|  |  |
| --- | --- |
| File:COMSATS new logo.jpg - Wikimedia Commons  OBJECT ORIENTED PROGRAMMING  *Class Assignment 4*  *Student Management System*  *+*  *Lab Task 4* | **submitted by:**  **Shahzaneer Ahmed**  **registration number:**  **sp21-bcs-087**  **submitted to:**  **mA’M sANEEHA aMIR**  **date of submission:**  **june 13, 2022** |

Student Management System

Add Screen

import java.awt.\*;

import javax.swing.\*;

import java.awt.event.\*;

public class AddScreen extends JFrame {

*// Data Members*

    private JLabel nameLabel, phoneNumberLabel, genderLabel, gpaJLabel, semesterLabel, sectionLabel, departmentNameLabel,

            departmentlocationLabel;

    private JTextField nameTextField, phoneNumberTextField,genderTextField, gpaTextField, semesterTextField, sectionTextField,

            departmentNameTextField, departmentLocationTextField;

    private JButton submit, home;

*// Constructor*

    public AddScreen() {

*this*.setSize(600, 600);

*this*.setForeground(Color.BLACK);

*this*.setVisible(true);

*this*.setLayout(new GridLayout(9, 2));

        nameLabel = new JLabel("Name");

        phoneNumberLabel = new JLabel("Phone Number");

        genderLabel = new JLabel("Gender");

        semesterLabel = new JLabel("Semtester");

        gpaJLabel = new JLabel("GPA");

        sectionLabel = new JLabel("Section");

        departmentNameLabel = new JLabel("Department Name");

        departmentlocationLabel = new JLabel("Department Location");

        nameTextField = new JTextField();

        phoneNumberTextField = new JTextField();

        genderTextField = new JTextField();

        gpaTextField = new JTextField();

        semesterTextField = new JTextField();

        sectionTextField = new JTextField();

        departmentNameTextField = new JTextField();

        departmentLocationTextField = new JTextField();

        submit = new JButton("Submit");

        home = new JButton("Home");

*// Addition of these Components*

*this*.add(nameLabel);

*this*.add(nameTextField);

*this*.add(phoneNumberLabel);

*this*.add(phoneNumberTextField);

*this*.add(genderLabel);

*this*.add(genderTextField);

*this*.add(semesterLabel);

*this*.add(semesterTextField);

*this*.add(gpaJLabel);

*this*.add(gpaTextField);

*this*.add(sectionLabel);

*this*.add(sectionTextField);

*this*.add(departmentNameLabel);

*this*.add(departmentNameTextField);

*this*.add(departmentlocationLabel);

*this*.add(departmentLocationTextField);

*this*.add(submit);

*this*.add(home);

        myActionListener a = new myActionListener();

        submit.addActionListener(a);

        home.addActionListener(a);

    }

*// Inner Class*

    public class myActionListener implements ActionListener {

        @Override

        public void actionPerformed(ActionEvent e) {

            if (e.getSource() == home) {

                dispose();

                new HomeScreen();

            }

            if (e.getSource() == submit) {

                String name = nameTextField.getText();

                String phoneNumber = phoneNumberTextField.getText();

                String gender = genderTextField.getText();

                int semester = Integer.parseInt(semesterTextField.getText());

                double gpa = Double.parseDouble(gpaTextField.getText());

                String section = sectionTextField.getText();

                String departmentName = departmentNameTextField.getText();

                String departmentLocation = departmentLocationTextField.getText();

                if (name.equals(null) || phoneNumber.equals(null) || gender.equals(null) || semester == 0 || gpa == 0 || section.equals(null)

                        || departmentName.equals(null) || departmentLocation.equals(null) || nameTextField.equals(null) || phoneNumberTextField.equals(null) || genderTextField.equals(null) || semesterTextField.equals(null) || gpaTextField.equals(null) || sectionTextField.equals(null) || departmentNameTextField.equals(null) || departmentLocationTextField.equals(null)) {

                    JOptionPane.showMessageDialog(null, "Please Enter All Fields to add your Details");

                }

                else {

                Department d = new Department(departmentName, departmentLocation);

                Student s = new Student(name, phoneNumber, gender, gpa, semester, section, d);

                OperationsStorage o = new OperationsStorage();

                o.writeToFile(s);

                }

            }

        }

    }

}

Delete Screen

import java.awt.\*;

import javax.swing.\*;

import java.awt.event.\*;

public class DeleteScreen extends JFrame {

    // Data Members

    private JLabel lblDelete;

    private JTextField nameTobeDeleted;

    private JButton delete;

    private JButton home;

