Slip no-1

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

```java
public class PrimeNumbers {
    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 number.");
            }
        }
    }

    // Method to check if a number is prime
    public static boolean isPrime(int number) {
        if (number <= 1) {
            return false; // 0 and 1 are not prime numbers
        }
        for (int i = 2; i <= Math.sqrt(number); i++) {
            if (number % i == 0) {
                return false; // Found a divisor, not prime
            }
        }
        return true; // No divisors found, it is prime
    }
}
```

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.

```java
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 display details
    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 the constructor of the abstract class
        this.department = department;
    }

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

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

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

        OfficeStaff[] staffArray = new OfficeStaff[n];

        // Input details for each OfficeStaff
        for (int i = 0; i < n; i++) {
            System.out.println("Enter details for Office Staff " + (i +
1));

            System.out.print("ID: ");
            int id = scanner.nextInt();
            scanner.nextLine(); // Consume the newline character
            System.out.print("Name: ");
            String name = scanner.nextLine();
            System.out.print("Department: ");
            String department = scanner.nextLine();

            staffArray[i] = new OfficeStaff(id, name, department);
        }

        // Display details of all Office Staff
        System.out.println("\nDetails of Office Staff:");
        for (OfficeStaff staff : staffArray) {
            staff.displayDetails();
        }

        scanner.close();
    }
}
```

Slip no-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)

```java
public class BMICalculator {
    public static void main(String[] args) {
        if (args.length != 4) {
            System.out.println("Usage: java BMICalculator <First Name>
<Last Name> <Weight (in kg)> <Height (in meters)>");
            return;
        }

        // Read command line arguments
        String firstName = args[0];
        String lastName = args[1];
        double weight = Double.parseDouble(args[2]);  // weight in
kilograms
        double height = Double.parseDouble(args[3]);  // height in meters

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

        // Display result
        System.out.println("Person: " + firstName + " " + lastName);
        System.out.println("Weight: " + weight + " kg");
        System.out.println("Height: " + height + " meters");
        System.out.println("BMI Index: " + String.format("%.2f", bmi));

        // Optional: Provide a basic interpretation of the BMI
        if (bmi < 18.5) {
            System.out.println("Underweight");
        } else if (bmi >= 18.5 && bmi < 24.9) {
            System.out.println("Normal weight");
        } else if (bmi >= 25 && bmi < 29.9) {
            System.out.println("Overweight");
        } else {
            System.out.println("Obese");
        }
    }
}
```

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.Scanner;

class CricketPlayer {
    String name;
    int noOfInnings;
    int noOfTimesNotOut;
    int totalRuns;
    double batAvg;

    // Constructor to initialize the player data
    public CricketPlayer(String name, int noOfInnings, int
noOfTimesNotOut, int totalRuns) {
        this.name = name;
        this.noOfInnings = noOfInnings;
        this.noOfTimesNotOut = noOfTimesNotOut;
        this.totalRuns = totalRuns;
        this.batAvg = 0.0;
    }

    // Static method to calculate the batting average
    public static void avg(CricketPlayer player) {
        if (player.noOfInnings - player.noOfTimesNotOut > 0) {
            player.batAvg = (double) player.totalRuns /
(player.noOfInnings - player.noOfTimesNotOut);
        } else {
            player.batAvg = player.totalRuns;  // if no dismissals,
average is total runs
        }
    }

    // Static method to sort players based on batting average
    public static void sort(CricketPlayer[] players) {
        Arrays.sort(players, (p1, p2) -> Double.compare(p2.batAvg,
p1.batAvg)); // Sorting in descending order of average
    }

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

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

        // Get the number of players
        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.println("Enter details for player " + (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

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

        // Calculate the batting average for each player
        for (CricketPlayer player : players) {
            CricketPlayer.avg(player);
        }

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

        // Display the sorted player details
        System.out.println("\nPlayer details in sorted order (based on
batting average):");
        for (CricketPlayer player : players) {
            player.display();
        }

        scanner.close();
    }
}
```

Slip no-3

Q1. Write a program to accept 'n' name of cities from the user and sort them in ascending order.

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

public class SortCities {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        // Get the number of cities from the user
        System.out.print("Enter the number of cities: ");
        int n = scanner.nextInt();
        scanner.nextLine(); // consume the newline character

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

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

        // Sort the city names 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.

```
class Patient {
    String patientName;
    int patientAge;
    double patientOxyLevel;
    double patientHRCTReport;
```

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

    public void checkPatientStatus() throws CovidPositiveException {
        if (patientOxyLevel < 95 && patientHRCTReport > 10) {
            throw new CovidPositiveException("Patient is Covid Positive(+)
and Need to Hospitalized");
        } else {
            displayPatientInfo();
        }
    }

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

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

public class Main {
    public static void main(String[] args) {
        Patient patient = new Patient("John Doe", 45, 93.5, 12.0);

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

Slip no-4

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

```java
import java.util.Scanner;

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

        // Input 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();

        // Create a 2D array
        int[][] matrix = new int[rows][cols];

        // Input the elements of the matrix
        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();
            }
        }

        // Transpose the matrix (swap rows with columns)
        int[][] transpose = new int[cols][rows];
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                transpose[j][i] = matrix[i][j];
            }
        }
        // Display the transposed matrix
        System.out.println("Transposed matrix:");
        for (int i = 0; i < cols; i++) {
            for (int j = 0; j < rows; j++) {
                System.out.print(transpose[i][j] + " ");
            }
            System.out.println();
        }

        scanner.close();
    }
}
```

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

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

class LoginScreenAWT extends Frame implements ActionListener {
    Label userLabel, passLabel, messageLabel;
    TextField userField;
    TextField passField;
    Button loginButton, clearButton;
    int attempts = 0;

    public LoginScreenAWT() {
        setTitle("Login Screen");
        setSize(400, 200);
        setLayout(new GridLayout(4, 2));

        userLabel = new Label("Username:");
        passLabel = new Label("Password:");

        userField = new TextField();
        passField = new TextField();
        passField.setEchoChar('*');
        loginButton = new Button("Login");
        clearButton = new Button("Clear");

        messageLabel = new Label();

        add(userLabel);
        add(userField);
        add(passLabel);
        add(passField);
        add(loginButton);
        add(clearButton);
        add(messageLabel);

        loginButton.addActionListener(this);
        clearButton.addActionListener(this);

        setVisible(true);
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent we) {
                dispose();
            }
        });
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == clearButton) {
            userField.setText("");
            passField.setText("");
            messageLabel.setText("");
        } else if (e.getSource() == loginButton) {
            String username = userField.getText();
            String password = passField.getText();

            if (attempts < 3) {
                if (username.equals(password)) {
                    messageLabel.setText("Login Successful!");
                } else {
                    attempts++;
                    messageLabel.setText("Incorrect Login. Attempts left:
" + (3 - attempts));
                    if (attempts >= 3) {
                        messageLabel.setText("Login Failed. No more
attempts.");
                        loginButton.setEnabled(false);
                    }
                }
            }
        }
    }

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

Slip no-5

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

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

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

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

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

```java
}

// Derived class State
class State extends Country {
    protected String stateName;
    protected 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: " + countryName);
        System.out.println("Continent: " + continentName);
    }
}

// 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 ic matrices:
 Addition
Multiplication

