islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CS4001NI Programming

Assessment Weightage & Type

30% Individual Coursework

Year and Semester

2019-20 Autumn

Student Name: Bijay Bharati

Group: L1C4

London Met ID: 19030824

College ID: NP01CP4A190041

Assignment Due Date: 5th June 2020

Assignment Submission Date: 1st June 2020

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

Contents

Figures

Tables

1 Introduction

This project deals with the implementation of a graphical user interface (GUI) for the previous coursework to hire full time and part time staff. A form is made where the user can enter all the necessary data and there are buttons to add vacancy, appoint staff, terminate staff and display the details for the job and the hired employee. BlueJ was used as IDE for the project. There is a class INGNepal which has GUI.

Unlike the last time, exception handling is also done. Concepts of typecasting are also used. This course work can be considered as an upgrade for the previous one.

The course work/ project demonstrates how a GUI can drastically improve the usability of a program and make a program or an application more presentable and less daunting to users.

2 Class Diagram

Figure 1: INGNepal Class Diagram

3 Pseudocode

Pseudocodes for different methods in INGNepal are listed below:

3.1 Pseudocode for main method

```
FUNCTION main (String [] args)
        INGNepal form = new INGNepal ();
        form.StaffHireForm ();
END FUNCTION
```

3.2 Pseudocode for StaffHireForm method

```
FUNCTION StaffHireForm()
        frm = new JFrame ("Staff Hire");
        CALL frm.setSize (755, 535);
        CALL frm.setLayout (null);
        CALL frm.setVisible (true);
        CALL frm.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
        DO
                lblTitleFT = new JLabel (“Vacancy Full Time”);
                setBounds (295,10,300,20);
                setFont ( Helvetica, plain, 18);
                frm.add (lblTitleFT);
        END DO
        DO
                lblVacancyNumberFT = new JLabel (“Vacancy Number”);
                setBounds (10, 40, 100, 20);
                setFont ( Helvetica, plain, 12);
                frm.add (lblVacancyNumberFT);
        END DO
```

```
DO
        txtVacancyNumberFT = new JTextField();
        setBounds(115, 40, 150, 20);
        frm.add(txtVacancyNumberFT);
END DO
DO
        lblDesignationFT = new JLabel("Designation:");
        setBounds(270, 40, 100, 20);
        setFont ( Helvetica, plain, 12;
        frm.add (lblDesignationFT);
END DO
DO
        txtDesignationFT = new JTextField();
        setBounds(360, 40, 150, 20);
        frm.add(txtDesignationFT);
END DO
DO
        lblJobTypeFT = new JLabel("Job Type:");
        setBounds(515, 40, 100, 20);
        setFont ( Helvetica, plain, 12);
        frm.add(lblJobTypeFT);
END DO
DO
        String jobTypeFT[] = { "Temporary", "Permanent" };
        cmbJobTypeFT = new JComboBox (jobTypeFT);
        setBounds(575, 40, 150, 20);
        frm.add(cmbJobTypeFT);
END DO
```

```
DO
        lblSalaryFT = new JLabel("Salary:");
        setBounds(10, 75, 100, 20);
        setFont ( Helvetica, plain, 12;
        frm.add(lblSalaryFT);
END DO
DO
        txtSalaryFT = new JTextField();
        setBounds(115, 75, 150, 20);
        frm.add(txtSalaryFT);
END DO
DO
        lblWorkingHoursFT = new JLabel("Working Hours:");
        setBounds(270, 75, 100, 20);
        setFont ( Helvetica, plain, 12);
        frm.add(lblWorkingHoursFT);
END DO
DO
        txtWorkingHoursFT = new JTextField();
        setBounds (360, 75, 150, 20);
        frm.add(txtWorkingHoursFT);
END DO
DO
        btnAddVacancyFT = new JButton("Add Vacancy");
        setBounds(515, 70, 210, 25);
        frm.add(btnAddVacancyFT);
        btnAddVacancyFT.addActionListener(this);
END DO
```

```
DO
        lblTitle2FT = new JLabel (“Employee Full Time”);
        setBounds (295,100,300,20);
        setFont ( Helvetica, plain, 18);
        frm.add (lblTitlePT);
END DO
DO
        lblVacancyNumber2FT = new JLabel (“Vacancy Number”);
        setBounds (10, 130, 100, 20);
        setFont ( Helvetica, plain, 12);
        frm.add (lblVacancyNumber2FT);
END DO
DO
        txtVacancyNumber2FT = new JTextField();
        setBounds(115, 130, 150, 20);
        frm.add(txtVacancyNumber2FT);
END DO
DO
        lblStaffNameFT = new JLabel("Staff Name:");
        setBounds (270, 130, 100, 20);
        setFont ( Helvetica, plain, 12);
        frm.add (lblStaffNameFT);
END DO
DO
        txtStaffNameFT = new JTextField();
        setBounds(360, 130, 150, 20);
        frm.add(txtStaffNameFT);
END DO
```

```
DO
        lblJoiningDateFT = new JLabel("Joined on:");
        setBounds (515, 130, 100, 20);
        setFont ( Helvetica, plain, 12);
        frm.add (lblJoiningDateFT);
END DO
DO
        txtJoiningDateFT = new JTextField();
        setBounds(575, 130, 150, 20);
        frm.add(txtJoiningDateFT);
END DO
DO
        lblQualificationFT = new JLabel("Qualification:");
        setBounds (10, 165, 100, 20);
        setFont ( Helvetica, plain, 12);
        frm.add (lblQualificationFT);
END DO
DO
        txtQualificationFT= new JTextField();
        setBounds(115, 165, 150, 20);
        frm.add(txtQualificationFT);
END DO
DO
        lblAppointedByFT = new JLabel("Appointed By:");
        setBounds (270, 165, 100, 20);
        setFont ( Helvetica, plain, 12);
        frm.add (lblAppointedByFT);
END DO
```

```
DO
        txtAppointedByFT = new JTextField();
        setBounds(360, 165, 150, 20);
        frm.add(txtAppointedByFT);
END DO
DO
        btnAppointStaffFT = new JButton("Appoint Staff");
        setBounds(515, 160, 210, 25);
        frm.add(btnAppointStaffFT);
        btnAppointStaffFT.addActionListener(this);
END DO
DO
        lblTitlePT = new JLabel (“Vacancy Part Time”);
        setBounds (295,10,300,20);
        setFont ( Helvetica, plain, 18);
        frm.add (lblTitlePT);
END DO
DO
        lblVacancyNumberPT = new JLabel (“Vacancy Number”);
        setBounds (10, 220, 100, 20);
        setFont ( Helvetica, plain, 12);
        frm.add (lblVacancyNumberPT);
END DO
```

```
DO
        txtVacancyNumberPT = new JTextField();
        setBounds(115, 220, 150, 20);
        frm.add(txtVacancyNumberPT);
END DO
DO
        lblDesignationPT = new JLabel("Designation:");
        setBounds(270, 220, 100, 20);
        setFont ( Helvetica, plain, 12;
        frm.add (lblDesignationPT);
END DO
DO
        txtDesignationPT = new JTextField();
        setBounds(360, 220, 150, 20);
        frm.add(txtDesignationPT);
END DO
DO
        lblJobTypePT = new JLabel("Job Type:");
        setBounds(515, 220, 100, 20);
        setFont ( Helvetica, plain, 12);
        frm.add(lblJobTypePT);
END DO
DO
        String jobTypePT[] = { "Temporary", "Permanent" };
        cmbJobTypePT = new JComboBox (jobTypePT);
        setBounds(575, 220, 150, 20);
        frm.add(cmbJobTypePT);
END DO
```

```
DO
        blWorkingHoursPT = new JLabel("Working Hours:");
        setBounds(10, 255, 100, 20);
        setFont ( Helvetica, plain, 12;
        frm.add(blWorkingHoursPT);
END DO
DO
        txtWorkingHoursPT = new JTextField();
        setBounds(115, 255, 150, 20);
        frm.add(txtWorkingHoursPT);
END DO
DO
        lblWagePerHrPT = new JLabel("Wage Per Hour::");
        setBounds(270, 255, 100, 20);
        setFont ( Helvetica, plain, 12);
        frm.add(lblWagePerHrPT);
END DO
DO
        txtWagePerHrPT = new JTextField();
        setBounds (360, 255, 150, 20);
        frm.add(txtWagePerHrPT);
END DO
DO
        lblShiftPT = new JLabel("Shift:");
        setBounds(515, 255, 100, 20);
        setFont ( Helvetica, plain, 12);
        frm.add(lblShiftPT);
END DO
```

```
DO
        String shiftPT[] = { "Morning", "Day", "Evening" };
        cmbShiftPT = new JComboBox(shiftPT);
        setBounds(575, 255, 150, 20);
        frm.add(cmbShiftPT);
END DO
DO
        btnAddVacancyPT= new JButton("Add Vacancy");
        setBounds(270, 290, 210, 25);
        frm.add(btnAddVacancyPT);
        btnAddVacancyPT.addActionListener(this);
END DO
DO
        lblTitle2PT = new JLabel (“Employee Part Time”);
        setBounds (295,100,300,20);
        setFont ( Helvetica, plain, 18);
        frm.add (lblTitle2PT);
END DO
DO
        lblVacancyNumber2PT = new JLabel (“Vacancy Number”);
        setBounds (10, 360, 100, 20);
        setFont ( Helvetica, plain, 12);
        frm.add (lblVacancyNumber2PT);
END DO
```

```
DO
        txtVacancyNumber2PT = new JTextField();
        setBounds(115, 360, 150, 20);
        frm.add(txtVacancyNumber2PT);
END DO
DO
        lblStaffNamePT = new JLabel("Staff Name:");
        setBounds (270, 360, 100, 20);
        setFont ( Helvetica, plain, 12);
        frm.add (lblStaffNamePT);
END DO
DO
        txtStaffNamePT = new JTextField();
        setBounds(360, 360, 150, 20);
        frm.add(txtStaffNamePT);
END DO
DO
        lblJoiningDatePT = new JLabel("Joined on:");
        setBounds (515, 360, 100, 20);
        setFont ( Helvetica, plain, 12);
        frm.add (lblJoiningDatePT);
END DO
DO
        txtJoiningDatePT = new JTextField();
        setBounds(575, 360, 150, 20);
        frm.add(txtJoiningDatePT);
END DO
```

```
DO
        lblQualificationPT = new JLabel("Qualification:");
        setBounds (10, 395, 100, 20);
        setFont ( Helvetica, plain, 12);
        frm.add (lblQualificationPT);
END DO
DO
        txtQualificationPT= new JTextField();
        setBounds(115, 395, 150, 20);
        frm.add(txtQualificationPT);
END DO
DO
        lblAppointedByPT = new JLabel("Appointed By:");
        setBounds (270, 395, 100, 20);
        setFont ( Helvetica, plain, 12);
        frm.add (lblAppointedByPT);
END DO
DO
        txtAppointedByPT = new JTextField();
        setBounds(360, 395, 150, 20);
        frm.add(txtAppointedByPT);
END DO
DO
        btnAppointStaffPT = new JButton("Appoint Staff");
        setBounds(515, 390, 210, 25);
        frm.add(btnAppointStaffPT);
        btnAppointStaffPT.addActionListener(this);
END DO
```

```
DO
        btnDisplay = new JButton("Display");
        setBounds(25, 430, 210, 50);
        frm.add(btnDisplay);
        btnDisplay.addActionListener(this);
END DO
DO
        btnClear = new JButton("Clear");
        setBounds(275, 430, 210, 50);
        frm.add(btnClear);
        btnClear.addActionListener(this);
END DO
DO
        lblTerminationVacancy = new JLabel("TerminationVac:")
        setBounds(515, 430, 100, 20);
        setFont ( Helvetica, plain, 12);
        frm.add(lblTerminationVacancy);
END DO
DO
        txtTerminationVacancy = new JTextField();
        setBounds(610, 430, 115, 20);
        frm.add(txtTerminationVacancy);
END DO
```

```
        DO
                btnTerminate = new JButton("Terminate");
                setBounds(515, 455, 210, 25);
                frm.add(btnTerminate);
                btnTerminate.addActionListener(this);
        END DO
END FUNCTION
```

3.3 Pseudocode for actionPerformed method

```
FUNCTION actionPerformed (ActionEvent click)
        IF (click == btnClear)
                txtVacancyNumberFT.setText("");
                txtDesignationFT.setText("");
                txtSalaryFT.setText("");
                txtWorkingHoursFT.setText("");
                txtVacancyNumber2FT.setText("");
                txtStaffNameFT.setText("");
                txtQualificationFT.setText("");
                txtAppointedByFT.setText("");
                txtJoiningDateFT.setText("");
                txtVacancyNumberPT.setText("");
                txtDesignationPT.setText("");
                txtWorkingHoursPT.setText("");
                txtWagePerHrPT.setText("");
                txtVacancyNumber2PT.setText("");
                txtStaffNamePT.setText("");
                txtQualificationPT.setText("");
                txtAppointedByPT.setText("");
                txtJoiningDatePT.setText("");
                txtTerminationVacancy.setText("");
```

```
            cmbJobTypeFT.setSelectedIndex(0);
            cmbJobTypePT.setSelectedIndex(0);
            cmbShiftPT.setSelectedIndex(0);
        END IF
        IF (click == btnAddVacancyFT)
            TRY
                vacancyNumber =Integer.parseInt(txtVacancyNumberFT.getText());
                designation = txtDesignationFT.getText();
                jobType = (cmbJobTypeFT.getSelectedItem()).toString();
                salary = Double.parseDouble(txtSalaryFT.getText());
                workingHour = Double.parseDouble(txtWorkingHoursFT.getText());
                vacancyInUse = false;
                IF(designation==“”)
                    showMessageDialog (Make sure to input proper values in all
                    the fields. )
                END IF
                ELSE
                    FOR(StaffHire obj: staffList)
                        IF(obj.getVacancyNumber==vacancyNumber)
                            vacancyInUse=true;
                            break;
                        END IF
                    END FOR
                    IF (vacancyInUse ==false)
                        FullTimeStaffHire obj = new FullTimeStaffHire
                        (vacancyNumber, designation, jobType,
                        salary,workingHour )
                        staffList.add(obj);
```

```
                showMessageDialog ( "Vacancy " + vacancyNumber
                + " added Total: " + staffList.size());
            END IF
            ELSE
                showMessageDialog("Vacancy " + vacancyNumber +
                " already in use Total: " + staffList.size())
            END ELSE
        END ELSE
    END TRY
    CATCH (Exception exp)
        showMessageDialog("Make sure to input proper values in all the
        fields.");
    END CATCH
END IF
IF (click == btnAppointStaffFT)
    TRY
        vacancyNumber =Integer.parseInt(txtVacancyNumberFT.getText());
        staffName = txtStaffNameFT.getText();
        joinedOn = txtJoiningDateFT.getText();
        qualification = txtQualificationFT.getText();
        appointedBy = txtAppointedByFT.getText();
        joined = false;
        vacancyInUse = false;;
        IF (staffName==”” OR joinedOn ==”” OR qualification==”” OR
        appointedBy==”” )
            showMessageDialog (Make sure to input proper values in all
            the fields. )
        END IF
```

```
                ELSE
                        FOR(StaffHire obj: staffList)
                                IF (obj.getVacancyNumber() == vacancyNumber)
                                        vacancyInUse = true;
                                        IF (obj instanceof FullTimeStaffHire)
                                                FullTimeStaffHire call =
                                                (FullTimeStaffHire) obj;
                                                IF (call.getJoined(joined) == true)
                                                        showMessageDialog("Staff has
                                                        been already hired");
                                                END IF
                                                ELSE
                                                        call.hireFullTimeStaff()
                                                        showMessageDialog(f"Staff
                                                         successfully appointed.")
                                                        break;
                                                END ELSE
                                        END IF
                                        ELSE
                                                showMessageDialog("Vacancy " +
                                                vacancyNumber " belongs to PART
                                                TIME, use a vacancy that belongs to
                                                FULL TIME");)
                                        END ELSE
                                END IF
                        END FOR
                IF(!vacancyInUse)
                        showMessageDialog("The vacancy you provided does not
                        exist, try again");
                END IF
        END ELSE
```

```
        END TRY
        CATCH (Exception exp1)
                showMessageDialog( "Make sure to input proper values in all the fields.");
        END CATCH
END IF
        END IF
        IF (click == btnAddVacancyPT)
                TRY
                        vacancyNumber=Integer.parseInt(txtVacancyNumberPT.getText());
                        designation = txtDesignationPT.getText();
                        jobType = (cmbJobTypePT.getSelectedItem()).toString();
                        wagePerHr = Double.parseDouble(txt wagePerHr.getText());
                        workingHour = Double.parseDouble(txtWorkingHoursFT.getText());
                        shift = (cmbShiftPT.getSelectedItem()).toString();
                        vacancyInUse = false;
                        IF(designation==””)
                                showMessageDialog (Make sure to input proper values in all
                                the fields. )
                        END IF
                        ELSE
                                FOR(StaffHire obj: staffList)
                                        IF(obj.getVacancyNumber==vacancyNumber)
                                                vacancyInUse=true;
                                                break;
                                        END IF
                                END FOR
                                IF (vacancyInUse ==false)
```

```
                    PartTimeStaffHire obj = new PartTimeStaffHire
                    (vacancyNumber, designation, jobType, workingHour,
                    wagePerHr, shift)

                    staffList.add(obj)

                    showMessageDialog ( "Vacancy " + vacancyNumber
                    + " added Total: " + staffList.size());

                END IF

                ELSE

                    showMessageDialog("Vacancy " + vacancyNumber +
                    " already in use Total: " + staffList.size())

                END ELSE

            END ELSE

        END TRY

        CATCH (Exception exp)

            showMessageDialog("Make sure to input proper values in all the
            fields.");

        END CATCH

    END IF

    IF (click == btnAppointStaffPT)

        TRY

            vacancyNumber=Integer.parseInt(txtVacancyNumberPT.getText());

            staffName = txtStaffNamePT.getText();

            joinedOn = txtJoiningDatePT.getText();

            qualification = txtQualificationPT.getText();

            appointedBy = txtAppointedByPT.getText();

            joined = false;

            vacancyInUse = false;;

            IF (staffName==”” OR joinedOn ==”” OR qualification==”” OR
            appointedBy==”” )

                showMessageDialog (Make sure to input proper values in all
                the fields. )
```

```
END IF
ELSE
    FOR(StaffHire obj: staffList)
        IF (obj.getVacancyNumber() == vacancyNumber)
            vacancyInUse = true;
            IF (obj instanceof PartTimeStaffHire)
                PartTimeStaffHire call =
                (PartTimeStaffHire) obj;
                IF (call.getJoined(joined) == true)
                    showMessageDialog("Staff has
                    been already hired");
                END IF
                ELSE
                    call.hirePartTimeStaff()
                    showMessageDialog(f"Staff
                     successfully appointed.")
                    break;
                END ELSE
            END IF
            ELSE
                showMessageDialog("Vacancy " +
                vacancyNumber " belongs to FULL
                TIME, use a vacancy that belongs to
                PART TIME");)
            END ELSE
        END IF
    END FOR
IF(!vacancyInUse)
    showMessageDialog("The vacancy you provided does not
    exist, try again");
END IF
```

```
                END ELSE
        END TRY
        CATCH (Exception exp1)
                showMessageDialog( "Make sure to input proper values in all the fields.");
        END CATCH
END IF
IF (click == btnTerminate)
        TRY
                terminationVacancy = Integer.parseInt(txtTerminationVacancy.getText());
                 joined = true;
                terminated = false;
                 vacancyInUse = false;
                FOR (StaffHire obj : staffList)
                        IF (obj.getVacancyNumber() == terminationVacancy)
                                vacancyInUse = true;
                                IF (obj instanceof PartTimeStaffHire)
                                        PartTimeStaffHire call = (PartTimeStaffHire) obj;
                                IF(getTerminated=false AND getJoined=true)
                                        call.terminateStaff();
                                        showMessageDialog("Staff terminated");
                                END IF
                                ELSE
                                        showMessageDialog("No staff has been hired to
                                        Vacancy " + terminationVacancy + " or does not
                                        belong to FULL TIME or has been terminated ");
                                END ELSE
                                ELSE
                                        showMessageDialog("No staff has been hired to
                                        Vacancy " + terminationVacancy + " or does not
                                        belong to PART TIME or has been terminated ");
```

```
                        END ELSE
                    END IF
                END FOR
                IF (!vacancyInUse)
                    showMessageDialog ("vacancy you provided does not exist);
                END IF
            END TRY
            CATCH (Exception exp4)
                showMessageDialog( "Make sure to input proper values in all the fields.");
            END CATCH
        END IF
        IF (click == btnDisplay)
            IF (staffList.size() == 0)
                showMessageDialog ("No data to show");
            END IF
            ELSE
                FOR (StaffHire obj : staffList)
                    IF (obj instanceof FullTimeStaffHire)
                        FullTimeStaffHire call = (FullTimeStaffHire) obj;
                        call.displayInfo();
                    END IF
                    IF (obj instanceof PartTimeStaffHire)
                        PartTimeStaffHire call = (PartTimeStaffHire) obj;
                        call.displayInfo();END IF
                END FOR
            END ELSE
        END IF
        END FUNCTION
```

4 Methods

4.1 main (String [] args)

Main method is a static method. Main method is used to call StaffHireForm () method which is instance method. To call the method, an object is made and the method is called. The main method just helps us call the method to show us the GUI.

