Set A

B: Write a program to calculate perimeter and area of rectangle. (hint: area length breadth, perimeter=2*(length+breadth))

```
import java.util.Scanner;

public class Rectangle {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter length of rectangle: ");
        double length = scanner.nextDouble();

        System.out.print("Enter breadth of rectangle: ");
        double breadth = scanner.nextDouble();

        // Calculate area
        double area = length * breadth;
        System.out.println("Area of rectangle: " + area);

        // Calculate perimeter
        double perimeter = 2 * (length + breadth);
        System.out.println("Perimeter of rectangle: " + perimeter);

        scanner.close();
    }
}
```

B: Write a menu driven program to perform the following operations

I. Calculate the volume of cylinder. (hint: Volume: $\pi \times r^2 \times h$)

II. Find the factorial of given number.

III. Check the number is Armstrong or not.

IV. Exit

```
import java.util.Scanner;

public class MenuDrivenProgram {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int choice;

        do {
            System.out.println("Menu:");
            System.out.println("1. Calculate the volume of cylinder");
            System.out.println("2. Find the factorial of a given number");
            System.out.println("3. Check if a number is Armstrong or not");
            System.out.println("4. Exit");
            System.out.print("Enter your choice: ");

            choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    System.out.print("Enter radius of cylinder: ");
```

```
            double radius = scanner.nextDouble();
            System.out.print("Enter height of cylinder: ");
            double height = scanner.nextDouble();
            double volume = Math.PI * Math.pow(radius, 2) * height;
            System.out.println("Volume of cylinder: " + volume);
            break;
        case 2:
            System.out.print("Enter a number to find factorial: ");
            int num = scanner.nextInt();
            long factorial = 1;
            for (int i = 1; i <= num; i++) {
                factorial *= i;
            }
            System.out.println("Factorial of " + num + " is: " + factorial);
            break;
        case 3:
            System.out.print("Enter a number to check
Armstrong: ");
            int number = scanner.nextInt();
            int originalNumber = number;
            int sum = 0;
            while (number != 0) {
                int remainder = number % 10;
                sum += Math.pow(remainder, 3);
                number /= 10;
            }
            if (sum == originalNumber) {
                System.out.println(originalNumber + " is an Armstrong number.");
            } else {
                System.out.println(originalNumber + " is not an Armstrong number.");
            }
            break;
```

```
            case 4:
                System.out.println("Exiting... Thank you!");
                break;
             default:
                System.out.println("Invalid choice. Please enter a
valid option.");
          }

          System.out.println(); // Blank line for readability

       } while (choice != 4);

       scanner.close();
   }
}
```

```
import java.util.Scanner;

public class ReverseArray {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the size of the array: ");
        int size = scanner.nextInt();

        int[] arr = new int[size];

        System.out.println("Enter elements of the array:");
        for (int i = 0; i < size; i++) {
            arr[i] = scanner.nextInt();
        }

        System.out.println("Array elements in reverse order:");
        for (int i = size - 1; i >= 0; i--) {
            System.out.print(arr[i] + " ");
        }

        scanner.close();
    }
}
```

Set B

```
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Calendar;

public class DateTimeFormats {
    public static void main(String[] args) {
        // Create a Date object to get the current date and time
        Date now = new Date();

        // Display date in "dd/MM/yyyy" format
        SimpleDateFormat dateFormat1 = new
SimpleDateFormat("dd/MM/yyyy");
        System.out.println("Current date is: " +
dateFormat1.format(now));

        // Display date in "MM-dd-yyyy" format
        SimpleDateFormat dateFormat2 = new
SimpleDateFormat("MM-dd-yyyy");
        System.out.println("Current date is: " +
dateFormat2.format(now));

        // Display date in "EEEE MMMM dd yyyy" format
        SimpleDateFormat dateFormat3 = new
SimpleDateFormat("EEEE MMMM dd yyyy");
        System.out.println("Current date is: " +
dateFormat3.format(now));
```

```
        // Display date and time in "EEE MMMM dd HH:mm:ss z
yyyy" format
        SimpleDateFormat dateFormat4 = new
SimpleDateFormat("EEE MMMM dd HH:mm:ss z yyyy");
        System.out.println("Current date and time is: " +
dateFormat4.format(now));

        // Display date and time in "dd/MM/yy HH:mm:ss a Z" format
        SimpleDateFormat dateFormat5 = new
SimpleDateFormat("dd/MM/yy HH:mm:ss a Z");
        System.out.println("Current date and time is: " +
dateFormat5.format(now));

        // Display time in "HH:mm:ss" format
        SimpleDateFormat timeFormat = new
SimpleDateFormat("HH:mm:ss");
        System.out.println("Current time is: " +
timeFormat.format(now));

        // Get current week of the year
        Calendar calendar = Calendar.getInstance();
        calendar.setTime(now);
        int weekOfYear = calendar.get(Calendar.WEEK_OF_YEAR);
        System.out.println("Current week of year is: " +
weekOfYear);

        // Get current week of the month
        int weekOfMonth =
calendar.get(Calendar.WEEK_OF_MONTH);
        System.out.println("Current week of month is: " +
weekOfMonth);
```

```
        // Get current day of the year
        int dayOfYear = calendar.get(Calendar.DAY_OF_YEAR);
        System.out.println("Current day of the year is: " +
dayOfYear);
    }
}
```

