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.

Database Create Statement

```
create table Student(studentName varchar(50), regNo varchar(30) primary key, marks1 int, marks2 int, marks3 int, total int, perc double, grade varchar(10));
```

StudentDB.java

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

class StudentDB {
    static Connection con;

    public static String generateGrade(double perc) {
        if (perc >= 90)
            return "A+";
        else if (perc >= 80)
            return "A";
        else if (perc >= 65)
            return "B+";
        else if (perc >= 55)
            return "B";
        else if (perc >= 45)
            return "C";
        else
            return "Fail";
    }

    public static void addStudent() throws SQLException {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter Student name : ");
        String name = sc.nextLine();
        System.out.println("Enter reg no : ");
        String regNo = sc.nextLine();
        System.out.println("Enter 3 subject marks : ");
        int m1 = sc.nextInt();
        int m2 = sc.nextInt();
        int m3 = sc.nextInt();
        int total = m1 + m2 + m3;
        double perc = total / 300;
        String grade = generateGrade(perc);
        PreparedStatement stmt = con.prepareStatement("insert into Student
(studentName, regNo, marks1, marks2, marks3, total, percentage, grade) values
(?,?,?,?,?,?,?,?)");
        stmt.setString(1, name);
        stmt.setString(2, regNo);
        stmt.setInt(3, m1);
        stmt.setInt(4, m2);
        stmt.setInt(5, m3);
        stmt.setInt(6, total);
        stmt.setDouble(7, perc);
        stmt.setString(8, grade);
        stmt.execute();
        System.out.println("Successfully saved the student data");
    }

    public static void getStudentData() throws SQLException{
        Statement stmt = con.createStatement();
        ResultSet result = stmt.executeQuery("select * from Student");
        while(result.next())
            displayStudent(result);
    }

    public static void displayStudent(ResultSet result) throws SQLException{
        System.out.println("Student Name : " + result.getString("studentName") + "\tReg
No : " + result.getString("regNo") + "\tMarks1 : " + result.getInt("marks1") + "\tMarks 2 :
" + result.getInt("marks2") + "\tMarks 3 : " + result.getInt("marsk3"));
        System.out.println("Total : " + result.getInt("total") + "\tPercentage : " +
result.getDouble("percentage") + "\tGrade : " + result.getString("grade") + "\n");
    }

    public static void main(String[] args) {
        try {
            Scanner sc = new Scanner(System.in);
            con = DriverManager.getConnection("jdbc:mysql://localhost:3306/MCA", "root",
"password");
            while (true) {
                System.out.println(
                        "\nStudent Operation\n1.Add Student Details\n2.Display
Student\n3.Exit\nEnter your choice:");
                int choice = sc.nextInt();
                switch (choice) {
                    case 1:
                        addStudent();
                        break;
                    case 2:
                        getStudentData();
                        break;
                    case 3:
                        con.close();
                        System.exit(0);
                        break;
                    default:
                        System.out.println("Invalid choice. Please enter a valid option.");
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

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

BankStaff.java

```
class Employee{
    private String name;
    private int id;

    public Employee(String name, int id) {
        this.name = name;
        this.id = id;
    }
    public double calculateSalary() {
        return 0.0;
    }

    public String getName() {
        return name;
    }

    public int getId() {
        return id;
    }
}

class Manager extends Employee{
    private double baseSalary;
    private double bonus;

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

    @Override
    public double calculateSalary() {
        return baseSalary + bonus;
    }
}

class Officer extends Employee{
    private double baseSalary;
    private double bonus;

    public Officer(String name, int id, double baseSalary, double bonus) {
        super(name, id);
        this.baseSalary = baseSalary;
        this.bonus = bonus;
    }

    @Override
    public double calculateSalary() {
        return baseSalary + bonus;
    }
}

class Clerk extends Employee{
    private double hourlyRate;
    private int hoursWorked;
    public Clerk(String name, int id, double hourlyRate, int hoursWorked) {
        super(name, id);
        this.hourlyRate = hourlyRate;
        this.hoursWorked = hoursWorked;
    }

    @Override
    public double calculateSalary() {
        return hourlyRate * hoursWorked;
    }
}

class BankStaff {
    public static void main(String[] args){
        Employee manager = new Manager("Alice", 101, 80000, 10000);
        Employee officer = new Officer("Bob", 102, 70000, 5000);
        Employee clerk = new Clerk("Charlie", 103, 20, 160);

        System.out.println(manager.getName() + "'s Salary: " +
manager.calculateSalary());
        System.out.println(officer.getName() + "'s Salary: " + officer.calculateSalary());
        System.out.println(clerk.getName() + "'s Salary: " + clerk.calculateSalary());
    }
}
```

OUTPUT:

```
javac BankStaff.java
java BankStaff
Alice's Salary: 90000.0
Bob's Salary: 75000.0
Charlie's Salary: 3200.0
```

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

University.java

```
class Student {
    private String name;
    private int id;

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

    public void showDetails() {
        System.out.println("Name: " + name + ", ID: " + id);
    }
}

class GraduateStudent extends Student {
    private String thesisTitle;

    public GraduateStudent(String name, int id, String thesisTitle) {
        super(name, id);
        this.thesisTitle = thesisTitle;
    }

    public void showDetails() {
        super.showDetails();
        System.out.println("Thesis Title: " + thesisTitle);
    }
}

class PhDStudent extends GraduateStudent {
    private String researchArea;

    public PhDStudent(String name, int id, String thesisTitle, String researchArea) {
        super(name, id, thesisTitle);
        this.researchArea = researchArea;
    }

    public void showDetails() {
        super.showDetails();
        System.out.println("Research Area: " + researchArea);
    }
}

class UndergraduateStudent extends Student {
    private String major;

    public UndergraduateStudent(String name, int id, String major) {
        super(name, id);
        this.major = major;
    }

    public void showDetails() {
        super.showDetails();
        System.out.println("Major: " + major);
    }
}

// Main class to demonstrate inheritance
public class University {
    public static void main(String[] args) {
        UndergraduateStudent uStudent = new UndergraduateStudent("Alice", 101,
"Computer Science");
        GraduateStudent gStudent = new GraduateStudent("Bob", 102, "Machine
Learning");
```

```
        PhDStudent pStudent = new PhDStudent("Charlie", 103, "Deep Learning",
"Artificial Intelligence");

        System.out.println("Undergraduate Student Details:");
        uStudent.showDetails();
        System.out.println();

        System.out.println("Graduate Student Details:");
        gStudent.showDetails();
        System.out.println();

        System.out.println("PhD Student Details:");
        pStudent.showDetails();
    }
}
```