    // Constructor

    public DeleteScreen() {

        // System.out.println("Delete Screen constructor ");

*this*.setSize(600, 600);

*this*.setForeground(Color.BLACK);

*this*.setVisible(true);

*this*.setLayout(new GridLayout(2, 2));

        lblDelete = new JLabel("Enter the Name of Student you want to Delete");

        nameTobeDeleted = new JTextField(20);

        delete = new JButton("Delete");

        home = new JButton("Home");

*this*.add(lblDelete);

*this*.add(nameTobeDeleted);

*this*.add(delete);

*this*.add(home);

*// adding Action Listener to the buttons*

        myActionListener a = new myActionListener();

        delete.addActionListener(a);

        home.addActionListener(a);

    }

    class myActionListener implements ActionListener {

        @Override

        public void actionPerformed(ActionEvent e) {

            if (e.getSource() == delete) {

                OperationsStorage o = new OperationsStorage();

                String name = nameTobeDeleted.getText();

                boolean found = o.removeStudent(name);

                if(found)

                    JOptionPane.showMessageDialog(null, "Student Deleted !");

                else

                    JOptionPane.showMessageDialog(null, "Student not found !");

            }

            else if (e.getSource() == home) {

                dispose();

                new HomeScreen();

            }

        }

    }

    }

Department

import java.io.Serializable;

public class Department implements Serializable {

    private String name;

    private String location;

    public Department() {

    }

    public Department(String name, String location) {

*this*.name = name;

*this*.location = location;

    }

    public String getName() {

        return *this*.name;

    }

    public void setName(String name) {

*this*.name = name;

    }

    public String getLocation() {

        return *this*.location;

    }

    public void setLocation(String location) {

*this*.location = location;

    }

    @Override

    public String toString() {

        return

            " Department Name :" + getName() + "\n" +

            ", location='" + getLocation() + "\n";

    }

}

Home Screen

// Wild Card import for all

import java.awt.\*;

import javax.swing.\*;

import java.awt.event.\*;

public class HomeScreen extends JFrame{

*// Data Members*

    JButton add , searchByName , searchByDepartment , update , delete , readAll;

*// Constructor*

    public HomeScreen() {

*this*.setSize(600, 600);

*this*.setForeground(Color.BLACK);

*this*.setVisible(true);

*this*.setLayout(new GridLayout(6, 1));

*// Create the buttons*

        add = new JButton("Add");

        searchByName = new JButton("Search By Name");

        searchByDepartment = new JButton("Search By Department");

        update = new JButton("Update");

        delete = new JButton("Delete");

        readAll = new JButton("Read All");

        // Add the buttons to the frame

        this.add(add);

        this.add(readAll);

        this.add(searchByName);

        this.add(searchByDepartment);

        this.add(update);

        this.add(delete);

        // adding Action Listener to the buttons

        myActionListener a = new myActionListener();

        add.addActionListener(a);

        searchByName.addActionListener(a);

        searchByDepartment.addActionListener(a);

        update.addActionListener(a);

        delete.addActionListener(a);

        readAll.addActionListener(a);

    }

    // Inner Class

    public class myActionListener implements ActionListener {

        @Override

        public void actionPerformed(ActionEvent e) {

            if (e.getSource() == add) {

                dispose();

                new AddScreen();

            }

            else if (e.getSource() == update) {

                dispose();

                new UpdateScreen();

            }

            else if (e.getSource() == searchByName) {

                new SearchByNameScreen();

            }

            else if (e.getSource() == searchByDepartment) {

                new SearchByDepartmentScreen();

            }

            else if (e.getSource() == delete) {

                dispose();

                new DeleteScreen();

            }

            else if(e.getSource() == readAll) {

                OperationsStorage o = new OperationsStorage();

                String details = o.readAll();

                JOptionPane.showMessageDialog(null,details);

            }

            }

        }

    }

MyObjectOutputStream

import java.io.IOException;

import java.io.ObjectOutputStream;

import java.io.OutputStream;

*//! Object writing class*

public class MyObjectOutputStream extends ObjectOutputStream{

*// Our custom Object Output Stream class -> which is used to input object data*

*// without writing the Stream headers.*

    public MyObjectOutputStream() throws IOException{

*super*();

    }

    public MyObjectOutputStream(OutputStream o) throws IOException{

*super*(o);

    }

    public void writeStreamHeader(){}

}

OperationsStorage

import java.io.EOFException;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.io.OutputStream;

import java.util.ArrayList;

import java.util.Arrays;

import java.awt.\*;

import javax.swing.\*;

import java.awt.event.\*;

public class OperationsStorage extends JFrame {

    //\* for update and delete operation we need to safe the objects in an arraylist first and then manipulate them  (Update or delete) and then write them into the

    //\* the file again therefore the following array list is created to store the objects

    // \* And to reduce the complexity of code we have added the object to list as soon as they are written in the file so that we can manipuate them afterwards

    private ArrayList<Student> a = new ArrayList<>();

    private File f = new File("Students.ser");

    //! Create for GUI

    public void writeToFile(Student s) {

        // file object

        // File f = new File("Students.ser");

        // Object for writing class (ObjectOutputStream)

        // Why this object is NULL?

