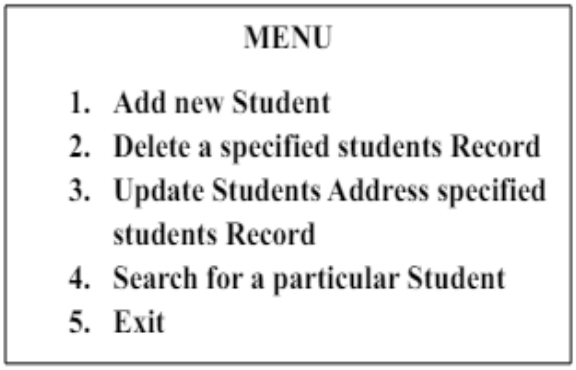
PART-B

1. Write a menu driven JDBC program to perform basic operations with Student Table.



Student

StRegNo	StName	Stdob	StAddress	StClass	StCourse

=

#Database: `studentdb`

```
CREATE TABLE `student` (
  `StRegNo` int(11) NOT NULL,
  `StName` varchar(255) NOT NULL,
  `Stdob` date NOT NULL,
  `StAddress` varchar(255) NOT NULL,
  `StClass` varchar(50) NOT NULL,
  `StCourse` varchar(50) NOT NULL
);
```

```
#StudentDatabase.java,

package studentmanagement;

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

```
public class StudentDatabase {

    static final String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
    static final String DB_URL = "jdbc:mysql://localhost:3306/StudentDB";

    static final String USER = "root";
    static final String PASS = "";

    public static void main(String[] args) {
        Connection conn = null;
        Scanner sc = new Scanner(System.in);

        try {
            Class.forName(JDBC_DRIVER);

            System.out.println("Connecting to database...");
            conn = DriverManager.getConnection(DB_URL, USER, PASS);

            while (true) {
                System.out.println("\nMENU");
                System.out.println("1. Add new Student");
                System.out.println("2. Delete a specified student's Record");
                System.out.println("3. Update Student's Address for specified
student's Record");
                System.out.println("4. Search for a particular Student");
                System.out.println("5. Exit");

                System.out.print("Enter your choice: ");
                int choice = sc.nextInt();

                switch (choice) {
                    case 1:
                        addStudent(conn);
                        break;
                    case 2:
                        deleteStudent(conn);
                        break;
                    case 3:
                        updateStudentAddress(conn);
                        break;
                    case 4:
                        searchStudent(conn);
                        break;
                    case 5:
                        System.out.println("Exiting...");
                        conn.close();
                        sc.close();
                        return;
```

```
                    default:
                        System.out.println("Invalid choice. Please enter a
number between 1 and 5.");
                }
            }

        } catch (SQLException | ClassNotFoundException se) {
        } finally {
            try {
                if (conn != null) conn.close();
            } catch (SQLException se) {
            }
        }
    }

    private static void addStudent(Connection conn) throws SQLException {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter Student details:");
        System.out.print("StReg No: ");
        int stRegNo = sc.nextInt();
        System.out.print("StName: ");
        String stName = sc.next();
        System.out.print("Stdob (YYYY-MM-DD): ");
        String stDob = sc.next();
        System.out.print("StAddress: ");
        String stAddress = sc.next();
        System.out.print("StClass: ");
        String stClass = sc.next();
        System.out.print("StCourse: ");
        String stCourse = sc.next();
        String sql = "INSERT INTO Student (StRegNo, StName, Stdob, StAddress,
StClass, StCourse) VALUES (?, ?, ?, ?, ?, ?)";
        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setInt(1, stRegNo);
            pstmt.setString(2, stName);
            pstmt.setString(3, stDob);
            pstmt.setString(4, stAddress);
            pstmt.setString(5, stClass);
            pstmt.setString(6, stCourse);

            int rowsInserted = pstmt.executeUpdate();
            if (rowsInserted > 0) {
                System.out.println("Student added successfully!");
            } else {
                System.out.println("Failed to add student.");
            }
        }
    }
```

```
    private static void deleteStudent(Connection conn) throws SQLException {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter StReg No of the student to delete: ");
        int stRegNo = sc.nextInt();

        String sql = "DELETE FROM Student WHERE StRegNo = ?";
        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setInt(1, stRegNo);
            int rowsDeleted = pstmt.executeUpdate();
            if (rowsDeleted > 0) {
                System.out.println("Student deleted successfully!");
            } else {
                System.out.println("Failed to delete student. Student not
found.");
            }
        }
    }

    private static void updateStudentAddress(Connection conn) throws
SQLException {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter StReg No of the student to update: ");
        int stRegNo = sc.nextInt();
        System.out.print("Enter new StAddress: ");
        String stAddress = sc.next();
        String sql = "UPDATE Student SET StAddress = ? WHERE StRegNo = ?";
        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, stAddress);
            pstmt.setInt(2, stRegNo);
            int rowsUpdated = pstmt.executeUpdate();
            if (rowsUpdated > 0) {
                System.out.println("Student's address updated successfully!");
            } else {
                System.out.println("Failed to update student's address.
Student not found.");
            }
        }
    }

    private static void searchStudent(Connection conn) throws SQLException {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter StReg No of the student to search: ");
        int stRegNo = sc.nextInt();
        String sql = "SELECT * FROM Student WHERE StRegNo = ?";
        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setInt(1, stRegNo);
```

