

# Software Construction & Development (SE3001)

## Sessional-II Exam

Date: November 4, 2024

Course Instructor(s)

Dr. Farooq Ahmed, Mr. Waqas Ali

Total Time: 1 hour

Total Marks: 30

Total Questions: 3

22i-2505

Roll No

SE-SA

Section

TKS

Student Signature

Do not write below this line

Attempt all questions on the answer sheet

**CLO 2: Implement software design patterns as part of software construction activity**

### Question 1

[10 marks]

Consider the UI mock-up provided above for the Exam Department at FAST NUCES Lahore. The form prompts the user to enter the exam name, allow selection of the exam day from a drop-down list displaying the names of the week (Monday, Tuesday, ...), and input the exam date in the format (dd/mm/yy). Invigilator names are hardcode as "Invigilator One" "Invigilator Two" and "Invigilator Three" in the dropdown. Additionally, the Level field should allow the user to select "PhD," "MS," or "BS" using radio buttons.

Exam Panel

Assign Exam Duty Form

Exam Name :

Exam Day :

Exam Date :

Invigilator Name :

Level : ☐ PhD ☐ MS ☒ BS

### CODE:

Implement Event Listener for ASSIGN button so that when user clicks on it, a record is inserted in the database to assign the duty of specified exam to selected invigilator on the mentioned date, if not assigned already. Assume that name of database is "exam\_db" and table is "exam\_table".

Organize the code properly using relevant architectural / design pattern.

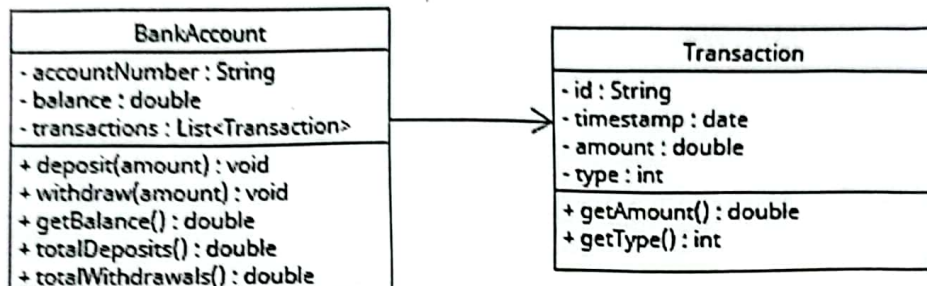
**NOTE:** No need to provide complete UI code as long as related event listener and necessary implementation is provided.

## CLO 3: Design Test Cases for a Software System

### Question 2

[10 marks]

Write unit tests (using JUnit) for the classes given in the diagram for a simple banking application:



Bank Account maintains a balance of amount that can be updated through deposit or withdraw transactions (type = 1 for deposit and type = -1 for withdraw), that are tracked. Deposit increases the balance and withdraw decreases the balance, appropriate to the transaction amount. Withdrawal transaction is rejected if the balance can become negative. Total deposits and total withdrawals, to and from the account, can be computed through respective functions using the transaction history.

## CLO 4: Use a version control system as part of software construction activity

### Question 3


[10 marks]

Consider that you are working on a new feature for a popular shopping store to predict customer retention. The feature is almost half-way through and you are well on track to complete it by the end of sprint. Meanwhile, your client requests an urgent fix for a security issue that is preventing the customers from signing into the application.

Making good use of Git Version Control System, answer the following questions precisely:

- What steps will you take to accommodate client's request (for the security issue) ?
- What steps will you take after completing the customer retention prediction feature at the end of sprint?
- How can Git help to bill the client for the services rendered (customer retention prediction feature as well as security issue) at the end of the sprint?

