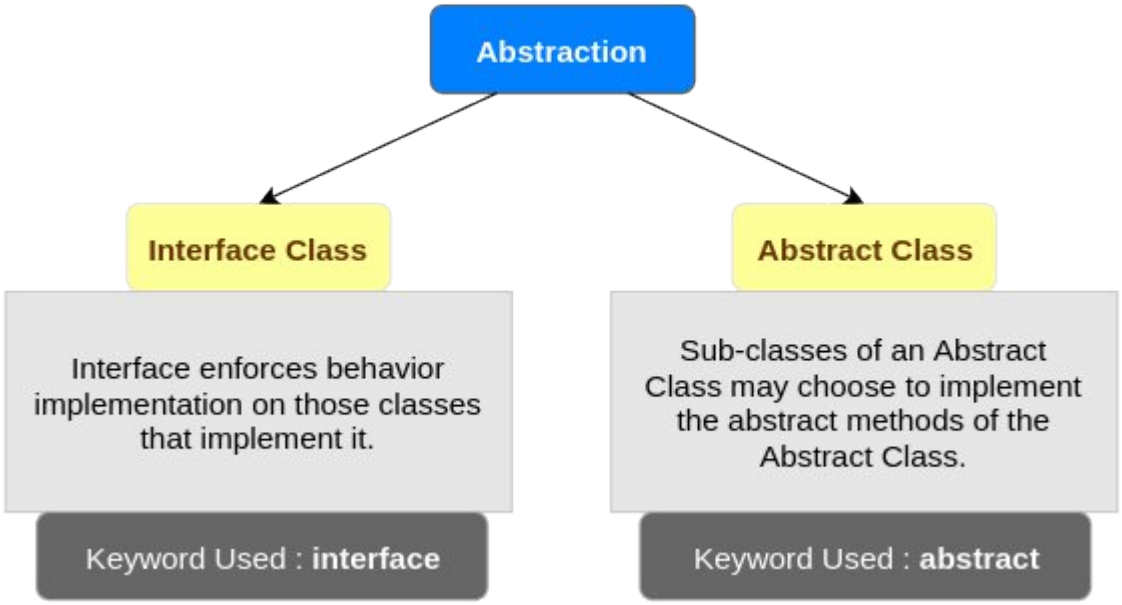
Child classes: Flipkart, Amazon, Ajio override placeOrder()

Use parent reference to call method.

Explain polymorphic behavior in online shopping.

---

26. ABSTRACTION



Interview Questions

1) What is abstraction in Java? Explain with a real-world example.

2) How is abstraction different from encapsulation?

3) How can you achieve abstraction in java?

4) What is the difference between an interface and an abstract class (before Java 8 vs after Java 8)?

5) Can an abstract class implement an interface?

6) What is abstract method?

7) If an abstract class contains a constructor, why are we still unable to create objects of an abstract class? Explain the reason behind having a constructor there?

8) Can we use default and static methods in an interface? Why were they introduced?

9) What is marker interface? Name two examples?

10) Can an interface contain variables? What is their default nature?

11) Explain Dynamic Method Dispatch in relation to abstraction?

12) If Java does not support multiple inheritance, how can a class implement multiple interfaces?

13) What happens when a subclass does not implement all abstract methods of its abstract parent class? Why does Java force this?

14) Why do interfaces in Java (before Java 8) allow only public abstract methods? Explain logically.

15) Is it mandatory for an abstract class to contain abstract methods?

16) Why can't a method in Java be declared as both abstract and synchronized at the same time? Explain logically.

Programs

1) Build an ATM abstraction:

Abstract class ATM with method process()

Withdraw and Deposit must override process()

Call using parent reference.

2)Create an interface Payment with pay().

Implement UPI, CreditCard, NetBanking and test.

3)Create an abstract class Employee:

properties: name, id

abstract method: calculateSalary()

Create PermanentEmployee and ContractEmployee with different salary formulas

4)Create an abstract class Order:

orderId, customerName

abstract method: calculateBill()

Create subclasses OnlineOrder, CashOnDeliveryOrder with different bill logics

5) Try changing the access modifier of the abstract method:

- public
- private
- protected
- default
- final abstract
- static abstract

Write down which are allowed and which cause compile errors

6) Write a Java program to demonstrate how interfaces help achieve loose coupling.

---

27. MULTITHREADING

1. Basics of Multitasking & Thread Fundamentals

Interview Questions

1. What is multitasking in Java?
2. What is the difference between process-based and thread-based multitasking?
3. What is a thread in Java?
4. What is multithreading, and why is it needed & advantages of multithreading?
5. What is the main thread in Java & How is the main thread created automatically by the JVM?
6. What happens if the main thread terminates before child threads?
7. How many ways can you create a thread in Java?
8. What is the difference between extending Thread class and implementing Runnable interface?

2. Thread Lifecycle and Methods

Interview Questions

1. Explain the lifecycle (states) of a thread in Java?
2. What are the different states in a thread's lifecycle?
3. What is the difference between start() and run() method?
4. What happens if we call run() directly instead of start()?
5. What does the join() method do?
6. What is the sleep() method?
7. What does yield() do?
8. What is interrupt(), and how does it work?
9. What is the isAlive() method used for?
10. What is the difference among sleep() and join() and yield() ?
11. Can we restart a dead thread in Java?

3. Thread Priorities & Naming

Interview Questions

1. What are thread priorities, and how are they assigned?
2. What is the default thread priority?
3. What are the constant values used for priorities (MIN_PRIORITY, NORM_PRIORITY, MAX_PRIORITY)?
4. Does thread priority guarantee order of execution?
5. How to set the name of a thread & How to get the name of a thread?
6. What happens if two threads have the same priority?
7. Can we change thread name and priority after it starts?
8. What is the difference between Thread.currentThread() and this?
9. Can we call start() method twice on the same thread?

