Enterprise Java Lab
Questions

Refer at your own risk.
If there are any mistakes please let me know

1. **Write a JDBC program to Accept Rno, sname and marks in 3 subjects .. Calculate total, average and Grade and add the record into the database. Finally display all the records. Use Prepared statement.**

Query:

Create table:

```
CREATE TABLE student (
    rno INT,
    name VARCHAR(255),
    marks1 INT,
    marks2 INT,
    marks3 INT,
    total INT,
    averege DOUBLE,
    grade VARCHAR(10)
);
```

Code:

```
import java.sql.*;
import java.util.*;

public class StudentRecords {
    public static void main(String[] args) {
        try {
            Scanner sc = new Scanner(System.in);
     Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/MCA",
"root", "password");
            System.out.println("Enter Student Rollno:");
            int rno = sc.nextInt();
            sc.nextLine(); // Consume newline character

            System.out.println("Enter Student name:");
```

```
        String name = sc.nextLine();

        System.out.println("Enter Student Mark1:");
        int m1 = sc.nextInt();

        System.out.println("Enter Student Mark2:");
        int m2 = sc.nextInt();

        System.out.println("Enter Student Mark3:");
        int m3 = sc.nextInt();

        // Prepare a SQL statement
         String sql = "INSERT INTO student (rno, name, marks1, marks2, marks3, total,
averege, grade) VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
        PreparedStatement p = con.prepareStatement(sql);

        int total = m1 + m2 + m3;
        double average = total / 3.0;
        String grade;
        if (average >= 90) {
            grade = "A";
        } else if (average >= 80) {
            grade = "B";
        } else if (average >= 70) {
            grade = "C";
        } else {
            grade = "D";
        }

        p.setInt(1, rno);
        p.setString(2, name);
        p.setInt(3, m1);
        p.setInt(4, m2);
        p.setInt(5, m3);
        p.setInt(6, total);
        p.setDouble(7, average);
        p.setString(8, grade);

        p.executeUpdate();
        System.out.println("Student Record added successfully");
```

```
            // Displaying all records
              ResultSet resultSet = con.createStatement().executeQuery("SELECT * FROM
student");
            while (resultSet.next()) {
                System.out.println("Roll No: " + resultSet.getInt("rno"));
                System.out.println("Name: " + resultSet.getString("name"));
                System.out.println("Marks 1: " + resultSet.getInt("marks1"));
                System.out.println("Marks 2: " + resultSet.getInt("marks2"));
                System.out.println("Marks 3: " + resultSet.getInt("marks3"));
                System.out.println("Total: " + resultSet.getInt("total"));
                System.out.println("Average: " + resultSet.getDouble("averege"));
                System.out.println("Grade: " + resultSet.getString("grade"));
                System.out.println();
            }
            con.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Output:

```
Enter Student Rollno:
123
Enter Student name:
Vivian
Enter Student Mark1:
99
Enter Student Mark2:
98
Enter Student Mark3:
97
Student Record added successfully
Roll No: 123
Name: Vivian
Marks 1: 99
Marks 2: 98
Marks 3: 97
```

Total: 294
Average: 98.0
Grade: A

2. **Write method overriding program for calculating the salary of different types of employees in a bank class.**

Code:

```
import java.util.Scanner;

class Employee {
    String name;
    double basicSalary;

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

   public double calculateSalary() {
       return basicSalary;
   }
   public String getName() {
       return name;
}
}
class Manager extends Employee {
    double bonus;

   public Manager(String name, double basicSalary,double bonus) {
       super(name, basicSalary);
       this.bonus=bonus;
   }

   public double calculateSalary() {
       return super.calculateSalary()+bonus;
   }
}
```

```
class Clerk extends Employee {
    double bonus;
    public Clerk(String name, double basicSalary,double bonus) {
        super(name, basicSalary);
        this.bonus=bonus;
    }


    public double calculateSalary() {
        return super.calculateSalary()+bonus;
    }
}

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

        System.out.println("Enter Manager's name: ");
        String managerName = scanner.nextLine();
        System.out.println("Enter Manager's basic salary: ");
        double managerBasicSalary = scanner.nextDouble();
        System.out.println("Enter Manager's bonus: ");
        double managerBonus = scanner.nextDouble();
                    Manager m = new Manager(managerName, managerBasicSalary,
managerBonus);

        System.out.println("Enter Clerk's name: ");
        scanner.nextLine(); // Consume newline
        String clerkName = scanner.nextLine();
        System.out.println("Enter Clerk's basic salary: ");
        double clerkBasicSalary = scanner.nextDouble();
        System.out.println("Enter Clearks's bonus: ");
        double clearkBonus = scanner.nextDouble();
        Clerk c = new Clerk(clerkName, clerkBasicSalary,clearkBonus);


                    System.out.println("\nManager: " + m.getName() + ", Salary: " +
m.calculateSalary());
        System.out.println("Clerk: " + c.getName() + ", Salary: " + c.calculateSalary());
```

```
    }
}
```

Output:

```
Enter Manager's name:
Vivian
Enter Manager's basic salary:
50000
Enter Manager's bonus:
5000
Enter Clerk's name:
Ramesh
Enter Clerk's basic salary:
65000
Enter Clearks's bonus:
2000

Manager: Vivian, Salary: 55000.0
Clerk: Ramesh, Salary: 67000.0
```

3. **Write a program to Show the multilevel and hierarchical inheritance for a student class.**

Code:

```
import java.util.Scanner;

class Student {
    int rollno;
    String name;

    public Student(String name, int rollno) {
        this.name = name;
        this.rollno = rollno;
    }

    public void displayInfo() {
        System.out.println("RollNo: " + rollno);
        System.out.println("Name: " + name);
    }
```

```
}
//heirarchical as well as multilevel
class Subject extends Student {
    String subjectName;

    public Subject(String name, int rollno, String subjectName) {
        super(name, rollno);
        this.subjectName = subjectName;
    }

    public void displaySubjectInfo() {
        super.displayInfo();
        System.out.println("Subject: " + subjectName);
    }
}
//multilevel
class Teacher extends Subject {
    String teacherName;

    public Teacher(String name, int rollno, String subjectName, String teacherName) {
        super(name, rollno, subjectName);
        this.teacherName = teacherName;
    }

    public void displayTeacherInfo() {
        super.displaySubjectInfo();
        System.out.println("Teacher Name: " + teacherName);
    }
}
//hierarchical
class Course extends Student {
    String courseName;

    public Course(String name, int rollno, String courseName) {
        super(name, rollno);
        this.courseName = courseName;
    }

    public void displayCourseInfo() {
        super.displayInfo();
```

```
        System.out.println("Course Name: " + courseName);
    }
}

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

        // Input for Subject
        System.out.println("Enter Student's name: ");
        String name = scanner.nextLine();
        System.out.println("Enter Roll No: ");
        int rollno = scanner.nextInt();
        scanner.nextLine(); // Consume newline
        System.out.println("Enter Subject: ");
        String subjectName = scanner.nextLine();
        Subject subject = new Subject(name, rollno, subjectName);

        // Input for Teacher
        System.out.println("Enter Teacher's name: ");
        String teacherName = scanner.nextLine();
        Teacher teacher = new Teacher(name, rollno, subjectName, teacherName);

        // Input for Course
        System.out.println("Enter Course: ");
        String courseName = scanner.nextLine();
        Course course = new Course(name, rollno, courseName);

        System.out.println("\nSubject Info:");
        subject.displaySubjectInfo();
        System.out.println("\nTeacher Info:");
        teacher.displayTeacherInfo();
        System.out.println("\nCourse Info:");
        course.displayCourseInfo();
    }
}
```

Output:

```
Enter Student's name:
Vivian
Enter Roll No:
177
Enter Subject:
Java
Enter Teacher's name:
Ananth Murthy
Enter Course:
MCA

Subject Info:
RollNo: 177
Name: Vivian
Subject: Java

Teacher Info:
RollNo: 177
Name: Vivian
Subject: Java
Teacher Name: Ananth Murthy

Course Info:
RollNo: 177
Name: Vivian
Course Name: MCA
```

Simple Code:
(U can refer this code also it just demonstrate both multi and hierarchical inheritance concept)