# National University of Computer and Emerging Sciences, Lahore Campus

	Course Name:	Software Construction & Development	Course Code:	CS-3001
	Degree Program:	BS(SE)	Semester:	Fall 2023
	Exam Duration:	60 Minutes	Total Marks:	45
	Paper Date:	10 - Nov - 2023	Weight:	15.00%
	Section:	ALL	Page(s):	6
	Exam Type:	Midterm-II		

Student Name:\_\_\_\_\_ Roll No.\_\_\_\_\_ Section:\_\_\_\_\_

**Instruction/Notes:** Attempt all questions. Do not use pencil or red ink. In case of confusion or ambiguity make a reasonable assumption. **Do not attach any extra sheet.** Use extra sheet for rough work only. A **double-sided hand-written** cheat sheet is allowed but it shouldn't be photo-copied.

## Question 1 [CLO-1]

5+10=15 points

Origin:

v

Destination:

v

Departure date:

Return date:

Passengers:

Adults:

-

1

+

Children:

-

0

+

Ticket:

☒ One-way

☐ Return

Search

Consider a UI mock-up given above for a Flight Reservation System. **Origin** and **Destination** are dropdowns populated with a list of cities. **Departure date** and **Return date** are text boxes that stores date in **mm/dd/yyyy** format. **Passengers** information maintains the count of **adults** and **children** intending to travel. The count for each can be updated through corresponding **increment (+)** and **decrement (-)** buttons. **Ticket** type can be either **One-way** or **Return**, controlled through a radio button group. If **One-way**, then **Return date** is disabled, but if ticket type is selected as **Return** then **Return date** textbox will be enabled.

Write event handling code for:

- enable / disable **Return date** textbox based upon Ticket type.
- increment / decrement count for adult and children passengers, using a single event handler.



Excise Department maintains the registration record of vehicles using the following class:

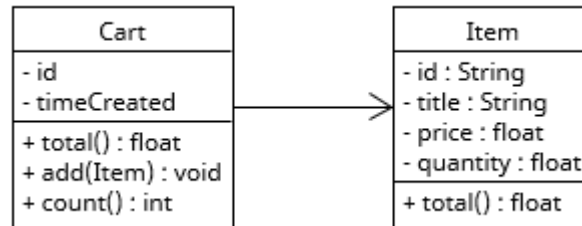
Vehicle
- registrationNumber : String - registrationDate : Date - engineNumber : String - make : String - modelName: String - modelYear : Integer - bodyType : String
+ save()

Suppose they want to simultaneously store the information both in: (a) a relational database management system (e.g. MySql, SQL Server, etc.); and (b) a text file where each vehicle takes one line with its values comma-separated; for the purpose of reliability. Write code to support this requirement using relevant architectural and design patterns.



**Question 3 [CLO-3]****10 points**

Write unit tests (using JUnit) for the following class diagram of a shopping cart system:




Item total provides the gross price (price x quantity) of a single item whereas Cart total provides the gross price of all items in a cart. Items can be added to the cart using add function. Count provides the number of items added to cart at any specific point in time.





# National University of Computer and Emerging Sciences, Lahore Campus

	Course Name:	Software Construction & Development	Course Code:	CS-3001
	Degree Program:	BS(SE)	Semester:	Fall 2023
	Exam Duration:	60 Minutes	Total Marks:	45
	Paper Date:	10 - Nov - 2023	Weight:	15.00%
	Section:	ALL	Page(s):	6
	Exam Type:	Midterm-II		

Student Name: \_\_\_\_\_ Roll No. \_\_\_\_\_ Section: \_\_\_\_\_

**Instruction/Notes:** Attempt all questions. Do not use pencil or red ink. In case of confusion or ambiguity make a reasonable assumption. **Do not attach any extra sheet.** Use extra sheet for rough work only. A **double-sided hand-written** cheat sheet is allowed but it shouldn't be photo-copied.

