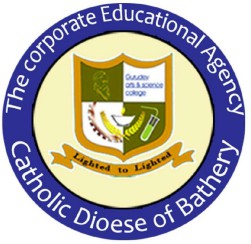
**GURUDEV ARTS AND SCIENCE COLLEGE**

MATHIL, PAYYANUR, KANNUR DIST.

(Affiliated to Kannur University)



**BACHELOR OF COMPUTER APPLICATION**

**ENTERPRISE JAVA PROGRAMMING**

**PRACTICAL RECORD**

:

:

**Name**

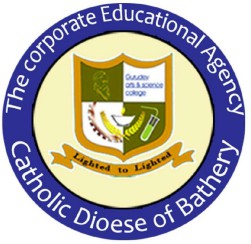
**Register No**



**GURUDEV ARTS AND SCIENCE COLLEGE**

MATHIL, PAYYANUR, KANNUR DIST.

(Affiliated to Kannur University)



**DEPARTMENT OF COMPUTER SCIENCE**

Certified that this is the bonafide record of practical work done by \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ of III BCA at Gurudev Arts and College, Mathil for the year 2024 to 2025.

Lecture Charge:



Examiner: 1.

2.

Head Of Dept. Computer Science

Submitted for University Examinations 2025

|  |  |  |
| --- | --- | --- |
| **S. No** | **Program** | **Page No.** |
| 1 | JDBC program to insert, delete, and update records into the Employee table. | **1-6** |
| 2 | JDBC program to connect to the Student table and implement record scrolling functions – first(), last(), next(), previous(), beforeFirst(), afterLast(), absolute(), and relative(). | **7-12** |
| 3 | JDBC program to display database metadata. | **13-14** |
| 4 | JDBC program to display ResultSet metadata. | **15–16** |
| 5 | RMI program for Complex number operations. | **17-19** |
| 6 | RMI program for Bank operations. | **20-23** |
| 7 | Create an HTML form to read student details such as Roll, Name, Age, Sex, Qualification, Percentage of Marks, etc., and write a servlet program that displays the same details. | **24–26** |
| 8 | Create an HTML form that reads a file name from the user. Write a servlet program that displays the contents of the specified file. | **27-28** |
| 9 | Session handling servlet that displays total number of visits to that page | **29-30** |
| 10 | CORBA program for Arithmetic operations. | **31-34** |

**TABLE OF CONTENTS**



**Q: JDBC program to insert, Delete and Update records into Employee table.**

import java.io.\*;

import java.sql.\*;

public class Employee {

static final String JDBC\_DRIVER = "com.mysql.cj.jdbc.Driver";

static final String DB\_URL = "jdbc:mysql://localhost/employee";

static final String USER = "root";

static final String PASS = "Mysql";

public static void main(String[] args) throws IOException {

int ch, upc;

int no, sal;

String name;

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

try {

Class.forName(JDBC\_DRIVER);

} catch (ClassNotFoundException e) {

System.out.println("Unable to load driver");

return;

}

try (Connection conn = DriverManager.getConnection(DB\_URL, USER, PASS)) {

System.out.println("\n.......Current Records.......\n");

System.out.println("ENO\tENAME\tSALARY\n");

try (Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery("SELECT \* FROM Employee1")) {

while (rs.next()) {

System.out.println(rs.getInt("eno") + "\t" + rs.getString("ename") + "\t" + rs.getInt("salary"));

}

}

do {

System.out.println("\n MENU \n 1.Insert \n 2.Update \n 3.Delete \n 4.Display \n 5.Exit \n Enter your choice:");

ch = Integer.parseInt(br.readLine());

switch (ch) {

case 1:

System.out.println("Enter employee number, name, and salary:");

no = Integer.parseInt(br.readLine());

name = br.readLine();

sal = Integer.parseInt(br.readLine());

String insertSQL = "INSERT INTO Employee1 (eno, ename, salary) VALUES (?, ?, ?)";

try (PreparedStatement pstmt = conn.prepareStatement(insertSQL)) {

pstmt.setInt(1, no);

pstmt.setString(2, name);

pstmt.setInt(3, sal);

pstmt.executeUpdate();

System.out.println("Records inserted");

}

break;

case 2:

System.out.println("Enter employee number of the record to be updated:");

no = Integer.parseInt(br.readLine());

System.out.println("Enter the new name and salary:");

name = br.readLine();

sal = Integer.parseInt(br.readLine());

