Java.

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; // 0 and 1 are not prime
		for (int i = 2; i <= Math.sqrt(num); i++) {
			if (num % i == 0) return false;
		}
		return true;
	}

	public static void main(String[] args) {
		if (args.length == 0) {
			System.out.println("Please provide numbers as command line arguments.");
			return;
		}

		System.out.println("Prime numbers in the array:");
		for (String arg : args) {
			try {
				int number = Integer.parseInt(arg);
				if (isPrime(number)) {
					System.out.println(number);
				}
			} catch (NumberFormatException e) {
				System.out.println(arg + " is not a valid integer.");
			}
		}
	}
}
```

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.

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

// Abstract class Staff
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();
}

// Subclass OfficeStaff
class OfficeStaff extends Staff {
    private String department;

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

    // Implementation of abstract method
    @Override
    public void displayDetails() {
        System.out.println("ID: " + id + ", Name: " + name + ", Department: " + department);
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        List<OfficeStaff> staffList = new ArrayList<>();

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

        // Create n OfficeStaff objects
        for (int i = 0; i < n; i++) {
            System.out.print("Enter ID: ");
            int id = scanner.nextInt();
            scanner.nextLine(); // Consume newline
            System.out.print("Enter Name: ");
            String name = scanner.nextLine();
            System.out.print("Enter Department: ");
            String department = scanner.nextLine();

            staffList.add(new OfficeStaff(id, name, department));
        }

        // Display all details
        System.out.println("\nOffice Staff Details:");
        for (OfficeStaff staff : staffList) {
            staff.displayDetails();
        }

        scanner.close();
    }
}
```

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) {
        if (args.length != 4) {
            System.out.println("Please provide First Name, Last Name, Weight (kg), and Height (m) as command line arguments.");
            return;
        }

        String firstName = args[0];
        String lastName = args[1];
        double weight;
        double height;

        try {
            weight = Double.parseDouble(args[2]);
            height = Double.parseDouble(args[3]);

            if (height <= 0) {
                System.out.println("Height must be greater than zero.");
                return;
            }

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

            // Display results
            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);
        } catch (NumberFormatException e) {
            System.out.println("Weight and Height must be valid numbers.");
        }
    }
}
```

Q2) Define a class CricketPlayer (name,no_of_innings,no_of_times_notout, totatruns, 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;
import java.util.Scanner;

class CricketPlayer {
    String name;
    int no_of_innings;
    int no_of_times_notout;
    int totalRuns;
    double batAvg;

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

    // Static method to calculate batting average
    public static double avg(int no_of_innings, int totalRuns) {
        if (no_of_innings == 0) return 0.0; // Prevent division by zero
        return (double) totalRuns / no_of_innings;
    }

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

    // Static method to sort players by batting average
    public static void sortByAvg(CricketPlayer[] players) {
        Arrays.sort(players, Comparator.comparingDouble(player -> player.batAvg));
    }
}

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

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

        CricketPlayer[] players = new CricketPlayer[n];

        // Input player details
        for (int i = 0; i < n; i++) {
            System.out.print("Enter name of player " + (i + 1) + ": ");
            String name = scanner.nextLine();
            System.out.print("Enter number of innings: ");
            int no_of_innings = scanner.nextInt();
            System.out.print("Enter number of times not out: ");
            int no_of_times_notout = scanner.nextInt();
            System.out.print("Enter total runs: ");
            int totalRuns = scanner.nextInt();
            scanner.nextLine(); // Consume newline

            players[i] = new CricketPlayer(name, no_of_innings, no_of_times_notout,
totalRuns);
        }

        // Sort players by batting average
        CricketPlayer.sortByAvg(players);

        // Display sorted player details
        System.out.println("\nPlayers sorted by batting average:");
        for (CricketPlayer player : players) {
            player.displayDetails();
        }

        scanner.close();
    }
}
```

Slip-3.

Q1) Write a program to accept 'n' name of cities from the user and sort them in ascending order.import java.util.Arrays;

```
import java.util.Scanner;

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

        // Accept the number of cities from the user
        System.out.print("Enter the number of cities: ");
        int n = scanner.nextInt();
        scanner.nextLine(); // Consume the newline

        // Create an array to hold the city names
        String[] cities = new String[n];

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

        // Sort the array 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);
        }

        scanner.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.

```
// Custom exception class
class CovidPositiveException extends Exception {
    public CovidPositiveException(String message) {
        super(message);
    }
}

// Patient class
class Patient {
    private String patientName;
    private int patientAge;
    private double patientOxyLevel;
    private int patientHRCReport;

    public Patient(String patientName, int patientAge, double patientOxyLevel, int
patientHRCReport) {
        this.patientName = patientName;
        this.patientAge = patientAge;
        this.patientOxyLevel = patientOxyLevel;
        this.patientHRCReport = patientHRCReport;
    }

    public void checkPatientCondition() throws CovidPositiveException {
        if (patientOxyLevel < 95 || patientHRCReport > 10) {
            throw new CovidPositiveException("Patient is Covid Positive(+) and needs to
be hospitalized.");
        }
    }

    public void displayInformation() {
        System.out.println("Patient Name: " + patientName);
        System.out.println("Patient Age: " + patientAge);
        System.out.println("Oxygen Level: " + patientOxyLevel + "%");
        System.out.println("HRCT Report Score: " + patientHRCReport);
    }
}

public class PatientTest {
    public static void main(String[] args) {
        // Create a Patient object
        Patient patient = new Patient("John Doe", 30, 94.5, 12);

        // Check the patient's condition and handle exceptions
```

```
            try {
                patient.checkPatientCondition();
            } catch (CovidPositiveException e) {
                System.out.println(e.getMessage());
                // Optionally, you can display patient info even if they need hospitalization
                patient.displayInformation();
            }
        }
    }
```

Slip-4.

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

```
import java.util.Scanner;

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

        // Accept the dimensions of the matrix
        System.out.print("Enter the number of rows: ");
        int rows = scanner.nextInt();
        System.out.print("Enter the number of columns: ");
        int cols = scanner.nextInt();

        // Initialize the matrix
        int[][] matrix = new int[rows][cols];

        // Input matrix elements
        System.out.println("Enter the elements of the matrix:");
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                matrix[i][j] = scanner.nextInt();
            }
        }

        // Print the original matrix
        System.out.println("Original Matrix:");
        printMatrix(matrix);

        // Transpose the matrix
        int[][] transposedMatrix = transposeMatrix(matrix);

        // Print the transposed matrix
        System.out.println("Transposed Matrix:");
        printMatrix(transposedMatrix);

        scanner.close();
    }

    // Method to transpose the matrix
    public static int[][] transposeMatrix(int[][] matrix) {
        int rows = matrix.length;
        int cols = matrix[0].length;
        int[][] transposed = new int[cols][rows];

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

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

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.*;

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

public class LoginScreen extends Frame implements ActionListener {
    private TextField usernameField;
    private TextField passwordField;
    private Label messageLabel;
    private Button loginButton;
    private Button clearButton;
    private int attempts;

    public LoginScreen() {
        // Frame properties
        setTitle("Login Screen");
        setSize(300, 200);
        setLayout(new FlowLayout());

        // Username and Password Labels and TextFields
        Label usernameLabel = new Label("Username:");
        usernameField = new TextField(15);

        Label passwordLabel = new Label("Password:");
        passwordField = new TextField(15);
        passwordField.setEchoChar('*'); // Mask password input

        // Buttons
        loginButton = new Button("Login");
        clearButton = new Button("Clear");

        // Message Label
        messageLabel = new Label("");

        // Adding components to the frame
        add(usernameLabel);
        add(usernameField);
        add(passwordLabel);
        add(passwordField);
        add(loginButton);
        add(clearButton);
        add(messageLabel);

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

        // Initialize attempts
        attempts = 3;

        // Set frame visibility and close operation
        setVisible(true);
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent we) {
                System.exit(0);
            }
        });
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == loginButton) {
            handleLogin();
        } else if (e.getSource() == clearButton) {
            clearFields();
        }
    }

    private void handleLogin() {
        String username = usernameField.getText();
        String password = passwordField.getText();

        try {
            if (!username.equals(password)) {
                attempts--;
                if (attempts <= 0) {
                    messageLabel.setText("No more attempts!");
                    loginButton.setEnabled(false); // Disable login button after 3 failed
attempts
                } else {
                    throw new InvalidCredentialsException("Username and password do
not match. Attempts left: " + attempts);
```

```
                }
            } else {
                messageLabel.setText("Login successful!");
            }
        } catch (InvalidCredentialsException e) {
            messageLabel.setText(e.getMessage());
        }
    }

    private void clearFields() {
        usernameField.setText("");
        passwordField.setText("");
        messageLabel.setText("");
        attempts = 3; // Reset attempts
        loginButton.setEnabled(true); // Re-enable login button
    }

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

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
class Continent {
    String continentName;

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

    public String getContinentName() {
        return continentName;
    }
}

// Intermediate class
class Country extends Continent {
    String countryName;

    public Country(String continentName, String countryName) {
        super(continentName);
        this.countryName = countryName;
    }

    public String getCountryName() {
        return countryName;
    }
}

// Derived class
class State extends Country {
    String stateName;
    String placeName;

    public State(String continentName, String countryName, String stateName, String
placeName) {
        super(continentName, countryName);
        this.stateName = stateName;
        this.placeName = placeName;
    }

    public void displayDetails() {
        System.out.println("Place: " + placeName);
        System.out.println("State: " + stateName);
        System.out.println("Country: " + getCountryName());
        System.out.println("Continent: " + getContinentName());
    }
}

// Main class
public class MultilevelInheritance {
    public static void main(String[] args) {
        // Create an object of State
        State state = new State("Asia", "India", "Maharashtra", "Mumbai");

        // Display details
        state.displayDetails();
    }
}
```

Q2) Write a menu driven program to perform the following operations on multidimensional array
ie matrices :
Addition
Multiplication
Exit.