4. Performing Multiple Tasks with Threads

Interview Questions

1. How to perform multiple tasks by multiple threads?
2. How to perform multiple tasks by a single thread?
3. Can a thread execute multiple run() methods?

4. What is a single-threaded model vs multi-threaded model?
5. How can anonymous inner classes or lambda expressions be used to create and run threads in Java? Explain with an example.

5. Synchronization & Deadlocks

Interview Questions

1. What is synchronization in Java &Why is synchronization needed?
2. What is the difference between synchronized method and synchronized block?
3. What is the lock object in synchronization?
4. What is static synchronization? Difference between object level synchronization & Class level synchronization? How to implement both?
5. What is a deadlock, how to implement it & how can you avoid it?

6. Inter-Thread Communication & Daemon Threads

Interview Questions

1. What is inter-thread communication? How to implement by using wait(), notify(), and notifyAll() methods?
2. Why can these (wait(), notify(), and notifyAll())methods be called only inside synchronized blocks?
3. What is the difference between sleep() and wait()?
4. What is a daemon thread? How do you create and identify a daemon thread?
5. What is the purpose of the volatile keyword in Java multithreading, and how does it differ from synchronization?

7. Thread Pool & Executor Framework

Interview Questions

1. What is a Thread Pool in Java?
2. What is the Executor Framework? How to implement thread creation, management, and scheduling by using thread pools?
3. What is the difference between Executor, ExecutorService, and Executors class?

8. Programming Questions on Multithreading

Interview Questions

1. Implement a simple thread-safe Singleton class?

2. Producer-Consumer problem using wait/notify.?
3. Print even and odd numbers using two threads.?
4. Print numbers 1–10 sequentially using 3 threads?
5. Write a program to implement deadlock?
6. Create custom ThreadPoolExecutor with core and max threads ?implement all above questions by using Executor Framework?
7. write a program to implement to object level locking and class level locking?

---

28. EXCEPTION HANDLING

Interview Questions

1. What is an exception in Java? How is it different from an error?
2. Explain the difference between checked and unchecked (runtime) exceptions with examples. How do they affect program flow?
3. What is the difference between throw and throws in Java?
4. What happens if a checked exception is not handled or declared? Explain exception propagation in Java?
5. Can a method override another method that throws exceptions? What are the rules regarding checked exceptions?
6. What is the role of a try block, catch block, and finally block? Does a finally block always execute?
7. When will the finally block not execute?
8. Can a try block exist without a catch block? If yes, under what conditions?
9. What are the rules for catching exceptions in hierarchical order in Java, and how does the multi-catch feature introduced in Java 7 help simplify exception handling?
10. What happens if an exception occurs inside a catch block?
11. Can constructors throw exceptions? How should be handled ?
12. A single try block can throw multiple exceptions like ArithmeticException and ArrayIndexOutOfBoundsException. How should be handle with single-catch block?
13. Explain best practices for writing exception-safe code, including the impact of over using try-catch blocks on performance and maintainability.

14. Consider a situation where a try block executes a return statement, but a finally block is also present that modifies a variable. Which value is returned? Explain how the finally block affects execution.
15. What is the use of printStackTrace(), getMessage(), and toString()?
16. What is the main advantage of using try-with-resources over a normal try-catch-finally block?
17. What is a custom exception in Java? When should we create a custom exception in real time application?
18. If both the catch block and the finally block throw exceptions, which one will be propagated to the caller, and why?
19. What is the difference between ClassNotFoundException and NoClassDefFoundError in Java? When does each of them occur? Explain with suitable examples.
20. What is a ClassCastException in Java? Write a program that demonstrates how this exception occurs and explain how to prevent it.

Programming Questions

1. Write a program that throws a custom exception if a user enters a negative number for age?.
2. write a program to demonstrate multiple exceptions in a single try block, such as arithmetic exception and array index out of bounds exception. handle them using separate catch blocks.
3. write a program to demonstrate nested try-catch blocks. include at least one inner try that throws an exception and show how it is handled.
4. write a program using a finally block. include a return statement in the try block and show what value is returned when the finally block modifies a variable.
5. write a program to demonstrate the difference between throw and throws. include a method that throws an exception and another method that declares it using throws.
6. write a program to demonstrate try-with-resources using a file or scanner. show how resources are automatically closed.
7. write a program to demonstrate exception chaining. create one exception that is caused by another and show how the original exception can be retrieved using getCause().

---

29. UNDERSTANDING FINAL KEYWORD AND GARBAGE COLLECTION

1.Final Keyword

Interview Questions

1) What is the difference between final, finally, and finalize() in Java?
2) Can we declare a class as final? What does it mean, and what happens if we try to inherit it?
3) Can a final variable be reinitialized? Explain with an example.
4) What is a blank final variable? How can we initialize it (including static ones)?
5) Can we declare a method as final? What does it prevent, and can it be overloaded or overridden?
6) Can you have a final reference variable pointing to a mutable object? What happens if you modify the contents of that object?
7) How can you make a class immutable using the final keyword?
8) Can a final local variable be used inside an anonymous inner class? Why?

Programs

1) Write a Java program to demonstrate that a final variable cannot be reassigned after initialization.
2) Create a program showing how a blank final variable can be initialized inside a constructor.
3) Write a Java program that defines a static final variable and initializes it using a static block.
4) Write a Java program where a final method is declared in a parent class and show that it cannot be overridden in the subclass.
5) Write a Java program where a final reference variable refers to a mutable object (like an ArrayList). Modify the contents and observe the behavior.
6) Create a Java program that defines a final variable in a parent class and accesses it through inheritance. Show that it cannot be changed by the subclass.