String updateSQL = "UPDATE Employee1 SET ename=?, salary=? WHERE eno=?";

try (PreparedStatement pstmt = conn.prepareStatement(updateSQL)) {

conn.setAutoCommit(false); // Start transaction

pstmt.setString(1, name);

pstmt.setInt(2, sal);

pstmt.setInt(3, no);

upc = pstmt.executeUpdate();

if (upc != 0) {

conn.commit();

System.out.println("Records updated");

} else {

System.out.println("No such record exists");

}

} catch (SQLException e) {

System.out.println("Exception occurred: " + e.getMessage() + "\nRecords not updated\n");

conn.rollback();

} finally {

conn.setAutoCommit(true); // Reset auto-commit

}

break;

case 3:

System.out.println("Enter employee number of the record you want to delete:");

no = Integer.parseInt(br.readLine());

String deleteSQL = "DELETE FROM Employee1 WHERE eno=?";

try (PreparedStatement pstmt = conn.prepareStatement(deleteSQL)) {

conn.setAutoCommit(false); // Start transaction

pstmt.setInt(1, no);

upc = pstmt.executeUpdate();

if (upc != 0) {

conn.commit();

System.out.println("Records deleted");

} else {

System.out.println("No such record exists");

}

} catch (SQLException e) {

System.out.println("Exception occurred: " + e.getMessage() + "\nRecords not deleted\n");

conn.rollback();

} finally {

conn.setAutoCommit(true); // Reset auto-commit

}

break;

case 4:

String selectSQL = "SELECT \* FROM Employee1";

System.out.println("ENO\tENAME\tSALARY\n");

try (Statement stmt = conn.createStatement();

ResultSet rs1 = stmt.executeQuery(selectSQL)) {

while (rs1.next()) {

System.out.println(rs1.getInt("eno") + "\t" + rs1.getString("ename") + "\t" + rs1.getInt("salary"));

}

}

break;

case 5:

System.out.println("Exiting...");

break;

default:

System.out.println("Enter a valid choice");

break;

}

} while (ch != 5);

} catch (SQLException e) {

System.out.println("Connection failed: " + e.getMessage());

}

}

}

**OUTPUT:**

.......Current Records.......

ENO ENAME SALARY

101 John 50000

102 Jane 60000

MENU

1.Insert

2.Update

3.Delete

4.Display

5.Exit

Enter your choice: 1

Enter employee number, name, and salary:

103

Doe

70000

Records inserted

MENU

1.Insert

2.Update

3.Delete

4.Display

5.Exit

Enter your choice: 4

ENO ENAME SALARY

101 John 50000

102 Jane 60000

103 Doe 70000

MENU

1.Insert

2.Update

3.Delete

4.Display

5.Exit

Enter your choice: 2

Enter employee number of the record to be updated:

103

Enter the new name and salary:

Smith

75000

Records updated

MENU

1.Insert

2.Update

3.Delete

4.Display

5.Exit

Enter your choice: 4

ENO ENAME SALARY

101 John 50000

102 Jane 60000

103 Smith 75000

MENU

1.Insert

2.Update

3.Delete

4.Display

5.Exit

Enter your choice: 3

Enter employee number of the record you want to delete:

103

Records deleted

MENU

1.Insert

2.Update

3.Delete

4.Display

5.Exit

Enter your choice: 4

ENO ENAME SALARY

101 John 50000

102 Jane 60000

MENU

1.Insert

2.Update

3.Delete

4.Display

5.Exit

Enter your choice: 5

Exiting...

**Q: JDBC program to connect to Student table. Implement the record scrolling functions – first(), last(), next(), previous(), beforeFirst(), afterLast(), absolute() and relative().**

import java.io.\*;

import java.sql.\*;

public class StudentRecords {

    static final String JDBC\_DRIVER = "com.mysql.cj.jdbc.Driver";

    static final String DB\_URL = "jdbc:mysql://localhost/employee";

    static final String USER = "root";

    static final String PASS = "Mysql";

    public static void main(String[] args) throws IOException {

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

        try {

            Class.forName(JDBC\_DRIVER);

        } catch (ClassNotFoundException e) {

            System.out.println("Unable to load driver");

            return;

        }

        try (Connection conn = DriverManager.getConnection(DB\_URL, USER, PASS);

             Statement stmt = conn.createStatement(ResultSet.TYPE\_SCROLL\_INSENSITIVE, ResultSet.CONCUR\_READ\_ONLY);

             ResultSet rs = stmt.executeQuery("SELECT \* FROM Student")) {

            if (!rs.isBeforeFirst()) {

                System.out.println("No records in Student table");

                return;