## Question 1 [CLO-1]

5+10=15 points

Origin:

Destination:

Departure date:

Return date:

Passengers:

Adults:

Children:

Ticket: ☒ One-way ☐ Return

Consider a UI mock-up given above for a Flight Reservation System. **Origin** and **Destination** are dropdowns populated with a list of cities. **Departure date** and **Return date** are text boxes that stores date in **mm/dd/yyyy** format. **Passengers** information maintains the count of **adults** and **children** intending to travel. The count for each can be updated through corresponding **increment (+)** and **decrement (-)** buttons. **Ticket** type can be either **One-way** or **Return**, controlled through a radio button group. If **One-way**, then **Return date** is disabled, but if ticket type is selected as **Return** then **Return date** textbox will be enabled.

Write event handling code for:

- (a) enable / disable **Return date** textbox based upon Ticket type.
- (b) increment / decrement count for adult and children passengers, using a single event handler.

## Solution

```
class FlightReservationUI extends JFrame{

    // components go here ...

    public FlightReservationUI(){
        // instantiate components and set layout and other properties ...
        TicketTypeListener ticketListener = new TicketTypeListener();
        PassengerCountListener countListener = new PassengerCountListener();
        onewayTicketRadioButton.addActionListener(ticketListener);
        returnTicketRadioButton.addActionListener(ticketListener);
        adultCountIncrementButton.addActionListener(countListener);
        adultCountDecrementButton.addActionListener(countListener);
        childrenCountIncrementButton.addActionListener(countListener);
        childrenCountDecrementButton.addActionListener(countListener);
    }

    private void increment(JTextField field){
        Integer currentValue = Integer.parseInt(field.getText());
        field.setText("" + (currentValue + 1));
    }

    private void decrement(JTextField field){
        Integer currentValue = Integer.parseInt(field.getText());
        field.setText("" + (currentValue - 1));
    }

    private class TicketTypeListener implements ActionListener{
        public void actionPerformed(ActionEvent e){
            if (e.getSource().equals(onewayTicketRadioButton){
                returnDateField.setEnabled(false);
            }

            if (e.getSource().equals(returnTicketRadioButton){
                returnDateField.setEnabled(true);
            }
        }
    }

    private class PassengerCountListener implements ActionListener{
        public void actionPerformed(ActionEvent e){
            if (e.getSource().equals(adultCountIncrementButton){
                increment(adultCountField);
            }

            if (e.getSource().equals(adultCountDecrementButton){
                decrement(adultCountField);
            }

            if (e.getSource().equals(childrenCountIncrementButton){
                increment(childrenCountField);
            }

            if (e.getSource().equals(childrenCountDecrementButton){
                decrement(childrenCountField);
            }
        }
    }
}
```

Excise Department maintains the registration record of vehicles using the following class:

Vehicle
- registrationNumber : String - registrationDate : Date - engineNumber : String - make : String - modelName: String - modelYear : Integer - bodyType : String
+ save()

Suppose they want to simultaneously store the information both in: (a) a relational database management system (e.g. MySQL, SQL Server, etc.); and (b) a text file where each vehicle takes one line with its values comma-separated; for the purpose of reliability. Write code to support this requirement using relevant architectural and design patterns.

### Solution

```

interface VehicleDAO{
    public boolean save(Hashtable<String,String> data);
}

class VehicleDbDAO implements VehicleDAO{
    public boolean save(Hashtable<String, String> data) {
        int count = 0;
        try{
            Connection conn = getConnection();    // assuming it is implemented
            PreparedStatement stmt = updateStatement(conn,data);
            count = stmt.executeUpdate();
            if (count == 0){
                stmt = insertStatement(conn,data);
                count = stmt.executeUpdate();
            }
        } catch(SQLException ex){
            return false;
        }
        return count > 0 ? true : false;
    }

    private PreparedStatement updateStatement(Connection conn,
        Hashtable<String,String> data) throws SQLException{
        String query = "update vehicle set regDate = ?, engNum = ?, make = ?, modelName
= ?, modelYear = ?, bodyType = ? where regNum = ?";
        PreparedStatement stmt = conn.prepareStatement(query);

        stmt.setString(1,data.get("registrationDate"));
        // other fields ...
        stmt.setString(7,data.get("registrationNumber"));

        return stmt;
    }
}

// similarly write insertStatement function

