SLIP 21

Q.1) Define a class MyDate (day, month, year,) with methods to accept and display a MyDate object. Accept date as dd,mm,yyyy. Throw user define exception "InvalidDateException" if the date is invalid.

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

class MyDate {
    private int day, month, year;

    public MyDate(int day, int month, int year) throws InvalidDateException {
        if (!isValidDate(day, month, year)) {
            throw new InvalidDateException("Invalid date: " + day + "/" + month + "/"
+ year);
        }
        this.day = day;
        this.month = month;
        this.year = year;
    }

    private boolean isValidDate(int day, int month, int year) {
        if (year < 1 || month < 1 || month > 12) return false;
        int[] daysInMonth = {31, (isLeapYear(year) ? 29 : 28), 31, 30, 31, 30, 31, 31,
30, 31, 30, 31};
        return day > 0 && day <= daysInMonth[month - 1];
    }

    private boolean isLeapYear(int year) {
        return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
    }
```

```
    public void displayDate() {
        System.out.println("Date: " + day + "/" + month + "/" + year);
    }
}

public class Main {
    public static void main(String[] args) {
        try {
            MyDate date = new MyDate(29, 2, 2024);  // Valid leap year date
            date.displayDate();

            MyDate invalidDate = new MyDate(31, 11, 2023);  // Invalid date
            invalidDate.displayDate();
        } catch (InvalidDateException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

Expected Output

date: 29/2/2024 Invalid date: 31/11/2023

Q.2)Create an employee class (id,name,deptname,salary). Define a default and parameterized constructor. Use 'this' keyword to initialize instance variables. Keep a count of objects created. Create objects using parameterized constructor and display the object count after each object is created (Use static member and method). Also display the contents of each object.

```
class Employee {
    private int id;
    private String name;
    private String deptName;
    private double salary;
```

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

    // Default constructor
    public Employee() {
        this(0, "Unknown", "Unknown", 0.0); // Calls parameterized constructor with
default values
    }

    // Parameterized constructor
    public Employee(int id, String name, String deptName, double salary) {
        this.id = id;
        this.name = name;
        this.deptName = deptName;
        this.salary = salary;
        objectCount++;  // Increment object count
    }

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

    // Method to display object details
    public void displayEmployee() {
        System.out.println("Employee ID: " + this.id);
        System.out.println("Employee Name: " + this.name);
        System.out.println("Department: " + this.deptName);
        System.out.println("Salary: " + this.salary);
        System.out.println("-------------------");
    }
}

public class Main {
    public static void main(String[] args) {
        // Creating objects using parameterized constructor
```

```
        Employee emp1 = new Employee(101, "Alice", "IT", 50000);
        emp1.displayEmployee();
        System.out.println("Total Objects: " + Employee.getObjectCount());

        Employee emp2 = new Employee(102, "Bob", "HR", 45000);
        emp2.displayEmployee();
        System.out.println("Total Objects: " + Employee.getObjectCount());

        Employee emp3 = new Employee(103, "Charlie", "Finance", 55000);
        emp3.displayEmployee();
        System.out.println("Total Objects: " + Employee.getObjectCount());
    }
}
```

Expected Output

```
Employee ID: 101
Employee Name: Alice
Department: IT
Salary: 50000.0
------------------
Total Objects: 1

Employee ID: 102
Employee Name: Bob
Department: HR
Salary: 45000.0
------------------
Total Objects: 2

Employee ID: 103
Employee Name: Charlie
Department: Finance
Salary: 55000.0
------------------
Total Objects: 3
```

Slip 22

Q.1) Write a program to create an abstract class named Shape that contains two integers and an empty method named PrintArea() Provide three classes named Rectangle, Triangle, and Circle such that each one of the classes extends the class Shape. each one of the classes contains only the method PrintArea() that prints the area of the given shape. (Use method Overriding).

```
// Abstract class Shape
abstract class Shape {
    protected int dimension1;
    protected int dimension2;

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

// Rectangle class that extends Shape
class Rectangle extends Shape {

    // Constructor to initialize dimensions
    public Rectangle(int length, int breadth) {
        this.dimension1 = length;
        this.dimension2 = breadth;
    }

    // Overriding PrintArea() method to calculate and print the area of a rectangle
    @Override
    void PrintArea() {
        int area = dimension1 * dimension2;
        System.out.println("Area of Rectangle: " + area);
    }
}

// Triangle class that extends Shape
```

```
class Triangle extends Shape {

    // Constructor to initialize dimensions
    public Triangle(int base, int height) {
        this.dimension1 = base;
        this.dimension2 = height;
    }

    // Overriding PrintArea() method to calculate and print the area of a triangle
    @Override
    void PrintArea() {
        double area = 0.5 * dimension1 * dimension2;
        System.out.println("Area of Triangle: " + area);
    }
}

// Circle class that extends Shape
class Circle extends Shape {

    // Constructor to initialize radius (dimension2 not used for Circle)
    public Circle(int radius) {
        this.dimension1 = radius;
        this.dimension2 = 0;  // Not needed for Circle
    }

    // Overriding PrintArea() method to calculate and print the area of a circle
    @Override
    void PrintArea() {
        double area = Math.PI * dimension1 * dimension1;
        System.out.println("Area of Circle: " + area);
    }
}

// Main class to test the shapes
public class Main {
    public static void main(String[] args) {
        // Create instances of different shapes and print their areas
```

```
        Shape rectangle = new Rectangle(5, 10);
        rectangle.PrintArea();

        Shape triangle = new Triangle(4, 6);
        triangle.PrintArea();

        Shape circle = new Circle(7);
        circle.PrintArea();
    }
}
```

Expected Output