OUTPUT:

javac University.java

java University

```
Undergraduate Student Details:
Name: Alice, ID: 101
Major: Computer Science

Graduate Student Details:
Name: Bob, ID: 102
Thesis Title: Machine Learning

PhD Student Details:
Name: Charlie, ID: 103
Thesis Title: Deep Learning
Research Area: Artificial Intelligence
```

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

Database Queries

```
create table EMS (id int primary key, name varchar(30), designation varchar(30), salary
int);

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
//
```

EMS.java

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

class EMS {
    static Connection con;

    public static void addEmployee(String name, String designation, int salary) throws
SQLException {
        CallableStatement stmt = con.prepareCall("{call add_emp(?,?,?)}");
        stmt.setString(1, name);
        stmt.setString(2, designation);
        stmt.setInt(3, salary);
        stmt.execute();
        System.out.println("Record Inserted");
    }
    public static 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");
    }

    public static void generateEmployeeReport() {
        try {
            Statement s = con.createStatement();
            ResultSet r = s.executeQuery("SELECT * FROM EMS");

            System.out.println("Employee Report:");
            System.out.println("ID\tName\tDesignation\tSalary\tDA\tHRA\tNetSalary");
            while (r.next()) {
                displayEmployee(r);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public static void displayEmployee(ResultSet r) throws SQLException {
        int id = r.getInt("id");
        String name = r.getString("name");
        String designation = r.getString("designation");
        int salary = r.getInt("salary");
        int da = r.getInt("da");
        int hra = r.getInt("hra");
        int netSalary = r.getInt("netSalary");
        System.out.println(id + "\t" + name + "\t" + designation + "\t" + salary + "\t" + da +
"\t" + hra + "\t" + netSalary);
    }

    public static void deleteEmployee(int empid) {
        try {
            Scanner sc = new Scanner(System.in);
            Statement s = con.createStatement();
            ResultSet r = s.executeQuery("SELECT * FROM EMS where id=" + empid);
            System.out.println("Employee Report:");
            System.out.println("ID\tName\tDesignation\tSalary\tDA\tHRA\tNetSalary");

            if (r.next()) { // Check if the result set contains any rows
                displayEmployee(r);

                System.out.println("Do you want to delete the record from the database?
(Y/N):");
                char ch = sc.next().charAt(0);
                if (ch == 'Y' || ch == 'y') {
                    PreparedStatement p = con.prepareStatement("DELETE FROM EMS
WHERE id=?");
                    p.setInt(1, empid);
                    p.executeUpdate();
                    System.out.println("Record Deleted successfully");
                }
            } else {
                System.out.println("No employee found with ID: " + empid);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    public static void main(String[] args) {
        try {
            String name, desig;
            int id;
            int salary;
            Scanner sc = new Scanner(System.in);
            con = DriverManager.getConnection("jdbc:mysql://localhost:3306/MCA", "root",
"password");
            while (true) {
                System.out.println( "\nEmployee Operation\n1.Add Employee\n2.Update
Employee\n3.Display Employee\n4.Delete\n5.Exit\nEnter your choice:");
                int choice = sc.nextInt();
                switch (choice) {
                    case 1:
                        System.out.println("Enter Employee name:");
                        sc.nextLine(); // consume newline
                        name = sc.nextLine();
                        System.out.println("Enter Designation:");
                        desig = sc.nextLine();
                        System.out.println("Enter Employee Salary:");
                        salary = sc.nextInt();
                        addEmployee(name, desig, salary);
                        break;
                    case 2:
                        System.out.println("Enter Employee Id:");
                        id = sc.nextInt();
                        sc.nextLine(); // consume newline
                        System.out.println("Enter Employee name:");
                        name = sc.nextLine();
                        System.out.println("Enter Designation:");
                        desig = sc.nextLine();
                        System.out.println("Enter Employee Salary:");
                        salary = sc.nextInt();
                        updateEmployee(id, name, desig, salary);
                        break;
                    case 3:
                        generateEmployeeReport();
                        break;
                    case 4:
                        System.out.println("Enter Employee Id:");
                        id = sc.nextInt();
                        deleteEmployee(id);
                        break;

                    case 5:
                        con.close();
                        System.exit(0);
                        break;
                    default:
                        System.out.println("Invalid choice. Please enter a valid option.");
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

}

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

AreaCalculator.java

```
abstract class Shape {
    abstract double calculateArea();
}

class Circle extends Shape {
    private double radius;

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

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

class Rectangle extends Shape {
    private double length;
    private double width;

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

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

```
class Triangle extends Shape {
    private double base;
    private double height;

    public Triangle(double base, double height) {
        this.base = base;
        this.height = height;
    }

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

// Main class to demonstrate dynamic method dispatch
public class AreaCalculator {
    public static void main(String[] args) {
        Shape circle = new Circle(5);
        Shape rectangle = new Rectangle(4, 6);
        Shape triangle = new Triangle(3, 7);

        System.out.println("Area of Circle: " + circle.calculateArea());
        System.out.println("Area of Rectangle: " + rectangle.calculateArea());
        System.out.println("Area of Triangle: " + triangle.calculateArea());
    }
}
```

OUTPUT:

```
javac AreaCalculator.java
```

```
java AreaCalculator
```

```
Area of Circle: 78.53981633974483
Area of Rectangle: 24.0
Area of Triangle: 10.5
```

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

ListOperations.java

```java
import java.util.*;

public class ListOperations {
    public static void main(String[] args) {
        // Creating two lists of strings
        List<String> list1 = new ArrayList<>();
        List<String> list2 = new ArrayList<>();

        list1.add("Apple");
        list1.add("Banana");

        list2.add("Dog");
        list2.add("Elephant");