b) Define a class MyNumber having one private int data member. Write a default constructor to initialize it to O and another constructor to initialize it to a value (Use this). Write methods is Negative, is Positive, isZero, isOdd, isEven. Create an object in main. Use command line arguments to pass a value to the object (Hint: convert string argument to integer) and perform the above tests. Provide javadoc comments for all constructors and methods and generate the html help file.

```
public class MyNumber {
    private int data;       // Private data member to hold the number

/*** Default constructor that initializes the data member to 0.*/

 public MyNumber() {
        this.data = 0;
    }

    /**
    * Parameterized constructor that initializes the data member to the specified value.
    * @param value The value to initialize the data member with.
    */
```

```
public MyNumber(int value) {
        this.data = value;
    }

    /**
     * Checks if the number is negative.
     * @return true if the number is negative, false otherwise.
     */

public boolean isNegative() {
        return this.data < 0;
    }

    /**
     * Checks if the number is positive.
     * @return true if the number is positive, false otherwise.
     */
    public boolean isPositive() {
        return this.data > 0;
    }

    /**
     * Checks if the number is zero.
     * @return true if the number is zero, false otherwise.
     */
    public boolean isZero() {
        return this.data == 0;
    }

    /**
     * Checks if the number is odd.
     * @return true if the number is odd, false otherwise.
```

```
     */
    public boolean isOdd() {
        return this.data % 2 != 0;
    }

    /**
     * Checks if the number is even.
     * @return true if the number is even, false otherwise.
     */
    public boolean isEven() {
        return this.data % 2 == 0;
    }

    public static void main(String[] args) {
        // Ensure there is at least one argument
        if (args.length < 1) {
            System.out.println("Please provide a number as a
command line argument.");
            return;
        }

        try {
            // Convert command line argument to integer
            int number = Integer.parseInt(args[0]);

            // Create a MyNumber object with the provided value
            MyNumber myNumber = new MyNumber(number);

            // Perform tests and print results
            System.out.println("Number: " + number);
            System.out.println("Is Negative? " +
myNumber.isNegative());
```

```
            System.out.println("Is Positive? " +
myNumber.isPositive());
            System.out.println("Is Zero? " + myNumber.isZero());
            System.out.println("Is Odd? " + myNumber.isOdd());
            System.out.println("Is Even? " + myNumber.isEven());

        } catch (NumberFormatException e) {
            System.out.println("Invalid number format. Please provide
an integer.");
        }
    }
}
```

c) Write a menu driven program to perform the following operations on multi dimensional array ie matrix:

I Addition

II Multiplication

III Transpose of any matrix.

IV. Exit

```
import java.util.Scanner;

public class MatrixOperations {

    private static Scanner scanner = new Scanner(System.in);

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

```
        int choice;
        do {
            System.out.println("Menu:");
            System.out.println("1. Addition");
            System.out.println("2. Multiplication");
            System.out.println("3. Transpose");
            System.out.println("4. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    addMatrices();
                    break;
                case 2:
                    multiplyMatrices();
                    break;
                case 3:
                    transposeMatrix();
                    break;
                case 4:
                    System.out.println("Exiting...");
                    break;
                default:
                    System.out.println("Invalid choice. Please try
again.");
            }
        } while (choice != 4);
    }

    private static void addMatrices() {
        System.out.print("Enter number of rows for matrices: ");
```

```
        int rows = scanner.nextInt();
        System.out.print("Enter number of columns for matrices: ");
        int cols = scanner.nextInt();

        int[][] matrix1 = new int[rows][cols];
        int[][] matrix2 = new int[rows][cols];
        int[][] result = new int[rows][cols];

        System.out.println("Enter elements of first matrix:");
        inputMatrix(matrix1);

        System.out.println("Enter elements of second matrix:");
        inputMatrix(matrix2);

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                result[i][j] = matrix1[i][j] + matrix2[i][j];
            }
        }

        System.out.println("Sum of matrices:");
        printMatrix(result);
    }

    private static void multiplyMatrices() {
        System.out.print("Enter number of rows for first matrix: ");
        int rows1 = scanner.nextInt();
        System.out.print("Enter number of columns for first matrix:
");
        int cols1 = scanner.nextInt();
```

```
        System.out.print("Enter number of rows for second matrix:
");
        int rows2 = scanner.nextInt();
        System.out.print("Enter number of columns for second
matrix: ");
        int cols2 = scanner.nextInt();

        if (cols1 != rows2) {
            System.out.println("Matrix multiplication is not
possible.");
            return;
        }

        int[][] matrix1 = new int[rows1][cols1];
        int[][] matrix2 = new int[rows2][cols2];
        int[][] result = new int[rows1][cols2];

        System.out.println("Enter elements of first matrix:");
        inputMatrix(matrix1);

        System.out.println("Enter elements of second matrix:");
        inputMatrix(matrix2);

        for (int i = 0; i < rows1; i++) {
            for (int j = 0; j < cols2; j++) {
                result[i][j] = 0;
                for (int k = 0; k < cols1; k++) {
                    result[i][j] += matrix1[i][k] * matrix2[k][j];
                }
            }
        }
```

```
    System.out.println("Product of matrices:");
    printMatrix(result);
}

private static void transposeMatrix() {
    System.out.print("Enter number of rows for matrix: ");
    int rows = scanner.nextInt();
    System.out.print("Enter number of columns for matrix: ");
    int cols = scanner.nextInt();

    int[][] matrix = new int[rows][cols];
    int[][] transposed = new int[cols][rows];

    System.out.println("Enter elements of matrix:");
    inputMatrix(matrix);

    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            transposed[j][i] = matrix[i][j];
        }
    }

    System.out.println("Transposed matrix:");
    printMatrix(transposed);
}

private static void inputMatrix(int[][] matrix) {
    for (int i = 0; i < matrix.length; i++) {
        for (int j = 0; j < matrix[i].length; j++) {
            matrix[i][j] = scanner.nextInt();
        }
    }
```

```
    }

    private static void printMatrix(int[][] matrix) {
        for (int[] row : matrix) {
            for (int element : row) {
                System.out.print(element + " ");
            }
            System.out.println();
        }
    }
}
```

Set C

a) Write a program to accept n names of countries and display them in descending order.

```
import java.util.Arrays;
import java.util.Scanner;

public class CountrySorter {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Input number of countries
        System.out.print("Enter the number of countries: ");
        int n = scanner.nextInt();
        scanner.nextLine();  // Consume newline
```

```
        // Input country names
        String[] countries = new String[n];
        System.out.println("Enter the names of the countries:");
        for (int i = 0; i < n; i++) {
            countries[i] = scanner.nextLine();
        }

        // Sort the countries in descending order
        Arrays.sort(countries, (a, b) -> b.compareTo(a));

        // Display sorted countries
        System.out.println("Countries in descending order:");
        for (String country : countries) {
            System.out.println(country);
        }

        scanner.close();
    }
}
```

Practical Slip_1 …

Q1) Write a Program to print all Prime numbers in an array of 'n' elements. (use command line arguments)

```
public class PrimeNumbers {

    // Method to check if a number is prime
    public static boolean isPrime(int num) {
        if (num <= 1) return false;
        if (num == 2) return true; // 2 is the only even prime number
        if (num % 2 == 0) return false; // No other even number can be prime

        for (int i = 3; i * i <= num; i += 2) {
            if (num % i == 0) return false;
        }
        return true;
    }

    public static void main(String[] args) {
        // Check if command-line arguments are provided
        if (args.length == 0) {
            System.out.println("Please provide an array of integers.");
            return;
        }