```
// Base class
class Person {
    void display() {
        System.out.println("I am a Person");
    }
}

// Derived class from Person (Multilevel Inheritance)
class Student extends Person {
    void display() {
```

```
        System.out.println("I am a Student");
    }
}

// Another derived class from Student (Multilevel Inheritance)
class CollegeStudent extends Student {
    void display() {
        System.out.println("I am a College Student");
    }
}

// Derived class from Person (Hierarchical Inheritance)
class Teacher extends Person {
    void display() {
        System.out.println("I am a Teacher");
    }
}

public class Main {
    public static void main(String[] args) {
        CollegeStudent collegeStudent = new CollegeStudent();
        collegeStudent.display(); // Output: I am a College Student

        Student student = new Student();
        student.display(); // Output: I am a Student

        Teacher teacher = new Teacher();
        teacher.display(); // Output: I am a Teacher

        Person person = new Person();
        person.display(); // Output: I am a Person
    }
}
```

Output:

```
I am a College Student
I am a Student
I am a Teacher
I am a Person
```

4. **Write a JDBC program to manage employee database. Use the callable statement to update and delete the record of employee database.**

Queries:

Create table

```
CREATE TABLE EMS (
   Id INT AUTO_INCREMENT PRIMARY KEY,
   Name VARCHAR(50),
   Salary INT,
   designation VARCHAR(50)
);
```

Delete procedure

```
DELIMITER //
CREATE PROCEDURE delete_emp (IN empid INT)
BEGIN
   DELETE FROM EMS WHERE id = empid;
END
//
```

Update Procedure

```
DELIMITER //
CREATE PROCEDURE update_emp (IN empid INT, IN updated_name VARCHAR(50),
IN updated_designation VARCHAR(50), IN updated_salary INT)
BEGIN
   UPDATE EMS SET Name = updated_name, designation = updated_designation,
Salary = updated_salary WHERE Id = empid;
END
//
```

Code:

```
import java.util.*;
```

```
import java.io.*;
import java.sql.*;

class EMS {
   Connection con;

   void updateEmployee(int id, String name, String designation, int salary) throws
SQLException {
      CallableStatement stmt = con.prepareCall("{call update_emp(?,?,?,?)}");
      stmt.setInt(1, id);
      stmt.setString(2, name);
      stmt.setString(3, designation);
      stmt.setInt(4, salary);
      stmt.execute();
      System.out.println("Record Updated");
   }

   void deleteEmployee(int empid) {
      try {
         CallableStatement stmt = con.prepareCall("{call delete_emp(?)}");
         stmt.setInt(1, empid);
         stmt.execute();
         System.out.println("Record Deleted successfully");
      } catch (SQLException e) {
         System.out.println("Error deleting record: " + e.getMessage());
      }
   }

   public static void main(String[] args) {
      try {
         EMS E = new EMS();
         Scanner sc = new Scanner(System.in);
         E.con = DriverManager.getConnection("jdbc:mysql://localhost:3306/MCA",
"root", "password");
         while (true) {
            System.out.println("\nEmployee Operation\n1.Update
Employee\n2.Delete\n3.Exit\nEnter your choice:");
            int choice = sc.nextInt();
            switch (choice) {
               case 1:
```

```
                System.out.println("Enter Employee Id:");
                int id = sc.nextInt();
                sc.nextLine(); // consume newline
                System.out.println("Enter Employee name:");
                String name = sc.nextLine();
                System.out.println("Enter Designation:");
                String desig = sc.nextLine();
                System.out.println("Enter Employee Salary:");
                int salary = sc.nextInt();
                E.updateEmployee(id, name, desig, salary);
                break;
            case 2:
                System.out.println("Enter Employee Id:");
                int empid = sc.nextInt();
                E.deleteEmployee(empid);
                break;
            case 3:
                E.con.close();
                System.exit(0);
                break;
            default:
                System.out.println("Invalid choice. Please enter a valid option.");
        }
      }
    } catch (SQLException e) {
        e.printStackTrace();
    }
  }
}
```

Output:

Add some records to the table

```
mysql> insert into EMS (name,salary,designation) values( 'Vivian',19000,'Cleark');
    -> //
Query OK, 1 row affected (0.02 sec)

mysql> insert into EMS (name,salary,designation) values( 'Ramesh',9000,'HR');
    -> //
```

```
Query OK, 1 row affected (0.02 sec)

Employee Operation
1.Update Employee
2.Delete
3.Exit
Enter your choice:
1
Enter Employee Id:
1
Enter Employee name:
vvn
Enter Designation:
Manager
Enter Employee Salary:
90000
Record Updated

Employee Operation
1.Update Employee
2.Delete
3.Exit
Enter your choice:
2
Enter Employee Id:
2
Record Deleted successfully


mysql> select * from EMS;
    -> //
+----+------+--------+-------------+
| Id | Name | Salary | designation |
+----+------+--------+-------------+
|  1 | vvn  |  90000 | Manager     |
+----+------+--------+-------------+
1 row in set (0.00 sec)
```

5. **Write a program to calculate the area of any three shapes using the concept of dynamic method dispatch.**

Code:

```
import java.util.Scanner;

abstract class Shape {
    abstract double calculateArea();
}

class Rectangle extends Shape {
    double length;
    double width;

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

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

class Circle extends Shape {
    double radius;

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

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

class Triangle extends Shape {
    double base;
    double height;

    Triangle(double base, double height) {
```

```
        this.base = base;
        this.height = height;
    }

    double calculateArea() {
        return 0.5 * base * height;
    }
}

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

        // Calculate area of rectangle
        System.out.println("Enter length and width of rectangle: ");
        double lengthR = sc.nextDouble();
        double widthR = sc.nextDouble();
        Shape rectangle = new Rectangle(lengthR, widthR);
        double areaR = rectangle.calculateArea();
        System.out.println("Area of the Rectangle: " + areaR);

        // Calculate area of circle
        System.out.println("Enter radius of circle: ");
        double radiusC = sc.nextDouble();
        Shape circle = new Circle(radiusC);
        double areaC = circle.calculateArea();
        System.out.println("Area of the Circle: " + areaC);

        // Calculate area of triangle
        System.out.println("Enter base and height of triangle: ");
        double baseT = sc.nextDouble();
        double heightT = sc.nextDouble();
        Shape triangle = new Triangle(baseT, heightT);
        double areaT = triangle.calculateArea();
        System.out.println("Area of the Triangle: " + areaT);
    }
}
```

Output:

```
Enter length and width of rectangle:
4 5
Area of the Rectangle: 20.0

Enter radius of circle:
5
Area of the Circle: 78.53981633974483

Enter base and height of triangle:
6 4
Area of the Triangle: 12.0
```

6. **Write a Java program to perform update operations on two lists using list interface methods.**

Code:

```
import java.util.*;

public class ListUpdateOperations {
    public static void main(String[] args) {
        // Create two lists
        ListUpdateOperations L = new ListUpdateOperations();
        Scanner sc = new Scanner(System.in);
        List<String> list1 = new ArrayList<>();
        List<String> list2 = new ArrayList<>();

        // Take input for list 1
        System.out.println("Enter elements for List 1 (separated by spaces):");
        String[] elementsList1 = sc.nextLine().split(" ");
        for (String element : elementsList1) {
            list1.add(element);
        }

        // Take input for list 2
        System.out.println("Enter elements for List 2 (separated by spaces):");
        String[] elementsList2 = sc.nextLine().split(" ");
        for (String element : elementsList2) {
            list2.add(element);
        }
```

