LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CS4001NI-Programming (Computing Group)

Assessment Weightage & Type 30%
Individual Coursework-2

Year and Semester

2019 autumn / 2020 spring

Student Name: Nabin Gurung
London Met ID: 19031160
College ID: np01cp4a190265

Assignment Due Date:

Assignment Submission Date:

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

Table of Contents

Table of Figures

Table of Tables

1. Introduction:

Java was created by James Gosling, Patrick Naughton, Chris Warth, Ed Frank, and Mike Sheridan at Sun Microsystems. Sun Microsystems was bought by Oracle in 2010. Java is initially called as "Oak" in previously. Java was not intended to be source-code perfect with some other language. It is an article arranged programming language.

This is an individual coursework which was given by our teacher in week 20th and due date of the coursework was on week 24th. This is an individual coursework that contains 30% of our absolute module grades.

Actually, this coursework isn't that simple for us all, we attempted to talk with our seniors and instructors and got the thoughts. We reconsider our earlier weeks lecture slides from google classroom. We watched various instructional exercises, recordings from YouTube. Researched in various sites like w3schools, w3resouces, cybrary and some more. We need to develop a Graphical User Interface (GUI) for Full Time and Part Time Staff and hire or appoint them in a click of a button. This project has four different class parent class Staff Hire, FullTime class, Part Time class and INGNepal. INGNepal is the main class of the program as it contains all the Graphical User Interface of the program and ArrayList that stores datas of Staff. The program consists of different Java Swing and AWT classes, with different types of component.The class also contains actioPerformed method, to trigger and event when the button is pressed.

The reason behind building this program is to Hire and Appoint Staffs that may be Full Time or Part Time, with help of GUI, where information or datas of the staff is passed through GUI and stored in ArrayList.

2. Tools Used:

- **Blue J:**

A free Java Development Environment proposed for juveniles, used by millions around the globe. BlueJ is a headway circumstance that grants you to make Java programs quickly and with no issue. BlueJ was made to help the learning and instructing of thing orchestrated programming, and its game plan changes from other progress conditions in this manner. (Blue J, 2020)

- **MS Word:**

Microsoft Word is a word processor distributed by Microsoft. It is one of the workplace profitability applications remembered for the Microsoft Office suite. Initially created by Charles Simonyi and Richard Brodie, it was first discharged in 1983. Microsoft Word permits you to make proficient quality archives, reports, letters, and list of qualifications. In contrast to a plain content manager, Microsoft Word has highlights including spell check, punctuation check, content and text style arranging, HTML support, picture support, propelled page format, and that's only the tip of the iceberg. (Computer Hope, 2020)

- **Draw.io:**

Draw.io is an open source site for building applications, and the world's most normally utilized program based end-client outlining applications. It is free online outline making programming for flowcharts, process charts, affiliation diagrams, and UML, ER and system charts. (Draw.io, 2020)

3. Class Diagram:

INGNepal
desigNationForFullTime: String -jobTypeForFullTime: String -staffNameForFullTime: String -joiningDateForFullTime: String -qualificationForFullTime: String -appointedByForFullTime: String -wagesPerHourForFullTime; : String -desigNationForPartTime: String -jobTypeForPartTime: String -staffNameForPartTime: String -joiningDateForPartTime: String -appointedByForPartTime: String -radioButton_Morning: String -radioButton_Day : String -qualificationForPartTime: String -yearFullTime: String -monthFullTime: String -dayFullTime;: String -yearPartTime: String -monthPartTime: String -dayPartTime: String -workingShiftsRadioButton : String -jobTypeCheckBox_FT: String -jobTypeCheckBox_PT: String vacNumForPartTime: int -workingHourForPartTime: int -salaryForPartTime: int -wagesPerHourForPartTime: int -vacNumForFullTime: int -salaryForFullTime: int workingHourForFullTime: int