4.2 staffHireForm ()

This method contains all the text fields, drop down menus and buttons. The method contains GUI of the INGNepal. All the things that the user can interact with is in this method.

4.3 actionPerformed (ActionEvent click)

This is arguably the most important method for the functionality of INGNepal. This method contains action listener and defines the functions or each button. It tells the program what to do when add vacancy, appoint staff, terminate, display, clear button is pressed. Exception handling and typecasting is also performed.

When clear button is pressed, all the values are reset.

When button to add vacancy for full time of part time staff is pressed, first it is checked if all the fields are filled, the vacancy is checked if it is valid or not and if all conditions are satisfied, a vacancy is added to the proper class i.e. to full time or part time.

When a button is pressed to appoint staff, first it is checked if the vacancy is valid or not, if the user enters the proper values etc. the vacancy is checked for full time and part time and an object is made for the appropriate class and stored in a list (staffList).

The button to terminate checks if the vacancy belongs to PartTimeStaffHire and if any staff has joined or not and if the conditions are valid, method to terminate staff is called and appropriate message is displayed

The display button displays the details of the job and the employee. The objects in the staffList are checked and proper methods to display them applied.

5 Testing

5.1 Compilation test

Table 1: Test 1

Test 1	
objective	To test if the java files are compiled or not
Action	• Files were compiled in BlueJ
Expected Results	• The java files should be striped in BlueJ before compiling and there should be no stripes after compiling • .class files should appear in the project directory after compiling
Actual Results	• The java files were striped in BlueJ before compiling and there were no stripes after compiling • .class files appeared in the project directory after compiling
Conclusion	Test is successful

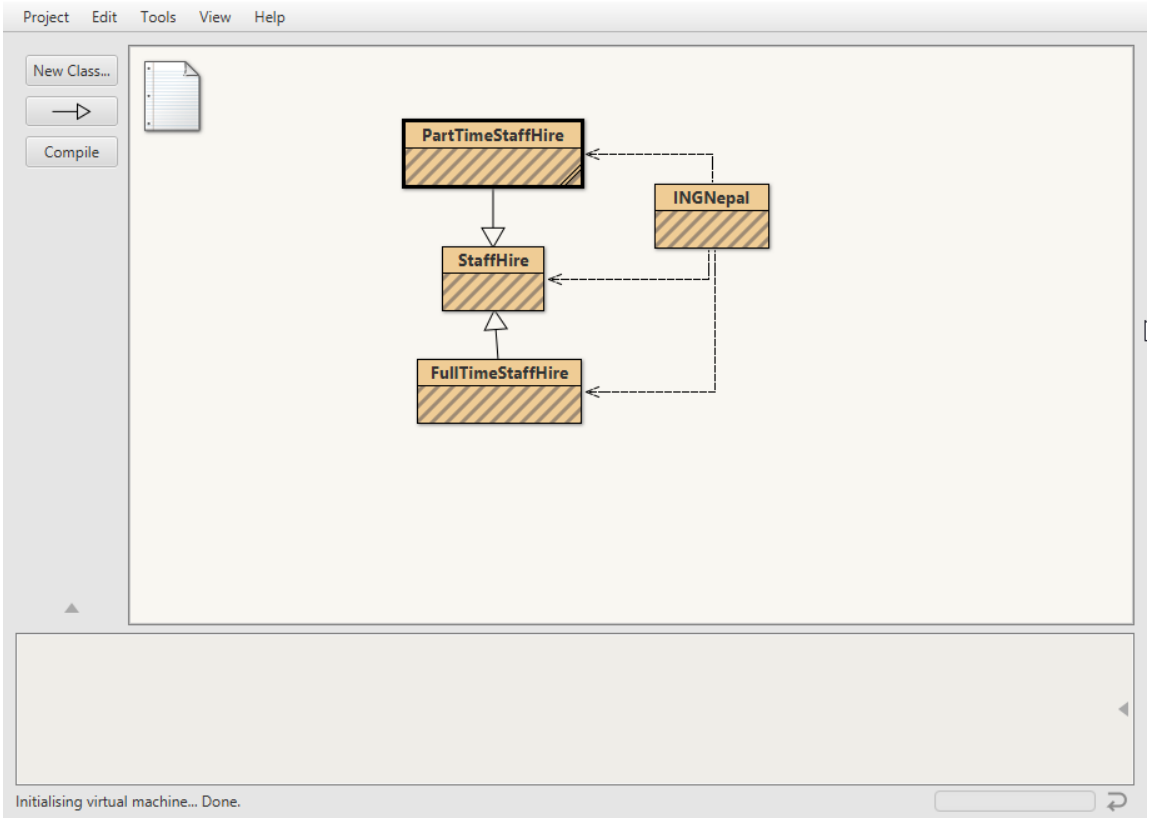

Figure 2: case: BlueJ before compiling

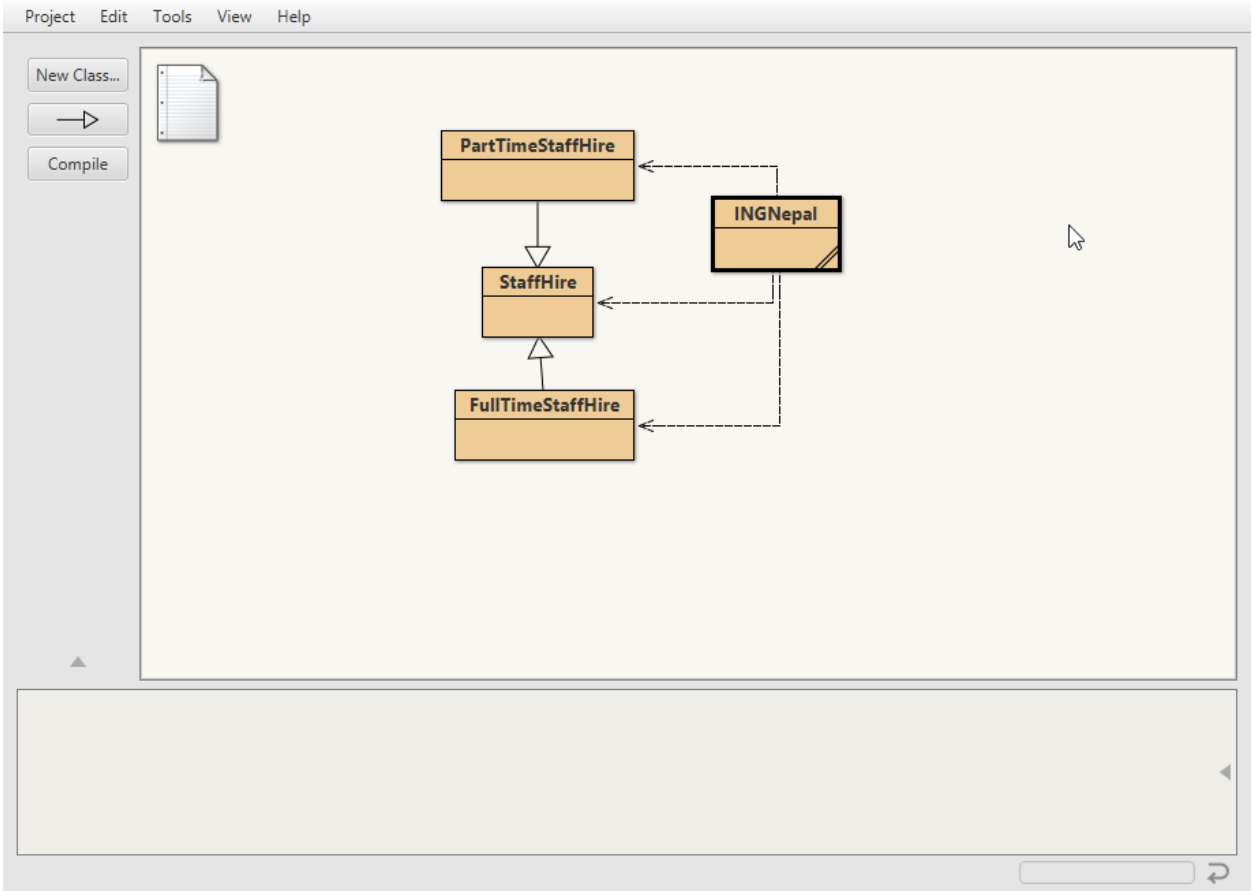

Figure 3: case: BlueJ after compiling

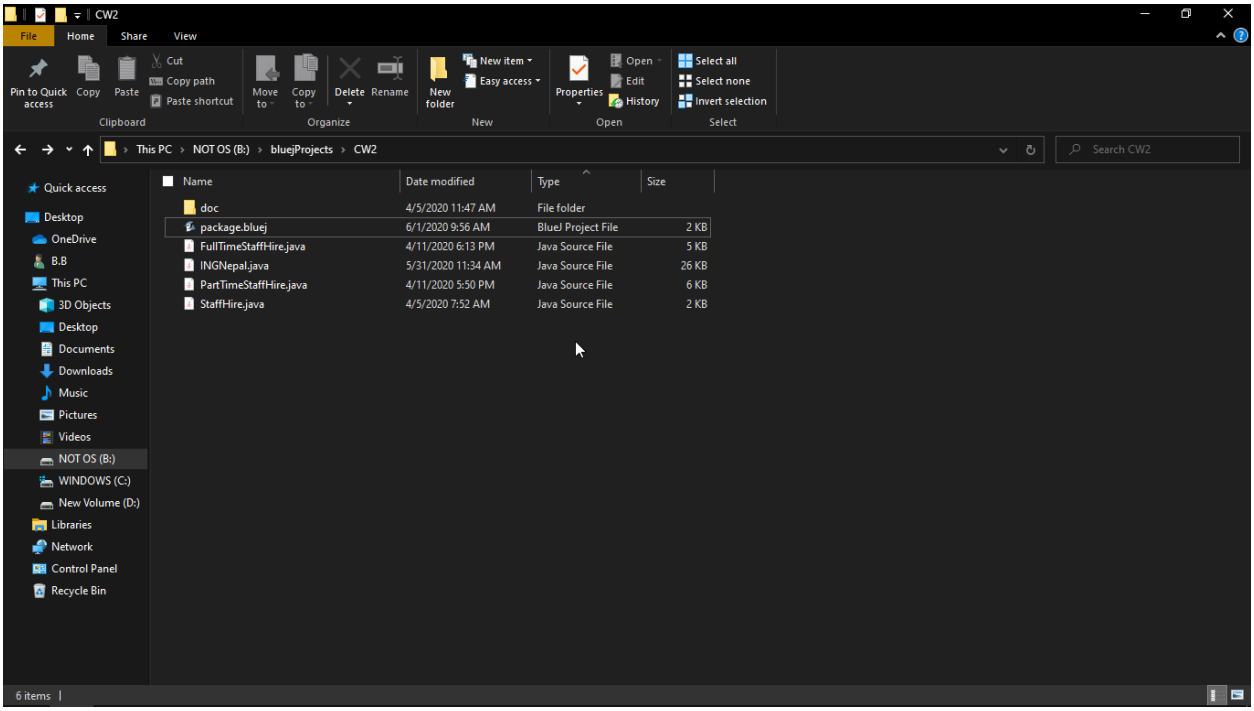

Figure 4: case: Project directory before compiling

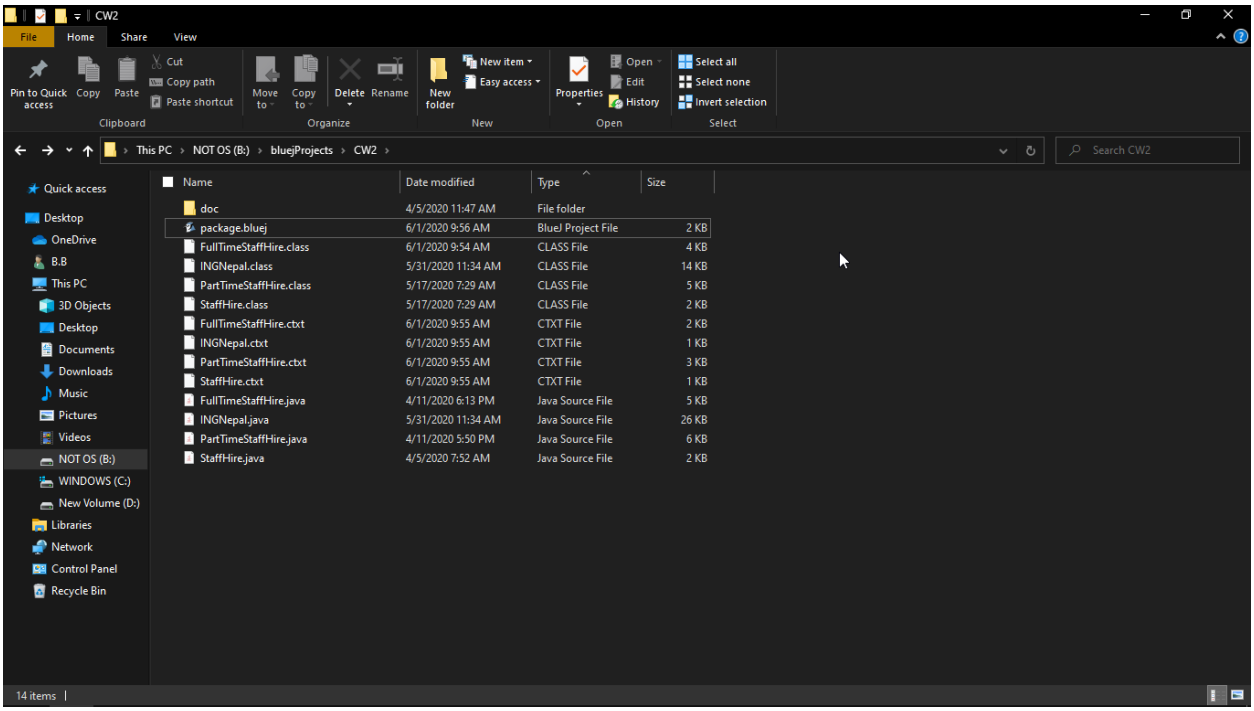

Figure 5: case: Project directory after compiling

5.2 Test to add vacancy (full time and part time)

Table 2: Test 2

Test 2	
objective	To test if vacancies are added properly or not
Action	• Button to add vacancy is pressed without any inputs. • Button to add vacancy is pressed after giving proper values. • Try to add same vacancy twice
Expected Results	• Error message should be displayed when not giving input. • Vacancy should be added when proper inputs are given
Actual Results	• Error message was displayed when given invalid or wrong input • Vacancy is added when given proper valid inputs • Cannot add same vacancy twice
Conclusion	Test is successful