        System.out.println("Original List1: " + list1);
        System.out.println("Original List2: " + list2);

        list1.set(1, "Blueberry");
        list1.add(0, "Avocado");

        list2.remove(1);
        list2.add("Giraffe");

        list1.addAll(list2);
        System.out.println("Updated List1: " + list1);
        System.out.println("Updated List2: " + list2);

        boolean containsApple = list1.contains("Apple");
        System.out.println("List1 contains 'Apple': " + containsApple);

        list2.clear();
        System.out.println("Cleared List2: " + list2);
    }
}
```

OUTPUT

javac ListOperations.java

java ListOperations

```
Original List1: [Apple, Banana]
Original List2: [Dog, Elephant]
Updated List1: [Avocado, Apple, Blueberry, Dog, Giraffe]
Updated List2: [Dog, Giraffe]
List1 contains 'Apple': true
Cleared List2: []
```

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).

Student,java

```
import java.io.*;

public class Student implements Serializable {
    private int studno;
    private String sname;
    private int marks1, marks2, marks3;
    private int total;
    private 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 = total / 3.0;
    }

    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);
    }

    public int getTotal() {
        return total;
    }

    public double getPercentage() {
        return percentage;
    }
}
```

StudentClient.java

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

public class StudentClient {
    public static void main(String[] args) {
        try {
            Socket socket = new Socket("localhost", 5000);
            ObjectOutputStream out = new
ObjectOutputStream(socket.getOutputStream());
            ObjectInputStream in = new ObjectInputStream(socket.getInputStream());
            Scanner scanner = new Scanner(System.in);

            System.out.print("Enter Student Number: ");
            int studno = scanner.nextInt();
            scanner.nextLine();
            System.out.print("Enter Student Name: ");
            String sname = scanner.nextLine();
            System.out.print("Enter Marks 1: ");
            int marks1 = scanner.nextInt();
            System.out.print("Enter Marks 2: ");
            int marks2 = scanner.nextInt();
            System.out.print("Enter Marks 3: ");
            int marks3 = scanner.nextInt();

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

            out.writeObject(student);

            student = (Student) in.readObject();

            student.dispStudent();

            scanner.close();
            in.close();
            out.close();
            socket.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

StudentServer.java

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

public class StudentServer {
    public static void main(String[] args) {
        try {
            ServerSocket serverSocket = new ServerSocket(5000);
            System.out.println("Server started and waiting for client...");

            Socket socket = serverSocket.accept();
            System.out.println("Client connected.");

            ObjectInputStream in = new ObjectInputStream(socket.getInputStream());
            ObjectOutputStream out = new
ObjectOutputStream(socket.getOutputStream());

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

            student.calculate();

            out.writeObject(student);

            in.close();
            out.close();
            socket.close();
            serverSocket.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

OUTPUT

javac Student,java
javac StudentClient,java
javac StudentServer,java

java StudentServer

```
Server started and waiting for client...
Client connected.
```

java StudentClient (in Seperate terminal)

```
Enter Student Number: 111
Enter Student Name: Bob
Enter Marks 1: 96
Enter Marks 2: 80
Enter Marks 3: 70
Student Number: 111
Student Name: Bob
Marks 1: 96
Marks 2: 80
Marks 3: 70
Total: 246
Percentage: 82.0
```

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

FinallyDemo.java

```
public class FinallyDemo {

    public static void main(String[] args) {
        System.out.println("Case 1: Normal execution without any exceptions");
        normalExecution();

        System.out.println("\nCase 2: Execution with an exception that is caught");
        exceptionCaught();

        System.out.println("\nCase 3: Execution with an exception that is not caught");
        exceptionNotCaught();

        System.out.println("\nCase 4: Exiting a method using System.exit()");
        try {
            exitMethod();
        } catch (Exception e) {
            System.out.println("Exception caught in main");
        }
    }

    public static void normalExecution() {
        try {
            System.out.println("Inside try block");
        } finally {
            System.out.println("Inside finally block");
        }
    }

    public static void exceptionCaught() {
        try {
            System.out.println("Inside try block");
            int result = 10 / 0;
        } catch (ArithmeticException e) {
            System.out.println("Exception caught: " + e);
        } finally {
            System.out.println("Inside finally block");
        }
    }

    public static void exceptionNotCaught() {
        try {
            System.out.println("Inside try block");
            int result = 10 / 0;
        } finally {
            System.out.println("Inside finally block");
        }
    }

    public static void exitMethod() {
        try {
            System.out.println("Inside try block");
            System.exit(0);
        } finally {
            System.out.println("Inside finally block");
        }
    }
}
```

OUTPUT

javac FinallyDemo.java

java FinallyDemo

```
Case 1: Normal execution without any exceptions
Inside try block
Inside finally block

Case 2: Execution with an exception that is caught
Inside try block
Exception caught: java.lang.ArithmeticException: / by zero
Inside finally block

Case 3: Execution with an exception that is not caught
Inside try block
Inside finally block
Exception caught in main
```

```
Case 4: Exiting a method using System.exit()
Inside try block
```

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

ChatServer.java

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

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

            while (true) {
                // Receive Message from client
                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);

                // Send message to client
                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.*;

public class ChatClient {
    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) {
                // Send message to server
                System.out.print("You: ");
                String message = reader.readLine();
                byte[] sendData = message.getBytes();

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

                // Received message from server
                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

```
javac ChatServer.java
javac ChatClient.java
```

java ChatServer

```
Client (127.0.0.1:56397): hello
Server: Hello from server
Client (127.0.0.1:56397): This is a message from the client
Server: this is the message from the server
```

java ChatClient (Run in different terminal)

```
You: hello
Server: Hello from server
You: This is a message from the client
Server: this is the message from the server
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.

DisplayUrl.java

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

OUTPUT
javac DisplayUrl.java

java DisplayUrl

```
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.

ThrowDemo.java

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

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

        try {
            int num = 10;
            int den = 0;
            if (den == 0) {
                throw new ArithmeticException("Denominator cannot be 0");
            }
            System.out.println("Division: " + num / den);
        } catch (ArithmeticException e) {
            System.out.println("Exception caught (Built-in): " + e.getMessage());
        }
        try {
            int balance = 100;
            int withdrawAmount = 200;
            if (withdrawAmount > balance) {
                throw new MyCustomException("Insufficient balance");
            }
            System.out.println("Withdrawal successful");
        } catch (MyCustomException e) {
            System.out.println("Exception caught (Custom): " + e.getMessage());
        }
    }
}
```

OUTPUT:

javac ThrowDemo.java

java ThrowDemo

```
Exception caught (Built-in): Denominator cannot be 0
Exception caught (Custom): Insufficient balance
```

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).