```
            try (ResultSet rs = pstmt.executeQuery()) {
                if (rs.next()) {
                    System.out.println("StReg No: " + rs.getInt("StRegNo"));
                    System.out.println("StName: " + rs.getString("StName"));
                    System.out.println("Stdob: " + rs.getDate("Stdob"));
                    System.out.println("StAddress: " +
rs.getString("StAddress"));
                    System.out.println("StClass: " + rs.getString("StClass"));
                    System.out.println("StCourse: " +
rs.getString("StCourse"));
                } else {
                    System.out.println("Student not found.");
                }
            }
        }
    }
}
```

OUTPUT:

```
MENU
1. Add new Student
2. Delete a specified student's Record
3. Update Student's Address for specified student's Record
4. Search for a particular Student
5. Exit
Enter your choice: 1
Enter Student details:
StReg No: 1006
StName: anuksha
Stdob (YYYY-MM-DD): 2006-03-16
StAddress: karkala
StClass: 10
StCourse: science
Student added successfully!

MENU
1. Add new Student
2. Delete a specified student's Record
3. Update Student's Address for specified student's Record
4. Search for a particular Student
5. Exit
Enter your choice: 1
Enter Student details:
StReg No: 2004
StName: rakshitha
Stdob (YYYY-MM-DD): 2008-04-17
```

StAddress: hebri
StClass: 12
StCourse: commerce
Student added successfully!

MENU
1. Add new Student
2. Delete a specified student's Record
3. Update Student's Address for specified student's Record
4. Search for a particular Student
5. Exit
Enter your choice: 1
Enter Student details:
StReg No: 1
StName: durga
Stdob (YYYY-MM-DD): 2006-06-16
StAddress: karkala
StClass: 11
StCourse: commerec
Student added successfully!

MENU
1. Add new Student
2. Delete a specified student's Record
3. Update Student's Address for specified student's Record
4. Search for a particular Student
5. Exit
Enter your choice: 2

MENU
1. Add new Student
2. Delete a specified student's Record
3. Update Student's Address for specified student's Record
4. Search for a particular Student
5. Exit
Enter your choice: 3
Enter StReg No of the student to update: 1006
Enter new StAddress: onthibettu
Student's address updated successfully!

MENU
1. Add new Student
2. Delete a specified student's Record
3. Update Student's Address for specified student's Record
4. Search for a particular Student
5. Exit
Enter your choice: 4
Enter StReg No of the student to search: 1006
StRegNo: 1006
StName: anuksha
Stdob: 2006-03-16
StAddress: onthibettu
StClass: 10
StCourse: science

2. Write a menu driven JDBC program to perform basic operations with Bank Table.

MENU

1. **Add new Account Holder information.**
2. **Amount Deposit**
3. **Amount Withdrawal (Maintain minimum balance 500 Rs)**
4. **Display all information**
5. **Exit**

Bank

ACC_NO	ACC_NAME	ACC_ADDRESS	BALANCE

=

Database: `bankdb`

CREATE TABLE `bank` (

`ID` int(11) (AUTO INCREMENT),

`ACC_NO` int(11) NOT NULL,

`ACC_NAME` varchar(255) NOT NULL,

`ACC_ADDRESS` varchar(255) NOT NULL,

`BALANCE` double NOT NULL CHECK (`BALANCE` >= 500)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