```
Area of Rectangle: 50
Area of Triangle: 12.0
Area of Circle: 153.93804002589985
```

Q.2) Write a program that handles all mouse events and shows the event name at the center of the window, red in color, when a mouse event is fired. (Use adapter classes).

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

public class MouseEventDemo extends JFrame {
    private String eventName = "";  // Variable to store the name of the event

    public MouseEventDemo() {
        // Set up the JFrame
        setTitle("Mouse Event Demo");
        setSize(400, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);

        // Add a custom mouse adapter to handle all mouse events
```

```
    addMouseListener(new MyMouseAdapter());
    addMouseMotionListener(new MyMouseAdapter());
}

// Overriding the paint method to display the event name in red at the center
public void paint(Graphics g) {
    super.paint(g);
    g.setColor(Color.RED);  // Set text color to red
    g.setFont(new Font("Arial", Font.BOLD, 20));  // Set font style
    FontMetrics fm = g.getFontMetrics();
    int x = (getWidth() - fm.stringWidth(eventName)) / 2;
    int y = (getHeight() / 2);
    g.drawString(eventName, x, y);  // Draw the event name at the center
}

// Custom adapter class to handle mouse events
class MyMouseAdapter extends MouseAdapter {
    // Mouse clicked event
    public void mouseClicked(MouseEvent e) {
        eventName = "Mouse Clicked";
        repaint();  // Repaint the window
    }

    // Mouse pressed event
    public void mousePressed(MouseEvent e) {
        eventName = "Mouse Pressed";
        repaint();
    }

    // Mouse released event
    public void mouseReleased(MouseEvent e) {
        eventName = "Mouse Released";
        repaint();
    }

    // Mouse entered event
    public void mouseEntered(MouseEvent e) {
```

```
            eventName = "Mouse Entered";
            repaint();
        }

        // Mouse exited event
        public void mouseExited(MouseEvent e) {
            eventName = "Mouse Exited";
            repaint();
        }

        // Mouse dragged event
        public void mouseDragged(MouseEvent e) {
            eventName = "Mouse Dragged";
            repaint();
        }

        // Mouse moved event
        public void mouseMoved(MouseEvent e) {
            eventName = "Mouse Moved";
            repaint();
        }
    }

    public static void main(String[] args) {
        new MouseEventDemo();  // Create and display the window
    }
}
```

Expected Output

When you interact with the window by clicking, dragging, or moving the mouse, the event name (e.g., "Mouse Clicked", "Mouse Moved") will be displayed in red in the center of the window.

Slip23

Q.1) Define a class MyNumber having one private int data member. Write a default constructor to initialize it to 0 and another constructor to initialize it to a

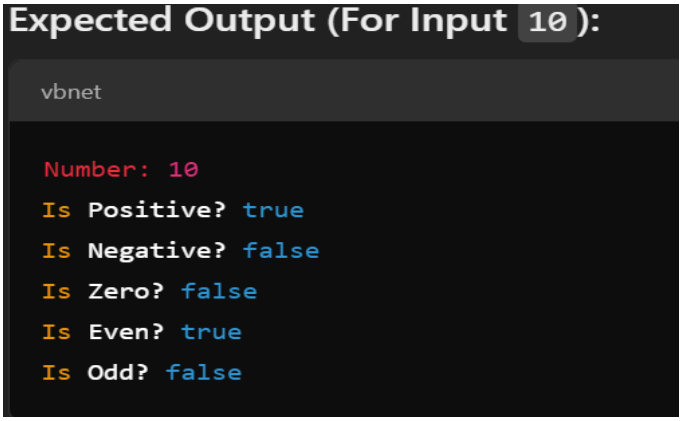
value. (use this). Write methods isNegative, isPositive, isZero, isOdd, isEven. Create an object in main. Use command line arguments to pass a value to the object.

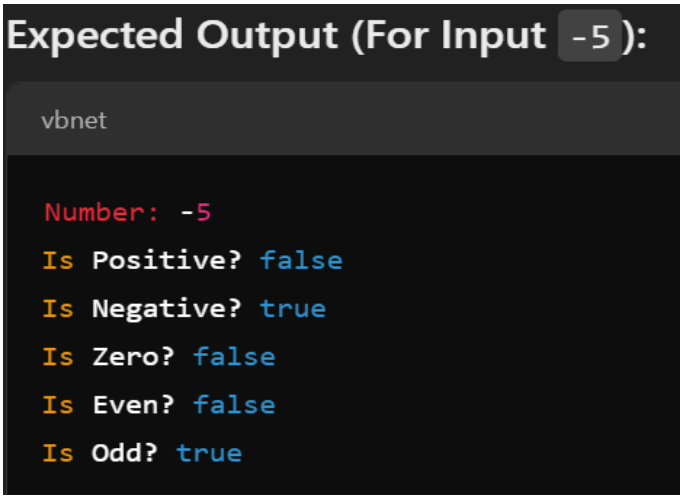
```
class MyNumber {
    // Private data member to store the number
    private int number;

    // Default constructor initializing the number to 0
    public MyNumber() {
        this.number = 0;
    }

    // Parameterized constructor to initialize the number to a given value
    public MyNumber(int number) {
        this.number = number;
    }

    // Method to check if the number is negative
    public boolean isNegative() {
        return this.number < 0;
    }

    // Method to check if the number is positive
    public boolean isPositive() {
        return this.number > 0;
    }

    // Method to check if the number is zero
    public boolean isZero() {
        return this.number == 0;
    }

    // Method to check if the number is odd
    public boolean isOdd() {
        return this.number % 2 != 0;
```

```
    }

    // Method to check if the number is even
    public boolean isEven() {
        return this.number % 2 == 0;
    }

    // Method to display the number
    public void displayNumber() {
        System.out.println("Number: " + this.number);
    }

    public static void main(String[] args) {
        // Check if the command line argument is passed
        if (args.length > 0) {
            // Parse the argument and create an object with the parameterized
constructor
            int value = Integer.parseInt(args[0]);
            MyNumber myNumber = new MyNumber(value);

            // Display the number and its properties
            myNumber.displayNumber();
            System.out.println("Is Positive? " + myNumber.isPositive());
            System.out.println("Is Negative? " + myNumber.isNegative());
            System.out.println("Is Zero? " + myNumber.isZero());
            System.out.println("Is Even? " + myNumber.isEven());
            System.out.println("Is Odd? " + myNumber.isOdd());
        } else {
            // If no argument is passed, use the default constructor
            MyNumber myNumber = new MyNumber();

            // Display the number and its properties
            myNumber.displayNumber();
            System.out.println("Is Positive? " + myNumber.isPositive());
            System.out.println("Is Negative? " + myNumber.isNegative());
            System.out.println("Is Zero? " + myNumber.isZero());
            System.out.println("Is Even? " + myNumber.isEven());
```

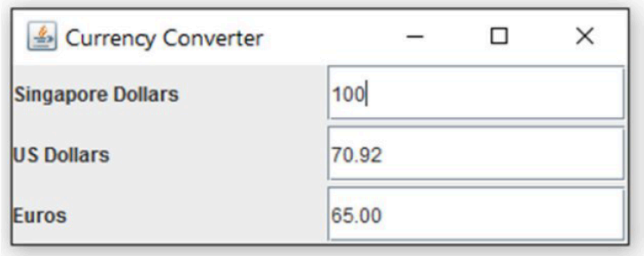
```
            System.out.println("Is Odd? " + myNumber.isOdd());
        }
    }
}
```

Expected Output (For Input 10):





Q.2) Write a simple currency converter as shown in the figure. User can enter the amount of "Singapore dollars", "US dollars", "Euros", in floating point number. The converted values shall be displayed to 2 decimal places. Assume that 1 USD = 1.41 SGD, 1 USD = 0.92 Euro, 1 SGD = 0.65 Euro.