```
import java.util.Scanner;

public class MatrixOperations {

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

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

            switch (choice) {
                case 1:
                    performAddition(scanner);
                    break;
                case 2:
                    performMultiplication(scanner);
                    break;
                case 3:
                    System.out.println("Exiting...");
                    break;
                default:
                    System.out.println("Invalid choice! Please choose again.");
            }
        } while (choice != 3);

        scanner.close();
    }

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

        int[][] matrixA = new int[rows][cols];
        int[][] matrixB = new int[rows][cols];
        int[][] result = new int[rows][cols];

        System.out.println("Enter elements of Matrix A:");
        fillMatrix(scanner, matrixA);

        System.out.println("Enter elements of Matrix B:");
        fillMatrix(scanner, matrixB);

        // Perform addition
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                result[i][j] = matrixA[i][j] + matrixB[i][j];
            }
        }

        System.out.println("Result of Addition:");
        printMatrix(result);
    }

    private static void performMultiplication(Scanner scanner) {
        System.out.print("Enter number of rows for Matrix A: ");
        int rowsA = scanner.nextInt();
        System.out.print("Enter number of columns for Matrix A (and rows for Matrix B): ");
        int colsA = scanner.nextInt();
        System.out.print("Enter number of columns for Matrix B: ");
        int colsB = scanner.nextInt();

        int[][] matrixA = new int[rowsA][colsA];
        int[][] matrixB = new int[colsA][colsB];
        int[][] result = new int[rowsA][colsB];

        System.out.println("Enter elements of Matrix A:");
        fillMatrix(scanner, matrixA);

        System.out.println("Enter elements of Matrix B:");
        fillMatrix(scanner, matrixB);

        // Perform multiplication
        for (int i = 0; i < rowsA; i++) {
            for (int j = 0; j < colsB; j++) {
                result[i][j] = 0; // Initialize the result cell
                for (int k = 0; k < colsA; k++) {
                    result[i][j] += matrixA[i][k] * matrixB[k][j];
                }
            }
        }

        System.out.println("Result of Multiplication:");
        printMatrix(result);
    }

    private static void fillMatrix(Scanner scanner, int[][] matrix) {
        for (int i = 0; i < matrix.length; i++) {
            for (int j = 0; j < matrix[i].length; j++) {
                System.out.print("Enter element [" + i + "][" + 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();
        }
    }
}
```

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 empSal;

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

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

    // Main method to test the Employee class
    public static void main(String[] args) {
        // Create an Employee object
        Employee employee = new Employee(101, "John Doe", "Software Engineer",
75000.00);

        // Display employee information using toString()
        System.out.println(employee);
    }
}
```

Q2) Create an abstract class “order” having members id, description. Create two subclasses
“PurchaseOrder” 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
abstract class Order {
    protected int id;
    protected String description;

    public Order(int id, String description) {
        this.id = id;
        this.description = description;
    }

    public abstract void acceptDetails();
    public abstract void displayDetails();
}

// Subclass for Purchase Order
class PurchaseOrder extends Order {
    private String customerName;
    public PurchaseOrder(int id, String description) {
        super(id, description);
    }

    @Override
    public void acceptDetails() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter Customer Name for Purchase Order: ");
        customerName = scanner.nextLine();
    }

    @Override
    public void displayDetails() {
        System.out.println("Purchase Order ID: " + id);
        System.out.println("Description: " + description);
        System.out.println("Customer Name: " + customerName);
    }
}

// Subclass for Sales Order
class SalesOrder extends Order {
    private String vendorName;

    public SalesOrder(int id, String description) {
        super(id, description);
    }

    @Override
    public void acceptDetails() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter Vendor Name for Sales Order: ");
        vendorName = scanner.nextLine();
    }

    @Override
    public void displayDetails() {
        System.out.println("Sales Order ID: " + id);
        System.out.println("Description: " + description);
        System.out.println("Vendor Name: " + vendorName);
    }
}

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

        // Create and accept details for 3 Purchase Orders
        PurchaseOrder[] purchaseOrders = new PurchaseOrder[3];
        for (int i = 0; i < 3; i++) {
            System.out.print("Enter ID for Purchase Order " + (i + 1) + ": ");
            int id = scanner.nextInt();
            scanner.nextLine(); // Consume newline
            System.out.print("Enter Description for Purchase Order " + (i + 1) + ": ");
            String description = scanner.nextLine();
            purchaseOrders[i] = new PurchaseOrder(id, description);
            purchaseOrders[i].acceptDetails();
        }

        // Create and accept details for 3 Sales Orders
        SalesOrder[] salesOrders = new SalesOrder[3];
        for (int i = 0; i < 3; i++) {
            System.out.print("Enter ID for Sales Order " + (i + 1) + ": ");
            int id = scanner.nextInt();
            scanner.nextLine(); // Consume newline
            System.out.print("Enter Description for Sales Order " + (i + 1) + ": ");
            String description = scanner.nextLine();
            salesOrders[i] = new SalesOrder(id, description);
            salesOrders[i].acceptDetails();
        }

        // Display all Purchase Orders
        System.out.println("\nPurchase Orders:");
        for (PurchaseOrder order : purchaseOrders) {
            order.displayDetails();
            System.out.println();
        }

        // Display all Sales Orders
        System.out.println("Sales Orders:");
        for (SalesOrder order : salesOrders) {
            order.displayDetails();
            System.out.println();
        }

        scanner.close();
    }
}
```

Slp-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.

```
public class BankAccount {
    private String accountNumber;
    private double balance;

    public BankAccount(String accountNumber) {
        this.accountNumber = accountNumber;
        this.balance = 0.0;
    }

    public void deposit(double amount) {
        if (amount <= 0) {
            throw new IllegalArgumentException("Deposit amount must be positive.");
        }
        balance += amount;
        System.out.printf("Deposited: $%.2f. New Balance: $%.2f%n", amount, balance);
    }

    public void withdraw(double amount) {
        if (amount <= 0) {
            throw new IllegalArgumentException("Withdrawal amount must be positive.");
        }
        if (amount > balance) {
            throw new IllegalArgumentException("Insufficient funds.");
        }
        balance -= amount;
        System.out.printf("Withdrew: $%.2f. New Balance: $%.2f%n", amount, balance);
    }

    public String getBalance() {
        return String.format("Account Number: %s, Balance: $%.2f", accountNumber,
balance);
    }

    // Example usage
    public static void main(String[] args) {
        BankAccount account = new BankAccount("123456789");
        account.deposit(1000);
        account.withdraw(200);
        System.out.println(account.getBalance());
    }
}
```

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.

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.Scanner;

public class ReverseAndChangeCase {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the path of the text file: ");
        String filePath = scanner.nextLine();

        try {
            StringBuilder content = new StringBuilder();
            BufferedReader reader = new BufferedReader(new FileReader(filePath));
            String line;

            // Read the file line by line
            while ((line = reader.readLine()) != null) {
                content.append(line).append("\n");
            }
            reader.close();

            // Reverse the content and change case
            String reversedContent = reverseAndChangeCase(content.toString());
            System.out.println("Reversed and Case Changed Content:\n");
            System.out.println(reversedContent);

        } catch (IOException e) {
            System.out.println("An error occurred: " + e.getMessage());
        } finally {
            scanner.close();
        }
    }

    private static String reverseAndChangeCase(String input) {
        StringBuilder reversed = new StringBuilder();
        // Traverse the input in reverse order
        for (int i = input.length() - 1; i >= 0; i--) {
            char currentChar = input.charAt(i);
            // Change case
            if (Character.isLowerCase(currentChar)) {
                reversed.append(Character.toUpperCase(currentChar));
            } else {
                reversed.append(Character.toLowerCase(currentChar));
            }
        }
        return reversed.toString();
    }
}
```

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
    public Sphere(double radius) {
        this.radius = radius;
    }

    // Method to calculate surface area
    public double calculateSurfaceArea() {
        return 4 * Math.PI * radius * radius; // Using Math.PI for better precision
    }

    // Method to calculate volume
    public double calculateVolume() {
        return (4.0 / 3.0) * Math.PI * radius * radius * radius; // Using Math.PI for better
precision
    }

    // Main method for testing
    public static void main(String[] args) {
        Sphere sphere = new Sphere(5.0); // Example radius
        System.out.printf("Surface Area: %.2f%n", sphere.calculateSurfaceArea());
        System.out.printf("Volume: %.2f%n", sphere.calculateVolume());
    }
}
```

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.

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

public class MouseEventExample extends JFrame {
    private JTextField textField;

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

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

        // Add a mouse listener to the frame
        addMouseListener(new MouseAdapter() {
            @Override
            public void mouseClicked(MouseEvent e) {
                // Get mouse click position and display it in the text field
                int x = e.getX();
                int y = e.getY();
                textField.setText("Mouse Clicked at: (" + x + ", " + y + ")");
            }
        });

        // Add a mouse motion listener to the frame
        addMouseMotionListener(new MouseMotionAdapter() {
            @Override
            public void mouseMoved(MouseEvent e) {
                // Update the text field with the current mouse position
                int x = e.getX();
                int y = e.getY();
                textField.setText("Mouse Moved to: (" + x + ", " + y + ")");
            }
        });

        setVisible(true);
    }

    public static void main(String[] args) {
        // Run the application
        SwingUtilities.invokeLater(() -> new MouseEventExample());
    }
}
```

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 isAM; // true for AM, false for PM

    // Constructor
    public Clock(int hours, int minutes, int seconds, boolean isAM) {
        if (isValidTime(hours, minutes, seconds)) {
            this.hours = hours;
            this.minutes = minutes;
            this.seconds = seconds;
            this.isAM = isAM;
        } else {
            throw new IllegalArgumentException("Invalid time values provided.");
        }
    }

    // Method to check the validity of time
    private boolean isValidTime(int hours, int minutes, int seconds) {
        return (hours >= 1 && hours <= 12) && (minutes >= 0 && minutes < 60) &&
(seconds >= 0 && seconds < 60);
    }

    // Method to set the time
    public void setTime(int hours, int minutes, int seconds, boolean isAM) {
        if (isValidTime(hours, minutes, seconds)) {
            this.hours = hours;
            this.minutes = minutes;
            this.seconds = seconds;
            this.isAM = isAM;
        } else {
            throw new IllegalArgumentException("Invalid time values provided.");
        }
    }

    // Method to get the time in AM/PM format
    public String getTime() {
        String amPm = isAM ? "AM" : "PM";
        return String.format("%02d:%02d:%02d %s", hours, minutes, seconds, amPm);
    }

    public static void main(String[] args) {
        try {
            Clock clock = new Clock(10, 30, 45, true);
```

```
                System.out.println("Current time: " + clock.getTime());

                clock.setTime(11, 59, 59, false);
                System.out.println("Updated time: " + clock.getTime());
            } catch (IllegalArgumentException e) {
                System.out.println(e.getMessage());
            }
        }
    }
```

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 objectsof class
 product and display the contents of each object and Also display the object count.

```
// Marker interface
interface ProductMarker {}

// Product class
class Product implements ProductMarker {
    private static int objectCount = 0; // static variable to count instances

    private int productId;
    private String productName;
    private double productCost;
    private int productQuantity;

    // Default constructor
    public Product() {
        this.productId = 0;
        this.productName = "Unknown";
        this.productCost = 0.0;
        this.productQuantity = 0;
        objectCount++;
    }

    // Parameterized constructor
    public Product(int productId, String productName, double productCost, int
productQuantity) {
        this.productId = productId;
        this.productName = productName;
        this.productCost = productCost;
        this.productQuantity = productQuantity;
        objectCount++;
    }

    // Method to display product details
    public void displayProduct() {
        System.out.println("Product ID: " + productId);
        System.out.println("Product Name: " + productName);
        System.out.println("Product Cost: $" + productCost);
        System.out.println("Product Quantity: " + productQuantity);
        System.out.println();
    }

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

public class Main {
    public static void main(String[] args) {
        // Creating Product objects
        Product product1 = new Product(101, "Laptop", 1200.00, 5);
        Product product2 = new Product(102, "Smartphone", 800.00, 10);
        Product product3 = new Product(); // Default constructor

        // Displaying product details
        product1.displayProduct();
        product2.displayProduct();
        product3.displayProduct();

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

Slip-10.