```
--------------------------------------------------------------------------
# bankDatabase.java
package bankmanagement;
```

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

public class BankDatabase {
    static final String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
    static final String DB_URL = "jdbc:mysql://localhost:3306/bankdb";
    static final String USER = "root";
    static final String PASS = "";

    public static void main(String[] args) {
        Connection conn = null;
        Scanner sc = new Scanner(System.in);
        try {
            Class.forName(JDBC_DRIVER);
            System.out.println("Connecting to database...");
            conn = DriverManager.getConnection(DB_URL, USER, PASS);
            while (true) {
                System.out.println("\nMENU");
                System.out.println("1. Add new Account Holder information");
                System.out.println("2. Amount Deposit");
                System.out.println("3. Amount Withdrawal (Maintain minimum
balance 500 Rs)");
                System.out.println("4. Display all information");
                System.out.println("5. Exit");
                System.out.print("Enter your choice: ");
                int choice = sc.nextInt();
                switch (choice) {
                    case 1:
                        addAccount(conn);
                        break;
                    case 2:
                        depositAmount(conn);
                        break;
                    case 3:
                        withdrawAmount(conn);
                        break;
                    case 4:
                        displayAllAccounts(conn);
                        break;
                    case 5:
                        System.out.println("Exiting...");
                        conn.close();
                        sc.close();
                        return;
                    default:
                        System.out.println("Invalid choice. Please enter a
number between 1 and 5.");
                }
```

```
            }
        } catch (SQLException | ClassNotFoundException se) {
        } finally {
            try {
                if (conn != null) conn.close();
            } catch (SQLException se) {
            }
        }
    }
    private static void addAccount(Connection conn) throws SQLException {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter Account Holder details:");
        System.out.print("Account_Number: ");
        int accNo = sc.nextInt();
        System.out.print("Account_Name: ");
        String accName = sc.next();
        System.out.print("Account_Address: ");
        String accAddress = sc.next();
        System.out.print("Balance: ");
        double balance = sc.nextDouble();

        if (balance < 500) {
            System.out.println("Error: Initial balance must be at least 500.
Account not created.");
        } else {
            String sql = "INSERT INTO Bank (ACC_NO, ACC_NAME, ACC_ADDRESS,
BALANCE) VALUES (?, ?, ?, ?)";
            try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
                pstmt.setInt(1, accNo);
                pstmt.setString(2, accName);
                pstmt.setString(3, accAddress);
                pstmt.setDouble(4, balance);
                int rowsInserted = pstmt.executeUpdate();
                if (rowsInserted > 0) {
                    System.out.println("Account added successfully!");
                } else {
                    System.out.println("Failed to add account.");
                }
            }
        }
    }
    private static void depositAmount(Connection conn) throws SQLException {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter Acount_Number: ");
        int accNo = sc.nextInt();
        System.out.print("Enter amount to deposit: ");
        double amount = sc.nextDouble();
        String sql = "UPDATE Bank SET BALANCE = BALANCE + ? WHERE ACC_NO = ?";
```

```
        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setDouble(1, amount);
            pstmt.setInt(2, accNo);
            int rowsUpdated = pstmt.executeUpdate();
            if (rowsUpdated > 0) {
                System.out.println("Amount deposited successfully!");
            } else {
                System.out.println("Failed to deposit amount. Account not
found.");
            }
        }
    }
    private static void withdrawAmount(Connection conn) throws SQLException {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter Acount_Number: ");
        int accNo = sc.nextInt();
        System.out.print("Enter amount to withdraw: ");
        double amount = sc.nextDouble();
        String sqlCheckBalance = "SELECT BALANCE FROM Bank WHERE ACC_NO = ?";
        String sqlUpdateBalance = "UPDATE Bank SET BALANCE = ? WHERE ACC_NO =
?";
        try (PreparedStatement pstmtCheck =
conn.prepareStatement(sqlCheckBalance)) {
            pstmtCheck.setInt(1, accNo);
            ResultSet rs = pstmtCheck.executeQuery();
            if (rs.next()) {
                double balance = rs.getDouble("BALANCE");
                if (balance - amount >= 500) {
                    try (PreparedStatement pstmtUpdate =
conn.prepareStatement(sqlUpdateBalance)) {
                        pstmtUpdate.setDouble(1, balance - amount);
                        pstmtUpdate.setInt(2, accNo);
                        int rowsUpdated = pstmtUpdate.executeUpdate();
                        if (rowsUpdated > 0) {
                            System.out.println("Amount withdrawn
successfully!");
                        } else {
                            System.out.println("Failed to withdraw amount.");
                        }
                    }
                } else {
                    System.out.println("Insufficient balance! Minimum balance
of Rs.500 should be maintained.");
                }
            } else {
                System.out.println("Account not found.");
            }
        }
```

```
    }
    private static void displayAllAccounts(Connection conn) throws
SQLException {
        String sql = "SELECT * FROM Bank";
        try (Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {
            while (rs.next()) {
                System.out.println("Acount_Number: " + rs.getInt("ACC_NO"));
                System.out.println("Acount_Name: " +
rs.getString("ACC_NAME"));
                System.out.println("Acount_Address: " +
rs.getString("ACC_ADDRESS"));
                System.out.println("Balance: " + rs.getDouble("BALANCE"));
                System.out.println("------------------------");
            }
        }
    }
}
```

OUTPUT:

```
MENU
1. Add new Account Holder information
2. Amount Deposit
3. Amount Withdrawal (Maintain minimum balance 500 Rs)

4. Display all information
5. Exit
Enter your choice: 1
Enter Account Holder details:
Account_Number: 12345
Account_Name: anuksha
Account_Address: udupi
Balance: 500
Account added successfully!

MENU
1. Add new Account Holder information
2. Amount Deposit
3. Amount Withdrawal (Maintain minimum balance 500 Rs)
4. Display all information
5. Exit
Enter your choice: 1
Enter Account Holder details:
Account_Number: 34567
Account_Name: rakshitha
Account_Address: hebri
Balance: 10000
Account added successfully!

MENU
1. Add new Account Holder information
2. Amount Deposit
3. Amount Withdrawal (Maintain minimum balance 500 Rs)
```

```
Enter Account Holder details:
Account_Number: 8907
Account_Name: durga
Account_Address: karkala
Balance: 20000
Account added successfully!

MENU
1. Add new Account Holder information
2. Amount Deposit
3. Amount Withdrawal (Maintain minimum balance 500 Rs)
4. Display all information
5. Exit
Enter your choice: 4
Acount_Number: 12345
Acount_Name: anuksha
Acount_Address: udupi
Balance: 500.0
-------------------------
Acount_Number: 34567
Acount_Name: rakshitha
Acount_Address: hebri
Balance: 10000.0
-------------------------
Acount_Number: 8907
Acount_Name: durga
Acount_Address: karkala
Balance: 20000.0
```

```
1. Add new Account Holder information
2. Amount Deposit
3. Amount Withdrawal (Maintain minimum balance 500 Rs)
4. Display all information
5. Exit
Enter your choice: 2
Enter Acount_Number: 12345
Enter amount to deposit: 10000
Amount deposited successfully!

MENU
1. Add new Account Holder information
2. Amount Deposit
3. Amount Withdrawal (Maintain minimum balance 500 Rs)
4. Display all information
5. Exit
Enter your choice: 3
Enter Acount_Number: 8907
Enter amount to withdraw: 5000
Amount withdrawn successfully!

MENU
1. Add new Account Holder information
2. Amount Deposit
3. Amount Withdrawal (Maintain minimum balance 500 Rs)
4. Display all information
5. Exit
Enter your choice: 4
Acount_Number: 12345
```

3. Write a Java class called Tax with methods for calculating Income Tax. Have this class as a servant and create a server program and register in the rmiregistry. Write a client program to invoke these remote methods of the servant and do the calculations. Accept inputs interactively.