-frameStaffHire: JFrame

-myPanel: JPanel

-txtFieldForVacancyNum_FT: JTextField

-txtFieldForVacancyNum_2_FT: JTextField

-txtFieldForStaffNam_FT: JTextField

-txtFieldForDesignation_FT: JTextField

-txtFieldForWgsPerHr_FT: JTextField

-txtFieldForAppointedBy_FT: JTextField

-txtFieldForSalaryFT: JTextField

-txtFieldForVacancyNum_PT: JTextField

-txtFieldForVacancyNum_2_PT: JTextField

-txtFieldForStaffNam_PT: JTextField

-txtFieldForDesignation_PT: JTextField

-txtFieldForWgsPerHr_PT: JTextField

-txtFieldForAppointedBy_PT: JTextField

-txtFieldForSalaryPT: JTextField

-checkBoxFullTime_FT :JCheckBox

-checkBoxPartTime_FT :JCheckBox

-checkBoxFullTime_PT :JCheckBox

-checkBoxPartTime_PT :JCheckBox

-radioButton_Morning_PT :JRadioButton

-radioButton_Day_PT :JRadioButton

-comboBoxWorkingHour_FT :JComboBox

-comboBoxQualification_FT :JComboBox

-comboBoxWorkingHour_PT :JComboBox

-comboBoxQualification_PT :JComboBox

-cmbYear_FT :JComboBox

-cmbMonth_FT :JComboBox

-cmbDay_FT :JComboBox

-cmbYear_PT :JComboBox

-cmbMonth_PT :JComboBox

-cmbDay_PT :JComboBox

-menuBar :JMenuBar

-fileMenu :JMenu -exitMenu :JMenu -searchMenu :JMenu -aboutMenu :JMenu -addMenu :JMenu -appointMenu :JMenu -displayMenu :JMenu -openMenuItem :JMenuItem -saveMenuItem :JMenuItem -clearMenuItem :JMenuItem -exitMenuItem :JMenuItem -fullMenuItem :JMenuItem -partMenuItem :JMenuItem -fullTimeToAppoint :JMenuItem -partTimeToAppoint :JMenuItem -fullTimeDisplayItem :JMenuItem -partTimeDisplayItem :JMenuItem -btnClear :JButton -btnDisplay :JButton -btnAppointFullTime :JButton -btnAddFullTime :JButton -btnAppointPartTime :JButton -btnAddPartTime :JButton -btnTerminate :JButton -btnTest :JButton
+myGUI(): void +actionPerformed(Action Event e): void