Exit

```java
import java.util.Scanner;

public class MatrixOperations {
    // Method to input a matrix
    public static int[][] inputMatrix(int rows, int cols) {
        Scanner scanner = new Scanner(System.in);
        int[][] matrix = new int[rows][cols];

        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();
            }
        }
        return matrix;
    }

    // Method to add two matrices
    public static int[][] addMatrices(int[][] matrixA, int[][] matrixB) {
        int rows = matrixA.length;
        int cols = matrixA[0].length;
        int[][] result = new int[rows][cols];

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

    // Method to multiply two matrices
    public static int[][] multiplyMatrices(int[][] matrixA, int[][]
matrixB) {
        int rowsA = matrixA.length;
        int colsA = matrixA[0].length;
        int rowsB = matrixB.length;
        int colsB = matrixB[0].length;
        if (colsA != rowsB) {
            throw new IllegalArgumentException("Matrices cannot be
multiplied: incompatible dimensions.");
        }

        int[][] result = new int[rowsA][colsB];

        for (int i = 0; i < rowsA; i++) {
            for (int j = 0; j < colsB; j++) {
                for (int k = 0; k < colsA; k++) {
                    result[i][j] += matrixA[i][k] * matrixB[k][j];
                }
            }
        }
        return result;
    }

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

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

        do {
            System.out.println("\nMenu:");
            System.out.println("1. Addition of two matrices");
            System.out.println("2. Multiplication of two matrices");
            System.out.println("3. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();
```

```java
            switch (choice) {
                case 1: {
                    System.out.print("Enter the number of rows: ");
                    int rows = scanner.nextInt();
                    System.out.print("Enter the number of columns: ");
                    int cols = scanner.nextInt();

                    System.out.println("Input first matrix:");
                    int[][] matrixA = inputMatrix(rows, cols);
                    System.out.println("Input second matrix:");
                    int[][] matrixB = inputMatrix(rows, cols);

                    int[][] sum = addMatrices(matrixA, matrixB);
                    System.out.println("Sum of the matrices:");
                    printMatrix(sum);
                    break;
                }
                case 2: {
                    System.out.print("Enter the number of rows for first
matrix: ");
                    int rowsA = scanner.nextInt();
                    System.out.print("Enter the number of columns for
first matrix: ");
                    int colsA = scanner.nextInt();

                    System.out.print("Enter the number of rows for second
matrix: ");
                    int rowsB = scanner.nextInt();
                    System.out.print("Enter the number of columns for
second matrix: ");
                    int colsB = scanner.nextInt();

                    if (colsA != rowsB) {
                        System.out.println("Matrices cannot be multiplied:
incompatible dimensions.");
                        break;
                    }

                    System.out.println("Input first matrix:");
                    int[][] matrixA = inputMatrix(rowsA, colsA);
                    System.out.println("Input second matrix:");
                    int[][] matrixB = inputMatrix(rowsB, colsB);

                    int[][] product = multiplyMatrices(matrixA, matrixB);
                    System.out.println("Product of the matrices:");
                    printMatrix(product);
                    break;
                }
                case 3:
                    System.out.println("Exiting...");
                    break;
                default:
                    System.out.println("Invalid choice! Please try
again.");
            }
        } while (choice != 3);

        scanner.close();
    }
}
```

Slip no-6

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

```
public class PrimeNumbers {
    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 number.");
            }
        }
    }

    // Method to check if a number is prime
    public static boolean isPrime(int number) {
        if (number <= 1) {
            return false; // 0 and 1 are not prime numbers
        }
        for (int i = 2; i <= Math.sqrt(number); i++) {
            if (number % i == 0) {
                return false; // Found a divisor, not prime
            }
        }
        return true; // No divisors found, it is prime
    }
}
```

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

```
import java.util.Scanner;

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

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

    // Abstract methods to be implemented by subclasses
```

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

// Subclass PurchaseOrder
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 (ID: " +
id + "): ");
        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);
        System.out.println();
    }
}

// Subclass SalesOrder
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 (ID: " + id +
"): ");
        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);
        System.out.println();
    }
}

// Main class
public class OrderTest {
    public static void main(String[] args) {
        // Create arrays for PurchaseOrder and SalesOrder
        PurchaseOrder[] purchaseOrders = new PurchaseOrder[3];
        SalesOrder[] salesOrders = new SalesOrder[3];

        // Accept details for Purchase Orders
        for (int i = 0; i < 3; i++) {
            purchaseOrders[i] = new PurchaseOrder(i + 1, "Purchase Order
#" + (i + 1));
            purchaseOrders[i].acceptDetails();
        }

        // Accept details for Sales Orders
        for (int i = 0; i < 3; i++) {
            salesOrders[i] = new SalesOrder(i + 1, "Sales Order #" + (i +
1));
            salesOrders[i].acceptDetails();
        }

        // Display details of Purchase Orders
        System.out.println("\nPurchase Orders Details:");
        for (PurchaseOrder po : purchaseOrders) {
            po.displayDetails();
        }
        // Display details of Sales Orders
        System.out.println("Sales Orders Details:");
        for (SalesOrder so : salesOrders) {
            so.displayDetails();
        }
    }
}
```

Slip no-7

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

a. Deposit a certain amount into an account

b. Withdraw a certain amount from an account

c. Return a Balance value specifying the amount with details

```
import java.util.Scanner;

// BankAccount class
class BankAccount {
    private String accountHolder;
    private double balance;

    public BankAccount(String accountHolder) {
        this.accountHolder = accountHolder;
        this.balance = 0.0; // Initial balance is zero
    }

    public void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
            System.out.println("Deposited: $" + amount);
        } else {
            System.out.println("Deposit amount must be positive.");
        }
    }
    public void withdraw(double amount) {
        if (amount > 0 && amount <= balance) {
            balance -= amount;
            System.out.println("Withdrawn: $" + amount);
        } else if (amount > balance) {
            System.out.println("Insufficient funds for withdrawal.");
        } else {
            System.out.println("Withdrawal amount must be positive.");
        }
    }

    public double getBalance() {
        return balance;
    }

    public String getAccountHolder() {
        return accountHolder;
    }
}

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

        // Create a bank account
        System.out.print("Enter account holder name: ");
        String accountHolderName = scanner.nextLine();
        BankAccount account = new BankAccount(accountHolderName);

        while (true) {
            System.out.println("\nBank Operations Menu:");
            System.out.println("1. Deposit");
            System.out.println("2. Withdraw");
            System.out.println("3. Check Balance");
            System.out.println("4. Exit");
            System.out.print("Select an option: ");
            int choice = scanner.nextInt();
            switch (choice) {
                case 1:
                    System.out.print("Enter amount to deposit: $");
                    double depositAmount = scanner.nextDouble();
                    account.deposit(depositAmount);
                    break;

                case 2:
                    System.out.print("Enter amount to withdraw: $");
                    double withdrawAmount = scanner.nextDouble();
                    account.withdraw(withdrawAmount);
                    break;

                case 3:
                    System.out.println("Current Balance for " +
account.getAccountHolder() + ": $" + account.getBalance());
                    break;

                case 4:
                    System.out.println("Exiting...");
                    scanner.close();
                    return;

                default:
                    System.out.println("Invalid option. Please try
again.");
            }
        }
    }
}
```

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;
```

```java
public class ReverseCaseFileContent {
    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();

        StringBuilder content = new StringBuilder();

        // Reading the file and storing the content
        try (BufferedReader br = new BufferedReader(new
FileReader(filePath))) {
            String line;
            while ((line = br.readLine()) != null) {
                content.append(line).append("\n");
            }
        } catch (IOException e) {
            System.out.println("Error reading the file: " +
e.getMessage());
            return;
        }

        // Displaying the content in reverse order with changed case
        String reversedContent = content.reverse().toString();
        String result = changeCase(reversedContent);

        System.out.println("Contents in reverse order and changed
case:\n");
        System.out.println(result);

        scanner.close();
    }

    // Method to change the case of each character
    private static String changeCase(String str) {
        StringBuilder changedCase = new StringBuilder();
        for (char c : str.toCharArray()) {
            if (Character.isLowerCase(c)) {
                changedCase.append(Character.toUpperCase(c));
            } else if (Character.isUpperCase(c)) {
                changedCase.append(Character.toLowerCase(c));
            } else {
                changedCase.append(c); // Non-alphabetic characters remain
unchanged
            }
        }
        return changedCase.toString();
    }
}
```

Slip no-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))

```
import java.util.Scanner;

public class Sphere {
    private double radius;

    // Constructor to initialize the radius
    public Sphere(double radius) {
        this.radius = radius;
    }

    // Method to calculate surface area
    public double surfaceArea() {
        return 4 * 3.14 * (radius * radius);
    }

    // Method to calculate volume
    public double volume() {
        return (4.0 / 3.0) * 3.14 * (radius * radius * radius);
    }

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

        // Input the radius
        System.out.print("Enter the radius of the sphere: ");
        double radius = scanner.nextDouble();

        // Create a Sphere object
        Sphere sphere = new Sphere(radius);

        // Display surface area and volume
        System.out.printf("Surface Area: %.2f%n", sphere.surfaceArea());
        System.out.printf("Volume: %.2f%n", sphere.volume());

        scanner.close();
    }
}
```

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

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

public class MouseEventExample extends JFrame {
    private JTextField textField;

    public MouseEventExample() {
        // Set up the frame
        setTitle("Mouse Events Example");
        setSize(400, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(null);  // Using null layout for manual positioning

        // Create a text field
        textField = new JTextField();
        textField.setBounds(50, 200, 300, 30); // Set position and size
        add(textField);