SalaryCalculator.java

```
import java.rmi.*;

public interface SalaryCalculator extends Remote {
    double calculateDA(double basic) throws RemoteException;
    double calculateHRA(double basic) throws RemoteException;
    double calculateNetSalary(double basic, double da, double hra) throws
RemoteException;
}
```

SalaryCalculatorImpl.java

```
import java.rmi.*;
import java.rmi.server.*;

public class SalaryCalculatorImpl extends UnicastRemoteObject implements
SalaryCalculator {

    protected SalaryCalculatorImpl() throws RemoteException {
        super();
    }

    public double calculateDA(double basic) throws RemoteException {
        return basic * 0.1; // DA is 10% of basic
    }

    public double calculateHRA(double basic) throws RemoteException {
        return basic * 0.2; // HRA is 20% of basic
    }

    public double calculateNetSalary(double basic, double da, double hra) throws
RemoteException {
        return basic + da + hra; // Net Salary = Basic + DA + HRA
    }
}
```

SalaryCalculatorServer.java

```
import java.rmi.registry.*;

public class SalaryCalculatorServer {
    public static void main(String[] args) {
        try {
            SalaryCalculator calculator = new SalaryCalculatorImpl();

            Registry registry = LocateRegistry.createRegistry(1099);
            registry.bind("SalaryCalculator", calculator);

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

SalaryCalculatorClient,java

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

public class SalaryCalculatorClient {
    public static void main(String[] args) {
        try {
            Registry registry = LocateRegistry.getRegistry("localhost", 1099);
            SalaryCalculator calculator = (SalaryCalculator)
registry.lookup("SalaryCalculator");

            Scanner scanner = new Scanner(System.in);
            System.out.print("Enter Employee Number: ");
            int empno = scanner.nextInt();
            scanner.nextLine(); // Consume newline
            System.out.print("Enter Employee Name: ");
            String ename = scanner.nextLine();
            System.out.print("Enter Basic Salary: ");
            double basic = scanner.nextDouble();

            // Call remote methods to calculate DA, HRA, and Net Salary
            double da = calculator.calculateDA(basic);
            double hra = calculator.calculateHRA(basic);
            double netSalary = calculator.calculateNetSalary(basic, da, hra);

            // Display the details
            System.out.println("Employee Number: " + empno);
            System.out.println("Employee Name: " + ename);
            System.out.println("Basic Salary: " + basic);
            System.out.println("Dearness Allowance (DA): " + da);
            System.out.println("House Rent Allowance (HRA): " + hra);
            System.out.println("Net Salary: " + netSalary);

            scanner.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

OUTPUT

```
javac SalaryCalculator.java
javac SalaryCalculatorImpl.java
javac SalaryCalculatorServer.java
javac SalaryCalculatorClient.java
```

javac SalaryCalculatorServer

```
Salary Calculator Server is ready.
```

javac SalaryCalculatorClient **(Run in different terminal)**

```
Enter Employee Number: 123
Enter Employee Name: Sam
Enter Basic Salary: 40000
Employee Number: 123
Employee Name: Sam
Basic Salary: 40000.0
Dearness Allowance (DA): 4000.0
House Rent Allowance (HRA): 8000.0
Net Salary: 52000.0
```

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

SalaryCalc.java

```java
public class SalaryCalc {

  public double calculateDA(double basic) {
    return basic * 0.10; // DA is 10% of basic
  }

  public double calculateHRA(double basic) {
    return basic * 0.20; // HRA is 20% of basic
  }

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

    public double calculateIT(double basic) {
        return basic * 0.05; // IT is 5% of basic
    }

    public double calculateNetSalary(double basic, double da, double hra, double pf,
double it) {
        return basic + da + hra - pf - it;
    }
}
```

Employee.java

```
public class Employee {
    private int empNo;
    private String empName;
    private double basic;
    private double da;
    private double hra;
    private double pf;
    private double it;
    private double netSalary;
    private SalaryCalc salaryCalc;

    // Constructor to initialize employee details and salary calculator
    public Employee(int empNo, String empName, double basic, SalaryCalc salaryCalc) {
        this.empNo = empNo;
        this.empName = empName;
        this.basic = basic;
        this.salaryCalc = salaryCalc;
        calculateSalaryComponents();
    }

    // Method to calculate all salary components
    private void calculateSalaryComponents() {
        this.da = salaryCalc.calculateDA(basic);
        this.hra = salaryCalc.calculateHRA(basic);
        this.pf = salaryCalc.calculatePF(basic);
        this.it = salaryCalc.calculateIT(basic);
        this.netSalary = salaryCalc.calculateNetSalary(basic, da, hra, pf, it);
    }

    // Method to display employee details
    public void displayEmployeeDetails() {
        System.out.println("Employee Number: " + empNo);
        System.out.println("Employee Name: " + empName);
        System.out.println("Basic Salary: " + basic);
        System.out.println("Dearness Allowance (DA): " + da);
        System.out.println("House Rent Allowance (HRA): " + hra);
        System.out.println("Provident Fund (PF): " + pf);
        System.out.println("Income Tax (IT): " + it);
        System.out.println("Net Salary: " + netSalary);
    }
}
```

EmployeeAggregation.java

```
import java.util.Scanner;

public class EmployeeAggregation {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        // Accept employee details from the user
        System.out.print("Enter Employee Number: ");
        int empNo = scanner.nextInt();
        scanner.nextLine(); // Consume newline
        System.out.print("Enter Employee Name: ");
        String empName = scanner.nextLine();
        System.out.print("Enter Basic Salary: ");
        double basic = scanner.nextDouble();

