Program on various ways to accept data through keyboard.

a. WAP to calculate Factorial of a number using manual input.

b. WAP to perform addition of two numbers using command line arguments

a.

```
import java.util.Scanner;

public class FactorialCalculator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number to find its factorial: ");
        int number = scanner.nextInt();
        int factorial = 1;
        for (int i = 1; i <= number; i++) {
            factorial *= i;
        }
        System.out.println("Factorial of " + number + " is: " + factorial);
    }
}
```

b.

```
public class AdditionWithCommandLineArgs {
    public static void main(String[] args) {
        if (args.length < 2) {
            System.out.println("Please provide two numbers as command line arguments.");
            return;
        }
        int num1 = Integer.parseInt(args[0]);
        int num2 = Integer.parseInt(args[1]);
        int sum = num1 + num2;
        System.out.println("Sum of " + num1 + " and " + num2 + " is: " + sum);
    }
}
```

Q. Bank Account wala question

a.

```
public class BankAccount {
    String depositorName;
    long accountNumber;
    String accountType;
    double balance;

    // Method to assign initial values
    public void initializeValues(String name, long accNumber, String accType, double initialBalance) {
        depositorName = name;
        accountNumber = accNumber;
        accountType = accType;
        balance = initialBalance;
    }

    // Method to deposit an amount
    public void deposit(double amount) {
        balance += amount;
        System.out.println("Amount deposited successfully. Current balance: " + balance);
    }

    // Method to withdraw an amount after checking balance
    public void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
            System.out.println("Amount withdrawn successfully. Current balance: " + balance);
        } else {
            System.out.println("Insufficient balance. Withdrawal not possible.");
        }
    }

    // Method to display the name and balance
    public void displayDetails() {
        System.out.println("Depositor Name: " + depositorName);
        System.out.println("Balance: " + balance);
    }

    // Main method for demonstration
    public static void main(String[] args) {
        BankAccount myAccount = new BankAccount();
        myAccount.initializeValues("John Doe", 123456789, "Savings", 5000);
        myAccount.displayDetails();
        myAccount.deposit(3000);
        myAccount.withdraw(2000);
    }
}
```

b.

```
public class Employee {
    String name;
    int age;

    long phoneNumber;

    double basicSalary;

    int presentDays;


    // Method to take input

    public void takeInput(String empName, int empAge, long phoneNum, double salary, int
daysPresent) {

        name = empName;

        age = empAge;

        phoneNumber = phoneNum;

        basicSalary = salary;

        presentDays = daysPresent;

    }


    // Method to display all values along with gross salary

    public void displayDetails() {

        System.out.println("Name: " + name);

        System.out.println("Age: " + age);

        System.out.println("Phone Number: " + phoneNumber);

        System.out.println("Basic Salary: " + basicSalary);

        double grossSalary = basicSalary * presentDays / 30;

        System.out.println("Gross Salary: " + grossSalary);

    }


    // Main method for demonstration

    public static void main(String[] args) {

        Employee employee = new Employee();

        employee.takeInput("Jane Doe", 30, 1234567890, 50000, 25);

        employee.displayDetails();

    }
```

```
}
```

Q. number of occurence of given character

```
public class CharacterCount {
    public static int countOccurrences(String inputString, char character) {
        int count = 0;
        for (int i = 0; i < inputString.length(); i++) {
            if (inputString.charAt(i) == character) {
                count++;
            }
        }
        return count;
    }

    public static void main(String[] args) {
        String inputString = "Hello, how are you?";
        char characterToCount = 'o';
        int occurrences = countOccurrences(inputString, characterToCount);
        System.out.println("Number of occurrences of '" + characterToCount + "': " + occurrences);
    }
}
```

Q. String reverse

```
public class SentenceReversal {
    public static String reverseSentence(String sentence) {
        StringBuffer stringBuffer = new StringBuffer(sentence);
        stringBuffer.reverse();
        String reversedSentence = stringBuffer.toString();
        return reversedSentence;
    }

    public static void main(String[] args) {
        String sentence = "Hello, how are you?";
        String reversedSentence = reverseSentence(sentence);
        System.out.println("Reversed Sentence: " + reversedSentence);
    }
}
```