            }

            while (true) {

                System.out.println("\nMENU\n1.First\n2.Last\n3.Next\n4.Previous\n5.Before First\n6.After Last\n7.Absolute\n8.Relative\n9.Exit\nEnter your choice:");

                int choice = Integer.parseInt(br.readLine());

                switch (choice) {

                    case 1:

                        if (rs.first()) {

                            displayRecord(rs);

                        } else {

                            System.out.println("No record found");

                        }

                        break;

                    case 2:

                        if (rs.last()) {

                            displayRecord(rs);

                        } else {

                            System.out.println("No record found");

                        }

                        break;

                    case 3:

                        if (rs.next()) {

                            displayRecord(rs);

                        } else {

                            System.out.println("No more records");

                        }

                        break;

                    case 4:

                        if (rs.previous()) {

                            displayRecord(rs);

                        } else {

                            System.out.println("No previous records");

                        }

                        break;

                    case 5:

                        rs.beforeFirst();

                        System.out.println("Moved before first record");

                        break;

                    case 6:

                        rs.afterLast();

                        System.out.println("Moved after last record");

                        break;

                    case 7:

                        System.out.println("Enter the row number to move to:");

                        int row = Integer.parseInt(br.readLine());

                        if (rs.absolute(row)) {

                            displayRecord(rs);

                        } else {

                            System.out.println("No such record");

                        }

                        break;

                    case 8:

                        System.out.println("Enter the relative position to move by:");

                        int pos = Integer.parseInt(br.readLine());

                        if (rs.relative(pos)) {

                            displayRecord(rs);

                        } else {

                            System.out.println("No such record");

                        }

                        break;

                    case 9:

                        System.out.println("Exiting...");

                        return;

                    default:

                        System.out.println("Enter a valid choice");

                        break;

                }

            }

        } catch (SQLException e) {

            System.out.println("Connection failed: " + e.getMessage());

        }

    }

    private static void displayRecord(ResultSet rs) throws SQLException {

        System.out.println("Student ID: " + rs.getInt("id") + ", Name: " + rs.getString("name") + ", Age: " + rs.getInt("age"));

    }

}

**OUTPUT**:

D:\NOTES\PRACTICAL\EJP>javac -cp ".;C:\Users\admin\Downloads\mysql-connector-j-9.2.0\mysql-connector-j-9.2.0\mysql-connector-j-9.2.0.jar" StudentRecords.java

D:\NOTES\PRACTICAL\EJP>java -cp ".;C:\Users\admin\Downloads\mysql-connector-j-9.2.0\mysql-connector-j-9.2.0\mysql-connector-j-9.2.0.jar" StudentRecords

MENU

1.First

2.Last

3.Next

4.Previous

5.Before First

6.After Last

7.Absolute

8.Relative

9.Exit

Enter your choice: 1

Student ID: 1, Name: Alice, Age: 20

MENU

1.First

2.Last

3.Next

4.Previous

5.Before First

6.After Last

7.Absolute

8.Relative

9.Exit

Enter your choice: 2

Student ID: 3, Name: Charlie, Age: 22

MENU

1.First

2.Last

3.Next

4.Previous

5.Before First

6.After Last

7.Absolute

8.Relative

9.Exit

Enter your choice: 3

No more records

MENU

1.First

2.Last

3.Next

4.Previous

5.Before First

6.After Last

7.Absolute

8.Relative

9.Exit

Enter your choice: 4

Student ID: 2, Name: Bob, Age: 21

MENU

1.First

2.Last

3.Next

4.Previous

5.Before First

6.After Last

7.Absolute

8.Relative

9.Exit

Enter your choice: 5

Moved before first record

MENU

1.First

2.Last

3.Next

4.Previous

5.Before First

6.After Last

7.Absolute

8.Relative

9.Exit

Enter your choice: 6

Moved after last record

MENU

1.First

2.Last

3.Next

4.Previous

5.Before First

6.After Last

7.Absolute

8.Relative

9.Exit

Enter your choice: 7

Enter the row number to move to: 2

Student ID: 2, Name: Bob, Age: 21

MENU

1.First

2.Last

3.Next

4.Previous

5.Before First

6.After Last

7.Absolute

8.Relative

9.Exit

Enter your choice: 8

Enter the relative position to move by: -1

Student ID: 1, Name: Alice, Age: 20

MENU

1.First

2.Last

3.Next

4.Previous

5.Before First

6.After Last

7.Absolute

8.Relative

9.Exit

Enter your choice: 9

Exiting...