        // Answer : Because when even the constructor is called, the object (File)

        // is created but we want to keep it in the try block

        ObjectOutputStream oos = null;

        // write to file

        try {

            if (!f.exists()) {

                f = new File("Students.ser");

            }

            if (f.exists()) {

                oos = new MyObjectOutputStream(new FileOutputStream(f, true));

                oos.writeObject(s); // write object to file

            } else {

                oos = new ObjectOutputStream(new FileOutputStream(f, true));

                oos.writeObject(s); // it will write the object to the file.

            }

        } catch (IOException e) {

            e.printStackTrace();

        }

        catch (Exception e) {

            System.err.println("Cannot Write Object");

        }

        // For closing File

        if (oos != null) {

            try {

                oos.close();

            } catch (IOException e) {

                e.printStackTrace();

            }

        }

    }

    //! Read for GUI

    public String readAll() {

        ObjectInputStream oo = null;

        StringBuilder details = new StringBuilder();

        if (!f.exists())

        return "File Not Found No Record!";

    if (f.exists()) {

        try {

            oo = new ObjectInputStream(new FileInputStream("Students.ser"));

            while (true) {

                // Reading object is below

                Student s = (Student) oo.readObject();

                details.append(s.toString());

            }

        } catch (ClassNotFoundException e) {

            e.printStackTrace();

        }

        catch (EOFException e) {

        }

        catch (FileNotFoundException e) {

            e.printStackTrace();

        }

        catch (IOException e) {

            e.printStackTrace();

        }

        finally {

            try {

                oo.close();

            } catch (IOException e) {

            }

        }

        return details.toString();

    }

    return "File Not Found No Record!";

    }

    //! Search for GUI

    public String searchStudentByName(String Name) {

        StringBuilder details = new StringBuilder();

        boolean foundSome = false;

        ObjectInputStream oo = null;

        try {

            oo = new ObjectInputStream(new FileInputStream("Students.ser"));

            while (true) {

                // Reading object is below

                Student s = (Student) oo.readObject();

                if (s.getName().equalsIgnoreCase(Name)) {

                    details.append(s.toString());

                    foundSome = true;

                }

            }

        } catch (ClassNotFoundException e) {

            e.printStackTrace();

        }

        catch (EOFException e) {

        }

        catch (FileNotFoundException e) {

            e.printStackTrace();

        }

        catch (IOException e) {

            e.printStackTrace();

        }

        finally {

            try {

                oo.close();

            } catch (IOException e) {

            }

        }

        if (!foundSome) {

            return "No Student Found !";

        }

        return details.toString();

    }

    public String searchStudentsByDepartment(String department) {

        StringBuilder details = new StringBuilder();

        ObjectInputStream oo = null;

        boolean foundSome = false;

        try {

            oo = new ObjectInputStream(new FileInputStream("Students.ser"));

            while (true) {

                // Reading object is below

                Student s = (Student) oo.readObject();

                if (s.getDepartment().getName().equalsIgnoreCase(department)) {

                    details.append(s.toString());

                    foundSome = true;

                }

            }

        } catch (ClassNotFoundException e) {

            e.printStackTrace();

        }

        catch (EOFException e) {

        }

        catch (FileNotFoundException e) {

            e.printStackTrace();

        }

        catch (IOException e) {

            e.printStackTrace();

        }

        finally {

            try {

                oo.close();

            } catch (IOException e) {

            }

        }

        if (!foundSome) {

            return "No Such Department Found !";

        }

        return details.toString();

    }

    //! UPDATE for GUI

    boolean updateGPA(String s, double newCGPA) {

        boolean found = false;

        // a.clear(); //clearing the list before adding objects

        ObjectInputStream oo = null;

        try {

            oo = new ObjectInputStream(new FileInputStream("Students.ser"));

            try {

                while (true) {

                    Student k = (Student) oo.readObject();

                    a.add(k);

                }

            } catch (EOFException e) {

            }

        } catch (FileNotFoundException e) {

            e.printStackTrace();

        } catch (IOException e) {

            e.printStackTrace();

        } catch (ClassNotFoundException e) {

            e.printStackTrace();

        }

        finally {

            try {

                oo.close();

            } catch (IOException e) {

            }

        }

        // firstly manipulating the desired object for updating the CGPA

        for (int i = 0; i < a.size(); i++) {

            if (a.get(i).getName().equalsIgnoreCase(s)) {

                found = true;

                a.get(i).setGPA(newCGPA);

            }

        }

        //\* now again writing the Arraylist Objects to the file. first time it will create the file again and only then it will append!

