Scaffold 3_c

Group Members:

King Aj Magalona, BSCS

Prince Oncada, BSCS

Alfred Ashley F. Andrion, BSIS

GITHUB LINK: https://github.com/KuuShyn/ShynLabCVG

Source Codes:

//CV.java

```
import java.io.File;
import java.io.IOException;

import org.apache.pdfbox.pdmodel.PDDocument;
import org.apache.pdfbox.pdmodel.PDPage;
import org.apache.pdfbox.pdmodel.PDPageContentStream;
import org.apache.pdfbox.pdmodel.font.PDFont;
import org.apache.pdfbox.pdmodel.font.PDType0Font;

import org.apache.pdfbox.pdmodel.graphics.image.PDImageXObject;

import java.util.LinkedList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

import java.awt.Color;

public class CV {
    int filecount = 0;

    String userHomeDir = System.getProperty("user.home");
    String path = userHomeDir + "\\Desktop\\CVs\\CV" + filecount + ".pdf";

    PDDocument cvDoc = new PDDocument();
    PDPage firstpage = new PDPage();
    PDPageContentStream contentStream;
    PDImageXObject pdPerson, pdTele, pdMail;

    File fileSerifBold;
    File fileSerifSemiBold;

    PDFont serifBold;
```

```
    PDFont serifSemiBold;

    Experience experience;
    Education education;
    Personal personal;
    Skills skills;
    Summary summary;

    // int expX = 170;
    int expTitle = 165;
    int expY = 180;

    int eduTitle = 165;
    int eduY = 180;

    int skillTitle = 165;
    int skillY = 180;

    // int summX = 170;
    int summY = 45;

    String eduDesc;

    public CV() throws IOException {
        cvDoc.addPage(firstpage);

        contentStream = new PDPageContentStream(cvDoc, firstpage,
PDPageContentStream.AppendMode.OVERWRITE, true, true);
        contentStream.setNonStrokingColor(new Color(63, 101, 146));
        contentStream.addRect(0, firstpage.getTrimBox().getHeight(), 150, -1000);
        contentStream.fill();
        contentStream.close();

        icons();

    }

    public void setDescEdu(String str) { this.eduDesc = str; }

    public void setPath(String path) { this.path = path; }

    public void setSummY(int y) { this.summY += y; }

    // public void setExpX(int x) { this.expX *= x; }

    public void setExpY(int y) { this.expY += y; }
```

```
    public void setEduY(int y) { this.eduY += y; }

    public void setEduTitle(int y) { this.eduTitle += y; }

    public void setSkillY(int y) { this.skillY += y; }

    public void setSkillTitle(int y) { this.skillTitle += y; }

    public void close() throws IOException { cvDoc.close(); }

    public void expTitle() throws IOException {
        load();

        contentStream = new PDPageContentStream(cvDoc, firstpage,
PDPageContentStream.AppendMode.APPEND, true, true);

        contentStream.beginText();
        contentStream.setNonStrokingColor(Color.black);
        contentStream.setFont(serifBold, 12);
        contentStream.setLeading(16.0f);
        contentStream.newLineAtOffset(170, firstpage.getTrimBox().getHeight() - 165);
        contentStream.showText("WORK EXPERIENCE");
        contentStream.newLine();
        contentStream.endText();
        contentStream.close();
    }

    public void eduTitle() throws IOException {
        load();

        contentStream = new PDPageContentStream(cvDoc, firstpage,
PDPageContentStream.AppendMode.APPEND, true, true);

        contentStream.beginText();
        contentStream.setNonStrokingColor(Color.black);
        contentStream.setFont(serifBold, 12);
        contentStream.setLeading(16.0f);
        contentStream.newLineAtOffset(170, firstpage.getTrimBox().getHeight() -
eduTitle);
        contentStream.showText("Education and Qualifications");
        contentStream.newLine();
        contentStream.endText();
        contentStream.close();
    }
```

```
    public void skillTitle() throws IOException {
        load();

        contentStream = new PDPageContentStream(cvDoc, firstpage,
PDPageContentStream.AppendMode.APPEND, true, true);

        contentStream.beginText();
        contentStream.setNonStrokingColor(Color.black);
        contentStream.setFont(serifBold, 12);
        contentStream.setLeading(16.0f);
        contentStream.newLineAtOffset(170, firstpage.getTrimBox().getHeight() -
skillTitle);
        contentStream.showText("Skills");
        contentStream.newLine();
        contentStream.endText();
        contentStream.close();
    }

    public void stream() throws IOException {
        contentStream = new PDPageContentStream(cvDoc, firstpage,
PDPageContentStream.AppendMode.APPEND, true, true);
    }

    public void save() throws IOException {
        cvDoc.save(path);

        Logger logger = Logger.getLogger(CV.class.getName());
        logger.log(Level.INFO, "CV has been Created");

    }

    private void load() throws IOException {
        fileSerifBold = new File(userHomeDir +
"\\Documents\\ShynLabCVG\\Fonts\\OpenSans-Bold.ttf");
        fileSerifSemiBold = new File(userHomeDir +
"\\Documents\\ShynLabCVG\\Fonts\\OpenSans-Semibold.ttf");

        serifBold = PDType0Font.load(cvDoc, fileSerifBold);
        serifSemiBold = PDType0Font.load(cvDoc, fileSerifSemiBold);
    }

    public void icons() throws IOException {

        contentStream = new PDPageContentStream(cvDoc, firstpage,
PDPageContentStream.AppendMode.APPEND, true, true);
```

```
        pdPerson = PDImageXObject.createFromFile(userHomeDir +
"\\Documents\\ShynLabCVG\\icons\\user.png", cvDoc);
        contentStream.drawImage(pdPerson, 14, 745, 10, 10);

        pdMail = PDImageXObject.createFromFile(userHomeDir +
"\\Documents\\ShynLabCVG\\icons\\mail.png", cvDoc);
        contentStream.drawImage(pdMail, 14, 706, 10, 10);

        pdTele = PDImageXObject.createFromFile(userHomeDir +
"\\Documents\\ShynLabCVG\\icons\\phone.png", cvDoc);
        contentStream.drawImage(pdTele, 14, 667, 10, 10);

        contentStream.close();
    }

    public void personal(String name, String email, String pNumber) throws
IOException {
        personal = new Personal();

        load();

        contentStream = new PDPageContentStream(cvDoc, firstpage,
PDPageContentStream.AppendMode.APPEND, true, true);

        contentStream.beginText();
        contentStream.setNonStrokingColor(Color.white);
        contentStream.setFont(serifBold, 14);
        contentStream.setLeading(16.0f);
        contentStream.newLineAtOffset(15, firstpage.getTrimBox().getHeight() - 25);
        contentStream.showText("Personal");
        contentStream.newLine();
        contentStream.endText();

        contentStream.beginText();
        contentStream.setLeading(16.0f);
        contentStream.setFont(serifBold, 10);
        contentStream.newLineAtOffset(30, firstpage.getTrimBox().getHeight() - 45);
        contentStream.showText("Name");
        contentStream.newLine();

        contentStream.setFont(serifSemiBold, 10);
        contentStream.showText(name);
        contentStream.newLine();
        contentStream.endText();

        contentStream.beginText();
```

```
        contentStream.setFont(serifBold, 10);
        contentStream.newLineAtOffset(30, firstpage.getTrimBox().getHeight() - 85);
        contentStream.showText("Email");
        contentStream.newLine();

        contentStream.setFont(serifSemiBold, 10);
        contentStream.showText(email);
        contentStream.newLine();
        contentStream.endText();

        contentStream.beginText();
        contentStream.setFont(serifBold, 10);
        contentStream.newLineAtOffset(30, firstpage.getTrimBox().getHeight() - 125);
        contentStream.showText("Phone NUmber");
        contentStream.newLine();
        contentStream.setFont(serifSemiBold, 10);
        contentStream.showText(pNumber);
        contentStream.newLine();
        contentStream.endText();

        contentStream.beginText();
        contentStream.setNonStrokingColor(Color.black);
        contentStream.setFont(serifBold, 18);
        contentStream.newLineAtOffset(170, firstpage.getTrimBox().getHeight() - 25);
        contentStream.showText(name);
        contentStream.endText();

        contentStream.close();

    }

    public void experience(String jobTitle, String city, String employer, String
startMoYe, String endMoYe,
            List<String> description) throws IOException {
        load();
        contentStream = new PDPageContentStream(cvDoc, firstpage,
PDPageContentStream.AppendMode.APPEND, true, true);

        contentStream.beginText();
        contentStream.setNonStrokingColor(Color.gray);
        contentStream.setFont(serifSemiBold, 10);
        contentStream.setLeading(16.0f);
        contentStream.newLineAtOffset(470, firstpage.getTrimBox().getHeight() - expY);
        contentStream.showText(startMoYe + " - " + endMoYe);
        contentStream.newLine();
        contentStream.endText();
```

```
        contentStream.beginText();
        contentStream.setNonStrokingColor(Color.black);
        contentStream.newLineAtOffset(175, firstpage.getTrimBox().getHeight() - expY);
        contentStream.setFont(serifBold, 10);
        contentStream.showText(jobTitle);
        contentStream.newLine();
        contentStream.setFont(serifSemiBold, 9);
        contentStream.showText(employer + ", " + city);
        contentStream.newLine();
        contentStream.setFont(serifSemiBold, 8);
        for (String txt : description) {

            contentStream.showText(txt.trim());
            contentStream.newLine();

        }

        contentStream.endText();

        contentStream.close();
    }

    public void education(String degree, String city, String school, String startMoye,
String endMoye, List<String> description)
            throws IOException {
        load();
        contentStream = new PDPageContentStream(cvDoc, firstpage,
PDPageContentStream.AppendMode.APPEND, true, true);

        contentStream.beginText();
        contentStream.setNonStrokingColor(Color.gray);
        contentStream.setFont(serifSemiBold, 10);
        contentStream.setLeading(16.0f);
        contentStream.newLineAtOffset(470, firstpage.getTrimBox().getHeight() - eduY);
        contentStream.showText(startMoye + " - " + endMoye);
        contentStream.newLine();
        contentStream.endText();

        contentStream.beginText();
        contentStream.setNonStrokingColor(Color.black);
        contentStream.newLineAtOffset(175, firstpage.getTrimBox().getHeight() - eduY);
        contentStream.setFont(serifBold, 10);
        contentStream.showText(degree);
        contentStream.newLine();
        contentStream.setFont(serifSemiBold, 9);
```

```
        contentStream.showText(school + ", " + city);
        contentStream.newLine();
        contentStream.setFont(serifSemiBold, 8);

        for (String txt : description) {

            contentStream.showText(txt.trim());
            contentStream.newLine();

        }

        contentStream.endText();

        contentStream.close();
    }

    public void skill(String skill, String skillLevel) throws IOException {
        load();
        contentStream = new PDPageContentStream(cvDoc, firstpage,
PDPageContentStream.AppendMode.APPEND, true, true);

        contentStream.beginText();
        contentStream.setFont(serifBold, 10);
        contentStream.setLeading(16.0f);
        contentStream.newLineAtOffset(175, firstpage.getTrimBox().getHeight() - skillY);
        contentStream.showText(skill + "                    " + skillLevel);
        contentStream.newLine();

        contentStream.endText();

        contentStream.close();
    }

    public void summary(String summary) throws IOException {
        load();

        contentStream = new PDPageContentStream(cvDoc, firstpage,
PDPageContentStream.AppendMode.APPEND, true, true);

        contentStream.beginText();
        contentStream.setFont(serifBold, 10);
        contentStream.setLeading(16.0f);

        contentStream.newLineAtOffset(170, firstpage.getTrimBox().getHeight() -
summY);
```

```
        contentStream.showText(summary);
        contentStream.endText();

        contentStream.close();

    }

}
```