        // Convert command-line arguments to an array of integers
        int[] numbers = new int[args.length];
```

```
        try {
            for (int i = 0; i < args.length; i++) {
                numbers[i] = Integer.parseInt(args[i]);
            }
        } catch (NumberFormatException e) {
            System.out.println("Invalid input. Please provide valid integers.");
            return;
        }

        // Print prime numbers from the array
        System.out.println("Prime numbers in the array:");
        for (int num : numbers) {
            if (isPrime(num)) {
                System.out.println(num);
            }
        }
    }
}
```

Run and compile

```
javac PrimeNumbers.java
java PrimeNumbers 2 3 4 5 6 7 8 9 10
```

Q2 Define an abstract class Staff with protected members id and name. Define a parameterized constructor. Define one subclass OfficeStaff with member department. Create n objects of OfficeStaff and display all details.

```
abstract class Staff {
    protected int id;
    protected String name;

    // Parameterized constructor
    public Staff(int id, String name) {
        this.id = id;
        this.name = name;
    }

    // Abstract method to be implemented by subclasses
    public abstract void displayDetails();
}

class OfficeStaff extends Staff {
    private String department;

    // Parameterized constructor
    public OfficeStaff(int id, String name, String department) {
        super(id, name);
        this.department = department;
    }

    // Implementation of the abstract method
    @Override
    public void displayDetails() {
```

```
        System.out.println("ID: " + id);
        System.out.println("Name: " + name);
        System.out.println("Department: " + department);
    }
}

public class Main {
    public static void main(String[] args) {
        // Create an array of OfficeStaff objects
        OfficeStaff[] staffArray = new OfficeStaff[3];

        // Initialize the OfficeStaff objects
        staffArray[0] = new OfficeStaff(1, "Alice", "HR");
        staffArray[1] = new OfficeStaff(2, "Bob", "Finance");
        staffArray[2] = new OfficeStaff(3, "Charlie", "IT");

        // Display details of all OfficeStaff objects
        System.out.println("Office Staff Details:");
        for (OfficeStaff staff : staffArray) {
            staff.displayDetails();
            System.out.println(); // Print a newline for better readability
        }
    }
}
```

Compile and Run

```
javac Main.java
java Main
```

Practical Slip_2 …

Q1) Write a program to read the First Name and Last Name of a person, his weight and height using command line arguments. Calculate the BMI Index which is defined as the individual's body mass divided by the square of their height. (Hint : BMI = Wts. In kgs/ (ht)2)

```
public class BMICalculator {
    public static void main(String[] args)
// Check if the required number of arguments is provided
        if (args.length != 4) {
            System.out.println("Usage: java BMICalculator <FirstName>
<LastName> <Weight(kg)> <Height(m)>");
            return;
        }

        // Parse command-line arguments
        String firstName = args[0];
        String lastName = args[1];
        double weight;
        double height;

        try {
            weight = Double.parseDouble(args[2]);
            height = Double.parseDouble(args[3]);
        } catch (NumberFormatException e) {
            System.out.println("Invalid number format. Weight and height
should be numeric.");
            return;
        }
```

```
        // Calculate BMI
        double bmi = weight / (height * height);

        // Display the result
        System.out.printf("Name: %s %s%n", firstName, lastName);
        System.out.printf("Weight: %.2f kg%n", weight);
        System.out.printf("Height: %.2f m%n", height);
        System.out.printf("BMI: %.2f%n", bmi);
    }
}
```

Compile and Run the program from the command line, passing the required arguments: Name Weight and Height

```
javac BMICalculator.java
java BMICalculator John Doe 70 1.75
```

Q2) Define a class CricketPlayer (name,no_of_innings,no_of_times_notout, totalruns, bat_avg). Create an array of n player objects Calculate the batting average for each player using static method avg(). Define a static sort method which sorts the array on the basis of average. Display the player details in sorted order.

```
import java.util.Arrays;
import java.util.Comparator;

class CricketPlayer {
    String name;
    int noOfInnings;
    int noOfTimesNotOut;
    int totalRuns;
```

```
    double batAvg;

    // Constructor
            public CricketPlayer(String name, int noOfInnings, int
noOfTimesNotOut, int totalRuns) {
        this.name = name;
        this.noOfInnings = noOfInnings;
        this.noOfTimesNotOut = noOfTimesNotOut;
        this.totalRuns = totalRuns;
        this.batAvg = avg(noOfInnings, noOfTimesNotOut, totalRuns);
    }

    // Static method to calculate batting average
     public static double avg(int noOfInnings, int noOfTimesNotOut, int
totalRuns) {
          if (noOfInnings - noOfTimesNotOut == 0) return 0; // To handle
division by zero
        return (double) totalRuns / (noOfInnings - noOfTimesNotOut);
    }

    // Method to display player details
    public void displayDetails() {
        System.out.printf("Name: %s%n", name);
        System.out.printf("Innings: %d%n", noOfInnings);
        System.out.printf("Times Not Out: %d%n", noOfTimesNotOut);
        System.out.printf("Total Runs: %d%n", totalRuns);
        System.out.printf("Batting Average: %.2f%n", batAvg);
        System.out.println();
    }

    // Static method to sort players based on their batting average
    public static void sortByAverage(CricketPlayer[] players) {
```

```
        Arrays.sort(players, Comparator.comparingDouble(p -> p.batAvg));
    }
}

public class Main {
    public static void main(String[] args) {
        // Create an array of CricketPlayer objects
        CricketPlayer[] players = new CricketPlayer[]{
            new CricketPlayer("Player1", 10, 2, 300),
            new CricketPlayer("Player2", 12, 4, 350),
            new CricketPlayer("Player3", 8, 1, 150)
        };

        // Sort players by their batting average
        CricketPlayer.sortByAverage(players);

        // Display details of all players in sorted order
        System.out.println("Cricket Player Details (Sorted by Batting Average):");
        for (CricketPlayer player : players) {
            player.displayDetails();
        }
    }
}
```

Run

```
javac Main.java
java Main
```

Practical Slip_3 …

Q1)write a programm in java to accept 'n' names of cities from the user and sort them in ascending order