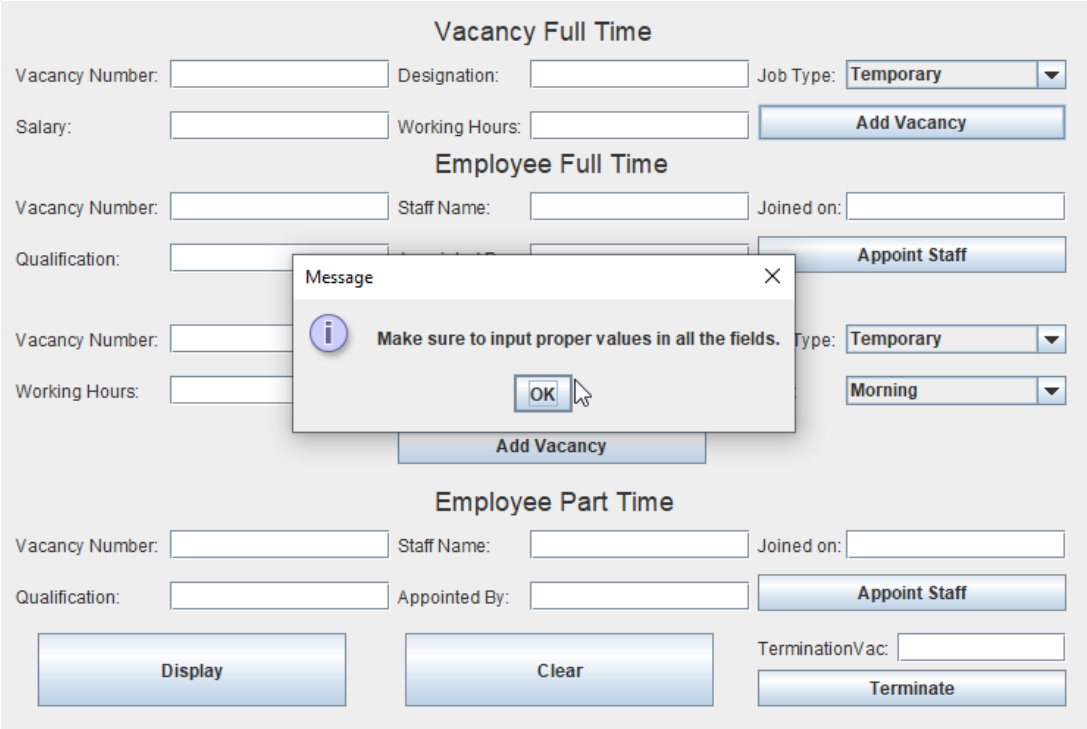

Figure 6: case: no values entered

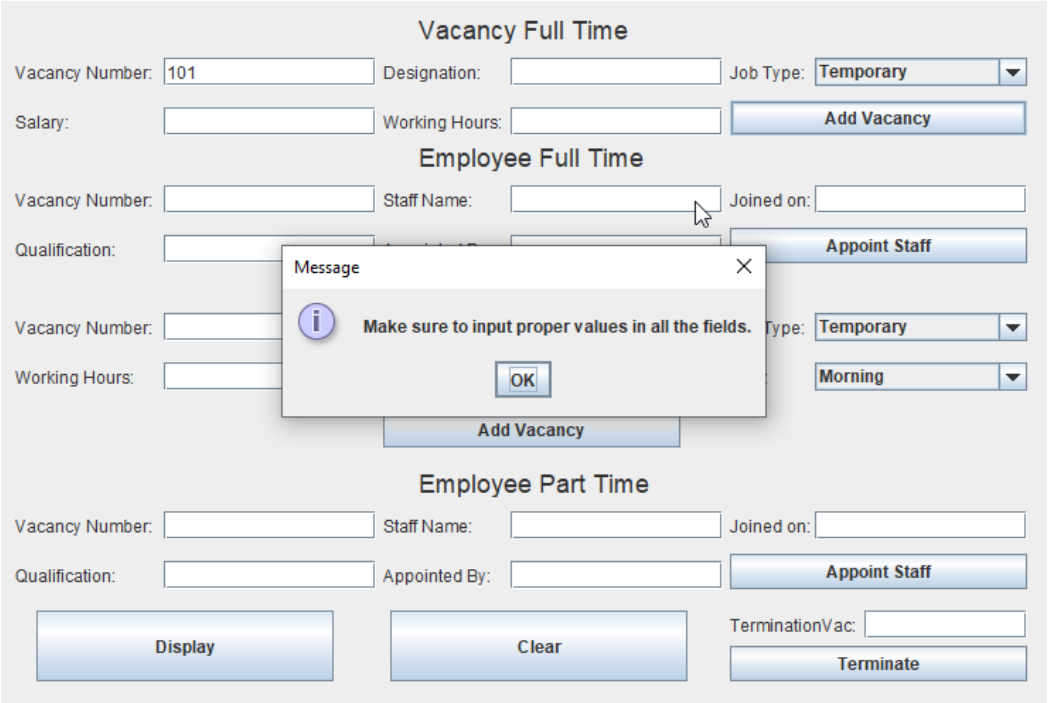

Figure 7: case: some values entered

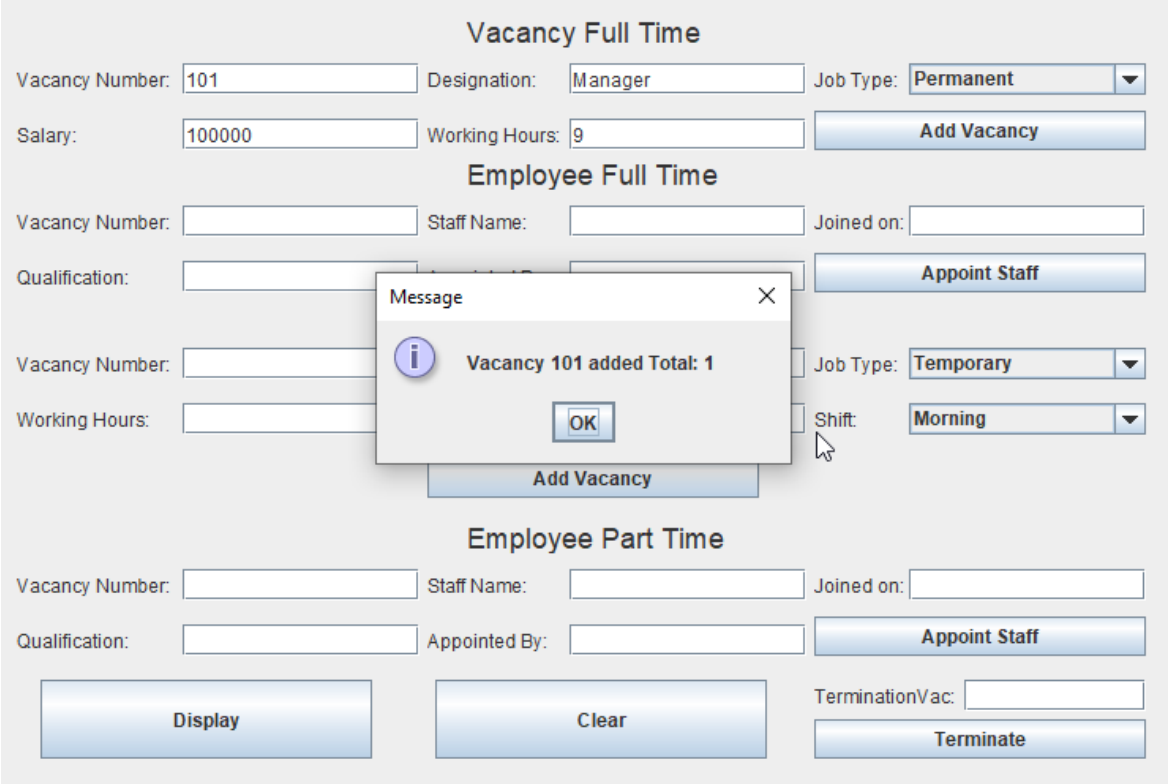

Figure 8: case: Vacancy successfully added

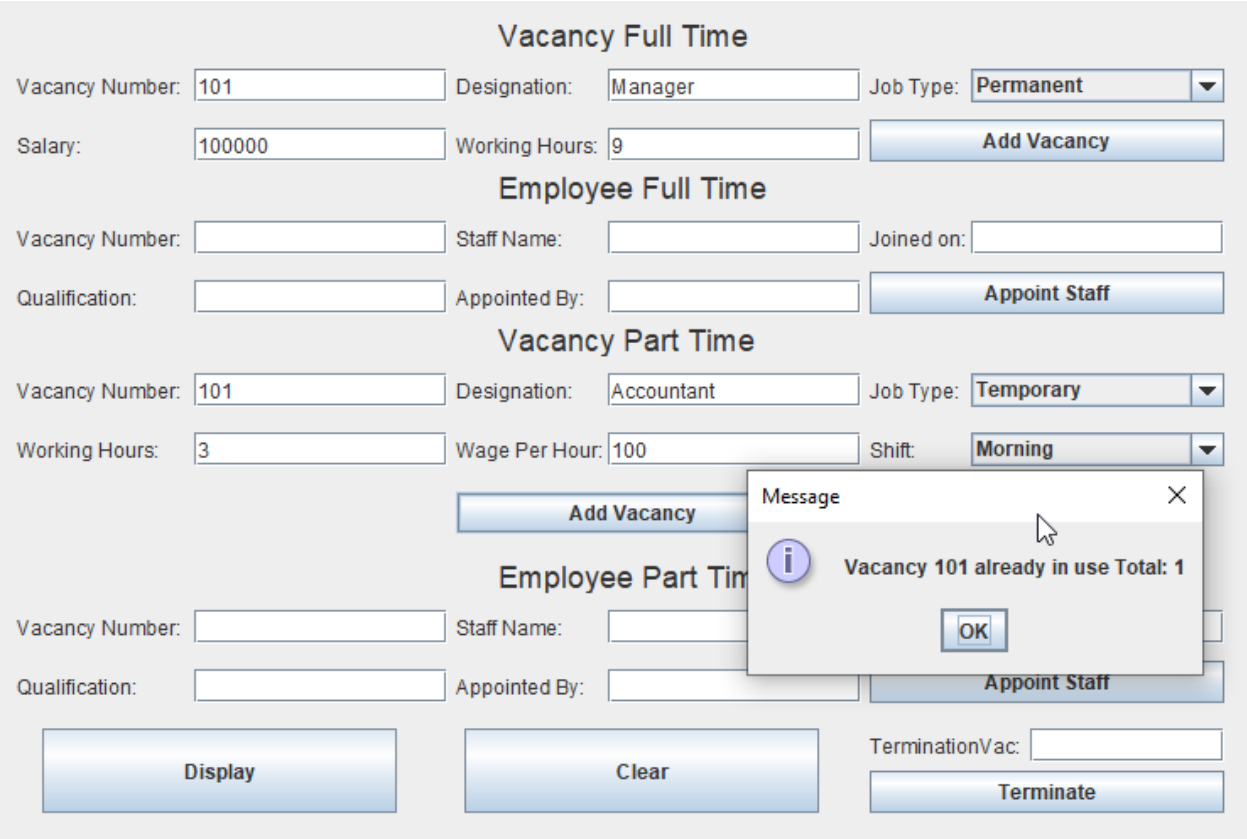

Figure 9: case: duplicate vacancy not added

5.3 Test to appoint staff

Table 3: Test 3

Test 3	
objective	To test buttons to appoint staff
Action	• Buttons to appoint staff are pressed
Expected Results	• Staff should be appointed according to vacancy • Cannot appoint staff if no vacancy or incorrect vacancy
Actual Results	• Staff were appointed according to vacancy • Cannot appoint staff if no vacancy or incorrect vacancy
Conclusion	Test is successful

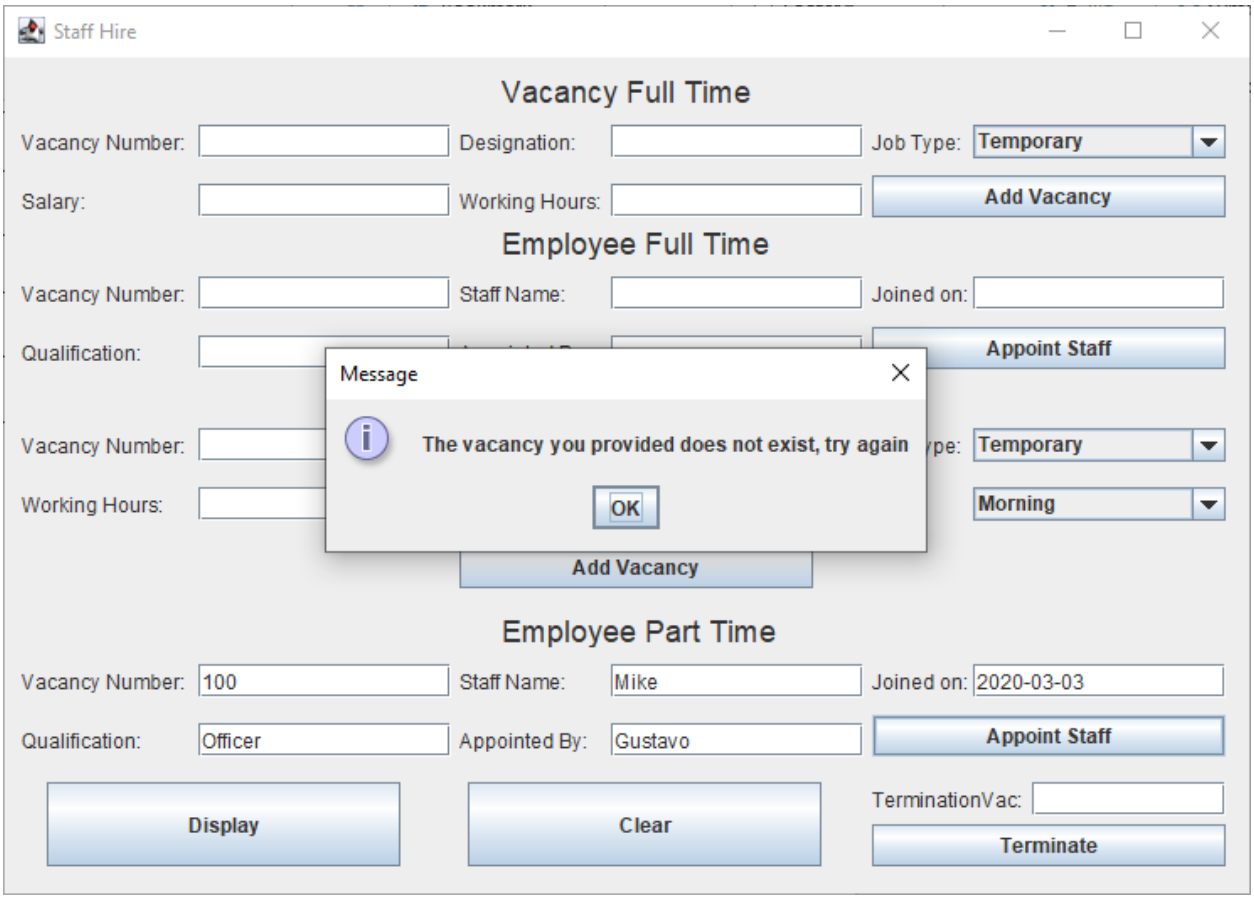

Figure 10: case: cannot hire staff if nonexistent vacancy is used

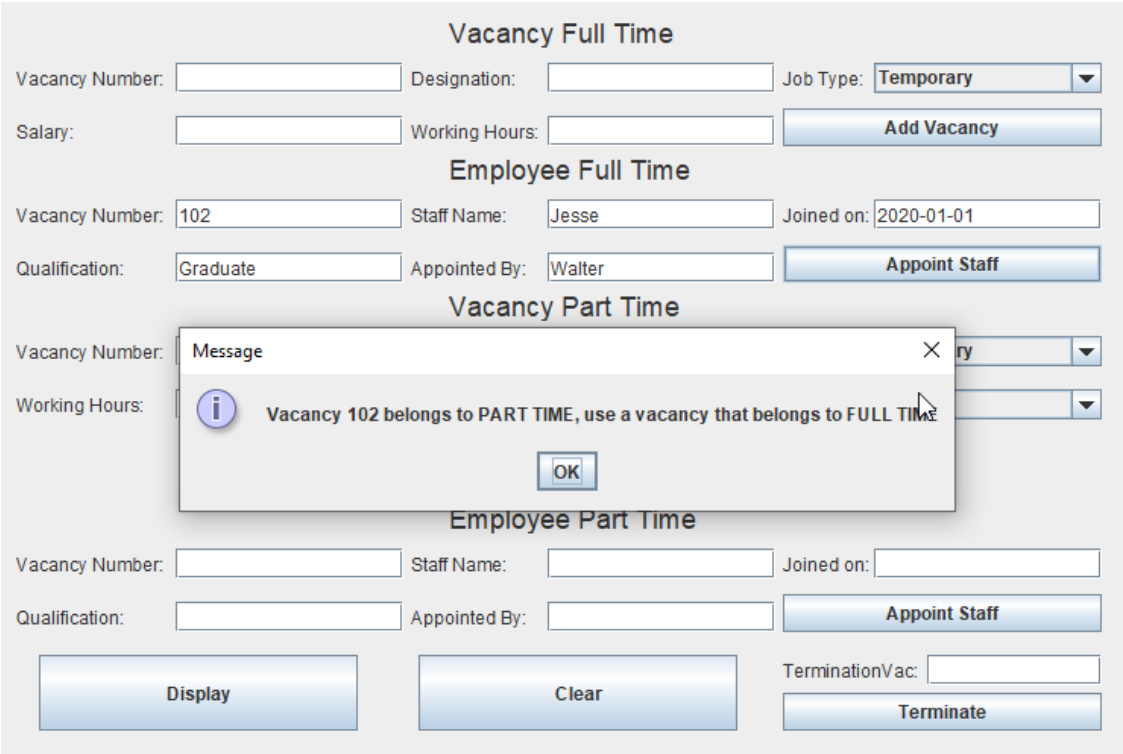

Figure 11: case: cannot hire staff if wrong vacancy used

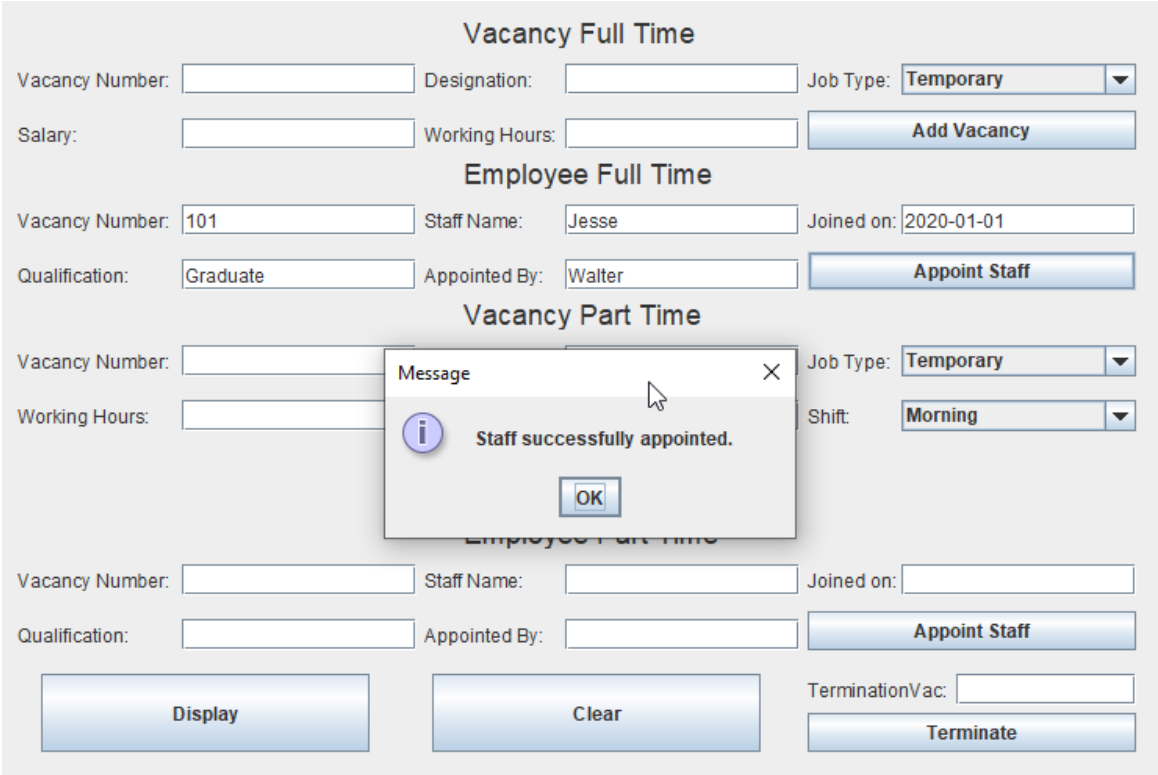

Figure 12: case: staff is hired if correct details are given

5.4 Test to terminate staff

Table 4: Test 4

Test 4	
objective	To test terminate staff
Action	• Terminate button is pressed by giving invalid vacancy numbers • Terminate button is tested using valid vacancy number • Another staff is hired in place of terminated staff
Expected Results	• Error message is displayed when giving invalid vacancy number • Staff is terminated when giving valid vacancy number • New staff can be hired
Actual Results	• Error message was displayed when giving invalid vacancy number • Staff was terminated when giving valid vacancy number • New staff was hired
Conclusion	Test is successful