<₹ 3,00,000	No Tax
₹ 3,00,001 to ₹ 6,00,000	5%
₹ 6,00,001 to ₹ 9,00,000	10%
₹ 9,00,001 to ₹ 12,00,000	15%
₹ 12,00,001 to ₹ 15,00,000	20%
>₹ 15,00,000	30%

=

New->java->java Application ->

Source Package -> right Click->new->java class-> name as Tax and create same ->

Tax.java

```
package income_tax;

import java.rmi.*;
public interface Tax extends Remote
{
public String query(int search) throws RemoteException;
}
```

SearchQuery.java

```
package income_tax;

import java.rmi.*;
import java.rmi.server.*;
public class SearchQuery extends UnicastRemoteObject
                         implements Tax
{

    SearchQuery() throws RemoteException
    {
        super();
    }

    public String query(int salary) throws RemoteException {
    double result = 0.0;

    if (salary <= 300000) {
        result = 0;
    } else if (salary >= 300001 && salary <= 600000) {
        result = (salary) * 0.05;
    } else if (salary >= 600001 && salary <= 900000) {
        result =(salary) * 0.10;
    } else if (salary >= 900001 && salary <= 1200000) {
        result =(salary) * 0.15;
    } else if (salary >= 1200001 && salary <= 1500000) {
        result =(salary) * 0.20;
    } else {
        result =(salary) * 0.30;
    }

    return Double.toString(result);
}
}
```

ClientRequest.java

```
package income_tax;

import java.rmi.*;
import java.util.Scanner;
public class ClientRequest
{
```

```
public static void main(String args[])
{
String answer;
try
{
                    System.out.print("Please input salary : ");
                    Scanner sc=new Scanner(System.in);
                    int sal=sc.nextInt();

Tax access =
(Tax)Naming.lookup("rmi://localhost:1900" + "/udupi");
answer = access.query(sal);
System.out.println("The calculated tax for the salary of " + sal +" is " +
answer+" ");
}
catch(Exception se)
{
System.out.println(se);
}
}
}
```

SearchServer.java

```
package income_tax;

import java.rmi.*;
import java.rmi.registry.*;
public class SearchServer {
    public static void main(String args[]) {
            try
            {
                Tax obj = new SearchQuery();

                LocateRegistry.createRegistry(1900);

                Naming.rebind("rmi://localhost:1900" + "/udupi",obj);

                System.out.println("Income Tax server is running...");
            }
            catch(Exception se)
            {
                System.out.println(se);
            }
}
}
```

Click run file frst Search Server and next run Client Request then come output

OUTPUT:

run:

Please input salary : 4000000

The calculated tax for the salary of 4000000 is 1200000.0

BUILD SUCCESSFUL (total time: 9 seconds)

4. Write a Java class called SimpleInterest with methods for calculating simple interest. Have this class as a servant and create a server program and register in the rmiregistry. Write a client program to invoke these remote methods of the servant and do the calculations. Accept inputs at command prompt.

=

 # SI.java

```
package simple_interest;

import java.rmi.*;
public interface SI extends Remote
{
    double calculateSimpleInterest(double principal, double rate, double time)
throws RemoteException;
}
```

 # SIQuery.java

```java
package simple_interest;

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

public class SIQuery extends UnicastRemoteObject implements SI {
    public SIQuery() throws RemoteException {
        super();
    }

    @Override
    public double calculateSimpleInterest(double principal, double rate,
double time) throws RemoteException {
        return (principal * rate * time) / 100.0;
    }
}
```

SIClient.java

```java
package simple_interest;

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

public class SIClient {
    public static void main(String[] args) {
        try {
            SI simpleInterest = (SI)
Naming.lookup("rmi://localhost/SimpleInterestService");

            Scanner scanner = new Scanner(System.in);
            System.out.print("Enter principal amount: ");
            double principal = scanner.nextDouble();
            System.out.print("Enter rate of interest: ");
            double rate = scanner.nextDouble();
            System.out.print("Enter time period (in years): ");
            double time = scanner.nextDouble();

            double interest =
simpleInterest.calculateSimpleInterest(principal, rate, time);

            System.out.println("Simple Interest: " + interest);
        } catch (Exception e) {
```

```
            System.err.println("Simple Interest client exception: " +
e.getMessage());
            e.printStackTrace();
        }
    }
}
```

 # SIServer.java

```
package simple_interest;

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

public class SIServer {
    public static void main(String[] args) {
        try {
            SI simpleInterest = new SIQuery();

            LocateRegistry.createRegistry(1099);

            Naming.rebind("SimpleInterestService", simpleInterest);

            System.out.println("Simple Interest server is running...");
        } catch (Exception e) {
            System.err.println("Simple Interest service exception: " +
e.getMessage());
            e.printStackTrace();
        }
    }
}
```

Frst run SIServer nxt run SIClient then come output

OUTPUT:

run:

Enter principal amount: 5000

Enter rate of interest: 2

Enter time period (in years): 5

Simple Interest: 500.0

BUILD SUCCESSFUL (total time: 24 seconds)

5. Write a Servlet Program to perform Insert, update and View operations on Employee Table

Employee

Name	Password	Email	Country

Add New Employee

Name: Rahul Kumar
Password: ••••••••
Email: rahulkk@gmail.com
Country: India
Save Employee

view employees

Employees List

Id	Name	Password	Email	Country	Edit
63	Amit Kumar	amtkmjj45	amitkumar@gmail.com	India	edit
61	Rahul Kumar	rahul4000	rahulkk@gmail.com	India	edit
62	Sonoo Jaiswal	sonoobsk	sonoojaiswal1987@gmail.com	India	edit
44	adarsh kumar	kkkkk	adarsh232@gmail.com	India	edit