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

        addMouseMotionListener(new MouseAdapter() {
            @Override
            public void mouseMoved(MouseEvent e) {
                // Optionally display mouse position while moving
                int x = e.getX();
                int y = e.getY();
                textField.setText("Mouse Moved at: (" + x + ", " + y +
")");
            }
        });
    }

    public static void main(String[] args) {
        // Create and display the frame
        MouseEventExample example = new MouseEventExample();
        example.setVisible(true);
    }
}
```

Slip no-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 String period; // "AM" or "PM"

    // Constructor
    public Clock(int hours, int minutes, int seconds) {
        if (isValidTime(hours, minutes, seconds)) {
            this.hours = hours;
            this.minutes = minutes;
            this.seconds = seconds;
            this.period = (hours < 12) ? "AM" : "PM"; // Set AM/PM based
on hours
        } else {
            throw new IllegalArgumentException("Invalid time provided.");
        }
    }

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

    // Method to set the time
    public void setTime(int hours, int minutes, int seconds) {
        if (isValidTime(hours, minutes, seconds)) {
            this.hours = hours;
            this.minutes = minutes;
            this.seconds = seconds;
            this.period = (hours < 12) ? "AM" : "PM"; // Update AM/PM
based on new hours
        } else {
            throw new IllegalArgumentException("Invalid time provided.");
        }
    }

    // Method to display the time in AM/PM format
    public String displayTime() {
        int displayHours = (hours % 12 == 0) ? 12 : hours % 12; // Convert
0 to 12 for display
        return String.format("%02d:%02d:%02d %s", displayHours, minutes,
seconds, period);
    }

    // Main method for testing
    public static void main(String[] args) {
        try {
            // Create a Clock instance
            Clock clock = new Clock(14, 30, 45); // 2:30:45 PM
            System.out.println("Current Time: " + clock.displayTime());

            // Set a new time
            clock.setTime(10, 15, 20); // 10:15:20 AM
            System.out.println("Updated Time: " + clock.displayTime());
        } 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 objects of class product and display the contents of each object and Also display the object count.

```
// Marker interface
interface ProductMarker {
    // This is a marker interface with no methods
}

// Product class implementing the marker interface
class Product implements ProductMarker {
    private static int objectCount = 0; // Static variable to count
objects
    private int productId;
    private String productName;
    private double productCost;
```

```java
    private int productQuantity;

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

    // 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++; // Increment object count
    }

    // Method to display product details
    public void displayProductDetails() {
        System.out.printf("Product ID: %d, Product Name: %s, Product Cost:
%.2f, Product Quantity: %d%n",
                          productId, productName, productCost,
productQuantity);
    }

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

    // Main method for testing
    public static void main(String[] args) {
        // Creating Product objects
        Product product1 = new Product(101, "Laptop", 750.50, 5);
        Product product2 = new Product(102, "Smartphone", 300.75, 10);
        Product product3 = new Product(); // Using default constructor
        // Displaying product details
        product1.displayProductDetails();
        product2.displayProductDetails();
        product3.displayProductDetails();

        // Displaying the object count
        System.out.println("Total Product Objects Created: " +
Product.getObjectCount());
    }
}
```

Slip no-10

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

```
// Functional interface
@FunctionalInterface
interface CubeCalculator {
    int calculateCube(int number);
}

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

        // Test the cube calculation
        int number = 3; // You can change this value to test with
different numbers
        int cube = cubeCalculator.calculateCube(number);

        // Display the result
        System.out.printf("The cube of %d is: %d%n", number, cube);
    }
}
```

Q2) Write a program to create a package name student. Define class StudentInfo with method to display information about student such as follno, 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

```java
package student;

import java.util.Scanner;

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

        // Accepting student details from the user
        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 studentClass = scanner.nextLine();

        double totalMarks = 0;
        int subjects = 6;

        // Accept marks for 6 subjects
        for (int i = 1; i <= subjects; i++) {
            System.out.print("Enter marks for subject " + i + ": ");
            totalMarks += scanner.nextDouble();
        }

        // Calculate percentage
        double percentage = (totalMarks / (subjects * 100)) * 100;

        // Create a StudentInfo object
        StudentInfo student = new StudentInfo(rollNo, name, studentClass,
percentage);
```

```
        // Display student information
        student.displayInfo();

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

Slip no-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.

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

// Class Cylinder implementing Operation interface
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
    }
}

// Main class to test the Cylinder class
public class CylinderVolumeCalculator {
    public static void main(String[] args) {
        // Create a Cylinder object with specific radius and height
        Cylinder cylinder = new Cylinder(5.0, 10.0); // Example values:
radius = 5, height = 10

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

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.

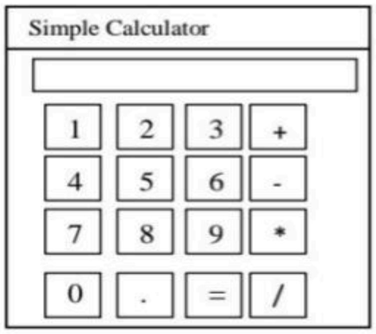
```
import java.util.Scanner;

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

        // Accepting username and password from the user
        System.out.print("Enter Username: ");
        String username = scanner.nextLine();

        System.out.print("Enter Password: ");
        String password = scanner.nextLine();

        // Checking if username and password are the same
        if (!username.equals(password)) {
            // Raise "Invalid Password" message
            System.out.println("Invalid Password: The username and
password cannot be the same.");
        } else {
            // Successful message (optional)
            System.out.println("Access Granted: Welcome " + username +
"!");
        }
        // Close the scanner
        scanner.close();
    }
}
```

Slip no-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 College
class College {
    protected int cno;        // College number
    protected String cname;    // College name
    protected String caddr;    // College address

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

    // Method to display college details
    public void displayCollegeDetails() {
        System.out.println("College Number: " + cno);
        System.out.println("College Name: " + cname);
        System.out.println("College Address: " + caddr);
    }
}

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

    // Constructor for Department
    public Department(int cno, String cname, String caddr, int dno, String
dname) {
        super(cno, cname, caddr); // Call to parent constructor
        this.dno = dno;
        this.dname = dname;
    }

    // Method to display department details
    public void displayDepartmentDetails() {
        // Display college details
        displayCollegeDetails();
        // Display department details
        System.out.println("Department Number: " + dno);
        System.out.println("Department Name: " + dname);
    }
}

// Main class to test the implementation
public class CollegeDepartmentTest {
    public static void main(String[] args) {
        // Create a Department object
        Department department = new Department(101, "ABC College", "123
Main St", 1, "Computer Science");

        // Display college and department details
        department.displayDepartmentDetails();
    }
}
```

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.*;
```

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

public class SimpleCalculator implements ActionListener {

    // Frame
    JFrame frame;

    // Textfield to display the result
    JTextField textField;

    // Number buttons and function buttons
    JButton[] numberButtons = new JButton[10];
    JButton[] functionButtons = new JButton[8];

    // Function buttons: add, subtract, multiply, divide, decimal, equals,
delete, clear
    JButton addButton, subButton, mulButton, divButton;
    JButton decButton, equButton, delButton, clrButton;

    // Panel for grid layout
    JPanel panel;

    // Variables to store values and operations
    double num1 = 0, num2 = 0, result = 0;
    char operator;

    public SimpleCalculator() {
        // Create the frame
        frame = new JFrame("Simple Calculator");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(420, 550);
        frame.setLayout(null);

        // Create the text field to display results
        textField = new JTextField();
        textField.setBounds(50, 25, 300, 50);
        textField.setEditable(false);
```

```
// Create the function 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");

// Add function buttons to the 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;

// Add action listeners to function buttons
for (int i = 0; i < 8; i++) {
    functionButtons[i].addActionListener(this);
}

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

// Panel for the grid layout
panel = new JPanel();
panel.setBounds(50, 100, 300, 300);
panel.setLayout(new GridLayout(4, 4, 10, 10));

// Add buttons to the 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 components to the frame
        frame.add(panel);
        frame.add(textField);
        frame.setVisible(true);
    }

    // Handling button clicks
    public void actionPerformed(ActionEvent e) {
        for (int i = 0; i < 10; i++) {
            if (e.getSource() == numberButtons[i]) {

textField.setText(textField.getText().concat(String.valueOf(i)));
            }
        }
        if (e.getSource() == decButton) {
            textField.setText(textField.getText().concat("."));
        }
        if (e.getSource() == addButton) {
            num1 = Double.parseDouble(textField.getText());
            operator = '+';
            textField.setText("");
        }
        if (e.getSource() == subButton) {
            num1 = Double.parseDouble(textField.getText());
            operator = '-';
            textField.setText("");
        }
        if (e.getSource() == mulButton) {
            num1 = Double.parseDouble(textField.getText());
            operator = '*';
            textField.setText("");
        }
        if (e.getSource() == divButton) {
            num1 = Double.parseDouble(textField.getText());
            operator = '/';
            textField.setText("");
        }
        if (e.getSource() == equButton) {
            num2 = Double.parseDouble(textField.getText());