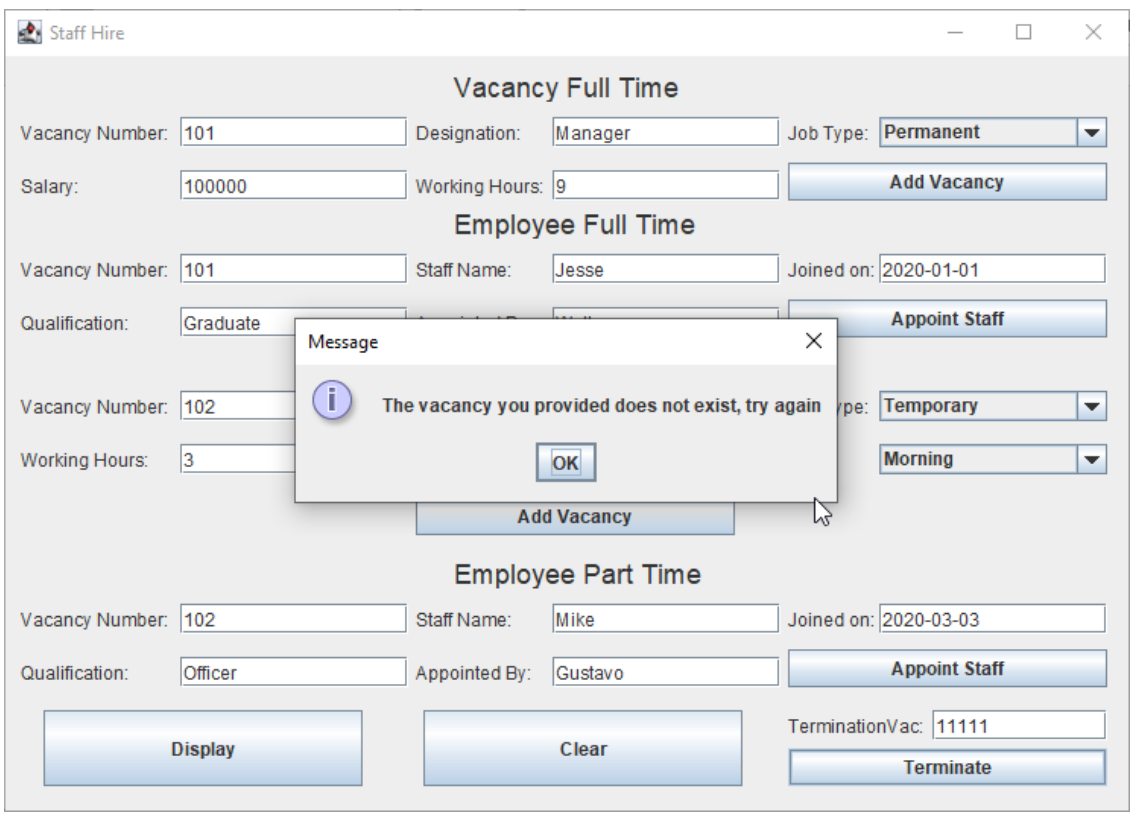

Figure 13: case: trying to terminate invalid vacancy

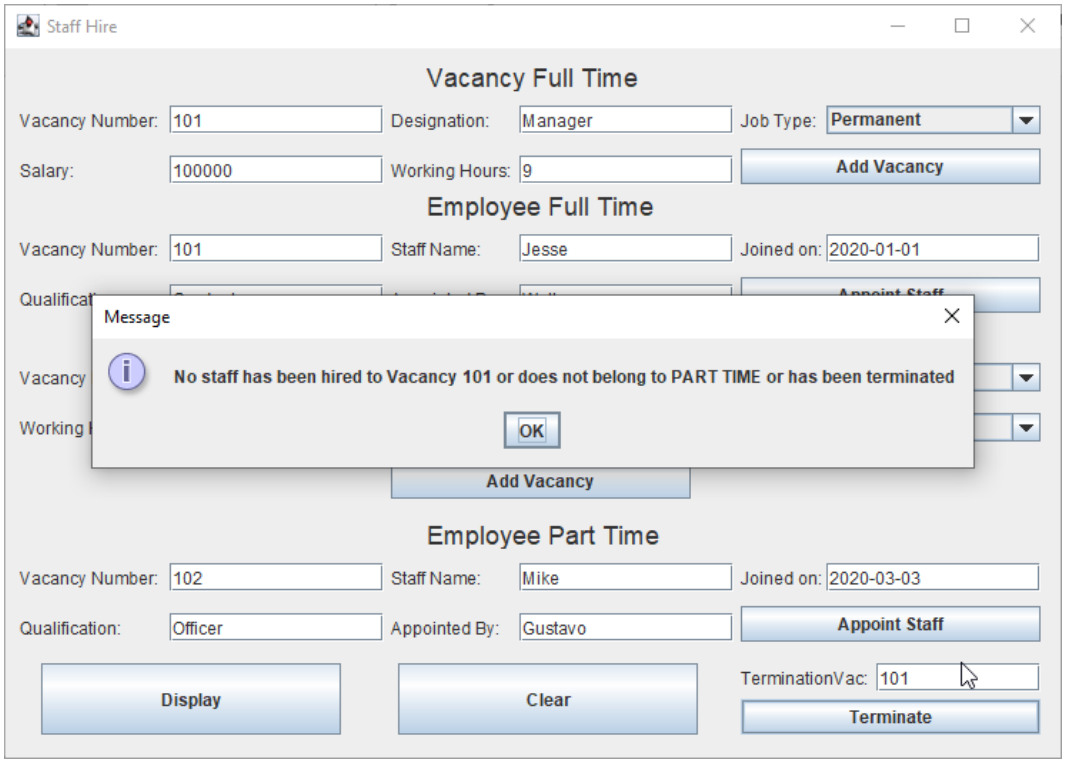

Figure 14: case: trying to terminate full time staff

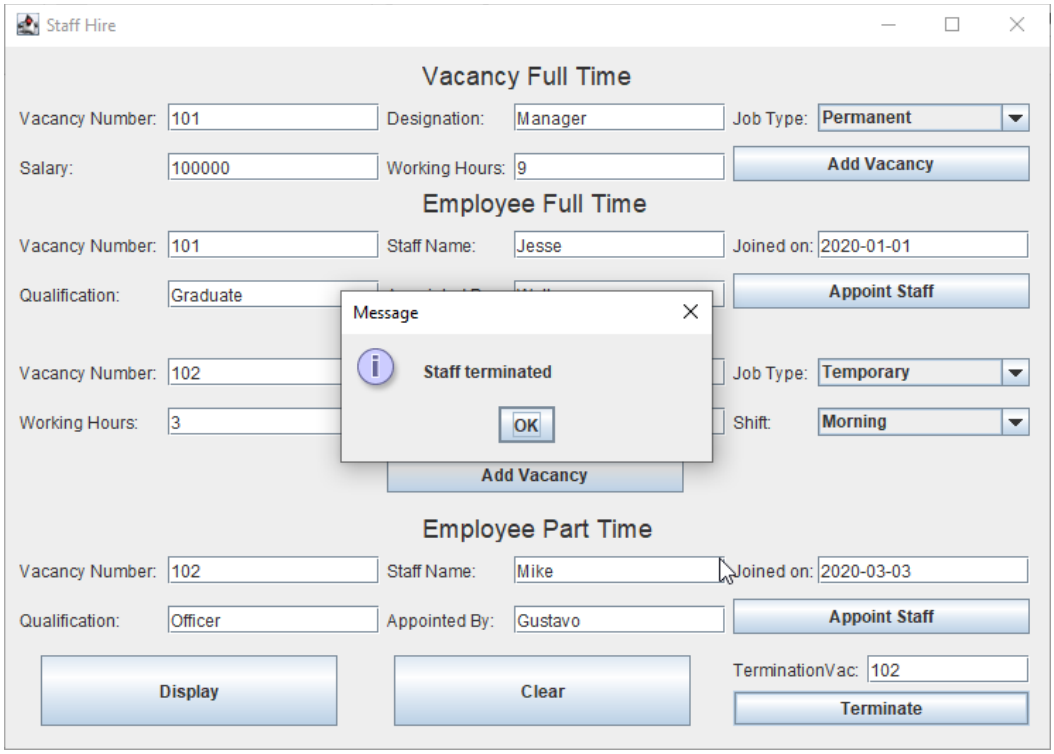

Figure 15: case: part time staff terminated

```
*_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-*

Part Time Staff Details

*_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-*

The details of the job are:

The vacancy number for the job is: 102
The designation for the job is: Accontant
The job type is: Temporary

The details of the staff are:

The name of the staff is: Mike
Mike's wage per hour is: 100.0
Mike's working hours are: 3.0
Mike joined in: 2020-03-03
Mike's qualifications are: Officer
Mike was appointed by: Gustavo
Mike's income per day is: 300.0
```

Figure 16: case: details before termination

```
*_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-*

Part Time Staff Details

*_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-*

The details of the job are:

The vacancy number for the job is: 102
The designation for the job is: Accontant
The job type is: Temporary

No staff has been hired for the job.
```

Figure 17: case: details after termination

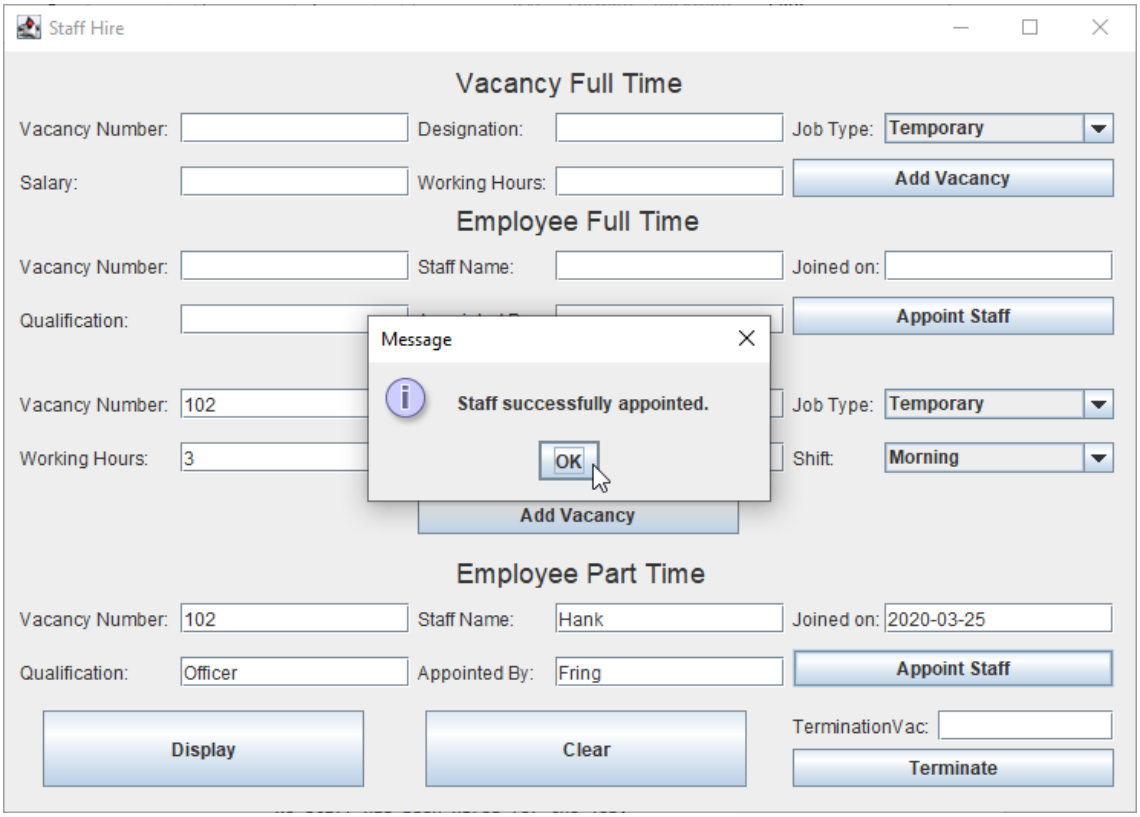

Figure 18: case: rehiring staff

```
*-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-*

Part Time Staff Details

*-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-*

The details of the job are:

The vacancy number for the job is: 102
The designation for the job is: Accountant
The job type is: Temporary

The details of the staff are:

The name of the staff is: Hank
Hank's wage per hour is: 100.0
Hank's working hours are: 3.0
Hank joined in: 2020-03-25
Hank's qualifications are: Officer
Hank was appointed by: Fring
Hank's income per day is: 300.0
```

Figure 19: case: staff is rehired

5.5 Test for Clear and Display button

Table 5: Test 5

Test 5	
objective	To test Clear and Display button
Action	• Clear button was pressed • Display button was pressed before any data was given and after some data given
Expected Results	• Everything is reset when Clear button is pressed. • Display button says no data to show when no data and prints the data in screen when there are some details to show
Actual Results	• Everything is reset when Clear button is pressed. • Display button says no data to show when no data and prints the data in screen when there are some details to show
Conclusion	Test is successful