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

```
// Define a functional interface
@FunctionalInterface
interface CubeCalculator {
    int calculateCube(int number);
}

public class CubeExample {
    public static void main(String[] args) {
        // Lambda expression to calculate the cube of a number
        CubeCalculator cubeCalc = (number) -> number * number * number;

        // Example number
        int number = 5;

        // Calculate the cube using the functional interface
        int cube = cubeCalc.calculateCube(number);

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

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
StudentPer with method to find percentage of the student. Accept student details like
rollno, name, class and marks of 6 subject from user.

```
src/
└── student/
    ├── StudentInfo.java
    └── StudentPer.java
```

```
package student;

public class StudentInfo {
    private int rollNo;
    private String name;
    private String className;

    // Constructor
    public StudentInfo(int rollNo, String name, String className) {
        this.rollNo = rollNo;
        this.name = name;
        this.className = className;
    }

    // Method to display student information
    public void displayInfo() {
        System.out.println("Roll No: " + rollNo);
        System.out.println("Name: " + name);
        System.out.println("Class: " + className);
    }
}
```

```
package student;

import java.util.Scanner;

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

        // Accept student details
        System.out.print("Enter Roll No: ");
        int rollNo = scanner.nextInt();
        scanner.nextLine(); // Consume newline

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

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

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

        // Create StudentInfo object
        StudentInfo studentInfo = new StudentInfo(rollNo, name, className);
        studentInfo.displayInfo();

        // Calculate and display percentage
        double percentage = calculatePercentage(marks);
        System.out.println("Percentage: " + percentage + "%");

        scanner.close();
    }

    // Method to calculate percentage
    public static double calculatePercentage(double[] marks) {
        double total = 0;
        for (double mark : marks) {
            total += mark;
        }
        return (total / (marks.length * 100)) * 100; // Assuming each subject is out of 100
    }
}
```

```
javac student/*.java

java student.StudentPer
```

Slip-11.
Q1) Define an interface “Operation” which has method volume( ).Define a constant PI having a value
3.142 Create a class cylinder which implements this interface (members – radius,height). Create
one object and calculate the volume.

```
public interface Operation {
    double PI = 3.142; // Constant for PI
    double volume(); // Method to calculate volume
}

public class Cylinder implements Operation {
    private double radius;
    private double height;

    // Constructor to initialize radius and height
    public Cylinder(double radius, double height) {
        this.radius = radius;
        this.height = height;
    }

    // Implementing the volume method
    @Override
    public double volume() {
        return PI * radius * radius * height; // Volume formula: πr²h
    }
    public static void main(String[] args) {
        // Create a Cylinder object
        Cylinder cylinder = new Cylinder(5.0, 10.0);

        // Calculate and display the volume
        double volume = cylinder.volume();
        System.out.printf("The volume of the cylinder is: %.2f\n", volume);
    }
}
```

How to compile this code.

```
javac Operation.java Cylinder.java

java Cylinder
```

Q2) Write a program to accept the username and password from user if username and password
are
not same then raise "Invalid Password" with appropriate msg.

```
import java.util.Scanner;

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

        // Accept username
        System.out.print("Enter your username: ");
        String username = scanner.nextLine();

        // Accept password
        System.out.print("Enter your password: ");
        String password = scanner.nextLine();

        // Check if username and password are the same
        if (username.equals(password)) {
            System.out.println("Invalid Password: Username and password cannot be the
same.");
        } else {
            System.out.println("Access granted!");
        }

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

javac UserAuthentication.java


java UserAuthentication
```

Slip-12.

Q1) Write a program to create parent class College(cno, cname, caddr) and derived class Department(dno, dname) from College. Write a necessary methods to display College details.

```
// Parent class
class College {
	private int cno;
	private String cname;
	private String caddr;

	public College(int cno, String cname, String caddr) {
		this.cno = cno;
		this.cname = cname;
		this.caddr = caddr;
	}

	public void displayDetails() {
		System.out.println("College Number: " + cno);
		System.out.println("College Name: " + cname);
		System.out.println("College Address: " + caddr);
	}
}

// Derived class
class Department extends College {
	private int dno;
	private String dname;

	public Department(int cno, String cname, String caddr, int dno, String dname) {
		super(cno, cname, caddr); // Call the constructor of the parent class
		this.dno = dno;
		this.dname = dname;
	}

	@Override
	public void displayDetails() {
		// Display college details first
		super.displayDetails();
		// Then display department details
		System.out.println("Department Number: " + dno);
		System.out.println("Department Name: " + dname);
	}
}

// Main class
public class Main {
	public static void main(String[] args) {
		Department dept = new Department(101, "Engineering College", "123 College St.",
10, "Computer Science");
		dept.displayDetails();
	}
}
```

Q2) Write a java program that works as a simple calculator. Use a grid layout to arrange buttons for
the digits and for the +, -, *, % operations. Add a text field to display the result.

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

public class SimpleCalculator extends JFrame implements ActionListener {

    // Declare all components
    JTextField display;
    JButton[] numberButtons = new JButton[10];
    JButton[] functionButtons = new JButton[8];
    JButton addButton, subButton, mulButton, divButton;
    JButton decButton, equButton, delButton, clrButton;
    JPanel panel;

    // Store values and operator
    double num1 = 0, num2 = 0, result = 0;
    char operator;

    // Constructor to set up the calculator UI
    public SimpleCalculator() {
        // Set frame properties
        setTitle("Simple Calculator");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(420, 550);
        setLayout(null);

        // Create the display field
        display = new JTextField();
        display.setBounds(50, 25, 300, 50);
        display.setEditable(false);
        add(display);

        // Define buttons
        addButton = new JButton("+");
        subButton = new JButton("-");
        mulButton = new JButton("*");
        divButton = new JButton("/");
        decButton = new JButton(".");
        equButton = new JButton("=");
        delButton = new JButton("Del");
        clrButton = new JButton("Clr");

        // Store function buttons in array
        functionButtons[0] = addButton;
        functionButtons[1] = subButton;
        functionButtons[2] = mulButton;
        functionButtons[3] = divButton;
        functionButtons[4] = decButton;
        functionButtons[5] = equButton;
        functionButtons[6] = delButton;
        functionButtons[7] = clrButton;
        // Set action listeners for function buttons
        for (int i = 0; i < 8; i++) {
            functionButtons[i].addActionListener(this);
        }

        // Create number buttons and add action listeners
        for (int i = 0; i < 10; i++) {
            numberButtons[i] = new JButton(String.valueOf(i));
            numberButtons[i].addActionListener(this);
        }

        // Create panel for buttons with grid layout (4 rows, 4 columns)
        panel = new JPanel();
        panel.setBounds(50, 100, 300, 300);
        panel.setLayout(new GridLayout(4, 4, 10, 10));

        // Add buttons to panel
        panel.add(numberButtons[1]);
        panel.add(numberButtons[2]);
        panel.add(numberButtons[3]);
        panel.add(addButton);
        panel.add(numberButtons[4]);
        panel.add(numberButtons[5]);
        panel.add(numberButtons[6]);
        panel.add(subButton);
        panel.add(numberButtons[7]);
        panel.add(numberButtons[8]);
        panel.add(numberButtons[9]);
        panel.add(mulButton);
        panel.add(decButton);
        panel.add(numberButtons[0]);
        panel.add(equButton);
        panel.add(divButton);

        // Add panel to frame
        add(panel);

        // Add delete and clear buttons separately
        delButton.setBounds(50, 430, 145, 50);
        clrButton.setBounds(205, 430, 145, 50);
        add(delButton);
        add(clrButton);

        setVisible(true);
    }

    // Perform calculator operations
    @Override
    public void actionPerformed(ActionEvent e) {
        for (int i = 0; i < 10; i++) {
            if (e.getSource() == numberButtons[i]) {
                display.setText(display.getText().concat(String.valueOf(i)));
            }
        }
        if (e.getSource() == decButton) {
            display.setText(display.getText().concat("."));
        }
        if (e.getSource() == addButton) {
            num1 = Double.parseDouble(display.getText());
            operator = '+';
            display.setText("");
        }
        if (e.getSource() == subButton) {
            num1 = Double.parseDouble(display.getText());
            operator = '-';
            display.setText("");
        }
        if (e.getSource() == mulButton) {
            num1 = Double.parseDouble(display.getText());
            operator = '*';
            display.setText("");
        }
        if (e.getSource() == divButton) {
            num1 = Double.parseDouble(display.getText());
            operator = '/';
            display.setText("");
        }
        if (e.getSource() == equButton) {
            num2 = Double.parseDouble(display.getText());

            switch (operator) {
                case '+':
                    result = num1 + num2;
                    break;
                case '-':
                    result = num1 - num2;
                    break;
                case '*':
                    result = num1 * num2;
                    break;
                case '/':
                    result = num1 / num2;
                    break;
            }
            display.setText(String.valueOf(result));
            num1 = result;
        }
        if (e.getSource() == clrButton) {
            display.setText("");
        }
        if (e.getSource() == delButton) {
            String string = display.getText();
            display.setText("");
            for (int i = 0; i < string.length() - 1; i++) {
                display.setText(display.getText() + string.charAt(i));
            }
        }
    }

    public static void main(String[] args) {
        new SimpleCalculator();
    }
}
```

Slip-13.

Q1) Write a program to accept a file name from command prompt, if the file exits then display
 number of words and lines in that file.

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;

public class FileWordLineCounter {
	public static void main(String[] args) {
		if (args.length != 1) {
			System.out.println("Usage: java FileWordLineCounter <filename>");
			return;
		}

		String fileName = args[0];
		File file = new File(fileName);

		if (!file.exists()) {
			System.out.println("File does not exist: " + fileName);
			return;
		}

		int lineCount = 0;
		int wordCount = 0;

		try (BufferedReader br = new BufferedReader(new FileReader(file))) {
			String line;
			while ((line = br.readLine()) != null) {
				lineCount++;
				// Split the line into words and count them
				String[] words = line.trim().split("\\s+");
				wordCount += words.length;
			}
		} catch (IOException e) {
			System.out.println("An error occurred while reading the file.");
			e.printStackTrace();
			return;
		}

		System.out.println("File: " + fileName);
		System.out.println("Number of lines: " + lineCount);
		System.out.println("Number of words: " + wordCount);
	}
}
```

Q2) Write a program to display the system date and time in various formats shown below:

```
Current date is : 31/08/2021
Current date is : 08-31-2021
Current date is : Tuesday August 31 2021
Current date and time is : Fri August 31 15:25:59 IST 2021
Current date and time is : 31/08/21 15:25:59 PM +0530.
```

```
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.time.ZoneId;

public class DateTimeFormats {
    public static void main(String[] args) {
        LocalDateTime now = LocalDateTime.now();

        // Format 1: Current date is : 31/08/2021
        DateTimeFormatter format1 = DateTimeFormatter.ofPattern("dd/MM/yyyy");
        System.out.println("Current date is : " + now.format(format1));

        // Format 2: Current date is : 08-31-2021
        DateTimeFormatter format2 = DateTimeFormatter.ofPattern("MM-dd-yyyy");
        System.out.println("Current date is : " + now.format(format2));

        // Format 3: Current date is : Tuesday August 31 2021
        DateTimeFormatter format3 = DateTimeFormatter.ofPattern("EEEE MMMM dd
yyyy");
        System.out.println("Current date is : " + now.format(format3));

        // Format 4: Current date and time is : Fri August 31 15:25:59 IST 2021
        DateTimeFormatter format4 = DateTimeFormatter.ofPattern("EEE MMMM dd
HH:mm:ss zzz yyyy");
        String formattedDateTime4 =
now.atZone(ZoneId.of("Asia/Kolkata")).format(format4);
        System.out.println("Current date and time is : " + formattedDateTime4);

        // Format 5: Current date and time is : 31/08/21 15:25:59 PM +0530
        DateTimeFormatter format5 = DateTimeFormatter.ofPattern("dd/MM/yy
hh:mm:ss a Z");
        String formattedDateTime5 =
now.atZone(ZoneId.of("Asia/Kolkata")).format(format5);
        System.out.println("Current date and time is : " + formattedDateTime5);
    }
}
```

```
javac DateTimeFormats.java

java DateTimeFormats
```

Slip-14.

Q1) Write a program to accept a number from the user, if number is zero then throw user defined
 exception “Number is 0” otherwise check whether no is prime or not (Use static keyword).

