Basic Java:

1) Write a program to calculate the area of a circle, rectangle, or triangle based on user input.

Source Code:

```
import java.util.*;

class Area {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.println("Choose shape: circle, rectangle, triangle");

        String shape = sc.next().toLowerCase();

        switch (shape) {

            case "circle":

                System.out.print("Enter radius: ");

                double r = sc.nextDouble();

                System.out.println("Area: " + (Math.PI * r * r));

                break;

            case "rectangle":

                System.out.print("Enter length and width: ");

                double l = sc.nextDouble(), w = sc.nextDouble();

                System.out.println("Area: " + (l * w));

                break;

            case "triangle":

                System.out.print("Enter base and height: ");

                double b = sc.nextDouble(), h = sc.nextDouble();

                System.out.println("Area: " + (0.5 * b * h));

                break;

            default:

                System.out.println("Invalid shape");

        }

    }

}
```

Output-

```
Choose shape: circle, rectangle, triangle
circle
Enter radius: 5
Area: 78.53981633974483
```

2) Create a program to check if a number is even or odd.

Source Code-

```
import java.util.Scanner;

public class EvenOdd {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter a number: ");

        int num = sc.nextInt();

        System.out.println(num % 2 == 0 ? "Even" : "Odd");

    }

}
```

Output-

```
Enter a number: 9
Odd
```

3) Implement a program to find the factorial of a given number.

Source Code-

```
import java.util.Scanner;

public class Factorial {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter a number: ");

        int num = sc.nextInt(), fact = 1;

        for (int i = 1; i <= num; i++)

            fact *= i;

        System.out.println("Factorial: " + fact);
```

```
    }

}
```

Output-

```
Enter a number: 5

Factorial: 120
```

4) Write a program to print the Fibonacci sequence up to a specified number.

Source Code-

```
import java.util.*;

public class Fibonacci {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter limit: ");

        int n = sc.nextInt(), a = 0, b = 1;

        for (int i = 0; i < n; i++) {

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

            int sum = a + b;

            a = b;

            b = sum;

        }

    }

}
```

Output-

```
Enter limit: 10

0 1 1 2 3 5 8 13 21 34
```

5) Use loops to print patterns like a triangle or square.

Source Code-

```
public class Pattern {

    public static void main(String[] args) {

        for (int i = 1; i <= 5; i++) {

            for (int j = 1; j <= i; j++)
```

```
                System.out.print("* ");

            System.out.println();

        }

    }

}
```

Output-

```
*

* *

* * *

* * * *

* * * * *
```

Data Types and Operators:

1) Explain the difference between primitive and reference data types with examples.

Primitive Data Types

- These are basic data types provided by Java.
- They store values directly in memory.
- Memory-efficient and faster in operations.
- Examples: int, double, char, boolean, etc.

```
public class PrimitiveExample {

    public static void main(String[] args) {

        int num = 10;  // Primitive type

        char letter = 'A';

        boolean isTrue = true;

        System.out.println("Integer: " + num);

        System.out.println("Character: " + letter);

        System.out.println("Boolean: " + isTrue);

    }

}
```

```
Integer: 10
```

```
Character: A

Boolean: true
```

Reference Data Types

- These store references (addresses) to objects in memory.
- Used for complex objects like Strings, Arrays, and Classes.
- Occupy more memory and are managed dynamically .

```
public class ReferenceExample {

    public static void main(String[] args) {

        String text = "Hello";  // Reference type

        int[] numbers = {1, 2, 3};  // Array (Reference type)

        System.out.println("String: " + text);

        System.out.println("Array First Element: " + numbers[0]);

    }

}

String: Hello

Array First Element: 1
```

2) Write a program to demonstrate the use of arithmetic, logical, and relational operators.

Source Code-

```
class Operators {

    public static void main(String[] args) {

        int a = 10, b = 5;

        System.out.println("Arithmetic: " + (a + b));

        System.out.println("Logical: " + (a > 5 && b < 10));

        System.out.println("Relational: " + (a > b));

    }

}
```

Output-

```
Arithmetic: 15

Logical: true

Relational: true
```

3) Create a program to convert a temperature from Celsius to Fahrenheit and vice versa.

Source Code-

```
import java.util.Scanner;

class Temperature {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter temperature: ");

        double temp = sc.nextDouble();

        System.out.println("C to F: " + ((temp * 9/5) + 32));

    }

}
```

Output-

```
Enter temperature: 100

C to F: 212.0
```

Control Flow Statements:

1) Write a program to check if a given number is prime using an if-else statement.

Source Code-

```
import java.util.Scanner;

public class PrimeCheck {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter a number: ");

        int num = sc.nextInt();

        boolean isPrime = true;

        if (num <= 1) {

            isPrime = false;
```

```
        } else {
            for (int i = 2; i <= Math.sqrt(num); i++) {
                if (num % i == 0) {
                    isPrime = false;
                    break;
                }
            }
        }
        if (isPrime)
            System.out.println(num + " is a Prime Number");
        else
            System.out.println(num + " is NOT a Prime Number");
        sc.close();
    }
}
```

Output-

```
Enter a number: 7

7 is a Prime Number
```

2) Implement a program to find the largest number among three given numbers using a conditional statement.