```
import java.util.Scanner;
import java.util.Arrays;

public class CitySorter {
    public static void main(String[] args) {
        // Create a Scanner object for input
        Scanner sc = new Scanner(System.in);

        // Ask the user for the number of cities
        System.out.print("Enter the number of cities: ");
        int n = sc.nextInt();

        // Clear the buffer
        sc.nextLine();

        // Create an array to store the names of the cities
        String[] cities = new String[n];

        // Accept the names of the cities from the user
        System.out.println("Enter the names of the cities:");
        for (int i = 0; i < n; i++) {
            System.out.print("City " + (i + 1) + ": ");
            cities[i] = sc.nextLine();
        }

        // Sort the cities in ascending order
        Arrays.sort(cities);
```

```
        // Display the sorted list of cities
        System.out.println("\nCities in ascending order:");
        for (String city : cities) {
            System.out.println(city);
        }

        // Close the scanner
        sc.close();
    }
}
```

Q2)Define a class patient (patient_name, patient_age, patient_oxy_level,patient_HRCT_report). Create an object of patient. Handle appropriate exception while patient oxygen level less than 95% and HRCT scan report greater than 10, then throw user defined Exception "Patient is Covid Positive(+) and Need to Hospitalized" otherwise display its information

```
// User-defined Exception class
class CovidPositiveException extends Exception {
    public CovidPositiveException(String message) {
        super(message);
    }
}

// Patient class definition
class Patient {
    String patient_name;
    int patient_age;
    double patient_oxy_level;
```

```
    double patient_HRCT_report;

    // Constructor
    public Patient(String name, int age, double oxy_level, double
HRCT_report) {
        this.patient_name = name;
        this.patient_age = age;
        this.patient_oxy_level = oxy_level;
        this.patient_HRCT_report = HRCT_report;
    }

    // Method to check the patient status
    public void checkPatientStatus() throws CovidPositiveException {
        if (patient_oxy_level < 95 && patient_HRCT_report > 10) {
            throw new CovidPositiveException("Patient is Covid Positive(+) and
Needs to be Hospitalized");
        } else {
            displayPatientInfo();
        }
    }

    // Method to display patient information
    public void displayPatientInfo() {
        System.out.println("Patient Name: " + patient_name);
        System.out.println("Patient Age: " + patient_age);
        System.out.println("Patient Oxygen Level: " + patient_oxy_level +
"%");
        System.out.println("Patient HRCT Report: " + patient_HRCT_report);
        System.out.println("Patient is not Covid Positive.");
    }

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

```
        // Creating an object of Patient
        Patient patient1 = new Patient("John Doe", 45, 92, 12);

        try {
            patient1.checkPatientStatus();
        } catch (CovidPositiveException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

Practical Slip_4 …

Q1)Write a program to print an array after changing the rows and columns of a given two-dimensional array.

```
public class TransposeMatrix {

    public static void main(String[] args) {
        int[][] originalMatrix = {
            {1, 2, 3},
            {4, 5, 6},
            {7, 8, 9}
        };

        int rowCount = originalMatrix.length;
        int colCount = originalMatrix[0].length;

        // Creating a new matrix to store the transpose
        int[][] transposedMatrix = new int[colCount][rowCount];
```

```
        // Transposing the matrix
        for (int i = 0; i < rowCount; i++) {
            for (int j = 0; j < colCount; j++) {
                transposedMatrix[j][i] = originalMatrix[i][j];
            }
        }

        // Printing the transposed matrix
        System.out.println("Transposed Matrix:");
        for (int i = 0; i < colCount; i++) {
            for (int j = 0; j < rowCount; j++) {
                System.out.print(transposedMatrix[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

Run and compile

```
javac TransposeMatrix.java
java TransposeMatrix
```

Q2) Write a program to design a screen using Awt that will take a user name and password. If the user name and password are not same, raise an Exception with appropriate message. User can have 3 login chances only. Use clear button to clear the TextFields.

```
import java.awt.*;
import java.awt.event.*;

public class LoginScreen extends Frame implements
ActionListener {
    // GUI components
    TextField usernameField, passwordField;
    Button loginButton, clearButton;
    Label messageLabel;
    int loginAttempts = 3;

    // Constructor to setup the GUI components
    public LoginScreen() {
        // Set up the Frame
        setTitle("Login Screen");
        setSize(400, 200);
        setLayout(new GridLayout(4, 2));

        // Create components
        Label usernameLabel = new Label("Username:");
        usernameField = new TextField(20);

        Label passwordLabel = new Label("Password:");
        passwordField = new TextField(20);
        passwordField.setEchoChar('*');  // To hide the password
characters

        loginButton = new Button("Login");
```

```
    clearButton = new Button("Clear");

    messageLabel = new Label();

    // Add components to the Frame
    add(usernameLabel);
    add(usernameField);
    add(passwordLabel);
    add(passwordField);
    add(loginButton);
    add(clearButton);
    add(messageLabel);

    // Add action listeners
    loginButton.addActionListener(this);
    clearButton.addActionListener(this);

    // Window close event
    addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent we) {
            System.exit(0);
        }
    });

    setVisible(true);
}

// Event handling
public void actionPerformed(ActionEvent ae) {
    if (ae.getSource() == loginButton) {
        try {
            checkLogin();
```

```
            } catch (Exception e) {
                messageLabel.setText(e.getMessage());
            }
        } else if (ae.getSource() == clearButton) {
            usernameField.setText("");
            passwordField.setText("");
            messageLabel.setText("");
        }
    }

    // Method to check login
    private void checkLogin() throws Exception {
        String username = usernameField.getText();
        String password = passwordField.getText();

        if (username.equals(password)) {
            messageLabel.setText("Login successful!");
        } else {
            loginAttempts--;
            if (loginAttempts > 0) {
                throw new Exception("Username and password do not
match. Attempts left: " + loginAttempts);
            } else {
                throw new Exception("Login failed. No attempts left.");
            }
        }
    }

    // Main method
    public static void main(String[] args) {
        new LoginScreen();
    }
```