Table 1: Class diagram for INGNepal

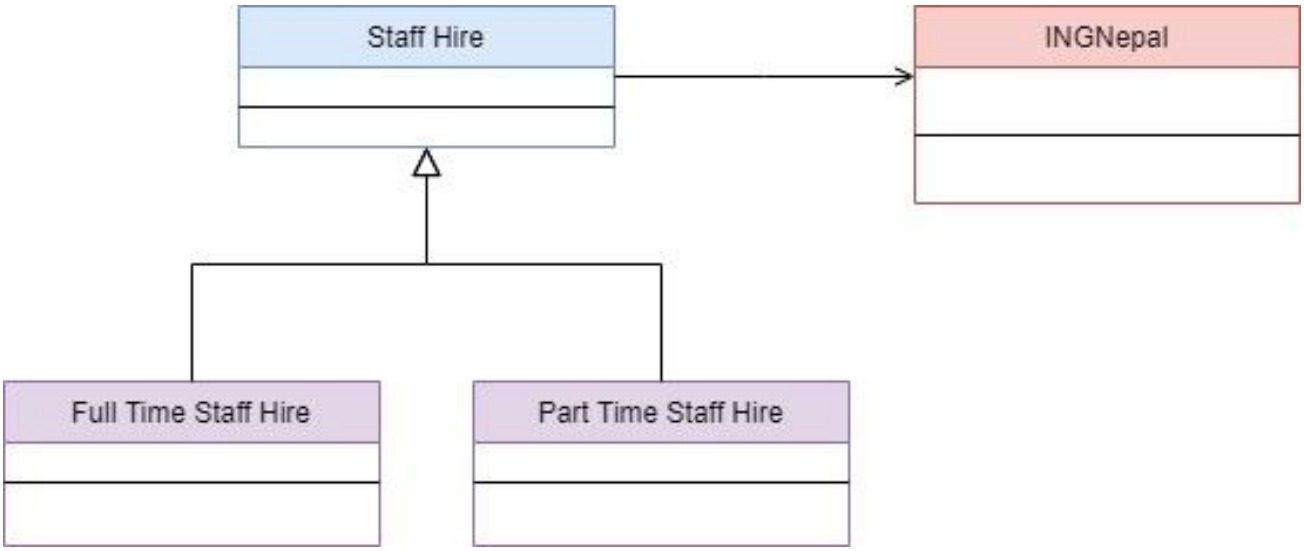

Table 2: Relational Class Diagram Between different classes

4. Method Description:

4.1 Method Description for INGNepal:

Method	Description
myGUI()	This is a non-parameterized instance method where all the codes of GUI are written.
actionPerformed(ActionEvent e)	This method is default method, made by JAVA, in which different coding is done, which is used to perform some action in a click of a button.