**Q: JDBC program to display database metadata.**

import java.sql.\*;

public class DatabaseMetadataExample {

static final String JDBC\_DRIVER = "com.mysql.cj.jdbc.Driver";

static final String DB\_URL = "jdbc:mysql://localhost/employee";

static final String USER = "root";

static final String PASS = "Mysql";

public static void main(String[] args) {

try {

// Load JDBC driver

Class.forName(JDBC\_DRIVER);

// Establish connection

try (Connection conn = DriverManager.getConnection(DB\_URL, USER, PASS)) {

// Get database metadata

DatabaseMetaData dbMetaData = conn.getMetaData();

// Display general database information

System.out.println("Database Product Name: " + dbMetaData.getDatabaseProductName());

System.out.println("Database Product Version: " + dbMetaData.getDatabaseProductVersion());

System.out.println("Database URL: " + dbMetaData.getURL());

System.out.println("Database Username: " + dbMetaData.getUserName());

System.out.println("Driver Name: " + dbMetaData.getDriverName());

System.out.println("Driver Version: " + dbMetaData.getDriverVersion());

// Display tables information

System.out.println("\nTables in the database:");

try (ResultSet tables = dbMetaData.getTables(null, null, "%", new String[] { "TABLE" })) {

while (tables.next()) {

System.out.println("Table: " + tables.getString("TABLE\_NAME"));

}

}

// Display columns information for a specific table (e.g., Employee1)

String tableName = "Employee";

System.out.println("\nColumns in the table " + tableName + ":");

try (ResultSet columns = dbMetaData.getColumns(null, null, tableName, "%")) {

while (columns.next()) {

System.out.println("Column: " + columns.getString("COLUMN\_NAME") + " | Type: " + columns.getString("TYPE\_NAME") + " | Size: " + columns.getInt("COLUMN\_SIZE"));

}

}

} catch (SQLException e) {

System.out.println("Connection failed: " + e.getMessage());

}

} catch (ClassNotFoundException e) {

System.out.println("Unable to load driver");

}

}

}

**OUTPUT:**

Database Product Name: MySQL

Database Product Version: 8.0.26

Database URL: jdbc:mysql://localhost/employee

Database Username: root

Driver Name: MySQL Connector/J

Driver Version: mysql-connector-java-8.0.26

Tables in the database:

Table: Employee1

Columns in the table Employee1:

Column: eno | Type: INT | Size: 11

Column: ename | Type: VARCHAR | Size: 100

Column: salary | Type: INT | Size: 11

**Q: JDBC program to display Resultset metadata.**

import java.sql.\*;

public class ResultSetMetadataExample {

static final String JDBC\_DRIVER = "com.mysql.cj.jdbc.Driver";

static final String DB\_URL = "jdbc:mysql://localhost/employee";

static final String USER = "root";

static final String PASS = "Mysql";

public static void main(String[] args) {

try {

// Load JDBC driver

Class.forName(JDBC\_DRIVER);

// Establish connection

try (Connection conn = DriverManager.getConnection(DB\_URL, USER, PASS);

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery("SELECT \* FROM Employee1")) {

// Get ResultSet metadata

ResultSetMetaData rsMetaData = rs.getMetaData();

// Display number of columns

int columnCount = rsMetaData.getColumnCount();

System.out.println("Number of columns: " + columnCount);

// Display column details

for (int i = 1; i <= columnCount; i++) {

System.out.println("Column " + i + ":");

System.out.println(" Name: " + rsMetaData.getColumnName(i));

System.out.println(" Type: " + rsMetaData.getColumnTypeName(i));

System.out.println(" Size: " + rsMetaData.getColumnDisplaySize(i));

System.out.println(" Nullable: " + rsMetaData.isNullable(i));

System.out.println(" Auto Increment: " + rsMetaData.isAutoIncrement(i));

}

} catch (SQLException e) {

System.out.println("Connection failed: " + e.getMessage());

}

} catch (ClassNotFoundException e) {

System.out.println("Unable to load driver");

}

}

}

**OUTPUT:**

Number of columns: 3

Column 1:

Name: eno

Type: INT

Size: 11

Nullable: 0

Auto Increment: true

Column 2:

Name: ename

Type: VARCHAR

Size: 100

Nullable: 1

Auto Increment: false

Column 3:

Name: salary

Type: INT

Size: 11

Nullable: 1

Auto Increment: false

**Q: RMI program for Complex number operation**

**Step 1: Define the Remote Interface**

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface ComplexInter extends Remote {

int add1(int a1, int b1) throws RemoteException;

int add2(int a2, int b2) throws RemoteException;

}

**Step 2: Implement the Remote Interface**

import java.rmi.server.UnicastRemoteObject;

import java.rmi.RemoteException;

public class ComplexImpl extends UnicastRemoteObject implements ComplexInter {

private static final long serialVersionUID = 1L; // Best practice for Serializable class

public ComplexImpl() throws RemoteException {

super();

