1. HelloWorldApp.java

Theory:

This program simply prints "Hello World!" to the console. It demonstrates the basic structure of a Java program, including a class and the main() method, which serves as the entry point of any Java application.

Viva Questions:

Q: What is the purpose of the main() method?

A: It is the entry point of any standalone Java program.

Q: Why is public static void main(String[] args) used in Java?

A: public allows access from outside, static allows the method to run without creating an object, void means it returns no value, and String[] args is used for command-line input.

Q: What does System.out.println() do?

A: It prints text to the console followed by a new line.

2. FactorialCalculator.java

Theory:

This program calculates the factorial of a number using recursion. The base case stops recursion when the number is 0 or 1. The recursive call breaks down the problem into smaller instances of itself.

Viva Questions:

Q: What is recursion?

A: A method where a function calls itself.

Q: What is the base case?

A: The stopping condition of recursion. For factorial, it’s when the number is 0 or 1.

Q: What happens if the base case is not present?

A: It leads to infinite recursion and a stack overflow.

3. Fibonnacci.java

Theory:

This program prints the Fibonacci series using iteration. The series starts with 0 and 1, and each term is the sum of the previous two.

Viva Questions:

Q: What is the Fibonacci series?

A: A series where each number is the sum of the two preceding numbers, starting from 0 and 1.

Q: How is it implemented in this program?

A: Using a for loop with variable updates.

Q: What are the first five terms of the Fibonacci series?

A: 0, 1, 1, 2, 3

4. FinalKeywordDemo.java

Theory:

This program demonstrates the final keyword in Java. A final class cannot be extended, a final method cannot be overridden, and a final variable cannot be changed after initialization.

Viva Questions:

Q: What does the final keyword do?

A: It prevents further modification of a variable, method, or class.

Q: Can we inherit a final class?

A: No, final classes cannot be subclassed.

Q: Can we modify a final object’s contents?

A: Yes, but we cannot reassign the final reference.

5. ExceptionDemo.java

Theory:

This program demonstrates Java's exception handling using try, catch, and finally blocks. It handles division by zero and custom age checks using ArithmeticException.

Viva Questions:

Q: What is exception handling?

A: A mechanism to handle runtime errors gracefully.

Q: What is the use of the finally block?

A: It executes code regardless of whether an exception occurs.

Q: What is the difference between checked and unchecked exceptions?

A: Checked exceptions are checked at compile-time; unchecked at runtime.

6. RectangleTest.java

Theory:

This program demonstrates the use of constructors (default, parameterized, copy), setters, getters, and encapsulation. It calculates and displays area and perimeter.

Viva Questions:

Q: What is a constructor?

A: A special method used to initialize objects.

Q: What is a copy constructor?

A: A constructor that creates a new object using another object of the same class.

Q: What is encapsulation?

A: Binding data and methods in a class and restricting direct access to data members.

7. CheckPrime.java

Theory:

This program checks if a randomly generated number is a prime. A number is prime if it is greater than 1 and has no positive divisors other than 1 and itself.

Viva Questions:

Q: What is a prime number?

A: A number greater than 1 that has only two factors: 1 and itself.

Q: How does the program check for prime?

A: By checking divisibility from 2 to num/2.

Q: Why is 1 not a prime number?

A: It has only one positive divisor, which is itself.

8. MultiThreadDemo.java

Theory:

This program demonstrates multithreading by extending the Thread class. It creates and runs two custom threads along with the main thread using start() and run() methods.

Viva Questions:

Q: What is multithreading?

A: Running multiple threads (tasks) concurrently.

Q: How do you create a thread using the Thread class?

A: By extending the Thread class and overriding the run() method.

Q: What does Thread.sleep() do?

A: It pauses the thread for a specified time.

9. RunnableDemo.java

Theory:

This program demonstrates multithreading using the Runnable interface. Runnable allows a class to be executed by a thread without extending the Thread class.

Viva Questions:

Q: What is the difference between Thread and Runnable?

A: Thread is a class; Runnable is an interface. Runnable allows multiple inheritance.

Q: Why might you prefer Runnable?

A: It enables the class to extend another class if needed.

Q: What method must be implemented from Runnable?

A: run()

10. CalculatorFrame.java

Theory:

This program is a basic GUI calculator built using Java Swing. It takes two numbers as input and performs arithmetic operations (add, subtract, multiply, divide) with button clicks.

Viva Questions:

Q: What is Java Swing?

A: A GUI toolkit for creating window-based applications.

Q: What is JButton used for?

A: To create clickable buttons in a GUI.

Q: How is event handling done in Swing?

A: Using ActionListener and the actionPerformed() method.

11. StringOperations.java

📘 Theory:

This program demonstrates different types of constructors in Java (default, parameterized, copy) and performs operations on two strings: converting to upper/lower case, concatenation, and reversal. It utilizes encapsulation through getter and setter methods and uses StringBuilder for reversing.

❓ Viva Questions:

Q1: What are the different types of constructors used in this program?

A1: Default constructor, parameterized constructor, and copy constructor.

Q2: How is string reversal implemented?

A2: Using StringBuilder(str).reverse().toString().

Q3: What is the purpose of encapsulation here?

A3: To protect the data by using private variables and public getter/setter methods.

Q4: Why is StringBuilder preferred for reversing strings?

A4: Because it is mutable and has a built-in reverse() method.

12. SumAverageCalculator.java

📘 Theory:

This program reads a list of integers from the user, calculates their sum using a loop, and computes the average. It demonstrates basic I/O, static methods, and array handling in Java.

❓ Viva Questions:

Q1: How is the sum calculated?

A1: Using a for-each loop that iterates over the array.

Q2: Why is the average cast to double?

A2: To ensure decimal accuracy in the result and avoid integer division.

Q3: What is the return type of computeAverage()?

A3: double.

Q4: Can you explain the difference between static and non-static methods?

A4: Static methods belong to the class, not instances, and can be called without creating an object.

13. TicTacToe.java

📘 Theory:

This is a console-based implementation of the Tic Tac Toe game. It uses a 2D array to represent the board and includes logic to check for wins and draws. The game runs in a loop until a player wins or the board is full.

❓ Viva Questions:

Q1: How is the game board represented?

A1: Using a 2D character array of size 3x3.

Q2: How does the program check if a player has won?

A2: By checking all rows, columns, and both diagonals for matching symbols.

Q3: What happens when a cell is already occupied?

A3: The program prompts the user to try again.

Q4: How does the game handle a draw condition?

A4: It checks if all cells are filled and no player has won.

14. StackDemo.java

📘 Theory:

This program implements a custom stack using an array. It demonstrates stack operations: push, pop, and display. It also handles overflow and underflow conditions.

❓ Viva Questions:

Q1: What is the role of the top variable?

A1: It indicates the current top of the stack.

Q2: How is overflow handled?

A2: By checking if top == size - 1 before pushing.

Q3: How is underflow handled?

A3: By checking if top == -1 before popping.

Q4: Why is the stack implemented using an array instead of a Stack class?

A4: To manually demonstrate the working of stack data structure and control overflow/underflow manually.