        // file object

        // f = new File("Students.ser");

        ObjectOutputStream oos = null;

        int counter = 0;

        try {

            for (int i = 0; i < a.size(); i++) {

                // System.out.println("After Manipulation reading loop - size of arraylist"+ i);

                if (counter > 0) {

                    // System.out.println("counter greater then 0");

                    oos = new MyObjectOutputStream(new FileOutputStream(f, true));

                    oos.writeObject(a.get(i));

                }

                else {

                    // System.out.println("counter is 0");

                    oos = new ObjectOutputStream(new FileOutputStream(f));

                    oos.writeObject(a.get(i));

                    counter++;

                }

            }

*// For closing File*

            if (oos != null) {

                oos.close();

            }

        }

        catch (IOException e) {

            e.printStackTrace();

        }

        return found;

    }

*//! DELETE for GUI*

    boolean removeStudent(String toBeDeletedStudent) {

        boolean found = false;

        ObjectInputStream oo = null;

        try {

            oo = new ObjectInputStream(new FileInputStream("Students.ser"));

            try {

                while (true) {

                    Student s = (Student) oo.readObject();

                    a.add(s);

                }

            } catch (EOFException e) {

                // Move to the next line broda

            }

            // now we will move sequentially..

            oo.close();

            // removing the specified object from the arraylist

            for (int i = 0; i < a.size(); i++) {

                if (a.get(i).getName().equalsIgnoreCase(toBeDeletedStudent)) {

                    found = true;

                    a.remove(i);

                }

            }

            // now again writing the Arraylist objects in the file first time we will create a new file and then we will append

            // Object for writing class (ObjectOutputStream)

            ObjectOutputStream oos = null;

            // write to file

            int counter = 0;

            if (a.size() > 0) {

                for (int i = 0; i < a.size(); i++) {

                    System.out.println("Writing again to the file");

                    if (counter > 0) {

                        // when you are running it for the second and afterwards iterations you will append the file

                        oos = new MyObjectOutputStream(new FileOutputStream(f, true));

                        oos.writeObject(a.get(i));

                    } else {

                        // for the first time you will create a new file

                        oos = new ObjectOutputStream(new FileOutputStream(f));

                        oos.writeObject(a.get(i));

                        counter++;

                    }

                }

                // For closing File

                if (oos != null) {

                    oos.close();

                }

            }

            else if (a.size() == 0) {

                System.out.println("File deleting");

                f.delete();

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

            }

        }

        catch (Exception e) {

            e.printStackTrace();

        }

        return found;

    }

}

Person

import java.io.Serializable;

public class Person implements Serializable {

    private String name;

    private String Phone;

    private String gender;

    public Person() {

    }

    public Person(String name, String Phone, String gender) {

        this.name = name;

        this.Phone = Phone;

        this.gender = gender;

    }

    public String getName() {

        return this.name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public String getPhone() {

        return this.Phone;

    }

    public void setPhone(String Phone) {

*this*.Phone = Phone;

    }

    public String getGender() {

        return *this*.gender;

    }

    public void setGender(String gender) {

*this*.gender = gender;

    }

    @Override

    public String toString() {

        return

            "Name: " + getName() + "\n" +

            "Phone='" + getPhone() + "\n" +

            "gender='" + getGender() + "\n";

    }

}

SearchByDepartmentScreen

import java.awt.\*;

import javax.swing.\*;

import java.awt.event.\*;

public class SearchByDepartmentScreen extends JFrame {

    private JLabel searchDepartment;

    private JTextField searchDepartmentTextField;

    private JButton searchDepartmentButton;

    private JButton home;

    public SearchByDepartmentScreen() {

*this*.setSize(600, 600);

*this*.setForeground(Color.BLACK);

*this*.setVisible(true);

*this*.setLayout(new GridLayout(2, 2));

        searchDepartment = new JLabel("Search By Department");

        searchDepartmentTextField = new JTextField();

        searchDepartmentButton = new JButton("Search");

        home = new JButton("Home");