2. Garbage Collector

Interview Questions

1) What is a Garbage Collector in Java?
2) When does an object become eligible for Garbage Collection?

3) Can we force Garbage Collection in Java?

4) What is the difference between Garbage Collector and Garbage Collection?

5) What does System.gc() do?

6) What is the purpose of the finalize() method?

7) How does automatic Garbage Collector work in Java?

8) Explain the Mark and Sweep algorithm used by JVM.

Programs :

1) Write a Java program to demonstrate when an object becomes eligible for Garbage Collection.

2) In the below code, identify how many objects are eligible for Garbage Collection after both references are set to null:

```
GCExample obj1 = new GCExample();
GCExample obj2 = new GCExample();
obj1.ref = obj2;
obj2.ref = obj1;
obj1 = null;
obj2 = null;
System.gc();
```

3) Write a program to override the finalize() method and show when it gets called during Garbage Collection.

4) Write a program to compare System.gc() and Runtime.getRuntime().gc() and observe their behavior.

30.INPUT OUTPUT STREAM

Interview Questions

1) What is the hierarchy of the InputStream and OutputStream classes?

2) What do you understand by an I/O stream in Java?

3) What are FileInputStream and FileOutputStream in Java?

4) What are the different ways to take input from the console in Java?

5) Difference between DataInputStream and BufferedInputStream.

6) Why InputStream and OutputStream are abstract classes, not interfaces.

7) Difference between flush() and close().

8) What does ObjectOutputStream do? (used for serialization).

9) What is the difference between FileWriter and PrintWriter

Serialization

Interview Questions

1) What is serialization in Java?

2) How can you make a class serializable in Java?

3) How can you prevent serialization in a subclass when the parent class implements the Serializable interface?

4) Can a serialized object be transferred over a network?

5) What is deserialization in Java?

6) What is the purpose of the transient keyword in serialization?

7) What is the difference between the Serializable and Externalizable interfaces?

Coding Questions

1) Write a Java program to serialize an object of a class and save it to a file.

2) Write a Java program to deserialize an object from a file and print its field values.

3) Write a Java program to demonstrate the effect of the transient keyword — serialize and deserialize an object, showing that transient fields are not saved.

4) Write a Java program where a parent class is serializable but the subclass is prevented from being serialized.

---

31. COLLECTION FRAMEWORK

Collection

Interview Questions:

1. What is Collection Framework? advantages of the Collection Framework in Java

2. Describe the Collection hierarchy in Java.

3. List down the primary interfaces provided by Java Collections Framework?
4. Why Collection doesn't extend the Cloneable and Serializable interfaces?
5. what is Generic Collection?
6. what is the difference between Collection and Array?

List Interface

Interview Questions:

1.what is the use of the List interface What are the key characteristics of a List in Java?
2.Can a List contain duplicate elements?
3.What are the main classes that implement the List interface?
4. How does List maintain insertion order?

ArrayList

Interview Questions:

1. What is ArrayList in Java? what are the various methods provided by it?
2. How would you convert an ArrayList to Array and an Array to ArrayList?
3. How does ArrayList work internally in Java, and how does it grow dynamically when elements are added?
4. what is The Intial capacity of the ArrayList and default capacity of the ArrayList?
5. Can you store null values in an ArrayList?
6. How can you make an ArrayList immutable?
7. Is ArrayList synchronized in Java? If not, how can you make an ArrayList synchronized?
8. What are the real-time use cases of ArrayList in Java, and when should it be used?
9. What are the advantages and disadvantages of Arraylist?
10. How many constructors are present in ArrayList?

LinkedList

Interview Questions:

1. What is a LinkedList in Java, and how many types of linked lists does Java support? Explain with examples.
2.How does LinkedList work internally?
3. What data structure is used to implement LinkedList?
4.What are the advantages and disadvantages of using LinkedList?
5.please example the when use in real time LinkedList examples?

6. Is LinkedList synchronized in Java? If not, how can you make an LinkedList synchronized?
7. How many constructors are present in LinkedList?

Vectors

Interview Questions:

1. what is a Vector in Java?
2. How does vector work internally ?
3. Why is Vector considered legacy?
4. Vector is a synchronized are not?
5. what is The Intial capacity of the Vector and default capacity of the Vector?
6. Can you store null values in an Vector?
7. How many constructors are present in Vector?

Stack

Interview Questions:

1. What is the Stack class in Java?
2. How is Stack implemented internally?
3. What are the methods of the Stack class (push, pop, peek, etc.)?
4. How many constructors are present in Vector?

Set Interface

Interview Questions:

1.What is the Set interface in Java Collections Framework?
2.Why does Set not allow duplicate elements?
3.How does set work internally?
4.What are the advantages and disadvantages of using set?
5.What are the main classes that implement the set interface?
6.Can you store null values in a set?

HashSet

Interview Questions:

1. What is HashSet in Java?
2. How does HashSet ensure uniqueness of elements?

3. Does HashSet maintain insertion order?
4. What is the internal data structure used by HashSet?
5. Can HashSet store null values? If yes, how many?
6. Can HashSet store duplicate values? If not, what happens if we try?
7. Why does HashSet use hashCode() and equals() methods?

LinkedHashset

Interview Questions:

1. What is LinkedHashSet in Java?
2. How does LinkedHashSet differ from HashSet?
3. What internal data structure does LinkedHashSet use?
4. Does LinkedHashSet maintain insertion order?
5. Can LinkedHashSet store null values?