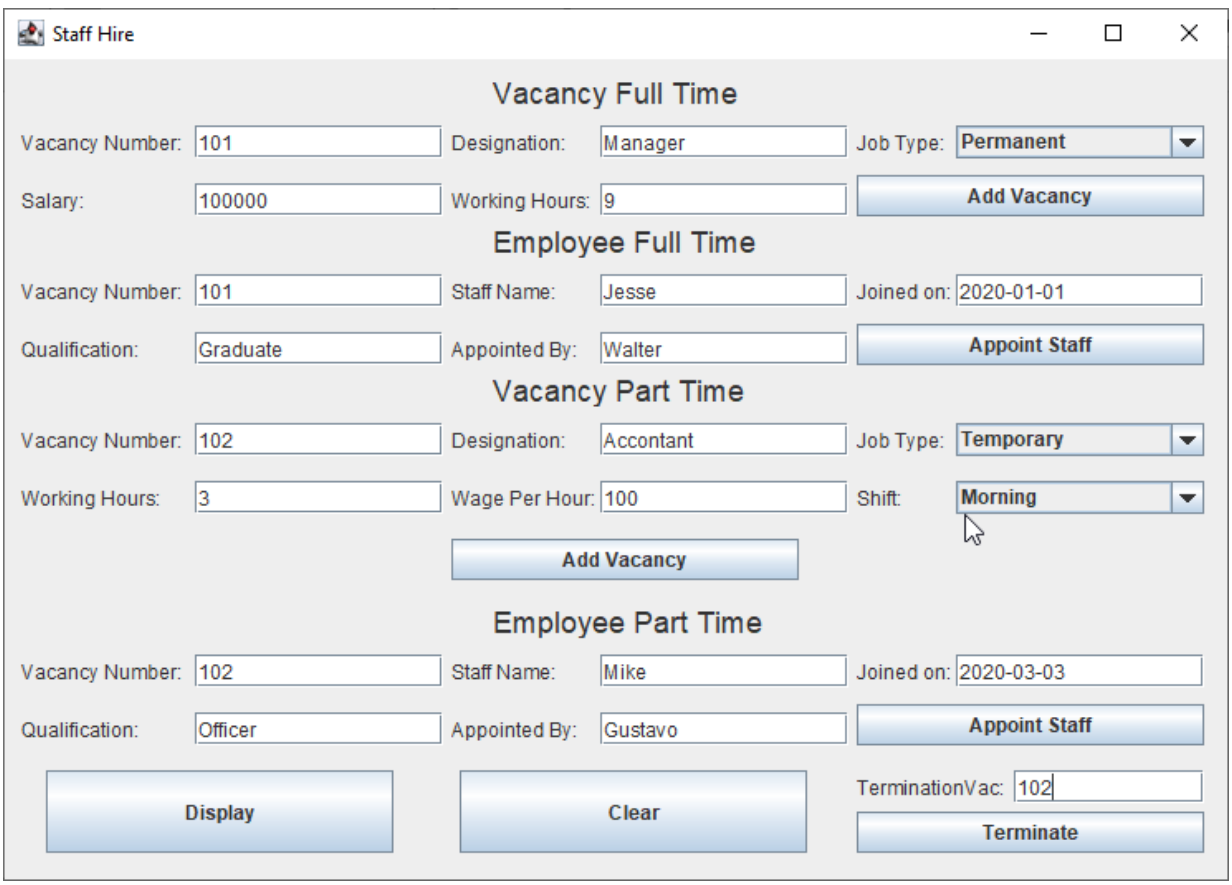

Figure 20: case: before Clear button is pressed

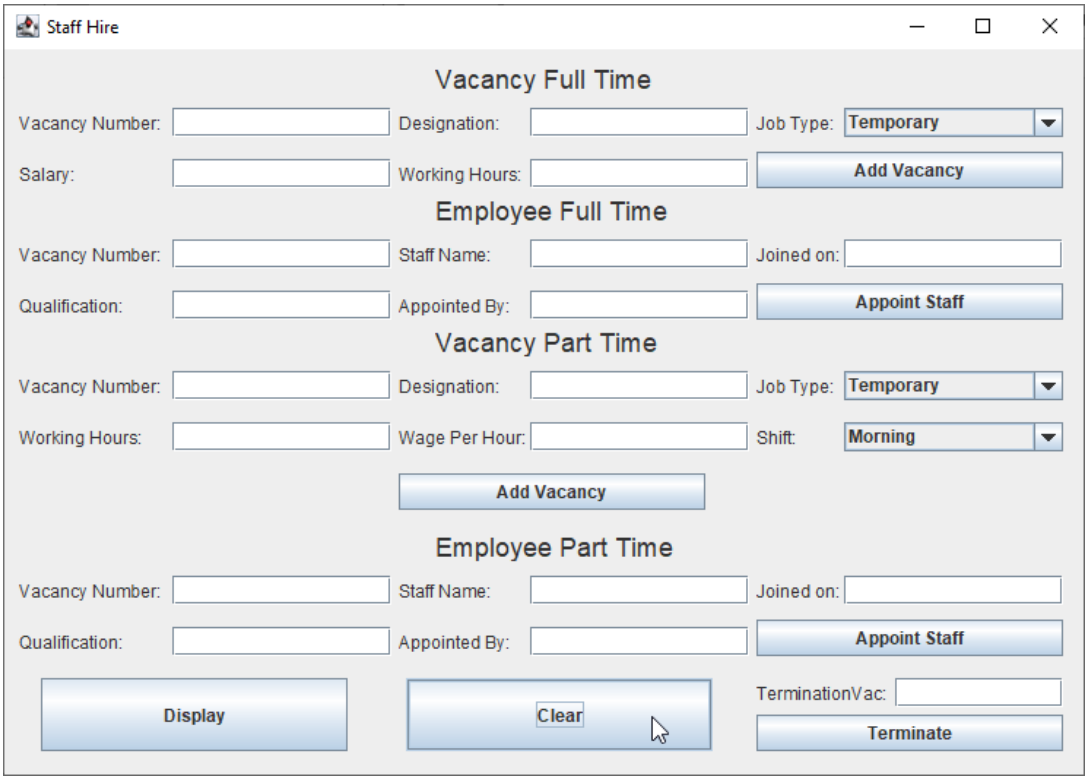

Figure 21: case: after Clear button is pressed

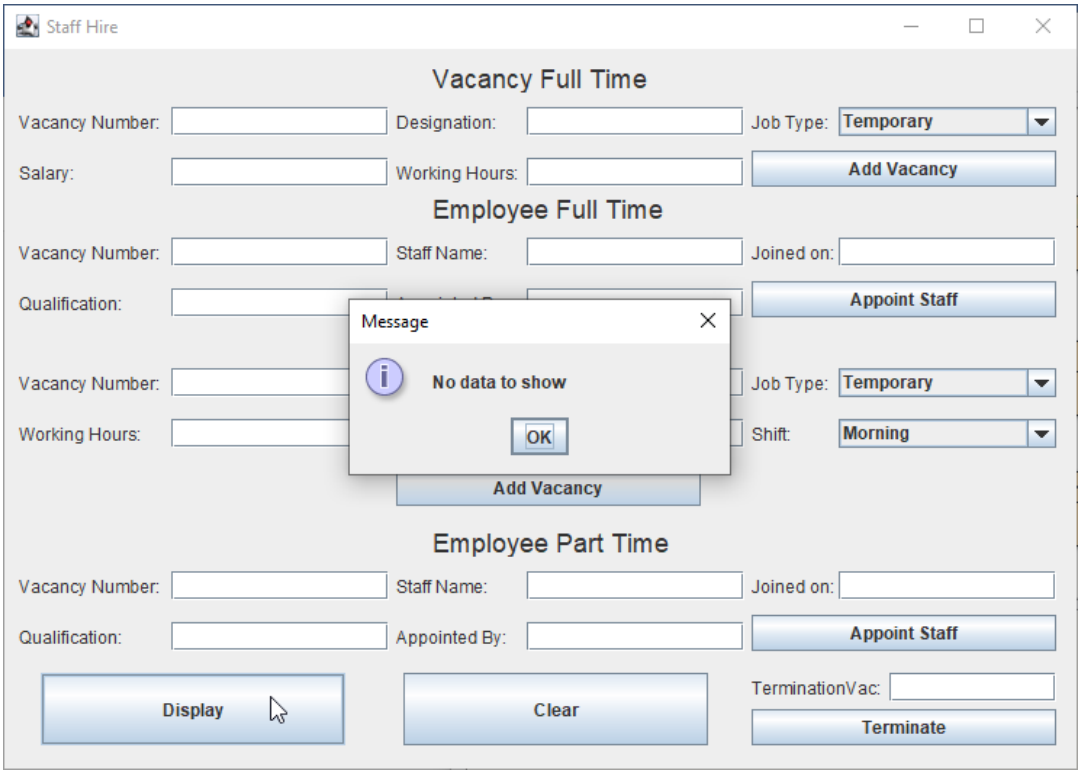

Figure 22: case: Display button pressed without any data

```
Options
*-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-*

Part Time Staff Details

*-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-*

The details of the job are:

The vacancy number for the job is: 102
The designation for the job is: Accountant
The job type is: Temporary

The details of the staff are:

The name of the staff is: Hank
Hank's wage per hour is: 100.0
Hank's working hours are: 3.0
Hank joined in: 2020-03-25
Hank's qualifications are: Officer
Hank was appointed by: Fring
Hank's income per day is: 300.0
*-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-*

Full Time Staff Details

*-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-*

The details of the job are:

The vacancy number for the job is: 101
The designation for the job is: Manager
The job type is: Permanent

The details of the staff are:

The name of the staff is: Jesse
Jesse's salary is: 100000.0
Jesse's working hours are: 9.0
Can only enter input while your programming is running
```

Figure 23: case: Display button pressed after there is some data

6 Error detection and correction

6.1 Error 1 incompatible types

```
if(click.getSource()==btnAddVacancyFT){
    int vacancyNumber=0;
    double salary=0;
    double workingHour=0;
    String jobType="";
    String designation="";
    try{
        vacancyNumber=Integer.parseInt(txtVacancyNumberFT.getText());
        designation=txtDesignationFT.getText();
        jobType=(cmbJobTypeFT.getSelectedItem()).toString();
        salary=Double.parseDouble(txtSalaryFT.getText());
        workingHour=Double.parseDouble(txtWorkingHourFT.getText());
        boolean vacancyNumberIsduplicate=false;
        if( vacancyNumberIsduplicate==false){
            FullTimeStaffHire obj= new FullTimeStaffHire(vacancyNumber,salary,workingHour,jobType,designation);
        }
    }
incompatible types: double cannot be converted to java.la
```

Figure 24: Error 1

6.1.1 Error 1 correction

The error was caused when arguments were passes in wrong order, error was handled by passing correct variables according to FullTimeStaffHire.

```
if(click.getSource()==btnAddVacancyFT){
    int vacancyNumber=0;
    double salary=0;
    double workingHour=0;
    String jobType="";
    String designation="";
    try{
        vacancyNumber=Integer.parseInt(txtVacancyNumberFT.getText());
        designation=txtDesignationFT.getText();
        jobType=(cmbJobTypeFT.getSelectedItem()).toString();
        salary=Double.parseDouble(txtSalaryFT.getText());
        workingHour=Double.parseDouble(txtWorkingHourFT.getText());
        boolean vacancyNumberIsduplicate=false;
        if( vacancyNumberIsduplicate==false){
            FullTimeStaffHire obj= new FullTimeStaffHire(vacancyNumber,designation,jobType,salary,workingHour);
        }
    }
```

Figure 25: Error 1 correction

6.2 Error 2 no arguments found

```
}
if(click.getSource()==btnAppointStaffFT){
    try{
        vacancyNumber=Integer.parseInt(txtVacancyNumber2FT.getText());
        staffName=txtStaffNameFT.getText();
        joinedOn=txtJoiningDateFT.getText();
        qualification=txtQualificationFT.getText();
        appointedBy=txtAppointedByFT.getText();

        boolean vacancyInUse=false;
        //staffList is an object of StaffHire
        for(StaffHire obj:staffList){
            if(obj.getVacancyNumber()==vacancyNumber){
            vacancyInUse=true; //vacancyInUse is set to true if a vacancy number is already in the list
            if(obj instanceof FullTimeStaffHire){//instanceof is used to check if the boject belongs to FullTimeStaf
                FullTimeStaffHire call=(FullTimeStaffHire)obj; //typecasting to object of FullTimeStaffHire to call
                if(call.getJoined()==true){
                  JOptionPa
                }
                else{
                    call.hi
                    JOption
                    break;
                }
```

```
method getJoined in class FullTimeStaffHire cannot be applied to given
types;
  required: boolean
  found: no arguments
  reason: actual and formal argument lists differ in length
```

Figure 26: Error 2

6.2.1 Error 2 correction

The error was caused when getJoined () was used without argument. The method has argument boolean joined in FullTimeStaffHire. Error was solved by introducing a boolean value joined and passing it.

```
vacancyNumber=Integer.parseInt(txtVacancyNumber2FT.getText());
staffName=txtStaffNameFT.getText();
joinedOn=txtJoiningDateFT.getText();
qualification=txtQualificationFT.getText();
appointedBy=txtAppointedByFT.getText();
boolean joined=false;
boolean vacancyInUse=false;
//staffList is an object of StaffHire
for(StaffHire obj:staffList){
    if(obj.getVacancyNumber()==vacancyNumber){
    vacancyInUse=true; //vacancyInUse is set to true if a vacancy number is already in the list
    if(obj instanceof FullTimeStaffHire){//instanceof is used to check if the boject belongs to FullTimeStaffHire
        FullTimeStaffHire call=(FullTimeStaffHire)obj; //typecasting to object of FullTimeStaffHire to call method to hire
        if(call.getJoined(joined)==true){
          JOptionPane.showMessageDialog(frm,"Staff has been already hired");
        }
        else{
```

Figure 27: Error 2 correction

6.3 Error 3 Number format exception

```
Options

Can only enter input while your programming is running

Exception in thread "AWT-EventQueue-0" java.lang.NumberFormatException: For input string: ""
        at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:65)
        at java.base/java.lang.Integer.parseInt(Integer.java:662)
        at java.base/java.lang.Integer.parseInt(Integer.java:770)
        at INGNepal.actionPerformed(INGNepal.java:434)
        at java.desktop/javax.swing.AbstractButton.fireActionPerformed(AbstractButton.java:1967)
        at java.desktop/javax.swing.AbstractButton$Handler.actionPerformed(AbstractButton.java:2308)
        at java.desktop/javax.swing.DefaultButtonModel.fireActionPerformed(DefaultButtonModel.java:405)
        at java.desktop/javax.swing.DefaultButtonModel.setPressed(DefaultButtonModel.java:262)
        at java.desktop/javax.swing.plaf.basic.BasicButtonListener.mouseReleased(BasicButtonListener.java:279)
        at java.desktop/java.awt.Component.processMouseEvent(Component.java:6632)
        at java.desktop/javax.swing.JComponent.processMouseEvent(JComponent.java:3342)
        at java.desktop/java.awt.Component.processEvent(Component.java:6397)
        at java.desktop/java.awt.Container.processEvent(Container.java:2263)
        at java.desktop/java.awt.Component.dispatchEventImpl(Component.java:5008)
        at java.desktop/java.awt.Container.dispatchEventImpl(Container.java:2321)
        at java.desktop/java.awt.Component.dispatchEvent(Component.java:4840)
        at java.desktop/java.awt.LightweightDispatcher.retargetMouseEvent(Container.java:4918)
```

Figure 28: Error 3

6.3.2 Error 3 correction

The error was caused because exception handling was not done in program, error was corrected by implementing exception handling in the program. When exception was encountered, the program was set to display some message.

```
} catch (Exception exp) {
    JOptionPane.showMessageDialog(frm, "Make sure to input proper values in all the fields.");
}
```

Figure 29: Error 3 correction

7 Conclusion

The project was to improve upon the previous course work and make it even more functional. GUI was added which made the program more presentable and user friendly. This project enhanced the features of the previous project. In the previous project, you had to enter data for full time and part time staff from different places. But now everything is included in a single place. Data for staff can be added, viewed and removed from a single place which makes it convenient and simple for the user.

The aim of programming is to find solution to our problems and make things easy for us. This project has successfully demonstrated that we can always find better approach to solve a problem, even just implementing a simple GUI can also add huge improvements and provide a smoother user experience.