Table 3: Method Description for INGNepal

5. Testing:

5.1 Test-1: Test that the program can be compiled and run using the command prompt

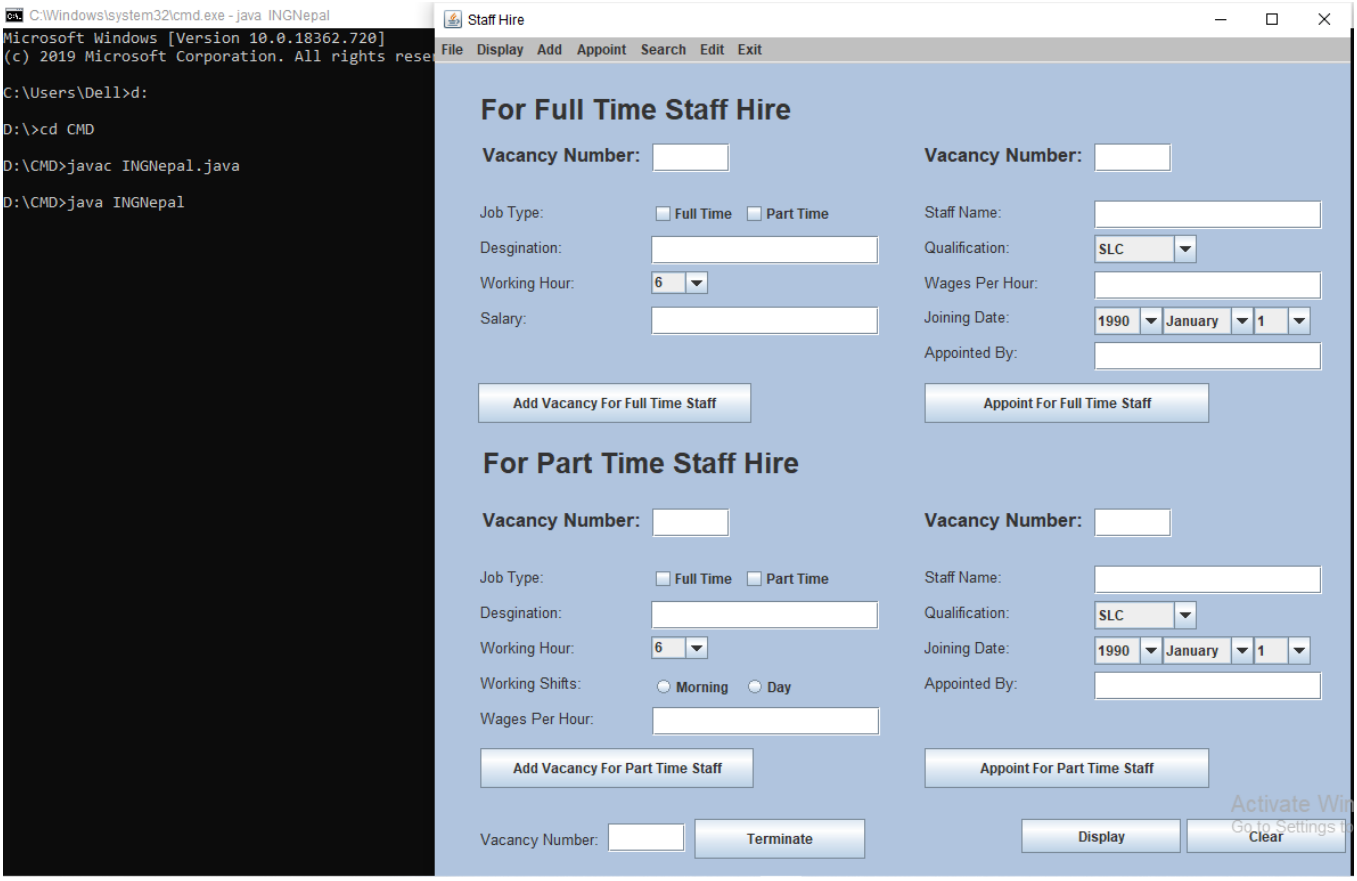

Figure 1: Test that the program can be compiled and run using the command prompt

5.2.1 Test-2: Test for Adding Vacancy for Full Time Staff

Objective	Adding Vacancy for Full Time Staff
Action	Different values are added to Full Time Staff by passing the values in textfields, CheckBox and ComboBox. - Vacancy Number: 10 - Job Type: Full Time - Designation: Optician - Working Hours: 9 - Salary: 125000
Expected Result	When the data is provided to textfield CheckBox and ComboBox, and "Add Vacancy for Full Time Staff" button is clicked, the data's should be added to Full Time Staff Hire Class.
Actual Result	When the value is provided to textfield CheckBox and ComboBox, and "Add Vacancy for Full Time Staff" button is clicked, the Vacancy Number is successfully added to Full Time Staff Hire Class.
Conclusion	Test is successful.