SortedSet

Interview Questions:

1. What is SortedSet in Java?
2. Which class commonly implements SortedSet?
3. Can SortedSet store null values? Why or why not?
4. What methods are introduced in SortedSet that are not in Set?

Treeset

Interview Questions:

1. What is TreeSet in Java?
2. Does TreeSet maintain any specific order? Which order?
3. Which data structure is used internally by TreeSet?
4. Can TreeSet store null values? Why or why not?
5. What is the default sorting mechanism used by TreeSet?

Queue Interface

Interview Questions:

1. What is the Queue interface in Java?
2. Does Queue allow null elements? Explain.
3. What are the common Queue implementations?

4. On which principle does a Queue work in Java?

PriorityQueue

Interview Questions:

1. What is PriorityQueue in Java?
2. Which data structure is used internally by PriorityQueue?
3. Does PriorityQueue maintain insertion order?
4. What is the default order used in PriorityQueue?
5. Does PriorityQueue allow null,duplicate elements?
6. How does PriorityQueue differ from Queue?

Deque

Interview Questions:

1. What is a Deque in Java and what does it stand for?
2. How is Deque different from a Queue?
3. What type of data structure behavior does Deque support?
4. Can Deque be used as both a stack and a queue?
5. Does Deque allow duplicate elements?
6. Does Deque allow null values? If not, why?

ArrayDeque

Interview Questions:

1. What is ArrayDeque in Java?
2. Which interfaces are implemented by ArrayDeque?
3. How does ArrayDeque differ from a standard Queue?
4. Can ArrayDeque be used as both a Queue and a Stack?
5. Does ArrayDeque allow null, duplicate values? Why or why not?
6. Which underlying data structure does ArrayDeque use?

Map Interface

Interview Questions:

1. What is the Map interface in Java?
2. How is Map different from List,Set and Queue ?

3. What are the main implementations of Map?
4. Can Map store duplicate keys or values?
5. What are the commonly used Map methods (put, get, remove, containsKey, containsValue)?
6. Can a Map store null keys and null values? Which implementations allow it?
7. What is the difference between keySet(), values(), and entrySet() methods?

HashMap

Interview Questions:

1. What is HashMap in Java?
2. Which interfaces does HashMap implement?
3. Can HashMap store null keys and null values? How many?
4. Does HashMap allow duplicate keys or duplicate values?
6. Does HashMap maintain insertion order?
7. How do you iterate over a HashMap?
8. How does HashMap store key-value pairs internally?
9. How does HashMap handle collisions?
10. What is the role of hashCode() and equals() in HashMap?

LinkedHashmap

Interview Questions:

1. What is LinkedHashMap in Java?
2. Which interfaces does LinkedHashMap implement?
3. Can LinkedHashMap store null keys and null values? How many?
4. Does LinkedHashMap maintain insertion order or sorted order?
5. How do you iterate through a LinkedHashMap?
6. How does LinkedHashMap handle collisions?

SortedMapInterface

Interview Questions:

1. What is the SortedMap interface in Java?
2. How is SortedMap different from Map?
3. Which class commonly implements SortedMap?
4. Does SortedMap allow null keys or null values?

5. Does SortedMap maintain keys in natural order or insertion order?
6. Can SortedMap store duplicate keys or values?
7. What are the key methods of SortedMap?
8. How does SortedMap maintain keys in sorted order?
9.

TreeMap

Interview Questions:

1. What is TreeMap in Java?
2. Which interfaces does TreeMap implement?
3. Does TreeMap allow null keys or null values?
4. Does TreeMap allow duplicate keys or values?
5. What is the natural ordering used by TreeMap?
6. How do you iterate through a TreeMap?
7. How does TreeMap maintain keys in sorted order internally?

Concurrent Collections

Interview Questions:

1. What are Concurrent Collections in Java?
2. Why do we need concurrent collections when we already have synchronized collections?
3. What is the difference between HashMap and ConcurrentHashMap?
4. What is the difference between Collections.synchronizedMap() and ConcurrentHashMap?
5. What is the use of CopyOnWriteArrayList in Java?
6. When should we prefer CopyOnWriteArrayList over ArrayList?
7. How does CopyOnWriteArrayList achieve thread-safety internally?

Differences

Interview Questions:

1. Differentiate between Collection and Collections?
2. Differentiate between Iterable and Iterator?
3. Differentiate between an Array and an ArrayList?
4. Differentiate between ArrayList and LinkedList?
5. Differentiate between Comparable and Comparator?
6. Differentiate between Set and Map?

7.Differentiate between List and Set?

8. Differentiate between List and Map?

9. Differentiate between Queue and Stack?

10.Differentiate between PriorityQueue and TreeSet?

11.Differentiate between the Singly Linked List and Doubly Linked List?

12.Differentiate between Iterator and Enumeration?

13.Differentiate between HashMap and HashTable?

14.Differentiate between HashSet and HashMap?

15.Differentiate between Iterator and ListIterator?

16.Differentiate between HashSet and TreeSet?

17.Differentiate between Queue and Deque?

18. Differentiate between HashMap and TreeMap?

19. Differentiate between ArrayList and Vector?

20. Differentiate between Fail-Fast Iterator and Fail-safe iterator?

21. Differentiate between Iterator and Spliterator?

22. Difference between WeakHashMap and IdentityHashMap?

Programs :

1) Write a Java program to remove duplicate elements from a List.

2) Write a program to iterate a List using for loop, Iterator, and forEach.

3) Write a program to find the maximum and minimum elements in a list of integers.

4) Write a program to sort a list in ascending and descending order.