}

public int add1(int a1, int b1) throws RemoteException {

return a1 + b1;

}

public int add2(int a2, int b2) throws RemoteException {

return a2 + b2;

}

}

**Step 3: Create the Server Program**

import java.rmi.Naming;

public class ComplexReg {

public static void main(String args[]) {

try {

ComplexImpl comp = new ComplexImpl();

Naming.rebind("com", comp);

System.out.println("Object Registered");

} catch (Exception e) {

e.printStackTrace();

}

}

}

**Step 4: Create the Client Program**

import java.rmi.Naming;

import java.io.\*;

public class ComplexClient {

public static void main(String args[]) throws IOException {

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

int x, y, z, w, ans1, ans2;

System.out.println("Enter 1st complex number:");

x = Integer.parseInt(br.readLine());

y = Integer.parseInt(br.readLine());

System.out.println("Enter 2nd complex number:");

z = Integer.parseInt(br.readLine());

w = Integer.parseInt(br.readLine());

try {

ComplexInter obj = (ComplexInter) Naming.lookup("rmi://localhost/com");

ans1 = obj.add1(x, z);

ans2 = obj.add2(y, w);

System.out.println("Sum = " + ans1 + " + i" + ans2);

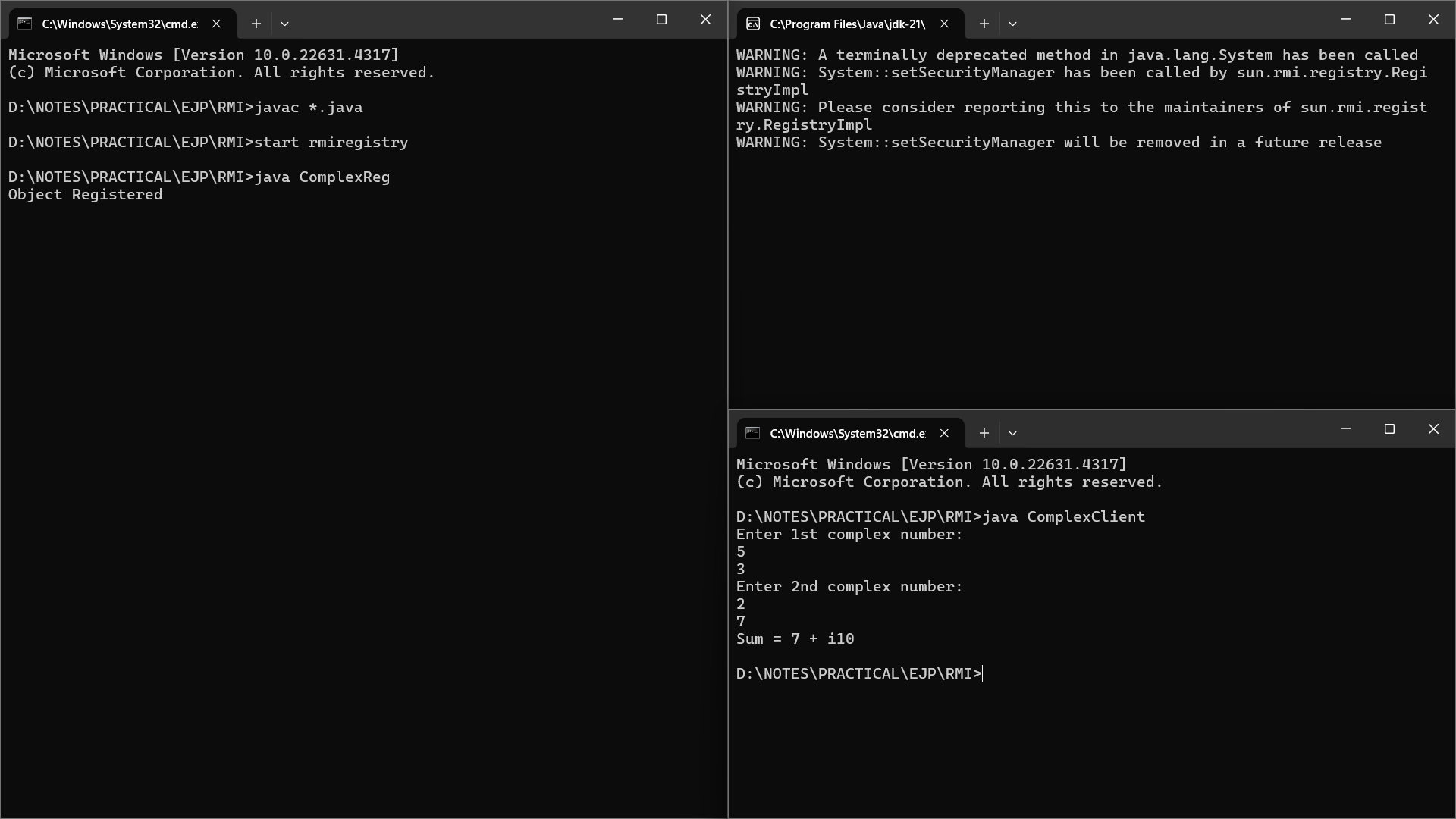
} catch (Exception e) {

System.out.println("Error: " + e);

}

}

}

**OUTPUT:**

**Q: RMI program for Bank operation.**

**Step 1: Define the Remote Interface:** Create an interface BankInter.java that extends java.rmi.Remote and declares the remote methods.

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface BankInter extends Remote {

void createAccount(String name) throws RemoteException;

void deposit(String name, double amount) throws RemoteException;

void withdraw(String name, double amount) throws RemoteException;

double getBalance(String name) throws RemoteException;

}

**Step 2: Implement the Remote Interface:** Create a class BankImpl.java that

implements the remote interface.

import java.rmi.server.UnicastRemoteObject;

import java.rmi.RemoteException;

import java.util.HashMap;

public class BankImpl extends UnicastRemoteObject implements BankInter {

private HashMap<String, Double> accounts = new HashMap<>();

public BankImpl() throws RemoteException {

super();

}

public void createAccount(String name) throws RemoteException {

accounts.putIfAbsent(name, 0.0);

}

public void deposit(String name, double amount) throws RemoteException {

accounts.put(name, accounts.getOrDefault(name, 0.0) + amount);

}

public void withdraw(String name, double amount) throws RemoteException {