Table 4: Test for adding Vacancy in Full Time Staff

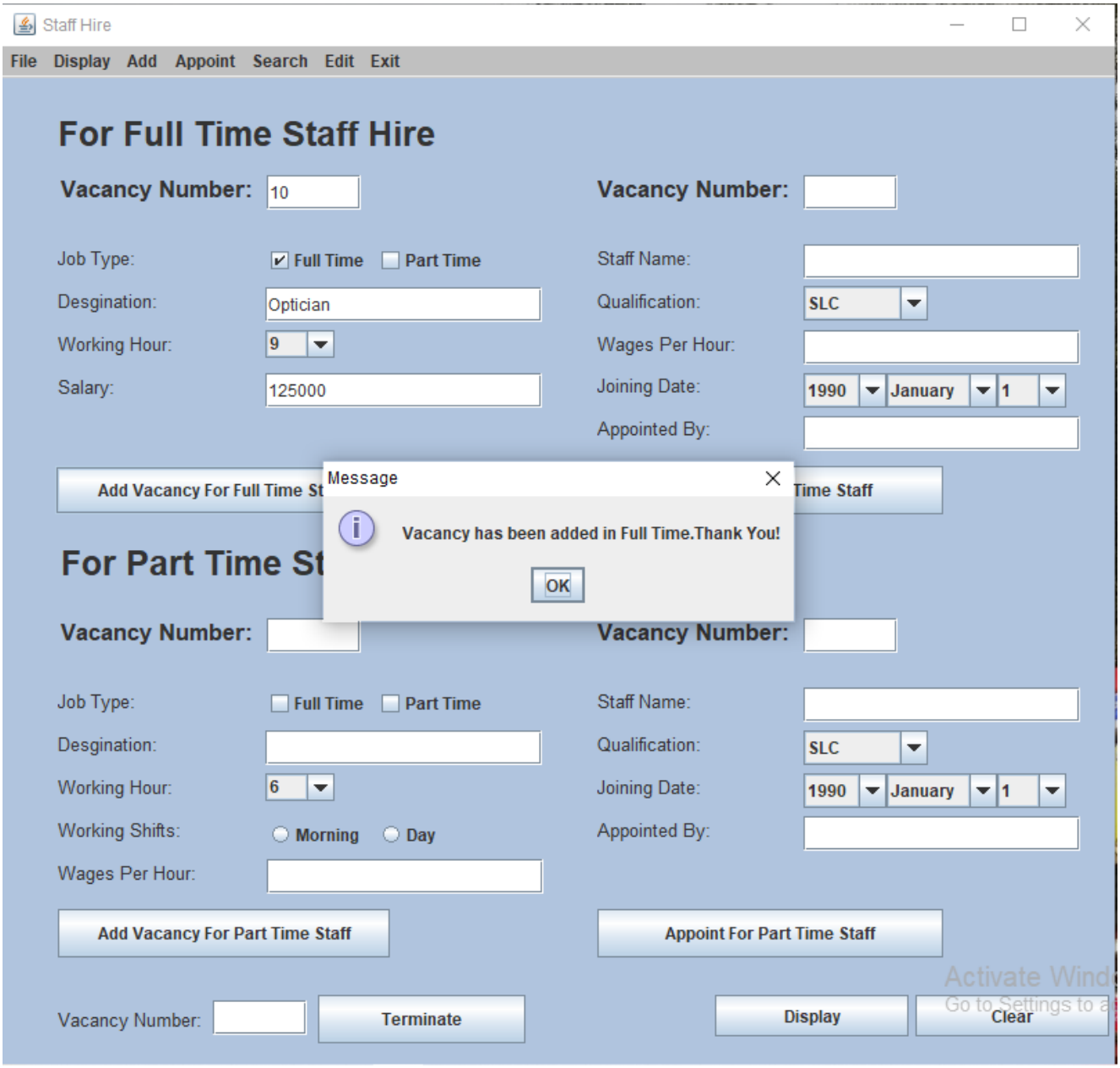

Figure 2: Screenshot of adding Vacancy for Full Time Staff

5.2.2 Test-2: Test for Adding Vacancy for Part Time Staff

Objective	Adding Vacancy for Part Time Staff
Action	Different values are added to Part Time Staff by passing the values in textfields, CheckBox and ComboBox. - Vacancy Number: 13 - Job Type: Part Time - Designation: Cloud Architect - Working Hours: 7 - Working Shifts: Day - Wages Per Hour: 345
Expected Result	When the data is provided to textfield CheckBox and ComboBox, and "Add Vacancy for Part Time Staff" button is clicked, the data's should be added to Part Time Staff Hire Class.
Actual Result	When the value is provided to textfield CheckBox and ComboBox, and "Add Vacancy for Full Time Staff" button is clicked, the data's are added to Part Time Staff Hire Class.
Conclusion	Test is successful.