Update Employee

Name: Amit Kumar Rana
Password: ••••••••
Email: amitkumar12@gmail.com
Country: India
Edit & Save

=

#Database: `employee_db`

CREATE TABLE `employee` (

`Id` int(11) NOT NULL,

`Name` varchar(100) DEFAULT NULL,

`Password` varchar(100) DEFAULT NULL,

`Email` varchar(100) DEFAULT NULL,

`Country` varchar(100) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_general_ci;

 # EmployeeServlet.java

```
package web;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/EmployeeServlet")
public class EmployeeServlet extends HttpServlet {
    private Connection getConnection() throws Exception {
        Class.forName("com.mysql.jdbc.Driver");
        return
DriverManager.getConnection("jdbc:mysql://localhost:3306/employee_db", "root",
"");
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter();
```

```
             Connection con = getConnection()) {
            String name = request.getParameter("name");
            String password = request.getParameter("password");
            String email = request.getParameter("email");
            String country = request.getParameter("country");

            PreparedStatement ps = con.prepareStatement("INSERT INTO employee
(Name, Password, Email, Country) VALUES (?, ?, ?, ?)");
            ps.setString(1, name);
            ps.setString(2, password);
            ps.setString(3, email);
            ps.setString(4, country);
            int i = ps.executeUpdate();
            if (i > 0) {
                out.println("<p>Employee added successfully!</p>");
                response.sendRedirect("EmployeeServlet");
            } else {
                out.println("<p>Failed to add employee.</p>");
            }
        } catch (Exception e) {
            throw new ServletException(e);
        }
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter();
             Connection con = getConnection()) {
            PreparedStatement ps = con.prepareStatement("SELECT * FROM
employee");
            ResultSet rs = ps.executeQuery();
            out.println("<html><head><title>Employees
List</title></head><body>");
            out.println("<h2>Employees List</h2>");
            out.println("<table
border='1'><tr><th>Name</th><th>Password</th><th>Email</th><th>Country</th><th
>Edit</th></tr>");
            while (rs.next()) {
                out.println("<tr>");
                out.println("<td>" + rs.getString("Name") + "</td>");
                out.println("<td>" + rs.getString("Password") + "</td>");
                out.println("<td>" + rs.getString("Email") + "</td>");
                out.println("<td>" + rs.getString("Country") + "</td>");
                out.println("<td><a href='EditUpdateEmployeeServlet?id=" +
rs.getInt("ID") + "'>Edit</a></td>");
                out.println("</tr>");
            }
            out.println("</table>");
```

```
            out.println("<br><a href='index.html'>Go to Index</a>");
            out.println("</body></html>");
        } catch (Exception e) {
            throw new ServletException(e);
        }
    }
}
```

EditUpdateEmployeeServlet.java

```
package web;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/EditUpdateEmployeeServlet")
public class EditUpdateEmployeeServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        String id = request.getParameter("id");

        try {
            Class.forName("com.mysql.jdbc.Driver");
            Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/employee_db", "root",
"");
            PreparedStatement ps = con.prepareStatement("SELECT * FROM
employee WHERE ID=?");
            ps.setString(1, id);
            ResultSet rs = ps.executeQuery();

            if (rs.next()) {
                out.println("<html><head><title>Update
Employee</title></head><body>");
                out.println("<h2>Update Employee</h2>");
```

```
                out.println("<form action='EditUpdateEmployeeServlet'
method='POST'>");
                out.println("<input type='hidden' name='id' value='" +
rs.getString("ID") + "'>");
                out.println("Name: <input type='text' name='name' value='" +
rs.getString("Name") + "'><br><br>");
                out.println("Password: <input type='password' name='password'
value='" + rs.getString("Password") + "'><br><br>");
                out.println("Email: <input type='email' name='email' value='"
+ rs.getString("Email") + "'><br><br>");
                out.println("Country: <select name='country'>");
                out.println("<option value='' disabled>Select your
country</option>");
                out.println("<option value='India'" +
(rs.getString("Country").equals("India") ? " selected" : "") +
">India</option>");
                out.println("<option value='UK'" +
(rs.getString("Country").equals("UK") ? " selected" : "") + ">UK</option>");
                out.println("<option value='USA'" +
(rs.getString("Country").equals("USA") ? " selected" : "") + ">USA</option>");
                out.println("</select><br><br>");
                out.println("<input type='submit' value='Update'>");
                out.println("</form>");
                out.println("</body></html>");
            } else {
                out.println("<p>No employee found with ID: " + id + "</p>");
            }
            con.close();
        } catch (Exception e) {
            out.println("<p>Error: " + e.getMessage() + "</p>");
        }
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        String id = request.getParameter("id");
        String name = request.getParameter("name");
        String password = request.getParameter("password");
        String email = request.getParameter("email");
        String country = request.getParameter("country");
        try {
            Class.forName("com.mysql.jdbc.Driver");
            Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/employee_db", "root",
"");
            PreparedStatement ps = con.prepareStatement("UPDATE employee SET
Name=?, Password=?, Email=?, Country=? WHERE ID=?");
```