}

Run and compile

```
javac LoginScreen.java
```

```
java LoginScreen
```

Practical Slip_5 …

Q1) Write a program for multilevel inheritance such that Country is inherited from Continent. State is inherited from Country. Display the place, State, Country and Continent.

```
// Base class Continent
class Continent {
    String continentName;

    Continent(String continentName) {
        this.continentName = continentName;
    }

    void displayContinent() {
        System.out.println("Continent: " + continentName);
    }
}

// Derived class Country, inheriting from Continent
class Country extends Continent {
```

```
    String countryName;

    Country(String continentName, String countryName) {
        super(continentName);  // Call to parent class constructor
        this.countryName = countryName;
    }

    void displayCountry() {
        displayContinent();
        System.out.println("Country: " + countryName);
    }
}

// Derived class State, inheriting from Country
class State extends Country {
    String stateName;
    String place;

        State(String continentName, String countryName, String
stateName, String place) {
        super(continentName, countryName);  // Call to parent class
constructor
        this.stateName = stateName;
        this.place = place;
    }

    void displayState() {
        displayCountry();
        System.out.println("State: " + stateName);
        System.out.println("Place: " + place);
```

```
    }
}

// Main class to test the multilevel inheritance
public class MultilevelInheritance {
    public static void main(String[] args) {
        // Create an object of State
                State state = new State("Asia", "India", "Karnataka",
"Bangalore");

        // Display the hierarchy
        state.displayState();
    }
}
```

Compile and Run

```
javac MultilevelInheritance.java
java MultilevelInheritance
```

Q2) Write a menu driven program to perform the following operations on multidimensional array ie matrices

1. **Addition**
2. **Multiplication**
3. **Exit**

```
import java.util.Scanner;

public class MatrixOperations {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int choice;

        while (true) {
            System.out.println("\nMatrix Operations Menu:\n1.
Addition\n2. Multiplication\n3. Exit\nEnter your choice: ");
            choice = scanner.nextInt();

            if (choice == 3) break;

            System.out.print("Enter dimensions (rows and
columns) of matrix 1: ");
            int rows1 = scanner.nextInt(), cols1 =
scanner.nextInt();
            int[][] matrix1 = new int[rows1][cols1];
            System.out.println("Enter elements of matrix 1:");
            for (int i = 0; i < rows1; i++)
                for (int j = 0; j < cols1; j++)
                    matrix1[i][j] = scanner.nextInt();

            if (choice == 1) { // Addition
                int[][] matrix2 = new int[rows1][cols1];
                System.out.println("Enter elements of matrix 2:");
                for (int i = 0; i < rows1; i++)
                    for (int j = 0; j < cols1; j++)
```

```
                matrix2[i][j] = scanner.nextInt();

            System.out.println("Resultant Matrix after
Addition:");
            for (int i = 0; i < rows1; i++) {
               for (int j = 0; j < cols1; j++)
                  System.out.print((matrix1[i][j] + matrix2[i][j]) + "
");
               System.out.println();
            }
         } else if (choice == 2) { // Multiplication
            System.out.print("Enter dimensions (rows and
columns) of matrix 2: ");
            int rows2 = scanner.nextInt(), cols2 =
scanner.nextInt();
            if (cols1 != rows2) {
               System.out.println("Multiplication not possible.
Columns of matrix 1 must equal rows of matrix 2.");
               continue;
            }
            int[][] matrix2 = new int[rows2][cols2], result = new
int[rows1][cols2];
            System.out.println("Enter elements of matrix 2:");
            for (int i = 0; i < rows2; i++)
               for (int j = 0; j < cols2; j++)
                  matrix2[i][j] = scanner.nextInt();

            System.out.println("Resultant Matrix after
Multiplication:");
            for (int i = 0; i < rows1; i++) {
```

```
                for (int j = 0; j < cols2; j++) {
                    result[i][j] = 0;
                    for (int k = 0; k < cols1; k++)
                        result[i][j] += matrix1[i][k] * matrix2[k][j];
                    System.out.print(result[i][j] + " ");
                }
                System.out.println();
            }
        } else {
            System.out.println("Invalid choice. Please choose
again.");
        }
    }
    scanner.close();
    System.out.println("Exiting the program.");
  }
}
```

Compile and Run

```
javac MatrixOperations.java
java MatrixOperations
```

```
Matrix Operations Menu:
1. Addition
2. Multiplication
3. Exit
Enter your choice:
```

Enter 1 for Add 2 for multiply and 3 for exit

Practical Slip_6 …

Q1) Write a program to display the Employee (Empid, Empname, Empdesignation, Empsal) information using toString().

```
class Employee {
    private int empId;
    private String empName;
    private String empDesignation;
    private double empSalary;

    // Constructor
    public Employee(int empId, String empName, String
empDesignation, double empSalary) {
        this.empId = empId;
        this.empName = empName;
        this.empDesignation = empDesignation;
        this.empSalary = empSalary;
    }

    // Overriding toString() method
    @Override
    public String toString() {
        return "Employee ID: " + empId + "\n" +
               "Employee Name: " + empName + "\n" +
               "Employee Designation: " + empDesignation + "\n" +
               "Employee Salary: " + empSalary;
    }

    // Main method to test the Employee class
    public static void main(String[] args) {
        // Creating an Employee object
```

```
        Employee emp = new Employee(101, "John Doe", "Software
Engineer", 75000.0);

        // Displaying Employee information using toString()
        System.out.println(emp);
    }
}
```