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

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

Slip no-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) {
        // Check if a file name is provided
        if (args.length == 0) {
            System.out.println("Please provide a file name as a command
line argument.");
            return;
        }

        // Get the file name from command line argument
        String fileName = args[0];
        File file = new File(fileName);

        // Check if the file exists
        if (!file.exists()) {
            System.out.println("The file '" + fileName + "' does not
exist.");
            return;
        }

        // Initialize counters for lines and words
        int lineCount = 0;
        int wordCount = 0;

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

        // Display the results
        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.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Date;

public class DateTimeFormats {
    public static void main(String[] args) {
        // Get the current date and time
        LocalDateTime now = LocalDateTime.now();

        // Define formatters for different formats
        DateTimeFormatter format1 =
DateTimeFormatter.ofPattern("dd/MM/yyyy");
        DateTimeFormatter format2 =
DateTimeFormatter.ofPattern("MM-dd-yyyy");
        DateTimeFormatter format3 = DateTimeFormatter.ofPattern("EEEE MMMM
dd yyyy");
        DateTimeFormatter format4 = DateTimeFormatter.ofPattern("EEE MMMM
dd HH:mm:ss zzz yyyy");
        DateTimeFormatter format5 = DateTimeFormatter.ofPattern("dd/MM/yy
hh:mm:ss a Z");

        // Display the current date and time in various formats
        System.out.println("Current date is: " + now.format(format1));
        System.out.println("Current date is: " + now.format(format2));
        System.out.println("Current date is: " + now.format(format3));
        System.out.println("Current date and time is: " +
now.format(format4));
        System.out.println("Current date and time is: " +
now.format(format5));
    }
}
```

Slip no-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;

// Custom exception class
```

```java
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; // 0 and 1 are not prime numbers
        }
        for (int i = 2; i <= Math.sqrt(number); i++) {
            if (number % i == 0) {
                return false; // Found a divisor, not prime
            }
        }
        return true; // No divisors found, it is prime
    }

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

        System.out.print("Enter a number: ");
        int number = scanner.nextInt();

        try {
            // Check if the number is zero and throw exception
            if (number == 0) {
                throw new ZeroException("Number is 0");
            }

            // Check if the number is prime
            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.

```
import SY.SYMarks;
import TY.TYMarks;

import java.util.Scanner;

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

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

    public void calculateGrade() {
        int totalComputerMarks = syMarks.getComputerTotal() +
tyMarks.getTheory();
        String grade;

        if (totalComputerMarks >= 70) {
            grade = "A";
        } else if (totalComputerMarks >= 60) {
            grade = "B";
        } else if (totalComputerMarks >= 50) {
            grade = "C";
        } else if (totalComputerMarks >= 40) {
            grade = "Pass Class";
        } else {
            grade = "FAIL";
        }

        displayResult(grade);
    }

    public void displayResult(String grade) {
        System.out.printf("Roll Number: %s, Name: %s, Total Computer
Marks: %d, Grade: %s%n",
                rollNumber, name, (syMarks.getComputerTotal() +
tyMarks.getTheory()), grade);
    }

    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

        for (int i = 0; i < n; i++) {
            System.out.printf("Enter details for student %d:\n", (i + 1));

            System.out.print("Roll Number: ");
            String rollNumber = scanner.nextLine();

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

            System.out.print("SY Computer Marks: ");
            int syComputer = scanner.nextInt();

            System.out.print("SY Maths Marks: ");
            int syMaths = scanner.nextInt();

            System.out.print("SY Electronics Marks: ");
            int syElectronics = scanner.nextInt();

            System.out.print("TY Theory Marks: ");
            int tyTheory = scanner.nextInt();

            System.out.print("TY Practical Marks: ");
            int tyPracticals = scanner.nextInt();
            scanner.nextLine(); // Consume newline

            SYMarks syMarks = new SYMarks(syComputer, syMaths,
syElectronics);
            TYMarks tyMarks = new TYMarks(tyTheory, tyPracticals);
            Student student = new Student(rollNumber, name, syMarks,
tyMarks);

            student.calculateGrade();
        }

        scanner.close();
    }
}
```

Slip no-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 FileCopyExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        // Accept the names of the two files
        System.out.print("Enter the name of the source file (with .txt
extension): ");
        String sourceFileName = scanner.nextLine();
        System.out.print("Enter the name of the destination file (with
.txt extension): ");
        String destinationFileName = scanner.nextLine();

        copyFileContents(sourceFileName, destinationFileName);
        scanner.close();
    }

    private static void copyFileContents(String sourceFileName, String
destinationFileName) {
        try (BufferedReader reader = new BufferedReader(new
FileReader(sourceFileName));
             BufferedWriter writer = new BufferedWriter(new
FileWriter(destinationFileName))) {

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

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.

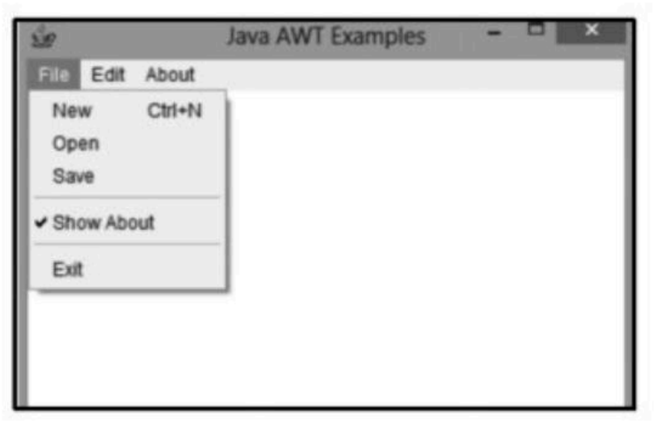
```java
// Base class: Account
class Account {
    protected String custName;
    protected String accNo;

    // 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(); // Call to Account default constructor
        this.savingBal = 0.0;
        this.minBal = 500.0; // Default minimum balance
    }

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

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

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

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

    // Method to display customer details
    public void displayCustomerDetails() {
        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("Deposit Amount: " + depositAmt);
        System.out.println("Withdrawal Amount: " + withdrawalAmt);
    }
}

// Main class to test the implementation
public class BankAccountDemo {
    public static void main(String[] args) {
        // Creating an object of AccountDetail using the parameterized
constructor
        AccountDetail account = new AccountDetail("John Doe", "123456",
1000.0, 500.0, 200.0, 100.0);

        // Displaying customer details
        account.displayCustomerDetails();
    }
}
```

Slip no-16

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

```
import java.util.Scanner;

// Functional interface
@FunctionalInterface
interface Square {
    double calculateSquare(double number);
}

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

        // Create a lambda expression for calculating the square
        Square squareFunction = (number) -> number * number;

        // Accept input from the user
        System.out.print("Enter a number to find its square: ");
        double inputNumber = scanner.nextDouble();

        // Calculate the square using the functional interface
        double result = squareFunction.calculateSquare(inputNumber);

        // Display the result
        System.out.println("The square of " + inputNumber + " is: " +
result);

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

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



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

public class MenuExample {
    // Constructor
    MenuExample() {
        // Creating a frame
        Frame frame = new Frame("Java AWT Examples");
        frame.setSize(400, 300);

        // Creating a menu bar
        MenuBar menuBar = new MenuBar();

        // Creating "File" menu and adding items
        Menu fileMenu = new Menu("File");
        MenuItem newFile = new MenuItem("New");
        MenuItem openFile = new MenuItem("Open");
        MenuItem saveFile = new MenuItem("Save");
        MenuItem showAbout = new MenuItem("Show About");
        MenuItem exitFile = new MenuItem("Exit");

        // Adding items to "File" menu
        fileMenu.add(newFile);
        fileMenu.add(openFile);
        fileMenu.add(saveFile);
        fileMenu.addSeparator();
        fileMenu.add(showAbout);
        fileMenu.addSeparator();
        fileMenu.add(exitFile);

        // Creating "Edit" and "About" menus (with no items for now)
        Menu editMenu = new Menu("Edit");
        Menu aboutMenu = new Menu("About");

        // Adding menus to the menu bar
        menuBar.add(fileMenu);
        menuBar.add(editMenu);
        menuBar.add(aboutMenu);

        // Adding the menu bar to the frame
        frame.setMenuBar(menuBar);

        // Adding a simple action listener to "Exit"
        exitFile.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                System.exit(0); // Exit the application
            }
        });

        // Show About Action
        showAbout.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                System.out.println("About Selected!");
            }
        });

        // Closing the window
        frame.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent we) {
                System.exit(0);
            }
        });