```
import java.util.Scanner;

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

public class PrimeChecker {
    // Static method to check if a number is prime
    public static boolean isPrime(int number) {
        if (number <= 1) {
            return false;
        }
        for (int i = 2; i <= Math.sqrt(number); i++) {
            if (number % i == 0) {
                return false;
            }
        }
        return true;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number: ");

        int number = scanner.nextInt();

        try {
            if (number == 0) {
                throw new ZeroException("Number is 0");
            } else {
                if (isPrime(number)) {
                    System.out.println(number + " is a prime number.");
                } else {
                    System.out.println(number + " is not a prime number.");
                }
            }
        } catch (ZeroException e) {
            System.out.println(e.getMessage());
        } finally {
            scanner.close();
        }
    }
}
```

Q2) Write a Java program to create a Package “SY” which has a class SYMarks (members –
ComputerTotal, MathsTotal, and ElectronicsTotal). Create another package TY which has a
class TYMarks (members – Theory, Practicals). Create ‘n’ objects of Student class (having
rollNumber, name, SYMarks and TYMarks). Add the marks of SY and TY computer subjects
and calculate the Grade (‘A’ for >= 70, ‘B’ for >= 60 ‘C’ for >= 50, Pass Class for > =40
else‘FAIL’) and display the result of the student in proper format.

```
package SY;

public class SYMarks {
	private int computerTotal;
	private int mathsTotal;
	private int electronicsTotal;

	public SYMarks(int computerTotal, int mathsTotal, int electronicsTotal) {
		this.computerTotal = computerTotal;
		this.mathsTotal = mathsTotal;
		this.electronicsTotal = electronicsTotal;
	}

	public int getComputerTotal() {
		return computerTotal;
	}

	public int getMathsTotal() {
		return mathsTotal;
	}

	public int getElectronicsTotal() {
		return electronicsTotal;
	}
}

package TY;

public class TYMarks {
	private int theory;
	private int practicals;

	public TYMarks(int theory, int practicals) {
		this.theory = theory;
		this.practicals = practicals;
	}

	public int getTheory() {
		return theory;
	}

	public int getPracticals() {
		return practicals;
	}
}
```

```
import SY.SYMarks;
import TY.TYMarks;
import java.util.Scanner;

public class Student {
    private int rollNumber;
    private String name;
    private SYMarks syMarks;
    private TYMarks tyMarks;

    public Student(int rollNumber, String name, SYMarks syMarks, TYMarks tyMarks) {
        this.rollNumber = rollNumber;
        this.name = name;
        this.syMarks = syMarks;
        this.tyMarks = tyMarks;
    }

    public String calculateGrade() {
        int totalMarks = syMarks.getComputerTotal() + tyMarks.getTheory();
        if (totalMarks >= 70) {
            return "A";
        } else if (totalMarks >= 60) {
            return "B";
        } else if (totalMarks >= 50) {
            return "C";
        } else if (totalMarks >= 40) {
            return "Pass Class";
        } else {
            return "FAIL";
        }
    }

    public void displayResult() {
        System.out.println("Roll Number: " + rollNumber);
        System.out.println("Name: " + name);
        System.out.println("Grade: " + calculateGrade());
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter number of students: ");
        int n = scanner.nextInt();
        scanner.nextLine();   // Consume newline

        Student[] students = new Student[n];

        for (int i = 0; i < n; i++) {
            System.out.print("Enter roll number for student " + (i + 1) + ": ");
            int rollNumber = scanner.nextInt();
            scanner.nextLine();   // Consume newline

            System.out.print("Enter name for student " + (i + 1) + ": ");
            String name = scanner.nextLine();

            System.out.print("Enter Computer Total marks: ");
            int computerTotal = scanner.nextInt();
            System.out.print("Enter Theory marks: ");
            int theory = scanner.nextInt();

            System.out.print("Enter Practicals marks: ");
            int practicals = scanner.nextInt();

            SYMarks syMarks = new SYMarks(computerTotal, 0, 0); // Only computer
total used
            TYMarks tyMarks = new TYMarks(theory, practicals);
            students[i] = new Student(rollNumber, name, syMarks, tyMarks);
        }

        System.out.println("\nResults:");
        for (Student student : students) {
            student.displayResult();
        }

        scanner.close();
    }
}
```

Create directories for packages:

```
mkdir SY
mkdir TY

javac SY/SYMarks.java
javac TY/TYMarks.java
javac Student.java

java Student
```

Slip-15.

Q1) Accept the names of two files and copy the contents of the first to the second. First file having
Book name and Author name in file.

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;

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

        System.out.print("Enter the name of the source file: ");
        String sourceFileName = scanner.nextLine();
        System.out.print("Enter the name of the destination file: ");
        String destinationFileName = scanner.nextLine();

        // Try-with-resources to ensure proper closing of resources
        try (BufferedReader reader = new BufferedReader(new
FileReader(sourceFileName));
             BufferedWriter writer = new BufferedWriter(new
FileWriter(destinationFileName))) {

            String line;
            // Read each line from the source file and write to the destination file
            while ((line = reader.readLine()) != null) {
                writer.write(line);
                writer.newLine(); // Add a new line after writing each line
            }

            System.out.println("Contents copied successfully from " + sourceFileName + "
to " + destinationFileName);

        } catch (IOException e) {
            System.out.println("An error occurred while copying the file: " +
e.getMessage());
        } finally {
            scanner.close();
        }
    }
}
```

Q2) Write a program to define a class Account having members custname, accno. Define default
and parameterized constructor. Create a subclass called SavingAccount with member savingbal,
minbal. Create a derived class AccountDetail that extends the class SavingAccount with
members, depositamt and withdrawalamt. Write a appropriate method to display customer
details.

```
// Base class Account
class Account {
    protected String custName;
    protected String accNo;

    // Default constructor
    public Account() {
        this.custName = "Unknown";
        this.accNo = "0000";
    }

    // Parameterized constructor
    public Account(String custName, String accNo) {
        this.custName = custName;
        this.accNo = accNo;
    }
}
```

```
// Subclass SavingAccount
class SavingAccount extends Account {
	protected double savingBal;
	protected double minBal;

	// Default constructor
	public SavingAccount() {
		super();
		this.savingBal = 0.0;
		this.minBal = 1000.0; // Setting a default minimum balance
	}

	// Parameterized constructor
	public SavingAccount(String custName, String accNo, double savingBal, double
minBal) {
		super(custName, accNo);
		this.savingBal = savingBal;
		this.minBal = minBal;
	}
}

// Derived class AccountDetail
class AccountDetail extends SavingAccount {
	private double depositAmt;
	private double withdrawalAmt;

	// Default constructor
	public AccountDetail() {
		super();
		this.depositAmt = 0.0;
		this.withdrawalAmt = 0.0;
	}

	// Parameterized constructor
	public AccountDetail(String custName, String accNo, double savingBal, double minBal)
{
		super(custName, accNo, savingBal, minBal);
		this.depositAmt = 0.0;
		this.withdrawalAmt = 0.0;
	}

	// Method to deposit amount
	public void deposit(double amount) {
		if (amount > 0) {
			savingBal += amount;
			depositAmt += amount;
			System.out.println("Deposited: " + amount);
		} else {
			System.out.println("Deposit amount must be positive.");
		}
	}

	// Method to withdraw amount
	public void withdraw(double amount) {
		if (amount > 0 && (savingBal - amount) >= minBal) {
			savingBal -= amount;
			withdrawalAmt += amount;
```

```
            System.out.println("Withdrawn: " + amount);
        } else {
            System.out.println("Insufficient balance or below minimum balance limit.");
        }
    }

    // Method to display customer details
    public void displayDetails() {
        System.out.println("Customer Name: " + custName);
        System.out.println("Account Number: " + accNo);
        System.out.println("Savings Balance: " + savingBal);
        System.out.println("Minimum Balance: " + minBal);
        System.out.println("Total Deposited Amount: " + depositAmt);
        System.out.println("Total Withdrawn Amount: " + withdrawalAmt);
    }

    public static void main(String[] args) {
        // Create an AccountDetail object
        AccountDetail account = new AccountDetail("John Doe", "12345", 5000, 1000);

        // Display initial account details
        account.displayDetails();

        // Deposit and withdraw operations
        account.deposit(1500);
        account.withdraw(2000);
        account.withdraw(5000); // Should fail due to minimum balance

        // Display final account details
        account.displayDetails();
    }
}
```

Slip-15.

Q1) Write a program to find the Square of given number using function interface.

```
import java.util.Scanner;

// Define a functional interface
@FunctionalInterface
interface Square {
    double calculateSquare(double number);
}

public class SquareCalculator {
    public static void main(String[] args) {
        // Create a lambda expression to implement the calculateSquare method
        Square squareFunction = (number) -> number * number;

        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number: ");
        double number = scanner.nextDouble();
        // Calculate the square using the functional interface
        double result = squareFunction.calculateSquare(number);

        System.out.println("The square of " + number + " is: " + result);

        scanner.close();
    }
}
```

Q2) Write a program to design a screen using Awt that,

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

public class AwtMenuExample extends Frame {

    public AwtMenuExample() {
        // Create the frame
        setTitle("Java AWT Examples");
        setSize(400, 400);
        setLayout(null);
        setVisible(true);

        // Create the MenuBar
        MenuBar mb = new MenuBar();

        // Create the File menu
        Menu file = new Menu("File");
        MenuItem newOption = new MenuItem("New");
        MenuItem open = new MenuItem("Open");
        MenuItem save = new MenuItem("Save");
        MenuItem exit = new MenuItem("Exit");
        file.add(newOption);
        file.add(open);
        file.add(save);
        file.addSeparator();    // separator line
        file.add(exit);

        // Create the Edit menu
        Menu edit = new Menu("Edit");

        // Create the About menu
        Menu about = new Menu("About");
        CheckboxMenuItem showAbout = new CheckboxMenuItem("Show About", true);
        about.add(showAbout);

        // Add menus to MenuBar
        mb.add(file);
        mb.add(edit);
        mb.add(about);

        // Set the MenuBar to the Frame
        setMenuBar(mb);
        // Add action listener for exit
        exit.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                System.exit(0);
            }
        });

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

    public static void main(String[] args) {
        new AwtMenuExample();
    }
}
```

Slip-17.

Q1) Design a Super class Customer (name, phone-number). Derive a class Depositor(accno , balance)
from Customer. Again, derive a class Borrower (loan-no, loan-amt) from Depositor. Write necessary member functions to read and display the details of ‘n’customers.