*this*.add(searchDepartment);

*this*.add(searchDepartmentTextField);

*this*.add(searchDepartmentButton);

*this*.add(home);

        myActionListener a = new myActionListener();

        searchDepartmentButton.addActionListener(a);

        home.addActionListener(a);

    }

*// Inner Class*

    public class myActionListener implements ActionListener {

        @Override

        public void actionPerformed(ActionEvent e) {

            if(e.getSource() == searchDepartmentButton) {

                OperationsStorage o = new OperationsStorage();

                String department = searchDepartmentTextField.getText();

                String details = o.searchStudentsByDepartment(department);

                JOptionPane.showMessageDialog(null, details);

            }

            else if(e.getSource() == home) {

                dispose();

            }

        }

    }

}

SearchByNameScreen

import java.awt.\*;

import javax.swing.\*;

import java.awt.event.\*;

public class SearchByNameScreen extends JFrame {

    private JLabel searchName;

    private JTextField searchNameTextField;

    private JButton searchNameButton;

    private JButton home;

    public SearchByNameScreen() {

*this*.setSize(600, 600);

*this*.setForeground(Color.BLACK);

*this*.setVisible(true);

*this*.setLayout(new GridLayout(2, 2));

        searchName = new JLabel("Search By Name");

        searchNameTextField = new JTextField();

        searchNameButton = new JButton("Search");

        home = new JButton("Home");

        this.add(searchName);

        this.add(searchNameTextField);

        this.add(searchNameButton);

        this.add(home);

        myActionListener a = new myActionListener();

        searchNameButton.addActionListener(a);

        home.addActionListener(a);

    }

    // Inner Class

    public class myActionListener implements ActionListener {

        @Override

        public void actionPerformed(ActionEvent e) {

            if(e.getSource() == searchNameButton) {

                OperationsStorage o = new OperationsStorage();

                String name = searchNameTextField.getText();

                String details =  o.searchStudentByName(name);

                JOptionPane.showMessageDialog(null, details);

            }

            else if(e.getSource() == home) {

                dispose();

            }

        }

    }

}

Student

import java.io.Serializable;

public class Student extends Person implements Serializable {

    private double GPA;

    private int semester;

    private String section;

    private Department department;

    public Student() {

        super();

    }

    public Student(String name , String phone , String gender ,double GPA, int semester, String section, Department department) {

        super(name, phone, gender);

        this.GPA = GPA;

        this.semester = semester;

        this.section = section;

        this.department = department;

    }

    public double getGPA() {

        return *this*.GPA;

    }

    public void setGPA(double GPA) {

*this*.GPA = GPA;

    }

    public int getSemester() {

        return *this*.semester;

    }

    public void setSemester(int semester) {

*this*.semester = semester;

    }

    public String getSection() {

        return *this*.section;

    }

    public void setSection(String section) {

*this*.section = section;

    }

    public Department getDepartment() {

        return this.department;

    }

    public void setDepartment(Department department) {

        this.department = department;

    }

    @Override

    public String toString() {

        return "------------------------------------------------\n" +

            super.toString() +

            " GPA : " + getGPA() + "\n" +

            "Semester :" + getSemester() + "\n" +

            "Section :" + getSection() + "\n" +

            "Department :" + getDepartment() + "\n"+

            "------------------------------------------------\n";

    }

}

UpdateScreen

import java.awt.\*;

import javax.swing.\*;

import java.awt.event.\*;

public class UpdateScreen extends JFrame {

    // Data Members

    private JLabel lblupdate;

    private JTextField nameTobeupdated;

    private JLabel lblnewGpa;

    private JTextField newGpa;

    private JButton update;

    private JButton home;

    // Constructor

    public UpdateScreen() {

        this.setSize(600, 600);

        this.setForeground(Color.BLACK);

        this.setVisible(true);

        this.setLayout(new GridLayout(3, 2));

        lblupdate = new JLabel("Enter the Name of Student whose GPA you want to repeat");

        nameTobeupdated = new JTextField(20);

        lblnewGpa = new JLabel("Enter the new GPA");

        newGpa = new JTextField(20);

        update = new JButton("Update");

        home = new JButton("Home");

*this*.add(lblupdate);

*this*.add(nameTobeupdated);

*this*.add(lblnewGpa);

*this*.add(newGpa);

*this*.add(update);

*this*.add(home);

*// adding Action Listener to the buttons*

        myActionListener a = new myActionListener();

        update.addActionListener(a);

        home.addActionListener(a);

