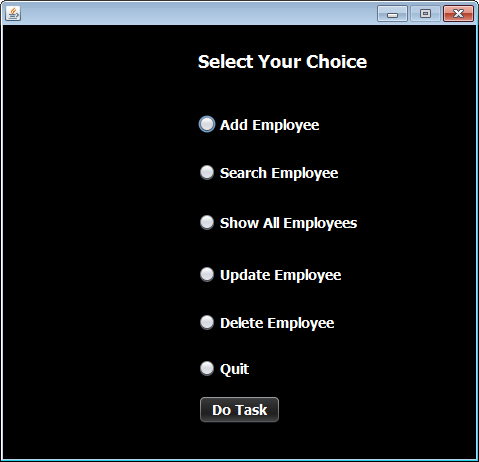
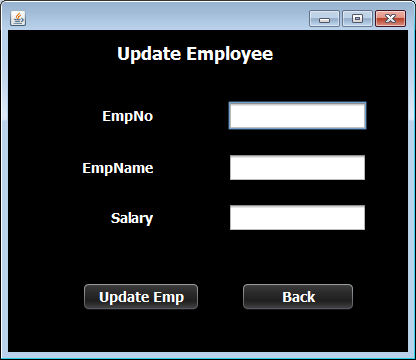
**TASK 1:**

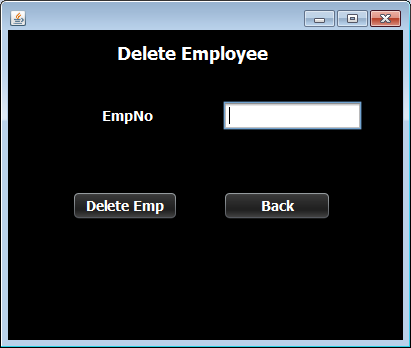
1. Modify the **OptionsFrame** UI to look as shown below.



2. Design the **UpdateEmployeeFrame** UI to look as shown below.



3. Design the **DeleteEmployeeFrame** UI to look as shown below.

****

**TASK 2:**

**WRITING THE CODE FOR BUTTON Button.png IN UpdateEmployeeFrame**

1. When the user will click the **Update Emp** button then it will update the record of employee in the **Employee** table.

2. Following are it's important points:

**a. It will first validate whether all the data has been properly filled all or not.**

**b. If not , then it will generate an error message and return.**

**c. Otherwise , it will create an Employee pojo object , fill all the values in it and pass it to the method updateEmployee( ) of the EmployeeDAO**

**d. If the method updateEmployee( ) returned true then it will display the message "Record updated successfully" , otherwise it will display an error message.**

**e. It will also handle NumberFormatException as well as any SQLException that will be thrown by the method updateEmployee( )**

**TASK 3:**

**WRITING THE CODE FOR BUTTON Button.png IN DeleteEmployeeFrame**

1. When the user will click the **Delete Emp** button then it will update the record of employee in the **Employee** table.

2. Following are it's important points:

**a. It will first validate whether employee number has been given or not.**

**b. If not , then it will generate an error message and return.**

**c. Otherwise , it will call the method deleteEmployee( ) method of EmployeeDAO passing it the Employee Number as argument**

**d. If the method deleteEmployee( ) returned true then it will display the message "Record deleted successfully" , otherwise it will display the message "record not found".**

**e. It will also handle NumberFormatException as well as any SQLException that will be thrown by the method deleteEmployee( )**