8 Appendix

8.1 Appendix 1

```
/**
 * @author Bijay Bharati
 * @version 0.01
 */
import java.awt.event.*;
import javax.swing.*;
import java.util.*;
import java.awt.*;

public class INGNepal implements ActionListener {
    ArrayList<StaffHire> staffList = new ArrayList<StaffHire>();

    JFrame frm;

    JLabel lblTitleFT, lblVacancyNumberFT, lblDesignationFT, lblJobTypeFT,
lblSalaryFT, lblWorkingHoursFT, lblTitle2FT,
            lblVacancyNumber2FT, lblStaffNameFT, lblQualificationFT, lblAppointedByFT,
lblJoiningDateFT,

            lblTitlePT, lblVacancyNumberPT, lblDesignationPT, lblJobTypePT,
lblWorkingHoursPT, lblShiftPT,
            lblWagePerHrPT, lblTitle2PT, lblVacancyNumber2PT, lblStaffNamePT,
lblQualificationPT, lblAppointedByPT,
            lblJoiningDatePT,

            lblTerminationVacancy;
```

```
    JTextField txtVacancyNumberFT, txtDesignationFT, txtSalaryFT, txtWorkingHoursFT,
txtVacancyNumber2FT,
            txtStaffNameFT, txtQualificationFT, txtAppointedByFT, txtJoiningDateFT,

            txtVacancyNumberPT, txtDesignationPT, txtWorkingHoursPT, txtWagePerHrPT,
txtVacancyNumber2PT,
            txtStaffNamePT, txtQualificationPT, txtAppointedByPT, txtJoiningDatePT,

            txtTerminationVacancy;

    JComboBox cmbJobTypeFT, cmbJobTypePT, cmbShiftPT;

    JButton btnAddVacancyFT, btnAppointStaffFT, btnAddVacancyPT,
btnAppointStaffPT, btnDisplay, btnClear, btnTerminate;

    int vacancyNumber;
    double salary;
    double workingHour;
    String jobType;
    String designation;
    String staffName;
    String joinedOn;
    String qualification;
    String appointedBy;
    double wagePerHr;
    String shift;
    int terminationVacancy;
    boolean joined;
    boolean vacancyInUse;
```

```
    boolean terminated;

    public static void main(String[] args) {
        INGNepal form = new INGNepal();
        form.staffHireForm();
    }

    public void staffHireForm() {
        frm = new JFrame("Staff Hire");         //Frame for GUI where all the buttons, labels,
text fields etx will be added
        frm.setSize(755, 535);
        frm.setLayout(null);
        frm.setVisible(true);
        frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        ////////////////////////////////FULL TIME//////////////////////////////////////////////
        lblTitleFT = new JLabel("Vacancy Full Time");
        lblTitleFT.setBounds(295, 10, 300, 20);
        lblTitleFT.setFont(new Font("Helvetica", Font.PLAIN, 18));
        frm.add(lblTitleFT);

        lblVacancyNumberFT = new JLabel("Vacancy Number:");
        lblVacancyNumberFT.setBounds(10, 40, 100, 20);
        lblVacancyNumberFT.setFont(new Font("sans", Font.PLAIN, 12));
        frm.add(lblVacancyNumberFT);

        txtVacancyNumberFT = new JTextField();
        txtVacancyNumberFT.setBounds(115, 40, 150, 20);
        frm.add(txtVacancyNumberFT);
```

```
lblDesignationFT = new JLabel("Designation:");
lblDesignationFT.setBounds(270, 40, 100, 20);
lblDesignationFT.setFont(new Font("sans", Font.PLAIN, 12));
frm.add(lblDesignationFT);

txtDesignationFT = new JTextField();
txtDesignationFT.setBounds(360, 40, 150, 20);
frm.add(txtDesignationFT);

lblJobTypeFT = new JLabel("Job Type:");
lblJobTypeFT.setBounds(515, 40, 100, 20);
lblJobTypeFT.setFont(new Font("sans", Font.PLAIN, 12));
frm.add(lblJobTypeFT);

String jobTypeFT[] = { "Temporary", "Permanent" };
cmbJobTypeFT = new JComboBox(jobTypeFT);
cmbJobTypeFT.setBounds(575, 40, 150, 20);
frm.add(cmbJobTypeFT);

lblSalaryFT = new JLabel("Salary:");
lblSalaryFT.setBounds(10, 75, 100, 20);
lblSalaryFT.setFont(new Font("sans", Font.PLAIN, 12));
frm.add(lblSalaryFT);

txtSalaryFT = new JTextField();
txtSalaryFT.setBounds(115, 75, 150, 20);
frm.add(txtSalaryFT);
```

```
lblWorkingHoursFT = new JLabel("Working Hours:");
lblWorkingHoursFT.setBounds(270, 75, 100, 20);
lblWorkingHoursFT.setFont(new Font("sans", Font.PLAIN, 12));
frm.add(lblWorkingHoursFT);

txtWorkingHoursFT = new JTextField();
txtWorkingHoursFT.setBounds(360, 75, 150, 20);
frm.add(txtWorkingHoursFT);

btnAddVacancyFT = new JButton("Add Vacancy");
btnAddVacancyFT.setBounds(515, 70, 210, 25);
frm.add(btnAddVacancyFT);
btnAddVacancyFT.addActionListener(this);
//////////////////////////////////FULL TIME//////////////////////////////////////////////////
lblTitle2FT = new JLabel("Employee Full Time");
lblTitle2FT.setBounds(295, 100, 300, 20);
lblTitle2FT.setFont(new Font("Helvetica", Font.PLAIN, 18));
frm.add(lblTitle2FT);

lblVacancyNumber2FT = new JLabel("Vacancy Number:");
lblVacancyNumber2FT.setBounds(10, 130, 100, 20);
lblVacancyNumber2FT.setFont(new Font("sans", Font.PLAIN, 12));
frm.add(lblVacancyNumber2FT);

txtVacancyNumber2FT = new JTextField();
txtVacancyNumber2FT.setBounds(115, 130, 150, 20);
frm.add(txtVacancyNumber2FT);
```

```
lblStaffNameFT = new JLabel("Staff Name:");
lblStaffNameFT.setBounds(270, 130, 100, 20);
lblStaffNameFT.setFont(new Font("sans", Font.PLAIN, 12));
frm.add(lblStaffNameFT);

txtStaffNameFT = new JTextField();
txtStaffNameFT.setBounds(360, 130, 150, 20);
frm.add(txtStaffNameFT);

lblJoiningDateFT = new JLabel("Joined on:");
lblJoiningDateFT.setBounds(515, 130, 100, 20);
lblJoiningDateFT.setFont(new Font("sans", Font.PLAIN, 12));
frm.add(lblJoiningDateFT);

txtJoiningDateFT = new JTextField();
txtJoiningDateFT.setBounds(575, 130, 150, 20);
frm.add(txtJoiningDateFT);

lblQualificationFT = new JLabel("Qualification:");
lblQualificationFT.setBounds(10, 165, 100, 20);
lblQualificationFT.setFont(new Font("sans", Font.PLAIN, 12));
frm.add(lblQualificationFT);

txtQualificationFT = new JTextField();
txtQualificationFT.setBounds(115, 165, 150, 20);
frm.add(txtQualificationFT);

lblAppointedByFT = new JLabel("Appointed By:");
```

```
lblAppointedByFT.setBounds(270, 165, 100, 20);
lblAppointedByFT.setFont(new Font("sans", Font.PLAIN, 12));
frm.add(lblAppointedByFT);

txtAppointedByFT = new JTextField();
txtAppointedByFT.setBounds(360, 165, 150, 20);
frm.add(txtAppointedByFT);

btnAppointStaffFT = new JButton("Appoint Staff");
btnAppointStaffFT.setBounds(515, 160, 210, 25);
frm.add(btnAppointStaffFT);
btnAppointStaffFT.addActionListener(this);
//////////////////////////////////FULL TIME//////////////////////////////////////////////////
//////////////////////////////////PART TIME//////////////////////////////////////////////////
lblTitlePT = new JLabel("Vacancy Part Time");
lblTitlePT.setBounds(295, 190, 300, 20);
lblTitlePT.setFont(new Font("Helvetica", Font.PLAIN, 18));
frm.add(lblTitlePT);

lblVacancyNumberPT = new JLabel("Vacancy Number:");
lblVacancyNumberPT.setBounds(10, 220, 100, 20);
lblVacancyNumberPT.setFont(new Font("sans", Font.PLAIN, 12));
frm.add(lblVacancyNumberPT);

txtVacancyNumberPT = new JTextField();
txtVacancyNumberPT.setBounds(115, 220, 150, 20);
frm.add(txtVacancyNumberPT);
```

```
lblDesignationPT = new JLabel("Designation:");
lblDesignationPT.setBounds(270, 220, 100, 20);
lblDesignationPT.setFont(new Font("sans", Font.PLAIN, 12));
frm.add(lblDesignationPT);

txtDesignationPT = new JTextField();
txtDesignationPT.setBounds(360, 220, 150, 20);
frm.add(txtDesignationPT);

lblJobTypePT = new JLabel("Job Type:");
lblJobTypePT.setBounds(515, 220, 100, 20);
lblJobTypePT.setFont(new Font("sans", Font.PLAIN, 12));
frm.add(lblJobTypePT);

String jobTypePT[] = { "Temporary", "Permanent" };
cmbJobTypePT = new JComboBox(jobTypePT);
cmbJobTypePT.setBounds(575, 220, 150, 20);
frm.add(cmbJobTypePT);

lblWorkingHoursPT = new JLabel("Working Hours:");
lblWorkingHoursPT.setBounds(10, 255, 100, 20);
lblWorkingHoursPT.setFont(new Font("sans", Font.PLAIN, 12));
frm.add(lblWorkingHoursPT);

txtWorkingHoursPT = new JTextField();
txtWorkingHoursPT.setBounds(115, 255, 150, 20);
frm.add(txtWorkingHoursPT);
```

```
lblWagePerHrPT = new JLabel("Wage Per Hour:");
lblWagePerHrPT.setBounds(270, 255, 100, 20);
lblWagePerHrPT.setFont(new Font("sans", Font.PLAIN, 12));
frm.add(lblWagePerHrPT);

txtWagePerHrPT = new JTextField();
txtWagePerHrPT.setBounds(360, 255, 150, 20);
frm.add(txtWagePerHrPT);

lblShiftPT = new JLabel("Shift:");
lblShiftPT.setBounds(515, 255, 100, 20);
lblShiftPT.setFont(new Font("sans", Font.PLAIN, 12));
frm.add(lblShiftPT);

String shiftPT[] = { "Morning", "Day", "Evening" };
cmbShiftPT = new JComboBox(shiftPT);
cmbShiftPT.setBounds(575, 255, 150, 20);
frm.add(cmbShiftPT);

btnAddVacancyPT = new JButton("Add Vacancy");
btnAddVacancyPT.setBounds(270, 290, 210, 25);
frm.add(btnAddVacancyPT);
btnAddVacancyPT.addActionListener(this);
//////////////////////////////////PART TIME//////////////////////////////////////////////////
lblTitle2PT = new JLabel("Employee Part Time");
lblTitle2PT.setBounds(295, 330, 300, 20);
lblTitle2PT.setFont(new Font("Helvetica", Font.PLAIN, 18));
frm.add(lblTitle2PT);
```

```
lblVacancyNumber2PT = new JLabel("Vacancy Number:");
lblVacancyNumber2PT.setBounds(10, 360, 100, 20);
lblVacancyNumber2PT.setFont(new Font("sans", Font.PLAIN, 12));
frm.add(lblVacancyNumber2PT);

txtVacancyNumber2PT = new JTextField();
txtVacancyNumber2PT.setBounds(115, 360, 150, 20);
frm.add(txtVacancyNumber2PT);

lblStaffNamePT = new JLabel("Staff Name:");
lblStaffNamePT.setBounds(270, 360, 100, 20);
lblStaffNamePT.setFont(new Font("sans", Font.PLAIN, 12));
frm.add(lblStaffNamePT);

txtStaffNamePT = new JTextField();
txtStaffNamePT.setBounds(360, 360, 150, 20);
frm.add(txtStaffNamePT);

lblJoiningDatePT = new JLabel("Joined on:");
lblJoiningDatePT.setBounds(515, 360, 100, 20);
lblJoiningDatePT.setFont(new Font("sans", Font.PLAIN, 12));
frm.add(lblJoiningDatePT);

txtJoiningDatePT = new JTextField();
txtJoiningDatePT.setBounds(575, 360, 150, 20);
frm.add(txtJoiningDatePT);
```

```
lblQualificationPT = new JLabel("Qualification:");
lblQualificationPT.setBounds(10, 395, 100, 20);
lblQualificationPT.setFont(new Font("sans", Font.PLAIN, 12));
frm.add(lblQualificationPT);

txtQualificationPT = new JTextField();
txtQualificationPT.setBounds(115, 395, 150, 20);
frm.add(txtQualificationPT);

lblAppointedByPT = new JLabel("Appointed By:");
lblAppointedByPT.setBounds(270, 395, 100, 20);
lblAppointedByPT.setFont(new Font("sans", Font.PLAIN, 12));
frm.add(lblAppointedByPT);

txtAppointedByPT = new JTextField();
txtAppointedByPT.setBounds(360, 395, 150, 20);
frm.add(txtAppointedByPT);

btnAppointStaffPT = new JButton("Appoint Staff");
btnAppointStaffPT.setBounds(515, 390, 210, 25);
frm.add(btnAppointStaffPT);
btnAppointStaffPT.addActionListener(this);
//////////////////////////////////PART TIME//////////////////////////////////////////////////
//////////////////////////////////DISPLAY CLEAR TERMINATE//////////////////////////////////
btnDisplay = new JButton("Display");
btnDisplay.setBounds(25, 430, 210, 50);
frm.add(btnDisplay);
btnDisplay.addActionListener(this);
```

```
        btnClear = new JButton("Clear");
        btnClear.setBounds(275, 430, 210, 50);
        frm.add(btnClear);
        btnClear.addActionListener(this);

        lblTerminationVacancy = new JLabel("TerminationVac:");
        lblTerminationVacancy.setBounds(515, 430, 100, 20);
        lblTerminationVacancy.setFont(new Font("sans", Font.PLAIN, 12));
        frm.add(lblTerminationVacancy);

        txtTerminationVacancy = new JTextField();
        txtTerminationVacancy.setBounds(610, 430, 115, 20);
        frm.add(txtTerminationVacancy);

        btnTerminate = new JButton("Terminate");
        btnTerminate.setBounds(515, 455, 210, 25);
        frm.add(btnTerminate);
        btnTerminate.addActionListener(this);
        ////////////////////////////////DISPLAY CLEAR TERMINATE//////////////////////////////////
    }

     public String getVacancyNumberFT(){
            return txtVacancyNumberFT.getText();
        }

    public void actionPerformed(ActionEvent click) {
```

```
        if (click.getSource() == btnClear) {
            txtVacancyNumberFT.setText("");
            txtDesignationFT.setText("");
            txtSalaryFT.setText("");
            txtWorkingHoursFT.setText("");
            txtVacancyNumber2FT.setText("");
            txtStaffNameFT.setText("");
            txtQualificationFT.setText("");
            txtAppointedByFT.setText("");
            txtJoiningDateFT.setText("");
            txtVacancyNumberPT.setText("");
            txtDesignationPT.setText("");
            txtWorkingHoursPT.setText("");
            txtWagePerHrPT.setText("");
            txtVacancyNumber2PT.setText("");
            txtStaffNamePT.setText("");
            txtQualificationPT.setText("");
            txtAppointedByPT.setText("");
            txtJoiningDatePT.setText("");
            txtTerminationVacancy.setText("");
            cmbJobTypeFT.setSelectedIndex(0);
            cmbJobTypePT.setSelectedIndex(0);
            cmbShiftPT.setSelectedIndex(0);
        }

        if (click.getSource() == btnAddVacancyFT) {
            try {
                // extracting values
```

```
            vacancyNumber = Integer.parseInt(txtVacancyNumberFT.getText());// parse
int because we extract the value
                                                // as String from the form
            designation = txtDesignationFT.getText();
            jobType = (cmbJobTypeFT.getSelectedItem()).toString();
            salary = Double.parseDouble(txtSalaryFT.getText());
            workingHour = Double.parseDouble(txtWorkingHoursFT.getText());
            vacancyInUse = false;
            // checking if vacancy number is already in use or not
            if (designation.equals("")) {
                JOptionPane.showMessageDialog(frm, "Make sure to input proper values
in all the fields.");
            } else {
                for (StaffHire obj : staffList) {
                    if (obj.getVacancyNumber() == vacancyNumber) {
                        vacancyInUse = true;
                        break;
                    }
                }
                if (vacancyInUse == false) {
                    FullTimeStaffHire obj = new FullTimeStaffHire(vacancyNumber,
designation, jobType, salary,
                            workingHour);
                    staffList.add(obj);
                    JOptionPane.showMessageDialog(frm,
                            "Vacancy " + vacancyNumber + " added Total: " + staffList.size());
                } else {
                    JOptionPane.showMessageDialog(frm,
                            "Vacancy " + vacancyNumber + " already in use Total: " +
staffList.size());
```

```
                }
            }
        } catch (Exception exp) {
            JOptionPane.showMessageDialog(frm, "Make sure to input proper values in
all the fields.");
        }
    }


    if (click.getSource() == btnAppointStaffFT) {
        try {
            vacancyNumber = Integer.parseInt(txtVacancyNumber2FT.getText());
            staffName = txtStaffNameFT.getText();
            joinedOn = txtJoiningDateFT.getText();
            qualification = txtQualificationFT.getText();
            appointedBy = txtAppointedByFT.getText();
            joined = false;
            vacancyInUse = false;
            // staffList is an object of StaffHire
            if (staffName.equals("") || joinedOn.equals("") || qualification.equals("") ||
appointedBy.equals("")) {
             JOptionPane.showMessageDialog(frm, "Make sure to input proper values in
all the fields.");
            } else {
                for (StaffHire obj : staffList) {
                    if (obj.getVacancyNumber() == vacancyNumber) {
                        vacancyInUse = true; // vacancyInUse is set to true if vacancy number
is already in the
                                //  vacancy number is already in the list
                        if (obj instanceof FullTimeStaffHire) {// instanceof is used to check if
the boject belongs
```

```
                                        // to FullTimeStaffHire
                        FullTimeStaffHire call = (FullTimeStaffHire) obj; // typecasting to
object of
                                            // FullTimeStaffHire to call method
                                           // to hire staff
                        if (call.getJoined(joined) == true) {
                            JOptionPane.showMessageDialog(frm, "Staff has been already
hired");
                        } else {
                            call.hireFullTimeStaff(staffName, joinedOn, qualification,
appointedBy);
                            JOptionPane.showMessageDialog(frm, "Staff successfully
appointed.");
                            break;
                        }
                    } else {
                        JOptionPane.showMessageDialog(frm, "Vacancy " +
vacancyNumber
                                + " belongs to PART TIME, use a vacancy that belongs to
FULL TIME");
                    }
                }
            }
            if (!vacancyInUse) {
                JOptionPane.showMessageDialog(frm, "The vacancy you provided does
not exist, try again");
            }
        }
    } catch (Exception exp1) {
        JOptionPane.showMessageDialog(frm, "Make sure to input proper values in
all the fields.");
```

```
        }
    }

    if (click.getSource() == btnAddVacancyPT) {
        try {
            vacancyNumber = Integer.parseInt(txtVacancyNumberPT.getText());
            designation = txtDesignationPT.getText();
            jobType = (cmbJobTypePT.getSelectedItem()).toString();
            workingHour = Double.parseDouble(txtWorkingHoursPT.getText());
            wagePerHr = Double.parseDouble(txtWagePerHrPT.getText());
            shift = (cmbShiftPT.getSelectedItem()).toString();
            vacancyInUse = false;
            if (designation.equals("")) {
                JOptionPane.showMessageDialog(frm, "Make sure to input proper values
in all the fields.");
            } else {
                for (StaffHire obj : staffList) {
                    if (obj.getVacancyNumber() == vacancyNumber) {
                        vacancyInUse = true;
                        break;
                    }
                }
                if (vacancyInUse == false) {
                    PartTimeStaffHire obj = new PartTimeStaffHire(vacancyNumber,
designation, jobType, workingHour,
                            wagePerHr, shift);
                    staffList.add(obj);
                    JOptionPane.showMessageDialog(frm,
                            "Vacancy " + vacancyNumber + " added Total: " + staffList.size());
```

```
                } else {
                    JOptionPane.showMessageDialog(frm,
                            "Vacancy " + vacancyNumber + " already in use Total: " +
staffList.size());
                }
            }
        } catch (Exception exp2) {
            JOptionPane.showMessageDialog(frm, "Make sure to input proper values in
all the fields.");
        }
    }

    if (click.getSource() == btnAppointStaffPT) {
        try {
            vacancyNumber = Integer.parseInt(txtVacancyNumber2PT.getText());
            staffName = txtStaffNamePT.getText();
            joinedOn = txtJoiningDatePT.getText();
            qualification = txtQualificationPT.getText();
            appointedBy = txtAppointedByPT.getText();
            joined = false;
            vacancyInUse = false;
            if (staffName.equals("") || joinedOn.equals("") || qualification.equals("") ||
appointedBy.equals("")) {
                JOptionPane.showMessageDialog(frm, "Make sure to input proper values
in all the fields.");
            } else {
                for (StaffHire obj : staffList) {
                    if (obj.getVacancyNumber() == vacancyNumber) {
                        vacancyInUse = true;
                        if (obj instanceof PartTimeStaffHire) {
```

```
                    PartTimeStaffHire call = (PartTimeStaffHire) obj;
                    if (call.getJoined(joined) == true) {
                        JOptionPane.showMessageDialog(frm, "Staff has been already
hired");
                    } else {
                        call.hirePartTimeStaff(staffName, joinedOn, qualification,
appointedBy);
                        JOptionPane.showMessageDialog(frm, "Staff successfully
appointed.");
                        break;
                    }
                } else {
                    JOptionPane.showMessageDialog(frm, "Vacancy " +
vacancyNumber
                        + " belongs to FULL TIME, use a vacancy that belongs to
PART TIME");
                }
            }
        }
        if (!vacancyInUse) {
            JOptionPane.showMessageDialog(frm, "The vacancy you provided does
not exist, try again");
        }
    }
} catch (Exception exp3) {
    JOptionPane.showMessageDialog(frm, "Make sure to input proper values in
all the fields.");
}
}


if (click.getSource() == btnTerminate) {
```

```
        try {
            terminationVacancy = Integer.parseInt(txtTerminationVacancy.getText());
            joined = true;
            terminated = false;
            vacancyInUse = false;
            for (StaffHire obj : staffList) {
                if (obj.getVacancyNumber() == terminationVacancy) {
                    vacancyInUse = true;


                    if (obj instanceof PartTimeStaffHire) {
                        PartTimeStaffHire call = (PartTimeStaffHire) obj;
                        if (call.getTerminated(terminated) == false && call.getJoined(joined)
== true) {//a staff who has joined but not terminated can be
                                                                        //terminated
                            call.terminateStaff();
                            JOptionPane.showMessageDialog(frm, "Staff terminated");
                        } else {
                            JOptionPane.showMessageDialog(frm,
                                    "No staff has been hired to Vacancy " + terminationVacancy
                                    + " or does not belong to PART TIME or has been terminated
");
                        }
                    } else {
                        JOptionPane.showMessageDialog(frm, "No staff has been hired to
Vacancy "
                                + terminationVacancy + " or does not belong to PART TIME or
has been terminated ");
                    }
                }
            }
```

```
            if (!vacancyInUse) {
                JOptionPane.showMessageDialog(frm, "The vacancy you provided does
not exist, try again");
            }
        } catch (Exception exp4) {
            JOptionPane.showMessageDialog(frm, "Make sure to input proper values in
all the fields.");
        }
    }


    if (click.getSource() == btnDisplay) {
        if (staffList.size() == 0) {
            JOptionPane.showMessageDialog(frm, "No data to show");
        } else {
            for (StaffHire obj : staffList) {
                if (obj instanceof FullTimeStaffHire) {
                    System.out.println("*_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-*");
                    System.out.println("\n" + "Full Time Staff Details" + "\n");
                    System.out.println("*_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-*");
                    FullTimeStaffHire call = (FullTimeStaffHire) obj;
                    call.displayInfo();
                }
                if (obj instanceof PartTimeStaffHire) {
                    System.out.println("*_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-*");
                    System.out.println("\n" + "Part Time Staff Details" + "\n");
                    System.out.println("*_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-*");
                    PartTimeStaffHire call = (PartTimeStaffHire) obj;
                    call.displayInfo();
                }
```

```
            }
        }
    }
  }
}
```