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

    public static void main(String[] args) {
        new MenuExample(); // Calling the constructor
    }
}
```

Slip no-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;

    // Method to read customer details
    public void readCustomerDetails() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter Customer Name: ");
        name = scanner.nextLine();
        System.out.print("Enter Phone Number: ");
        phoneNumber = scanner.nextLine();
    }

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

```java
    }
}

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

    // Method to read depositor details
    public void readDepositorDetails() {
        readCustomerDetails(); // Call method from Customer
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter Account Number: ");
        accNo = scanner.nextLine();
        System.out.print("Enter Balance: ");
        balance = scanner.nextDouble();
    }

    // Method to display depositor details
    public void displayDepositorDetails() {
        displayCustomerDetails(); // Call method from Customer
        System.out.println("Account Number: " + accNo);
        System.out.println("Balance: " + balance);
    }
}

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

    // Method to read borrower details
    public void readBorrowerDetails() {
        readDepositorDetails(); // Call method from Depositor
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter Loan Number: ");
        loanNo = scanner.nextLine();
        System.out.print("Enter Loan Amount: ");
        loanAmt = scanner.nextDouble();
    }
    // Method to display borrower details
    public void displayBorrowerDetails() {
        displayDepositorDetails(); // Call method from Depositor
        System.out.println("Loan Number: " + loanNo);
        System.out.println("Loan Amount: " + loanAmt);
    }
}

// Main class to test the implementation
public class CustomerDetailsDemo {
    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 the newline character

        Borrower[] customers = new Borrower[n];

        // Reading details of n customers
        for (int i = 0; i < n; i++) {
            System.out.println("\nEntering details for Customer " + (i +
1) + ":");
            customers[i] = new Borrower();
            customers[i].readBorrowerDetails();
        }

        // Displaying details of n customers
        System.out.println("\nCustomer Details:");
        for (int i = 0; i < n; i++) {
            System.out.println("\nDetails of Customer " + (i + 1) + ":");
            customers[i].displayBorrowerDetails();
        }

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

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

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

public class StringManipulationApp {
    public static void main(String[] args) {
        // Create a JFrame
        JFrame frame = new JFrame("String Manipulation");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400, 300);
        frame.setLayout(new FlowLayout());

        // Create text boxes
        JTextField textField1 = new JTextField(15);
        JTextField textField2 = new JTextField(15);
        JTextField textField3 = new JTextField(15);
        textField3.setEditable(false); // Make third text box read-only

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

        // Action listener for concatenation button
        concatButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String str1 = textField1.getText();
                String str2 = textField2.getText();
                String result = str1 + str2; // Concatenate strings
                textField3.setText(result); // Display result
            }
        });

        // Action listener for reverse button
        reverseButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String str1 = textField1.getText();
                String reversed = new
StringBuilder(str1).reverse().toString(); // Reverse string
                textField3.setText(reversed); // Display result
            }
        });

        // Add 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(textField3);

        // Set the frame visibility
        frame.setVisible(true);
    }
}
```

Slip no18

Q1) Write a program to implement Border Layout Manager.

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

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

        // Create buttons for each region
        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 buttons to the frame using BorderLayout
        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);

        // Set the frame visibility
        frame.setVisible(true);
    }
}
```

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 noOfInnings;
    int noOfTimesNotOut;
    int totalRuns;
    double batAvg;

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

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

    // Method to display player details
    @Override
    public String toString() {
        return "Player Name: " + name + ", Innings: " + noOfInnings + ",
Not Out: " + noOfTimesNotOut +
                ", Total Runs: " + totalRuns + ", Batting Average: " +
batAvg;
    }
}

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

        System.out.print("Enter number of players: ");
        int n = scanner.nextInt();
        CricketPlayer[] players = new CricketPlayer[n];

        // Input details for each player
        for (int i = 0; i < n; i++) {
            System.out.println("\nEnter details for Player " + (i + 1) +
":");
```

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

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

        // Sort players based on batting average
        Arrays.sort(players, new Comparator<CricketPlayer>() {
            @Override
            public int compare(CricketPlayer p1, CricketPlayer p2) {
                return Double.compare(p2.batAvg, p1.batAvg); // Sort in
descending order
            }
        });

        // Display player details in sorted order
        System.out.println("\nPlayer details in sorted order based on
batting average:");
        for (CricketPlayer player : players) {
            System.out.println(player);
        }

        scanner.close();
    }
}
```

Slip no-19

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

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

```java
public class DiagonalSum {
    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 (n): ");
        int n = scanner.nextInt();
        System.out.print("Enter the number of columns (m): ");
        int m = scanner.nextInt();

        // Ensure it's a square matrix
        if (n != m) {
            System.out.println("Please enter a square matrix (same number
of rows and columns).");
            return;
        }

        // Create a two-dimensional array
        int[][] matrix = new int[n][m];

        // Accept 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 < m; j++) {
                System.out.print("Element [" + i + "][" + j + "]: ");
                matrix[i][j] = scanner.nextInt();
            }
        }

        // Calculate the sum of diagonal elements
        int primaryDiagonalSum = 0;
        int secondaryDiagonalSum = 0;

        for (int i = 0; i < n; i++) {
            primaryDiagonalSum += matrix[i][i]; // Primary diagonal
            secondaryDiagonalSum += matrix[i][n - 1 - i]; // Secondary
diagonal
        }
        // Display the results
        System.out.println("Sum of primary diagonal elements: " +
primaryDiagonalSum);
        System.out.println("Sum of secondary diagonal elements: " +
secondaryDiagonalSum);
        System.out.println("Total sum of diagonal elements: " +
(primaryDiagonalSum + secondaryDiagonalSum));

        scanner.close();
    }
}
```

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 the 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 = {
                "Operating Systems",
                "Database Management Systems",
                "Software Engineering",
                "Web Technology",
                "Computer Networks",
                "Data Structures",
                "Object-Oriented Programming",
                "Theory of Computation"
        };
        JComboBox<String> subjectComboBox = new JComboBox<>(subjects);
        frame.add(subjectComboBox);

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

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

        // Set the frame to be visible
        frame.setVisible(true);
    }
}
```

Slip no-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
class Continent {
    protected String continentName;

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

    public String getContinentName() {
        return continentName;
    }
}

// Derived class from Continent
class Country extends Continent {
    protected String countryName;

    public Country(String continentName, String countryName) {
        super(continentName); // Call to the constructor of the base class
        this.countryName = countryName;
    }

    public String getCountryName() {
        return countryName;
    }
}

// Further derived class from Country
class State extends Country {
    private String stateName;
    private String placeName;

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

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

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

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

```
import java.util.Scanner;

// Class for Addition
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;
    }
}

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

// Test class to demonstrate functionality
public class TestOperation {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        // Create objects of Addition and Maximum classes
        Addition addition = new Addition();
        Maximum maximum = new Maximum();

        // Input two integers for addition and maximum
        System.out.print("Enter first integer for addition: ");
        int num1 = scanner.nextInt();
        System.out.print("Enter second integer for addition: ");
        int num2 = scanner.nextInt();

        // Demonstrate addition
        int sum = addition.add(num1, num2);
        System.out.println("Sum of " + num1 + " and " + num2 + ": " +
sum);

        // Input two float values for subtraction
        System.out.print("Enter first float for subtraction: ");
        float float1 = scanner.nextFloat();
        System.out.print("Enter second float for subtraction: ");
        float float2 = scanner.nextFloat();

        // Demonstrate subtraction
        float difference = addition.subtract(float1, float2);
        System.out.println("Difference between " + float1 + " and " +
float2 + ": " + difference);

        // Input two integers for maximum calculation
        System.out.print("Enter first integer for maximum: ");
        int maxNum1 = scanner.nextInt();
        System.out.print("Enter second integer for maximum: ");
        int maxNum2 = scanner.nextInt();

        // Demonstrate maximum
        int max = maximum.max(maxNum1, maxNum2);
        System.out.println("Maximum of " + maxNum1 + " and " + maxNum2 +
": " + max);