        SalaryCalc salaryCalculator = new SalaryCalc();
        Employee employee = new Employee(empNo, empName, basic, salaryCalculator);
        employee.displayEmployeeDetails();

        scanner.close();
    }
}
```

OUTPUT

```
javac Employee.java
javac SalaryCalc.java
javac EmployeeAggregation.java

java EmployeeAggregation

Enter Employee Number: 1234
Enter Employee Name: John
Enter Basic Salary: 50000
Employee Number: 1234
Employee Name: John
Basic Salary: 50000.0
Dearness Allowance (DA): 5000.0
```

```
House Rent Allowance (HRA): 10000.0
Provident Fund (PF): 6000.0
Income Tax (IT): 2500.0
Net Salary: 56500.0
```

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.

Database Command

```
create table Employee(empName varchar(50), empNo int primary key, basic double, da double, hra double, net double);
```

EmployeeDB.java

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

class EmployeeDB {
    static Connection con;


    public static void addEmployee() throws SQLException {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter Emp no : ");
        int empNo = sc.nextInt();
        sc.nextLine();
        System.out.println("Enter Employee name : ");
        String name = sc.nextLine();
        System.out.println("Enter Basic salary : ");
        double basic = sc.nextDouble();
        System.out.println("Successfully saved the Employee data");
        double da = basic * 0.1;
        double hra = basic * 0.12;
        double netSalary = basic + da + hra;

        PreparedStatement st = con.prepareStatement("insert into employee (empName,
empNo, basic, da, hra, net) values (?,?,?,?,?,?)");
        st.setString(1, name);
        st.setInt(2, empNo);
        st.setDouble(3, basic);
        st.setDouble(4, da);
        st.setDouble(5, hra);
        st.setDouble(6, netSalary);
        st.executeQuery();
        System.out.println("Successfully saved the employee details");
    }

    public static void getEmployeeData() throws SQLException{
        Statement stmt = con.createStatement();
        ResultSet result = stmt.executeQuery("select * from Employee");
        while(result.next())
            displayEmployee(result);
    }

    public static void displayEmployee(ResultSet result) throws SQLException{
        System.out.println("Emp Name : " + result.getString("empName") + "\tEmp No : " +
result.getInt("empNo") + "\tBasic : " + result.getDouble("basic") + "\tDA : " +
result.getDouble("da") + "\tHRA : " + result.getDouble("hra")+ "\tNet : " +
result.getDouble("net"));
    }

    public static void main(String[] args) {
        try {
            Scanner sc = new Scanner(System.in);
            con = DriverManager.getConnection("jdbc:mysql://localhost:3306/MCA", "vikas",
"vikasrai");
            while (true) {
                System.out.println("Employee Operation\n1.Add Employee Details\n2.Display
Employee\n3.Exit\nEnter your choice:");
                int choice = sc.nextInt();
                switch (choice) {
                    case 1:
                        addEmployee();
                        break;
                    case 2:
                        getEmployeeData();
                        break;
                    case 3:
                        con.close();
                        System.exit(0);
                        break;
                    default:
                        System.out.println("Invalid choice. Please enter a valid option.");
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

OUTPUT

javac EmployeeDB.java

java EmployeeDB

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.

Bank.java

```
public interface Bank {
    void readCustomerInfo();
}
```

ICICI.java

```
public interface ICICI extends Bank {
    void calculateInterest();
    void displayDetails();
}
```

Axis.java

```
public interface Axis extends Bank {
    void calculateInterest();
    void displayDetails();
}
```

ICICICustomer.java

```
import java.util.Scanner;

public class ICICICustomer implements ICICI {

    private String name;
    private int accNo;
    private double balance;
    private double interest;

    public void readCustomerInfo() {
         Scanner sc = new Scanner(System.in);
        System.out.print("Enter Customer Name: ");
        name = sc.nextLine();
        System.out.print("Enter Account Number: ");
        accNo = sc.nextInt();
        System.out.print("Enter Account Balance: ");
        balance = sc.nextDouble();
    }

    public void calculateInterest() {
        interest = balance * 0.08;
    }

    public void displayDetails() {
        System.out.println("Customer Name: " + name);
        System.out.println("Account Number: " + accNo);
        System.out.println("Account Balance: " + balance);
        System.out.println("Calculated Interest: " + interest);
    }

}
```

AxisCustomer.java

```
import java.util.Scanner;

public class AxisCustomer implements Axis {

    private String name;
    private int accNo;
    private double balance;
    private double interest;

    public void readCustomerInfo() {
         Scanner sc = new Scanner(System.in);
        System.out.print("Enter Customer Name: ");
        name = sc.nextLine();
        System.out.print("Enter Account Number: ");
        accNo = sc.nextInt();
        System.out.print("Enter Account Balance: ");
        balance = sc.nextDouble();
    }

    public void calculateInterest() {
        interest = balance * 0.05;
    }

    public void displayDetails() {
        System.out.println("Customer Name: " + name);
        System.out.println("Account Number: " + accNo);
        System.out.println("Account Balance: " + balance);
        System.out.println("Calculated Interest: " + interest);
    }

}
```

BankCustomer.java

```
public class BankCustomer {

    public static void main(String[] args) {
        AxisCustomer asixAxisCustomer = new AxisCustomer();
        ICICICustomer iciciCustomer = new ICICICustomer();

        asixAxisCustomer.readCustomerInfo();
        asixAxisCustomer.calculateInterest();
        asixAxisCustomer.displayDetails();

        iciciCustomer.readCustomerInfo();
        iciciCustomer.calculateInterest();
        iciciCustomer.displayDetails();

    }

}
```

OUTPUT

javac *.java **(Compiles all the java files)**

java BankCustomer.java

```
Enter Customer Name: Axis Customer
Enter Account Number: 122334
Enter Account Balance: 30000

Customer Name: Axis Customer
Account Number: 122334
Account Balance: 30000.0
Calculated Interest: 1500.0

Enter Customer Name: ICICI Customer
Enter Account Number: 3662553
Enter Account Balance: 30000

Customer Name: ICICI Customer
Account Number: 3662553
Account Balance: 30000.0
Calculated Interest: 2400.0
```

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

SetOperations.java

OUTPUT:

javac SetOperations.java

```
import java.util.HashSet;
import java.util.Set;

public class SetOperations {

    public static void main(String[] args) {
        // Creating two sets
        Set<Integer> set1 = new HashSet<>();
        Set<Integer> set2 = new HashSet<>();

