Data Structures Project

ABDULLOH ALANSARI

Group Members:

Mohammed Hadad 2240328

Abdullah Alansari 2240003

Source codes

```
First source code:
public class ProjectM {
 public static void main(String [] args)
 {
    Employee_record x=new
Employee_record();
 }
}
Second source code:
public class Node{
    String Id;
   String data;
  Node next;
  public Node(String id,String data)
  {
```

```
this.data=data;
    this.Id=id;
  }
}
Third source code:
import java.awt.BorderLayout;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import javax.swing.*;
public class Employee record extends JFrame
  Node head = null;
```

```
int counter = 0;
  ArrayList<String> hold_Id = new
ArrayList<String>(100);
  public Employee_record() {
    super.setTitle("Employee record
management system");
super.setDefaultCloseOperation(JFrame.EXIT_
ON CLOSE);
    bild();
    super.setVisible(true);
    super.pack();
    super.setLocationRelativeTo(null);
  }
  public void bild() {//important Note: TF is
shortcut for TextField and A shortcut for Add
```

and U For Update And D for Delete And S for search

```
//add employees records
    JLabel ID1 = new JLabel("ID");
    JTextField TFA1 = new JTextField(10);
    JLabel firstName = new JLabel("First
Name");
    JTextField TFA2 = new JTextField(10);
    JLabel LastName = new JLabel("Last
Name");
    JTextField TFA2n = new JTextField(10);
    JLabel First_Day_Work1 = new
JLabel("First Day Work");
    JTextField TFA3 = new JTextField(10);
    JLabel Phone1 = new JLabel("Phone");
    JTextField TFA4 = new JTextField(10);
    JLabel Address1 = new JLabel("Address");
    JTextField TFA5 = new JTextField(10);
```

```
JLabel Work_Hours1 = new JLabel("Work
hours");
    JTextField TFA6 = new JTextField(10);
    JLabel salary1 = new JLabel("salary");
    JTextField TFA7 = new JTextField(10);
    //update employees records
    JLabel ID2 = new JLabel("ID");
    JTextField TFU1 = new JTextField(10);
    //update
    JLabel USalary = new JLabel("ID");
    JTextField TFUS = new JTextField(10);
    //Search fro employees
    JLabel Text1 = new JLabel("Enter Employ
ID to Search or to Delete");
    JTextField TFS = new JTextField(10);
```

```
//deleate employees
    //Buttons
    JButton Insert = new JButton("Insert");
    JButton Delete = new JButton("Delete");
    JButton Update = new JButton("Update");
    JButton disblay = new JButton("disblay");
    JButton Search = new JButton("Search");
    JButton Salary = new JButton("Update
Salary");
    // TextArea
    JTextArea area = new JTextArea(30, 40);
    area.setWrapStyleWord(true);
    JScrollPane s = new JScrollPane(area);
    area.setEditable(false);
    area.setLineWrap(true);
    //Panels
    JPanel panel1 = new JPanel();
```

```
JPanel panel2 = new JPanel();
JPanel panel2n = new JPanel();
JPanel panel3 = new JPanel();
JPanel panel4 = new JPanel();
JPanel panel5 = new JPanel();
JPanel panel6 = new JPanel();
JPanel panel7 = new JPanel();
JPanel panel8 = new JPanel();
JPanel panel9 = new JPanel();
JPanel panel10 = new JPanel();
JPanel panel11 = new JPanel();
JPanel panel12 = new JPanel();
JPanel panel13 = new JPanel();
JPanel panel14 = new JPanel();
JPanel panel15 = new JPanel();
JPanel panel16 = new JPanel();
JPanel panel17 = new JPanel();
JPanel panel18 = new JPanel();
JPanel All = new JPanel();
```

```
JPanel Button Panel = new JPanel();
    //Empty panel for Design
    JPanel empty1 = new JPanel();
    JPanel empty2 = new JPanel();
    JPanel empty3 = new JPanel();
    JPanel empty4 = new JPanel();
    //set panel Layout
    panel1.setLayout(new GridLayout(2, 1));
    panel2.setLayout(new GridLayout(2, 1));
    panel2n.setLayout(new GridLayout(2, 1));
    panel3.setLayout(new GridLayout(2, 1));
    panel4.setLayout(new GridLayout(2, 1));
    panel5.setLayout(new GridLayout(2, 1));
    panel6.setLayout(new GridLayout(2, 1));
    panel7.setLayout(new GridLayout(2, 1));
    panel8.setLayout(new GridLayout(1, 8,
20, 20));//contain all the panel from 1-7
```

```
panel9.setLayout(new GridLayout(2, 2));
panel10.setLayout(new GridLayout(2, 1));
panel11.setLayout(new GridLayout(2, 1));
panel12.setLayout(new GridLayout(2, 1));
panel13.setLayout(new GridLayout(2, 1));
panel14.setLayout(new GridLayout(2, 1));
panel15.setLayout(new GridLayout(2, 1));
panel16.setLayout(new GridLayout(2, 1));
panel17.setLayout(new GridLayout(1, 7, 20, 20));//contain all the panel from 10-16
panel18.setLayout(new GridLayout(2, 1));
```

```
All.setLayout(new GridLayout(5, 1)); //containg panel 8 and 9 and 17
```