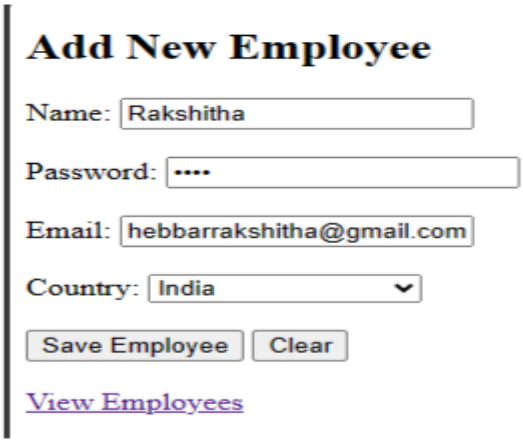
```
            ps.setString(1, name);
            ps.setString(2, password);
            ps.setString(3, email);
            ps.setString(4, country);
            ps.setString(5, id);
            int i = ps.executeUpdate();
            if (i > 0) {
                out.println("<p>Employee details updated successfully!</p>");
                response.sendRedirect("EmployeeServlet");
            } else {
                out.println("<p>Failed to update employee details.</p>");
            }
            con.close();
        } catch (Exception e) {
            out.println("<p>Error: " + e.getMessage() + "</p>");
        }
    }
}
```

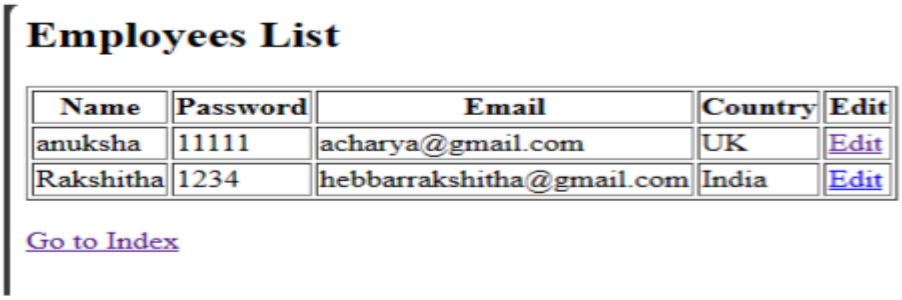
#index.html

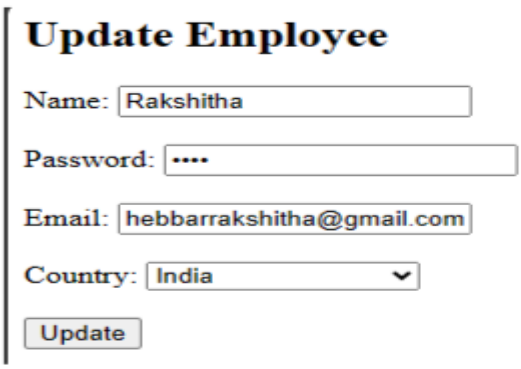
```
<!DOCTYPE html>
<html>
<head>
    <title>Add New Employee</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
    <h2>Add New Employee</h2>
    <form action="EmployeeServlet" method="POST">
        <label for="name">Name:</label>
        <input type="text" id="name" name="name" required><br><br>

        <label for="password">Password:</label>
        <input type="password" id="password" name="password" required><br><br>

        <label for="email">Email:</label>
        <input type="email" id="email" name="email" required><br><br>

        <label for="country">Country:</label>
        <select id="country" name="country" required>
            <option value="" disabled selected>Select your country</option>
            <option value="India">India</option>
            <option value="UK">UK</option>
            <option value="USA">USA</option>
```

```
        </select>
        <br><br>

        <input type="submit" value="Save Employee">
        <input type="reset" value="Clear">
    </form>
<br>
    <a href="EmployeeServlet">View Employees</a>
</body>
</html>
```

OUTPUT:







6. Write a java JSP program to get student information through a HTML and create a JAVA Bean Class, populate Bean and Display the same information through another JSP

=

Index.jsp

```
<!DOCTYPE html>
<html>
    <head>
        <title>Registration</title>
        <style>
            body{
                font-family: Arial, sans-serif;
                padding: 20px;
            }
            form{
                max-width: 400px;
                margin: 0 auto;
            }
            label {
                display: inline-block;
                widows: 100px;
                margin-bottom: 5px;
            }
        </style>
    </head>
    <body>
    <center><h1>Registration Page</h1></center>
    <form action="StudInfo.jsp" method="post">
        <label for="regNo"> Regn Num:</label>
        <input type="text" id="regNo" name="regNo" maxlength="30" /><br>

        <label for="firstName"> First Name:</label>
        <input type="text" id="firstName" name="firstName" maxlength="30"
/><br>

        <label for="lastName"> Last Name:</label>
        <input type="text" id="lastName" name="lastName" maxlength="30" /><br>

        <label for="course"> Course:</label>
        <input type="text" id="course" name="course" maxlength="30" /><br>

        <label for="sem">Sem:</label>
```

```
        <input type="text" id="sem" name="sem" maxlength="30" /><br>

        <input type="submit" value="Submit" />
        <input type="reset" value="Reset" />
    </form>
</body>
</html>
```

StudInfo.jsp

```
<!DOCTYPE html>
<html>
    <head>
        <title>Register User</title>
    </head>
    <body>
        <jsp:useBean id="stud" scope="session" class="java_beans.Bean">
            <jsp:setProperty name="stud" property="*"/>
        </jsp:useBean>
        <p>Your Registration Number is : <jsp:getProperty name="stud"
property="regNo" />.</p>
        <p>Your First Name is : <jsp:getProperty name="stud"
property="firstName" />.</p>
        <p>Your Last Name is : <jsp:getProperty name="stud"
property="lastName" />.</p>
        <p>Your Course is : <jsp:getProperty name="stud" property="course"
/>.</p>
        <p>Your Semester is : <jsp:getProperty name="stud" property="sem"
/>.</p>

    </body>
</html>
```