```
import java.util.Scanner;

// Superclass Customer
class Customer {
    protected String name;
    protected String phoneNumber;

    // Constructor
    public Customer(String name, String phoneNumber) {
        this.name = name;
        this.phoneNumber = phoneNumber;
    }

    // Method to display customer details
    public void displayCustomerDetails() {
        System.out.println("Customer Name: " + name);
        System.out.println("Phone Number: " + phoneNumber);
    }
}

// Derived class Depositor
class Depositor extends Customer {
    protected String accNo;
    protected double balance;

    // Constructor
    public Depositor(String name, String phoneNumber, String accNo, double balance) {
        super(name, phoneNumber);
        this.accNo = accNo;
        this.balance = balance;
    }

    // Method to display depositor details
    @Override
    public void displayCustomerDetails() {
        super.displayCustomerDetails();
        System.out.println("Account Number: " + accNo);
        System.out.println("Balance: " + balance);
    }
}

// Derived class Borrower
class Borrower extends Depositor {
    private String loanNo;
    private double loanAmt;

    // Constructor
    public Borrower(String name, String phoneNumber, String accNo, double balance,
String loanNo, double loanAmt) {
        super(name, phoneNumber, accNo, balance);
        this.loanNo = loanNo;
        this.loanAmt = loanAmt;
    }

    // Method to display borrower details
    @Override
    public void displayCustomerDetails() {
        super.displayCustomerDetails();
        System.out.println("Loan Number: " + loanNo);
        System.out.println("Loan Amount: " + loanAmt);
    }
}

// Main class to read and display details of n customers
public class CustomerDetails {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

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

        // Array to store customers
        Borrower[] customers = new Borrower[n];

        for (int i = 0; i < n; i++) {
            System.out.println("\nEntering details for customer " + (i + 1) + ":");

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

            System.out.print("Enter Phone Number: ");
            String phoneNumber = scanner.nextLine();
            System.out.print("Enter Account Number: ");
            String accNo = scanner.nextLine();

            System.out.print("Enter Balance: ");
            double balance = scanner.nextDouble();
            scanner.nextLine(); // Consume newline

            System.out.print("Enter Loan Number: ");
            String loanNo = scanner.nextLine();

            System.out.print("Enter Loan Amount: ");
            double loanAmt = scanner.nextDouble();
            scanner.nextLine(); // Consume newline

            // Create a new Borrower object and store it in the array
            customers[i] = new Borrower(name, phoneNumber, accNo, balance, loanNo,
loanAmt);
        }

        System.out.println("\nCustomer Details:");
        for (Borrower customer : customers) {
            customer.displayCustomerDetails();
            System.out.println(); // Add an empty line for better readability
        }

        scanner.close();
    }
}
```

```
javac CustomerDetails.java

java CustomerDetails
```

Q2) Write Java program to design three text boxes and two buttons using swing. Enter different
strings in first and second textbox. On clicking the First command button, concatenation of
two strings should be displayed in third text box and on clicking second command button,
reverse of string should display in third text box.

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

public class StringManipulationApp {
    // Creating the components
    private JTextField textField1;
    private JTextField textField2;
    private JTextField resultField;
    private JButton concatButton;
    private JButton reverseButton;

    public StringManipulationApp() {
        // Creating the frame
        JFrame frame = new JFrame("String Manipulation");
        frame.setSize(400, 200);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLayout(new FlowLayout());

        // Creating text fields
        textField1 = new JTextField(15);
        textField2 = new JTextField(15);
        resultField = new JTextField(15);
        resultField.setEditable(false); // Make result field read-only

        // Creating buttons
        concatButton = new JButton("Concatenate");
        reverseButton = new JButton("Reverse");

        // Adding action listeners to the buttons
        concatButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String text1 = textField1.getText();
                String text2 = textField2.getText();
                String concatenated = text1 + text2;
                resultField.setText(concatenated);
            }
        });

        reverseButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String text1 = textField1.getText();
                String reversed = new StringBuilder(text1).reverse().toString();
                resultField.setText(reversed);
            }
        });

        // Adding components to the frame
        frame.add(new JLabel("String 1:"));
        frame.add(textField1);
        frame.add(new JLabel("String 2:"));
        frame.add(textField2);
        frame.add(concatButton);
        frame.add(reverseButton);
        frame.add(new JLabel("Result:"));
        frame.add(resultField);

        // Make the frame visible
        frame.setVisible(true);
    }

    public static void main(String[] args) {
        // Run the application
        SwingUtilities.invokeLater(() -> new StringManipulationApp());
    }
}

javac StringManipulationApp.java
```

```
java StringManipulationApp
```

Slip-18.

Q1) Write a program to implement Border Layout Manager.

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

public class BorderLayoutExample {
    public static void main(String[] args) {
        // Create a new frame
        JFrame frame = new JFrame("Border Layout Example");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400, 300);

        // Set the layout manager
        frame.setLayout(new BorderLayout());

        // Create components
        JButton buttonNorth = new JButton("North");
        JButton buttonSouth = new JButton("South");
        JButton buttonEast = new JButton("East");
        JButton buttonWest = new JButton("West");
        JButton buttonCenter = new JButton("Center");

        // Add components to the frame in different regions
        frame.add(buttonNorth, BorderLayout.NORTH);
        frame.add(buttonSouth, BorderLayout.SOUTH);
        frame.add(buttonEast, BorderLayout.EAST);
        frame.add(buttonWest, BorderLayout.WEST);
        frame.add(buttonCenter, BorderLayout.CENTER);

        // Make the frame visible
        frame.setVisible(true);
    }
}
```

```
javac BorderLayoutExample.java

java BorderLayoutExample
```

Q2) Define a class CricketPlayer (name,no_of_innings,no_of_times_notout, totatruns, 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;
import java.util.Scanner;

// Class representing a cricket player
```

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

    // Constructor
    public CricketPlayer(String name, int noOfInnings, int noOfTimesNotOut, int totalRuns)
{
        this.name = name;
        this.noOfInnings = noOfInnings;
        this.noOfTimesNotOut = noOfTimesNotOut;
        this.totalRuns = totalRuns;
        this.battingAvg = calculateAvg(); // Calculate average upon object creation
    }

    // Static method to calculate batting average
    public static double avg(int totalRuns, int noOfInnings, int noOfTimesNotOut) {
        return (noOfInnings == 0) ? 0 : (double) totalRuns / (noOfInnings -
(noOfTimesNotOut));
    }

    // Method to calculate batting average for the instance
    private double calculateAvg() {
        return avg(totalRuns, noOfInnings, noOfTimesNotOut);
    }

    // Static method to sort players by batting average
    public static void sortByAverage(CricketPlayer[] players) {
        Arrays.sort(players, Comparator.comparingDouble(player -> player.battingAvg));
    }

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

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

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

        CricketPlayer[] players = new CricketPlayer[n];

        for (int i = 0; i < n; i++) {
            System.out.printf("Enter details for player %d:\n", (i + 1));
            System.out.print("Name: ");
            String name = scanner.nextLine();

            System.out.print("Number of innings: ");
```

```
                int noOfInnings = scanner.nextInt();

                System.out.print("Number of times not out: ");
                int noOfTimesNotOut = scanner.nextInt();

                System.out.print("Total runs: ");
                int totalRuns = scanner.nextInt();
                scanner.nextLine(); // Consume newline

                // Create a new CricketPlayer object
                players[i]   =   new   CricketPlayer(name,   noOfInnings,   noOfTimesNotOut,
totalRuns);
            }

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

            System.out.println("\nPlayers sorted by batting average:");
            for (CricketPlayer player : players) {
                player.display();
            }

            scanner.close();
        }
}
```

```
javac CricketPlayerDetails.java

java CricketPlayerDetails
```

Slip-19.

Q1) Write a program to accept the two dimensional array from user and display sum of its diagonal
elements.

```
import java.util.Scanner;

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

            // Accepting the size of the matrix
            System.out.print("Enter the size of the square matrix (n x n): ");
            int n = scanner.nextInt();

            // Declaring the two-dimensional array
            int[][] matrix = new int[n][n];

            // Accepting the elements of the matrix from the user
            System.out.println("Enter the elements of the matrix:");
            for (int i = 0; i < n; i++) {
```

```
            for (int j = 0; j < n; j++) {
                System.out.print("Element at [" + i + "][" + j + "]: ");
                matrix[i][j] = scanner.nextInt();
            }
        }

        // Calculating the sum of diagonal elements
        int diagonalSum = 0;
        for (int i = 0; i < n; i++) {
            diagonalSum += matrix[i][i]; // Sum for the main diagonal
        }

        // Displaying the sum of diagonal elements
        System.out.println("The sum of the diagonal elements is: " + diagonalSum);

        // Closing the scanner
        scanner.close();
    }
}
```

```
javac DiagonalSum.java

java DiagonalSum
```

Q2) Write a program which shows the combo box which includes list of T.Y.B.Sc.(Comp. Sci)
 subjects. Display the selected subject in a text field.

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

public class SubjectSelector {
    public static void main(String[] args) {
        // Create a new frame
        JFrame frame = new JFrame("T.Y.B.Sc. (Comp. Sci) Subjects");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400, 200);
        frame.setLayout(new FlowLayout());

        // Create a combo box with subjects
        String[] subjects = {
            "Data Structures",
            "Database Management Systems",
            "Operating Systems",
            "Computer Networks",
            "Software Engineering",
            "Web Technologies",
            "Object-Oriented Programming"
        };

        JComboBox<String> subjectComboBox = new JComboBox<>(subjects);

        // Create a text field to display the selected subject
        JTextField selectedSubjectField = new JTextField(20);
        selectedSubjectField.setEditable(false); // Make it read-only

        // Add an action listener to the combo box
        subjectComboBox.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String selectedSubject = (String) subjectComboBox.getSelectedItem();
                selectedSubjectField.setText(selectedSubject);    //    Display    selected
subject
            }
        });

        // Add components to the frame
        frame.add(new JLabel("Select a subject:"));
        frame.add(subjectComboBox);
        frame.add(new JLabel("Selected Subject:"));
        frame.add(selectedSubjectField);

        // Make the frame visible
        frame.setVisible(true);
    }
}
```

```
javac SubjectSelector.java

java SubjectSelector
```

Slip-20.