panel8.setBorder(BorderFactory.createTitledB
order("Add Employees Records"));

panel9.setBorder(BorderFactory.createTitledB

```
order("Serach for Employees Records or
delete it"));
panel17.setBorder(BorderFactory.createTitled
Border("Update Employees Records"));
panel18.setBorder(BorderFactory.createTitled
Border("Update Employees Salary"));
    //add labels and TextFildes to panel
    panel1.add(ID1);
    panel1.add(TFA1);
    panel2.add(firstName);
    panel2.add(TFA2);
    panel2n.add(LastName);
    panel2n.add(TFA2n);
    panel3.add(First_Day_Work1);
```

```
panel3.add(TFA3);
panel4.add(Phone1);
panel4.add(TFA4);
panel5.add(Address1);
panel5.add(TFA5);
panel6.add(Work_Hours1);
panel6.add(TFA6);
panel7.add(salary1);
panel7.add(TFA7);
panel8.add(panel1);
panel8.add(panel2);
panel8.add(panel2n);
panel8.add(panel3);
panel8.add(panel4);
```

```
panel8.add(panel5);
panel8.add(panel6);
panel8.add(panel7);
panel9.add(Text1);
panel9.add(empty1);
panel9.add(TFS);
panel9.add(empty2);
panel10.add(ID2);
panel10.add(TFU1);
panel17.add(panel10);
panel17.add(panel11);
panel17.add(panel12);
panel17.add(panel13);
panel17.add(panel14);
panel17.add(panel15);
panel17.add(panel16);
```

```
panel18.add(USalary);
    panel18.add(empty3);
    panel18.add(TFUS);
    panel18.add(empty4);
    All.add(panel8);
    All.add(panel9);
    All.add(panel17);
    All.add(panel18);
    Insert.addActionListener(new
ActionListener() {
      @Override
      public void
actionPerformed(ActionEvent e) {
        try {
```

```
String hold_id = TFA1.getText();
```

```
Node newNode = new
Node(hold_id, "-" + TFA2.getText() + "-" +
TFA2n.getText() + TFA3.getText() + "-" +
TFA4.getText() + "-" + TFA5.getText() + "-" +
TFA6.getText() + "-" + TFA7.getText());
           newNode.next = null;
           Node temp = head;
           if (check(hold id)) {
             if (head == null ) {
               newNode.next = head;
               head = newNode;
JOptionPane.showMessageDialog(null,
"Reccord Successfuly created");
             } else {
```

```
++counter;
               while (temp.next != null) {
                 temp = temp.next;
               }
               temp.next = newNode;
JOptionPane.showMessageDialog(null,
"Reccord Successfuly created");
           } else {
JOptionPane.showMessageDialog(null, "the ID
are Alrady taken");
        } catch (Exception x) {
JOptionPane.showMessageDialog(null,
x.getMessage());
```

```
}
    });
    disblay.addActionListener(new
ActionListener() {
       @Override
      public void
actionPerformed(ActionEvent e) {
         Node temp = head;
         String show = "";
         try {
           ArrayList<String> x = new
ArrayList<String>();
           while (temp != null) {
             x.add(temp.ld + " " + temp.data
+ "\n");
             temp = temp.next;
           }
```

```
int i = x.size() - 1;
           while (i > -1) {
              show += x.get(i);
             --i;
           }
           area.setText(show);
         } catch (Exception x) {
JOptionPane.showMessageDialog(null,
x.getMessage());
    });
    Delete.addActionListener(new
ActionListener() {
       @Override
```

```
public void
actionPerformed(ActionEvent e) {
        try {
           String idToDelete = TFS.getText();
           boolean deleted =
deleteEmployee(idToDelete);
           if (deleted) {
JOptionPane.showMessageDialog(null,
"Record deleted successfully");
           } else {
JOptionPane.showMessageDialog(null,
"Record not found");
         } catch (Exception ex) {
JOptionPane.showMessageDialog(null, "Error:
" + ex.getMessage());
         }
```

```
}
       private boolean deleteEmployee(String
id) {
         try {
           hold_Id.remove(id);