Bean.java

```
package java_beans;

public class Bean implements java.io.Serializable{
    private String regNo;
    private String firstName;
    private String lastName;
    private String course;
    private int sem;
    public String getRegNo(){
```

```
        return regNo;
    }
    public String getFirstName(){
        return firstName;
    }
    public String getLastName(){
        return lastName;
    }
    public String getCourse(){
        return course;
    }
    public int getSem(){
        return sem;
    }
    public void setRegNo(String newRegNo){
        this.regNo = newRegNo;
    }
     public void setFirstName(String newFirstName){
        this.firstName = newFirstName;
    }
     public void setLastName(String newLastName){
        this.lastName = newLastName;
    }
     public void setCourse(String newCourse){
        this.course = newCourse;
    }
     public void setSem(int newSem){
        this.sem = newSem;
    }
}
```

OUTPUT:

Your Registration Number is : 1234.

Your First Name is : Rakshitha.

Your Last Name is : Hebbar.

Your Course is : bca.

Your Semester is : 5.

7. Write a menu driven program to create a linked list and perform the following operations.

a. to Insert some Elements at the Specified Position

b. swap two elements in a linked list

c. to Iterate a LinkedList in Reverse Order

d. to Compare Two LinkedList

e. to Convert a LinkedList to ArrayList

=

 # list.java

```
package linked_list;
import java.util.*;

public class list {
    public static void main(String[] args) {
        LinkedList<Integer> linkedList = new LinkedList<>();
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("\nMENU");
            System.out.println("1. Insert Elements at Specified Position");
            System.out.println("2. Swap Two Elements");
            System.out.println("3. Iterate LinkedList in Reverse Order");
            System.out.println("4. Compare Two LinkedLists");
            System.out.println("5. Convert LinkedList to ArrayList");
            System.out.println("6. Exit");

            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();
            switch (choice) {
                case 1:
                    System.out.print("Enter element to insert: ");
                    int element = scanner.nextInt();
                    System.out.print("Enter position to insert: ");
                    int position = scanner.nextInt();
                    linkedList.add(position, element);
                    System.out.println("Element " + element + " inserted at
position " + position);
                    System.out.println("Inserted elements: " + linkedList);
```

```
                    break;
                case 2:
                    System.out.print("Enter first element index to swap: ");
                    int first = scanner.nextInt();
                    System.out.print("Enter second element index to swap: ");
                    int second = scanner.nextInt();
                    Collections.swap(linkedList, first, second);
                    System.out.println("Elements swapped successfully!");
                    System.out.println("New list after swapping: " +
linkedList);
                    break;
                case 3:
                    ListIterator<Integer> iterator =
linkedList.listIterator(linkedList.size());
                    System.out.println("LinkedList in Reverse Order: " +
reverseList(linkedList));
                    break;
                case 4:
                    System.out.println("Enter elements for the second
LinkedList (space-separated, -1 to stop):");
                    LinkedList<Integer> linkedList2 = new LinkedList<>();
                    while (true) {
                        int num = scanner.nextInt();
                        if (num == -1) break;
                        linkedList2.add(num);
                    }
                    System.out.println(compareLinkedLists(linkedList,
linkedList2) ? "LinkedLists are equal." : "LinkedLists are not equal.");
                    break;
                case 5:
                    System.out.println("LinkedList converted to ArrayList: " +
new ArrayList<>(linkedList));
                    break;
                case 6:
                    System.out.println("Exiting...");
                    scanner.close();
                    return;
                default:
                    System.out.println("Invalid choice. Please enter a number
between 1 and 6.");
            }
        }
    }

    private static List<Integer> reverseList(LinkedList<Integer> list) {
        LinkedList<Integer> reversedList = new LinkedList<>(list);
        Collections.reverse(reversedList);
        return reversedList;
```

```
    }

    private static boolean compareLinkedLists(LinkedList<Integer> list1,
LinkedList<Integer> list2) {
        List<Integer> sortedList1 = new ArrayList<>(list1);
        List<Integer> sortedList2 = new ArrayList<>(list2);
        Collections.sort(sortedList1);
        Collections.sort(sortedList2);
        return sortedList1.equals(sortedList2);
    }
}
```

OUTPUT:

run:

MENU

1. Insert Elements at Specified Position

2. Swap Two Elements

3. Iterate LinkedList in Reverse Order

4. Compare Two LinkedLists

5. Convert LinkedList to ArrayList

6. Exit

Enter your choice: 1

Enter element to insert: 2

Enter position to insert: 0

Element 2 inserted at position 0

Inserted elements: [2]

MENU

1. Insert Elements at Specified Position

2. Swap Two Elements

3. Iterate LinkedList in Reverse Order

4. Compare Two LinkedLists

5. Convert LinkedList to ArrayList

6. Exit

Enter your choice: 1

Enter element to insert: 3

Enter position to insert: 1

Element 3 inserted at position 1

Inserted elements: [2, 3]

MENU

1. Insert Elements at Specified Position

2. Swap Two Elements

3. Iterate LinkedList in Reverse Order

4. Compare Two LinkedLists

5. Convert LinkedList to ArrayList

6. Exit

Enter your choice: 1

Enter element to insert: 3

Enter position to insert: 2

Element 3 inserted at position 2

Inserted elements: [2, 3, 3]

MENU

1. Insert Elements at Specified Position

2. Swap Two Elements

3. Iterate LinkedList in Reverse Order

4. Compare Two LinkedLists

5. Convert LinkedList to ArrayList

6. Exit

Enter your choice: 2

Enter first element index to swap: 0

Enter second element index to swap: 2

Elements swapped successfully!