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

public class CurrencyConverter extends JFrame implements KeyListener {
    // Declare the text fields for input and output
    JTextField sgDollarField, usDollarField, euroField;

    // Conversion rates
    final double USD_TO_SGD = 1.41;
    final double USD_TO_EURO = 0.92;
    final double SGD_TO_EURO = 0.65;

    public CurrencyConverter() {
        // Set up the frame
        setTitle("Currency Converter");
        setSize(300, 200);
        setLayout(new GridLayout(3, 2));

        // Create and add labels and text fields to the frame
        JLabel sgDollarLabel = new JLabel("Singapore Dollars");
        sgDollarField = new JTextField();
        sgDollarField.addKeyListener(this);

        JLabel usDollarLabel = new JLabel("US Dollars");
        usDollarField = new JTextField();
        usDollarField.setEditable(false);

        JLabel euroLabel = new JLabel("Euros");
        euroField = new JTextField();
```

```
    euroField.setEditable(false);

    // Add the components to the frame
    add(sgDollarLabel);
    add(sgDollarField);
    add(usDollarLabel);
    add(usDollarField);
    add(euroLabel);
    add(euroField);

    // Final frame settings
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setVisible(true);
}

// Implement KeyListener methods
@Override
public void keyTyped(KeyEvent e) {}

@Override
public void keyPressed(KeyEvent e) {}

@Override
public void keyReleased(KeyEvent e) {
    try {
        // Get the value from the Singapore Dollar field
        double sgd = Double.parseDouble(sgDollarField.getText());

        // Convert SGD to USD and Euros
        double usd = sgd / USD_TO_SGD;
        double euro = sgd * SGD_TO_EURO;

        // Set the values in US Dollar and Euro fields
        usDollarField.setText(String.format("%.2f", usd));
        euroField.setText(String.format("%.2f", euro));
    } catch (NumberFormatException ex) {
        // Clear the fields if input is not a valid number
```

```
            usDollarField.setText("");
            euroField.setText("");
        }
    }

    public static void main(String[] args) {
        // Run the application
        new CurrencyConverter();
    }
}
```

Slip 24

Q.1) Create an abstract class 'Bank' with an abstract method 'getBalance'. Rs.100, Rs.150, and Rs.200 are deposited in banks A, B, and C respectively. Bank A, Bank B, and Bank C are subclasses of class 'Bank' each having a method named 'getBalance'. Call this method by creating an object of each of the three classes.

```
// Abstract class Bank with an abstract method getBalance
abstract class Bank {
    // Abstract method to be implemented by subclasses
    public abstract int getBalance();
}

// BankA subclass with Rs. 100 deposited
class BankA extends Bank {
    private int balance = 100;

    @Override
    public int getBalance() {
        return balance;
    }
}

// BankB subclass with Rs. 150 deposited
class BankB extends Bank {
```

```
    private int balance = 150;

    @Override
    public int getBalance() {
        return balance;
    }
}

// BankC subclass with Rs. 200 deposited
class BankC extends Bank {
    private int balance = 200;

    @Override
    public int getBalance() {
        return balance;
    }
}

// Main class to demonstrate the functionality
public class Main {
    public static void main(String[] args) {
        // Create objects of each bank class
        Bank bankA = new BankA();
        Bank bankB = new BankB();
        Bank bankC = new BankC();

        // Call getBalance method for each bank and display the result
        System.out.println("Balance in Bank A: Rs." + bankA.getBalance());
        System.out.println("Balance in Bank B: Rs." + bankB.getBalance());
        System.out.println("Balance in Bank C: Rs." + bankC.getBalance());
    }
}
```

Expected Output:

Balance in Bank A: Rs.100
Balance in Bank B: Rs.150

Balance in Bank C: Rs.200

Q.2) Program that displays three concentric circles where ever the user clicks the mouse on a frame. The program must exit when user clicks 'X' on the frame.

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

public class ConcentricCircles extends JFrame implements MouseListener {
    private int x = -1, y = -1; // Initial position for the circles

    public ConcentricCircles() {
        // Set up the frame
        setTitle("Concentric Circles");
        setSize(400, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        addMouseListener(this);
        setVisible(true);
    }

    // Override paint method to draw the circles
    @Override
    public void paint(Graphics g) {
        super.paint(g);

        // Draw the concentric circles if a valid click has occurred
        if (x != -1 && y != -1) {
            // Draw three circles with increasing radii
```

```
            g.setColor(Color.DARK_GRAY);
            g.drawOval(x - 75, y - 75, 150, 150); // Outer circle
            g.drawOval(x - 50, y - 50, 100, 100); // Middle circle
            g.drawOval(x - 25, y - 25, 50, 50);   // Inner circle
        }
    }

    // MouseListener methods
    @Override
    public void mouseClicked(MouseEvent e) {
        // Update the coordinates based on mouse click position
        x = e.getX();
        y = e.getY();

        // Repaint the frame to draw circles at new position
        repaint();
    }

    @Override
    public void mousePressed(MouseEvent e) {}
    @Override
    public void mouseReleased(MouseEvent e) {}
    @Override
    public void mouseEntered(MouseEvent e) {}
    @Override
    public void mouseExited(MouseEvent e) {}

    // Main method to run the program
    public static void main(String[] args) {
        new ConcentricCircles();
    }
}
```

Expected Output:

Wherever you click on the frame, three concentric circles will appear, as shown in the image you provided. The program will exit when the 'X' button is clicked.

Slip 25

Q.1) Create a class Student (rollno, name, class, per) to read student information from the console and display them (using BufferedReader class).