Q1) Write a Program to illustrate 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;

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

    public String getContinentName() {
        return continentName;
    }
}

// Derived class: Country
class Country extends Continent {
    String countryName;

    public Country(String continentName, String countryName) {
        super(continentName); // Call the constructor of Continent
        this.countryName = countryName;
    }
    public String getCountryName() {
        return countryName;
    }
}

// Derived class: State
class State extends Country {
    String stateName;
    String placeName;

    public State(String continentName, String countryName, String stateName, String
placeName) {
        super(continentName, countryName); // Call the constructor of Country
        this.stateName = stateName;
        this.placeName = placeName;
    }

    // Method to display details
    public void displayDetails() {
        System.out.println("Place: " + placeName);
        System.out.println("State: " + stateName);
        System.out.println("Country: " + getCountryName());
        System.out.println("Continent: " + getContinentName());
    }
}

// Main class
public class MultilevelInheritanceExample {
    public static void main(String[] args) {
        // Create a State object
        State state = new State("Asia", "India", "Maharashtra", "Mumbai");

        // Display the details
        state.displayDetails();
    }
}
```

```
javac MultilevelInheritanceExample.java

java MultilevelInheritanceExample
```

Q2) Write a package for Operation, which has two classes, Addition and Maximum. Addition has
two methods add () and subtract (), which are used to add two integers and subtract two, float values respectively. Maximum has a method max () to display the maximum of two integers.

```
// File: Operation/Addition.java
package Operation;

public class Addition {
    // Method to add two integers
    public int add(int a, int b) {
        return a + b;
    }

    // Method to subtract two float values
    public float subtract(float a, float b) {
        return a - b;
    }
}

// File: Operation/Maximum.java
package Operation;

public class Maximum {
    // Method to find the maximum of two integers
    public int max(int a, int b) {
        return (a > b) ? a : b;
    }
}

// File: TestOperation.java
import Operation.Addition;
import Operation.Maximum;

public class TestOperation {
    public static void main(String[] args) {
        // Create objects of Addition and Maximum classes
        Addition addition = new Addition();
        Maximum maximum = new Maximum();

        // Testing Addition methods
        int sum = addition.add(5, 10);
        float difference = addition.subtract(10.5f, 5.2f);
        System.out.println("Sum: " + sum);
        System.out.println("Difference: " + difference);

        // Testing Maximum method
        int max = maximum.max(15, 20);
        System.out.println("Maximum: " + max);
    }
}
```

```
javac Operation/Addition.java Operation/Maximum.java TestOperation.java

java TestOperation
```

Slip-21.

Q1) Define a class MyDate(Day, Month, year) with methods to accept and display a MyDateobject.
 Accept date as dd,mm,yyyy. Throw user defined exception "InvalidDateException" if the date
 is invalid.

```
// Custom exception for invalid date
class InvalidDateException extends Exception {
    public InvalidDateException(String message) {
        super(message);
    }
}

// MyDate class to represent a date
class MyDate {
    private int day;
    private int month;
    private int year;

    // Constructor
    public MyDate(int day, int month, int year) throws InvalidDateException {
        this.day = day;
        this.month = month;
        this.year = year;
        validateDate();
    }

    // Validate the date
    private void validateDate() throws InvalidDateException {
        if (month < 1 || month > 12) {
            throw new InvalidDateException("Month must be between 1 and 12.");
        }

        // Days in each month
        int[] daysInMonth = {31, 28 + (isLeapYear() && month == 2 ? 1 : 0), 31, 30, 31,
                              30, 31, 31, 30, 31, 30, 31};

        if (day < 1 || day > daysInMonth[month - 1]) {
            throw new InvalidDateException("Invalid day for the given month.");
        }
    }

    // Check if the year is a leap year
    private boolean isLeapYear() {
        return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
    }

    // Display the date
    public String display() {
        return String.format("%02d/%02d/%d", day, month, year);
    }

    // Main method for testing
    public static void main(String[] args) {
        try {
            MyDate date1 = new MyDate(29, 2, 2024);   // Valid leap year date
            System.out.println(date1.display());
        } catch (InvalidDateException e) {
            System.out.println(e.getMessage());
        }

        try {
            MyDate date2 = new MyDate(31, 4, 2021);   // Invalid date
            System.out.println(date2.display());
        } catch (InvalidDateException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

Q2) 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 member to count the number of Employee objects created
    private static int count = 0;

    // Default constructor
    public Employee() {
        this.id = 0;
        this.name = "Unknown";
        this.deptName = "Unknown";
        this.salary = 0.0;
        count++;
        displayCount();
    }

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

    // Method to display the count of objects created
    public static void displayCount() {
        System.out.println("Number of Employee objects created: " + count);
    }

    // Method to display the details of the Employee
    public void displayDetails() {
        System.out.printf("ID: %d, Name: %s, Department: %s, Salary: %.2f%n", id, name,
deptName, salary);
    }

    // Main method for testing
    public static void main(String[] args) {
        Employee emp1 = new Employee(1, "Alice", "HR", 50000);
        emp1.displayDetails();

        Employee emp2 = new Employee(2, "Bob", "IT", 60000);
        emp2.displayDetails();

        Employee emp3 = new Employee(); // Default constructor
        emp3.displayDetails();
    }
}
```

Slip-22.

Q1) 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 contain only
 the method printArea() that prints the area of the given shape. (use method overriding).

```
// Abstract class Shape
abstract class Shape {
    // Attributes for the dimensions of the shape
    protected int dimension1; // Could represent width for Rectangle, base for Triangle, radius for Circle
    protected int dimension2; // Could represent height for Rectangle, height for Triangle, unused for Circle

    // Constructor
    public Shape(int dimension1, int dimension2) {
        this.dimension1 = dimension1;
        this.dimension2 = dimension2;
    }

    // Abstract method to print area
    public abstract void printArea();
}

// Rectangle class that extends Shape
class Rectangle extends Shape {
    public Rectangle(int width, int height) {
        super(width, height);
    }

    @Override
    public void printArea() {
        int area = dimension1 * dimension2; // Area = width * height
        System.out.println("Area of Rectangle: " + area);
    }
}

// Triangle class that extends Shape
class Triangle extends Shape {
    public Triangle(int base, int height) {
        super(base, height);
    }

    @Override
    public void printArea() {
        double area = 0.5 * dimension1 * dimension2; // Area = 0.5 * base * height
        System.out.println("Area of Triangle: " + area);
    }
}

// Circle class that extends Shape
class Circle extends Shape {
    public Circle(int radius) {
        super(radius, 0); // dimension2 is unused for Circle
    }

    @Override
    public void printArea() {
        double area = Math.PI * dimension1 * dimension1; // Area = π * radius^2
        System.out.println("Area of Circle: " + area);
    }
}

// Main class to test the shapes
public class Main {
    public static void main(String[] args) {
        Shape rectangle = new Rectangle(5, 10);
        rectangle.printArea();

        Shape triangle = new Triangle(6, 8);
        triangle.printArea();

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

Q2) 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 javax.swing.*;
import java.awt.*;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;

public class MouseEventExample extends JFrame {
    private String eventName = "";

    public MouseEventExample() {
        setTitle("Mouse Event Example");
        setSize(400, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        // Add a MouseAdapter to handle mouse events
        addMouseListener(new MouseAdapter() {
            @Override
            public void mouseClicked(MouseEvent e) {
                eventName = "Mouse Clicked";
                repaint();
            }

            @Override
            public void mousePressed(MouseEvent e) {
                eventName = "Mouse Pressed";
                repaint();
            }

            @Override
            public void mouseReleased(MouseEvent e) {
                eventName = "Mouse Released";
                repaint();
            }

            @Override
            public void mouseEntered(MouseEvent e) {
                eventName = "Mouse Entered";
                repaint();
            }

            @Override
            public void mouseExited(MouseEvent e) {
                eventName = "Mouse Exited";
                repaint();
            }
        });
    }

    @Override
    public void paint(Graphics g) {
        super.paint(g);
        // Draw the event name at the center of the window
        g.setColor(Color.RED);
        g.setFont(new Font("Arial", Font.BOLD, 24));
        FontMetrics fm = g.getFontMetrics();
        int x = (getWidth() - fm.stringWidth(eventName)) / 2;
        int y = (getHeight() + fm.getHeight()) / 2;
        g.drawString(eventName, x, y);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            MouseEventExample frame = new MouseEventExample();
            frame.setVisible(true);
        });
    }
}
```

Slip-23.

Q1) 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.

```
public class MyNumber {
    private int value;

    // Default constructor
    public MyNumber() {
        this.value = 0; // Initialize to 0
    }

    // Constructor to initialize with a value
    public MyNumber(int value) {
        this.value = value; // Use this to refer to the instance variable
    }

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

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

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

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

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

    // Main method to create an object and test the methods
    public static void main(String[] args) {
        int inputValue;

        // Check if an argument is provided
        if (args.length > 0) {
            // Try to parse the first argument as an integer
            try {
                inputValue = Integer.parseInt(args[0]);
                MyNumber myNumber = new MyNumber(inputValue);

                // Display results
                System.out.println("Number: " + inputValue);
```

```
                    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("Please provide a valid integer as a command line
argument.");
                }
            } else {
                // No arguments provided
                System.out.println("Please provide an integer value as a command line
argument.");
            }
        }
}
```