8.2 Appendix 2

Class diagrams

Simplified class diagram for the project

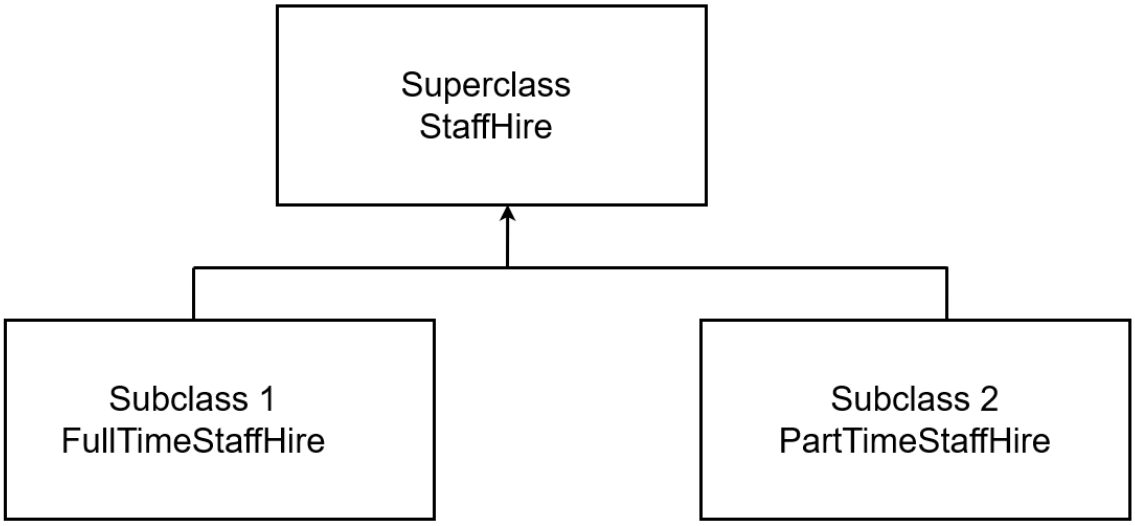

Figure: simplified class diagram for the project

Class diagram for StaffHire

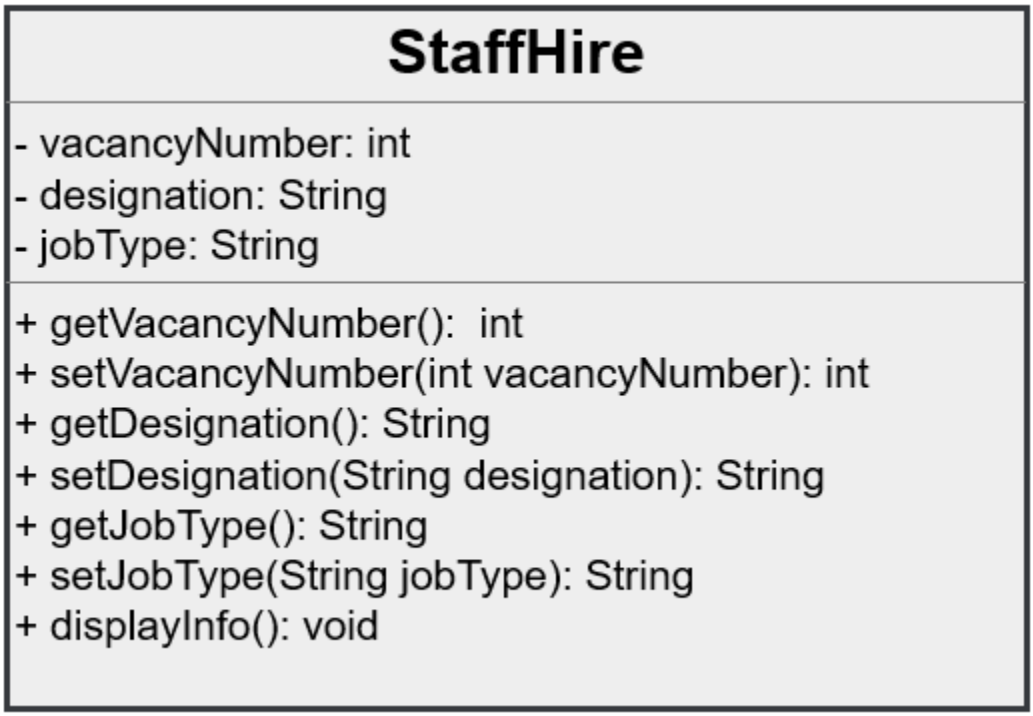

Figure: StaffHire class diagram

Class diagram for FullTimeStaffHire

Figure: FullTimeStaffHire class diagram

Class diagram for PartTimeStaffHire

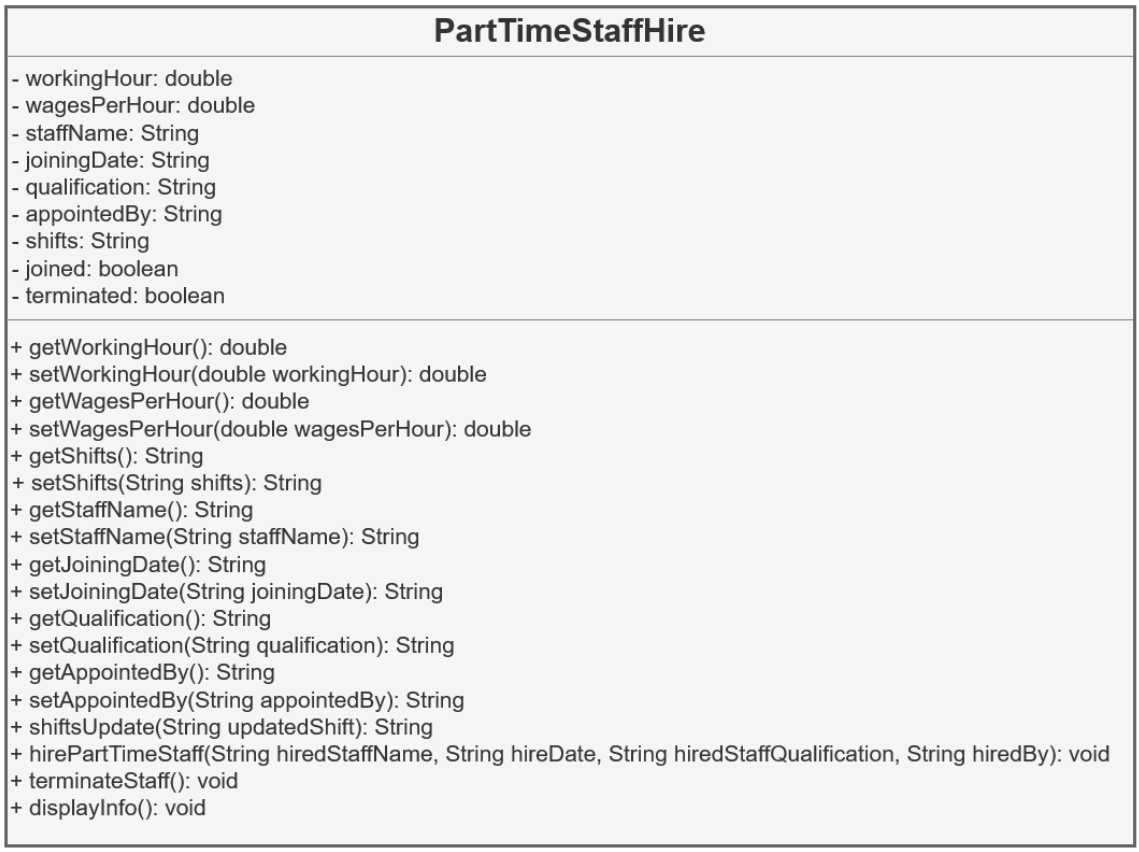

Figure: PartTimeStaffHire class diagram

Actual class diagram for the project

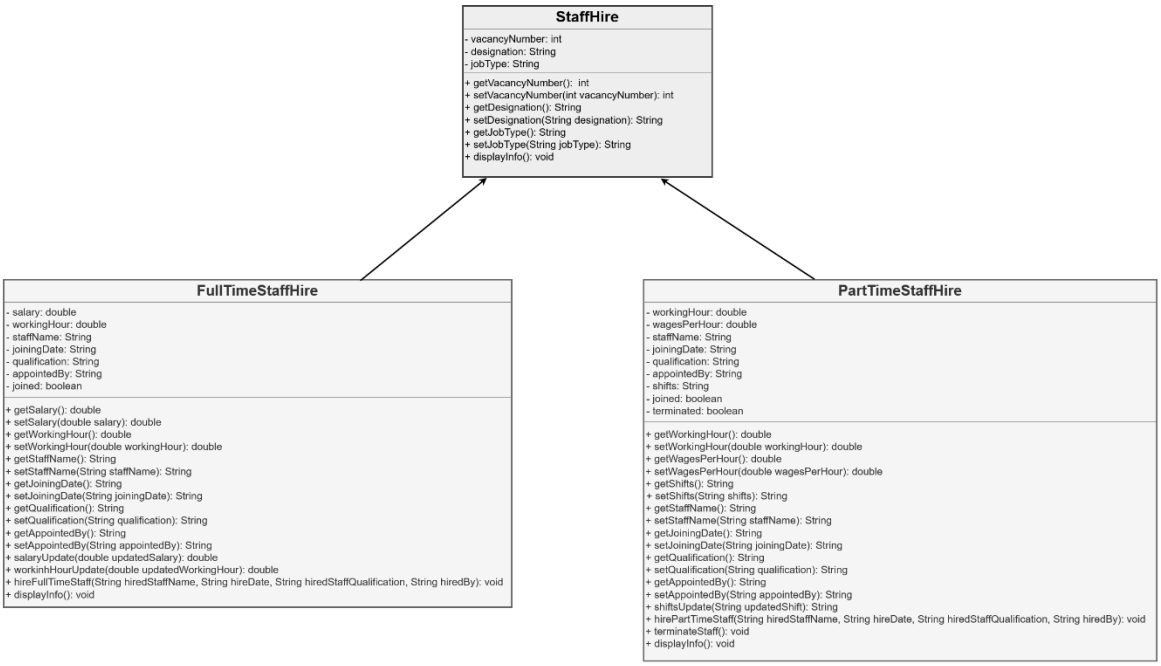

Figure: Actual class diagram for the project

Pseudocode

Pseudocode for StaffHire

```
DO
    Public class StaffHire {
    DO
    Private int vacancyNumber;
    Private String designation;
    Private String jobType;
    END DO
    DO
        Public StaffHire (vacancyNumber, designation, jobType){
        DO
            This.vacancyNumber = vacancyNumber;
            This.designation = designation;
            This.jobType = jobType;
        END DO
    }
    END DO
    DO
        Public int getVacancyNumber () {
        DO
            Return this.vacancyNumber;
        END DO
    }
    END DO
    DO
        Public void setVacancyNumber (int vacancyNumber) {
        DO
            This.jobType=jobType;
```

```
            END DO
        }
        END DO
        DO
            Public String getDesignation () {
            DO
                Return this.designation;
            END DO
        }
        END DO
        DO
            Public void setDesignation (String designation) {
            DO
                This.designation= designation;
            END DO
        }
        END DO
        DO
            Public int getJobType () {
            DO
                Return this.jobType;
            END DO
        }
        END DO
        DO
            Public void setJobType (String jobType) {
            DO
                This.jobType=jobType;
```

```
            END DO
        }
        END DO
        DO
            Public void displayInfo () {
            DO
                PRINT "\n" + "The vacancy number for the job is: "+
                getVacancyNumber ();
                PRINT "The designation for the job is: "+ getDesignation ();
                PRINT "The job type is: "+ getJobType ();
            END DO
            }
        END DO
}
END DO
```

Pseudocode for FullTimeStaffHire

```
DO
    Public class FullTimeStaffHire extends StaffHire {
    DO
        Private double salary;
        Private double workingHour;
        Private String staffName;
        Private String joiningDate;
        Private String qualification;
        Private String appointedBy;
        Private boolean joined;
    END DO
    DO
        Public FullTimeStaffHire (vacancyNumber, designation, jobType, salary,
        workingHour){
        DO
            Super (vacancyNumber, designation, jobType);
            This.salary = salary;
            This.workingHour = workingHour;
            staffName = “”;
            joiningDate = “”;
            qualification = “”;
            appointedBy = “”;
            joined = false;
        END DO
    }
    END DO
    DO
        Public double getSalary () {
```

```
DO
        Return this.Salary;
END DO
}
END DO
DO
        Public void setSalary (double salary) {
        DO
                This.salary=salary;
        END DO
        }
END DO
DO
        Public double getWorkingHour () {
DO
        Return this.workingHour;
END DO
}
END DO
DO
        Public void setworkingHour (double workingHour) {
        DO
                This.workingHour=workingHour;
        END DO
        }
END DO
DO
        Public String getStaffName () {
```

```
DO
        Return this.staffName;
END DO
}
END DO
DO
        Public void setStaffName (String staffName) {
        DO
                This.staffName=staffName;
        END DO
        }
END DO
DO
        Public String getJoiningDate () {
DO
        Return this.joiningDate;
END DO
}
END DO
DO
        Public void setJoiningDate (String joiningDate) {
        DO
                This.joiningDate=joiningDate;
        END DO
        }
END DO
DO
        Public String getQualification () {
```

```
DO
        Return this.qualification;
END DO
}
END DO
DO
        Public void setQualification (String qualification) {
        DO
                This.qualification = qualification;
        END DO
        }
END DO
DO
        Public String getAppointedBy () {
DO
        Return this.appointedBy;
END DO
}
END DO
DO
        Public void setAppointedBy (String appointedBy) {
        DO
                This.appointedBy = appointedBy;
        END DO
        }
END DO
DO
        Public double salaryUpdate (double updatedSalary) {
```

```
        DO
            IF (joined == true)
            DO
                PRINT "\n" + "The salary is already set and the staff has
                joined"
            END DO
            ELSE
            DO
                This.salary = updatedSalary;
            END DO
            END IF
            DO
                Return updatedSalary;
            END DO
        END DO
    }
    END DO
    DO
        Public double workingHourUpdate (double updatedWorkingHour) {
        DO
            DO
                This.workingHour = updatedWorkingHour;
            END DO
            DO
                Return updatedWorkingHour;
            END DO
        END DO
    }
    END DO
```

```
DO
    Public void hireFullTimeStaff (String hiredStaffName, String hireDate,
    String hiredStaffQualification, String hiredBy) {
    DO
        IF (joined == true)
        DO
            PRINT "\n" + "The staff has already joined";
            PRINT "Name of staff: "+ getStaffName ();
            PRINT getStaffName()+ "'s joining date:" + getJoiningDate();
        END DO
        ELSE
        DO
            This.staffName = hiredStaffName;
            This.joiningDate = hireDate;
            This.qualification = hiredStaffQualification;
            This.appointedBy = hiredBy;
            joined = true;
        END DO
        END IF
    END DO
}
END DO
DO
    Public void displayInfo () {
    DO
        PRINT "\n" + "The details of the job are:"
        DO
            Super.displayInfo ();
        END DO
```

```
            END DO
            IF (joined == true)
            DO
                PRINT "\n" + "The details of the staff are:" + "\n";
                PRINT "The name of the staff is:" + getStaffName ();
                PRINT getStaffName () + "'s salary is:" + getSalary ();
                PRINT getStaffName () +"'s working hours are:"+getWorkingHour();
                PRINT getStaffName () + "joined in:" + getJoiningDate ();
                PRINT getStaffName () "'s qualifications are:" + getQualification ();
                PRINT getStaffName () "wad appointed by:" + getAppointedBy ();
            END DO
            ELSE
            DO
                PRINT "\n" + "No staff has been hired for the job.";
            END DO
            END IF
        }
        END DO
}
END DO
```

Pseudocode for PartTimeStaffHire

```
DO
        Public class PartTimeStaffHire extends StaffHire {
        DO
            Private double workingHour;
            Private double wagesPerHour;
            Private String staffName;
            Private String joiningDate;
```

```
        Private String qualification;
        Private String appointedBy;
        Private boolean joined;
        Private boolean terminated;
    END DO
    DO
        Public FullTimeStaffHire (vacancyNumber, designation, jobType, salary,
        workingHour, wagesPerHour, shifts) {
        DO
            Super (vacancyNumber, designation, jobType);
            This.workingHour = workingHour;
            This.wagesPerHour = wagesPerHour;
            This.shifts = shifts;
            staffName = “”;
            joiningDate = “”;
            qualification = “”;
            appointedBy = “”;
            joined = false;
            terminated = false;
        END DO
    }
    END DO
    DO
        Public double getWorkingHour () {
    DO
        Return this.workingHour;
    END DO
    }
    END DO
```

```
DO
        Public void setWorkingHour (double workingHour) {
        DO
                This.workingHour = workingHour;
        END DO
        }
END DO
DO
        Public double getWagesPerHour () {
DO
        Return this.wagesPerHour;
END DO
}
END DO
DO
        Public void setWagesPerHour (double wagesPerHour) {
        DO
                This.wagesPerHour=wagesPerHour;
        END DO
        }
END DO
DO
        Public String getShifts () {
DO
        Return this.shifts;
END DO
}
END DO
```

```
DO
        Public void setShifts (String shifts) {
        DO
                This.shifts = shifts;
        END DO
        }
END DO

DO
        Public String getStaffName () {
DO
        Return this.staffName;
END DO
}
END DO
DO
        Public void setStaffName (String staffName) {
        DO
                This.staffName=staffName;
        END DO
        }
END DO
DO
        Public String getJoiningDate () {
DO
        Return this.joiningDate;
END DO
}
```

```
END DO
DO
        Public void setJoiningDate (String joiningDate) {
        DO
                This.joiningDate=joiningDate;
        END DO
        }
END DO
DO
        Public String getQualification () {
DO
        Return this.qualification;
END DO
}
END DO
DO
        Public void setQualification (String qualification) {
        DO
                This.qualification = qualification;
        END DO
        }
END DO
DO
        Public String getAppointedBy () {
DO
        Return this.appointedBy;
END DO
}
```

```
END DO
DO
    Public void setAppointedBy (String appointedBy) {
    DO
        This.appointedBy = appointedBy;
    END DO
    }
END DO
DO
    Public double shiftsUpdate (String updatedShift) {
    DO
        IF (joined == true)
        DO
            PRINT “\n” + “The shifts are set and the staff has joined”
        END DO
        ELSE
        DO
            This.shifts = updatedShift;
        END DO
        END IF
        DO
            Return updatedShift;
        END DO
    END DO
}
END DO
DO
    Public void hirePartTimeStaff (String hiredStaffName, String hireDate,
    String hiredStaffQualification, String hiredBy) {
```

```
        DO
            IF (joined == true)
            DO
                PRINT “\n” + “The staff has already joined”;
                PRINT “Name of staff: “+ getStaffName ();
                PRINT getStaffName()+ “’s joining date:“ + getJoiningDate();
            END DO
            ELSE
            DO
                This.staffName = hiredStaffName;
                This.joiningDate = hireDate;
                This.qualification = hiredStaffQualification;
                This.appointedBy = hiredBy;
                joined = true;
                terminated = false;
            END DO
            END IF
        END DO
    }
    END DO
    DO
        Public void terminateStaff () {
        DO
            IF (terminated == true) {
            DO
                PRINT “\n” + “”The staff has already been terminated”;
            END DO
            }
```

```
                ELSE {
                DO
                        This.staffName = “”;
                        This.joiningDate = “”;
                        This.appointedBy = “”;
                        joined = false;
                        terminated = true;
                }
                END DO
                END IF
        }
        END DO
        DO
                Public void displayInfo () {
                DO
                        PRINT “\n” + “The details of the job are:”
                        DO
                                Super.displayInfo ();
                        END DO
                END DO
                IF (joined == true)
                DO
                        PRINT “\n” + “The details of the staff are:” + “\n”;
                        PRINT “The name of the staff is:” + getStaffName ();
                        PRINT getStaffName () + “’s wage per hour is:” +
                        getWagesPerHour ();
                        PRINT getStaffName () +“’s working hours are:”+getWorkingHour();
                        PRINT getStaffName () + “joined in:” + getJoiningDate ();
                        PRINT getStaffName () “’s qualifications are:” + getQualification ();
```

```
                PRINT getStaffName () “wad appointed by:” + getAppointedBy ();
                PRINT getStaffName () “’s income per day is:” +
                ( getWorkingHour()*getWorkingHour());
            END DO
            ELSE
            DO
                PRINT “\n” + “No staff has been hired for the job.”;
            END DO
            END IF
        }
        END DO
}
END DO
```

Method descriptions

Methods in StaffHire

StaffHire

This method is a constructor. It has int vacancyNumber, String designation and String jobType as parameters.

Accessors and Mutators

Accessor methods are the get method in StaffHire. A property of the object is returned by this method. It may be int, String, double etc. They return value of a private field.

- getVacancyNumber () returns int value
- getDesignation () returns String value
- getJobType () returns String value

Mutator methods are the set methods. They are void and do not return value. They accept a parameter and fix a value for a private field.

- setVacancyNumber () sets int value to vacancyNumber
- setDesignation () sets String value to designation
- setJobType () sets String value to jobType

displayInfo ()

This method is used to display the details of the job i.e. vacancyNumber, designation and jobType.

Methods in FullTimeStaffHire

FullTimeStaffHire

This is a constrictor. It has int vacancyNumber, String designation, String jobType, double salary, double workingHour as parameters. It uses super keyword to call from StaffHire. Some fields are set empty and joined is set to false in this method.

Accessors and Mutators

Accessor methods are the get method in StaffHire. A property of the object is returned by this method. It may be int, String, double etc. They return value of a private field. The getter methods return the values which they are specified. This can be seen in the class diagram.

Mutator methods are the set methods. They are void and do not return value. They accept a parameter and fix a value for a private field. The setter methods are specified to fix a certain value to a certain field. This can be seen in the class diagram.

salaryUpdate

This method returns a double value. Condition joined is checked. If joined is true, a message saying salary is set is displayed but if joined is false, the value of salary is updated.

workingHourUpdate
This method returns a double value. The value of workingHour is updated.

hireFullTimeStaff
This is a method to hire staff. If joined is true a suitable message is displayed along with staff name and joining date else, staffName, joiningDate, qualification and appointedBy are updated and joined is set to true.

displayInfo
This method displays the job details by calling displayInfo method from StaffHire and checks joined. If joined is true all the details of the staff are displayed otherwise, a message saying no staff has been hired is printed.

Methods in PartTimeStaffHire

PartTimeStaffHire
This is a constrictor. It has int vacancyNumber, String designation, String jobType, double salary, double workingHour, double wagesPerHour and String shifts as parameters. It uses super keyword to call from StaffHire. Some fields are set empty and joined and terminated are set to false in this method.