Table 5: Test for adding Vacancy in Part Time Staff

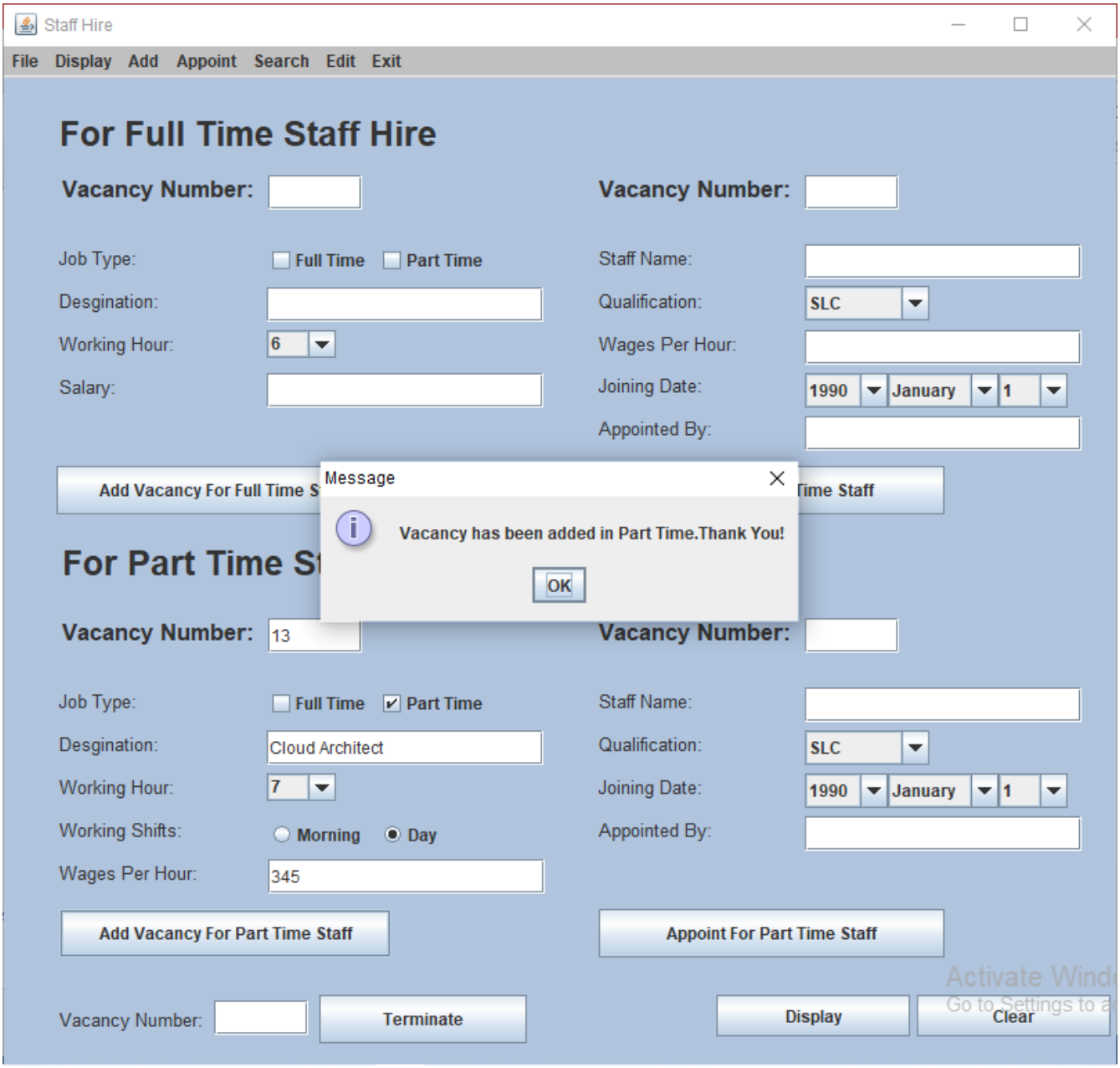

Figure 3: Screenshot of adding Vacancy for Part Time Staff

5.2.3 Test-2: Test for Appointing Full Time Staff

Objective	Appointing Full Time Staff
Action	Different values are added to Full Time Staff by passing the values in textfields, CheckBox and ComboBox. - Vacancy Number: 13 - Staff Name: Ashok Mishra - Qualification: Master - Wages Per Hour: 450 - Joining Date:1995/June/11 - Appointed By: Rabikesh Adhikari
Expected Result	When the data is provided to textfield CheckBox and ComboBox, and “Appoint for Full Time Staff” button is clicked, the data’s should be added to Full Time Staff Hire Class and then appointed.
Actual Result	When the value is provided to textfield CheckBox and ComboBox, and “Appoint for Full Time Staff” button is clicked, the data’s are added to Full Time Staff Hire Class and then appointed.
Conclusion	Test is successful.