```

```

public class VehicleFileDAO implements IVehicleDAO{

    File file;
    Hashtable<String,ArrayList<String>> contents;

    public VehicleFileDAO(String path){
        file = new File(path);
        contents = new Hashtable<>();
    }

    public boolean save(Hashtable<String, String> data) {
        ArrayList<String> row = new ArrayList<>();
        row.add(data.get("registrationNumber"));
        // similarly add other fields ...

        if(contents.get(data.get("registrationNumber")) != null){
            contents.replace(data.get("id"), row);
        } else{
            contents.put(data.get("registrationNumber"), row);
        }

        write();
        return true;
    }

    private void write(){
        try{
            BufferedWriter writer = new BufferedWriter(new FileWriter(file));
            for(ArrayList<String> row : contents.values()){
                for(String col : row){
                    writer.append(col + ",");
                }
                writer.append("\n");
            }
            writer.close();
        }
        catch(IOException ex){ }
    }
}

class Vehicle{
    private String registrationNumber;
    // other fields go here

    private ArrayList<IVehicleDAO> datasources;

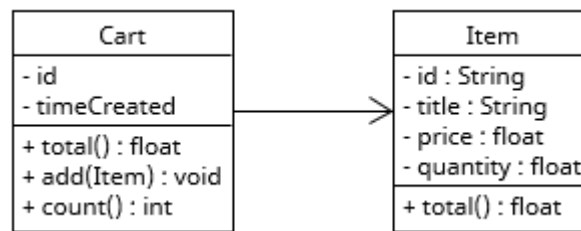
    public Vehicle(){
        // instantiation and initialization
        datasources = new ArrayList<>();
        datasources.add(new VehicleDbDAO());
        datasources.add(new VehicleFileDAO());
    }

    public void save(){
        Hashtable<String,String> data = new Hashtable<>();
        data.put("registrationNumber",registrationNumber); // set other fields similarly

        for(IVehicleDAO datasource : datasources){
            datasource.save(data);
        }
    }
}

```

Write unit tests (using JUnit) for the following class diagram of a shopping cart system:



Item total provides the gross price (price x quantity) of a single item whereas Cart total provides the gross price of all items in a cart. Items can be added to the cart using add function. Count provides the number of items added to cart at any specific point in time.

### Solution

```
class ItemTest{
```

```
    @Test
    public void testTotal() {
        Item i1 = new Item("i1", "...", 100, 3);
        assertEquals(300, i1.total());

        Item i2 = new Item("i2", "...", 200, 1);
        assertEquals(200, i2.total());

        Item i3 = new Item("i3", "...", 200, 0);
        assertEquals(0, i3.total());
    }
}
```

```
class CartTest{
```

```
    Cart cart;

    public void setup() {
        cart = new Cart();
        Item i1 = new Item("i1", "...", 100, 3);
        cart.add(i1);

        Item i2 = new Item("i2", "...", 200, 1);
        cart.add(i2);

        Item i3 = new Item("i3", "...", 300, 1);
        cart.add(i3);
    }

    @Test
    public void testCount() {
        assertEquals(3, cart.count());
    }

    @Test
    public void testAdd() {
        assertEquals(3, cart.count());
    }
}
```

```

    Item i4 = new Item("i4", "...", 400, 1);
    cart.add(i4);
    assertEquals(4, cart.count());


    Item i5 = new Item("i5", "...", 500, 0);
    cart.add(i5);
    assertEquals(4, cart.count());
}

@Test
public void testTotal(){
    if(cart.count() == 3){
        assertEquals(800, cart.total());
    }

    if(cart.count() == 4){
        assertEquals(1200, cart.total());
    }
}
}