5) Write a program to reverse a list using Collections.reverse() and manually without using it.

6) Write a program to add, remove, and retrieve elements from both ends of a LinkedList.

7) Write a program to compare insertion and retrieval performance between ArrayList and LinkedList.

8) Write a program to demonstrate synchronization in a Vector.

9) Write a program to iterate through a Vector using Enumeration.

10) Write a program to check if a string has balanced parentheses using a Stack.

11) Write a program to reverse a string using a Stack.

12) Write a program to evaluate a postfix expression using a Stack.

13) Implement a Queue using LinkedList and perform enqueue and dequeue operations.

14) Write a program to find the first non-repeating character in a string using a Queue.

15) Write a program to print elements of a PriorityQueue in natural order.

16) Write a program to use a PriorityQueue with a custom comparator for descending order.

17) Write a program to find the k largest elements in an array using a PriorityQueue.

18) Demonstrate addFirst(), addLast(), removeFirst(), and removeLast() operations in a Deque.

19) Implement a palindrome checker using a Deque.

20) Write a program to remove duplicate elements from a list using a HashSet.

21) Write a program to find union and intersection of two sets.

22) Write a program to maintain insertion order while removing duplicates using a LinkedHashSet.

23) Write a program to store and retrieve elements in sorted order using a TreeSet.

24) Write a program to count the frequency of characters in a string using a HashMap.

25) Write a program to find duplicate elements in an array using a HashMap.

26) Write a program to iterate through a HashMap using entrySet(), keySet(), and values().

27) Write a program to sort a HashMap by keys.

28) Write a program to sort a HashMap by values.

29) Write a program to store elements in sorted key order using a TreeMap.

30) Write a program to use a custom comparator in a TreeMap for descending key order.

31) Write a program to demonstrate thread-safe operations using a Hashtable.

32) Write a program to check how null key and value behave in a Hashtable.

33) Write a program to perform thread-safe operations using a ConcurrentHashMap.

34) Write a program to sort a list of custom objects (like Employee) by name or salary using Comparator.

35) Write a program to find the most frequently occurring element in a list using a Map.

36) Write a program to convert one type of collection to another (e.g., List → Set, Set → Map, etc.).

---

32. JAVA UTILITY CLASSES

Arrays Class

1. What is the Arrays class in Java?
2. What are the key methods of the Arrays class?
3. What is the difference between Arrays.equals() and == operator and Arrays.deepEquals()?
4. How do you copy an array in Java using Arrays (Arrays.copyOf())?
5. What does Arrays.fill() do?

6. How do you convert an array to a list using Arrays?
7. What is Arrays.compare() used for?
8. How to convert an array to Stream using Arrays.stream()?
9. How does Arrays.sort() work internally? Can we sort in descending order using Arrays.sort()?What is its time complexity?
10. What's the difference between Arrays.toString() and Arrays.deepToString()?
11. How does Arrays.binarySearch() work? What does Arrays.binarySearch() return if the element is not found?
12. What happens if I pass a primitive int[] array to Arrays.asList()?
13. What is a Arrays.spliterator() in Java? How to use it and where to use it ?

Collections Class

1. What is the Collections class in Java?
2. What is the difference between Collection and Collections?
3. What are some important methods of the Collections class?
4. What does Collections.sort() do? How is Collections.sort() different from List.sort()?
5. How does Collections.reverse() work?
6. How to find min() and max() using Collections?
7. What does Collections.binarySearch() do?
8. What does Collections.frequency() do?
9. What is the purpose of Collections.copy()?
10. How do you make a collection thread-safe? By use Collections.synchronizedList() Collections. synchronizedSet(),Collections.synchronizedMap() ?
11. How do you make a collection read-only (unmodifiable)?
12. What does Collections.disjoint() do?
13. What's the difference between Collections.emptyList() and Collections.singletonList()?

Collectors Class

1. What is Collectors in Java? Is Collectors part of the Collections framework?
2. What are the most commonly used methods of Collectors?
3. Difference between Collections and Collectors?
4. What does Collectors.toList(), Collectors.toSet(), Collectors.toMap() do?
5. What happens if duplicate keys occur in Collectors.toMap()?

6. What is the difference between toMap() and toConcurrentMap()?
7. What is the purpose of Collectors.counting()?Can Collectors.counting() be replaced with stream.count()? Which one is preferred?
8. What are summingInt, summingDouble, and summingLong used for?
9. What is averagingInt(), and how is it different from summingInt()?
10. What is Collectors.joining() used for?
11. What is groupingBy() in Java Streams?
12. What is the difference between groupingBy() and partitioningBy()?
13. What is nested grouping, and how can you achieve it using Collectors?
14. Where we can use Collectors. reducing() method?

Random Class

1. What is Random class in Java?
2. How to generate random numbers using Random? Difference between Math.random() and Random class?
3. How does the Random class generate pseudo-random numbers internally?

Date Class

1. What is java.util.Date?
2. How to get the current date and time?
3. Difference between Date and Calendar.
4. How to convert Date to LocalDate or LocalDateTime?
5. How do you format a Date using SimpleDateFormat with a custom pattern?
6. Why Date class is considered legacy?

Calender Class

1. What is Calendar class in Java? Important Methods of Calendar?
2. How to convert Calendar to LocalDateTime (Java 8+)?
3. How to get the current year, month, and day using Calendar?
4. How do you roll a date forward or backward using Calendar?