```
System.out.println("List 1: " + list1);
System.out.println("List 2: " + list2);
System.out.println("Enter the list number (1 or 2) to update an element:");
int listNumber = sc.nextInt();
sc.nextLine(); // Consume newline
if (listNumber == 1) {
    if (!list1.isEmpty()) {
        System.out.println("Current List 1: " + list1);
        System.out.println("Enter the index to update (0-based):");
        int indexToUpdate = sc.nextInt();
        sc.nextLine(); // Consume newline
        if (indexToUpdate >= 0 && indexToUpdate < list1.size()) {
            System.out.println("Enter the new element:");
            String newElement = sc.nextLine();
            list1.set(indexToUpdate, newElement);
            System.out.println("Element updated in List 1.");
            System.out.println("Updated List 1: " + list1);
        } else {
            System.out.println("Invalid index.");
        }
    } else {
        System.out.println("List 1 is empty.");
    }
} else if (listNumber == 2) {
    if (!list2.isEmpty()) {
        System.out.println("Current List 2: " + list2);
        System.out.println("Enter the index to update (0-based):");
        int indexToUpdate = sc.nextInt();
        sc.nextLine(); // Consume newline
        if (indexToUpdate >= 0 && indexToUpdate < list2.size()) {
            System.out.println("Enter the new element:");
            String newElement = sc.nextLine();
            list2.set(indexToUpdate, newElement);
            System.out.println("Element updated in List 2.");
            System.out.println("Updated List 2: " + list2);
        } else {
            System.out.println("Invalid index.");
        }
    } else {
        System.out.println("List 2 is empty.");
```

```
            }
        } else {
            System.out.println("Invalid list number.");
        }
    }
}
```

Output:

```
Enter elements for List 1 (separated by spaces):
1 2 3
Enter elements for List 2 (separated by spaces):
4 5 6
List 1: [1, 2, 3]
List 2: [4, 5, 6]
Enter the list number (1 or 2) to update an element:
1
Current List 1: [1, 2, 3]
Enter the index to update (0-based):
2
Enter the new element:
4
Element updated in List 1.
Updated List 1: [1, 2, 4]
```

7. **Create class Student with data elements studno, sname, marks1, marks2, marks3, total, percentage. Create method getStudent() which will accept studno, sname, marka1, marks2, marks3. Have calculate() method which will calculate total and percentage. Have dispStudent() which will display the student details. In the client program create an object of class Student and accept details for it. Send it to a server program where you calculate the total and percentage. Display the student details in the client program. (Use TCP/IP).**

Code:

Student.java

```
import java.io.Serializable;
```

```
public class Student implements Serializable {
     int studno;
     String sname;
     int marks1;
     int marks2;
     int marks3;
     int total;
     double percentage;

    public void getStudent(int studno, String sname, int marks1, int marks2, int marks3) {
        this.studno = studno;
        this.sname = sname;
        this.marks1 = marks1;
        this.marks2 = marks2;
        this.marks3 = marks3;
    }

    public void calculate() {
        total = marks1 + marks2 + marks3;
        percentage = ((double) total / 300) * 100;
    }

    public void dispStudent() {
        System.out.println("Student Number: " + studno);
        System.out.println("Student Name: " + sname);
        System.out.println("Marks 1: " + marks1);
        System.out.println("Marks 2: " + marks2);
        System.out.println("Marks 3: " + marks3);
        System.out.println("Total: " + total);
        System.out.println("Percentage: " + percentage);
    }
}
```

StudentServer.java

```
import java.io.*;
import java.net.*;
```

```
public class StudentServer {
    public static void main(String[] args) {
        try {
            ServerSocket ss = new ServerSocket(1234);
            System.out.println("Server waiting for client...");

            Socket s = ss.accept();
            System.out.println("Client connected.");

           ObjectInputStream objin = new ObjectInputStream(s.getInputStream());
            ObjectOutputStream objout = new ObjectOutputStream(s.getOutputStream());

            Student student = (Student) objin.readObject();

            student.calculate();

            objout.writeObject(student);

            objin.close();
            objout.close();
            s.close();
            ss.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

StudentClient.java

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

public class StudentClient {
    public static void main(String[] args) {
        try {
            Socket s = new Socket("localhost", 1234);
            System.out.println("Connected to server.");
```

```
            ObjectOutputStream objout = new ObjectOutputStream(s.getOutputStream());
            ObjectInputStream objin = new ObjectInputStream(s.getInputStream());

            Scanner sc = new Scanner(System.in);
            System.out.print("Enter student number: ");
            int studno = sc.nextInt();
            sc.nextLine(); // consume newline
            System.out.print("Enter student name: ");
            String sname = sc.nextLine();
            System.out.print("Enter marks for Subject 1: ");
            int marks1 = sc.nextInt();
            System.out.print("Enter marks for Subject 2: ");
            int marks2 = sc.nextInt();
            System.out.print("Enter marks for Subject 3: ");
            int marks3 = sc.nextInt();

            Student student = new Student();
            student.getStudent(studno, sname, marks1, marks2, marks3);
            objout.writeObject(student);

            student = (Student) objin.readObject();
            student.dispStudent();

            objout.close();
            objin.close();
            s.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Output:

In first Terminal

PS D:\MCA\Programs\TCPIP> javac Student.java
PS D:\MCA\Programs\TCPIP> javac StudentServer.java
PS D:\MCA\Programs\TCPIP> java StudentServer

Server waiting for client...
Client connected.

In second Terminal
PS D:\MCA\Programs\TCPIP> javac StudentClient.java
PS D:\MCA\Programs\TCPIP> java StudentClient
Connected to server.
Enter student number: 177
Enter student name: Vivian
Enter marks for Subject 1: 99
Enter marks for Subject 2: 98
Enter marks for Subject 3: 96
Student Number: 177
Student Name: Vivian
Marks 1: 99
Marks 2: 98
Marks 3: 96
Total: 293
Percentage: 97.66666666666667

8. **Create a program to demonstrate the different situations where finally block can be used.**

Code:

```
import java.io.*;

public class FinallyDemo {

    // Situation 1: Handling exceptions
    public static void divideNumbers(int a, int b) {
        try {
            int result = a / b;
            System.out.println("Division result: " + result);
        } catch (ArithmeticException e) {
            System.out.println("Error: Cannot divide by zero!");
        } finally {
            System.out.println("Finally block executed.");
        }
    }
```

```
// Situation 2: Resource cleanup (e.g., file handling)
public static void openFile(String filename) {
    BufferedReader reader = null;
    try {
        reader = new BufferedReader(new FileReader(filename));
        System.out.println("File opened successfully.");
    } catch (IOException e) {
        System.out.println("Error: File not found!");
    } finally {
        try {
            if (reader != null)
                reader.close();
            System.out.println("File closed.");
        } catch (IOException e) {
            System.out.println("Error: Unable to close the file.");
        }
    }
}

// Situation 3: Cleanup operations (e.g., closing database connection)
public static void databaseOperation() {
    try {
        // Simulating some database operation
        System.out.println("Performing database operation...");
        throw new Exception("Database operation failed!");
    } catch (Exception e) {
        System.out.println("Error: " + e.getMessage());
    } finally {
        System.out.println("Closing database connection.");
    }
}

public static void main(String[] args) {
    // Situation 1: Handling exceptions
    divideNumbers(10, 0);

    // Situation 2: Resource cleanup (e.g., file handling)
    openFile("example.txt");

    // Situation 3: Cleanup operations (e.g., closing database connection)
```

```
        databaseOperation();
    }
}
```

Output:

```
ERROR!
Error: Cannot divide by zero!
Finally block executed.

ERROR!
Error: File not found!
File closed.
Performing database operation…

ERROR!
Error: Database operation failed!
Closing database connection.
```

9. Create a simple Java chat application using UDP protocol.

UDPChatServer.java

Code:

```
import java.io.*;
import java.net.*;

public class UDPChatServer {
    public static void main(String[] args) {
        try {
            DatagramSocket socket = new DatagramSocket(9876);

            while (true) {
                byte[] receiveData = new byte[1024];
                DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
                socket.receive(receivePacket);
```

```
            String clientMessage = new String(receivePacket.getData(), 0,
receivePacket.getLength());
            InetAddress clientAddress = receivePacket.getAddress();
            int clientPort = receivePacket.getPort();

            System.out.println("Client (" + clientAddress.getHostAddress() + ":" + clientPort + "): "
+ clientMessage);

            BufferedReader serverReader = new BufferedReader(new
InputStreamReader(System.in));
            System.out.print("Server: ");
            String serverMessage = serverReader.readLine();
            byte[] sendData = serverMessage.getBytes();

            DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length,
clientAddress, clientPort);
            socket.send(sendPacket);
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
  }
}
```