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

class Student {
    // Declare fields
    private int rollno;
    private String name;
    private String className;
    private float per;

    // Constructor to initialize fields
    public Student(int rollno, String name, String className, float per) {
        this.rollno = rollno;
        this.name = name;
        this.className = className;
        this.per = per;
    }

    // Method to display student information
    public void display() {
        System.out.println("Student Information:");
        System.out.println("Roll No: " + rollno);
        System.out.println("Name: " + name);
        System.out.println("Class: " + className);
        System.out.println("Percentage: " + per + "%");
    }
}

public class StudentInfo {
    public static void main(String[] args) {
        // Using BufferedReader to read input
```

```
        BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));

        try {
            // Read student details from the console
            System.out.println("Enter Roll Number: ");
            int rollno = Integer.parseInt(reader.readLine());

            System.out.println("Enter Name: ");
            String name = reader.readLine();

            System.out.println("Enter Class: ");
            String className = reader.readLine();

            System.out.println("Enter Percentage: ");
            float per = Float.parseFloat(reader.readLine());

            // Create a Student object with the provided data
            Student student = new Student(rollno, name, className, per);

            // Display the student information
            student.display();

        } catch (IOException e) {
            System.out.println("An error occurred while reading input.");
            e.printStackTrace();
        }
    }
}
```

Expected Output

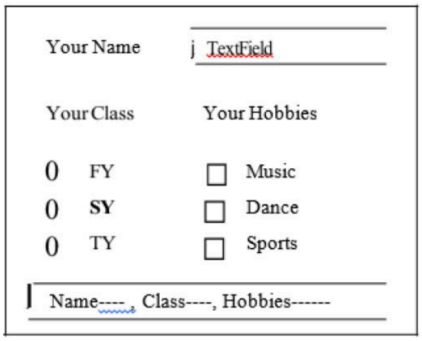
When you run the program, it will prompt you to enter the student's information, and then it will display that information back to you. Here's an example

```
Enter Roll Number:
101
```

Enter Name:
John Doe
Enter Class:
10th Grade
Enter Percentage:
85.5

Student Information:
Roll No: 101
Name: John Doe
Class: 10th Grade
Percentage: 85.5%

Q.2) Create the following GUI screen using appropriate layout manager. Accept the name, class, hobbies from the user and display the selected options in a text box.




```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class StudentInfoGUI extends JFrame implements ActionListener {

    JTextField nameField;
    JRadioButton fy, sy, ty;
```

```
JCheckBox music, dance, sports;
JTextArea displayArea;

public StudentInfoGUI() {
    setTitle("Student Information");
    setSize(400, 300);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setLayout(new BorderLayout());

    // Panel for Name and Class
    JPanel panel1 = new JPanel(new GridLayout(3, 2));

    JLabel nameLabel = new JLabel("Your Name:");
    nameField = new JTextField(20);

    panel1.add(nameLabel);
    panel1.add(nameField);

    // Panel for Class
    JLabel classLabel = new JLabel("Your Class:");
    panel1.add(classLabel);

    JPanel classPanel = new JPanel(new FlowLayout());
    fy = new JRadioButton("FY");
    sy = new JRadioButton("SY");
    ty = new JRadioButton("TY");

    ButtonGroup classGroup = new ButtonGroup();
    classGroup.add(fy);
    classGroup.add(sy);
    classGroup.add(ty);

    classPanel.add(fy);
    classPanel.add(sy);
    classPanel.add(ty);

    panel1.add(classPanel);
```

```
    // Panel for Hobbies
    JLabel hobbiesLabel = new JLabel("Your Hobbies:");
    panel1.add(hobbiesLabel);

    JPanel hobbiesPanel = new JPanel(new FlowLayout());
    music = new JCheckBox("Music");
    dance = new JCheckBox("Dance");
    sports = new JCheckBox("Sports");

    hobbiesPanel.add(music);
    hobbiesPanel.add(dance);
    hobbiesPanel.add(sports);

    panel1.add(hobbiesPanel);

    add(panel1, BorderLayout.NORTH);

    // Text Area to Display Output
    displayArea = new JTextArea();
    displayArea.setEditable(false);
    add(new JScrollPane(displayArea), BorderLayout.CENTER);

    // Submit Button
    JButton submitButton = new JButton("Submit");
    submitButton.addActionListener(this);
    add(submitButton, BorderLayout.SOUTH);
}

@Override
public void actionPerformed(ActionEvent e) {
    String name = nameField.getText();
    String selectedClass = "";

    if (fy.isSelected()) {
        selectedClass = "FY";
    } else if (sy.isSelected()) {
```

```
            selectedClass = "SY";
        } else if (ty.isSelected()) {
            selectedClass = "TY";
        }

        StringBuilder hobbies = new StringBuilder();
        if (music.isSelected()) {
            hobbies.append("Music ");
        }
        if (dance.isSelected()) {
            hobbies.append("Dance ");
        }
        if (sports.isSelected()) {
            hobbies.append("Sports ");
        }

        String result = "Name: " + name + ", Class: " + selectedClass + ", Hobbies: "
+ hobbies.toString();
        displayArea.setText(result);
    }

    public static void main(String[] args) {
        StudentInfoGUI frame = new StudentInfoGUI();
        frame.setVisible(true);
    }
}
```

Expected Output

When the user fills in their name, selects a class and hobbies, and presses "Submit", the selected values are displayed in the `JTextArea` at the bottom.

Slip 26

Q.1) Define a item class (item_number, item_name, item_price). Define a default and parameterized constructor. Keep a count of objects created. Create objects using parameterized constructor and display the object count after each object is created.(Use static member and method).Also display the contents of each object.

```
class Item {
    // Instance variables
    private int item_number;
    private String item_name;
    private double item_price;

    // Static variable to count the number of objects created
    private static int objectCount = 0;

    // Default constructor
    public Item() {
        this.item_number = 0;
        this.item_name = "Unknown";
        this.item_price = 0.0;
        incrementCount();
    }

    // Parameterized constructor
    public Item(int item_number, String item_name, double item_price) {
        this.item_number = item_number;
        this.item_name = item_name;
        this.item_price = item_price;
        incrementCount();
    }

    // Static method to increment object count
    private static void incrementCount() {
```

```
        objectCount++;
    }

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

    // Method to display the contents of the object
    public void displayItem() {
        System.out.println("Item Number: " + item_number);
        System.out.println("Item Name: " + item_name);
        System.out.println("Item Price: " + item_price);
        System.out.println("-----------------------");
    }

    public static void main(String[] args) {
        // Create the first object using the parameterized constructor
        Item item1 = new Item(101, "Laptop", 750.50);
        System.out.println("Object count after creating item1: " +
Item.getObjectCount());
        item1.displayItem();

        // Create the second object using the parameterized constructor
        Item item2 = new Item(102, "Smartphone", 500.99);
        System.out.println("Object count after creating item2: " +
Item.getObjectCount());
        item2.displayItem();

        // Create the third object using the parameterized constructor
        Item item3 = new Item(103, "Tablet", 300.75);
        System.out.println("Object count after creating item3: " +
Item.getObjectCount());
        item3.displayItem();
    }
}
```

Expected Output

```
Object count after creating item1: 1
Item Number: 101
Item Name: Laptop
Item Price: 750.5
------------------------
Object count after creating item2: 2
Item Number: 102
Item Name: Smartphone
Item Price: 500.99
------------------------
Object count after creating item3: 3
Item Number: 103
Item Name: Tablet
Item Price: 300.75
------------------------
```

Q.2) Define a class 'Donor' to store the below mentioned details of a blood donor. Name, age, address, contact number, blood group, date of last donation. Create 'n' objects of this class for all the regular donors at Pune Write these objects to a file, read these objects from the file and display only those donor's details whose blood group is 'A+ve' and had not donated for the recent 6 months.