//CVG.java

```
import javax.swing.*;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
import java.awt.*;
import java.io.File;
import java.io.IOException;

public class CVG extends JFrame implements ActionListener, WindowListener {
    private int currentCard = 1;

    String userHomeDir = System.getProperty("user.home");

    private JButton btnPrevious, btnNext, btnSave;
    private JPanel cPanel, btnPanel;

    private CardLayout card;

    Experience experience;

    Education education;
    Personal personal;
    Skills skills;
    Summary summary;

    CV cv;

    boolean dtlsPersonal, dtlsExp, dtlsEdu, dtlsSkill, dtlsSumm;

    public CVG() throws IOException {

        // Holds the the thing yehe
        cPanel = new JPanel();

        // Holds the Card containers
```

```
card = new CardLayout();

// Sets the Layout container of the panel to card
cPanel.setLayout(card);

// initialize classes
personal = new Personal();
experience = new Experience();
education = new Education();
skills = new Skills();
summary = new Summary();

cv = new CV();

// Card containers
cPanel.add(personal, "1");
cPanel.add(experience, "2");
cPanel.add(education, "3");
cPanel.add(skills, "4");
cPanel.add(summary, "5");

// panel that holds buttons
btnPanel = new JPanel();

btnPrevious = new JButton("Previous");
btnNext = new JButton("Next");
btnSave = new JButton("save");

btnPrevious.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
btnNext.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
btnSave.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));

// add buttons to btnPanel
btnPanel.add(btnPrevious).setEnabled(false);
;
btnPanel.add(btnNext);
btnPanel.add(btnSave).setEnabled(false);

add(cPanel);
add(btnPanel, BorderLayout.SOUTH);

addWindowListener(this);

btnPrevious.addActionListener(this);
btnNext.addActionListener(this);
btnSave.addActionListener(this);
```

```
    setTitle("CV Generator");
    setSize(350, 250);

    setVisible(true);
    setDefaultCloseOperation(DISPOSE_ON_CLOSE);
    setLocationRelativeTo(null);
}

public static void main(String[] args) throws IOException { new Menu(); }

// @Override
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == btnPrevious) {
        if (currentCard > 1) {
            currentCard--;
            card.show(cPanel, "" + currentCard);

        }
        if (currentCard == 1) {

            setSize(350, 250);
            setLocationRelativeTo(null);

            btnPrevious.setEnabled(false);
        }
        if (currentCard == 2 || currentCard == 3) {
            setSize(500, 500);

            setLocationRelativeTo(null);

        }
        if (currentCard == 4) {
            setSize(500, 250);
            setLocationRelativeTo(null);

        }

        if (currentCard != 5)
            btnNext.setEnabled(true);
        btnSave.setEnabled(false);
    }

    if (e.getSource() == btnNext) {
        if (currentCard < 5) {
            currentCard++;
```

```
            card.show(cPanel, "" + currentCard);
        }
        if (currentCard != 1) {
            setSize(500, 500);
            setLocationRelativeTo(null);

            btnPrevious.setEnabled(true);
        }
        if (currentCard == 4) {
            setSize(500, 250);
            setLocationRelativeTo(null);

        }

        if (currentCard == 2) {
            if (personal.pValidation()) {
                dtlsPersonal = true;
            }
        }

        if (currentCard == 5) {
            btnNext.setEnabled(false);
            btnSave.setEnabled(true);
        }

    }
    if (e.getSource() == btnSave) {
        try {
            if (personal.pValidation()) {

                cv.personal(personal.getNames(), personal.getEmail(),
personal.getPhoneNum());
                dtlsPersonal = true;

            }
            if (experience.getChkbox()) {
                cv.expTitle();
                if (experience.wrkValidation()) {
                    for (ExpDetails expDetails : experience.getDetails()) {

                        cv.experience(expDetails.getJobTitle(), expDetails.getCityTown(),
expDetails.getEmployer(),
                                expDetails.getStartMoYe(), expDetails.getEndMoye(),
expDetails.getDesc());

                        cv.setExpY(80);
```

```
                    cv.setEduY(100);
                    cv.setEduTitle(100);

                    cv.setSkillY(100);
                    cv.setSkillTitle(100);

                }
                dtlsExp = true;

            }

        }
        else {
            dtlsExp = true;
        }

        if (education.edukValidation()) {
            for (EduDetails eduDetails : education.getDetails()) {

                cv.education(eduDetails.getDegree(), eduDetails.getCityTown(),
eduDetails.getSchool(),
                        eduDetails.getStartMoYe(), eduDetails.getEndMoye(),
eduDetails.getDesc());

                cv.setEduY(80);

                cv.setSkillY(100);
                cv.setSkillTitle(100);
            }
            dtlsEdu = true;

        }
        if (skills.skillValidation()) {
            for (SkillDetails skillDetails : skills.getDetails()) {
                cv.skill(skillDetails.getSkill(), skillDetails.getLevel());

                cv.setSkillY(50);
            }
            dtlsSkill = true;

        }

        if (!summary.getSummaryDesc().isEmpty()) {
            for (String text : summary.getSummaryDesc()) {
                cv.summary(text);
```

```
                    cv.setSummY(15);
                }
                dtlsSumm = true;

            }

            if (dtlsPersonal && dtlsExp && dtlsEdu && dtlsSumm) {
                int filecount = 0;

                cv.eduTitle();
                cv.skillTitle();
                File folder = new File(userHomeDir + "\\Desktop\\CVs\\");
                File[] listOfFiles = folder.listFiles();
                for (File file : listOfFiles) {
                    int temp = listOfFiles.length;
                    if (file.exists()) {
                        filecount++;
                        if (filecount == temp) {

                            String path = userHomeDir + "\\Desktop\\CVs\\CV" + filecount +
".pdf";

                            cv.setPath(path);
                        }
                    }
                }
                cv.save();
                dispose();
                new Menu().setVisible(true);

            }

        }
        catch (IOException e1) {

            e1.printStackTrace();
        }

    }

}

@Override
public void windowClosed(WindowEvent e) {
    // TODO Auto-generated method stub
}
```

```
    @Override
    public void windowOpened(WindowEvent e) {
        // TODO Auto-generated method stub

    }

    @Override
    public void windowClosing(WindowEvent e) {
        try {
            cv.close();
            new Menu().setVisible(true);
        }
        catch (IOException e1) {
            e1.printStackTrace();
        }

    }

    @Override
    public void windowIconified(WindowEvent e) {
        // TODO Auto-generated method stub

    }

    @Override
    public void windowDeiconified(WindowEvent e) {
        // TODO Auto-generated method stub

    }

    @Override
    public void windowActivated(WindowEvent e) {
        // TODO Auto-generated method stub

    }

    @Override
    public void windowDeactivated(WindowEvent e) {
        // TODO Auto-generated method stub

    }
}
```