Table 6: Test for Appointing Full Time Staff

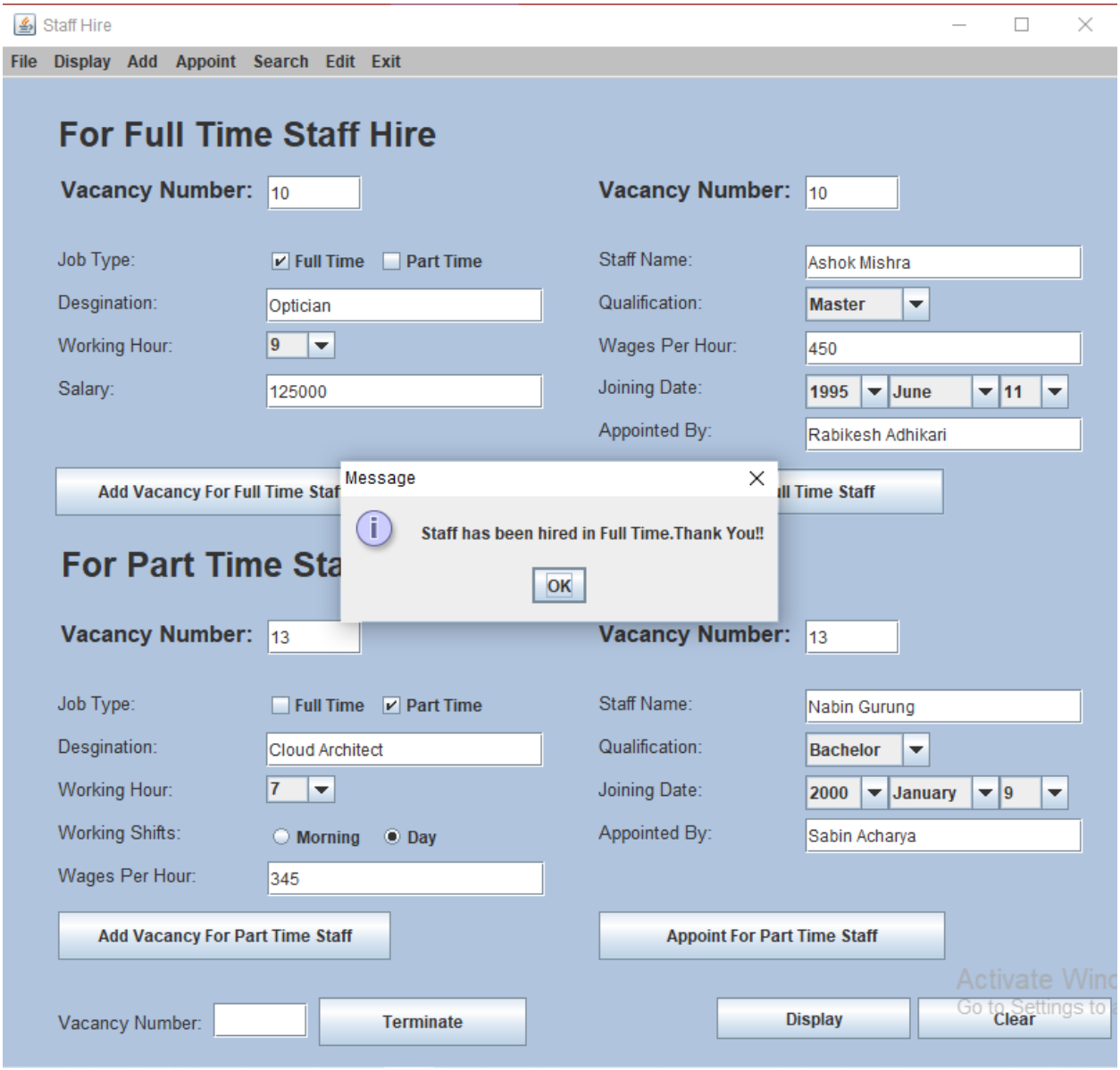

Figure 4: Screenshot of appointing Full Time Staff

5.2.4 Test-2: Test for Appointing Part Time Staff

Objective	Appointing Part Time Staff
Action	Different values are added to Full Time Staff by passing the values in textfields, CheckBox and ComboBox. - Vacancy Number: 13 - Staff Name: Nabin Gurung - Qualification: Bachelor - Joining Date:2000/January/1 - Appointed By: Sabin Acharya
Expected Result	When the data is provided to textfield CheckBox and ComboBox, and “Appoint for Part Time Staff” button is clicked, the data’s should be added to Part Time Staff Hire Class and then appointed.
Actual Result	When the value is provided to textfield CheckBox and ComboBox, and “Appoint for Part Time Staff” button is clicked, the data’s are added to Part Time Staff Hire Class and then appointed.
Conclusion	Test is successful.