UDPChatClient.java

```
import java.io.*;
import java.net.*;

public class UDPChatClient {
   public static void main(String[] args) {
      try {
         DatagramSocket socket = new DatagramSocket();

         InetAddress serverAddress = InetAddress.getByName("localhost");
         int serverPort = 9876;

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

         while (true) {
            System.out.print("You: ");
            String message = reader.readLine();
            byte[] sendData = message.getBytes();
```

```
            DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length,
serverAddress, serverPort);
            socket.send(sendPacket);

            byte[] receiveData = new byte[1024];
            DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
            socket.receive(receivePacket);

            String serverMessage = new String(receivePacket.getData(), 0,
receivePacket.getLength()).trim();
            System.out.println("Server: " + serverMessage);
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
  }
}
```

Output:

First Compile and run UDPChatServer.java

```
PS D:\MCA\Programs> javac UDPChatServer.java
PS D:\MCA\Programs> java UDPChatServer
Client (127.0.0.1:52989): Hii Server..
Server: May I know who is this?
Client (127.0.0.1:52989): I am Vivian from 1st MCA.
Server: Okay
Client (127.0.0.1:52989): See you later
Server: Bye
```

Then Compile and run UDPChatClient.java

```
PS D:\MCA\Programs> javac UDPChatClient.java
PS D:\MCA\Programs> java UDPChatClient
You: Hii Server..
Server: May I know who is this?
You: I am Vivian from 1st MCA.
Server: Okay
You: See you later
Server: Bye
You:
```

10. Display the contents of any URL. Display the host name as well as the port at which it is listening as well as the protocol used.

Code:

```
import java.io.*;
import java.net.*;

class DisplayURl {
    public static void main(String[] args) {
        try {
            URL url = new URL("http://172.16.2.10/index.html/");
            URLConnection conn = url.openConnection();

            System.out.println("Host name: " + url.getHost());
            System.out.println("Port: " + url.getPort());
            System.out.println("Protocol: " + url.getProtocol());

           // Read the content from the URL
            BufferedReader r = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
            String line;

            System.out.println("Content:");
            while ((line = r.readLine()) != null) {
                System.out.println(line);
            }
            r.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Sample Output:

```
Host name: 172.16.2.10
Port: -1
Protocol: http
Content:
<!DOCTYPE html>
<html>
<head>
```

```
    <title>Sample Page</title>
</head>
<body>
    <h1>Hello, World!</h1>
    <p>This is a sample HTML page.</p>
</body>
</html>
```

11. Write a program to demonstrate the use of throw keyword in the following cases.

i. To throw built-in exception

ii. To throw user-defined exception.

Code:

```
import java.util.*;

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

public class ExceptionDemo {

    void built_in_exception() {
        try {
            int x = 10 / 0;
        } catch (ArithmeticException e) {
            System.out.println(e);
        }
    }

    void UserDefinedException(int num) throws CustomException {
        if (num < 0) {
            // Throw CustomException explicitly
            throw new CustomException("Negative numbers are not allowed");
        } else {
            System.out.println("No exception thrown. Number is: " + num);
        }
```

```
    }

    public static void main(String[] args) {
        ExceptionDemo demo = new ExceptionDemo();

        // Call built_in_exception method
        System.out.println("Calling built_in_exception method:");
        demo.built_in_exception();
        System.out.println();

        // Call UserDefinedException method
        System.out.println("Calling UserDefinedException method:");
        try {
            demo.UserDefinedException(-5); // This will throw CustomException
        } catch (CustomException e) {
            System.out.println("Caught user-defined exception: " + e.getMessage());
        }
    }
}
```

Output:

```
Calling built_in_exception method:
java.lang.ArithmeticException: / by zero

Calling UserDefinedException method:
Caught user-defined exception: Negative numbers are not allowed
```

12. Accept empno, ename, basic into local variables on the client side. Pass the basic to remote methods da(), hra(), net() which will receive basic and calculate da. Hra and net and return them to the client. Display empo, ename, basic, da, hra, net on the client side. (Use RMI).

Code:

SalaryCalculation.java

```
import java.rmi.RemoteException;
import java.rmi.Remote;

interface SalaryCalculation extends Remote {
```

```
    double CalculateDA(double basic) throws RemoteException;
    double CalculateHRA(double basic) throws RemoteException;
    double CalculateNet(double basic) throws RemoteException;
}
```

CalculationImplementation.java

```
import java.rmi.server.UnicastRemoteObject;
import java.rmi.RemoteException;

class CalculationImplementation extends UnicastRemoteObject implements SalaryCalculation {
    CalculationImplementation() throws RemoteException {
        super();
    }

    public double CalculateDA(double basic) {
        return basic * 0.08;
    }

    public double CalculateHRA(double basic) {
        return basic * 0.1;
    }

    public double CalculateNet(double basic) {
        return basic + CalculateDA(basic) + CalculateHRA(basic);
    }
}
```

EmployeeClient .java

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

public class EmployeeClient {
    public static void main(String[] args) {
        try {
            Scanner sc = new Scanner(System.in);
            SalaryCalculation stub = (SalaryCalculation)
Naming.lookup("rmi://localhost:5001/employee");

            System.out.println("Enter EMP no");
            String empNo = sc.nextLine();
            System.out.println("Enter Emp Name");
```

```
            String empName = sc.nextLine();
            System.out.println("Enter Basic salary");
            double basic = sc.nextDouble();

            double da, hra, net;

            da = stub.CalculateDA(basic);
            hra = stub.CalculateHRA(basic);
            net = stub.CalculateNet(basic);

            System.out.println("Emp No : " + empNo + "\tEmp Name : " + empName + "\tBasic : " +
basic);
            System.out.println("Da : " + da + "\tHRA : " + hra + "\tNet Salary : " + net);
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

EmployeeServer.java

```
import java.rmi.*;

public class EmployeeServer {
    public static void main(String[] args){
        try {
            // Start RMI registry on port 5001
            java.rmi.registry.LocateRegistry.createRegistry(5001);

            // Create remote object
            SalaryCalculation stub = new CalculationImplementation();

            // Bind the remote object to the registry
            Naming.rebind("rmi://localhost:5001/employee", stub);

            System.out.println("Server ready");
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

Steps to Run:

1.Compile Java Files:

javac SalaryCalculation.java
javac CalculationImplementation.java
javac EmployeeClient.java
javac EmployeeServer.java

2.Start RMI Registry:

start rmiregistry

3.Run Server Application:

PS D:\MCA\Programs\RMI> java EmployeeServer
Server ready

4.Run Client Application:

PS D:\MCA\Programs\RMI> java EmployeeClient
Enter EMP no
123
Enter Emp Name
Vivian
Enter Basic salary
12000
Emp No : 123 Emp Name : Vivian Basic : 12000.0
Da : 960.0 HRA : 1200.0 Net Salary : 14160.0

13. Write a program to calculate DA, HRA, PF, IT and Net of an employee using the concept of Association and Aggregation.

Meaning:

Association: It's like saying two classes or objects are related somehow, but they can exist independently of each other. For instance, a `Student` class may be associated with a `Course` class because students enroll in courses.

Aggregation: It's when one class contains another class as a part. For example, a `Car` class may contain `Wheel` objects. The wheels are part of the car, but they can also exist separately or be shared among different cars.