//EduDetails.java

```
import java.util.LinkedList;
import java.util.List;
```

```
public class EduDetails {
   private String degree, city, school, startMoYe, startMonth, startYear, endMoYe,
endMonth, endYear;
   private List<String> desc;

   public EduDetails(String degree, String city, String school, String startMonth, String
startYear, String endMonth,
         String endYear, List<String> desc) {
      this.degree = degree;
      this.city = city;
      this.school = school;
      this.startMoYe = startMonth + " " + startYear;
      this.startMonth = startMonth;
      this.startYear = startYear;
      this.endMoYe = endMonth + " " + endYear;
      this.endMonth = endMonth;
      this.endYear = endYear;
      this.desc = desc;
   }

   public String getDegree() { return degree; }

   public String getCityTown() { return city; }

   public String getSchool() { return school; }

   public String getStartMoYe() { return startMoYe; }

   public String getStartMonth() { return startMonth; }

   public String getStartYear() { return startYear; }

   public String getEndMoye() { return endMoYe; }

   public String getEndMonth() { return endMonth; }

   public String getEndYear() { return endYear; }


   public List<String> getDesc() {
      List<String> lines = new LinkedList<>();
      for (String line : desc)
         lines.add(line);
      return lines;
```

```
    }
}
```

//Education.java

```
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import java.awt.Color;
import java.awt.*;

import javax.swing.BorderFactory;
import javax.swing.DefaultListModel;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JLabel;
import javax.swing.JList;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.ScrollPaneConstants;

import java.util.Arrays;
import java.util.LinkedList;
import java.util.List;
import java.util.stream.Collectors;

public class Education extends JPanel implements ActionListener {
    private JLabel lblEducation, lblDegree, lblCity, lblSchool, lblStartDate, lblEndDate,
lblDescription;
    private JTextField txtDegree, txtCity, txtSchool;
    private JComboBox<String> cboStartMonth, cboEndMonth;
    private JComboBox<Integer> cboStartYear, cboEndYear;
    private JTextArea txtDescription;

    private JScrollPane scroll;
    private JButton btnAddUpdate, btnEdit, btnDelete;

    int value;

    List<EduDetails> list = new LinkedList<EduDetails>();

    private static final String FONTSTYLE = "Berlin Sans FB";

    public Education() {
```

```
        String[] months = {
                "January", "Febuary", "March", "April", "May", "June", "July", "August",
"September", "October", "November",
                "December"
        };
        Integer[] years = new Integer[62];
        int sartYear = 1960;
        int endYear = 2022;

        for (int i = 0; i < 62; i++) {
            years[i] = sartYear;
            sartYear++;
            if (sartYear == endYear) {
                years[i] = endYear;
                break;
            }
        }

        lblEducation = new JLabel("Education");
        lblEducation.setFont(new Font("Consolas", Font.PLAIN, 20));
        lblEducation.setBounds(20, 15, 180, 20);

        lblDegree = new JLabel("Degree");
        lblDegree.setFont(new Font(FONTSTYLE, Font.PLAIN, 15));
        lblDegree.setBounds(60, 35, 75, 20);

        lblCity = new JLabel("City/Town");
        lblCity.setFont(new Font(FONTSTYLE, Font.PLAIN, 15));
        lblCity.setBounds(41, 65, 75, 20);

        lblSchool = new JLabel("School");
        lblSchool.setFont(new Font(FONTSTYLE, Font.PLAIN, 15));
        lblSchool.setBounds(65, 95, 75, 20);

        lblStartDate = new JLabel("Start Date");
        lblStartDate.setFont(new Font(FONTSTYLE, Font.PLAIN, 14));
        lblStartDate.setBounds(40, 128, 75, 20);

        lblEndDate = new JLabel("End Date");
        lblEndDate.setFont(new Font(FONTSTYLE, Font.PLAIN, 14));
        lblEndDate.setBounds(245, 128, 75, 20);

        lblDescription = new JLabel("Description");
        lblDescription.setFont(new Font(FONTSTYLE, Font.PLAIN, 15));
        lblDescription.setBounds(40, 185, 75, 20);
```

```
        txtDegree = new JTextField();
        txtDegree.setBounds(120, 38, 150, 20);

        txtCity = new JTextField();
        txtCity.setBounds(120, 68, 150, 20);

        txtSchool = new JTextField();
        txtSchool.setBounds(120, 98, 150, 20);

        cboStartMonth = new JComboBox<>(months);
        cboStartMonth.setBounds(40, 153, 95, 20);

        cboStartYear = new JComboBox<>(years);
        cboStartYear.setBounds(140, 153, 95, 20);

        cboEndMonth = new JComboBox<>(months);
        cboEndMonth.setBounds(245, 153, 95, 20);

        cboEndYear = new JComboBox<>(years);
        cboEndYear.setBounds(345, 153, 95, 20);
        txtDescription = new JTextArea("", 0, 0);

        txtDescription.setBorder(BorderFactory.createLineBorder(Color.BLACK));


        scroll = new JScrollPane(txtDescription);

scroll.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_AL
WAYS);
        scroll.setBounds(40, 230, 400, 100);


        btnAddUpdate = new JButton("Add");
        btnAddUpdate.setBounds(370, 15, 80, 30);

btnAddUpdate.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));

        btnEdit = new JButton("Edit");
        btnEdit.setBounds(370, 55, 80, 30);
        btnEdit.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));

        btnDelete = new JButton("Delete");
        btnDelete.setBounds(370, 95, 80, 30);
        btnDelete.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
```

```
        btnAddUpdate.addActionListener(this);
        btnEdit.addActionListener(this);
        btnDelete.addActionListener(this);

        add(btnAddUpdate);
        add(btnEdit).setEnabled(false);
        add(btnDelete).setEnabled(false);

        add(lblEducation);

        add(lblDegree);
        add(lblCity);
        add(lblSchool);
        add(lblStartDate);
        add(lblEndDate);
        add(lblDescription);

        add(txtDegree);
        add(txtCity);
        add(txtSchool);

        add(cboStartMonth);
        add(cboStartYear);
        add(cboEndMonth);
        add(cboEndYear);

        add(scroll);
        setLayout(null);

    }
    public List<EduDetails> getDetails() { return list; }

    public static String capitalizeFully(String str) {
        if (str == null || str.isEmpty()) {
            return str;
        }