```

## National University of Computer and Emerging Sciences, Lahore Campus

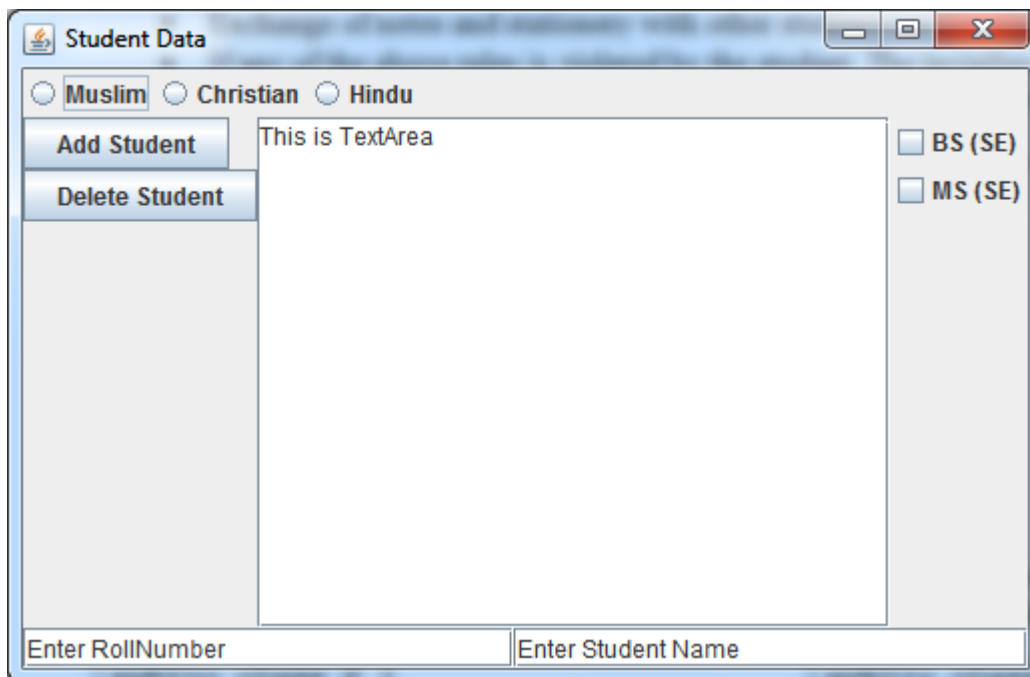
	Course Name:	Software Construction & Development	Section:	ALL
	Program:	BS (Software Engineering)	Semester:	Fall 2022
	Duration:	1 Hour	Total Marks:	30
	Evaluation Type:	Mid 2 Exam	Weight:	15 %
	Course Code:	SE3001	Page(s)	8
	Name:		Roll Number:	

### Important Note:

- The quality of the code will affect the marks.
- Students will receive **ZERO** marks if the answers are plagiarized.
- Use of **mobile phones, internet, and, ANY type of smart devices** during the exam is strictly prohibited.
- Discussion with other students is not allowed.
- Exchange of notes and stationery with other students are not allowed.
- If any of the above rules is violated by the student. The invigilator has the right to file DC case against that student and the invigilator also has the right to take your exam away and ask you to leave the exam hall.

**Question 1:** Write the java code so that it produces the layout shown in the Figure below and implement the window closing event that generates an alert with message “Window is closing!” by using window adapter.