Code:

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

```
class Employee {
    private String name;
    private double basicSalary;

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

    public double getBasicSalary() {
        return basicSalary;
    }
}

class SalaryCalculator {
     private Employee employee; // Association: SalaryCalculator has a reference to an
Employee

    public SalaryCalculator(Employee employee) {
        this.employee = employee;
    }

    public double calculateDA() {
        // DA is 25% of basic salary
        return 0.25 * employee.getBasicSalary();
    }

    public double calculateHRA() {
        // HRA is 15% of basic salary
        return 0.15 * employee.getBasicSalary();
    }

    public double calculatePF() {
        // PF is 12% of basic salary
        return 0.12 * employee.getBasicSalary();
    }

    public double calculateIT() {
        // Assuming a flat rate of 10% for simplicity
```

```
        return 0.10 * (employee.getBasicSalary() + calculateDA() + calculateHRA());
    }

    public double calculateNetSalary() {
        // Net Salary = Basic Salary + DA + HRA - PF - IT
                return employee.getBasicSalary() + calculateDA() + calculateHRA() -
calculatePF() - calculateIT();
    }
}

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

        // Input employee details
        System.out.print("Enter employee name: ");
        String name = scanner.nextLine();
        System.out.print("Enter basic salary: ");
        double basicSalary = scanner.nextDouble();

        // Create an employee
        Employee employee = new Employee(name, basicSalary);

        // Create a SalaryCalculator for the employee
           SalaryCalculator calculator = new SalaryCalculator(employee); // Aggregation:
SalaryCalculator aggregates an Employee

        // Calculate and print salary components
        System.out.println("\nCalculating Salary Components...");
        System.out.println("DA: " + calculator.calculateDA());
        System.out.println("HRA: " + calculator.calculateHRA());
        System.out.println("PF: " + calculator.calculatePF());
        System.out.println("IT: " + calculator.calculateIT());
        System.out.println("Net Salary: " + calculator.calculateNetSalary());

    }
}
```

Output:

```
Enter employee name: Vivian
Enter basic salary: 12000

Calculating Salary Components...
DA: 3000.0
HRA: 1800.0
PF: 1440.0
IT: 1680.0
Net Salary: 13680.0
```

14. Write a JDBC program to Accept empno, ename and basic .. Calculate da, hra and net and add the record into the database . Finally display all the records.

Create a Table:

```
CREATE TABLE employee (
    empno INT PRIMARY KEY,
    ename VARCHAR(50),
    basic INT,
    da DOUBLE,
    hra DOUBLE,
    net DOUBLE
);
```

Code:

```
import java.sql.*;
import java.util.*;

public class EmployeeRecords {
    public static void main(String[] args) {
        try {
            Scanner sc = new Scanner(System.in);
     Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/MCA",
"root", "password");

            System.out.println("Enter Employee Number:");
            int empno = sc.nextInt();
            sc.nextLine(); // Consume newline character
```

```
        System.out.println("Enter Employee Name:");
        String ename = sc.nextLine();

        System.out.println("Enter Basic Salary:");
        int basic = sc.nextInt();

        // Calculate allowances and net salary
        double da = basic * 0.1; // Assuming DA is 10% of basic salary
        double hra = basic * 0.2; // Assuming HRA is 20% of basic salary
        double net = basic + da + hra;

        // Prepare a SQL statement to insert the record
            String sql = "INSERT INTO employee (empno, ename, basic, da, hra, net)
VALUES (?, ?, ?, ?, ?, ?)";
        PreparedStatement pstmt = con.prepareStatement(sql);
        pstmt.setInt(1, empno);
        pstmt.setString(2, ename);
        pstmt.setInt(3, basic);
        pstmt.setDouble(4, da);
        pstmt.setDouble(5, hra);
        pstmt.setDouble(6, net);

        // Execute the SQL statement to insert the record
        int rowsAffected = pstmt.executeUpdate();
        if (rowsAffected > 0) {
            System.out.println("Employee Record added successfully");
        } else {
            System.out.println("Failed to add employee record");
        }

        // Displaying all records
          ResultSet resultSet = con.createStatement().executeQuery("SELECT * FROM
employee");
        while (resultSet.next()) {
            System.out.println("Employee Number: " + resultSet.getInt("empno"));
            System.out.println("Employee Name: " + resultSet.getString("ename"));
            System.out.println("Basic Salary: " + resultSet.getInt("basic"));
            System.out.println("DA: " + resultSet.getDouble("da"));
            System.out.println("HRA: " + resultSet.getDouble("hra"));
```

```
            System.out.println("Net Salary: " + resultSet.getDouble("net"));
            System.out.println();
        }

        // Close the connection
        con.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
  }
}
```

Output:

```
Enter Employee Number:
12345
Enter Employee Name:
Vivian
Enter Basic Salary:
12000
Employee Record added successfully

Employee Number: 12345
Employee Name: Vivian
Basic Salary: 12000
DA: 1200.0
HRA: 2400.0
Net Salary: 15600.0
```

15. **Create an interface Bank with method readCustomerInfo(). Using interface inheritance, create interfaces ICICI and Axis with methods calculateInterest() and displayDetails(). Create a class to implement the interfaces.**

Code:

```
import java.util.Scanner;

interface Bank {
   void readCustomerInfo();
```

```
}

interface ICICI extends Bank {
   void calculateInterest();
}

interface AXIS extends Bank {
   void displayDetails();
}

class BankImpl implements ICICI, AXIS {
   String customerName;
   int balance;
   double interest;

   Scanner sc = new Scanner(System.in);

   public void readCustomerInfo() {
      System.out.println("Enter Customer Name:");
      customerName = sc.nextLine();
      System.out.println("Enter Balance:");
      balance = sc.nextInt();
   }

   public void calculateInterest() {
      interest = balance * 0.05; // Assuming interest rate is 5%
   }

   public void displayDetails() {
      System.out.println("Customer Name: " + customerName);
      System.out.println("Balance: " + balance);
      System.out.println("Interest: " + interest);
   }

   public static void main(String[] args) {
      BankImpl B = new BankImpl();
      B.readCustomerInfo();
      B.calculateInterest();
      B.displayDetails();
   }
```

```
}
```

Output:

```
Enter Customer Name:
Vivian
Enter Balance:
5000
Customer Name: Vivian
Balance: 5000
Interest: 250.0
```

16. Write a Java program to perform union, intersection and difference operations on two sets using set interface methods.

Code:

```
import java.util.*;

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

        // Input for set1
        System.out.println("Enter elements for set1 (separated by spaces):");
        String[] input1 = scanner.nextLine().split(" ");
        Set<Integer> set1 = new HashSet<>();
        for (String str : input1) {
            set1.add(Integer.parseInt(str));
        }

        // Input for set2
        System.out.println("Enter elements for set2 (separated by spaces):");
        String[] input2 = scanner.nextLine().split(" ");
        Set<Integer> set2 = new HashSet<>();
        for (String str : input2) {
            set2.add(Integer.parseInt(str));
        }

        // Union operation
        Set<Integer> union = new HashSet<>(set1);
        union.addAll(set2);
        System.out.println("Union: " + union);

        // Intersection operation
        Set<Integer> intersection = new HashSet<>(set1);
        intersection.retainAll(set2);
        System.out.println("Intersection: " + intersection);

        // Difference operation (set1 - set2)
        Set<Integer> difference = new HashSet<>(set1);
```

```
        difference.removeAll(set2);
        System.out.println("Difference (set1 - set2): " + difference);

        // Difference operation (set2 - set1)
        difference = new HashSet<>(set2);
        difference.removeAll(set1);
        System.out.println("Difference (set2 - set1): " + difference);

        scanner.close();
    }
}
```

Output:

```
Enter elements for set1 (separated by spaces):
1 2 3 4 5
Enter elements for set2 (separated by spaces):
4 5 6 7 8
Union: [1, 2, 3, 4, 5, 6, 7, 8]
Intersection: [4, 5]
Difference (set1 - set2): [1, 2, 3]
Difference (set2 - set1): [6, 7, 8]
```

17. Write a Java Program to create a chatting application using threads.

Code:

ChatServer.java

```
import java.net.*;
import java.io.*;

public class ChatServer{
        public static void main(String[] args) {
         try {
            DatagramSocket socket = new DatagramSocket(9876);

            while (true) {
                byte[] receiveData = new byte[1024];
```

```
            DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
            socket.receive(receivePacket);

            String clientMessage = new String(receivePacket.getData(), 0,
receivePacket.getLength());
            InetAddress clientAddress = receivePacket.getAddress();
            int clientPort = receivePacket.getPort();

            System.out.println("Client (" + clientAddress.getHostAddress() + ":" + clientPort +
"): " + clientMessage);

            BufferedReader serverReader = new BufferedReader(new
InputStreamReader(System.in));
            System.out.print("Server: ");
            String serverMessage = serverReader.readLine();
            byte[] sendData = serverMessage.getBytes();

            DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, clientAddress, clientPort);
            socket.send(sendPacket);
        }
      } catch (IOException e) {
        e.printStackTrace();
      }
   }


}
```