        return Arrays.stream(str.split("\\s+")).map(t -> t.substring(0, 1).toUpperCase() +
t.substring(1).toLowerCase())
                .collect(Collectors.joining(" "));
    }

    public String getJobTitle() { return capitalizeFully(txtDegree.getText()); }

    public String getCityTown() { return capitalizeFully(txtCity.getText()); }
```

```
    public String getEmployer() { return capitalizeFully(txtSchool.getText()); }

    public String getStartMonth() { return (String) cboStartMonth.getSelectedItem(); }

    public String getStartYear() { return (Integer) cboStartYear.getSelectedItem() + ""; }

    public String getEndMonth() { return (String) cboEndMonth.getSelectedItem(); }

    public String getEndYear() { return (Integer) cboEndYear.getSelectedItem() + ""; }

    public List<String> getDesc() {
        List<String> lines = new LinkedList<>();
        for (String line : txtDescription.getText().split("\\n"))
            lines.add(line);
        return lines;
    }



    public boolean nextPressed(boolean bool){
        return bool;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getSource().equals(btnAddUpdate)) {
            if (btnAddUpdate.getText().equalsIgnoreCase("Update")) {
                int i = 0;
                System.out.println("Updated!");
                btnAddUpdate.setText("Add");
                for (EduDetails eduDetails : list) {
                    i = list.indexOf(eduDetails);
                    if (i == value) {
                        System.out.println("should return list:" + i);

                        list.set(i,
                                new EduDetails(this.getJobTitle(), this.getCityTown(),
this.getEmployer(), this.getStartMonth(),
                                        this.getStartYear(), this.getEndMonth(), this.getEndYear(),
this.getDesc()));


                        clearFields();
                    }
                }
                btnDelete.setEnabled(false);
```

```
        }
        else {
            if (isFieldEmpty()) {
            System.out.println("added");
            add();

            btnEdit.setEnabled(true);

            clearFields();
          }
        }

      }
      if (e.getSource().equals(btnEdit)) {
        DefaultListModel<String> t1 = new DefaultListModel<String>();

        JList<String> jl = new JList<String>(t1);

        String hold = "";
        try {
          for (EduDetails degree : list) {

            t1.addElement(degree.getDegree());
          }

          int option = JOptionPane.showConfirmDialog(null, jl, "Education: Edit", 0);
          if (option == JOptionPane.YES_OPTION) {
            btnDelete.setEnabled(true);
            value = jl.getSelectedIndex();

            for (EduDetails eduDetails : list) {
              if (list.indexOf(eduDetails) == value) {
                setFields(eduDetails.getDegree(), eduDetails.getCityTown(),
eduDetails.getSchool(),
                    eduDetails.getStartMonth(), eduDetails.getStartYear(),
eduDetails.getEndMonth(),
                    eduDetails.getEndYear(), hold);

                for (String desc : eduDetails.getDesc()) {
                  hold += desc + "\n";
                  setFields(eduDetails.getDegree(), eduDetails.getCityTown(),
eduDetails.getSchool(),
                      eduDetails.getStartMonth(), eduDetails.getStartYear(),
eduDetails.getEndMonth(),
                      eduDetails.getEndYear(), hold);
```

```
                    }

                }

            }
            if (btnAddUpdate.getText().equalsIgnoreCase("Add"))
                btnAddUpdate.setText("Update");

        }

      }
      catch (NullPointerException nullP) {
          nullP.printStackTrace();
      }
    }

    if (e.getSource().equals(btnDelete)) {
      for (EduDetails eduDetails : list) {
          if (list.indexOf(eduDetails) == value) {
              System.out.println("should return");
              list.remove(eduDetails);

          }
      }
      clearFields();
      btnDelete.setEnabled(false);
      btnAddUpdate.setText("Add");

    }
  }


  public void add(){
    list.add(new EduDetails(this.getJobTitle(), this.getCityTown(), this.getEmployer(),
this.getStartMonth(),
                this.getStartYear(), this.getEndMonth(), this.getEndYear(),
this.getDesc()));
  }


  public boolean isFieldEmpty() {
    if (!this.getJobTitle().isEmpty() && !this.getCityTown().isEmpty() &&
!this.getEmployer().isEmpty() && !txtDescription.getText().isEmpty())

      return true;
    else
```

```
            return false;
    }

    public boolean edukValidation() {
        if (!list.isEmpty())
            return true;

        else {
            return false;
        }
    }

    public void enable() {
        btnAddUpdate.setEnabled(true);
        // btnEdit.setEnabled(true);
        // btnDelete.setEnabled(true);
        txtDegree.setEnabled(true);
        txtCity.setEnabled(true);
        txtSchool.setEnabled(true);
        cboStartMonth.setEnabled(true);
        cboStartYear.setEnabled(true);
        cboEndMonth.setEnabled(true);
        cboEndYear.setEnabled(true);
        txtDescription.setEnabled(true);

    }

    public void disable() {
        btnAddUpdate.setEnabled(false);
        btnEdit.setEnabled(false);
        btnDelete.setEnabled(false);
        txtDegree.setEnabled(false);
        txtCity.setEnabled(false);
        txtSchool.setEnabled(false);
        cboStartMonth.setEnabled(false);
        cboStartYear.setEnabled(false);
        cboEndMonth.setEnabled(false);
        cboEndYear.setEnabled(false);
        txtDescription.setEnabled(false);

    }

    public void setFields(String job, String city, String employer, String startMOnth,
String startYear, String endMonth,
            String endYear, String desc) {
```

```
        txtDegree.setText(job);
        txtCity.setText(city);
        txtSchool.setText(employer);
        cboStartMonth.setSelectedItem(startMOnth);
        cboStartYear.setSelectedItem(startYear);
        cboEndMonth.setSelectedItem(endMonth);
        cboEndYear.setSelectedItem(endYear);
        txtDescription.setText(desc);

    }

    public void clearFields() {
        txtDegree.setText("");
        txtCity.setText("");
        txtSchool.setText("");
        cboStartMonth.setSelectedIndex(0);
        cboStartYear.setSelectedIndex(0);
        cboEndMonth.setSelectedIndex(0);
        cboEndYear.setSelectedIndex(0);
        txtDescription.setText("");
    }

}
```