Q. Program on Constructor and Constructor Overloading

```
//Contructor Loading and overloading
public class Constructor {
    int value;
    String name;

    Constructor() {
        value = 0;
        name = "Default";
    }



    Constructor(int v, String n) {
        value = v;
        name = n;
    }

    Constructor(int v) {
        value = v;
        name = "Overloaded";
    }


    public static void main(String[] args) {
        Constructor obj1 = new Constructor();
        Constructor obj2 = new Constructor(5, "John");
        Constructor obj3 = new Constructor(10);

        System.out.println("Object 1 - Value: " + obj1.value + ", Name: " + obj1.name);
        System.out.println("Object 2 - Value: " + obj2.value + ", Name: " + obj2.name);
        System.out.println("Object 3 - Value: " + obj3.value + ", Name: " + obj3.name);
    }
}
```

Q. Implementing 3 classes

```
// Interface Sports
```

```
interface Sports {
    int score = 10; // Default score for demonstration purposes
}

// Student class
class Student {
    int rollNo;

    // Constructor
    Student(int rollNo) {
        this.rollNo = rollNo;
    }
}

// Test class derived from Student
class Test extends Student {
    int sem1Marks;
    int sem2Marks;

    // Constructor
    Test(int rollNo, int sem1Marks, int sem2Marks) {
        super(rollNo);
        this.sem1Marks = sem1Marks;
        this.sem2Marks = sem2Marks;
    }
}

// Result class with multiple inheritance from Test and Sports
class Result extends Test implements Sports {
    int total;
    // Constructor

    Result(int rollNo, int sem1Marks, int sem2Marks) {

        super(rollNo, sem1Marks, sem2Marks);

        this.total = sem1Marks + sem2Marks + score;

    }

}


// Main class for execution

public class Main {

    public static void main(String[] args) {

        Result result = new Result(123, 80, 85);

        System.out.println("Roll No: " + result.rollNo);

        System.out.println("Semester 1 Marks: " + result.sem1Marks);

        System.out.println("Semester 2 Marks: " + result.sem2Marks);

        System.out.println("Total Marks: " + result.total);

    }

}
```

Q. Abstract classes

```
abstract class Shape {

    abstract double calculateArea();

}


class Circle extends Shape {

    double radius;


    Circle(double radius) {
        this.radius = radius;
    }

    double calculateArea() {
        return Math.PI * radius * radius;
    }
}

class Rectangle extends Shape {
    double length;
    double width;

    Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    double calculateArea() {
        return length * width;
    }
}

class Triangle extends Shape {
    double base;
    double height;

    Triangle(double base, double height) {
        this.base = base;
        this.height = height;
    }
    double calculateArea() {
        return 0.5 * base * height;
    }
}

public class Main {
    public static void main(String[] args) {
        Circle circle = new Circle(5);
        Rectangle rectangle = new Rectangle(4, 6);
        Triangle triangle = new Triangle(3, 4);

        System.out.println("Area of Circle: " + circle.calculateArea());
        System.out.println("Area of Rectangle: " + rectangle.calculateArea());
        System.out.println("Area of Triangle: " + triangle.calculateArea());
    }
}
```

Q. Exception handling using try/catch throw

```
public class ExceptionHandlingExample {
    public static void main(String[] args) {
        try {
            int a = 10, b = 0;
            int result = a / b;
            System.out.println("Result: " + result);
        } catch (ArithmeticException e) {
            System.out.println("ArithmeticException: " + e.getMessage());
        }

        try {
            int[] arr = {1, 2, 3};
            System.out.println("Value at index 5: " + arr[5]);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("ArrayIndexOutOfBoundsException: " + e.getMessage());
        } finally {
            System.out.println("Finally block always executes.");
        }

        try {
            throw new CustomException("This is a custom exception message.");
        } catch (CustomException e) {
            System.out.println("CustomException: " + e.getMessage());
        }
    }

    static class CustomException extends Exception {
        public CustomException(String message) {
            super(message);
        }
    }
}
```

Q. No Match Exception wala question

```java
class NoMatchException extends Exception {
    public NoMatchException(String message) {
        super(message);
    }
}

public class Main {
    public static void main(String[] args) {
        String inputString = "India";

        try {
            if (!inputString.equals("India")) {
                throw new NoMatchException("String does not match 'India'");
            } else {
                System.out.println("String matches 'India'");
            }
        } catch (NoMatchException e) {
            System.out.println("NoMatchException: " + e.getMessage());
        }
    }
}
```