```
import java.io.*;
import java.util.*;
import java.time.*;
import java.time.format.DateTimeFormatter;

class Donor implements Serializable {
    // Donor attributes
    private String name;
    private int age;
    private String address;
    private String contactNumber;
    private String bloodGroup;
    private String lastDonationDate;
```

```
    // Constructor to initialize donor details
    public Donor(String name, int age, String address, String contactNumber,
String bloodGroup, String lastDonationDate) {
        this.name = name;
        this.age = age;
        this.address = address;
        this.contactNumber = contactNumber;
        this.bloodGroup = bloodGroup;
        this.lastDonationDate = lastDonationDate;
    }

    // Getter methods for blood group and last donation date
    public String getBloodGroup() {
        return bloodGroup;
    }

    public String getLastDonationDate() {
        return lastDonationDate;
    }

    // Method to display donor details
    public void display() {
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
        System.out.println("Address: " + address);
        System.out.println("Contact Number: " + contactNumber);
        System.out.println("Blood Group: " + bloodGroup);
        System.out.println("Last Donation Date: " + lastDonationDate);
        System.out.println("-----------------------------------");
    }

    // Check if the donor hasn't donated in the last 6 months
    public boolean hasNotDonatedForSixMonths() {
        LocalDate lastDonation = LocalDate.parse(lastDonationDate,
DateTimeFormatter.ofPattern("dd-MM-yyyy"));
        LocalDate sixMonthsAgo = LocalDate.now().minusMonths(6);
```

```
        return lastDonation.isBefore(sixMonthsAgo);
    }

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

        try {
            // Writing Donor objects to file
            System.out.println("Enter the number of donors: ");
            n = sc.nextInt();
            sc.nextLine(); // consume the newline character

            List<Donor> donors = new ArrayList<>();
            for (int i = 0; i < n; i++) {
                System.out.println("Enter details of donor " + (i + 1) + ": ");
                System.out.print("Name: ");
                String name = sc.nextLine();
                System.out.print("Age: ");
                int age = sc.nextInt();
                sc.nextLine(); // consume the newline character
                System.out.print("Address: ");
                String address = sc.nextLine();
                System.out.print("Contact Number: ");
                String contactNumber = sc.nextLine();
                System.out.print("Blood Group: ");
                String bloodGroup = sc.nextLine();
                System.out.print("Last Donation Date (dd-MM-yyyy): ");
                String lastDonationDate = sc.nextLine();

                donors.add(new Donor(name, age, address, contactNumber,
bloodGroup, lastDonationDate));
            }

            // Write donor objects to a file
            FileOutputStream fos = new FileOutputStream("donors.dat");
            ObjectOutputStream oos = new ObjectOutputStream(fos);
```

```
            oos.writeObject(donors);
            oos.close();
            fos.close();

            // Reading Donor objects from file
            FileInputStream fis = new FileInputStream("donors.dat");
            ObjectInputStream ois = new ObjectInputStream(fis);
            List<Donor> savedDonors = (List<Donor>) ois.readObject();
            ois.close();
            fis.close();

            // Display donors whose blood group is A+ve and haven't donated in the
last 6 months
            System.out.println("Donors with blood group 'A+ve' who haven't donated
in the last 6 months:");
            for (Donor donor : savedDonors) {
                if (donor.getBloodGroup().equals("A+ve") &&
donor.hasNotDonatedForSixMonths()) {
                    donor.display();
                }
            }

        } catch (Exception e) {
            e.printStackTrace();
        }

        sc.close();
    }
}
```

Input Example

```
Enter the number of donors: 2

Enter details of donor 1:
Name: John Doe
Age: 30
Address: Pune, India
```

Contact Number: 1234567890
Blood Group: A+ve
Last Donation Date (dd-MM-yyyy): 01-01-2023

Enter details of donor 2:
Name: Jane Smith
Age: 28
Address: Pune, India
Contact Number: 0987654321
Blood Group: O+ve
Last Donation Date (dd-MM-yyyy): 01-07-2023

Output Example

Donors with blood group 'A+ve' who haven't donated in the last 6 months:
Name: John Doe
Age: 30
Address: Pune, India
Contact Number: 1234567890
Blood Group: A+ve
Last Donation Date: 01-01-2023

Slip 27

Q.1) Define an employee class with suitable attributes having GetSalary() method, which returns salary withdrawn by a particular employee. Write a class manager which extends a class employee, overwrite the GetSalary() method, which will return salary of manager by adding Travelling Allowance, House Rent Allowance etc.

```
// Base class Employee
class Employee {
```

```
    protected String name;
    protected int id;
    protected double baseSalary;

    // Constructor to initialize employee details
    public Employee(String name, int id, double baseSalary) {
        this.name = name;
        this.id = id;
        this.baseSalary = baseSalary;
    }

    // Method to return base salary of employee
    public double GetSalary() {
        return baseSalary;
    }

    // Method to display employee details
    public void displayDetails() {
        System.out.println("Employee Name: " + name);
        System.out.println("Employee ID: " + id);
        System.out.println("Base Salary: " + baseSalary);
    }
}

// Manager class extending Employee class
class Manager extends Employee {
    private double travelAllowance;
    private double houseRentAllowance;

    // Constructor to initialize manager details
    public Manager(String name, int id, double baseSalary, double
travelAllowance, double houseRentAllowance) {
        super(name, id, baseSalary); // Calling Employee class constructor
        this.travelAllowance = travelAllowance;
        this.houseRentAllowance = houseRentAllowance;
    }
```

```
    // Overriding GetSalary method to include allowances
    @Override
    public double GetSalary() {
        return baseSalary + travelAllowance + houseRentAllowance;
    }

    // Method to display manager details
    @Override
    public void displayDetails() {
        super.displayDetails();
        System.out.println("Travel Allowance: " + travelAllowance);
        System.out.println("House Rent Allowance: " + houseRentAllowance);
        System.out.println("Total Salary: " + GetSalary());
    }
}