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

```
    }
}
```

Slip no-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.

```
import java.util.Scanner;

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

// 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 {
        if (!isValidDate(day, month, year)) {
            throw new InvalidDateException("Invalid Date: " + day + "/" +
month + "/" + year);
        }
        this.day = day;
        this.month = month;
        this.year = year;
    }

    // Method to check if the date is valid
    private boolean isValidDate(int day, int month, int year) {
        if (month < 1 || month > 12) {
            return false;
```

```java
        }
        if (day < 1 || day > daysInMonth(month, year)) {
            return false;
        }
        return true;
    }

    // Method to get the number of days in a month, considering leap years
    private int daysInMonth(int month, int year) {
        switch (month) {
            case 1: case 3: case 5: case 7: case 8: case 10: case 12:
                return 31;
            case 4: case 6: case 9: case 11:
                return 30;
            case 2:
                if (isLeapYear(year)) {
                    return 29;
                } else {
                    return 28;
                }
            default:
                return 0;
        }
    }

    // Method to check if a year is a leap year
    private boolean isLeapYear(int year) {
        if (year % 400 == 0 || (year % 100 != 0 && year % 4 == 0)) {
            return true;
        }
        return false;
    }

    // Method to display the date
    public void displayDate() {
        System.out.println("Date: " + day + "/" + month + "/" + year);
    }

    // Method to accept the date from user input
    public static MyDate acceptDate() throws InvalidDateException {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter day: ");
        int day = scanner.nextInt();
        System.out.print("Enter month: ");
        int month = scanner.nextInt();
        System.out.print("Enter year: ");
        int year = scanner.nextInt();

        // Return the created MyDate object
        return new MyDate(day, month, year);
    }
}

// Test class to demonstrate the MyDate class and exception handling
public class TestDate {
    public static void main(String[] args) {
        try {
            // Accept date from user and display
            MyDate date = MyDate.acceptDate();
            date.displayDate();
        } catch (InvalidDateException e) {
            // Catch and display the invalid date exception message
            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;
    private static int objectCount = 0;  // Static variable to keep track
of object count
```

```
    // Default constructor
    public Employee() {
        this(0, "Unknown", "Unknown", 0.0); // Default values, using the
parameterized constructor
    }

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

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

    // Method to display the contents of the object
    public void display() {
        System.out.println("ID: " + this.id);
        System.out.println("Name: " + this.name);
        System.out.println("Department Name: " + this.deptname);
        System.out.println("Salary: " + this.salary);
        System.out.println("------------------------");
    }

    public static void main(String[] args) {
        // Creating first employee object
        Employee emp1 = new Employee(101, "John Doe", "IT", 55000.0);
        emp1.display();
        System.out.println("Number of Employee objects created: " +
Employee.getObjectCount());

        // Creating second employee object
        Employee emp2 = new Employee(102, "Jane Smith", "HR", 60000.0);
        emp2.display();
        System.out.println("Number of Employee objects created: " +
Employee.getObjectCount());

        // Creating third employee object
        Employee emp3 = new Employee(103, "David Johnson", "Finance",
75000.0);
        emp3.display();
        System.out.println("Number of Employee objects created: " +
Employee.getObjectCount());
    }
}
```

Slip no-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 {
    // Fields to store dimensions
    int dim1, dim2;

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

// Rectangle class extends Shape
class Rectangle extends Shape {
    // Constructor to initialize dimensions of the rectangle
    Rectangle(int length, int breadth) {
        this.dim1 = length;
        this.dim2 = breadth;
    }

    // Overriding printArea() method to calculate and display the area of
a rectangle
    @Override
    void printArea() {
        int area = dim1 * dim2;
        System.out.println("Rectangle Area: " + area);
    }
}

// Triangle class extends Shape
class Triangle extends Shape {
    // Constructor to initialize base and height of the triangle
    Triangle(int base, int height) {
        this.dim1 = base;
        this.dim2 = height;
    }

    // Overriding printArea() method to calculate and display the area of
a triangle
    @Override
    void printArea() {
        double area = 0.5 * dim1 * dim2;
        System.out.println("Triangle Area: " + area);
    }
}

// Circle class extends Shape
class Circle extends Shape {
    // Constructor to initialize radius of the circle
    Circle(int radius) {
        this.dim1 = radius; // Only one dimension needed for the circle
    }

    // Overriding printArea() method to calculate and display the area of
a circle
    @Override
    void printArea() {
        double area = Math.PI * dim1 * dim1; // π * r^2
        System.out.println("Circle Area: " + area);
    }
}

// Main class to test the shapes
```

```java
public class ShapeTest {
    public static void main(String[] args) {
        // Create objects of Rectangle, Triangle, and Circle
        Rectangle rect = new Rectangle(10, 5); // length = 10, breadth = 5
        Triangle tri = new Triangle(6, 8);     // base = 6, height = 8
        Circle cir = new Circle(7);            // radius = 7

        // Call printArea() method for each object
        rect.printArea();
        tri.printArea();
        cir.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).

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

public class MouseEventDemo extends JFrame {

    // Label to display the mouse event name
    private JLabel label;

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

        // Create a label to display the mouse event name
        label = new JLabel("", SwingConstants.CENTER);
        label.setFont(new Font("Arial", Font.BOLD, 20));
        label.setForeground(Color.RED);
        add(label, BorderLayout.CENTER);
```

```java
// Add a mouse listener using a MouseAdapter class
addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        label.setText("Mouse Clicked");
    }

    @Override
    public void mouseEntered(MouseEvent e) {
        label.setText("Mouse Entered");
    }

    @Override
    public void mouseExited(MouseEvent e) {
        label.setText("Mouse Exited");
    }

    @Override
    public void mousePressed(MouseEvent e) {
        label.setText("Mouse Pressed");
    }

    @Override
    public void mouseReleased(MouseEvent e) {
        label.setText("Mouse Released");
    }
});

addMouseMotionListener(new MouseAdapter() {
    @Override
    public void mouseMoved(MouseEvent e) {
        label.setText("Mouse Moved");
    }

    @Override
    public void mouseDragged(MouseEvent e) {
        label.setText("Mouse Dragged");
    }
});
```

```
        // Set the frame visible
        setVisible(true);
    }

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

Slip no-23

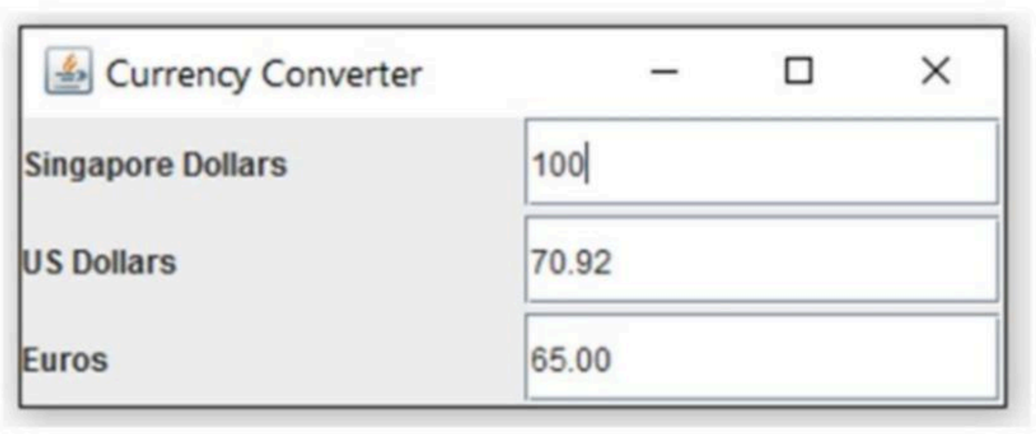
Q1) Define a class MyNumber having one private int data member. Write a default constructor to initialize it to 0 and another constructor to initialize it to a value (Use this). Write methods isNegative, is Positive, isZero, isOdd, isEven. Create an object in main. Use command line arguments to pass a value to the Object.

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

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

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

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

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

```java
    }

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

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

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

    // Method to display the number and its properties
    public void displayProperties() {
        System.out.println("Number: " + number);
        if (isNegative()) {
            System.out.println("The number is Negative.");
        } else if (isPositive()) {
            System.out.println("The number is Positive.");
        } else {
            System.out.println("The number is Zero.");
        }

        if (isOdd()) {
            System.out.println("The number is Odd.");
        } else {
            System.out.println("The number is Even.");
        }
    }

    // Main method to create an object and pass command-line arguments
    public static void main(String[] args) {
        if (args.length == 0) {
            System.out.println("Please provide a number as a command-line
argument.");
```

```
            return;
        }

        try {
            // Parse the command-line argument to an integer
            int inputNumber = Integer.parseInt(args[0]);

            // Create an object of MyNumber using the parameterized
constructor
            MyNumber myNum = new MyNumber(inputNumber);

            // Display the properties of the number
            myNum.displayProperties();
        } catch (NumberFormatException e) {
            System.out.println("Invalid input! Please enter a valid
integer.");
        }
    }
}
```

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 SGID=0.65 Euro.