Locale Class

1. What is Locale class in Java ?why is it important in internationalization? Important Methods of Locale class?

2. What are the predefined Locale constants in Java? (Locale.US, Locale.UK, etc.)
3. How to get all available locales?

StringTokenizer Class

1. What is StringTokenizer in Java? How do you create a StringTokenizer object?
2. Important Methods of StringTokenizer ?
3. Difference Between StringTokenizer, split(), and Scanner?

Formatter Class

1. What is Formatter class in Java? Important Methods of Formatter?
2. Name some common format specifiers (%d, %f, %s, %c) ?
3. Difference Between Formatter and String.format() ?
4. What is IllegalFormatException and when is it thrown?

Date and Time

1. What are the main classes in the java.time package?
2. How do you format a Date object in Java?
3. How do you parse a string into a date?
4. How to format LocalDate, LocalTime, LocalDateTime, and ZonedDateTime?
5. What are common format patterns (dd, MM, yyyy, HH, mm, ss, a)?
6. Why is SimpleDateFormat not thread-safe?

---

33. JAVA 8 FEATURES

Lambda Expressions

Interview Questions

1. What is a lambda expression in Java 8?

2. Why were lambda expressions introduced?
3. What is the syntax of a lambda expression?
4. How do lambda expressions improve code readability?
5. Can a lambda expression have multiple parameters?
6. Can we use this keyword inside a lambda expression?
7. How is a lambda expression different from an anonymous inner class?
8. What is the functional interface requirement for a lambda?
9. Can a lambda expression throw exceptions?
10. Can we return values from a lambda expression?

Programs

1. Write a lambda expression to print all elements of a list.
2. Write a lambda expression to find even numbers from a list.
3. Write a lambda expression to sort a list of strings alphabetically.
4. Create a thread using a lambda expression.
5. Write a program to calculate the sum of all numbers in a list using lambda.

Functional Interfaces

Interview Questions

---

1. What is a functional interface?
2. What annotation is used to define a functional interface?
3. Can a functional interface have multiple abstract methods?
4. Name some predefined functional interfaces in **java.util.function** package.
5. What is the difference between Function, Predicate, and Consumer?
6. Can a functional interface have default or static methods?
7. What happens if you remove @FunctionalInterface annotation?

Programs

1. Create a custom functional interface and implement it using a lambda.
2. Use a predefined Predicate to check if a number is even.
3. Use a Function to convert a string to uppercase.
4. Use a Consumer to print all items in a list.
5. Use a Supplier to return a random number.

Default Methods in Interface

Interview Questions

1. What is a default method in an interface?
2. Why were default methods introduced in Java 8?
3. How are default methods different from abstract methods?
4. Can we override a default method in an implementing class?
5. What if two interfaces have the same default method name?
6. Can we call a default method from another default method?

Programs

1. Create an interface with a default method and implement it in a class.
2. Demonstrate method conflict resolution when two interfaces have same default method.

Static Methods in Interface

Interview Questions

1. Can an interface have static methods in Java 8?
2. How do we call a static method of an interface?
3. Can static methods be inherited by implementing classes?
4. What is the purpose of static methods inside an interface?

Programs

1. Create an interface with a static method and call it from the main method.
2. Show what happens if a class defines the same static method name.

Private Methods in Interface

Interview Questions

1. What is a private method in an interface?
2. Why were private methods added to interfaces in Java 9 (extension of Java 8 feature)?
3. Can we call private methods from outside the interface?
4. Can a private method be static or non-static?

5. How do private methods help in code reusability inside interfaces?

Programs

1. Write an interface with one private static method and one default method calling it.
2. Demonstrate how private methods reduce code duplication in interfaces.

Method References

Interview Questions

---

1. What is a method reference in Java 8?
2. What are the different types of method references?
3. How does method reference differ from a lambda expression?
4. What is the syntax for method reference?
5. Can we use method references with constructors?
6. Can method references refer to overloaded methods?

Programs

1. Use a method reference to print a list of names.
2. Use a constructor reference to create objects.
3. Use a static method reference for mathematical operations.
4. Replace a lambda with a method reference in a Stream operation.

STREAM API

Interview Questions

---

1. What is the Stream API in Java 8?
2. How is Stream different from Collection?
3. What are intermediate and terminal operations?
4. What is the difference between map() and flatMap()?
5. Can a Stream be reused?
6. How do you create a Stream in Java?
7. What are short-circuiting operations in Stream?
8. What is lazy evaluation in Stream?

9. Difference between findFirst() and findAny().
10. What are parallel streams?

Programs

1. Find even numbers from a list using Stream.
2. Convert a list of strings to uppercase using Stream.
3. Count how many names start with "A" using Stream.
4. Find the maximum and minimum element using Stream.
5. Sort a list of integers using Stream.
6. Flatten a list of lists using flatMap().
7. Find average of numbers using Stream.

INTERMEDIATE OPERATIONS

1. filter()

Concept: Filters elements based on a condition.

Logical Questions:

1. Given a list of integers, print only even numbers using streams.
2. From a list of strings, print only those that start with the letter "A".
3. From a list of Employee objects, print names of employees having salary greater than 50,000.
4. Given a list of numbers, print only prime numbers using streams.

2. map()

Concept: Transforms elements into another form.

Logical Questions:

1. Convert a list of strings into uppercase.
2. Given a list of integers, return a list of their squares.
3. Extract employee names from a list of Employee objects.
4. Convert a list of numbers into their string representation.

3. flatMap()

Concept: Flattens nested structures like lists of lists.

Logical Questions:

1. Given a list of lists of integers, flatten it into a single list of integers.
2. Given a list of sentences, return a list of all unique words.
3. Convert a list of departments, where each department has a list of employees, into a list of all employees.

4. distinct()

Concept: Removes duplicate elements.