Source Code-

```
import java.util.Scanner;

public class LargestNumber {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter three numbers: ");

        int a = sc.nextInt();

        int b = sc.nextInt();

        int c = sc.nextInt();
```

```
        int largest = (a > b) ? ((a > c) ? a : c) : ((b > c) ? b : c);

        System.out.println("Largest number: " + largest);

        sc.close();

    }

}
```

Output-

```
Enter three numbers: 12 45 23
Largest  Number: 45
```

3) Use a for loop to print a multiplication table.

Source Code-

```
public class MultiplicationTable {

    public static void main(String[] args) {

        int num = 5;  // Change num for different table

        for (int i = 1; i <= 10; i++) {

            System.out.println(num + " x " + i + " = " + (num * i));

        }

    }

}
```

Output-

```
5 x 1 = 5

5 x 2 = 10

5 x 3 = 15

5 x 4 = 20

5 x 5 = 25

5 x 6 = 30

5 x 7 = 35

5 x 8 = 40

5 x 9 = 45

5 x 10 = 5
```

4) Create a program to calculate the sum of even numbers from 1 to 10 using a while loop.

Source Code-

```
public class SumEvenNumbers {

    public static void main(String[] args) {

        int sum = 0, i = 2;

        while (i <= 10) {

            sum += i;

            i += 2;

        }

        System.out.println("Sum of even numbers from 1 to 10: " + sum);

    }

}
```

Output-

```
Sum of even numbers from 1 to 10: 30
```

Arrays:

1) Write a program to find the average of elements in an array.

Source Code-

```
public class AverageArray {

    public static void main(String[] args) {

        int[] arr = {10, 20, 30, 40, 50};

        int sum = 0;

        for (int num : arr) {

            sum += num;

        }

        double average = (double) sum / arr.length;

        System.out.println("Average: " + average);

    }

}
```

Output-

```
Average: 30.0
```

2) Implement a function to sort an array in ascending order using bubble sort or selection sort.

Source Code-

```
import java.util.Arrays;

public class BubbleSort {

    public static void main(String[] args) {

        int[] arr = {5, 2, 9, 1, 5, 6};

        for (int i = 0; i < arr.length - 1; i++) {

            for (int j = 0; j < arr.length - i - 1; j++) {

                if (arr[j] > arr[j + 1]) {

                    int temp = arr[j];

                    arr[j] = arr[j + 1];

                    arr[j + 1] = temp;

                }

            }

        }

        System.out.println("Sorted Array: " + Arrays.toString(arr));

    }

}
```

Output-

```
Sorted Array: [1, 2, 5, 5, 6, 9]
```

3) Create a program to search for a specific element within an array using linear search.

Source Code-

```
import java.util.Scanner;

public class LinearSearch {

    public static void main(String[] args) {

        int[] arr = {10, 20, 30, 40, 50};

        Scanner sc = new Scanner(System.in);
```

```
        System.out.print("Enter element to search: ");
        int key = sc.nextInt();
        boolean found = false;
        for (int i = 0; i < arr.length; i++) {
            if (arr[i] == key) {
                found = true;
                System.out.println("Element found at index: " + i);
                break;
            }
        }
        if (!found) {
            System.out.println("Element not found.");
        }
        sc.close();
    }
}
```

Output-

```
Enter element to search: 30
Element found at index: 2
```

Object Oriented Programming (OOP):

1) Create a class to represent a student with attributes like name, roll number, and marks.

Implement inheritance to create a "GraduateStudent" class that extends the "Student" class with additional features.

Source Code-

```
class Student {
    String name;
    int rollNumber;
    public Student(String name, int rollNumber) {
        this.name = name;
        this.rollNumber = rollNumber;
```

```
    }

    public void display() {

        System.out.println("Name: " + name + ", Roll No: " + rollNumber);

    }

}

class GraduateStudent extends Student {

    String specialization;

    public GraduateStudent(String name, int rollNumber, String
specialization) {

        super(name, rollNumber);

        this.specialization = specialization;

    }

    public void display() {

        super.display();

        System.out.println("Specialization: " + specialization);

    }

}

public class InheritanceExample {

    public static void main(String[] args) {

        GraduateStudent gradStudent = new GraduateStudent("Amit", 102,
"Computer Science");

        gradStudent.display();

    }

}
```

Output-

```
Name: Amit, Roll No: 102

Specialization: Computer Science
```

2) Demonstrate polymorphism by creating methods with the same name but different parameters in a parent and child class.

Source Code-

```
class Calculator {

    public int add(int a, int b) {

        return a + b;

    }

    public double add(double a, double b) {

        return a + b;

    }

}

public class PolymorphismDemo {

    public static void main(String[] args) {

        Calculator calc = new Calculator();

        System.out.println(calc.add(5, 10));      // Calls int method

        System.out.println(calc.add(5.5, 10.5));  // Calls double method

    }

}
```

Output-

```
15

16.0
```

3) Explain the concept of encapsulation with a suitable example.