//ExpDetails.java

```
import java.util.LinkedList;
import java.util.List;

public class ExpDetails {
    private String jobTitle, city, employer, startMoYe, startMonth, startYear, endMoYe,
endMonth, endYear;
    private List<String> desc;

    public ExpDetails(String jobTitle, String city, String employer, String startMonth,
String startYear, String endMonth,
            String endYear, List<String> desc) {
        this.jobTitle = jobTitle;
        this.city = city;
        this.employer = employer;
        this.startMoYe = startMonth + " " + startYear;
        this.startMonth = startMonth;
        this.startYear = startYear;
        this.endMoYe = endMonth + " " + endYear;
        this.endMonth = endMonth;
        this.endYear = endYear;
        this.desc = desc;
```

```
    }

    public String getJobTitle() { return jobTitle; }

    public String getCityTown() { return city; }

    public String getEmployer() { return employer; }

    public String getStartMoYe() { return startMoYe; }

    public String getStartMonth() { return startMonth; }

    public String getStartYear() { return startYear; }

    public String getEndMoye() { return endMoYe; }

    public String getEndMonth() { return endMonth; }

    public String getEndYear() { return endYear; }

    public List<String> getDesc() {
        List<String> lines = new LinkedList<>();
        for (String line : desc)
            lines.add(line);
        return lines;

    }
}
```