Compile and Run

```
javac Employee.java
```

```
java Employee
```

Q2) Create an abstract class "order" having members id, description. Create two subclasses "Purchase Order" and "Sales Order" having members customer name and Vendor name respectively. Definemethods accept and display in all cases. Create 3 objects each of Purchase Order and Sales Order and accept and display details.

```
import java.util.Scanner;

abstract class Order {
    protected int id;
    protected String description;

    // Abstract methods for accepting and displaying details
    abstract void accept(Scanner scanner);
    abstract void display();
}
```

```
class PurchaseOrder extends Order {
    private String vendorName;

    @Override
    void accept(Scanner scanner) {
        System.out.print("Enter Purchase Order ID: ");
        id = scanner.nextInt();
        scanner.nextLine();  // Consume newline left-over
        System.out.print("Enter Purchase Order Description: ");
        description = scanner.nextLine();
        System.out.print("Enter Vendor Name: ");
        vendorName = scanner.nextLine();
    }

    @Override
    void display() {
        System.out.println("\nPurchase Order Details:");
        System.out.println("ID: " + id);
        System.out.println("Description: " + description);
        System.out.println("Vendor Name: " + vendorName);
    }
}

class SalesOrder extends Order {
    private String customerName;

    @Override
    void accept(Scanner scanner) {
        System.out.print("Enter Sales Order ID: ");
        id = scanner.nextInt();
        scanner.nextLine();  // Consume newline left-over
```

```
        System.out.print("Enter Sales Order Description: ");
        description = scanner.nextLine();
        System.out.print("Enter Customer Name: ");
        customerName = scanner.nextLine();
    }

    @Override
    void display() {
        System.out.println("\nSales Order Details:");
        System.out.println("ID: " + id);
        System.out.println("Description: " + description);
        System.out.println("Customer Name: " + customerName);
    }
}

public class OrderTest {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Create and accept details for PurchaseOrder objects
        PurchaseOrder[] purchaseOrders = new PurchaseOrder[3];
        for (int i = 0; i < 3; i++) {
            purchaseOrders[i] = new PurchaseOrder();
            System.out.println("\nEnter details for Purchase Order " +
(i + 1) + ":");
            purchaseOrders[i].accept(scanner);
        }

        // Create and accept details for SalesOrder objects
        SalesOrder[] salesOrders = new SalesOrder[3];
        for (int i = 0; i < 3; i++) {
            salesOrders[i] = new SalesOrder();
```

```
            System.out.println("\nEnter details for Sales Order " + (i +
1) + ":");
            salesOrders[i].accept(scanner);
        }

        // Display PurchaseOrder details
        for (int i = 0; i < 3; i++) {
            purchaseOrders[i].display();
        }

        // Display SalesOrder details
        for (int i = 0; i < 3; i++) {
            salesOrders[i].display();
        }

        scanner.close();
    }
}
```

Compile and Run

```
javac OrderTest.java
```

```
java OrderTest
```

The program will prompt you to enter the details for three `PurchaseOrder` and three `SalesOrder` objects and then display the entered details for each order.

Sample

```
Enter details for Purchase Order 1:
Enter Purchase Order ID: 101
Enter Purchase Order Description: Office Supplies
Enter Vendor Name: ABC Supplies

Enter details for Purchase Order 2:
Enter Purchase Order ID: 102
Enter Purchase Order Description: Furniture
Enter Vendor Name: XYZ Furniture

Enter details for Purchase Order 3:
Enter Purchase Order ID: 103
Enter Purchase Order Description: IT Equipment
Enter Vendor Name: TechCorp
```

Sale

```
Enter details for Sales Order 1:
Enter Sales Order ID: 201
Enter Sales Order Description: Laptop Sale
Enter Customer Name: John Doe

Enter details for Sales Order 2:
Enter Sales Order ID: 202
Enter Sales Order Description: Printer Sale
Enter Customer Name: Jane Smith

Enter details for Sales Order 3:
Enter Sales Order ID: 203
Enter Sales Order Description: Monitor Sale
Enter Customer Name: Mike Johnson
```

OUTPUT

```
Purchase Order Details:
ID: 101
Description: Office Supplies
Vendor Name: ABC Supplies

Purchase Order Details:
ID: 102
Description: Furniture
Vendor Name: XYZ Furniture

Purchase Order Details:
ID: 103
Description: IT Equipment
Vendor Name: TechCorp

Sales Order Details:
ID: 201
Description: Laptop Sale
Customer Name: John Doe
```

Practical Slip_7 …

Q1) Design a class for Bank. Bank Class should support following operations;

a. Deposit a certain amount into an account

b. Withdraw a certain amount from an account

c. Return a Balance value specifying the amount with details.

```
import java.util.HashMap;
import java.util.Map;

public class Bank {

    // Inner class to represent a bank account
    private class Account {
        private double balance;

        public Account(double initialBalance) {
            this.balance = initialBalance;
        }

        public void deposit(double amount) {
            if (amount > 0) {
                this.balance += amount;
                System.out.println("Deposited: $" + amount);
            } else {
                System.out.println("Deposit amount must be positive.");
            }
        }

        public boolean withdraw(double amount) {
            if (amount > 0 && amount <= this.balance) {
                this.balance -= amount;
```

```
            System.out.println("Withdrew: $" + amount);
            return true;
        } else {
            System.out.println("Insufficient funds or invalid
amount.");
            return false;
        }
    }

    public double getBalance() {
        return this.balance;
    }
}

// Map to store accounts with account number as the key
private Map<String, Account> accounts = new HashMap<>();

// Method to create a new account
public void createAccount(String accountNumber, double
initialBalance) {
    if (!accounts.containsKey(accountNumber)) {
        accounts.put(accountNumber, new
Account(initialBalance));
        System.out.println("Account created with number: " +
accountNumber);
    } else {
        System.out.println("Account already exists.");
    }
}

// Method to deposit money into an account
public void deposit(String accountNumber, double amount) {
```

```
        Account account = accounts.get(accountNumber);
        if (account != null) {
            account.deposit(amount);
        } else {
            System.out.println("Account not found.");
        }
    }

    // Method to withdraw money from an account
    public void withdraw(String accountNumber, double amount) {
        Account account = accounts.get(accountNumber);
        if (account != null) {
            account.withdraw(amount);
        } else {
            System.out.println("Account not found.");
        }
    }

    // Method to get the balance of an account
    public void getBalance(String accountNumber) {
        Account account = accounts.get(accountNumber);
        if (account != null) {
            System.out.println("Account number: " + accountNumber
+ ", Balance: $" + account.getBalance());
        } else {
            System.out.println("Account not found.");
        }
    }
}

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

```
        Bank bank = new Bank();

        // Create accounts
        bank.createAccount("12345", 1000.00);
        bank.createAccount("67890", 500.00);

        // Deposit money
        bank.deposit("12345", 200.00);
        bank.deposit("67890", 50.00);

        // Withdraw money
        bank.withdraw("12345", 150.00);
        bank.withdraw("67890", 100.00);  // Should fail due to
insufficient funds

        // Get balance
        bank.getBalance("12345");
        bank.getBalance("67890");
    }
}
```

Compile and Run

```
javac Bank.java Main.java
```

```
java Main
```

Expected Output

```
Account created with number: 12345
Account created with number: 67890
Deposited: $200.0
Deposited: $50.0
Withdrew: $150.0
Insufficient funds or invalid amount.
Account number: 12345, Balance: $1050.0
Account number: 67890, Balance: $450.0
```

Q2) Write a program to accept a text file from user and display the contents of a file in reverse order and change its case.

Before writing the code create a .txt file and save the file in red hat root directory and provide the path of file at the time of compilation.