```
javac MyNumber.java


java MyNumber -5.
```

Q2) Write a simple currency converter, as shown in the figure. User can enter the amount of
"Singapore Dollars", "US Dollars", or "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.event.ActionEvent;
import java.awt.event.ActionListener;

public class SwingCurrencyConverter extends JFrame {

    private JTextField sgdField;
    private JTextField usdField;
    private JTextField euroField;

    private static final double USD_TO_SGD = 1.41;
    private static final double USD_TO_EURO = 0.92;
    private static final double SGD_TO_EURO = 0.65;

    public SwingCurrencyConverter() {
        // Set up the frame
        setTitle("Currency Converter");
        setSize(300, 150);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLayout(null);

        // Singapore Dollar label and text field
        JLabel sgdLabel = new JLabel("Singapore Dollars");
        sgdLabel.setBounds(10, 10, 150, 25);
        add(sgdLabel);

        sgdField = new JTextField();
        sgdField.setBounds(150, 10, 100, 25);
        add(sgdField);
        // US Dollar label and text field
        JLabel usdLabel = new JLabel("US Dollars");
        usdLabel.setBounds(10, 40, 150, 25);
        add(usdLabel);

        usdField = new JTextField();
        usdField.setBounds(150, 40, 100, 25);
        add(usdField);

        // Euro label and text field
        JLabel euroLabel = new JLabel("Euros");
        euroLabel.setBounds(10, 70, 150, 25);
        add(euroLabel);

        euroField = new JTextField();
        euroField.setBounds(150, 70, 100, 25);
        add(euroField);

        // Add action listeners for Singapore Dollar field
        sgdField.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                convertFromSGD();
            }
        });

        // Add action listeners for US Dollar field
        usdField.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                convertFromUSD();
            }
        });

        // Add action listeners for Euro field
        euroField.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                convertFromEuro();
            }
        });
    }

    private void convertFromSGD() {
        try {
            double sgd = Double.parseDouble(sgdField.getText());
            double usd = sgd / USD_TO_SGD;
            double euro = sgd * SGD_TO_EURO;
            usdField.setText(String.format("%.2f", usd));
            euroField.setText(String.format("%.2f", euro));
        } catch (NumberFormatException ex) {
            usdField.setText("Invalid input");
            euroField.setText("Invalid input");
        }
    }
    private void convertFromUSD() {
        try {
            double usd = Double.parseDouble(usdField.getText());
            double sgd = usd * USD_TO_SGD;
            double euro = usd * USD_TO_EURO;
            sgdField.setText(String.format("%.2f", sgd));
            euroField.setText(String.format("%.2f", euro));
        } catch (NumberFormatException ex) {
            sgdField.setText("Invalid input");
            euroField.setText("Invalid input");
        }
    }

    private void convertFromEuro() {
        try {
            double euro = Double.parseDouble(euroField.getText());
            double usd = euro / USD_TO_EURO;
            double sgd = euro / SGD_TO_EURO;
            usdField.setText(String.format("%.2f", usd));
            sgdField.setText(String.format("%.2f", sgd));
        } catch (NumberFormatException ex) {
            usdField.setText("Invalid input");
            sgdField.setText("Invalid input");
        }
    }

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

Slip-24.

Q1) 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. 'BankA', 'BankB' and 'BankC'
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
abstract class Bank {
    // Abstract method
    abstract int getBalance();
}

// Subclass for Bank A
class BankA extends Bank {
    private int balance;

    public BankA() {
        this.balance = 100; // Rs.100 deposited
    }
    @Override
    int getBalance() {
        return balance;
    }
}

// Subclass for Bank B
class BankB extends Bank {
    private int balance;

    public BankB() {
        this.balance = 150; // Rs.150 deposited
    }

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

// Subclass for Bank C
class BankC extends Bank {
    private int balance;

    public BankC() {
        this.balance = 200; // Rs.200 deposited
    }

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

// Main class to test the functionality
public class Main {
    public static void main(String[] args) {
        Bank bankA = new BankA();
        Bank bankB = new BankB();
        Bank bankC = new BankC();

        // Calling getBalance method
        System.out.println("Balance in Bank A: " + bankA.getBalance());
        System.out.println("Balance in Bank B: " + bankB.getBalance());
        System.out.println("Balance in Bank C: " + bankC.getBalance());
    }
}
```

Q2) 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.MouseAdapter;
import java.awt.event.MouseEvent;

public class ConcentricCircles extends JFrame {
    private int x = -1;
    private int y = -1;

    public ConcentricCircles() {
        setTitle("Concentric Circles");
        setSize(400, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Add mouse listener to capture the click location
        addMouseListener(new MouseAdapter() {
            @Override
            public void mouseClicked(MouseEvent e) {
                // Update the center coordinates of the circles
                x = e.getX();
                y = e.getY();
                repaint(); // Repaint the frame with the new circles
            }
        });
    }

    @Override
    public void paint(Graphics g) {
        super.paint(g);
        if (x != -1 && y != -1) {
            Graphics2D g2 = (Graphics2D) g;

            // Draw the three concentric circles with decreasing sizes
            g2.setColor(Color.LIGHT_GRAY);
            g2.fillOval(x - 60, y - 60, 120, 120);   // Outer circle

            g2.setColor(Color.GRAY);
            g2.fillOval(x - 40, y - 40, 80, 80);     // Middle circle

            g2.setColor(Color.DARK_GRAY);
            g2.fillOval(x - 20, y - 20, 40, 40);     // Inner circle
        }
    }

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

Slip-25.

Q1) 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 {
	private String rollNo;
	private String name;
	private String className;
	private float percentage;

	// Constructor
	public Student(String rollNo, String name, String className, float percentage) {
		this.rollNo = rollNo;
		this.name = name;
		this.className = className;
		this.percentage = percentage;
	}

	// Method to display student information
	public void displayInfo() {
		System.out.println("Roll No: " + rollNo);
		System.out.println("Name: " + name);
		System.out.println("Class: " + className);
		System.out.println("Percentage: " + percentage);
	}
}

public class Main {
	public static void main(String[] args) {
		BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));

		try {
			// Reading student information from the console
			System.out.print("Enter Roll No: ");
			String rollNo = reader.readLine();

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

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

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

			// Creating a Student object
			Student student = new Student(rollNo, name, className, percentage);
```

```
            // Displaying student information
            System.out.println("\nStudent Information:");
            student.displayInfo();
        } catch (IOException e) {
            System.out.println("An error occurred while reading input: " + e.getMessage());
        } catch (NumberFormatException e) {
            System.out.println("Invalid percentage format. Please enter a valid number.");
        }
    }
}
```

Q2) Create the following GUI screen using appropriate layout manager. Accept the name, class,
hobbies from the user and display the selected options in a textbox.

Slip-26.

Q1) 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 {
    private int itemNumber;
    private String itemName;
    private double itemPrice;

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

    // Default constructor
    public Item() {
        objectCount++;
    }

    // Parameterized constructor
    public Item(int itemNumber, String itemName, double itemPrice) {
        this.itemNumber = itemNumber;
        this.itemName = itemName;
        this.itemPrice = itemPrice;
        objectCount++;
    }
    // Static method to get the object count
    public static int getObjectCount() {
        return objectCount;
    }

    // Method to display item details
    public void displayItem() {
        System.out.println("Item Number: " + itemNumber);
        System.out.println("Item Name: " + itemName);
        System.out.println("Item Price: " + itemPrice);
        System.out.println();
    }
}

public class Main {
    public static void main(String[] args) {
        // Creating objects using parameterized constructor
        Item item1 = new Item(101, "Laptop", 75000.00);
        System.out.println("Object Count after creating item1: " + Item.getObjectCount());
        item1.displayItem();

        Item item2 = new Item(102, "Smartphone", 30000.00);
        System.out.println("Object Count after creating item2: " + Item.getObjectCount());
        item2.displayItem();

        Item item3 = new Item(103, "Tablet", 25000.00);
        System.out.println("Object Count after creating item3: " + Item.getObjectCount());
        item3.displayItem();
    }
}
```

Q2) Define a class ‘Donor’ to store the below mentioned details of a blood donor. name, age,
address, contactnumber, bloodgroup, 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 donors’ details whose blood group is ‘A+ve’ and had not donated for the
recent six months.

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

class Donor implements Serializable {
    private String name;
    private int age;
    private String address;
    private String contactNumber;
    private String bloodGroup;
    private LocalDate dateOfLastDonation;
    // Constructor
    public Donor(String name, int age, String address, String contactNumber, String
bloodGroup, LocalDate dateOfLastDonation) {
        this.name = name;
        this.age = age;
        this.address = address;
        this.contactNumber = contactNumber;
        this.bloodGroup = bloodGroup;
        this.dateOfLastDonation = dateOfLastDonation;
    }

    // Getters
    public String getBloodGroup() {
        return bloodGroup;
    }

    public LocalDate getDateOfLastDonation() {
        return dateOfLastDonation;
    }

    // Method to display donor details
    public void displayDetails() {
        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("Date of Last Donation: " + dateOfLastDonation);
        System.out.println();
    }
}
```

```
import java.io.*;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.List;

public class Main {
    private static final String FILE_NAME = "donors.dat";

    public static void main(String[] args) {
        // Create some donor objects
        List<Donor> donors = new ArrayList<>();
        donors.add(new Donor("John Doe", 30, "123 Street, Pune", "9876543210", "A+ve",
LocalDate.now().minusMonths(8)));
        donors.add(new Donor("Jane Smith", 28, "456 Avenue, Pune", "8765432109",
"B+ve", LocalDate.now().minusMonths(3)));
        donors.add(new Donor("Alice Brown", 35, "789 Boulevard, Pune", "7654321098",
"A+ve", LocalDate.now().minusMonths(7)));

        // Write donors to a file
        writeDonorsToFile(donors);
        // Read donors from the file and display specific ones
        readAndDisplayDonors();
    }

    private static void writeDonorsToFile(List<Donor> donors) {
        try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(FILE_NAME))) {
            oos.writeObject(donors);
        } catch (IOException e) {
            System.err.println("Error writing to file: " + e.getMessage());
        }
    }

    private static void readAndDisplayDonors() {
        try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream(FILE_NAME))) {
            List<Donor> donors = (List<Donor>) ois.readObject();
            LocalDate sixMonthsAgo = LocalDate.now().minusMonths(6);

            System.out.println("Eligible Donors (A+ve and not donated in the last 6
months):");
            for (Donor donor : donors) {
                if ("A+ve".equals(donor.getBloodGroup()) &&
donor.getDateOfLastDonation().isBefore(sixMonthsAgo)) {
                    donor.displayDetails();
                }
            }
        } catch (IOException | ClassNotFoundException e) {
            System.err.println("Error reading from file: " + e.getMessage());
        }
    }
}
```

Slip-27.

Q1) 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, override the getSalary() method, which will return salary of manager by adding traveling allowance, house rent allowance etc.

```
class Employee {
    protected String name;
    protected double baseSalary;

    // Constructor
    public Employee(String name, double baseSalary) {
        this.name = name;
        this.baseSalary = baseSalary;
    }

    // Method to get salary
    public double getSalary() {
        return baseSalary;
    }
    // Method to display employee details
    public void displayDetails() {
        System.out.println("Employee Name: " + name);
        System.out.println("Base Salary: " + baseSalary);
    }
}


class Manager extends Employee {
    private double travelingAllowance;
    private double houseRentAllowance;

    // Constructor
    public Manager(String name, double baseSalary, double travelingAllowance, double
houseRentAllowance) {
        super(name, baseSalary);
        this.travelingAllowance = travelingAllowance;
        this.houseRentAllowance = houseRentAllowance;
    }

    // Overriding getSalary method
    @Override
    public double getSalary() {
        return baseSalary + travelingAllowance + houseRentAllowance;
    }

    // Method to display manager details
    @Override
    public void displayDetails() {
        super.displayDetails(); // Call the parent class method
        System.out.println("Traveling Allowance: " + travelingAllowance);
        System.out.println("House Rent Allowance: " + houseRentAllowance);
        System.out.println("Total Salary: " + getSalary());
    }
}


public class Main {
    public static void main(String[] args) {
        // Creating an Employee object
        Employee employee = new Employee("John Doe", 50000);
        System.out.println("Employee Details:");
        employee.displayDetails();
        System.out.println("Total Salary: " + employee.getSalary());

        System.out.println();

        // Creating a Manager object
        Manager manager = new Manager("Jane Smith", 70000, 10000, 15000);
        System.out.println("Manager Details:");
        manager.displayDetails();
    }
}
```

Q2) 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) {
        if (args.length == 0) {
            System.out.println("Please provide a file or directory path as a command line
argument.");
            return;
        }

        File file = new File(args[0]);

        if (file.exists()) {
            if (file.isDirectory()) {
                deleteTextFiles(file);
            } else if (file.isFile()) {
                displayFileDetails(file);
            }
        } else {
            System.out.println("The specified file or directory does not exist.");
        }
    }

    private static void deleteTextFiles(File directory) {
        File[] files = directory.listFiles();
        int deletedCount = 0;

        if (files != null && files.length > 0) {
            Scanner scanner = new Scanner(System.in);
            System.out.println("The following text files will be deleted from the directory:");
            for (File file : files) {
                if (file.isFile() && file.getName().endsWith(".txt")) {
                    System.out.println(file.getName());
                }
            }

            System.out.print("Do you want to delete these files? (yes/no): ");
            String confirmation = scanner.nextLine();

            if ("yes".equalsIgnoreCase(confirmation)) {
                for (File file : files) {
                    if (file.isFile() && file.getName().endsWith(".txt")) {
                        if (file.delete()) {
                            deletedCount++;
                            System.out.println(file.getName() + " deleted.");
                        } else {
```

```
                        System.out.println("Failed to delete " + file.getName());
                    }
                }
            }
        } else {
            System.out.println("No files were deleted.");
        }

        System.out.println("Total files deleted: " + deletedCount);
    } else {
        System.out.println("No text files found in the directory.");
    }
}

private static void displayFileDetails(File file) {
    System.out.println("File Details:");
    System.out.println("Name: " + file.getName());
    System.out.println("Absolute Path: " + file.getAbsolutePath());
    System.out.println("Readable: " + file.canRead());
    System.out.println("Writable: " + file.canWrite());
    System.out.println("Executable: " + file.canExecute());
    System.out.println("Size (in bytes): " + file.length());
    System.out.println("Last Modified: " + new java.util.Date(file.lastModified()));
}
}
```

```
javac FileDirectoryChecker.java
java FileDirectoryChecker /path/to/directory/or/file.
```

Slip-28.