double balance = accounts.getOrDefault(name, 0.0);

if (balance >= amount) {

accounts.put(name, balance - amount);

} else {

throw new RemoteException("Insufficient balance");

}

}

public double getBalance(String name) throws RemoteException {

return accounts.getOrDefault(name, 0.0);

}

}

**Step 3: Create the Server Program:** Create a class BankServer.java that registers the remote object with the RMI registry.

import java.rmi.Naming;

public class BankServer {

public static void main(String[] args) {

try {

Naming.rebind("rmi://localhost/BankService", new BankImpl());

System.out.println("Bank Service is running...");

} catch (Exception e) {

e.printStackTrace();

}

}

}

**Step 4: Create the Client Program:** Create a class BankClient.java that looks up the remote object and invokes its methods.

import java.rmi.Naming;

import java.util.Scanner;

public class BankClient {

public static void main(String[] args) {

try {

BankInter bank = (BankInter) Naming.lookup("rmi://localhost/BankService");

Scanner sc = new Scanner(System.in);

while (true) {

System.out.println("\n1. Create Account\n2. Deposit\n3. Withdraw\n4. Balance\n5. Exit");

System.out.print("Enter choice: ");

int choice = sc.nextInt();

sc.nextLine();

if (choice == 5) break;

System.out.print("Enter Name: ");

String name = sc.nextLine();

switch (choice) {

case 1:

bank.createAccount(name);

System.out.println("Account Created.");

break;

case 2:

System.out.print("Enter amount: ");

bank.deposit(name, sc.nextDouble());

System.out.println("Deposited.");

break;

case 3:

System.out.print("Enter amount: ");

bank.withdraw(name, sc.nextDouble());

System.out.println("Withdrawn.");

break;

case 4:

System.out.println("Balance: " + bank.getBalance(name));

break;

default:

System.out.println("Invalid choice!");

}

}

sc.close();

} catch (Exception e) {

e.printStackTrace();

}

}

}

**OUTPUT:**

D:\NOTES\PRACTICAL\EJP\RMI\_Bank>javac \*.java

D:\NOTES\PRACTICAL\EJP\RMI\_Bank>start rmiregistry

D:\NOTES\PRACTICAL\EJP\RMI\_Bank>java BankServer

Bank Service is running...

***//open new cmd window and move to RMI program folder***

D:\NOTES\PRACTICAL\EJP\RMI\_Bank>java BankClient

1. Create Account

2. Deposit

3. Withdraw

4. Balance

5. Exit

Enter choice: 1

Enter Name: Valentino Rossi

Account Created.

1. Create Account

2. Deposit

3. Withdraw

4. Balance

5. Exit

Enter choice: 2

Enter Name: Valentino Rossi

Enter amount: 2000000000

Deposited.