        // Adding elements to set1
        set1.add(1);
        set1.add(2);
        set1.add(3);
        set1.add(4);

        // Adding elements to set2
        set2.add(3);
        set2.add(4);
        set2.add(5);
        set2.add(6);

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

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

        // Perform Difference operation (set1 - set2)
        Set<Integer> differenceSet1 = new HashSet<>(set1);
        differenceSet1.removeAll(set2);
        System.out.println("Difference of set1 and set2 (set1 - set2): " + differenceSet1);

        // Perform Difference operation (set2 - set1)
        Set<Integer> differenceSet2 = new HashSet<>(set2);
        differenceSet2.removeAll(set1);
        System.out.println("Difference of set2 and set1 (set2 - set1): " + differenceSet2);
    }
}
```

java SetOperations

```
Union of set1 and set2: [1, 2, 3, 4, 5, 6]
Intersection of set1 and set2: [3, 4]
Difference of set1 and set2 (set1 - set2): [1, 2]
Difference of set2 and set1 (set2 - set1): [5, 6]
```

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

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

javac *.java

java ChatServer

```
Client (127.0.0.1:56397): hello
Server: Hello from server
Client (127.0.0.1:56397): This is a message from the client
```

```
Server: this is the message from the server
```

java ChatClient (Run in different terminal)

```
You: hello
Server: Hello from server
You: This is a message from the client
Server: this is the message from the server
You:
```

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

BankAccountApp.java

```java
import java.util.*;

class BankAccount {
    int account_number;
    int balance;

    BankAccount(int acc_no, int bal) {
        account_number = acc_no;
        balance = bal;
    }

    synchronized void deposit(int amount) {
        System.out.println("Depositing " + amount + " into account " + account_number);
        balance += amount;
        System.out.println("New balance after deposit: " + balance);
    }
    synchronized void withdraw(int amount) throws InsufficientException {
        if (balance < amount) {
            throw new InsufficientException("Insufficient balance");
        }
        System.out.println("Withdrawing " + amount + " from account " +
account_number);
        balance -= amount;
        System.out.println("New balance after withdrawal: " + balance);
    }

    void balanceEnquiry() {
        System.out.println("Balance in account " + account_number + " is: " + balance);
    }

    void calculateInterest() {
        double interest = balance * 0.08;
        System.out.println("Interest in account " + account_number + " is: " + interest);
    }
}

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

class DepositThread extends Thread {
    private BankAccount account;
    private int amount;

    public DepositThread(BankAccount account, int amount) {
        this.account = account;
        this.amount = amount;
    }

    public void run() {
        account.deposit(amount);
    }
}

class WithdrawThread extends Thread {
    private BankAccount account;
    private int amount;

    public WithdrawThread(BankAccount account, int amount) {
        this.account = account;
        this.amount = amount;
    }

    public void run() {
        try {
            account.withdraw(amount);
        } catch (InsufficientException e) {
            System.out.println(e.getMessage());
        }
    }
}

public class BankAccountApp {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the Account Number:");
        int account_number = sc.nextInt();
        System.out.println("Enter the Initial Balance:");
        int balance = sc.nextInt();
        BankAccount s = new BankAccount(account_number, balance);
        // Simulate deposit and withdrawal using threads
        Thread depositThread = new DepositThread(s, 100);
        Thread withdrawThread = new WithdrawThread(s, 50);

        depositThread.start();
        withdrawThread.start();

        // Wait for threads to finish
        try {
            depositThread.join();
            withdrawThread.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        s.calculateInterest();
        s.balanceEnquiry();
    }
}
```

OUTPUT

javac BankAccountApp.java

java BankAccountApp

```
Enter the Account Number:
1234
Enter the Initial Balance:
10000
Depositing 100 into account 1234
New balance after deposit: 10100
Withdrawing 50 from account 1234
```

```
New balance after withdrawal: 10050
Interest in account 1234 is: 804.0
Balance in account 1234 is: 10050
```

19.Write a program to demonstrate Inter-thread communication

InterThreadCommunication.java

```
class SharedResource {
    private int data;
    private boolean newData = false;

    // Method for the producer thread to set data
    synchronized void setData(int data) {
        // Wait until the consumer thread consumes the existing data
        while (newData) {
            try {
                wait(); // Release the lock and wait for notify
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        this.data = data;
        System.out.println("Produced: " + data);
        newData = true;
        notify(); // Notify the consumer thread that new data is available
    }

    // Method for the consumer thread to get data
    synchronized int getData() {
        // Wait until the producer thread sets new data
        while (!newData) {
            try {
                wait(); // Release the lock and wait for notify
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        newData = false;
        notify(); // Notify the producer thread that data has been consumed
        System.out.println("Consumed: " + data);
        return data;
    }
}

class Producer extends Thread {
    private SharedResource sharedResource;

    public Producer(SharedResource sharedResource) {
        this.sharedResource = sharedResource;
    }

    public void run() {
        for (int i = 0; i < 5; i++) {
            sharedResource.setData(i);
            try {
                sleep(1000); // Simulate some processing time
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

class Consumer extends Thread {
    private SharedResource sharedResource;
    public Consumer(SharedResource sharedResource) {
        this.sharedResource = sharedResource;
    }

    public void run() {
        for (int i = 0; i < 5; i++) {
            int data = sharedResource.getData();
            try {
                sleep(1000); // Simulate some processing time
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

public class InterThreadCommunication {
    public static void main(String[] args) {
        SharedResource sharedResource = new SharedResource();
        Producer producer = new Producer(sharedResource);
        Consumer consumer = new Consumer(sharedResource);
        producer.start();
        consumer.start();
    }
}
```

OUTPUT

javac InterThreadCommunication.java

java InterThreadCommunication

```
Produced: 0
Consumed: 0
Produced: 1
Consumed: 1
Produced: 2
Consumed: 2
Produced: 3
Consumed: 3
Produced: 4
Consumed: 4
```

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

ByteStreamFileCopy.java