Q1) 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 FileInfoChecker {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Prompt the user for a filename
        System.out.print("Enter the file name (with path if not in current directory): ");
        String fileName = scanner.nextLine();

        // Create a File object
        File file = new File(fileName);

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

                // Check if the file is readable
                System.out.println("Readable: " + file.canRead());

                // Check if the file is writable
                System.out.println("Writable: " + file.canWrite());

                // Determine the type of file
                if (file.isDirectory()) {
                    System.out.println("Type: Directory");
                } else {
                    System.out.println("Type: File");
                }

                // Get the length of the file in bytes
                System.out.println("Length (in bytes): " + file.length());
            } else {
                System.out.println("File exists: No");
            }

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

```
javac FileInfoChecker.java
java FileInfoChecker
```

```
Enter the file name (with path if not in current directory): example.txt
File exists: Yes
Readable: true
Writable: true
Type: File
Length (in bytes): 1024
```

Q2) 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. Hints: To display a floating-point number in a specific format (e.g., 1 decimal place), use the static method String.format(), which has the same form as printf(). For example, String.format("%.1f", 1.234) returns String "1.2".

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

public class SwingTemperatureConverter extends JFrame {

    private JTextField celsiusField;
    private JTextField fahrenheitField;

    public SwingTemperatureConverter() {
        // Set up the frame
        setTitle("Temperature Converter");
        setSize(300, 100);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLayout(null);

        // Create Celsius label and text field
        JLabel celsiusLabel = new JLabel("Celsius:");
        celsiusLabel.setBounds(10, 10, 80, 25);
        add(celsiusLabel);

        celsiusField = new JTextField();
        celsiusField.setBounds(100, 10, 150, 25);
        add(celsiusField);

        // Create Fahrenheit label and text field
        JLabel fahrenheitLabel = new JLabel("Fahrenheit:");
        fahrenheitLabel.setBounds(10, 40, 80, 25);
        add(fahrenheitLabel);

        fahrenheitField = new JTextField();
        fahrenheitField.setBounds(100, 40, 150, 25);
        add(fahrenheitField);

        // Add action listeners for Celsius field
        celsiusField.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                convertToFahrenheit();
            }
        });

        // Add action listeners for Fahrenheit field
        fahrenheitField.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                convertToCelsius();
            }
        });
    }

    private void convertToFahrenheit() {
        try {
            double celsius = Double.parseDouble(celsiusField.getText());
            double fahrenheit = celsius * 9 / 5 + 32;
            fahrenheitField.setText(String.format("%.1f", fahrenheit));
        } catch (NumberFormatException ex) {
            fahrenheitField.setText("Invalid input");
        }
    }

    private void convertToCelsius() {
        try {
            double fahrenheit = Double.parseDouble(fahrenheitField.getText());
            double celsius = (fahrenheit - 32) * 5 / 9;
            celsiusField.setText(String.format("%.1f", celsius));
        } catch (NumberFormatException ex) {
            celsiusField.setText("Invalid input");
        }
    }
    public static void main(String[] args) {
        SwingTemperatureConverter frame = new SwingTemperatureConverter();
        frame.setVisible(true);
    }
}
```

Slip-29.

Q1) 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.List;

class Customer {
    private int custno;
    private String custname;
    private String contactnumber;
    private String custaddr;

    public Customer(int custno, String custname, String contactnumber, String custaddr) {
        this.custno = custno;
        this.custname = custname;
        this.contactnumber = contactnumber;
        this.custaddr = custaddr;
    }

    public String getContactNumber() {
        return contactnumber;
    }

    public void displayDetails() {
        System.out.println("Customer No: " + custno);
        System.out.println("Customer Name: " + custname);
        System.out.println("Contact Number: " + contactnumber);
        System.out.println("Customer Address: " + custaddr);
    }
}

class CustomerDatabase {
    private List<Customer> customers;

    public CustomerDatabase() {
        customers = new ArrayList<>();
    }

    public void addCustomer(Customer customer) {
        customers.add(customer);
    }

    public void searchByContact(String contactNumber) {
        for (Customer customer : customers) {
            if (customer.getContactNumber().equals(contactNumber)) {
                customer.displayDetails();
                return;
            }
        }
        System.out.println("Customer not found.");
    }
}

public class Main {
    public static void main(String[] args) {
        // Create a customer database
        CustomerDatabase db = new CustomerDatabase();

        // Add some customers
        db.addCustomer(new Customer(1, "Alice Johnson", "1234567890", "123 Elm St"));
        db.addCustomer(new Customer(2, "Bob Smith", "0987654321", "456 Oak St"));

        // Search for a customer by contact number
        String searchContact = "1234567890";
        System.out.println("Searching for customer with contact number: " +
searchContact);
        db.searchByContact(searchContact);

        searchContact = "0000000000";
        System.out.println("Searching for customer with contact number: " +
searchContact);
        db.searchByContact(searchContact);
    }
}
```

Q2) 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.List;
import java.util.Scanner;

class Vehicle {
    protected String company;
    protected double price;

    public Vehicle(String company, double price) {
        this.company = company;
        this.price = price;
    }

    public void displayInfo() {
        System.out.println("Company: " + company);
        System.out.println("Price: $" + price);
    }
```

```java
}

class LightMotorVehicle extends Vehicle {
    private double mileage;

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

    @Override
    public void displayInfo() {
        super.displayInfo();
        System.out.println("Mileage: " + mileage + " km/l");
    }
}

class HeavyMotorVehicle extends Vehicle {
    private double capacityInTons;

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

    @Override
    public void displayInfo() {
        super.displayInfo();
        System.out.println("Capacity: " + capacityInTons + " tons");
    }
}

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

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

        for (int i = 0; i < n; i++) {
            System.out.println("Enter details for vehicle " + (i + 1) + ":");
            System.out.print("Enter type of vehicle (1 for Light, 2 for Heavy): ");
            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) {
                System.out.print("Enter mileage: ");
                double mileage = scanner.nextDouble();
                vehicles.add(new LightMotorVehicle(company, price, mileage));
            } else if (type == 2) {
                System.out.print("Enter capacity in tons: ");
                double capacityInTons = scanner.nextDouble();
                vehicles.add(new HeavyMotorVehicle(company, price, capacityInTons));
            } else {
                System.out.println("Invalid type. Please enter 1 for Light or 2 for Heavy.");
                i--; // Decrement i to repeat this iteration
            }
            scanner.nextLine(); // Consume newline
        }

        System.out.println("\nVehicle Information:");
        for (Vehicle vehicle : vehicles) {
            vehicle.displayInfo();
            System.out.println("-------------------------");
        }

        scanner.close();
    }
}
```

Slip-30.

Q1) 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).

```
import java.util.Scanner;

class Person {
    private String personName;
    private String aadharNo;
    private String panNo;

    // Constructor to initialize Person object
    public Person(String personName, String aadharNo, String panNo) {
        this.personName = personName;
        this.aadharNo = aadharNo;
        this.panNo = panNo;
    }

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

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Person[] persons = new Person[5];

        // Accepting information for 5 persons
        for (int i = 0; i < 5; i++) {
            System.out.println("Enter details for Person " + (i + 1) + ":");

            System.out.print("Enter 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 information for each person
        System.out.println("\nPerson Information:");
        for (Person person : persons) {
            person.displayInfo();
        }

        scanner.close();
    }
}
```

Q2) Write a program that creates a user interface to perform integer divisions. The user enters two
 numbers in the text fields, Number1 and Number2. The division of Number1 and Number2 is
 displayed in the Result field when the Divide button is clicked. If Number1 or Number2 were
 not an integer, the program would throw a NumberFormatException. If Number2 were Zero,
 the program would throw an Arithmetic Exception Display the exception in a message dialog box.

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

public class DivisionCalculator {
    public static void main(String[] args) {
        // Create the frame
        JFrame frame = new JFrame("Integer Division Calculator");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(300, 200);
        frame.setLayout(null);

        // Create components
        JLabel label1 = new JLabel("Number 1:");
        label1.setBounds(20, 20, 80, 25);
        JTextField number1Field = new JTextField();
        number1Field.setBounds(100, 20, 160, 25);
        JLabel label2 = new JLabel("Number 2:");
        label2.setBounds(20, 60, 80, 25);
        JTextField number2Field = new JTextField();
        number2Field.setBounds(100, 60, 160, 25);

        JButton divideButton = new JButton("Divide");
        divideButton.setBounds(100, 100, 80, 25);
        JTextField resultField = new JTextField();
        resultField.setBounds(100, 140, 160, 25);
        resultField.setEditable(false);

        // Add components to frame
        frame.add(label1);
        frame.add(number1Field);
        frame.add(label2);
        frame.add(number2Field);
        frame.add(divideButton);
        frame.add(resultField);

        // Action listener for the Divide button
        divideButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                try {
                    String num1Str = number1Field.getText();
                    String num2Str = number2Field.getText();

                    // Parse integers
                    int num1 = Integer.parseInt(num1Str);
                    int num2 = Integer.parseInt(num2Str);

                    // Perform division
                    if (num2 == 0) {
                        throw new ArithmeticException("Division by zero is not
allowed.");
                    }
                    int result = num1 / num2;

                    // Display result
                    resultField.setText(String.valueOf(result));
                } catch (NumberFormatException ex) {
                    JOptionPane.showMessageDialog(frame, "Please enter valid
integers.", "Input Error", JOptionPane.ERROR_MESSAGE);
                } catch (ArithmeticException ex) {
                    JOptionPane.showMessageDialog(frame, ex.getMessage(), "Math
Error", JOptionPane.ERROR_MESSAGE);
                }
            }
        });

        // Make the frame visible
        frame.setVisible(true);
    }
}
```