// Main class to test the Employee and Manager classes
public class Main {
    public static void main(String[] args) {
        // Creating an Employee object
        Employee emp = new Employee("John Doe", 101, 30000);
        System.out.println("Employee Details:");
        emp.displayDetails();
        System.out.println("Employee Salary: " + emp.GetSalary());
        System.out.println("--------------------------");

        // Creating a Manager object
        Manager mgr = new Manager("Jane Smith", 102, 50000, 10000, 15000);
        System.out.println("Manager Details:");
        mgr.displayDetails();
        System.out.println("Manager Salary: " + mgr.GetSalary());
    }
}
```

Output Example

```
Employee Details:
Employee Name: John Doe
```

```
Employee ID: 101
Base Salary: 30000.0
Employee Salary: 30000.0
--------------------------
Manager Details:
Employee Name: Jane Smith
Employee ID: 102
Base Salary: 50000.0
Travel Allowance: 10000.0
House Rent Allowance: 15000.0
Total Salary: 75000.0
Manager Salary: 75000.0
```

Q.2) Write a program to accept a string as command line argument and check whether it is a file or directory. Also perform operations as follows:
i) If it is a directory, delete all text files in that directory. Confirm delete operation from user before deleting text files. Also, display a count showing the number of files deleted, if any, from the directory.
ii) If it is a file display various details of that file..

```
import java.io.*;
import java.util.Scanner;

public class FileDirectoryChecker {
    public static void main(String[] args) {
        // Check if argument is passed
        if (args.length != 1) {
            System.out.println("Please provide a valid file or directory path as a
command-line argument.");
            return;
        }

        // Create a File object from the command-line argument
```

```
        File fileOrDir = new File(args[0]);

        // Check if it's a directory
        if (fileOrDir.isDirectory()) {
            System.out.println(args[0] + " is a directory.");

            // List all text files in the directory
            File[] textFiles = fileOrDir.listFiles((dir, name) ->
name.toLowerCase().endsWith(".txt"));

            if (textFiles != null && textFiles.length > 0) {
                System.out.println("Text files in the directory:");

                // Show all the text files and ask for confirmation to delete
                for (File textFile : textFiles) {
                    System.out.println(textFile.getName());
                }

                Scanner scanner = new Scanner(System.in);
                System.out.print("Do you want to delete all the text files? (yes/no): ");
                String confirmation = scanner.nextLine();

                // If user confirms deletion
                if (confirmation.equalsIgnoreCase("yes")) {
                    int count = 0;
                    for (File textFile : textFiles) {
                        if (textFile.delete()) {
                            count++;
                        }
                    }
                    System.out.println(count + " text file(s) deleted from the directory.");
                } else {
                    System.out.println("No files were deleted.");
                }

            } else {
                System.out.println("No text files found in the directory.");
```

```
            }

        // Check if it's a file
        } else if (fileOrDir.isFile()) {
            System.out.println(args[0] + " is a file.");
            displayFileDetails(fileOrDir);

        // If neither file nor directory exists
        } else {
            System.out.println("The provided path does not exist.");
        }
    }

    // Method to display file details
    public static void displayFileDetails(File file) {
        System.out.println("File Details:");
        System.out.println("File Name: "    + file.getName());
        System.out.println("Absolute Path: " + file.getAbsolutePath());
        System.out.println("File Size: "    + file.length() + " bytes");
        System.out.println("Readable: "     + file.canRead());
        System.out.println("Writable: "     + file.canWrite());
        System.out.println("Executable: "   + file.canExecute());
        System.out.println("Last Modified: " + file.lastModified());
    }
}
```

Expected Output:

Case 1: If the argument is a directory with .txt files

/path/to/directory is a directory.
Text files in the directory:
file1.txt
file2.txt
Do you want to delete all the text files? (yes/no): yes
2 text file(s) deleted from the directory.

Case 2: If the argument is a file

/path/to/file.txt is a file.

File Details:

File Name: file.txt

Absolute Path: /path/to/file.txt

File Size: 2048 bytes

Readable: true

Writable: true

Executable: false

Last Modified: 1633059200000

Case 3: If the argument does not exist

The provided path does not exist.

Slip 28

Q.1) Write a program that reads on file name from the user, then displays information about whether the file exists, whether the file is readable, whether the file is writable, the type of file, and the length of the file in bytes.

```
import java.io.File;

import java.util.Scanner;


public class FileInfo {

    public static void main(String[] args) {

        // Create a Scanner object to get input from the user
```

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

// Prompt the user to enter the file name
System.out.print("Enter the file name with its path: ");
String fileName = scanner.nextLine();

// Create a File object using the input from the user
File file = new File(fileName);

// Check if the file exists
if (file.exists()) {
    System.out.println("File exists.");

    // Check if the file is readable
    if (file.canRead()) {
        System.out.println("File is readable.");
    } else {
        System.out.println("File is not readable.");
    }

    // Check if the file is writable
    if (file.canWrite()) {
```

```
            System.out.println("File is writable.");
        } else {
            System.out.println("File is not writable.");
        }

        // Check if the file is a directory or a regular file
        if (file.isDirectory()) {
            System.out.println("It is a directory.");
        } else {
            System.out.println("It is a regular file.");
        }

        // Display the length of the file
        System.out.println("File length: " + file.length() + " bytes");
    } else {
        System.out.println("The file does not exist.");
    }

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

Expected Output:

Case 1: File exists and is readable and writable

Enter the file name with its path: /path/to/file.txt
File exists.
File is readable.
File is writable.
It is a regular file.
File length: 2048 bytes

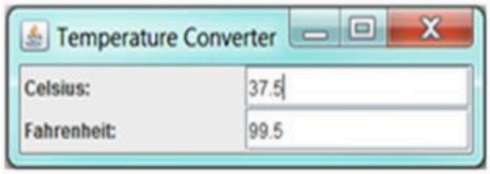
Case 2: File does not exist

Enter the file name with its path: /path/to/nonexistentfile.txt
The file does not exist.

Case 3: Directory input

Enter the file name with its path: /path/to/directory
File exists.
File is readable.
File is writable.
It is a directory.
File length: 4096 bytes

Q.2) Write a program called SwingTemperatureConverter to convert temperature values between Celsius and Fahrenheit. User can enter either the Celsius or the Fahrenheit value in floating point number. Hint: to display a floating point number in a specific format, (e.g. 1 decimal place), use the static method StringFormat(), which has the same form as printf() for example, String.Format("%If", 1.234) returns String "1.2".




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

public class SwingTemperatureConverter extends JFrame {
    private JTextField celsiusField;
    private JTextField fahrenheitField;

    public SwingTemperatureConverter() {
        // Create a frame with a title
        setTitle("Temperature Converter");

        // Create labels
        JLabel celsiusLabel = new JLabel("Celsius:");
        JLabel fahrenheitLabel = new JLabel("Fahrenheit:");

        // Create text fields
        celsiusField = new JTextField(10);
        fahrenheitField = new JTextField(10);

        // Set default values in text fields
        celsiusField.setText("0.0");
        fahrenheitField.setText("32.0");

        // Add action listeners to handle conversions
        celsiusField.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                convertCelsiusToFahrenheit();
            }
        });

        fahrenheitField.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
```

```
            convertFahrenheitToCelsius();
        }
    });

    // Layout the components in a grid
    setLayout(new java.awt.GridLayout(2, 2));

    // Add components to the frame
    add(celsiusLabel);
    add(celsiusField);
    add(fahrenheitLabel);
    add(fahrenheitField);

    // Set default close operation and size
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setSize(300, 100);
    setVisible(true);
}

// Convert Celsius to Fahrenheit
private void convertCelsiusToFahrenheit() {
    try {
        double celsius = Double.parseDouble(celsiusField.getText());
        double fahrenheit = celsius * 9.0 / 5.0 + 32.0;
        fahrenheitField.setText(String.format("%.1f", fahrenheit));
    } catch (NumberFormatException e) {
        JOptionPane.showMessageDialog(this, "Please enter a valid number.");
    }
}

// Convert Fahrenheit to Celsius
private void convertFahrenheitToCelsius() {
    try {
        double fahrenheit = Double.parseDouble(fahrenheitField.getText());
        double celsius = (fahrenheit - 32.0) * 5.0 / 9.0;
        celsiusField.setText(String.format("%.1f", celsius));
    } catch (NumberFormatException e) {
```

```
            JOptionPane.showMessageDialog(this, "Please enter a valid number.");
        }
    }

    public static void main(String[] args) {
        // Run the temperature converter
        new SwingTemperatureConverter();
    }
}
```