Accessors and Mutators
Accessor methods are the get method in StaffHire. A property of the object is returned by this method. It may be int, String, double etc. They return value of a private field. The getter methods return the values which they are specified. This can be seen in the class diagram.

Mutator methods are the set methods. They are void and do not return value. They accept a parameter and fix a value for a private field. The setter methods are specified to fix a certain value to a certain field. This can be seen in the class diagram.

shiftsUpdate
This method returns a String value. Condition joined is checked. If joined is true, a message saying shifts are set is displayed but if joined is false, the value of shifts is updated.

hirePartTimeStaff
This is a method to hire staff. If joined is true a suitable message is displayed along with staff name and joining date else, staffName, joiningDate, qualification and appointedBy are updated, joined is set to true and terminated is set to false.

terminateStaff
This is a method to terminate staff. First, terminated is checked if it is true, a message saying the staff has been terminated is displayed else staffName, joiningDate, qualification and appointedBy are set to empty and joined is false and terminated true.

displayInfo
This method displays the job details by calling displayInfo method from StaffHire and checks joined. If joined is true all the details of the staff are displayed otherwise, a message saying no staff has been hired is printed.

Errors and corrections

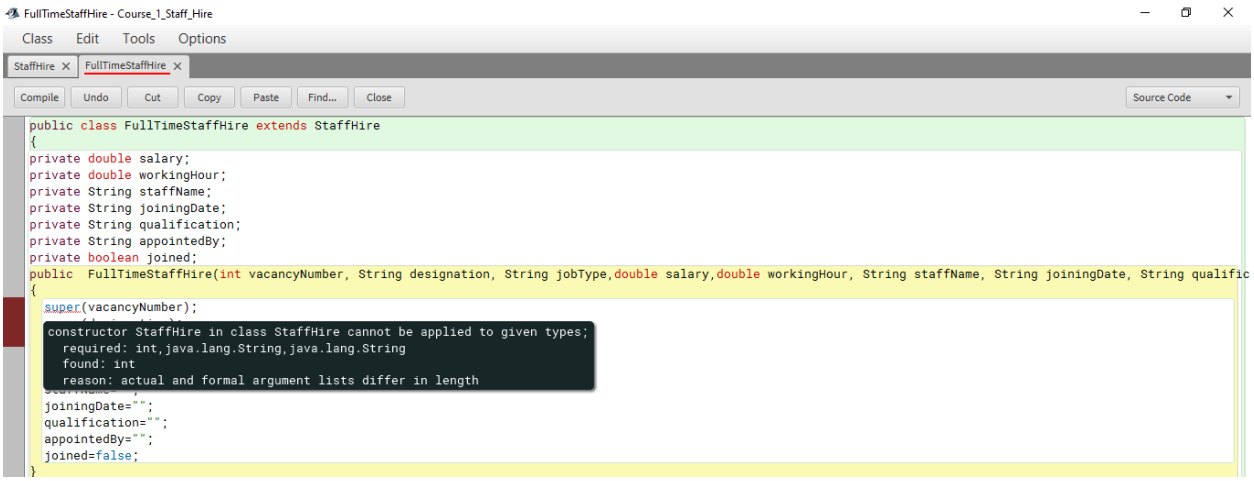

Figure: Error 1

Error 1 was a syntax error

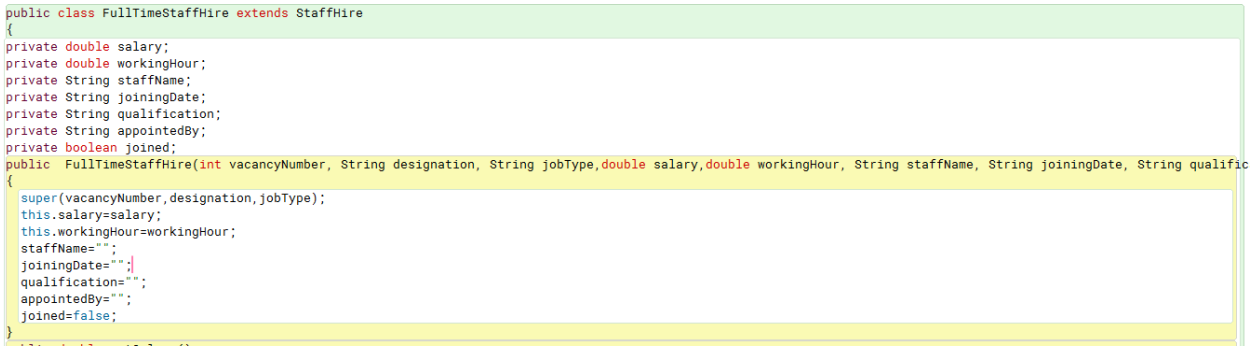

Figure: Corrected error 1

The error was easily corrected with the help of a correct syntax.

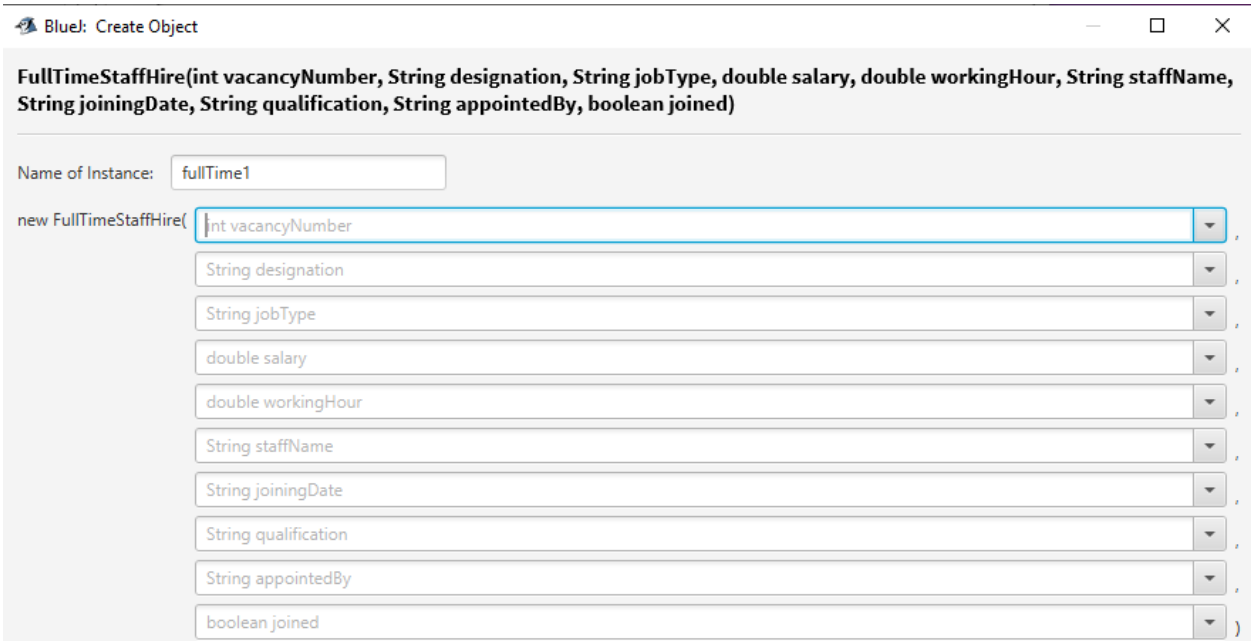

Figure: Error 2

This error was caused by giving more than required parameters.

```
public class FullTimeStaffHire extends StaffHire
{
private double salary;
private double workingHour;
private String staffName;
private String joiningDate;
private String qualification;
private String appointedBy;
private boolean joined;
public FullTimeStaffHire(int vacancyNumber, String designation, String jobType,double salary,double workingHour, String staffName, String joiningDate, String qualifi
{
  super(vacancyNumber,designation,jobType);//Super is a reference variable which in this case refers to the mentioned variables in StaffHire.
  this.salary=salary;
  this.workingHour=workingHour;
  staffName="";
  joiningDate="";
  qualification="";
  appointedBy="";
  joined=false;
}
```

Figure 30: More than required parameters are given

```
public FullTimeStaffHire(int vacancyNumber, String designation, String jobType,double salary,double workingHour)
{
  super(vacancyNumber,designation,jobType);//Super is a reference variable which in this case refers to the mentioned variables in StaffHire.
  this.salary=salary;
  this.workingHour=workingHour;
  staffName="";
  joiningDate="";
  qualification="";
  appointedBy="";
  joined=false;
}
```

Figure: Correction for error 2

The error was corrected by giving only the required parameters

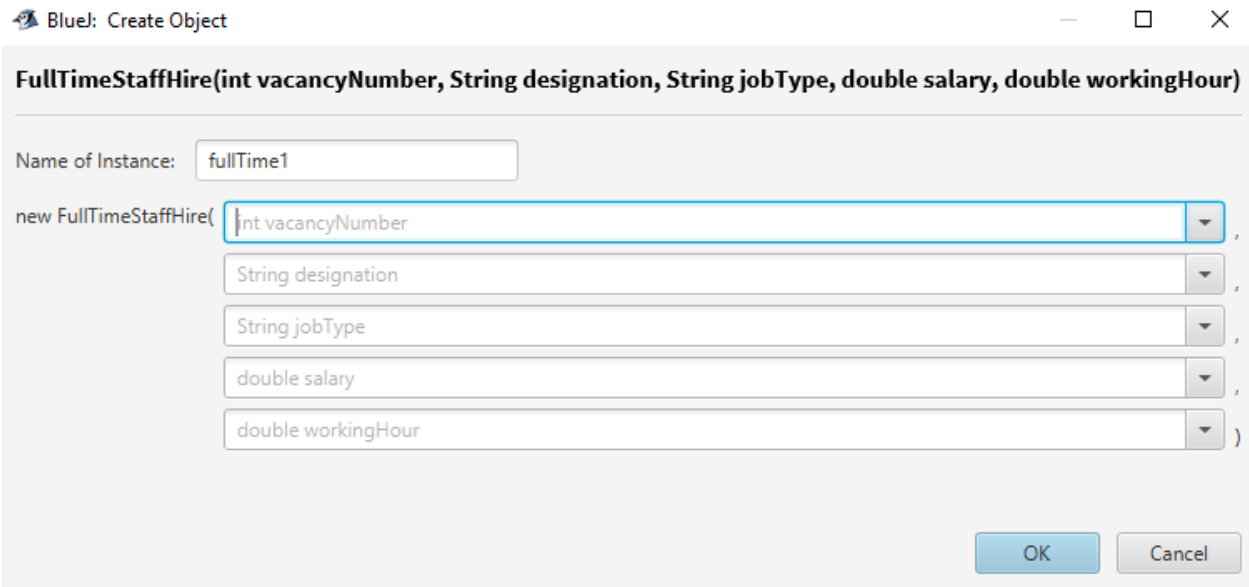

Figure: Result after correcting error 2

```
public String shiftsUpdate(String updatedShift) {
    if (joined = true) {
        System.out.println("\n" + "The shifts are set and the staff has joined");
    } else {
        this.shifts = updatedShift;
    }
    return updatedShift; // Since the method is not void, a value must be returned, in this case it is the updated shift.
}
```

Figure: Error 3

Here, the way of comparing the value of joined is incorrect.

```
public String shiftsUpdate(String updatedShift) {
    if (joined == true) {
        System.out.println("\n" + "The shifts are set and the staff has joined");
    } else {
        this.shifts = updatedShift;
    }
    return updatedShift; // Since the method is not void, a value must be returned, in this case it is the updated shift.
}
```

Figure: Error 3 correction

Error is corrected by comparing the value properly.

Code for StaffHire

```
public class StaffHire {
    private int vacancyNumber;
    private String designation;
    private String jobType;

    public StaffHire(int vacancyNumber, String designation, String jobType) {
        this.vacancyNumber = vacancyNumber;
        this.designation = designation;
        this.jobType = jobType;
    }

    public int getVacancyNumber() {
        return this.vacancyNumber;
    }

    public void setVacancyNumber(int vacancyNumber) {
        this.vacancyNumber = vacancyNumber;
    }

    public String getDesignation() {
        return this.designation;
    }

    public void setDesignation(String designation) {
        this.designation = designation;
    }
```

```
    public String getJobType() {
        return this.jobType;
    }

    public void setJobType(String jobType) {
        this.jobType = jobType;
    }

    public void displayInfo() {
        System.out.println("\n" + "The vacancy number for the job is: " +
getVacancyNumber());
        System.out.println("The designation for the job is: " + getDesignation());
        System.out.println("The job type is: " + getJobType());
    }
}
```

Code for FullTimeStaffHire

```
public class FullTimeStaffHire extends StaffHire {
    private double salary;
    private double workingHour;
    private String staffName;
    private String joiningDate;
    private String qualification;
    private String appointedBy;
    private boolean joined;

    public FullTimeStaffHire(int vacancyNumber, String designation, String jobType, doub
le salary, double workingHour) {
```

```
        super(vacancyNumber, designation, jobType);
        this.salary = salary;
        this.workingHour = workingHour;
        staffName = "";
        joiningDate = "";
        qualification = "";
        appointedBy = "";
        joined = false;
    }

    public double getSalary() {
        return this.salary;
    }

    public void setSalary(double salary) {
        this.salary = salary;
    }

    public double getWorkingHour() {
        return this.workingHour;
    }

    public void setWorkingHour(double workingHour) {
        this.workingHour = workingHour;
    }

    public String getStaffName() {
        return this.staffName;
```

```
    }

    public void setStaffName(String staffName) {
        this.staffName = staffName;
    }

    public String getJoiningDate() {
        return this.joiningDate;
    }

    public void setJoiningDate(String joiningDate) {
        this.joiningDate = joiningDate;
    }

    public String getQualification() {
        return this.qualification;
    }

    public void setQualification(String qualification) {
        this.qualification = qualification;
    }

    public String getAppointedBy() {
        return this.appointedBy;
    }

    public void setAppointedBy(String appointedBy) {
        this.appointedBy = appointedBy;
```

```
    }

    public double salaryUpdate(double updatedSalary) {
        if (joined == true) {
            System.out.println("\n" + "The salary is already set and the staff has joined.");
        } else {
            this.salary = updatedSalary;

        }
        return updatedSalary;
    }

    public double workingHourUpdate(double updatedWorkingHour) {
            this.workingHour = updatedWorkingHour;
            return updatedWorkingHour;
    }

    public void hireFullTimeStaff(String hiredStaffName, String hireDate, String hiredStaff
Qualification,
            String hiredBy) {
        if (joined == true) {
            System.out.println("\n" + "The staff has already joined");
            System.out.println("Name of staff: " + getStaffName());
            System.out.println(getStaffName() + "'s joining date: " + getJoiningDate());
        } else {
            this.staffName = hiredStaffName;
            this.joiningDate = hireDate;
            this.qualification = hiredStaffQualification;
            this.appointedBy = hiredBy;
```

```
            joined = true;
        }
    }

    public void displayInfo() {
        System.out.println("\n" + "The details of the job are:");
        super.displayInfo();
        if (joined == true) {
            System.out.println("\n" + "The details of the staff are: " + "\n");
            System.out.println("The name of the staff is: " + getStaffName());
            System.out.println(getStaffName() + "'s salary is: " + getSalary());
            System.out.println(getStaffName() + "'s working hours are: " + getWorkingHour()
);
            System.out.println(getStaffName() + " joined in: " + getJoiningDate());
            System.out.println(getStaffName() + "'s qualifications are: " + getQualification());
            System.out.println(getStaffName() + " was appointed by: " + getAppointedBy());
        } else {
            System.out.println("\n" + "No staff has been hired for the job.");
        }
    }
}
```

Code for PartTimeStaffHire

```
public class PartTimeStaffHire extends StaffHire {
    private double workingHour;
    private double wagesPerHour;
    private String staffName;
    private String joiningDate;
    private String qualification;
```

```
    private String appointedBy;
    private String shifts;
    private boolean joined;
    private boolean terminated;

    public PartTimeStaffHire(int vacancyNumber, String designation, String jobType, dou
ble workingHour,
            double wagesPerHour, String shifts) {
        super(vacancyNumber, designation, jobType);
        this.workingHour = workingHour;
        this.wagesPerHour = wagesPerHour;
        this.shifts = shifts;
        staffName = "";
        joiningDate = "";
        qualification = "";
        appointedBy = "";
        joined = false;
        terminated = false;
    }

    public double getWorkingHour() {
        return this.workingHour;
    }

    public void setWorkingHour(double workingHour) {
        this.workingHour = workingHour;
    }

    public double getWagesPerHour() {
```

```
        return this.wagesPerHour;
    }

    public void setWagesPerHour(double wagesPerHour) {
        this.wagesPerHour = wagesPerHour;
    }

    public String getShifts() {
        return this.shifts;
    }

    public void setShifts(String shifts) {
        this.shifts = shifts;
    }

    public String getStaffName() {
        return this.staffName;
    }

    public void setStaffName(String staffName) {
        this.staffName = staffName;
    }

    public String getJoiningDate() {
        return this.joiningDate;
    }

    public void setJoiningDate(String joiningDate) {
```

```
        this.joiningDate = joiningDate;
    }

    public String getQualification() {
        return this.qualification;
    }

    public void setQualification(String qualification) {
        this.qualification = qualification;
    }

    public String getAppointedBy() {
        return this.appointedBy;
    }

    public void setAppointedBy(String appointedBy) {
        this.appointedBy = appointedBy;
    }

    public String shiftsUpdate(String updatedShift) {
        if (joined == true) {
            System.out.println("\n" + "The shifts are set and the staff has joined");
        } else {
            this.shifts = updatedShift;
        }
        return updatedShift;
    }
```

```
    public void hirePartTimeStaff(String hiredStaffName, String hireDate, String hiredStaf
fQualification,
            String hiredBy) {
        if (joined == true) {
            System.out.println("\n" + "The staff has already joined");
            System.out.println("Name of staff: " + getStaffName());
            System.out.println(getStaffName() + "'s joining date: " + getJoiningDate());
        } else {
            this.staffName = hiredStaffName;
            this.joiningDate = hireDate;
            this.qualification = hiredStaffQualification;
            this.appointedBy = hiredBy;
            joined = true;
            terminated = false;
        }
    }

    public void terminateStaff() {
        if (terminated == true) {
            System.out.println("\n" + "The staff has already been terminated");
        } else {
            this.staffName = "";
            this.joiningDate = "";
            this.qualification = "";
            this.appointedBy = "";
            joined = false;
            terminated = true;
        }
```

```
    }

    public void displayInfo() {
        System.out.println("\n" + "The details of the job are:");
        super.displayInfo();
        if (joined == true) {
            System.out.println("\n" + "The details of the staff are: " + "\n");
            System.out.println("The name of the staff is: " + getStaffName());
            System.out.println(getStaffName() + "'s wage per hour is: " + getWagesPerHour(
));
            System.out.println(getStaffName() + "'s working hours are: " + getWorkingHour()
);
            System.out.println(getStaffName() + " joined in: " + getJoiningDate());
            System.out.println(getStaffName() + "'s qualifications are: " + getQualification());
            System.out.println(getStaffName() + " was appointed by: " + getAppointedBy());
            System.out.println(getStaffName() + "'s income per day is: " + (getWorkingHour(
) * getWagesPerHour()));
        } else {
            System.out.println("\n" + "No staff has been hired for the job.");
        }
    }
}
```