    }

    class myActionListener implements ActionListener {

        @Override

        public void actionPerformed(ActionEvent e) {

            if (e.getSource() == update) {

                OperationsStorage o = new OperationsStorage();

                String name = nameTobeupdated.getText();

                double newgpa = Double.parseDouble(newGpa.getText());

                boolean found = o.updateGPA(name, newgpa);

                if(found)

                    JOptionPane.showMessageDialog(null, "GPA Updated !");

                else

                    JOptionPane.showMessageDialog(null, "Student not found !");

            }

            else if (e.getSource() == home) {

                dispose();

                new HomeScreen();

                }

            }

        }

    }

Runner

public class Runner {

    public static void main(String[] args) {

        new HomeScreen();

    }

}

Class Assignment 4

AddScreen

import java.awt.\*;

import javax.swing.\*;

import java.awt.event.\*;

public class AddScreen extends JFrame {

    JLabel nameLabel, jobDesignationLabel, jobSalaryLabel, headNameLabel;

    JTextField nameTextField, jobDesignationTextField, jobSalaryTextField, headNameTextField;

    JButton submit, home;

    public AddScreen() {

        setTitle("Add Employee");

        setSize(600, 600);

        setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        setLayout(new GridLayout(5, 2));

        setVisible(true);

        nameLabel = new JLabel("Name");

        jobDesignationLabel = new JLabel("Job Designation");

        jobSalaryLabel = new JLabel("Job Salary");

        headNameLabel = new JLabel("Head Name");

        nameTextField = new JTextField();

        jobDesignationTextField = new JTextField();

        jobSalaryTextField = new JTextField();

        headNameTextField = new JTextField();

        submit = new JButton("Submit");

        home = new JButton("Home");

        add(nameLabel);

        add(nameTextField);

        add(jobDesignationLabel);

        add(jobDesignationTextField);

        add(jobSalaryLabel);

        add(jobSalaryTextField);

        add(headNameLabel);

        add(headNameTextField);

        add(submit);

        add(home);

        myActionListener a = new myActionListener();

        submit.addActionListener(a);

        home.addActionListener(a);

    }

    public class myActionListener implements ActionListener {

    @Override

    public void actionPerformed(ActionEvent e) {

        if (e.getSource() == submit) {

            FileOperations o = new FileOperations();

            String designation = jobDesignationTextField.getText();

            double salary = Double.parseDouble(jobSalaryTextField.getText());

            Job j = new Job(designation, salary);

            String name = nameTextField.getText();

            String headName = headNameTextField.getText();

            Employee emp = new Employee(name, j, headName);

            o.writeToFile(emp);

            JOptionPane.showMessageDialog(null, "Employee added");

        }

        else if (e.getSource() == home) {

            dispose();

            new HomeScreen();

        }

    }

}

}

Employee

import java.io.Serializable;

public class Employee implements Serializable{

    private String name;

    private Job j;

    private String headName;

    public Employee() {

    }

    public Employee(String name, Job j, String headName) {

*this*.name = name;

*this*.j = j;

*this*.headName = headName;

    }

    public String getName() {

        return *this*.name;

    }

    public void setName(String name) {

*this*.name = name;

    }

    public Job getJ() {

        return *this*.j;

    }

    public void setJ(Job j) {

*this*.j = j;

    }

    public String getHeadName() {

        return *this*.headName;

    }

    public void setHeadName(String headName) {

*this*.headName = headName;

    }

    @Override

    public String toString() {

        return

            " Name = " + getName() + "\n" +

            " Job = " + getJ().toString() + "\n" +

            " Head-Name='" + getHeadName() + "\n";

    }

}

FileOperations

import java.io.EOFException;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.util.ArrayList;

public class FileOperations {

    private ArrayList<Employee> a = new ArrayList<>();

    private File f = new File("employee.ser");

    //! Create for GUI

    public void writeToFile(Employee emp) {

        // file object

        // File f = new File("employee.ser");

        // Object for writing class (ObjectOutputStream)

        // Why this object is NULL?

        // Answer : Because when even the constructor is called, the object (File)

        // is created but we want to keep it in the try block

        ObjectOutputStream oos = null;

        // write to file

        try {

            if (!f.exists()) {

                f = new File("employee.ser");

            }

            if (f.exists()) {

                oos = new MyObjectOutputStream(new FileOutputStream(f, true));

                oos.writeObject(emp); // write object to file

            } else {

                oos = new ObjectOutputStream(new FileOutputStream(f, true));

                oos.writeObject(emp); // it will write the object to the file.