```java
import java.io.*;

public class ByteStreamFileCopy {
    // Method to copy file using ByteStream
    public static void copyByByteStream(File source, File destination) throws
IOException {
        InputStream inputStream = null;
        OutputStream outputStream = null;
        try {
            inputStream = new FileInputStream(source);
            outputStream = new FileOutputStream(destination);
            byte[] buffer = new byte[1024];
            int length;
            while ((length = inputStream.read(buffer)) != -1) {
                outputStream.write(buffer, 0, length);
            }
            System.out.println("File copied successfully using ByteStream.");
        } finally {
            if (inputStream != null) {
                inputStream.close();
            }
            if (outputStream != null) {
                outputStream.close();
            }
        }
    }


    public static void main(String[] args) {
        File sourceFile = new File("source.txt");
        File destinationFileByteStream = new File("destination_byte_stream.txt");

        try {
            // Copy file using ByteStream
            copyByByteStream(sourceFile, destinationFileByteStream);

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

OUTPUT

```
javac ByteStreamFileCopy.java

java ByteStreamFileCopy
File copied successfully using ByteStream.
```

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

NOTE: Create a file with name sample.txt before running the program and add some random sentences in it to count the number of consonants, digits and vowels

FileCharacterCount.java

```
public class FileCharacterCount {
    public static void main(String[] args) {
        String filePath = "sample.txt"; // Path to your file

        // Initialize counters
        int vowelCount = 0;
        int consonantCount = 0;
        int digitCount = 0;

        try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
            String line;
            while ((line = reader.readLine()) != null) {
                // Convert the line to lowercase for case-insensitive matching
                line = line.toLowerCase();

                // Iterate through each character in the line
                for (int i = 0; i < line.length(); i++) {
                    char ch = line.charAt(i);
                    if (Character.isLetter(ch)) {
                        // Check if the character is a vowel or consonant
                        if (isVowel(ch)) {
                            vowelCount++;
                        } else {
                            consonantCount++;
                        }
                    } else if (Character.isDigit(ch)) {
```

```
                    digitCount++;
                }
            }
        }

        // Display the counts
        System.out.println("Vowels: " + vowelCount);
        System.out.println("Consonants: " + consonantCount);
        System.out.println("Digits: " + digitCount);
    } catch (IOException e) {
        e.printStackTrace();
    }
}

// Method to check if a character is a vowel
private static boolean isVowel(char ch) {
    return ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u';
}
}
```

OUTPUT

javac FileCharacterCount.java

java FileCharacterCount

```
Vowels: 24
Consonants: 49
Digits: 12
```

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

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

CharacterStreamFileCopy.java

```
import java.io.*;

public class CharacterStreamFileCopy {

    // Method to copy file using CharacterStream
    public static void copyByCharacterStream(File source, File destination) throws
IOException {
        Reader reader = null;
        Writer writer = null;
        try {
            reader = new FileReader(source);
            writer = new FileWriter(destination);
            char[] buffer = new char[1024];
            int length;
            while ((length = reader.read(buffer)) != -1) {
                writer.write(buffer, 0, length);
            }
            System.out.println("File copied successfully using CharacterStream.");
        } finally {
            if (reader != null) {
                reader.close();
            }
            if (writer != null) {
                writer.close();
            }
        }
    }
    public static void main(String[] args) {
        File sourceFile = new File("source.txt");
        File destinationFileCharacterStream = new
File("destination_character_stream.txt");

        try {
            copyByCharacterStream(sourceFile, destinationFileCharacterStream);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

OUTPUT:

javac CharacterStreamFileCopy.java

java CharacterStreamFileCopy

```
File copied successfully using CharacterStream.
```

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

KeyPrinter.java

```
import javax.swing.*;
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);

        getContentPane().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.

AdapterKeyEvents.java

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

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

    public AdapterKeyEvents() {
        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) {

            AdapterKeyEvents frame = new AdapterKeyEvents();
            frame.setVisible(true);
        };

}
```

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

FactorialCalculator.java

```
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(new JScrollPane(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) {
        SwingUtilities.invokeLater(() -> {
            FactorialCalculator calculator = new FactorialCalculator();
            calculator.setVisible(true);
        });
    }
```

```
}
```

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

FibonacciGenerator.java

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

public class FibonacciGenerator extends JFrame {
    JTextField inputField;
    JTextArea outputField;
    JButton generateButton;

    public FibonacciGenerator(){
        setTitle("Fibonacci Generator");
        setSize(400, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new FlowLayout());

        inputField = new JTextField(10);
        generateButton = new JButton("Generate");
        generateButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e){
                generateFibonacci();
            }
        });

        outputField = new JTextArea(5, 30);
        outputField.setEditable(false);
        add(new JLabel("Enter the no to generate the sequence"));
        add(inputField);
        add(generateButton);
        add(outputField);
    }

    void generateFibonacci(){
        try{
            outputField.setText("");
            int number = Integer.parseInt(inputField.getText());
            int i = 0, n1 = 0, n2 = 1, n3;
            while(i < number){
                n3 = n1 + n2;
                String oldText = outputField.getText();
                outputField.setText(oldText + "  " + Integer.toString(n1));
                n1 = n2;
                n2 = n3;
                i++;
            }
        }catch(NumberFormatException e){
            outputField.setText("Please enter a proper number");
        }
    }

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

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

ConcatenateString.java

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

public class ConcatenateString extends JFrame {
    JTextField textField1;
    JTextField textField2;
    JTextArea outputField;
    JButton generateButton;

    public ConcatenateString(){
        setTitle("Concatenate String");
        setSize(400, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new FlowLayout());

        textField1 = new JTextField(10);
        textField2 = new JTextField(10);

        generateButton = new JButton("Concatename");
        generateButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e){
                concatString();
            }
        });

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

        add(new JLabel("Enter the first text"));
        add(textField1);
        add(new JLabel("Enter the second text"));
        add(textField2);
        add(generateButton);
        add(outputField);
    }

    void concatString(){
        String text1 = textField1.getText();
        String text2 = textField2.getText();
        outputField.setText("");
        outputField.setText(text1 + text2);
    }

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

28.Write a program to Demonstrate thread synchronization.

ThreadSync.java

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

    public synchronized void increment() {
        count++;
    }
    public int getCount() {
        return count;
    }
}

class CounterThread extends Thread {
    private Counter counter;

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

    public void run() {
        for (int i = 0; i < 1000; i++) {
            counter.increment();
        }
    }
}

public class ThreadSync {
    public static void main(String[] args) {
        Counter counter = new Counter();

        CounterThread t1 = new CounterThread(counter);
        CounterThread t2 = new CounterThread(counter);
        CounterThread t3 = new CounterThread(counter);

        t1.start();
        t2.start();
        t3.start();

        try {
            t1.join();
            t2.join();
            t3.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

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

OUTPUT

javac ThreadSync.java

java ThreadSync