//Experience.java

```
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.awt.Color;
import java.awt.*;

import javax.swing.BorderFactory;
import javax.swing.DefaultListModel;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JComboBox;
import javax.swing.JLabel;
import javax.swing.JList;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
```

```
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.ScrollPaneConstants;

import java.util.Arrays;
import java.util.LinkedList;
import java.util.List;
import java.util.stream.Collectors;

public class Experience extends JPanel implements ActionListener, ItemListener {
   private JLabel lblExperience, lblJobTitle, lblCity, lblEmployer, lblStartDate,
lblEndDate, lblDescription;
   private JTextField txtJobTitle, txtCity, txtEmployer;
   private JComboBox<String> cboStartMonth, cboEndMonth;
   private JComboBox<Integer> cboStartYear, cboEndYear;
   private JTextArea txtDescription;
   private JCheckBox cbHasExperience;
   private JScrollPane scroll;
   private JButton btnAddUpdate, btnEdit, btnDelete;


   int value;

   boolean chkbox;

   private static final String FONTSTYLE = "Berlin Sans FB";
   List<ExpDetails> list = new LinkedList<ExpDetails>();

   public Experience() {

      String[] months = {
            "January", "Febuary", "March", "April", "May", "June", "July", "August",
"September", "October", "November",
            "December"
      };
      Integer[] years = new Integer[63];
      int sartYear = 1960;
      int endYear = 2022;

      for (int i = 0; i < 63; i++) {
         years[i] = sartYear;
         sartYear++;
         if (sartYear == endYear) {
            years[i] = endYear;
            break;
```

```
        }
    }

    cbHasExperience = new JCheckBox();
    cbHasExperience.setBounds(185, 13, 20, 20);
    cbHasExperience.setToolTipText("Tick the box if you have Work Experience,
otherwise leave it unticked.");

    lblExperience = new JLabel("Work Experience");
    lblExperience.setFont(new Font("Consolas", Font.PLAIN, 20));
    lblExperience.setBounds(20, 15, 180, 20);

    lblJobTitle = new JLabel("Job Title");
    lblJobTitle.setFont(new Font(FONTSTYLE, Font.PLAIN, 15));
    lblJobTitle.setBounds(53, 35, 75, 20);

    lblCity = new JLabel("City/Town");
    lblCity.setFont(new Font(FONTSTYLE, Font.PLAIN, 15));
    lblCity.setBounds(41, 65, 75, 20);

    lblEmployer = new JLabel("Employer");
    lblEmployer.setFont(new Font(FONTSTYLE, Font.PLAIN, 15));
    lblEmployer.setBounds(43, 95, 75, 20);

    lblStartDate = new JLabel("Start Date");
    lblStartDate.setFont(new Font(FONTSTYLE, Font.PLAIN, 14));
    lblStartDate.setBounds(40, 128, 75, 20);

    lblEndDate = new JLabel("End Date");
    lblEndDate.setFont(new Font(FONTSTYLE, Font.PLAIN, 14));
    lblEndDate.setBounds(245, 128, 75, 20);

    lblDescription = new JLabel("Description");
    lblDescription.setFont(new Font(FONTSTYLE, Font.PLAIN, 15));
    lblDescription.setBounds(40, 185, 75, 20);

    txtJobTitle = new JTextField();
    txtJobTitle.setBounds(120, 38, 150, 20);

    txtCity = new JTextField();
    txtCity.setBounds(120, 68, 150, 20);

    txtEmployer = new JTextField();
    txtEmployer.setBounds(120, 98, 150, 20);

    cboStartMonth = new JComboBox<>(months);
```

```
        cboStartMonth.setBounds(40, 153, 95, 20);

        cboStartYear = new JComboBox<>(years);
        cboStartYear.setBounds(140, 153, 95, 20);

        cboEndMonth = new JComboBox<>(months);
        cboEndMonth.setBounds(245, 153, 95, 20);

        cboEndYear = new JComboBox<>(years);
        cboEndYear.setBounds(345, 153, 95, 20);

        txtDescription = new JTextArea("", 0, 0);
        txtDescription.setBorder(BorderFactory.createLineBorder(Color.BLACK));
        txtDescription.setEnabled(false);

        scroll = new JScrollPane(txtDescription);

scroll.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_AL
WAYS);
        scroll.setBounds(40, 230, 400, 100);

        btnAddUpdate = new JButton("Add");
        btnAddUpdate.setBounds(370, 15, 80, 30);

btnAddUpdate.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));

        btnEdit = new JButton("Edit");
        btnEdit.setBounds(370, 55, 80, 30);
        btnEdit.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));

        btnDelete = new JButton("Delete");
        btnDelete.setBounds(370, 95, 80, 30);
        btnDelete.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));

        btnAddUpdate.addActionListener(this);
        btnEdit.addActionListener(this);
        btnDelete.addActionListener(this);

        add(btnAddUpdate).setEnabled(false);
        add(btnEdit).setEnabled(false);
        add(btnDelete).setEnabled(false);

        cbHasExperience.addItemListener(this);

        add(cbHasExperience);
```

```
        add(lblExperience);

        add(lblJobTitle);
        add(lblCity);
        add(lblEmployer);
        add(lblStartDate);
        add(lblEndDate);
        add(lblDescription);

        add(txtJobTitle).setEnabled(false);
        add(txtCity).setEnabled(false);

        add(txtEmployer).setEnabled(false);

        add(cboStartMonth).setEnabled(false);
        add(cboStartYear).setEnabled(false);
        add(cboEndMonth).setEnabled(false);
        add(cboEndYear).setEnabled(false);

        // add(txtDescription);
        add(scroll);
        setLayout(null);

    }

    public List<ExpDetails> getDetails() { return list; }

    public static String capitalizeFully(String str) {
        if (str == null || str.isEmpty()) {
            return str;
        }

        return Arrays.stream(str.split("\\s+")).map(t -> t.substring(0, 1).toUpperCase() +
t.substring(1).toLowerCase())
                .collect(Collectors.joining(" "));
    }

    public String getJobTitle() { return capitalizeFully(txtJobTitle.getText()); }

    public String getCityTown() { return capitalizeFully(txtCity.getText()); }

    public String getEmployer() { return capitalizeFully(txtEmployer.getText()); }

    public String getStartMonth() { return (String) cboStartMonth.getSelectedItem(); }

    public String getStartYear() { return (Integer) cboStartYear.getSelectedItem() + ""; }
```

```
    public String getEndMonth() { return (String) cboEndMonth.getSelectedItem(); }

    public String getEndYear() { return (Integer) cboEndYear.getSelectedItem() + ""; }

    public List<String> getDesc() {
        List<String> lines = new LinkedList<>();
        for (String line : txtDescription.getText().split("\\n"))
            lines.add(line);
        return lines;
    }



    public boolean nextPressed(boolean bool){
        return bool;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getSource().equals(btnAddUpdate)) {
            if (btnAddUpdate.getText().equalsIgnoreCase("Update")) {
                int i = 0;
                System.out.println("Updated!");
                btnAddUpdate.setText("Add");
                for (ExpDetails expDetails : list) {
                    i = list.indexOf(expDetails);
                    if (i == value) {
                        System.out.println("should return list:" + i);

                        list.set(i,
                                new ExpDetails(this.getJobTitle(), this.getCityTown(),
this.getEmployer(), this.getStartMonth(),
                                        this.getStartYear(), this.getEndMonth(), this.getEndYear(),
this.getDesc()));
                        System.out.println(expDetails.getJobTitle());

                        clearFields();
                    }
                }
                btnDelete.setEnabled(false);

            }
            else {
                    if (isFieldEmpty()) {
                    System.out.println("added");
```

```
                list.add(new ExpDetails(this.getJobTitle(), this.getCityTown(),
this.getEmployer(), this.getStartMonth(),
                this.getStartYear(), this.getEndMonth(), this.getEndYear(),
this.getDesc()));

                btnEdit.setEnabled(true);

                clearFields();
            }
          }

       }
       if (e.getSource().equals(btnEdit)) {
          DefaultListModel<String> t1 = new DefaultListModel<String>();

          JList<String> jl = new JList<String>(t1);

          String hold = "";
          try {
             for (ExpDetails pastJobs : list) {

                t1.addElement(pastJobs.getJobTitle());
             }

             int option = JOptionPane.showConfirmDialog(null, jl, "Work Experiences:
Edit", 0);
             if (option == JOptionPane.YES_OPTION) {
                btnDelete.setEnabled(true);
                value = jl.getSelectedIndex();

                for (ExpDetails expDetails : list) {
                   if (list.indexOf(expDetails) == value) {
                      setFields(expDetails.getJobTitle(), expDetails.getCityTown(),
expDetails.getEmployer(),
                            expDetails.getStartMonth(), expDetails.getStartYear(),
expDetails.getEndMonth(),
                            expDetails.getEndYear(), hold);

                      for (String desc : expDetails.getDesc()) {
                         hold += desc + "\n";
                         setFields(expDetails.getJobTitle(), expDetails.getCityTown(),
expDetails.getEmployer(),
                               expDetails.getStartMonth(), expDetails.getStartYear(),
expDetails.getEndMonth(),
                               expDetails.getEndYear(), hold);
                      }
```

```
                }

            }
            if (btnAddUpdate.getText().equalsIgnoreCase("Add"))
                btnAddUpdate.setText("Update");

        }

    }
    catch (NullPointerException nullP) {
        nullP.printStackTrace();
    }
}

if (e.getSource().equals(btnDelete)) {
    for (ExpDetails expDetails : list) {
        if (list.indexOf(expDetails) == value) {
            System.out.println("should return");
            list.remove(expDetails);

        }
    }
    clearFields();
    btnDelete.setEnabled(false);
    btnAddUpdate.setText("Add");

}
}

@Override
public void itemStateChanged(ItemEvent e) {
    if (e.getStateChange() == 1) {
        enable();
        chkbox = true;
    }
    else {
        chkbox = false;
        disable();
        clearFields();
    }
}



public boolean getChkbox(){
```

```
        return chkbox;
    }

    public boolean isFieldEmpty() {
        if (!this.getJobTitle().isEmpty() && !this.getCityTown().isEmpty() &&
!this.getEmployer().isEmpty() && !txtDescription.getText().isEmpty())

            return true;
        else
            return false;
    }

    public boolean wrkValidation() {
        if (!list.isEmpty())
            return true;

        else {
            return false;
        }
    }

    public void enable() {
        btnAddUpdate.setEnabled(true);
        // btnEdit.setEnabled(true);
        // btnDelete.setEnabled(true);
        txtJobTitle.setEnabled(true);
        txtCity.setEnabled(true);
        txtEmployer.setEnabled(true);
        cboStartMonth.setEnabled(true);
        cboStartYear.setEnabled(true);
        cboEndMonth.setEnabled(true);
        cboEndYear.setEnabled(true);
        txtDescription.setEnabled(true);

    }

    public void disable() {
        btnAddUpdate.setEnabled(false);
        btnEdit.setEnabled(false);
        btnDelete.setEnabled(false);
        txtJobTitle.setEnabled(false);
        txtCity.setEnabled(false);
        txtEmployer.setEnabled(false);
        cboStartMonth.setEnabled(false);
        cboStartYear.setEnabled(false);
        cboEndMonth.setEnabled(false);
```

```
        cboEndYear.setEnabled(false);
        txtDescription.setEnabled(false);

    }

    public void setFields(String job, String city, String employer, String startMOnth,
String startYear, String endMonth,
            String endYear, String desc) {

        txtJobTitle.setText(job);
        txtCity.setText(city);
        txtEmployer.setText(employer);
        cboStartMonth.setSelectedItem(startMOnth);
        cboStartYear.setSelectedItem(startYear);
        cboEndMonth.setSelectedItem(endMonth);
        cboEndYear.setSelectedItem(endYear);
        txtDescription.setText(desc);

    }

    public void clearFields() {
        txtJobTitle.setText("");
        txtCity.setText("");
        txtEmployer.setText("");
        cboStartMonth.setSelectedIndex(0);
        cboStartYear.setSelectedIndex(0);
        cboEndMonth.setSelectedIndex(0);
        cboEndYear.setSelectedIndex(0);
        txtDescription.setText("");
    }

}
```