Singapore Dollars	100
US Dollars	70.92
Euros	65.00

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

public class CurrencyConverter {
    public static void main(String[] args) {
        // Create JFrame
        JFrame frame = new JFrame("Currency Converter");
        frame.setSize(400, 200);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLayout(null);

        // Create text fields
        JTextField sgdField = new JTextField();
        sgdField.setBounds(150, 20, 100, 30);
        frame.add(sgdField);

        JTextField usdField = new JTextField();
        usdField.setBounds(150, 60, 100, 30);
        frame.add(usdField);

        JTextField euroField = new JTextField();
        euroField.setBounds(150, 100, 100, 30);
        frame.add(euroField);

        // Labels
        JLabel sgdLabel = new JLabel("Singapore Dollars");
        sgdLabel.setBounds(20, 20, 120, 30);
        frame.add(sgdLabel);

        JLabel usdLabel = new JLabel("US Dollars");
        usdLabel.setBounds(20, 60, 120, 30);
        frame.add(usdLabel);

        JLabel euroLabel = new JLabel("Euros");
        euroLabel.setBounds(20, 100, 120, 30);
        frame.add(euroLabel);

        // Button to calculate
        JButton convertButton = new JButton("Convert");
        convertButton.setBounds(270, 20, 100, 30);
        frame.add(convertButton);
```

```java
        // Conversion rates
        double usdToSgd = 1.41;
        double usdToEuro = 0.92;
        double sgdToEuro = 0.65;

        DecimalFormat df = new DecimalFormat("0.00");

        // ActionListener for button
        convertButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String sgdText = sgdField.getText();
                String usdText = usdField.getText();
                String euroText = euroField.getText();

                if (!sgdText.isEmpty()) {
                    double sgd = Double.parseDouble(sgdText);
                    usdField.setText(df.format(sgd / usdToSgd));
                    euroField.setText(df.format(sgd * sgdToEuro));
                } else if (!usdText.isEmpty()) {
                    double usd = Double.parseDouble(usdText);
                    sgdField.setText(df.format(usd * usdToSgd));
                    euroField.setText(df.format(usd * usdToEuro));
                } else if (!euroText.isEmpty()) {
                    double euro = Double.parseDouble(euroText);
                    usdField.setText(df.format(euro / usdToEuro));
                    sgdField.setText(df.format(euro / sgdToEuro));
                }
            }
        });

        frame.setVisible(true);
    }
}
```

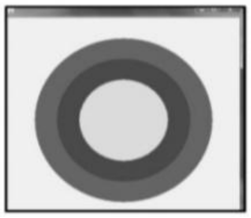
Slip no-24

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.

```java
// Abstract class Bank
abstract class Bank {
    // Abstract method to get balance
    public abstract int getBalance();
}

// Class BankA which extends Bank
class BankA extends Bank {
    private int balance;

    // Constructor to initialize balance
    public BankA() {
        this.balance = 100;  // Rs. 100 deposited
    }

    // Override the abstract method getBalance
    @Override
    public int getBalance() {
        return balance;
    }
}

// Class BankB which extends Bank
class BankB extends Bank {
    private int balance;

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

    // Override the abstract method getBalance
    @Override
    public int getBalance() {
        return balance;
    }
}
```

```
// Class BankC which extends Bank
class BankC extends Bank {
    private int balance;

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

    // Override the abstract method getBalance
    @Override
    public int getBalance() {
        return balance;
    }
}

// Main class to test the program
public class BankTest {
    public static void main(String[] args) {
        // Create objects for BankA, BankB, and BankC
        Bank bankA = new BankA();
        Bank bankB = new BankB();
        Bank bankC = new BankC();

        // Display balances for each bank
        System.out.println("Balance in Bank A: Rs. " +
bankA.getBalance());
        System.out.println("Balance in Bank B: Rs. " +
bankB.getBalance());
        System.out.println("Balance in Bank C: Rs. " +
bankC.getBalance());
    }
}
```

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 = -100;
    private int y = -100;

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

        // Add mouse listener to detect clicks on the frame
        addMouseListener(new MouseAdapter() {
            @Override
            public void mouseClicked(MouseEvent e) {
                // Update the coordinates with the mouse click location
                x = e.getX();
                y = e.getY();
                repaint(); // Call repaint to redraw the circles
            }
        });
    }

    @Override
    public void paint(Graphics g) {
        super.paint(g);
        // Draw three concentric circles centered at the clicked position
        g.drawOval(x - 30, y - 30, 60, 60);
        g.drawOval(x - 60, y - 60, 120, 120);
        g.drawOval(x - 90, y - 90, 180, 180);
```

```java
    }

    public static void main(String[] args) {
        // Create the frame and make it visible
        ConcentricCircles frame = new ConcentricCircles();
        frame.setVisible(true);
    }
}
```

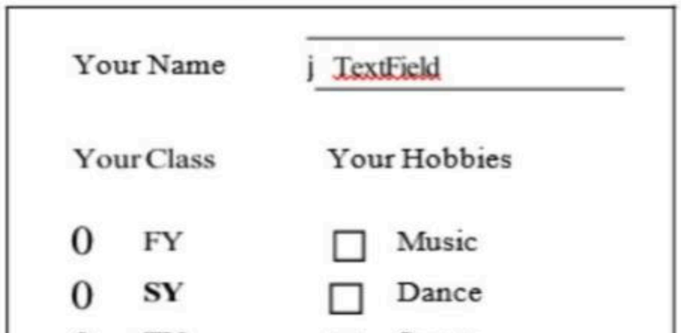
Slip no-25

Q1) Create a class Student(rollno, name,class, per), to read student information from the console and display them (Using BufferedReader class)

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

class Student {
    // Data members of the Student class
    private int rollno;
    private String name;
    private String studentClass;
    private double percentage;

    // Constructor to initialize student details
    public Student(int rollno, String name, String studentClass, double
percentage) {
        this.rollno = rollno;
        this.name = name;
        this.studentClass = studentClass;
        this.percentage = percentage;
    }

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

public class StudentDetails {
    public static void main(String[] args) throws IOException {
        // Create a BufferedReader object to read input from the console
        BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));

        // Read student details from the console
        System.out.print("Enter Student Roll No: ");
        int rollno = Integer.parseInt(reader.readLine());

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

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

        System.out.print("Enter Student Percentage: ");
        double percentage = Double.parseDouble(reader.readLine());

        // Create a Student object with the provided information
        Student student = new Student(rollno, name, studentClass,
percentage);

        // Display the student details
        System.out.println("\nStudent Information:");
        student.display();
    }
}
```

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

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.



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

public class UserInfoForm {
    public static void main(String[] args) {
        // Frame creation
        JFrame frame = new JFrame("User Info Form");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400, 300);

        // Create a panel
        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(6, 2));

        // Your Name
        JLabel nameLabel = new JLabel("Your Name");
        JTextField nameField = new JTextField();
        panel.add(nameLabel);
        panel.add(nameField);

        // Your Class
        JLabel classLabel = new JLabel("Your Class");
        ButtonGroup classGroup = new ButtonGroup();
        JRadioButton fyRadio = new JRadioButton("FY");
        JRadioButton syRadio = new JRadioButton("SY");
        JRadioButton tyRadio = new JRadioButton("TY");
        classGroup.add(fyRadio);
        classGroup.add(syRadio);
        classGroup.add(tyRadio);
        panel.add(classLabel);

        JPanel classPanel = new JPanel();
        classPanel.setLayout(new GridLayout(3, 1));
        classPanel.add(fyRadio);
        classPanel.add(syRadio);
        classPanel.add(tyRadio);
        panel.add(classPanel);

        // Your Hobbies
        JLabel hobbiesLabel = new JLabel("Your Hobbies");
        JCheckBox musicCheckBox = new JCheckBox("Music");
        JCheckBox danceCheckBox = new JCheckBox("Dance");
        JCheckBox sportsCheckBox = new JCheckBox("Sports");

        panel.add(hobbiesLabel);

        JPanel hobbiesPanel = new JPanel();
        hobbiesPanel.setLayout(new GridLayout(3, 1));
        hobbiesPanel.add(musicCheckBox);
        hobbiesPanel.add(danceCheckBox);
        hobbiesPanel.add(sportsCheckBox);
        panel.add(hobbiesPanel);

        // Output TextField
        JTextField outputField = new JTextField();
        outputField.setEditable(false);

        // Submit Button
        JButton submitButton = new JButton("Submit");
        submitButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String name = nameField.getText();
                String selectedClass = fyRadio.isSelected() ? "FY" :
syRadio.isSelected() ? "SY" : tyRadio.isSelected() ? "TY" : "";
                String hobbies = "";
                if (musicCheckBox.isSelected()) hobbies += "Music ";
                if (danceCheckBox.isSelected()) hobbies += "Dance ";
                if (sportsCheckBox.isSelected()) hobbies += "Sports ";

                outputField.setText("Name: " + name + " | Class: " +
selectedClass + " | Hobbies: " + hobbies);
            }
        });

        panel.add(submitButton);
        panel.add(outputField);

        // Add panel to frame and display
        frame.add(panel);
        frame.setVisible(true);
    }
}
```