1. Create Account

2. Deposit

3. Withdraw

4. Balance

5. Exit

Enter choice: 4

Enter Name: Valentino Rossi

Balance: 2.0E9

**Q: Create an HTML form to read student details such as Roll, name,age, sex, qualification, percentage of marks etc. Write a servlet program that displays the same details.**

**Step 1: Create the HTML Form:** Create an HTML file named studentForm.html with the following content:

<!DOCTYPE html>

<html>

<head>

<title>Student Details Form</title>

</head>

<body>

<h2>Student Details Form</h2>

<form action="StudentServlet" method="post">

<label for="roll">Roll Number:</label>

<input type="text" id="roll" name="roll"><br><br>

<label for="name">Name:</label>

<input type="text" id="name" name="name"><br><br>

<label for="age">Age:</label>

<input type="text" id="age" name="age"><br><br>

<label for="sex">Sex:</label>

<input type="text" id="sex" name="sex"><br><br>

<label for="qualification">Qualification:</label>

<input type="text" id="qualification" name="qualification"><br><br>

<label for="percentage">Percentage of Marks:</label>

<input type="text" id="percentage" name="percentage"><br><br>

<input type="submit" value="Submit">

</form>

</body>

</html>

**Step 2: Create the Servlet:** Create a servlet class named StudentServlet.java:

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

public class StudentServlet extends HttpServlet {

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

response.setContentType("text/html");

PrintWriter out = response.getWriter();

String roll = request.getParameter("roll");

String name = request.getParameter("name");

String age = request.getParameter("age");

String sex = request.getParameter("sex");

String qualification = request.getParameter("qualification");

String percentage = request.getParameter("percentage");

out.println("<html><body>");

out.println("<h2>Student Details</h2>");

out.println("<p>Roll Number: " + roll + "</p>");

out.println("<p>Name: " + name + "</p>");

out.println("<p>Age: " + age + "</p>");

out.println("<p>Sex: " + sex + "</p>");

out.println("<p>Qualification: " + qualification + "</p>");

out.println("<p>Percentage of Marks: " + percentage + "</p>");

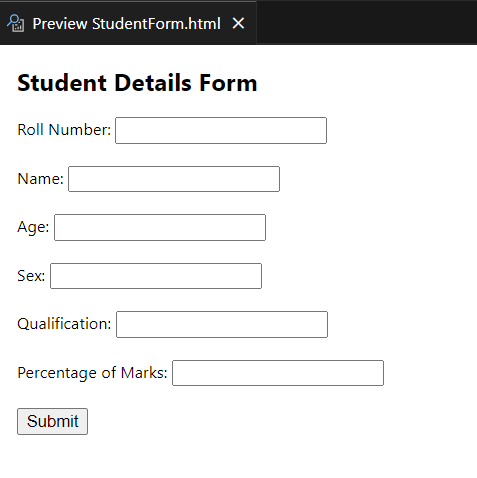
out.println("</body></html>");

out.close();

}

}

**OUTPUT:**



**Q: Create an HTML form that reads a file name from the user. Write a servlet program that displays the contents of the file, specified by the user.**

**Step 1: Create the HTML Form:** Create an HTML file named fileForm.html with the following content:

<!DOCTYPE html>

<html>

<head>

<title>File Reader Form</title>

</head>

<body>

<h2>File Reader Form</h2>

<form action="FileServlet" method="post">

<label for="filename">Enter the file name:</label>

<input type="text" id="filename" name="filename"><br><br>

<input type="submit" value="Submit">

</form>

</body>

</html>

**Step 2: Create the Servlet:** Create a servlet class named FileServlet.java:

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class FileServlet extends HttpServlet {

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

response.setContentType("text/html");

PrintWriter out = response.getWriter();

String filename = request.getParameter("filename");

File file = new File(filename);

out.println("<html><body>");

out.println("<h2>Contents of the file: " + filename + "</h2>");

if (file.exists() && file.isFile()) {

try (BufferedReader br = new BufferedReader(new FileReader(file))) {

String line;

while ((line = br.readLine()) != null) {

out.println(line + "<br>");

}

} catch (IOException e) {

out.println("<p>Error reading file: " + e.getMessage() + "</p>");

}

} else {

out.println("<p>File does not exist or is not a regular file.</p>");

}

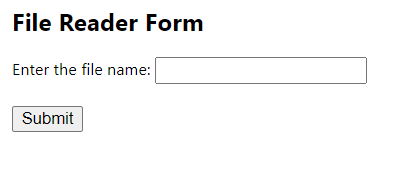
out.println("</body></html>");

out.close();