New list after swapping: [3, 3, 2]

MENU

1. Insert Elements at Specified Position

2. Swap Two Elements

3. Iterate LinkedList in Reverse Order

4. Compare Two LinkedLists

5. Convert LinkedList to ArrayList

6. Exit

Enter your choice: 3

LinkedList in Reverse Order:

[2, 3, 3]

MENU

1. Insert Elements at Specified Position

2. Swap Two Elements

3. Iterate LinkedList in Reverse Order

4. Compare Two LinkedLists

5. Convert LinkedList to ArrayList

6. Exit

Enter your choice: 4

Enter elements for the second LinkedList (space-separated, -1 to stop):

234

3

4

3

-1

LinkedLists are not equal.

MENU

1. Insert Elements at Specified Position

2. Swap Two Elements

3. Iterate LinkedList in Reverse Order

4. Compare Two LinkedLists

5. Convert LinkedList to ArrayList

6. Exit

Enter your choice: 4

Enter elements for the second LinkedList (space-separated, -1 to stop):

2 3 3

-1

LinkedLists are equal.

MENU

1. Insert Elements at Specified Position

2. Swap Two Elements

3. Iterate LinkedList in Reverse Order

4. Compare Two LinkedLists

5. Convert LinkedList to ArrayList

6. Exit

Enter your choice: 5

LinkedList converted to ArrayList: [3, 3, 2]

MENU

1. Insert Elements at Specified Position

2. Swap Two Elements

3. Iterate LinkedList in Reverse Order

4. Compare Two LinkedLists

5. Convert LinkedList to ArrayList

6. Exit

Enter your choice: 6

Exiting...

BUILD SUCCESSFUL (total time: 2 minutes 25 seconds)

8. Implement a java application based on the MVC design pattern. Input student Rolnlo, name ,marks in three subject calculate result and grade and display the result in neat format.

Percentage of Marks	Grade
Above 90%	A
80% to 90%	B
70% to 80%	C
60% to 70%	D
Below 60%	E

=

StudentMVC.java

```
package mvc;
import java.util.Scanner;

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

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

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

        System.out.print("Enter marks for Subject 1 : ");
        int mark1 = scanner.nextInt();

        System.out.print("Enter marks for Subject 2 : ");
        int mark2 = scanner.nextInt();

        System.out.print("Enter marks for Subject 3 : ");
```

```
        int mark3 = scanner.nextInt();

        scanner.close();

        StudentModel student = new StudentModel(rollNo, name, mark1, mark2,
mark3);
        StudentController controller = new StudentController(student);
        StudentView view = new StudentView();
        view.displayStudentDetails(controller.getStudentDetails());
    }
    }
```

StudentController.java

```
package mvc;

public class StudentController {
    private StudentModel student;

    public StudentController(StudentModel student) {
        this.student = student;
    }

    public String getStudentDetails(){
        StringBuilder details = new StringBuilder();
        details.append("Student Roll No :
").append(student.getRollNo()).append("\n");
        details.append("Student Name :
").append(student.getName()).append("\n");
        details.append("Marks in Subject 1 :
").append(student.getMark1()).append("\n");
        details.append("Marks in Subject 2 :
").append(student.getMark2()).append("\n");
        details.append("Marks in Subject 3 :
").append(student.getMark3()).append("\n");
        details.append("Percentage : ").append(String.format("%.2f",
student.calculatePercentage())).append("%\n");
        details.append("Grade : ").append(student.getGrade()).append("\n");
        return details.toString();
    }
}
```

StudentView.java

```
package mvc;
```

```java
public class StudentView {
    public void displayStudentDetails(String studentDetails){
        System.out.println(studentDetails);
    }

}
```

StudentModel.java

```java
package mvc;

public class StudentModel {
    private String rollNo;
    private String name;
    private int mark1;
    private int mark2;
    private int mark3;

    public StudentModel(String rollNo, String name, int mark1, int mark2, int
mark3){
        this.rollNo = rollNo;
        this.name = name;
        this.mark1 = mark1;
        this.mark2 = mark2;
        this.mark3 = mark3;
    }
    public String getRollNo(){
        return rollNo;
    }

    public String getName(){
        return name;
    }

    public int getMark1(){
        return mark1;
    }

    public int getMark2(){
        return mark2;
    }

    public int getMark3(){
        return mark3;
```

```
    }
    public double calculatePercentage(){
        return (mark1 + mark2 + mark3)/3.0;
    }
    public String getGrade(){
        double percentage = calculatePercentage();
        if(percentage > 90){
            return "A";
        }else if(percentage >80){
            return "B";
        }else if(percentage > 70){
            return "C";
        }else if(percentage >60){
            return "D";
        }else{
            return "E";
        }

    }

}
```

OUTPUT:

run:

Enter Roll Number : 018

Enter name : durgashree

Enter marks for Subject 1 : 98

Enter marks for Subject 2 : 99

Enter marks for Subject 3 : 95

Student Roll No : 018

Student Name : durgashree

Marks in Subject 1 : 98

Marks in Subject 2 : 99

Marks in Subject 3 : 95

Percentage : 97.33%

Grade : A