Encapsulation is defined as the wrapping up of data under a single unit. It is the mechanism that binds together code and the data it manipulates.

Source Code-

```
class Student {

    private String name;

    private int rollNumber;

    private double marks;
```

```
    public Student(String name, int rollNumber, double marks) {
        this.name = name;
        this.rollNumber = rollNumber;
        this.marks = marks;
    }
    public void display() {
        System.out.println("Name: " + name + ", Roll No: " + rollNumber +
", Marks: " + marks);
    }
}
public class EncapsulationDemo {
    public static void main(String[] args) {
        Student student = new Student("Rahul", 101, 85.5);
        student.display();
    }
}
```

Output-

```
Name: Rahul, Roll No: 101, Marks: 85.5
```

String Manipulation:

1) Write a program to reverse a given string.

Source Code-

```
import java.util.Scanner;
public class ReverseString {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String str = sc.nextLine();
        String reversed = new StringBuilder(str).reverse().toString();
        System.out.println("Reversed String: " + reversed);
        sc.close();
```

```
    }

}
```

Output-

```
Enter a string: Java

Reversed String: avaJ
```

2) Implement a function to count the number of vowels in a string.

Source Code-

```
public class VowelCount {

    public static void main(String[] args) {

        String str = "Hello World";

        int count = 0;

        for (char ch : str.toLowerCase().toCharArray()) {

            if ("aeiou".indexOf(ch) != -1) {

                count++;

            }

        }

        System.out.println("Number of vowels: " + count);

    }

}
```

Output-

```
Number of vowels: 3
```

3) Create a program to check if two strings are anagrams.

Source Code-

```
import java.util.Arrays;

public class AnagramCheck {

    public static boolean isAnagram(String str1, String str2) {

        char[] arr1 = str1.replaceAll("\\s",
"").toLowerCase().toCharArray();

        char[] arr2 = str2.replaceAll("\\s",
"").toLowerCase().toCharArray();

        Arrays.sort(arr1);

        Arrays.sort(arr2);
```

```
        return Arrays.equals(arr1, arr2);
    }
    public static void main(String[] args) {
        System.out.println(isAnagram("listen", "silent"));  // true
        System.out.println(isAnagram("hello", "world"));   // false
    }
}
```

Output-

```
true
false
```

Advanced Topics:

1) Explain the concept of interfaces and abstract classes with examples.

Source Code-

```
interface Vehicle {
    void start();
}
abstract class Car implements Vehicle {
    abstract void fuelType();
}
class Tesla extends Car {
    public void start() {
        System.out.println("Tesla starts silently!");
    }
    public void fuelType() {
        System.out.println("Tesla runs on electricity.");
    }
}
public class InterfaceAbstractDemo {
    public static void main(String[] args) {
```

```
        Tesla myCar = new Tesla();

        myCar.start();

        myCar.fuelType();

    }

}
```

Output-

```
Tesla starts silently!

Tesla runs on electricity.
```

2) Create a program to handle exceptions using try-catch blocks.

Source Code-

```
public class ExceptionHandling {

    public static void main(String[] args) {

        try {

            int result = 10 / 0;

            System.out.println(result);

        } catch (ArithmeticException e) {

            System.out.println("Error: Division by zero is not allowed.");

        }

    }

}
```

Output-

```
Error: Division by zero is not allowed.
```

3) Implement a simple file I/O operation to read data from a text file.

Source Code-

```
import java.io.*;

public class FileReadExample {

    public static void main(String[] args) {

        try {

            BufferedReader br = new BufferedReader(new
FileReader("file.txt"));
```

```
            String line;
            while ((line = br.readLine()) != null) {
                System.out.println(line);
            }
            br.close();
        } catch (IOException e) {
            System.out.println("Error reading file: " + e.getMessage());
        }
    }
}
```

Output-

```
Hello
```

4) Explore multithreading in Java to perform multiple tasks concurrently.

Source Code-

```
class MyThread extends Thread {
    public void run() {
        for (int i = 1; i <= 5; i++) {
            System.out.println(Thread.currentThread().getName() + " -
Count: " + i);
            try {
                Thread.sleep(500);
            } catch (InterruptedException e) {
                System.out.println(e);
            }
        }
    }
}
public class MultiThreadingDemo {
    public static void main(String[] args) {
```

```
        MyThread t1 = new MyThread();

        MyThread t2 = new MyThread();

        t1.start();

        t2.start();

    }

}
```

Output-

```
Thread-1 - Count: 1

Thread-0 - Count: 1

Thread-0 - Count: 2

Thread-1 - Count: 2

Thread-1 - Count: 3

Thread-0 - Count: 3

Thread-1 - Count: 4

Thread-0 - Count: 4

Thread-0 - Count: 5

Thread-1 - Count: 5
```