[Marks: 10, CLO 2]

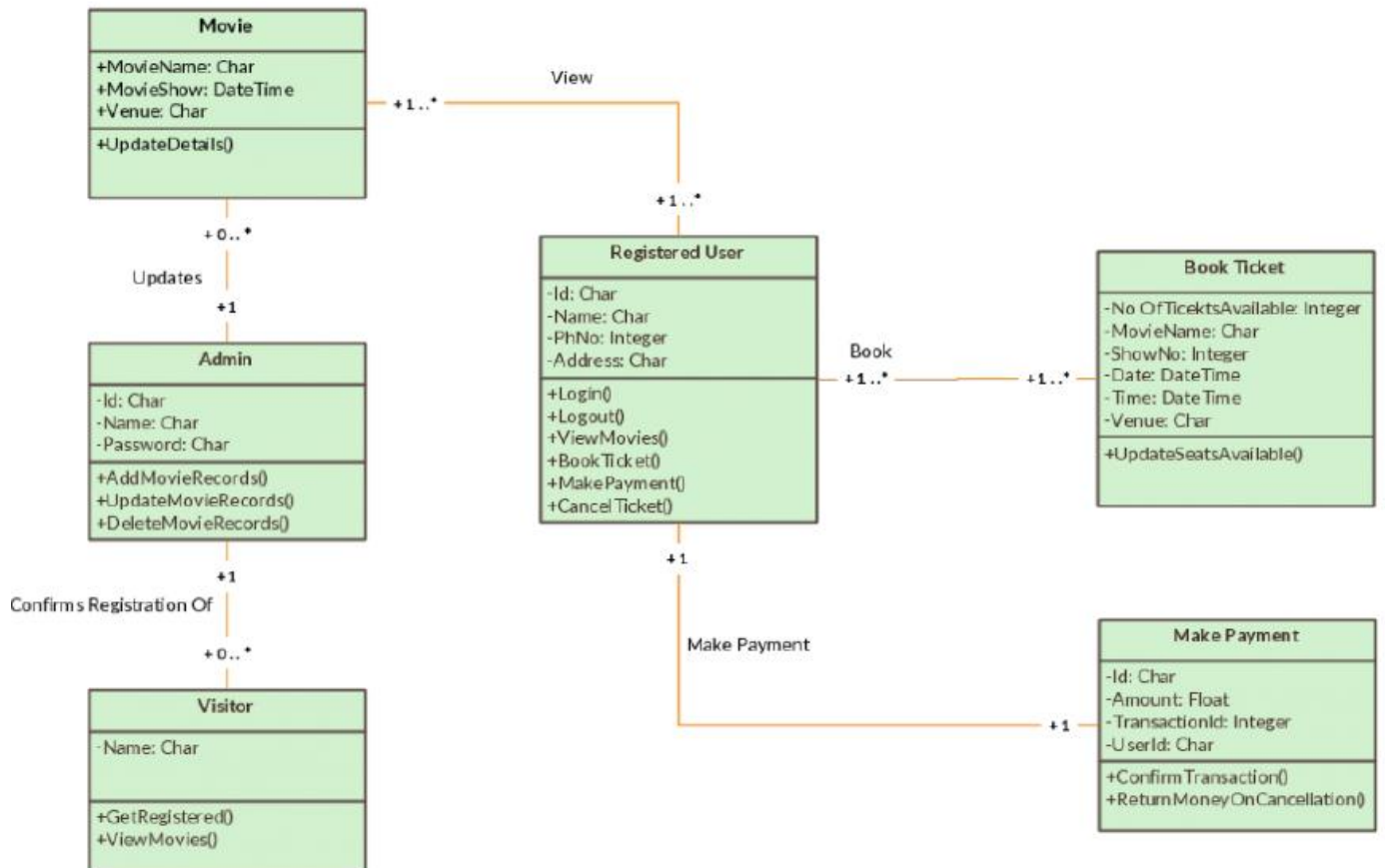






**Question 2: [Marks: 10, CLO 1]**

The following UML class diagram represents the design of “Movie Ticket booking system”. Suggest/propose the GUI(S) and their interconnected flow only for the said system (Note: java code is not required).