Logical Questions:

1. Remove duplicate numbers from a list.
2. Given a list of strings, print only unique strings.
3. From a list of employees, print unique department names.

5. sorted()

Concept: Sorts elements in natural or custom order.

Logical Questions:

1. Sort a list of integers in ascending and descending order.
2. Sort a list of employees by salary.
3. Sort strings by their length.
4. Sort names alphabetically ignoring case.

6. limit() and skip()

Concept: Truncate or skip elements in a stream.

Logical Questions:

1. Print first 5 elements from a list of numbers.
2. Skip first 3 elements and print the rest.
3. Given a list of students sorted by marks, print the top 3 scorers.
4. Find the second-highest salary using skip() and limit().

7. peek()

Concept: Used for debugging, performs an action without altering stream.

Logical Questions:

1. Print each element before filtering even numbers.

2. Log intermediate results while mapping data.
3. Trace employee objects before and after applying a filter.

TERMINAL OPERATIONS

1. forEach()

Concept: Iterates through stream elements.

Logical Questions:

1. Print all elements of a list.
2. Print names of employees in uppercase.
3. Print all numbers greater than 100.

2. collect()

Concept: Collects results into a collection or map.

Logical Questions:

1. Collect all even numbers into a list.
2. Collect all unique words into a Set.
3. Collect employee names into a List.
4. Group employees by department using Collectors.groupingBy().

3. count()

Concept: Counts elements in the stream.

Logical Questions:

1. Count number of strings that start with “A”.
2. Count even numbers in a list.
3. Count number of employees earning above 80,000.

4. reduce()

Concept: Reduces stream to a single value.

Logical Questions:

1. Find the sum of all integers in a list.
2. Find the maximum number using reduce().

3. Concatenate all strings using reduce().
4. Multiply all numbers in a list.

5. findFirst() / findAny()

Concept: Returns first or any matching element.

Logical Questions:

1. Find first even number in a list.
2. Find any employee from IT department.
3. Find first string with length greater than 5

---

6. anyMatch(), allMatch(), noneMatch()

Concept: Boolean match operations.

Logical Questions:

1. Check if any number in list is divisible by 5.
2. Check if all numbers are positive.
3. Check if no employee has a null name.
4. Check if all students passed (marks > 35).

7. min(), max()

Concept: Find minimum or maximum element based on comparator.

Logical Questions:

1. Find the smallest number in a list.
2. Find employee with highest salary.
3. Find string with minimum length.
4. Find oldest person in a list.

8. toArray()

Concept: Convert stream to array.

Logical Questions:

1. Convert a stream of integers into an array.
2. Convert list of employee names to an array.
3. Convert unique sorted numbers into an array.

Advanced Logical Questions on Stream API

1. Given a sentence, find the word that has the hight length.
   input: String s = "I am learning streams API in Java";

   output: learning

2. Remove duplicates from the string and return in the same order.
   input: String s = "dabcadefg";

   output: dabcdefg

3. Given a sentence, find the word that has the 2nd(nth) highest length.
   input: String s = "I am learning streams API in Java";

   output: streams

4. Find the 2nd highest length word in a sentence.
   input: String s = "I am learning streams API in Java";

   output: 7

5. Given a sentence, find the occurrence of each word.
   input: String s = "I am learning streams API in Java Java";

   output: {Java=2, in=1, streams=1, I=1, API=1, learning=1, am=1}

6. Given a sentence, find the words with a specified number of vowels.
   No. of vowels : 2

   input: String s = "I am learning streams API in Java";

   output: streams

API

Java

7. Given a list of Integers, divide it into lists on having an even number and the other having an odd number.

input: int[] arr = {1,2,3,4,5,6,7,8};

output: [[1, 3, 5, 7], [2, 4, 6, 8]]

8. Given a word, find the occurrence of each character.

input: String s = "Mississippi";

output: {p=2, s=4, i=4, M=1}

9. Given an int[] array, re-arrange the elements to form the highest/lowest possible.

input: int []arr = {1, 2, 3, 5, 4};

output: 12345

54321

10. Given an array find the sum of unique elements

input: int []arr = {1,6,7,8,1,1,8,8,7};

output: 22

11. Given a string, find the first non-repeated character.

input: String s = "Hello World";

output: H

12. Guven a string, find the first repeated character

input: String s = "Hello World";

output: l

13. Given an array of Integers, group the numbers by the range in which they belong.

input: int []arr= {2, 3, 10, 14, 20, 24, 30, 34, 40, 44, 50, 54};

output: {0=[2, 3], 10=[10, 14], 20=[20, 24], 30=[30, 34], 40=[40, 44], 50=[50, 54]}

14. Given a list of Strings, create a list that contains only integers.

input: String [] s = {"abc", "123", "456", "xyz"};

output: [123, 456]

15. Find the product of the first two elements from an array.

String [] s = {"pat", "tap", "pan", "nap", "Team", "tree", "meat"};

Output : [[pat, tap], [pan, nap], [Team, meat], [tree]]

COLLECTORS CLASS

Interview Questions

1. What is the Collectors class in Java 8?
2. What is the purpose of the collect() method?
3. Difference between toList(), toSet(), and toMap().
4. How do we use groupingBy() and partitioningBy()?
5. What is joining() used for?
6. Can we create a custom collector?

Programs

1. Collect Stream elements into a List using Collectors.toList().
2. Group employees by department using Collectors.groupingBy().
3. Count how many times each word appears using Collectors.counting().
4. Join names into a single string using Collectors.joining().

forEach() Method

Interview Questions

1. What is the forEach() method in Java 8?
2. On which interfaces is forEach() defined?
3. How is forEach() different from a regular for loop?
4. Can forEach() modify the collection while iterating?
5. Can we use break or return inside forEach()?