            }

        } catch (IOException e) {

            e.printStackTrace();

        }

        catch (Exception e) {

            System.err.println("Cannot Write Object");

        }

        // For closing File

        if (oos != null) {

            try {

                oos.close();

            } catch (IOException e) {

                e.printStackTrace();

            }

        }

    }

*//! Read for GUI*

    public String readAll() {

        ObjectInputStream oo = null;

        StringBuilder details = new StringBuilder();

        if (!f.exists())

        return "File Not Found No Record!";

    if (f.exists()) {

        try {

            oo = new ObjectInputStream(new FileInputStream("employee.ser"));

            while (true) {

                // Reading object is below

                Employee emp = (Employee) oo.readObject();

                details.append(emp.toString());

            }

        } catch (ClassNotFoundException e) {

            e.printStackTrace();

        }

        catch (EOFException e) {

        }

        catch (FileNotFoundException e) {

            e.printStackTrace();

        }

        catch (IOException e) {

            e.printStackTrace();

        }

        finally {

            try {

                oo.close();

            } catch (IOException e) {

            }

        }

        return details.toString();

    }

    return "File Not Found No Record!";

    }

*//! Search for GUI*

    public String searchEmployeeWithSalaryGreaterorEqualToFiftyThousand(String Name) {

        StringBuilder details = new StringBuilder();

        boolean foundSome = false;

        ObjectInputStream oo = null;

        try {

            oo = new ObjectInputStream(new FileInputStream("employee.ser"));

            while (true) {

*// Reading object is below*

                Employee emp = (Employee) oo.readObject();

                if ((emp.getName().equalsIgnoreCase(Name)) && (emp.getJ().getSalary() >= 50000)){

                    details.append(emp.toString());

                    foundSome = true;

                }

            }

        } catch (ClassNotFoundException e) {

            e.printStackTrace();

        }

        catch (EOFException e) {

        }

        catch (FileNotFoundException e) {

            e.printStackTrace();

        }

        catch (IOException e) {

            e.printStackTrace();

        }

        finally {

            try {

                oo.close();

            } catch (IOException e) {

            }

        }

        if (!foundSome) {

            return "No Student Found !";

        }

        return details.toString();

    }

    //! UPDATE for GUI

    boolean updateJob(String name, String newDesignation , double newSalary) {

        boolean found = false;

        ObjectInputStream oo = null;

        try {

            oo = new ObjectInputStream(new FileInputStream("employee.ser"));

            try {

                while (true) {

                    Employee emp = (Employee) oo.readObject();

                    a.add(emp);

                }

            } catch (EOFException e) {

            }

        } catch (FileNotFoundException e) {

            e.printStackTrace();

        } catch (IOException e) {

            e.printStackTrace();

        } catch (ClassNotFoundException e) {

            e.printStackTrace();

        }

        finally {

            try {

                oo.close();

            } catch (IOException e) {

            }

        }

        // firstly manipulating the desired object for updating the Job

        for (int i = 0; i < a.size(); i++) {

            if (a.get(i).getName().equalsIgnoreCase(name)) {

                found = true;

                a.get(i).getJ().setDesignation(newDesignation);

                a.get(i).getJ().setSalary(newSalary);

            }

        }

        //\* now again writing the Arraylist Objects to the file. first time it will create the file again and only then it will append!

        ObjectOutputStream oos = null;

        int counter = 0;

        try {

            for (int i = 0; i < a.size(); i++) {

                if (counter > 0) {

                    oos = new MyObjectOutputStream(new FileOutputStream(f, true));

                    oos.writeObject(a.get(i));

                }

                else {

                    oos = new ObjectOutputStream(new FileOutputStream(f));

                    oos.writeObject(a.get(i));

                    counter++;

                }

            }

            // For closing File

            if (oos != null) {

                oos.close();

            }

        }

        catch (IOException e) {

            e.printStackTrace();

        }

        return found;

    }

}

HomeScreen

import java.awt.\*;

import javax.swing.\*;

import java.awt.event.\*;

public class HomeScreen extends JFrame {

    private JButton writeButton;

    private JButton readButton;

    private JButton searchButton;

    private JButton updateButton;

    public HomeScreen() {

        setTitle("Home");

        setSize(500, 500);

        setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        setLayout(new GridLayout(4, 1));

        setVisible(true);

        writeButton = new JButton("Write");

        readButton = new JButton("Read");

        searchButton = new JButton("Search");

        updateButton = new JButton("Update");

        add(writeButton);

        add(readButton);

        add(searchButton);

        add(updateButton);

        myActionListener a = new myActionListener();

        writeButton.addActionListener(a);

        readButton.addActionListener(a);

        searchButton.addActionListener(a);

        updateButton.addActionListener(a);