//FileDetails.java

```
public class FileDetails {
    String name, path;
    public FileDetails(String name, String path){
        this.name = name;
        this.path = path;
    }

    public String getFile(){
        return name;
    }

    public String getPath(){
        return path;
```

```
    }
}
```

//Menu.java

```
import java.io.File;
import java.io.IOException;
import java.util.LinkedList;
import java.util.List;


import javax.swing.DefaultListModel;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JList;
import javax.swing.JOptionPane;


import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.Desktop;


public class Menu extends JFrame implements ActionListener{
    private JComboBox<String> cbochoice;
    private JButton btnOk;
    String userHomeDir = System.getProperty("user.home");


    File folder = new File(userHomeDir + "\\Desktop\\CVs\\");
    File[] listOfFiles = folder.listFiles();


    String choice[] = {
            "Show History", "Create Resume", "Exit"
    };


    public Menu(){


        cbochoice = new JComboBox<String>(choice);
        cbochoice.setBounds(70, 50, 130, 20);
```

```
    btnOk = new JButton("OK");
    btnOk.setBounds(95, 100, 70, 20);


    btnOk.addActionListener(this);


    add(cbochoice);
    add(btnOk);



    setTitle("CV Menu");

    setSize(300, 200);
    setLocationRelativeTo(null);
    setLayout(null);
    setVisible(true);
    setDefaultCloseOperation(EXIT_ON_CLOSE);


}



@Override
public void actionPerformed(ActionEvent e) { // TODO Auto-generated method stub
    int i = cbochoice.getSelectedIndex();


    if(i == 0){
        showHistory();
    }


    if(i == 1){
        try {
            new CVG();
            setVisible(false);
            if (folder.mkdir()) {
                System.out.println("Folder does not exist. Creating a new one");
            }
            else {
                System.out.println("Folder Alreaedy exist!");
```

```
            }
        }
        catch (IOException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
    }
    if(i == 2){
        System.exit(0);
    }
}

private void showHistory() {
    try {
        List<FileDetails> list = new LinkedList<FileDetails>();


        boolean isOpen = false;


        int value = 0;


        DefaultListModel<String> t1 = new DefaultListModel<String>();
        JList<String> jl = new JList<String>(t1);


        for (File file : listOfFiles) {


            if (file.isFile()) {
                list.add(new FileDetails(file.getName(), file.getAbsolutePath()));
            }


        }


        for (FileDetails cv : list) {
            t1.addElement(cv.getFile());


        }
        int option = JOptionPane.showConfirmDialog(null, jl, "Show History", 0);
        if (option == JOptionPane.YES_OPTION) {
```

```
            value = jl.getSelectedIndex();
            for (FileDetails cv : list) {
                if (list.indexOf(cv) == value) {
                    if (Desktop.isDesktopSupported()) {
                        File myFile = new File(cv.getPath());


                        Desktop.getDesktop().open(myFile);


                    }
                }
            }
        }
    } catch (Exception e) {
        // TODO: handle exception
    }


  }
}
```

//Personal.java

```
import java.awt.Font;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.util.Arrays;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import java.util.stream.Collectors;

import javax.swing.*;

public class Personal extends JPanel implements KeyListener {
   private JTextField txtFirstName, txtLastName, txtEmail, txtPhoneNum;
   private JLabel lblPersonal, lblFirstName, lblLastName, lblEmail, lblPhoneNum;

   private static final String FONTSTYLE = "Berlin Sans FB";

   public Personal() {
      lblPersonal = new JLabel("Personal Details");
      lblPersonal.setFont(new Font("Consolas", Font.PLAIN, 20));
      lblPersonal.setBounds(20, 15, 180, 20);

      lblFirstName = new JLabel("First Name");
```

```
lblFirstName.setFont(new Font(FONTSTYLE, Font.PLAIN, 15));
lblFirstName.setBounds(40, 35, 75, 20);

lblLastName = new JLabel("Last Name");
lblLastName.setFont(new Font(FONTSTYLE, Font.PLAIN, 15));
lblLastName.setBounds(40, 65, 75, 20);

lblEmail = new JLabel("Email");
lblEmail.setFont(new Font(FONTSTYLE, Font.PLAIN, 15));
lblEmail.setBounds(74, 95, 40, 20);

lblPhoneNum = new JLabel("Phone Number");
lblPhoneNum.setFont(new Font(FONTSTYLE, Font.PLAIN, 15));
lblPhoneNum.setBounds(15, 125, 105, 20);

txtFirstName = new JTextField();
txtFirstName.setBounds(120, 38, 150, 20);

txtLastName = new JTextField();
txtLastName.setBounds(120, 68, 150, 20);

txtEmail = new JTextField();
txtEmail.setBounds(120, 98, 150, 20);

txtPhoneNum = new JTextField();
txtPhoneNum.setBounds(120, 128, 150, 20);

// Add Labels into this panel
add(lblPersonal);
add(lblFirstName);
add(lblLastName);
add(lblEmail);
add(lblPhoneNum);

// Add Text Fields into this Panel
add(txtFirstName);
add(txtLastName);
add(txtEmail);
add(txtPhoneNum);

// Add Key listners to this panel
txtFirstName.addKeyListener(this);
txtLastName.addKeyListener(this);
txtEmail.addKeyListener(this);
txtPhoneNum.addKeyListener(this);
```

```
        setLayout(null);

    }

    public void errorMessage(String txt) { JOptionPane.showMessageDialog(null, txt); }

    private static String capitalizeFully(String str) {
        if (str == null || str.isEmpty()) {
            return str;
        }

        return Arrays.stream(str.split("\\s+")).map(t -> t.substring(0, 1).toUpperCase() +
t.substring(1).toLowerCase())
                .collect(Collectors.joining(" "));
    }

    public String getNames() { return getFirstName() + " " + getLastName(); }

    private String getFirstName() { return capitalizeFully(txtFirstName.getText()); }

    private String getLastName() { return capitalizeFully(txtLastName.getText()); }

    public String getEmail() {
        return capitalizeFully(txtEmail.getText());

    }

    public String getPhoneNum() { return txtPhoneNum.getText(); }

    public boolean emailValildation() {
        final String EMAIL_PATTERN = "^[_A-Za-z0-9-\\+]+(\\.[_A-Za-z0-9-]+)*@"
                + "[A-Za-z0-9-]+(\\.[A-Za-z0-9]+)*(\\.[A-Za-z]{2,})$";

        Pattern pattern = Pattern.compile(EMAIL_PATTERN);
        Matcher matchEmail = pattern.matcher(getEmail());

        if (this.getEmail().isEmpty()) {
            errorMessage("Email Field is empty");

            return true;
        }

        else if (!matchEmail.matches()) {
            errorMessage("wrong email format");
            return true;
```

```
    }
    else {

        return false;
    }

}

public boolean pValidation() {
    if (this.getFirstName().isEmpty()) {
        errorMessage("First name Field is empty");
        return false;
    }
    else if (this.getLastName().isEmpty()) {
        errorMessage("Last name Field is empty");
        return false;
    }
    else if (emailValiIdation()) {

        return false;
    }
    else if (this.getPhoneNum().isEmpty()) {
        errorMessage("Phone Number Field is empty");
        return false;
    }
    else {
        return true;

    }

}



@Override
public void keyTyped(KeyEvent e) {
    if (e.getSource().equals(txtPhoneNum)) {
        if (!(Character.isDigit(e.getKeyChar()))) {
            e.consume();
        }
    }



}
```

```
    @Override
    public void keyPressed(KeyEvent e) {
        // TODO Auto-generated method stub

    }

    @Override
    public void keyReleased(KeyEvent e) {
        // TODO Auto-generated method stub
    }

}
```