Slip 29

Q.1) Write a program to create a class Customer (CustNo, CustName, ContactNumber, CustAddr). Write a method to search the customer name with given contact number and display the details.

```
import java.util.ArrayList;
import java.util.Scanner;

class Customer {
    private int custNo;
    private String custName;
    private String contactNumber;
    private String custAddr;

    // Constructor to initialize Customer object
    public Customer(int custNo, String custName, String contactNumber, String
custAddr) {
        this.custNo = custNo;
        this.custName = custName;
        this.contactNumber = contactNumber;
        this.custAddr = custAddr;
    }

    // Getter for contact number
    public String getContactNumber() {
        return contactNumber;
```

```
    }

    // Method to display customer details
    public void displayCustomerDetails() {
        System.out.println("Customer No: " + custNo);
        System.out.println("Customer Name: " + custName);
        System.out.println("Contact Number: " + contactNumber);
        System.out.println("Customer Address: " + custAddr);
        System.out.println("----------------------------");
    }
}

public class CustomerSearch {

    // Method to search customer by contact number
    public static void searchCustomerByContact(ArrayList<Customer> customers,
String contactNumber) {
        boolean found = false;
        for (Customer customer : customers) {
            if (customer.getContactNumber().equals(contactNumber)) {
                customer.displayCustomerDetails();
                found = true;
                break;
            }
        }
        if (!found) {
            System.out.println("No customer found with contact number: " +
contactNumber);
        }
    }

    public static void main(String[] args) {
        // Create a list of customers
        ArrayList<Customer> customers = new ArrayList<>();

        // Add some customers to the list
```

```
        customers.add(new Customer(101, "John Doe", "1234567890", "123 Main St, City A"));
        customers.add(new Customer(102, "Jane Smith", "0987654321", "456 Elm St, City B"));
        customers.add(new Customer(103, "Mike Johnson", "1122334455", "789 Oak St, City C"));

        // Input contact number to search for
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter contact number to search: ");
        String contactNumber = scanner.nextLine();

        // Search for the customer by contact number
        searchCustomerByContact(customers, contactNumber);
    }
}
```

Expected Output:

If the contact number exists:

Enter contact number to search: 1234567890
Customer No: 101
Customer Name: John Doe
Contact Number: 1234567890
Customer Address: 123 Main St, City A

If the contact number does not exist:

Enter contact number to search: 5555555555

No customer found with contact number: 5555555555

Q.2) Write a program to create a super-class vehicle having members company and price. derive two different classes, LightMotorVehicle, (mileage) and HeavyMotorVehicle, (capacity_in_tons). Accept the information for "n" vehicles

and display the information in appropriate form. while taking data, ask user about the type of vehicle first.

```
import java.util.ArrayList;

import java.util.Scanner;


// Superclass Vehicle

class Vehicle {

    protected String company;

    protected double price;


    // Constructor for Vehicle class

    public Vehicle(String company, double price) {

        this.company = company;

        this.price = price;

    }


    // Method to display vehicle information

    public void display() {

        System.out.println("Company: " + company);

        System.out.println("Price: " + price);

    }

}
```

```
// Subclass for LightMotorVehicle
class LightMotorVehicle extends Vehicle {
    private double mileage;

    // Constructor for LightMotorVehicle
    public LightMotorVehicle(String company, double price, double mileage) {
        super(company, price);
        this.mileage = mileage;
    }

    // Method to display LightMotorVehicle information
    @Override
    public void display() {
        super.display();
        System.out.println("Mileage: " + mileage + " km/l");
    }
}

// Subclass for HeavyMotorVehicle
class HeavyMotorVehicle extends Vehicle {
    private double capacityInTons;
```

```
    // Constructor for HeavyMotorVehicle
    public HeavyMotorVehicle(String company, double price, double
capacityInTons) {
        super(company, price);
        this.capacityInTons = capacityInTons;
    }

    // Method to display HeavyMotorVehicle information
    @Override
    public void display() {
        super.display();
        System.out.println("Capacity: " + capacityInTons + " tons");
    }
}

public class VehicleInfoSystem {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        ArrayList<Vehicle> vehicles = new ArrayList<>();

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

```
        for (int i = 0; i < n; i++) {
            System.out.println("Enter the type of vehicle (1 for Light Motor Vehicle, 2 for Heavy Motor Vehicle): ");
            int type = scanner.nextInt();
            scanner.nextLine(); // Consume newline