Programs

1. Print all list elements using forEach().
2. Use forEach() with lambda to print keys and values of a Map.
3. Use forEach() with method reference.

Optional Class

Interview Questions

1. What is Optional in Java 8?
2. Why was Optional introduced?
3. How do we create Optional objects?
4. Difference between isPresent() and ifPresent().
5. How do we provide a default value using orElse()?
6. What is the difference between orElse() and orElseGet()?
7. Can Optional be serialized?
8. Is it a good practice to use Optional for fields or parameters?

Programs

1. Create an Optional for a null string and print a default value.
2. Use Optional.ofNullable() and ifPresent() to print non-null values.
3. Use orElse() to handle a missing value.
4. Demonstrate difference between orElse() and orElseGet().

Parallel Array Sorting

Interview Questions

1. What is Arrays.parallelSort()?

2. How is parallelSort() different from Arrays.sort()?
3. When should we use parallelSort()?
4. Is parallelSort() thread-safe?
5. What algorithm does parallelSort() use?

Programs

1. Sort an integer array using Arrays.parallelSort().
2. Sort a string array using parallelSort().
3. Measure performance difference between sort() and parallelSort().

New Date and Time API (java.time)

Interview Questions

1. What is the new Date-Time API in Java 8?
2. What are the main classes in the new API?
3. Difference between LocalDate, LocalTime, and LocalDateTime.
4. What is Period and Duration?
5. How to format and parse dates using DateTimeFormatter?
6. What are the advantages of the new Date-Time API over java.util.Date?
7. Is the new Date-Time API thread-safe?

Programs

1. Display the current date, time, and date-time.
2. Add 7 days to current date.
3. Find difference between two dates using Period.
4. Format a date into dd-MM-yyyy pattern.
5. Convert between LocalDateTime and Instant.

34. INNER CLASS

Interview Questions

1) What is inner class?
2) Can we declare an inner class inside an interface
3) Can an interface be declared inside a class?
4) How can you access the outer class object inside an inner class?
5) What is the fully qualified name of an inner class?
6) Can an inner class be abstract, final, or static?
7) Can an inner class override methods of its outer class?
8) How do you access a private inner class from another class?
9) Can an inner class shadow a variable name from its outer class?
10) How can we resolve naming conflicts between outer and inner class members?

Member Inner Class

Interview Questions

1) What is a member inner class?
2) Can a member inner class have constructors?
3) How do you create an object of a member inner class?
4) Can a member inner class have private/protected access modifiers?
5) Can we declare static methods inside a member inner class?
6) Can a member inner class extend a class and implement an interface?

Static Nested Class

Interview Questions

1) What is a static nested class?
2) Difference between static nested class & member inner class?
3) Can static nested class access this of outer class?
4) How do you create an instance of static nested class?
5) Can static nested class have static methods?
6) Why do we prefer static nested class?
7) Can a static nested class access outer class private variables?

Local Inner Class

Interview Questions

1) What is a Local Inner Class?

2) Can we declare static methods inside a Local Inner Class?

3) Why can Local Inner Class only access effectively final variables?

4) Where can you use Local Inner Classes?

5) Can Local Inner Class be marked public or private?

6) How to access Local Inner Class outside the method?

7) Example use case in real-time? (Security, event handling)?

Anonymous Inner Class

Interview Questions

1) What is an Anonymous Inner Class?

2) Why does it not need a name?

3) Can we create a constructor inside anonymous class?

4) Can anonymous inner class implement multiple interfaces?

5) Is anonymous inner class compiled? If yes, what is file name?

6) What is the disadvantage of anonymous inner class?

7) Difference between lambda expression and anonymous inner class?

Nested Interface

Interview Questions

1) What is a Nested Interface?

2) Why do we declare an interface inside a class?

3) How do you implement a nested interface?

4) What are access rules for nested interfaces?

5) Real-time use case?

Programs

1) Write a Student Management system using member inner class:

Outer class: School (name, location)

Inner class: Student (name, rollNo, grade)

Display student info along with school info.

2) Implement a Company class with static nested class Employee:

Outer class: Company (companyName)

Nested class: Employee (id, name, salary)

Print employee info along with company name.

3) Create a Runnable thread using anonymous inner class
Print numbers 1 to 5 with thread name.

4) Write an interface Calculator with method int multiply(int a, int b)
Provide implementation using anonymous inner class.

5) Create a Shape class with nested interface Drawable
Interface method: draw()
Implement Drawable in Circle and Rectangle classes.

6) Create a multi-level inner class program:
Outer → Inner → InnerInner
Access all levels and print variables.

7) Implement a Thread using anonymous inner class inside a method of an outer class.

---

35. ENUM

Questions

1) What is an enum in Java, and why is it used?
2) Can an enum implement an interface or extend a class?
3) Can an enum declared inside a class have constructors and methods?
4) Is an enum inside a class implicitly static? Explain.
5) How can we access a nested enum from another class?
6) From which Java version can we use enum in a switch statement?
7) What happens if the enum variable in a switch is null?
8) What method is used to iterate through enum constants?

9) What does the ordinal() method return for enum constants?

10) What is EnumSet and EnumMap? Why are they preferred for enum iteration?

Programs

1) Write a Java program that defines an enum inside a class and prints its constants.
2) Write a program that defines an enum inside a class with a constructor and a custom method.
3) Write code to print enum constants along with their ordinal values.
4) Write code where each enum constant has a custom method, and you call it during iteration.
5) Write a code example where enum iteration triggers specific logic (method calls) for each constant.