//SkillDetails.java

```
public class SkillDetails {
    private String skill, skillLvel ;


    public SkillDetails(String skill, String skillLevel) {
        this.skill = skill;
        this.skillLvel = skillLevel;

    }

    public String getSkill() { return skill; }

    public String getLevel() { return skillLvel; }




}
```

//Skills.java

```
import java.awt.Font;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JLabel;
import javax.swing.JList;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.DefaultListModel;
import javax.swing.JButton;
import javax.swing.JComboBox;
```

```
import java.util.Arrays;
import java.util.LinkedList;
import java.util.List;
import java.util.stream.Collectors;

public class Skills extends JPanel implements ActionListener {
    private JLabel lblSkillsTTL, lblSkill;
    private JTextField txtSkill;
    private JButton btnAddUpdate, btnEdit, btnDelete;

    private JComboBox<String> cboLevel;
    private static final String FONTSTYLE = "Berlin Sans FB";

    List<SkillDetails> list = new LinkedList<SkillDetails>();

    int value;

    public Skills() {
        String[] level = {
                "Expert", "Experience", "Skillful", "Intermediate", "Begineer"
        };

        lblSkillsTTL = new JLabel("Skills");
        lblSkillsTTL.setFont(new Font("Consolas", Font.PLAIN, 20));
        lblSkillsTTL.setBounds(20, 15, 180, 20);

        lblSkill = new JLabel("Skill");
        lblSkill.setFont(new Font(FONTSTYLE, Font.PLAIN, 15));
        lblSkill.setBounds(40, 35, 75, 20);

        txtSkill = new JTextField();
        txtSkill.setBounds(70, 35, 150, 20);

        cboLevel = new JComboBox<>(level);
        cboLevel.setBounds(240, 35, 100, 20);

        btnAddUpdate = new JButton("Add");
        btnAddUpdate.setBounds(370, 15, 80, 30);

btnAddUpdate.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));

        btnEdit = new JButton("Edit");
        btnEdit.setBounds(370, 55, 80, 30);
        btnEdit.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
```

```
        btnDelete = new JButton("Delete");
        btnDelete.setBounds(370, 95, 80, 30);
        btnDelete.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));

        btnAddUpdate.addActionListener(this);
        btnEdit.addActionListener(this);
        btnDelete.addActionListener(this);

        add(btnAddUpdate);
        add(btnEdit).setEnabled(false);
        add(btnDelete).setEnabled(false);

        add(lblSkillsTTL);
        add(lblSkill);
        add(txtSkill);

        add(cboLevel);

        setLayout(null);
    }

    public List<SkillDetails> getDetails() { return list; }


    public static String capitalizeFully(String str) {
        if (str == null || str.isEmpty()) {
            return str;
        }

        return Arrays.stream(str.split("\\s+")).map(t -> t.substring(0, 1).toUpperCase() +
t.substring(1).toLowerCase())
                .collect(Collectors.joining(" "));
    }


    public String getSkill() { return capitalizeFully(txtSkill.getText()); }

    public String getLevel() { return (String) cboLevel.getSelectedItem(); }

    @Override
    public void actionPerformed(ActionEvent e) { // TODO Auto-generated method stub
        if (e.getSource().equals(btnAddUpdate)) {
            if (btnAddUpdate.getText().equalsIgnoreCase("Update")) {
                int i = 0;
                System.out.println("Updated!");
                btnAddUpdate.setText("Add");
```

```
            for (SkillDetails skillDetails : list) {
                i = list.indexOf(skillDetails);
                if (i == value) {
                    System.out.println("should return list:" + i);

                    list.set(i, new SkillDetails(this.getSkill(), this.getLevel()));

                    txtSkill.setText("");
                }
            }
            btnDelete.setEnabled(false);

        }
        else {
            if (isFieldEmpty()) {
                System.out.println("added");
                list.add(new SkillDetails(this.getSkill(), this.getLevel()));

                btnEdit.setEnabled(true);

                txtSkill.setText("");

            }
        }

    }

    if (e.getSource().equals(btnEdit)) {
        DefaultListModel<String> t1 = new DefaultListModel<String>();

        JList<String> jl = new JList<String>(t1);

        try {
            for (SkillDetails skill : list) {

                t1.addElement(skill.getSkill());
            }

            int option = JOptionPane.showConfirmDialog(null, jl, "Skill: Edit", 0);
            if (option == JOptionPane.YES_OPTION) {
                btnDelete.setEnabled(true);
                value = jl.getSelectedIndex();

                for (SkillDetails skillDetails : list) {
                    if (list.indexOf(skillDetails) == value) {
                        setFields(skillDetails.getSkill(), skillDetails.getLevel());
```

```
                }
                if (btnAddUpdate.getText().equalsIgnoreCase("Add"))
                    btnAddUpdate.setText("Update");

            }

        }
    }
    catch (NullPointerException nullP) {
        nullP.printStackTrace();
    }

}

if (e.getSource().equals(btnDelete)) {
    for (SkillDetails skillDetails : list) {
        if (list.indexOf(skillDetails) == value) {
            System.out.println("should return");
            list.remove(skillDetails);

        }
    }
    txtSkill.setText("");

    btnDelete.setEnabled(false);
    btnAddUpdate.setText("Add");

}
}

public boolean isFieldEmpty() {
    if (!this.getSkill().isEmpty() && !this.getLevel().isEmpty())

        return true;
    else
        return false;
}

public boolean skillValidation() {
    if (!list.isEmpty())
        return true;

    else {
        return false;
    }
```

```
    }

    public void setFields(String skill, String level) {

        txtSkill.setText(skill);
        cboLevel.setSelectedItem(level);

    }

}
```

//Summary.java

```
import javax.swing.BorderFactory;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.ScrollPaneConstants;

import java.awt.Font;
import java.awt.Color;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

import java.util.ArrayList;
import java.util.List;

public class Summary extends JPanel implements KeyListener {
    private JLabel lblSummaryTtl;
    private JTextArea txtSummaryDesc;
    private JScrollPane scroll;


    public Summary() {

        lblSummaryTtl = new JLabel("Professional Summary");
        lblSummaryTtl.setFont(new Font("Consolas", Font.PLAIN, 20));
        lblSummaryTtl.setBounds(20, 15, 230, 20);

        txtSummaryDesc = new JTextArea("",0,0);
        // txtSummaryDesc.setBounds(40, 70, 400, 100);
        txtSummaryDesc.setBorder(BorderFactory.createLineBorder(Color.BLACK));
        // txtSummaryDesc.setLineWrap(true);
        // txtSummaryDesc.setWrapStyleWord(true);

        scroll = new JScrollPane(txtSummaryDesc);
```

```
scroll.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_AL
WAYS);
        scroll.setBounds(40, 70, 400, 100);

        txtSummaryDesc.addKeyListener(null);
        add(lblSummaryTtl);
        add(scroll);

        setLayout(null);
    }

    public List<String> getSummaryDesc(){
            List<String> lines = new ArrayList<>();
        for(String line : txtSummaryDesc.getText().split("\\n"))
            lines.add(line);
        return lines;
    }



    @Override
    public void keyTyped(KeyEvent e) { // TODO Auto-generated method stub

     }

    @Override
    public void keyPressed(KeyEvent e) { // TODO Auto-generated method stub
     }

    @Override
    public void keyReleased(KeyEvent e) { // TODO Auto-generated method stub
     }

}
```