ChatClient.java

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

class ClientChatThread extends Thread{
```

```
    public void run(){
        try {
            DatagramSocket socket = new DatagramSocket();

            InetAddress serverAddress = InetAddress.getByName("localhost");
            int serverPort = 9876;

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

            while (true) {
                System.out.print("You: ");
                String message = reader.readLine();
                byte[] sendData = message.getBytes();

                DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, serverAddress, serverPort);
                socket.send(sendPacket);

                byte[] receiveData = new byte[1024];
                DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
                socket.receive(receivePacket);

                String serverMessage = new String(receivePacket.getData(), 0,
receivePacket.getLength()).trim();
                System.out.println("Server: " + serverMessage);
            }

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

public class ChatClient {
```

```
    public static void main(String[] args) {
    ClientChatThread ct = new ClientChatThread();
    ct.start();
  }
}
```

Output:

In first terminal

```
PS D:\MCA\Programs\TCPIP> javac ChatServer.java
PS D:\MCA\Programs\TCPIP> java ChatServer
Client (127.0.0.1:63463): Hello Bro
Server: Hii Bhai
Client (127.0.0.1:63463): How are You?
Server: Fine.WBU?
Client (127.0.0.1:63463):Fine
Server:
```

In Second terminal

```
PS D:\MCA\Programs\TCPIP> javac ChatClient.java
PS D:\MCA\Programs\TCPIP> java ChatClient
You: Hello Bro
Server: Hii Bhai
You: How are You?
Server: Fine.WBU?
You:Fine
```

18. Write a program to perform banking operations using threads.

Note: You can write the code of Question 19 to answer this question But you can't write this code to Inter-thread communication because it should have notify() wait() methods

Code:

```
import java.util.Scanner;

class BankAccount {
  private int balance;
```

```
    public BankAccount(int initialBalance) {
        balance = initialBalance;
    }

    public synchronized void deposit(int amount) {
        balance += amount;
        System.out.println("Deposit: " + amount + " Balance: " + balance);
    }

    public synchronized void withdraw(int amount) {
        if (balance >= amount) {
            balance -= amount;
            System.out.println("Withdrawal: " + amount + " Balance: " + balance);
        } else {
            System.out.println("Insufficient funds for withdrawal.");
        }
    }

    public int getBalance() {
        return balance;
    }
}

class Transaction implements Runnable {
    private BankAccount account;
    private int amount;
    private boolean isDeposit;

    public Transaction(BankAccount account, int amount, boolean isDeposit) {
        this.account = account;
        this.amount = amount;
        this.isDeposit = isDeposit;
    }

    @Override
    public void run() {
```

```
        if (isDeposit) {
            account.deposit(amount);
        } else {
            account.withdraw(amount);
        }
    }
}

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

        System.out.print("Enter initial balance: ");
        int initialBalance = scanner.nextInt();
        BankAccount account = new BankAccount(initialBalance);

        while (true) {
            System.out.println("Choose operation:");
            System.out.println("1. Deposit");
            System.out.println("2. Withdraw");
            System.out.println("3. Exit");

            int choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    System.out.print("Enter deposit amount: ");
                    int depositAmount = scanner.nextInt();
                    Thread depositThread = new Thread(new Transaction(account,
depositAmount, true));
                    depositThread.start();
                    break;
                case 2:
                    System.out.print("Enter withdrawal amount: ");
                    int withdrawalAmount = scanner.nextInt();
                    Thread withdrawThread = new Thread(new Transaction(account,
withdrawalAmount, false));
```

```
                withdrawThread.start();
                break;
            case 3:
            System.exit(0);
            default:
                System.out.println("Invalid choice.");
        }
    }

  }
}
```

Output:

```
PS D:\MCA\Programs\JFrame> javac Main.java
PS D:\MCA\Programs\JFrame> java Main
Enter initial balance: 500
Choose operation:
1. Deposit
2. Withdraw
3. Exit
1
Enter deposit amount: 1000
Choose operation:
1. Deposit
2. Withdraw
3. Exit
Deposit: 1000 Balance: 1500
2
Enter withdrawal amount: 1000
Choose operation:
1. Deposit
2. Withdraw
3. Exit
Withdrawal: 1000 Balance: 500
```

19. Write a program to demonstrate Inter-thread communication

NOTE: You can write the same code for Question 18 (banking operations using threads), Question 19 (inter-thread communication), and Question 28 (thread synchronization). But please ensure that you understand the concepts so that if asked, you can answer them.

Thread Synchronization:

Synchronized blocks are used to ensure that only one thread can access critical sections of code, such as deposit and withdrawal operations, at a time. This prevents conflicts and maintains consistency in the account balance.

Inter-thread Communication:

Using `wait()`, `notify()`, and `notifyAll()` methods helps threads communicate effectively.

Code:

```
class Customer {
    int amount = 10000;

    synchronized void withdraw(int amount) {
        System.out.println("going to withdraw...");
        if (this.amount < amount) {
            System.out.println("Less balance; waiting for deposit...");
            try {
                wait();
            } catch (InterruptedException e) {}
        }
        this.amount -= amount;
        System.out.println("Withdraw completed...");
    }

    synchronized void deposit(int amount) {
        System.out.println("going to deposit...");
        this.amount += amount;
        System.out.println("Deposit completed... ");
        notify();
    }
```

```
}

class Test {
    public static void main(String args[]) {
        final Customer c = new Customer();

        new Thread() {
            public void run() {
                c.withdraw(15000);
            }
        }.start();

        new Thread() {
            public void run() {
                c.deposit(10000);
            }
        }.start();
    }
}
```

Output:

```
PS D:\MCA\Programs> javac Test.java
PS D:\MCA\Programs> java Test
going to withdraw...
Less balance; waiting for deposit...
going to deposit...
Deposit completed...
Withdraw completed...
```

20. Write a program to copy the contents of one file to another using Byte Stream classes.

NOTE: Create a file with name source.txt with some data before running the program

Code:

```
import java.io.*;
public class FileCopy {
    public static void main(String[] args) {
        String sourceFile = "source.txt"; // Path to the source file
        String targetFile = "target.txt"; // Path to the target file

        try {
            // Create FileInputStream to read from the source file
            FileInputStream fis = new FileInputStream(sourceFile);

            // Create FileOutputStream to write to the target file
            FileOutputStream fos = new FileOutputStream(targetFile);

            int bytesRead;
            byte[] buffer = new byte[1024]; // Buffer to hold bytes read from the source file
            while ((bytesRead = fis.read(buffer)) != -1) {
                fos.write(buffer, 0, bytesRead); // Write bytes to the target file
            }
            fis.close();
            fos.close();
           System.out.println("File copied successfully.");
        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

Output:

```
PS D:\MCA\Programs> javac FileCopy.java
PS D:\MCA\Programs> java FileCopy
File copied successfully.
```

21. Write a program to count the number of vowels, consonants and digits in a file.

Note: **Create a file with name input.txt with some data before running the program**

Code:

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

public class CountCharsInFile {
    public static void main(String[] args) {
        String fileName = "input.txt";

        int vowels = 0;
        int consonants = 0;
        int digits = 0;

        try {
            // Open the file
            File file = new File(fileName);
            Scanner sc = new Scanner(file);

            // Read each character from the file
            while (sc.hasNext()) {
                String line = sc.nextLine().toLowerCase(); // Convert to lowercase for
case-insensitive comparison
                for(int i=0; i<line.length(); i++) {
                    char ch = line.charAt(i);
                    // Check if the character is a vowel, consonant, or digit
                    if (Character.isLetter(ch)) {
                        if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {
                            vowels++;
                        } else {
                            consonants++;
```

```
                }
            } else if (Character.isDigit(ch)) {
                digits++;
            }
        }
    }
    System.out.println("Number of vowels: " + vowels);
    System.out.println("Number of consonants: " + consonants);
    System.out.println("Number of digits: " + digits);
} catch (FileNotFoundException e) {
    System.out.println("File not found: " + fileName);
}
  }
}
```

Output:

Number of vowels: 34

Number of consonants: 62

Number of digits: 10

22. Write a program to copy the contents of one file to another using Character Stream classes.

Code:

```
import java.io.*;

public class FileCopy {
    public static void main(String[] args) {
        String sourceFile = "source.txt"; // Path to the source file
        String targetFile = "target.txt"; // Path to the target file

        try (FileReader reader = new FileReader(sourceFile);
             FileWriter writer = new FileWriter(targetFile)) {
            int character;
```

```
            while ((character = reader.read()) != -1) {
                writer.write(character);
            }

            System.out.println("File copied successfully.");
        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

Output:

```
PS D:\MCA\Programs> javac FileCopy.java
PS D:\MCA\Programs> java FileCopy
File copied successfully.
```

23. Write a program to demonstrate Key board events using JFrame

Note: The main difference lies in how keyboard events are managed. In the program for Question 23 (Keyboard events using JFrame), all methods of the `KeyListener` interface must be implemented, even if they're not used (we have to provide the definition). Meanwhile, in the program for Question 24 (Keyboard events using Adapter class), this process is streamlined by using the `KeyAdapter` class, which requires implementing only the necessary methods instead of all of them. This approach enhances readability and reduces unnecessary code. Rest all is almost same

Code:

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

public class KeyPrinter extends JFrame implements KeyListener {
    JTextArea textArea;
```

```
    public KeyPrinter() {
        setTitle("Key Printer");
        setSize(300, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        textArea = new JTextArea(10, 20);
        textArea.setEditable(false);
        textArea.setFocusable(true);
        textArea.addKeyListener(this);

        add(textArea);

        setVisible(true);
    }

    public void keyPressed(KeyEvent e) {
        char keyChar = e.getKeyChar();
        textArea.append("Key Pressed: " + keyChar + "\n");
    }

    public void keyReleased(KeyEvent e) {
        // Do nothing for key release
    }

    public void keyTyped(KeyEvent e) {
        // Do nothing for key type
    }

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

24. Write a program to demonstrate Key board events using Adapter class.

Note: The main difference lies in how keyboard events are managed. In the program for Question 23 (Keyboard events using JFrame), all methods of the `KeyListener` interface must be implemented, even if they're not used (we have to provide the definition). Meanwhile, in the program for Question 24 (Keyboard events using Adapter class), this process is streamlined by using the `KeyAdapter` class, which requires implementing only the necessary methods instead of all of them. This approach enhances readability and reduces unnecessary code. Rest all is almost same

Code:

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

public class AdapterKeyEvent extends JFrame {
    JLabel statusLabel; // Label to display status

    public AdapterKeyEvent() {
        setTitle("Adapter Key Events Example");
        setSize(300, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new FlowLayout());

        // Set the frame to focusable and request focus
        setFocusable(true);
        requestFocus();

        // Initialize status label
        statusLabel = new JLabel("Press a key...");
        add(statusLabel);

        // Register the frame to listen for keyboard events
        addKeyListener(new KeyAdapter() {
            public void keyPressed(KeyEvent ke) {
                statusLabel.setText("Key Down");
            }

            public void keyReleased(KeyEvent ke) {
                statusLabel.setText("Key Up");
```

```
        }
    });
  }

  public static void main(String[] args) {

      AdapterKeyEvent frame = new AdapterKeyEvent();
      frame.setVisible(true);
    };
}
```

25. Write a program to find factorial of a number using swing components.

Code:

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

public class FactorialCalculator extends JFrame {
  JTextField inputField;
  JTextArea resultArea;

  public FactorialCalculator() {
    setTitle("Factorial Calculator");
    setSize(300, 200);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setLayout(new FlowLayout());

    inputField = new JTextField(10);
    JButton calculateButton = new JButton("Calculate");
    calculateButton.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        calculateFactorial();
      }
    });

    resultArea = new JTextArea(5, 20);
    resultArea.setEditable(false);

    add(new JLabel("Enter a number: "));
    add(inputField);
    add(calculateButton);
    add(resultArea);
```

```
    }

    private void calculateFactorial() {
        String input = inputField.getText();
        try {
            int number = Integer.parseInt(input);
            long factorial = 1;
            for (int i = 1; i <= number; i++) {
                factorial *= i;
            }
            resultArea.setText("Factorial of " + number + " is: " + factorial);
        } catch (NumberFormatException ex) {
            resultArea.setText("Please enter a valid integer.");
        }
    }

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

26. Write a program to display first N fibonacci numbers in a text area

Code:

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

public class Fibonacci extends JFrame {
    JTextField inputField;
    JTextArea resultArea;

    public Fibonacci() {
        setTitle("Fibonacci Numbers");
        setSize(300, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new FlowLayout());

        inputField = new JTextField(10);
        JButton calculateButton = new JButton("Calculate");
```

```
    JButton clearButton = new JButton("Clear"); // Renamed to clearButton for clarity
    calculateButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            String input = inputField.getText();
            int n = Integer.parseInt(input);

            int f1 = 0, f2 = 1;
            for (int i = 0; i < n; i++) {
                resultArea.append(f1 + " ");
                int f3 = f1 + f2;
                f1 = f2;
                f2 = f3;
            }
        }
    });

    clearButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            // Clear the resultArea text
            resultArea.setText("");
        }
    });

    resultArea = new JTextArea(5, 20);
    resultArea.setEditable(false);

    add(new JLabel("Enter a number: "));
    add(inputField);
    add(calculateButton);
    add(clearButton); // Add Clear button
    add(new JScrollPane(resultArea));
}

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

```
}
```

27. Write a program to concatenate two strings using swing components.

Code:

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

public class StringConcat extends JFrame {
    JTextField inputField1, inputField2;
    JTextArea resultArea;
    JLabel label1,label2;

    public StringConcat() {
        setTitle("String Concatenator");
        setSize(400, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new FlowLayout());

        inputField1 = new JTextField(10);
        inputField2 = new JTextField(10);
        JButton concatenateButton = new JButton("Concatenate");
        concatenateButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {

                    concatenateStrings();

            }
        });

        resultArea = new JTextArea();
        resultArea.setEditable(false);
```

```
        label1 = new JLabel("Enter first string: ");
        add(label1);
        add(inputField1);
        label2 = new JLabel("Enter Seond string: ");
        add(label2);
        add(inputField2);
        add(concatenateButton);
        add(resultArea);
    }

    private void concatenateStrings(){
        String input1 = inputField1.getText();
        String input2 = inputField2.getText();
        String concatenatedString = input1 + input2;

        resultArea.setText("Concatenated string: " + concatenatedString);

    }


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

28. Write a program to Demonstrate thread synchronization.

Note: You can write the code of either Question 19 or 18 to answer this question

Code:

```
class Counter {
    private int count = 0;
```

```
    public synchronized void increment() {
    count++;
    }

    public synchronized void decrement() {
    count--;
    }

    public synchronized int getCount() {
    return count;
    }
}

class IncrementThread extends Thread {
    private Counter counter;

    public IncrementThread(Counter counter) {
    this.counter = counter;
    }

    public void run() {
    for (int i = 0; i < 5; i++) {
       counter.increment();
        System.out.println("Incremented: " + counter.getCount());
    }
    }
}

class DecrementThread extends Thread {
    private Counter counter;

    public DecrementThread(Counter counter) {
    this.counter = counter;
    }

    public void run() {
    for (int i = 0; i < 5; i++) {
       counter.decrement();
        System.out.println("Decremented: " + counter.getCount());
    }
    }
}

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

```
        Counter counter = new Counter();

        IncrementThread incrementThread = new IncrementThread(counter);
        DecrementThread decrementThread = new DecrementThread(counter);

        incrementThread.start();
        decrementThread.start();

        try {
            incrementThread.join();
            decrementThread.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        System.out.println("Final Count: " + counter.getCount());
        }
}
```

Output:

Incremented: 1

Incremented: 1

Incremented: 2

Incremented: 3

Incremented: 4

Decremented: 0

Decremented: 3

Decremented: 2

Decremented: 1

Decremented: 0

Final Count: 0

29. Write a program to Demonstrate object serialization and de-serialization.

Meaning:

Serialized:

- Serialized refers to the process of converting an object into a stream of bytes, which can then be stored in a file, transmitted over a network, or saved in a database. This allows the object's state to be preserved and reconstructed later.