```
Final count: 3000
```

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

Person.java

```
import java.io.Serializable;

public class Person implements Serializable {
    private static final long serialVersionUID = 1L;

    private String name;
    private int age;

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

    public String toString() {
        return "Person{name='" + name + "', age=" + age + "}";
    }
}
```

SerializeDemo.java

```
public class SerializeDemo {
    public static void main(String[] args) {
        Person person = new Person("John Doe", 30);

        // Serialize the person object to a file
        try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream("person.ser"))) {
            oos.writeObject(person);
            System.out.println("Serialization successful: " + person);
        } catch (IOException e) {
            e.printStackTrace();
        }

        // Deserialize the person object from the file
        try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream("person.ser"))) {
            Person deserializedPerson = (Person) ois.readObject();
            System.out.println("Deserialization successful: " + deserializedPerson);
        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
}
```

OUTPUT

javac *.java

java SerializeDemo.java

```
Serialization successful: Person{name='John Doe', age=30}
Deserialization successful: Person{name='John Doe', age=30}
```

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

Polymorphism.java

```java
// Base class
class Shape {
    // Method to calculate area, intended to be overridden
    public void area() {
        System.out.println("Calculating area of shape");
    }
}

// Derived class Circle that overrides the area method
class Circle extends Shape {
    private double radius;

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

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

// Derived class Rectangle that overrides the area method
class Rectangle extends Shape {
    private double length;
    private double width;

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

    @Override
    public void area() {
        double area = length * width;
        System.out.println("Area of Rectangle: " + area);
    }
}

// Class demonstrating static polymorphism through method overloading
class ShapeAreaCalculator {
    // Method to calculate area of a rectangle
    public double calculateArea(double length, double width) {
        return length * width;
    }

    // Method to calculate area of a circle
    public double calculateArea(double radius) {
        return Math.PI * radius * radius;
    }
}

public class Polymorphism {
    public static void main(String[] args) {
        // Demonstrating static polymorphism
        ShapeAreaCalculator calculator = new ShapeAreaCalculator();
        System.out.println("Area of Rectangle (static): " + calculator.calculateArea(5.0,
3.0));
        System.out.println("Area of Circle (static): " + calculator.calculateArea(4.0));

        // Demonstrating dynamic polymorphism
        Shape myShape;

        myShape = new Circle(4.0);
        myShape.area();  // Should print: Area of Circle

        myShape = new Rectangle(5.0, 3.0);
        myShape.area();  // Should print: Area of Rectangle
    }
}
```

OUTPUT

javac Polymorphism.java

java Polymorphism

```
Area of Rectangle (static): 15.0
Area of Circle (static): 50.26548245743669
Area of Circle: 50.26548245743669
Area of Rectangle: 15.0
```

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

Database Command

```
create table Factorial(number int primary key, factorial int);
```

FactorialDB.java

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

public class FactorialDB extends JFrame {
    JTextField inputField;
    JTextArea outputField;
    JButton generateButton;
    Connection con;

    public FactorialDB(){
        setTitle("Factorial Generator");
        setSize(400, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new FlowLayout());

        inputField = new JTextField(10);
        generateButton = new JButton("Generate");
        generateButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e){
                generateFactorial();
            }
        });

        outputField = new JTextArea(5, 30);
        outputField.setEditable(false);
        add(new JLabel("Enter the no"));
        add(inputField);
        add(generateButton);
        add(outputField);
    }

    void generateFactorial(){
        try{
            outputField.setText("");
            int number = Integer.parseInt(inputField.getText());
            con = DriverManager.getConnection("jdbc:mysq://localhost:3306/MCA", "root",
"password");
            int factorial = 1;
            for(int i = number; i > 1; i--)
                factorial *= i;
            PreparedStatement st = con.prepareStatement("insert into Factorial (number,
factorial) values(?,?)");
            st.setInt(1, number);
            st.setInt(2, factorial);
            st.execute();
            outputField.setText("The factorial of " + number + " is " + factorial + ". Data
saved to database successfully!");
            st.close();
            con.close();
        }catch(NumberFormatException e){
            outputField.setText("Please enter a proper number");
        }catch(SQLException e2){
            outputField.setText("SQL EXCEPTION : " + e2.getMessage());
        }
    }

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

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

Database Command

```
create table Concatenate(text1 varchar(30), text2 varchar(30), concat varchar(60));
```

ConcatenateDB.java

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

public class ConcatenateDB extends JFrame {
    JTextField textField1;
    JTextField textField2;
    JTextArea outputField;
    JButton generateButton;
    Connection con;

    public ConcatenateDB(){
        setTitle("Concatenate String");
        setSize(400, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new FlowLayout());

        textField1 = new JTextField(10);
        textField2 = new JTextField(10);

        generateButton = new JButton("Concatename");
        generateButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e){
                concatString();
            }
        });

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

        add(new JLabel("Enter the first text"));
        add(textField1);
        add(new JLabel("Enter the second text"));
        add(textField2);
        add(generateButton);
        add(outputField);
    }

    void concatString(){
        try {
            String text1 = textField1.getText();
            String text2 = textField2.getText();
            outputField.setText("");
            con = DriverManager.getConnection("jdbc:mysq://localhost:3306/MCA", "root",
"password");
            PreparedStatement st = con.prepareStatement("insert into Concatenate(text1,
text2, concat) values(?, ?, ?)");
            st.setString(1, text1);
            st.setString(2, text2);
            st.setString(3, text1 + text2);
            st.executeUpdate();
            outputField.setText(text1 + text2 + "\nData added to database successfully");
            st.close();
            con.close();
        } catch (SQLException e) {
            outputField.setText("SQL EXCEPTION : " + e.getMessage());
        }
    }

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