```
import java.io.*;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.Scanner;

public class FileProcessor {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Prompt user for the file path
        System.out.print("Enter the path of the text file: ");
        String filePath = scanner.nextLine();

        try {
            // Read the contents of the file into a String
            String content = new
String(Files.readAllBytes(Paths.get(filePath)));

            // Reverse the content and change its case
            String transformedContent =
reverseAndChangeCase(content);

            // Display the transformed content
```

```
            System.out.println("Transformed Content:");
            System.out.println(transformedContent);
        } catch (IOException e) {
            System.out.println("Error reading the file: " +
e.getMessage());
        }

        scanner.close();
    }

    private static String reverseAndChangeCase(String content) {
        StringBuilder reversedContent = new
StringBuilder(content).reverse();
        String result = changeCase(reversedContent.toString());
        return result;
    }

    private static String changeCase(String content) {
        StringBuilder changedCaseContent = new StringBuilder();
        for (char ch : content.toCharArray()) {
            if (Character.isUpperCase(ch)) {

changedCaseContent.append(Character.toLowerCase(ch));
            } else if (Character.isLowerCase(ch)) {

changedCaseContent.append(Character.toUpperCase(ch));
            } else {
                changedCaseContent.append(ch); // keep
non-alphabetic characters unchanged
            }
        }
        return changedCaseContent.toString();
```

```
    }
}
```

Compile and Run

```
javac FileProcessor.java
```

```
java FileProcessor
```

The program will prompt you to enter the path of the text file. After entering the path, it will display the transformed content.

Practical Slip_8 …

Q1)Create a class sphere, to calculate the volume and surface area of sphere.
(Hint: Surface area=4*3.14(r*r), Volume=(4/3)3.14(r*r*r))

```
public class Sphere {
    private double radius;

    // Constructor to initialize the sphere with a given radius
    public Sphere(double radius) {
        if (radius > 0) {
            this.radius = radius;
        } else {
            throw new IllegalArgumentException("Radius must be
positive.");
        }
    }

    // Method to calculate the surface area of the sphere
    public double getSurfaceArea() {
        return 4 * Math.PI * Math.pow(radius, 2);
    }

    // Method to calculate the volume of the sphere
    public double getVolume() {
        return (4.0 / 3.0) * Math.PI * Math.pow(radius, 3);
    }

    // Method to display the radius, surface area, and volume
    public void displayInfo() {
        System.out.println("Radius: " + radius);
```

```
        System.out.println("Surface Area: " + getSurfaceArea());
        System.out.println("Volume: " + getVolume());
    }

    // Main method to test the Sphere class
    public static void main(String[] args) {
        // Create a Sphere object with a specific radius
        Sphere sphere = new Sphere(5.0);

        // Display the information about the sphere
        sphere.displayInfo();
    }
}
```

Compile and Run

```
javac Sphere.java
```

```
java Sphere
```

Q2) Design a screen to handle the Mouse Events such as MOUSE_MOVED and MOUSE_CLICKED and display the position of the Mouse_Click in a TextField.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.MouseMotionListener;

public class MouseEventDemo extends JFrame {

    private JTextField textField;
```

```
public MouseEventDemo() {
    // Set up the frame
    setTitle("Mouse Event Demo");
    setSize(400, 300);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setLayout(new BorderLayout());

    // Create a JTextField to display mouse position
    textField = new JTextField();
    textField.setEditable(false);
    add(textField, BorderLayout.SOUTH);

    // Create a panel to capture mouse events
    JPanel panel = new JPanel();
    panel.setPreferredSize(new Dimension(400, 250));
    add(panel, BorderLayout.CENTER);

    // Add MouseListener to handle mouse clicks
    panel.addMouseListener(new MouseListener() {
        @Override
        public void mouseClicked(MouseEvent  e) {
            // Get mouse click position
            int x = e.getX();
            int y = e.getY();
            // Update the text field with mouse click position
  textField.setText("Mouse Clicked at: X=" + x + ", Y=" + y);
        }

        @Override
        public void mousePressed(MouseEvent  e) {
            // Not used
```

```
        }

        @Override
        public void mouseReleased(MouseEvent  e) {
            // Not used
        }

        @Override
        public void mouseEntered(MouseEvent  e) {
            // Not used
        }

        @Override
        public void mouseExited(MouseEvent  e) {
            // Not used
        }
    });

    // Add MouseMotionListener to handle mouse movements
(optional)
    panel.addMouseMotionListener(new MouseMotionListener()
{
        @Override
        public void mouseDragged(MouseEvent  e) {
            // Optional: Handle mouse dragging if needed
        }

        @Override
        public void mouseMoved(MouseEvent  e) {
            // Optional: Handle mouse movement if needed
            // e.g., update status bar with current mouse position
        }
```

```
        });
    }

    public static void main(String[] args) {
        // Create and show the GUI
        SwingUtilities.invokeLater(() -> {
            MouseEventDemo frame = new MouseEventDemo();
            frame.setVisible(true);
        });
    }
}
```

Compile and Run

```
javac MouseEventDemo.java
```

```
java MouseEventDemo
```

Practical Slip_9 …

Q1) Define a “Clock" class that does the following ;
a. Accept Hours, Minutes and Seconds
b. Check the validity of numbers
c. Set the time to AM/PM mode
Use the necessary constructors and methods to do the above task

```
public class Clock {
    private int hours;
    private int minutes;
    private int seconds;
    private boolean isPM;

    // Constructor to initialize the clock with hours, minutes, and
seconds
    public Clock(int hours, int minutes, int seconds) {
        setTime(hours, minutes, seconds);
    }

    // Method to set the time with validation
    public void setTime(int hours, int minutes, int seconds) {
        if (isValidTime(hours, minutes, seconds)) {
            this.hours = hours;
            this.minutes = minutes;
            this.seconds = seconds;
            this.isPM = hours >= 12; // Determine AM/PM
            if (this.hours > 12) {
                this.hours -= 12; // Convert to 12-hour format
            }
            if (this.hours == 0) {
                this.hours = 12; // Midnight or noon
            }
```

```
        } else {
            throw new IllegalArgumentException("Invalid time values.");
        }
    }

    // Method to validate time
    private boolean isValidTime(int hours, int minutes, int seconds) {
        return (hours >= 0 && hours < 24) && (minutes >= 0 && minutes <
60) && (seconds >= 0 && seconds < 60);
    }

    // Method to display time in 24-hour format
    public String getTime24HourFormat() {
        return String.format("%02d:%02d:%02d", hours, minutes,
seconds);
    }

    // Method to display time in 12-hour format with AM/PM
    public String getTime12HourFormat() {
        int displayHours = (hours == 0 || hours == 12) ? 12 : hours % 12;
        String period = isPM ? "PM" : "AM";
        return String.format("%02d:%02d:%02d %s", displayHours,
minutes, seconds, period);
    }

    // Method to display clock information
    public void displayTime() {
        System.out.println("24-Hour Format: " +
getTime24HourFormat());
        System.out.println("12-Hour Format: " +
getTime12HourFormat());
    }

    // Main method to test the Clock class
    public static void main(String[] args) {
```

```
        try {
            Clock clock = new Clock(14, 30, 45); // 2:30:45 PM
            clock.displayTime();
        } catch (IllegalArgumentException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

Compile and Run