Slip no-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 {
    // Instance variables
    private int item_number;
    private String item_name;
    private double item_price;

    // Static member to keep track of object count
    private static int itemCount = 0;

    // Default constructor
    public Item() {
        this.item_number = 0;
        this.item_name = "Unknown";
        this.item_price = 0.0;
        itemCount++; // Increment item count when an object is created
    }
```

```java
    // Parameterized constructor
    public Item(int item_number, String item_name, double item_price) {
        this.item_number = item_number;
        this.item_name = item_name;
        this.item_price = item_price;
        itemCount++; // Increment item count when an object is created
    }

    // Static method to return the count of objects created
    public static int getItemCount() {
        return itemCount;
    }

    // Method to display item details
    public void displayItem() {
        System.out.println("Item Number: " + item_number);
        System.out.println("Item Name: " + item_name);
        System.out.println("Item Price: $" + item_price);
        System.out.println();
    }

    // Main method to test the functionality
    public static void main(String[] args) {
        // Creating objects using the parameterized constructor
        Item item1 = new Item(101, "Laptop", 1500.99);
        System.out.println("Item 1 Details:");
        item1.displayItem();
        System.out.println("Total Items Created: " + Item.getItemCount());

        Item item2 = new Item(102, "Smartphone", 999.99);
        System.out.println("Item 2 Details:");
        item2.displayItem();
        System.out.println("Total Items Created: " + Item.getItemCount());

        Item item3 = new Item(103, "Headphones", 199.99);
        System.out.println("Item 3 Details:");
        item3.displayItem();
        System.out.println("Total Items Created: " + Item.getItemCount());
    }
}
```

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;
import java.time.temporal.ChronoUnit;

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

    // Parameterized constructor
    public Donor(String name, int age, String address, String
contactNumber, String bloodGroup, String lastDonationDate) {
        this.name = name;
        this.age = age;
        this.address = address;
        this.contactNumber = contactNumber;
        this.bloodGroup = bloodGroup;
        this.lastDonationDate = lastDonationDate;
    }

    // Method to check if the donor is A+ and hasn't donated in the last 6
months
    public boolean isEligibleDonor() {
        if (!bloodGroup.equals("A+ve")) {
            return false; // Filter only A+ve blood group
        }
        // Check if last donation date is older than 6 months
```

```java
        DateTimeFormatter formatter =
DateTimeFormatter.ofPattern("dd/MM/yyyy");
        LocalDate lastDonation = LocalDate.parse(lastDonationDate,
formatter);
        LocalDate currentDate = LocalDate.now();
        long monthsSinceLastDonation =
ChronoUnit.MONTHS.between(lastDonation, currentDate);
        return monthsSinceLastDonation >= 6;
    }

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

    // Main method to create donor objects and write/read them to/from a
file
    public static void main(String[] args) {
        Donor[] donors = {
            new Donor("John Doe", 35, "Pune", "1234567890", "A+ve",
"01/01/2023"),
            new Donor("Jane Smith", 28, "Pune", "0987654321", "B+ve",
"15/03/2023"),
            new Donor("Mark Taylor", 42, "Pune", "5556667777", "A+ve",
"05/10/2022"),
            new Donor("Emily Clark", 31, "Pune", "8889991111", "O+ve",
"02/02/2023"),
            new Donor("Lucas Adams", 29, "Pune", "2223334444", "A+ve",
"10/07/2023")
        };

        // File where donor objects will be stored
        String fileName = "donors.dat";
        // Write donor objects to the file
        try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(fileName))) {
            for (Donor donor : donors) {
                oos.writeObject(donor);
            }
            System.out.println("Donor objects written to file
successfully.");
        } catch (IOException e) {
            e.printStackTrace();
        }

        // Read donor objects from the file and filter based on blood
group and donation date
        System.out.println("\nDonors with A+ve blood group who haven't
donated in the last 6 months:");
        try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream(fileName))) {
            while (true) {
                try {
                    Donor donor = (Donor) ois.readObject();
                    if (donor.isEligibleDonor()) {
                        donor.displayDonorDetails();
                    }
                } catch (EOFException eof) {
                    break; // End of file reached
                }
            }
        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
}
```

Slip no-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.

```java
// Base class Employee
class Employee {
    // Attributes
    private String name;
    private double basicSalary;

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

    // Method to get the salary
    public double getSalary() {
        return basicSalary; // Base salary for employee
    }

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

// Derived class Manager that extends Employee
class Manager extends Employee {
    // Allowances
    private double travelAllowance;
    private double houseRentAllowance;

    // Constructor
    public Manager(String name, double basicSalary, double
travelAllowance, double houseRentAllowance) {
        super(name, basicSalary); // Call to the superclass constructor
        this.travelAllowance = travelAllowance;
        this.houseRentAllowance = houseRentAllowance;
    }
    // Overriding getSalary() method
    @Override
    public double getSalary() {
        // Calculate manager's salary including allowances
        return super.getSalary() + travelAllowance + houseRentAllowance;
    }

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

// Main class to test the implementation
public class Main {
    public static void main(String[] args) {
        // Creating an Employee object
        Employee emp1 = new Employee("John Doe", 50000);
        emp1.displayDetails();
        System.out.println("-----------------------------");

        // Creating a Manager object
        Manager mgr1 = new Manager("Alice Smith", 70000, 5000, 10000);
        mgr1.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.File;
import java.util.Scanner;

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

        // Get the input path from the command line argument
        String path = args[0];
        File file = new File(path);

        // Check if the path is a directory
        if (file.isDirectory()) {
            System.out.println("It is a directory. Deleting text
files...");

            // Get the list of files in the directory
            File[] files = file.listFiles();
            if (files != null && files.length > 0) {
                int deleteCount = 0;
                Scanner scanner = new Scanner(System.in);
                System.out.print("Are you sure you want to delete all text
files in this directory? (yes/no): ");
                String confirmation = scanner.nextLine();

                // Proceed if user confirms
                if (confirmation.equalsIgnoreCase("yes")) {
                    for (File f : files) {
                        // Check if the file is a text file
                        if (f.isFile() && f.getName().endsWith(".txt")) {
                            if (f.delete()) {
                                System.out.println("Deleted: " +
f.getName());
                                deleteCount++;
                            } else {
                                System.out.println("Failed to delete: " +
f.getName());
                            }
                        }
                    }
                } else {
                    System.out.println("Delete operation canceled.");
                }

                // Display the number of files deleted
                System.out.println("Total text files deleted: " +
deleteCount);
            } else {
                System.out.println("The directory is empty or does not
exist.");
            }
        }
        // Check if the path is a file
        else if (file.isFile()) {
            System.out.println("It is a file. Displaying file details:");
            System.out.println("File Name: " + file.getName());
            System.out.println("File Path: " + file.getAbsolutePath());
            System.out.println("File Size: " + file.length() + " bytes");
            System.out.println("Is Readable: " + file.canRead());
            System.out.println("Is Writable: " + file.canWrite());
            System.out.println("Is Executable: " + file.canExecute());
        }
        // If neither, display a message
        else {
            System.out.println("The specified path is neither a file nor a
directory.");
        }
    }
}
```

Slip no-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.

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

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

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

    // Method to display customer details
    public void displayDetails() {
        System.out.println("Customer Number: " + custNo);
        System.out.println("Customer Name: " + custName);
        System.out.println("Contact Number: " + contactNumber);
        System.out.println("Customer Address: " + custAddr);
    }

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

public class CustomerSearch {
    public static void main(String[] args) {
        // Create a list to store customers
        ArrayList<Customer> customerList = new ArrayList<>();
        // Sample customers
        customerList.add(new Customer(1, "Alice Smith", "1234567890", "123
Main St"));
        customerList.add(new Customer(2, "Bob Johnson", "9876543210", "456
Elm St"));
        customerList.add(new Customer(3, "Charlie Brown", "5551234567",
"789 Oak St"));

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

        // Search for customer by contact number
        boolean found = false;
        for (Customer customer : customerList) {
            if (customer.getContactNumber().equals(contactNumber)) {
                customer.displayDetails();
                found = true;
                break;
            }
        }

        if (!found) {
            System.out.println("Customer not found with contact number: "
+ contactNumber);
        }

        scanner.close();
    }
}
```