           if (head == null) {
              return false;
           }
           if (head.Id.equals(id)) {
              head = head.next;
              return true;
           }
            Node prev = head;
            Node current = head.next;
           while (current != null) {
              if (current.Id.equals(id)) {
```

```
prev.next = current.next;
                return true;
             prev = current;
             current = current.next;
           }
           return false;
         } catch (Exception ex) {
           throw new
RuntimeException("Error deleting employee: "
+ ex.getMessage());
         }
    });
    Search.addActionListener(new
ActionListener() {
       @Override
```

```
public void
actionPerformed(ActionEvent e) {
         try {
           String idToSearch = TFS.getText();
           Node result =
searchEmployee(idToSearch);
           if (result != null) {
             displayRecord(result);
           } else {
JOptionPane.showMessageDialog(null,
"Record not found");
         } catch (Exception ex) {
JOptionPane.showMessageDialog(null, "Error:
" + ex.getMessage());
```

```
private Node searchEmployee(String id)
{
         try {
           Node current = head;
           while (current != null) {
             if (current.Id.equals(id)) {
                return current;
             }
             current = current.next;
           }
           return null;
         } catch (Exception ex) {
           throw new
RuntimeException("Error searching employee:
" + ex.getMessage());
      }
      private void displayRecord(Node
record) {
```

```
try {
           StringBuilder message = new
StringBuilder();
           message.append("ID:
").append(record.Id).append("\n");
           message.append("data:
").append(record.data).append("\n");
JOptionPane.showMessageDialog(null,
message.toString());
        } catch (Exception ex) {
           throw new
RuntimeException("Error displaying record: "
+ ex.getMessage());
    });
```

```
Update.addActionListener(new
ActionListener() {
       @Override
       public void
actionPerformed(ActionEvent e) {
         try {
           String idToUpdate =
TFU1.getText();
           Node result =
searchEmployee(idToUpdate);
               for(int i=0;i<hold_Id.size();++i)</pre>
           {
if(idToUpdate.equalsIgnoreCase(hold Id.get(i)
))
             {
                hold_Id.remove(i);
              }
           }
           if (head.Id == null) {
```

```
JOptionPane.showMessageDialog(null, "id is
null Reccord cant be done ");
           if (result != null) {
             updateRecord(result);
           }
         } catch (Exception ex) {
JOptionPane.showMessageDialog(null, "Error
Updateing record: " + ex.getMessage());
      }
      private Node searchEmployee(String id)
         try {
           Node current = head;
```

```
while (current != null) {
             if (current.Id.equals(id)) {
                return current;
             }
             current = current.next;
         } catch (Exception ex) {
JOptionPane.showMessageDialog(null, "Error
searching employee: " + ex.getMessage());
         return null;
      }
      private void updateRecord(Node
record) {
         try {
```

```
String Id =
JOptionPane.showInputDialog("Enter new
Id:");
           if(check(Id)){
           String data =
JOptionPane.showInputDialog("Enter new
data:Enter the First Name,Enter The second
Name, first day work, phone, address, work
hours ,salary (note:you need to put - after
each one)");
           record.Id = Id;
           record.data = data;
JOptionPane.showMessageDialog(null,
"Record updated successfully");
           }else
JOptionPane.showMessageDialog(null, "the ID
```