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

            System.out.print("Enter price: ");
            double price = scanner.nextDouble();

            if (type == 1) {  // Light Motor Vehicle
                System.out.print("Enter mileage (in km/l): ");
                double mileage = scanner.nextDouble();
                vehicles.add(new LightMotorVehicle(company, price, mileage));
            } else if (type == 2) {  // Heavy Motor Vehicle
                System.out.print("Enter capacity (in tons): ");
                double capacityInTons = scanner.nextDouble();
                vehicles.add(new HeavyMotorVehicle(company, price, capacityInTons));
            } else {
```

```
                System.out.println("Invalid vehicle type.");
            }
        }

        // Display vehicle information
        System.out.println("\nVehicle Information:");
        for (Vehicle vehicle : vehicles) {
            vehicle.display();
            System.out.println("----------------------");
        }

        scanner.close();
    }
}
```

Example Input/Output:

Input:

Enter the number of vehicles: 2

Enter the type of vehicle (1 for Light Motor Vehicle, 2 for Heavy Motor Vehicle): 1

Enter company name: Honda

Enter price: 500000

Enter mileage (in km/l): 18

Enter the type of vehicle (1 for Light Motor Vehicle, 2 for Heavy Motor Vehicle): 2

Enter company name: Volvo

Enter price: 3000000

Enter capacity (in tons): 12

Output:

Vehicle Information:

Company: Honda

Price: 500000.0

Mileage: 18.0 km/l

Company: Volvo

Price: 3000000.0

Capacity: 12.0 tons

Slip 30

Q.1) Write program to define class Person with data member as PersonName, Aadharno, Panno. Accept information for 5 objects and display appropriate information. (Use this keyword).

```
class Person {
    // Data members
    private String personName;
    private String aadharNo;
```

```
    private String panNo;

    // Constructor to initialize Person details
    public Person(String personName, String aadharNo, String panNo) {
        this.personName = personName;  // Using 'this' to refer to the current object
        this.aadharNo = aadharNo;
        this.panNo = panNo;
    }

    // Method to display Person details
    public void display() {
        System.out.println("Person Name: " + this.personName);
        System.out.println("Aadhar No: " + this.aadharNo);
        System.out.println("PAN No: " + this.panNo);
        System.out.println("----------------------");
    }
}

public class PersonInfo {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Person[] persons = new Person[5]; // Array to store 5 Person objects
```

```
// Accepting information for 5 objects
for (int i = 0; i < 5; i++) {
    System.out.println("Enter details for person " + (i + 1) + ":");
    System.out.print("Enter Person Name: ");
    String name = scanner.nextLine();

    System.out.print("Enter Aadhar No: ");
    String aadharNo = scanner.nextLine();

    System.out.print("Enter PAN No: ");
    String panNo = scanner.nextLine();

    // Create a new Person object and store it in the array
    persons[i] = new Person(name, aadharNo, panNo);
}

// Displaying the information for each person
System.out.println("\nDetails of the Persons:");
for (Person person : persons) {
    person.display();
}
```

```
        scanner.close();
    }
}
```

Sample Output:

```
Enter details for person 1:
Enter Person Name: John Doe
Enter Aadhar No: 1234-5678-9123
Enter PAN No: ABCDE1234F

Enter details for person 2:
Enter Person Name: Jane Doe
Enter Aadhar No: 9876-5432-1098
Enter PAN No: FGHIJ5678L

Enter details for person 3:
Enter Person Name: Alice
Enter Aadhar No: 4321-6789-1234
Enter PAN No: LMNOP9876P

Enter details for person 4:
Enter Person Name: Bob
Enter Aadhar No: 8765-4321-0987
Enter PAN No: QRSTU7654R

Enter details for person 5:
Enter Person Name: Charlie
Enter Aadhar No: 5678-1234-9876
Enter PAN No: VWXYZ6543S

Details of the Persons:
Person Name: John Doe
```

Aadhar No: 1234-5678-9123
PAN No: ABCDE1234F

Person Name: Jane Doe
Aadhar No: 9876-5432-1098
PAN No: FGHIJ5678L

Person Name: Alice
Aadhar No: 4321-6789-1234
PAN No: LMNOP9876P

Person Name: Bob
Aadhar No: 8765-4321-0987
PAN No: QRSTU7654R

Person Name: Charlie
Aadhar No: 5678-1234-9876
PAN No: VWXYZ6543S

Q.2) Write a program that creates a user interface to perform integer divisions. The user enters two numbers in the text fields, number 1 and number 2. The division of number 1 and number 2 is displayed in the result field when the divide button is clicked. If number 1 or number 2 were not an integer, the program would throw a NumberFormatException. If number 2 were zero, the program would throw an arithmetic exception display the exception in a message dialog box.

Ans:-

```
import javax.swing.*;

import java.awt.*;

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

```
public class DivisionCalculator extends JFrame {
    // GUI components
    private JTextField number1Field, number2Field, resultField;
    private JButton divideButton;

    public DivisionCalculator() {
        // Set the title of the window
        setTitle("Integer Division Calculator");

        // Create labels and text fields for input and result
        JLabel number1Label = new JLabel("Number 1: ");
        JLabel number2Label = new JLabel("Number 2: ");
        JLabel resultLabel = new JLabel("Result: ");

        number1Field = new JTextField(10);
        number2Field = new JTextField(10);
        resultField = new JTextField(10);
        resultField.setEditable(false);  // Result field should not be editable

        // Create the "Divide" button
        divideButton = new JButton("Divide");
```

```
// Set the layout of the frame
setLayout(new GridLayout(4, 2));

// Add components to the frame
add(number1Label);
add(number1Field);
add(number2Label);
add(number2Field);
add(resultLabel);
add(resultField);
add(divideButton);

// Set action listener for the divide button
divideButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        try {
            // Parse input numbers
            int num1 = Integer.parseInt(number1Field.getText());
            int num2 = Integer.parseInt(number2Field.getText());
```

```
                // Perform division and display result
                int result = num1 / num2;
                resultField.setText(String.valueOf(result));
            } catch (NumberFormatException ex) {
                // Handle non-integer inputs
                JOptionPane.showMessageDialog(null, "Please enter valid integers.", "Error", JOptionPane.ERROR_MESSAGE);
            } catch (ArithmeticException ex) {
                // Handle division by zero
                JOptionPane.showMessageDialog(null, "Cannot divide by zero.", "Error", JOptionPane.ERROR_MESSAGE);
            }
        }
    });

    // Set frame size and default close operation
    setSize(400, 150);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setVisible(true);
}

public static void main(String[] args) {
    // Create and display the calculator window
```

```
        new DivisionCalculator();
    }
}
```

Sample Output:

When you enter two numbers, the result is displayed as:

Number 1: 10

Number 2: 2

Result: 5