    }

    public class myActionListener implements ActionListener {

        @Override

        public void actionPerformed(ActionEvent e) {

            if (e.getSource() == writeButton) {

                dispose();

                new AddScreen();

            }

            else if (e.getSource() == readButton) {

                FileOperations o = new FileOperations();

                String data = o.readAll();

                JOptionPane.showMessageDialog(null , data);

            }

            else if (e.getSource() == searchButton) {

                dispose();

                new SearchScreen();

            }

            else if (e.getSource() == updateButton) {

                dispose();

                new UpdateScreen();

            }

        }

    }

}

Job

import java.io.Serializable;

public class Job implements Serializable{

    private String designation;

    private double salary;

    public Job() {

    }

    public Job(String designation, double salary) {

*this*.designation = designation;

*this*.salary = salary;

    }

    public String getDesignation() {

        return *this*.designation;

    }

    public void setDesignation(String designation) {

*this*.designation = designation;

    }

    public double getSalary() {

        return *this*.salary;

    }

    public void setSalary(double salary) {

        this.salary = salary;

    }

    @Override

    public String toString() {

        return

            " Designation =" + getDesignation() + "\n" +

            " Salary= " + getSalary() + "\n";

    }

}

SearchScreen

import java.awt.\*;

import javax.swing.\*;

import java.awt.event.\*;

public class SearchScreen extends JFrame {

    private JLabel nameLabel;

    private JTextField nameTextField;

    private JButton search, home;

    public SearchScreen() {

        setTitle("Search");

        setSize(500, 500);

        setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        setLayout(new GridLayout(2, 2));

        setVisible(true);

        nameLabel = new JLabel("Name");

        nameTextField = new JTextField();

        search = new JButton("Search");

        home = new JButton("Home");

        add(nameLabel);

        add(nameTextField);

        add(search);

        add(home);

        myActionListener a = new myActionListener();

        search.addActionListener(a);

        home.addActionListener(a);

    }

    public class myActionListener implements ActionListener {

        @Override

        public void actionPerformed(ActionEvent e) {

            if (e.getSource() == search) {

                FileOperations o = new FileOperations();

                String name = nameTextField.getText();

                String data = o.searchEmployeeWithSalaryGreaterorEqualToFiftyThousand(name);

                JOptionPane.showMessageDialog(null, data);

            }

            else if (e.getSource() == home) {

                dispose();

                new HomeScreen();

            }

        }

    }

}

UpdateScreen

import java.awt.\*;

import javax.swing.\*;

import java.awt.event.\*;

public class UpdateScreen extends JFrame {

    JLabel nameLabel, newJobDesignationLabel, newJobSalaryLabel;

    JTextField nameTextField, newJobDesignationTextField, newJobSalaryTextField;

    JButton update, home;

    public UpdateScreen() {

        setTitle("Update Job");

        setSize(500, 500);

        setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        setLayout(new GridLayout(4, 2));

        setVisible(true);

        nameLabel = new JLabel("Name");

        newJobDesignationLabel = new JLabel("New Job Designation");

        newJobSalaryLabel = new JLabel("New Job Salary");

        nameTextField = new JTextField();

        newJobDesignationTextField = new JTextField();

        newJobSalaryTextField = new JTextField();

        update = new JButton("Update");

        home = new JButton("Home");

        add(nameLabel);

        add(nameTextField);

        add(newJobDesignationLabel);

        add(newJobDesignationTextField);

        add(newJobSalaryLabel);

        add(newJobSalaryTextField);

        add(update);

        add(home);

        myActionListener a = new myActionListener();

        update.addActionListener(a);

        home.addActionListener(a);

    }

    public class myActionListener implements ActionListener {

        @Override

        public void actionPerformed(ActionEvent e) {

            if (e.getSource() == update) {

                System.out.println("Update Triggered");

                FileOperations o = new FileOperations();

                String name = nameTextField.getText();

                String newJobDesignation = newJobDesignationTextField.getText();

                double newJobSalary = Double.parseDouble(newJobSalaryTextField.getText());

                boolean found = o.updateJob(name, newJobDesignation, newJobSalary);

                if(found) {

                    JOptionPane.showMessageDialog(null, "Job Updated");

                }

                else {

                    JOptionPane.showMessageDialog(null, "No such Person Found");

                }

            }

            else if (e.getSource() == home) {

                dispose();

                new HomeScreen();

            }

        }

    }

}