are Alrady taken");

```
}
         } catch (Exception ex) {
JOptionPane.showMessageDialog(null,"Error
updating record: ");
    });
    Salary.addActionListener(new
ActionListener() {
       @Override
      public void
actionPerformed(ActionEvent e) {
         try {
           String v = TFUS.getText();
           String hold_info = "";
           String ID = "";
           Node result = searchEmployee(v);
```

```
String hold = result.Id + "-" +
result.data;
            String[] x = hold.split("-");
            String First_Name = "";
            String last Name = "";
            String First Day Work = "";
            String Phone1 = "";
            String Address = "";
            String containSalary = "";
            String containhours = "";
            int i = 0;
            while (i < x.length) {
              ID = x[i];
              ++i;
              First_Name = x[i];
```

```
hold_info += First_Name + "-";
             ++i;
             last Name = x[i];
              hold_info += last_Name + "-";
             ++i;
             First_Day_Work = x[i];
              hold_info += First_Day_Work + "-
н.
             ++i;
             Phone1 = x[i];
              hold info += Phone1 + "-";
             ++i;
             Address = x[i];
              hold info += Address + "-";
             ++i;
             containhours = x[i];
              hold_info += containhours;
             ++i;
             containSalary = x[i];
```

```
}
           double holdslary =
Double.parseDouble(containSalary);
           int holdhours =
Integer.parseInt(containhours);
           if (holdhours > 32) {
             int extra_hours = holdhours - 32;
             double newprcent = extra hours
* 2;
             double extraSalary = (newprcent
/ 100.0) * holdslary;
             double newSalary = holdslary +
extraSalary;
             Node new Info = new Node(ID,
hold_info + "-" + newSalary);
```

++i;

```
updateRecord(result, hold_info +
"-" + newSalary, ID);
           } else {
             double newSalary = holdslary;
             updateRecord(result, hold info +
"-" + newSalary, ID);
JOptionPane.showMessageDialog(null,
"Unfortunately you didn't get an increase in
your salary as your hours of work are 32 or
less.");
           }
         } catch (Exception x) {
JOptionPane.showMessageDialog(null, "Error
updating salary: " + x.getMessage());
         }
```

```
}
       private Node searchEmployee(String id)
{
         try {
           Node current = head;
           while (current != null) {
              if (current.Id.equals(id)) {
                return current;
              }
              current = current.next;
           }
           return null;
         } catch (Exception ex) {
JOptionPane.showMessageDialog(null, "Error
```

searching for employee: " + ex.getMessage());

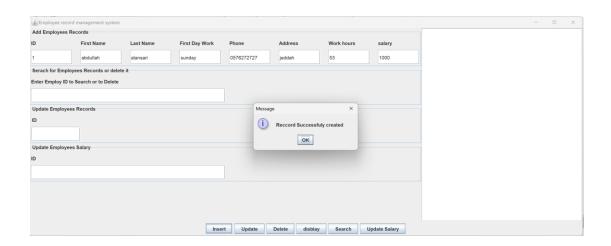
```
}
         return null;
      }
       private void updateRecord(Node
record, String info, String id) {
         try {
           Node temp = head;
           while (temp != null) {
             if (temp.ld.equalsIgnoreCase(id))
                temp.data = info;
             }
             temp = temp.next;
           }
```