Deserialized:

- Deserialized refers to the process of reconstructing an object from its serialized form (a stream of bytes). During deserialization, the byte stream is read and used to recreate the original object, restoring its state to what it was before serialization.

Code:

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

class Person implements Serializable {
	private String name;
	private int age;

	public Person(String name, int age) {
	this.name = name;
	this.age = age;
	}

	public String toString() {
	return "Name: " + name + ", Age: " + age;
	}
}

public class SerializationDemo {
	public static void serializeObject() {
	Scanner scanner = new Scanner(System.in);

	// Prompt user for input
	System.out.print("Enter name: ");
	String name = scanner.nextLine();
	System.out.print("Enter age: ");
	int age = scanner.nextInt();

	// Create Person object
	Person person = new Person(name, age);

	// Serialize object
	try (FileOutputStream fileOut = new FileOutputStream("person.ser");
	   ObjectOutputStream objectOut = new ObjectOutputStream(fileOut)) {
	   objectOut.writeObject(person);
	   System.out.println("Serialized Object: " + person);
	   System.out.println("Object serialized successfully.");
	} catch (IOException e) {
	   e.printStackTrace();
```

```
        }

        scanner.close();
        }

        public static Person deserializeObject() {
        Person person = null;
        try (FileInputStream fileIn = new FileInputStream("person.ser");
            ObjectInputStream objectIn = new ObjectInputStream(fileIn)) {
            person = (Person) objectIn.readObject();
             System.out.println("Deserialized Object: " + person);
            System.out.println("Object deserialized successfully.");
        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
        }
        return person;
        }

        public static void main(String[] args) {
        // Serialize object
        serializeObject();

        // Deserialize object
        deserializeObject();
        }
}
```

Output:

```
PS D:\MCA\Programs> javac SerializationDemo.java
PS D:\MCA\Programs> java SerializationDemo
Enter name: Vivian
Enter age: 23
Serialized Object: Name: Vivian, Age: 23
Object serialized successfully.
Deserialized Object: Name: Vivian, Age: 23
Object deserialized successfully.
```

30. Write a program to Demonstrate static and dynamic polymorphism.

Code:

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

```
class Animal {
    public void makeSound() {
        System.out.println("The animal makes a sound");
    }
}

class Dog extends Animal {
    public void makeSound() {
        System.out.println("The dog barks");
    }
}

class DynamicDemo {
    void add(int a, int b) {
        System.out.println("Sum of integers: " + (a + b));
    }

    void add(double a, double b) {
        System.out.println("Sum of doubles: " + (a + b));
    }
}

public class Main1 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        DynamicDemo D = new DynamicDemo();

        System.out.println("Static polymorphism (method overloading):");
        System.out.print("Enter two integers to add: ");
        int int1 = scanner.nextInt();
        int int2 = scanner.nextInt();
        D.add(int1, int2);
```

```
        System.out.print("Enter two doubles to add: ");
        double double1 = scanner.nextDouble();
        double double2 = scanner.nextDouble();
        D.add(double1, double2);

        System.out.println("\nDynamic polymorphism (method overriding):");
        Animal A = new Animal();
        A.makeSound();

      Animal dog = new Dog();
        dog.makeSound();
    }
}
```

Output:

Static polymorphism (method overloading):

Enter two integers to add: 5 4

Sum of integers: 9

Enter two doubles to add: 6 8

Sum of doubles: 14.0

Dynamic polymorphism (method overriding):

The animal makes a sound

The dog barks

31. Write a program to find factorial of a number using swing components. Add the number read and the factorial to the mysql table .

Create Table:

```
mysql> create table factorial(no int,fact int);
```

Code:

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

public class FactorialCalculator extends JFrame {
    Connection con;
    JTextField inputField;
    JTextArea resultArea;


    public FactorialCalculator() {
        try {
            con = DriverManager.getConnection("jdbc:mysql://localhost:3306/MCA", "root",
"password");
        } catch (SQLException e) {
            e.printStackTrace();
        }

        setTitle("Factorial Calculator");
        setSize(300, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new FlowLayout());

        inputField = new JTextField(10);
        JButton calculateButton = new JButton("Calculate");
        calculateButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                try {
                    calculateFactorial();
                } catch (SQLException ex) {
                    ex.printStackTrace();
                }
            }
        });

        resultArea = new JTextArea(5, 20);
        resultArea.setEditable(false);

        add(new JLabel("Enter a number: "));
        add(inputField);
        add(calculateButton);
        add(new JScrollPane(resultArea));
    }
```

```
    private void calculateFactorial() throws SQLException {
        String input = inputField.getText();
        try {
            int number = Integer.parseInt(input);
            int factorial = 1;
            for (int i = 1; i <= number; i++) {
                factorial *= i;
            }
            resultArea.setText("Factorial of " + number + " is: " + factorial);
            PreparedStatement p = con.prepareStatement("insert into factorial (no,fact)
values(?,?)");
            p.setInt(1, number);
            p.setInt(2, factorial);
            p.executeUpdate();
        } catch (NumberFormatException ex) {
            resultArea.setText("Please enter a valid integer.");
        }
    }

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

Output:

```
select * from factorial;
    -> //
+------+------+
| no   | fact |
+------+------+
|    2 |    2 |
|    5 |  120 |
|    6 |  720 |
+------+------+
3 rows in set (0.00 sec)
```

32. Write a program to concatenate two strings. Copy the strings and concatenated string to database table.

Create Table

```
CREATE TABLE concat (
    id INT AUTO_INCREMENT PRIMARY KEY,
    string1 VARCHAR(255),
    string2 VARCHAR(255),
    concatenated_string VARCHAR(510)
);
```

Code:

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

public class StringConcatenator extends JFrame {
    Connection con;
    JTextField inputField1, inputField2;
    JTextArea resultArea;

    public StringConcatenator() {
        try {
            con = DriverManager.getConnection("jdbc:mysql://localhost:3306/MCA", "root",
"password");
        } catch (SQLException e) {
            e.printStackTrace();
        }

        setTitle("String Concatenator");
        setSize(400, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new FlowLayout());

        inputField1 = new JTextField(10);
        inputField2 = new JTextField(10);
        JButton concatenateButton = new JButton("Concatenate");
        concatenateButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                try {
                    concatenateStrings();
                } catch (SQLException ex) {
                    ex.printStackTrace();
                }
            }
```

```
        });

        resultArea = new JTextArea(5, 30);
        resultArea.setEditable(false);

        add(new JLabel("Enter first string: "));
        add(inputField1);
        add(new JLabel("Enter second string: "));
        add(inputField2);
        add(concatenateButton);
        add(new JScrollPane(resultArea));
    }

    private void concatenateStrings() throws SQLException {
        String input1 = inputField1.getText();
        String input2 = inputField2.getText();
        String concatenatedString = input1 + input2;

        resultArea.setText("Concatenated string: " + concatenatedString);

        try {
            PreparedStatement p = con.prepareStatement("INSERT INTO concat (string1,
string2, concatenated_string) VALUES (?, ?, ?)");
            p.setString(1, input1);
            p.setString(2, input2);
            p.setString(3, concatenatedString);
            p.executeUpdate();
            System.out.println("Strings and concatenated string added to database
successfully.");
        } catch (SQLException ex) {
            ex.printStackTrace();
            resultArea.setText("Error: Unable to add strings to database.");
        }
    }

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

Output:

```
adminmca@MCA-Lab1:~/Downloads$ javac StringConcatenator.java
adminmca@MCA-Lab1:~/Downloads$ java StringConcatenator
Strings and concatenated string added to database successfully.
```

```
mysql> select * from concat;
    -> //
+----+---------+---------+--------------------+
| id | string1 | string2 | concatenated_string |
+----+---------+---------+--------------------+
|  1 | 123     | 123     | 123123             |
|  2 | Vivian  | Serrao  | VivianSerrao       |
+----+---------+---------+--------------------+
```