```
javac Clock.java
```

```
java Clock
```

Q2) Write a program to using marker interface create a class Product (product_id, product_name, product_cost, product_quantity) default and parameterized constructor. Create objects of class product and display the contents of each object and Also display the object count.

```
// Marker interface
public interface ProductMarker {
    // This interface does not declare any methods or fields
}

public class Product implements ProductMarker {
    private int product_id;
    private String product_name;
    private double product_cost;
    private int product_quantity;
```

```
    private static int objectCount = 0; // Static variable to keep
track of the number of objects

    // Default constructor
    public Product() {
        this.product_id = 0;
        this.product_name = "Unknown";
        this.product_cost = 0.0;
        this.product_quantity = 0;
        objectCount++;
    }

    // Parameterized constructor
    public Product(int product_id, String product_name,
double product_cost, int product_quantity) {
        this.product_id = product_id;
        this.product_name = product_name;
        this.product_cost = product_cost;
        this.product_quantity = product_quantity;
        objectCount++;
    }

    // Method to display product details
    public void displayProductDetails() {
        System.out.println("Product ID: " + product_id);
        System.out.println("Product Name: " + product_name);
        System.out.println("Product Cost: $" + product_cost);
        System.out.println("Product Quantity: " +
product_quantity);
        System.out.println();
```

```
    }

    // Static method to get the count of Product objects
    public static int getObjectCount() {
        return objectCount;
    }

    // Main method to test the Product class
    public static void main(String[] args) {
        // Create objects using both constructors
        Product p1 = new Product(); // Default constructor
        Product p2 = new Product(101, "Laptop", 799.99, 5); //
Parameterized constructor
     Product p3 = new Product(102, "Smartphone", 499.99, 10);
// Parameterized constructor
// Display the details of each product
        p1.displayProductDetails();
        p2.displayProductDetails();
        p3.displayProductDetails();

        // Display the total count of Product objects
        System.out.println("Total number of Product objects
created: " + Product.getObjectCount());
    }
}
```

Compile and Run

```
javac ProductMarker.java Product.java
```

```
java Product
```

Slip 10

Q1) Write a program to find the cube of given number using functional interface.

```
interface CubeCalculator {
    // Abstract method to calculate the cube of a number
    double calculateCube(double number);
}

public class CubeUsingFunctionalInterface {

    public static void main(String[] args) {
        // Define a lambda expression for calculating the cube
        CubeCalculator cubeCalculator = (number) ->
Math.pow(number, 3);

        // Test the lambda expression with a given number
        double number = 5.0;
        double cube = cubeCalculator.calculateCube(number);

        // Display the result
        System.out.println("The cube of " + number + " is: " +
cube);
    }
}
```

Compile and Run

```
javac CubeUsingFunctionalInterface.java
```

```
java CubeUsingFunctionalInterface
```

Q2) Write a program to create a package name student. Define class StudentInfo with method to display information about student such as rollno, class, and percentage. Create another class Student Per with method to find percentage of the student. Accept student details like rollno, name, class and marks of 6 subject from user.

Step 1: Create the Package and Define the `StudentInfo` and `StudentPer` Classes

File: `StudentInfo.java`

```
package student;

public class StudentInfo {
    private int rollno;
    private String name;
    private String studentClass;

    // Constructor to initialize StudentInfo
    public StudentInfo(int rollno, String name, String studentClass) {
        this.rollno = rollno;
        this.name = name;
        this.studentClass = studentClass;
```

```
    }

    // Method to display student information
    public void displayInfo() {
        System.out.println("Roll Number: " + rollno);
        System.out.println("Name: " + name);
        System.out.println("Class: " + studentClass);
    }
}
```

File: `StudentPer.java`

```
package student;

public class StudentPer {
    private double[] marks;

    // Constructor to initialize StudentPer with marks
    public StudentPer(double[] marks) {
        this.marks = marks;
    }

    // Method to calculate percentage
    public double calculatePercentage() {
        double totalMarks = 0;
        for (double mark : marks) {
            totalMarks += mark;
        }
```

```
        return (totalMarks / (marks.length * 100)) * 100; // Assuming
each subject is out of 100
    }
}
```

Step 2: Create the Main Class

File: `Main.java`

```
import student.StudentInfo;

import student.StudentPer;


import java.util.Scanner;


public class Main {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);


        // Accept student details from user

        System.out.print("Enter Roll Number: ");

        int rollno = scanner.nextInt();

        scanner.nextLine();  // Consume newline
```

```
        System.out.print("Enter Name: ");
        String name = scanner.nextLine();

        System.out.print("Enter Class: ");
        String studentClass = scanner.nextLine();

        double[] marks = new double[6];
        System.out.println("Enter marks for 6 subjects:");
        for (int i = 0; i < marks.length; i++) {
            System.out.print("Subject " + (i + 1) + ": ");
            marks[i] = scanner.nextDouble();
        }

        // Create StudentInfo and StudentPer objects
        StudentInfo studentInfo = new StudentInfo(rollno, name,
studentClass);
        StudentPer studentPer = new StudentPer(marks);
```

```
        // Display student information
        studentInfo.displayInfo();
// Calculate and display percentage
        double percentage = studentPer.calculatePercentage();
        System.out.println("Percentage: " + percentage + "%");
        scanner.close();
    }
}
```

Compile and Run

```
javac student/StudentInfo.java student/StudentPer.java Main.java
```

```
java Main
```