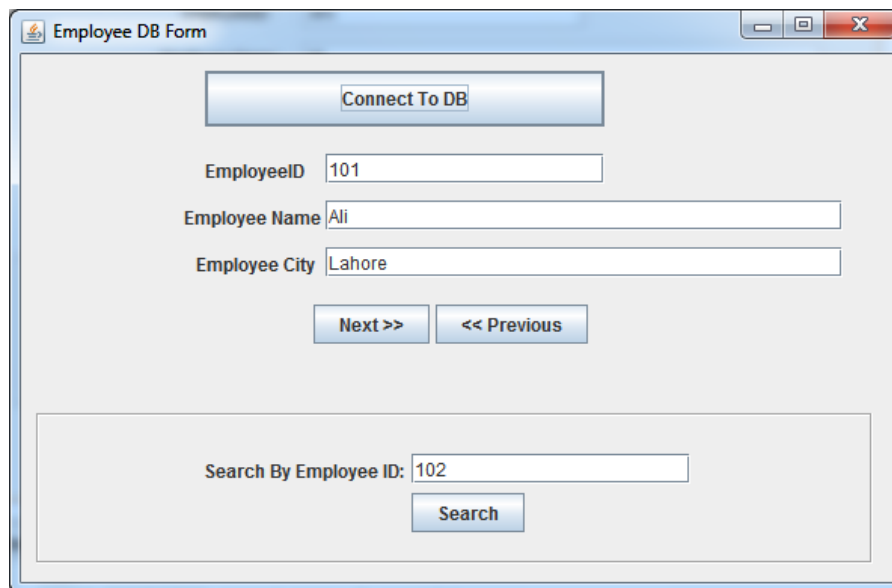


**Question 3: [Marks: 10, CLO 2]**

- Assume there is only one table that is 'Employee' in the underlying Database (consider either SQL Server or MySQL) having four employee records.
- Assume that below GUI (Employee DB Form) is already developed and all instance variables of given 'Employee\_DB\_Form' class are available to you.

You need to write the java code for the following four operations only:

- 1) By Clicking on the 'Connect To DB button', load all employee data from underlying DB and initially populate/fill the EmployeeID, EmployeeName, EmployeeCity Text fields with first row of the underlying 'Employee' Table.
- 2) By clicking on the "Next >>" button: DB cursor should scroll to the next row of the Employee table and populate/fill the current row data to the corresponding EmployeeID, EmployeeName, EmployeeCity Text fields.
- 3) Similarly, By clicking on the "<< Previous" button: DB cursor should scroll to the previous row and populate/fill the current row data to the corresponding EmployeeID, EmployeeName, EmployeeCity Text fields.
- 4) By Clicking on the 'Search' button, it should search the given EmployeeID from the underlying Employee table, if the employee record is not found: an alert should be generated with message of "Employee not found", otherwise (if found), jump the cursor to the corresponding given Employee ID row and populate/fill the already founded row data to their corresponding text fields.



Employee		
EmployeeID	EmployeeName	EmployeeCity
101	Ali	Lahore
102	Kashif	Karachi
103	Babar	Islamabad
104	Haris	Peshawar
*		

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

public class Employee_DB_Form extends JFrame{
    ResultSet rs=null;
    JTextField txt_empID=new JTextField();
    JTextField txt_empName=new JTextField();
    JTextField txt_empCity=new JTextField();

    JTextField txt_search=new JTextField();

    public Employee_DB_Form() {
        initComponents();
    }
    .....
    .....
    private void btn_connectDB_ActionPerformed(java.awt.event.ActionEvent evt) {
        // Write operation 1 code here
```

}

**private void btn\_next\_ActionPerformed(java.awt.event.ActionEvent evt) {**

**// Write operation 2 code here**

}

```
private void btn_previous_ActionPerformed(java.awt.event.ActionEvent evt) {
```

```
    // Write operation 3 code here
```

```
}
```

```
private void btn_search_ActionPerformed(java.awt.event.ActionEvent evt) {
```

```
    // Write operation 4 code here
```

```
}
```

```
}
```