}

}

**OUTPUT:**



**Q: Session handling servlet that displays total number of visits to that page**

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import java.io.PrintWriter;

@WebServlet("/SessionCounterServlet")

public class SessionCounterServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

response.setContentType("text/html");

PrintWriter out = response.getWriter();

// Get the current session or create a new one

HttpSession session = request.getSession(true);

// Retrieve the visit count

Integer visitCount = (Integer) session.getAttribute("visitCount");

// If it's the user's first visit, initialize the counter

if (visitCount == null) {

visitCount = 1;

} else {

visitCount++;

}

// Store updated count in session

session.setAttribute("visitCount", visitCount);

// Display the visit count to the user

out.println("<html><body>");

out.println("<h2>Session Handling Example</h2>");

out.println("<p>Total Visits: " + visitCount + "</p>");

out.println("</body></html>");

}

}

**OUTPUT:**

**Q:** **CORBA program for arithmetic operation**

**1. Arithematic.idl (Corrected IDL)**

module ArithmeticModule {

interface Arithmetic {

float add(in float a, in float b);

float sub(in float a, in float b);

float mul(in float a, in float b);

float div(in float a, in float b);

};

};

**2. Generate CORBA Stubs**

idlj -fall Arithematic.idl

**3. ArithematicImp.java (Implementation Class)**

import ArithmeticModule.ArithmeticPOA;

public class ArithmeticImp extends ArithmeticPOA {

public float add(float a, float b) {

return a + b;

}

public float sub(float a, float b) {

return a - b;

}

public float mul(float a, float b) {

return a \* b;

}

public float div(float a, float b) {

if (b == 0) {

throw new ArithmeticException("Division by zero!");

}

return a / b;

}

}

**4. Reg.java (Server Registration)**

import ArithmeticModule.\*;

import org.omg.CORBA.\*;

import org.omg.CosNaming.\*;

import org.omg.PortableServer.\*;

public class Reg {

public static void main(String[] args) {

try {

ORB orb = ORB.init(args, null);

POA rootpoa = POAHelper.narrow(orb.resolve\_initial\_references("RootPOA"));

rootpoa.the\_POAManager().activate();

ArithmeticImp arithmeticImpl = new ArithmeticImp();

org.omg.CORBA.Object ref = rootpoa.servant\_to\_reference(arithmeticImpl);

Arithmetic href = ArithmeticHelper.narrow(ref);

org.omg.CORBA.Object objRef = orb.resolve\_initial\_references("NameService");

NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);

ncRef.rebind(ncRef.to\_name("ArithmeticService"), href);

System.out.println("Arithmetic Server Ready...");

orb.run();

} catch (Exception e) {

e.printStackTrace();

}

}

}

**5. Client.java (CORBA Client)**

import ArithmeticModule.\*;

import org.omg.CORBA.\*;

import org.omg.CosNaming.\*;

import java.io.\*;

public class Client {

public static void main(String[] args) {

try {

ORB orb = ORB.init(args, null);

org.omg.CORBA.Object objRef = orb.resolve\_initial\_references("NameService");

NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);

Arithmetic arith = ArithmeticHelper.narrow(ncRef.resolve\_str("ArithmeticService"));

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

System.out.println("Enter two numbers:");

float a = Float.parseFloat(rd.readLine());

float b = Float.parseFloat(rd.readLine());

System.out.println("Sum = " + arith.add(a, b));

System.out.println("Subtract = " + arith.sub(a, b));

System.out.println("Product = " + arith.mul(a, b));

System.out.println("Division = " + arith.div(a, b));

} catch (Exception e) {

e.printStackTrace();

}

}

}

**OUTPUT:**

D:\NOTES\PRACTICAL\EJP\CORBA\_Aritmetic> idlj -fall Arithmetic.idl

D:\NOTES\PRACTICAL\EJP\CORBA\_Aritmetic> javac ArithmeticModule/\*.java D:\NOTES\PRACTICAL\EJP\CORBA\_Aritmetic> ArithmeticImp.java Client.java Reg.java

D:\NOTES\PRACTICAL\EJP\CORBA\_Aritmetic> tnameserv -ORBInitialPort 1050

D:\NOTES\PRACTICAL\EJP\CORBA\_Aritmetic> java Reg -ORBInitialPort 1050

Arithmetic Server Ready...

//open new cmd window

D:\NOTES\PRACTICAL\EJP\CORBA\_Aritmetic> java Client -ORBInitialPort 1050

Enter two numbers:

5

2

Sum = 7.0

Subtract = 3.0

Product = 10.0

Division = 2.5