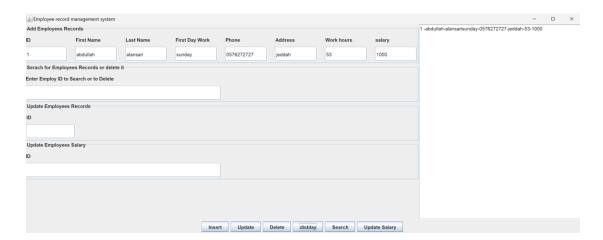
```
JOptionPane.showMessageDialog(null,
"Record updated successfully");
        } catch (Exception ex) {
JOptionPane.showMessageDialog(null, "Error
updating record: " + ex.getMessage());
    });
    //add Buttons
    Button_Panel.add(Insert);
    Button Panel.add(Update);
    Button Panel.add(Delete);
    Button_Panel.add(disblay);
    Button Panel.add(Search);
    Button_Panel.add(Salary);
```

```
super.add(All, BorderLayout.WEST);
    super.add(s, BorderLayout.EAST);
    super.add(Button_Panel,
BorderLayout.SOUTH);
  }
  public boolean check(String id) {
    try {
      for (int i = 0; i < hold_ld.size(); ++i) {
         if
(hold_Id.get(i).equalsIgnoreCase(id)) {
           return false;
      }
      hold_Id.add(id);
      return true;
```

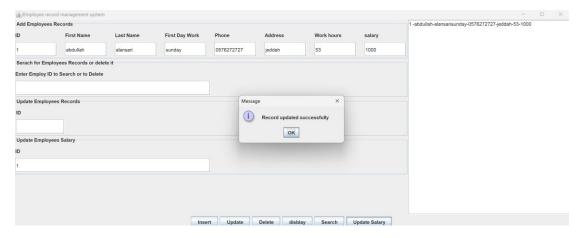
```
} catch (Exception ex) {
    JOptionPane.showMessageDialog(null,
"Error occurred in check method: " +
    ex.getMessage());
    return false;
    }
}
```

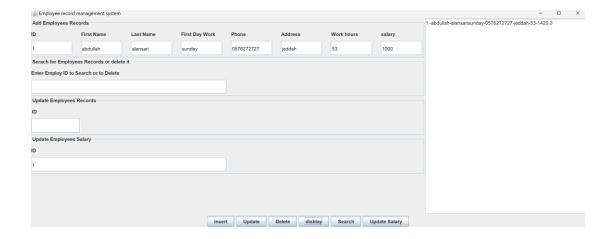
Photos of some of the program's outputs



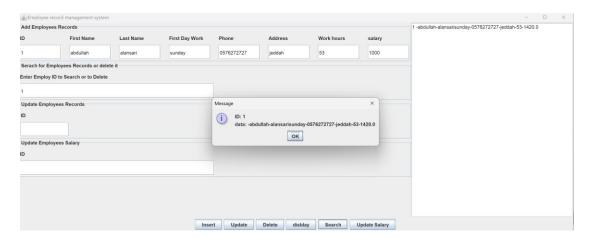


After update salary

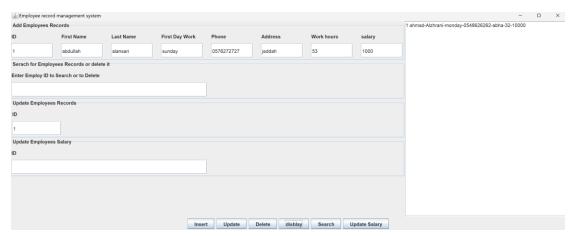




Search



Update record



Delete Record