Table 7: Test for Appointing Part Time Staff

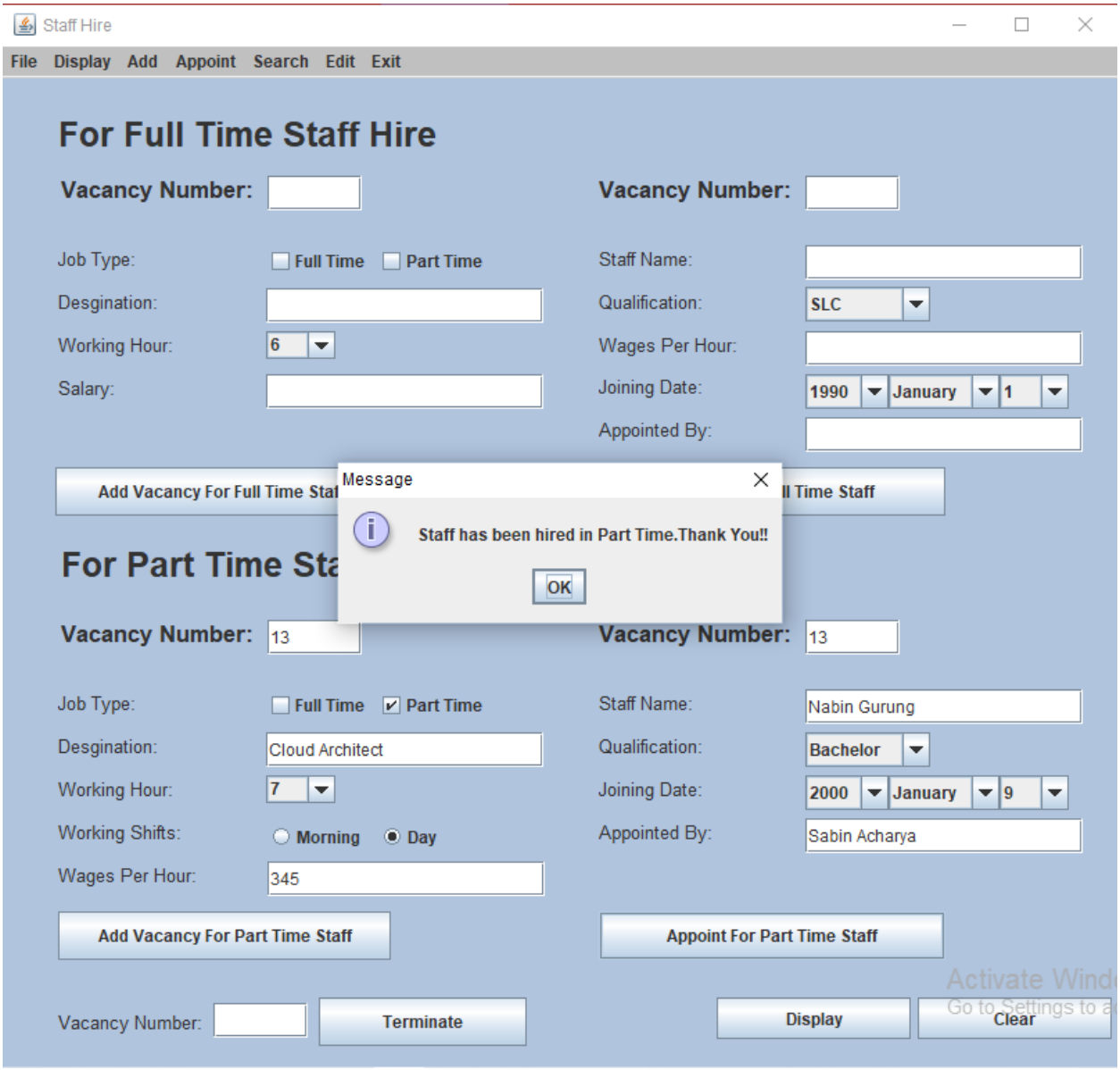

Figure 5: Screenshot of appointing Part Time Staff

5.2.5 Test-2: Test for Terminating Part Time Staff

Objective	Terminating Part Time Staff
Action	Vacancy Number, which is already added in Part Time Staff Class, is provided in Text Field - Vacancy Number: 13
Expected Result	When the data is provided to textfield, and "Terminate" button is clicked, the data's should be Terminated
Actual Result	When the data is provided to textfield, and "Terminate" button is clicked, the data's is Terminated
Conclusion	Test is successful.

Table 8: Test for Terminating Part Time Staff

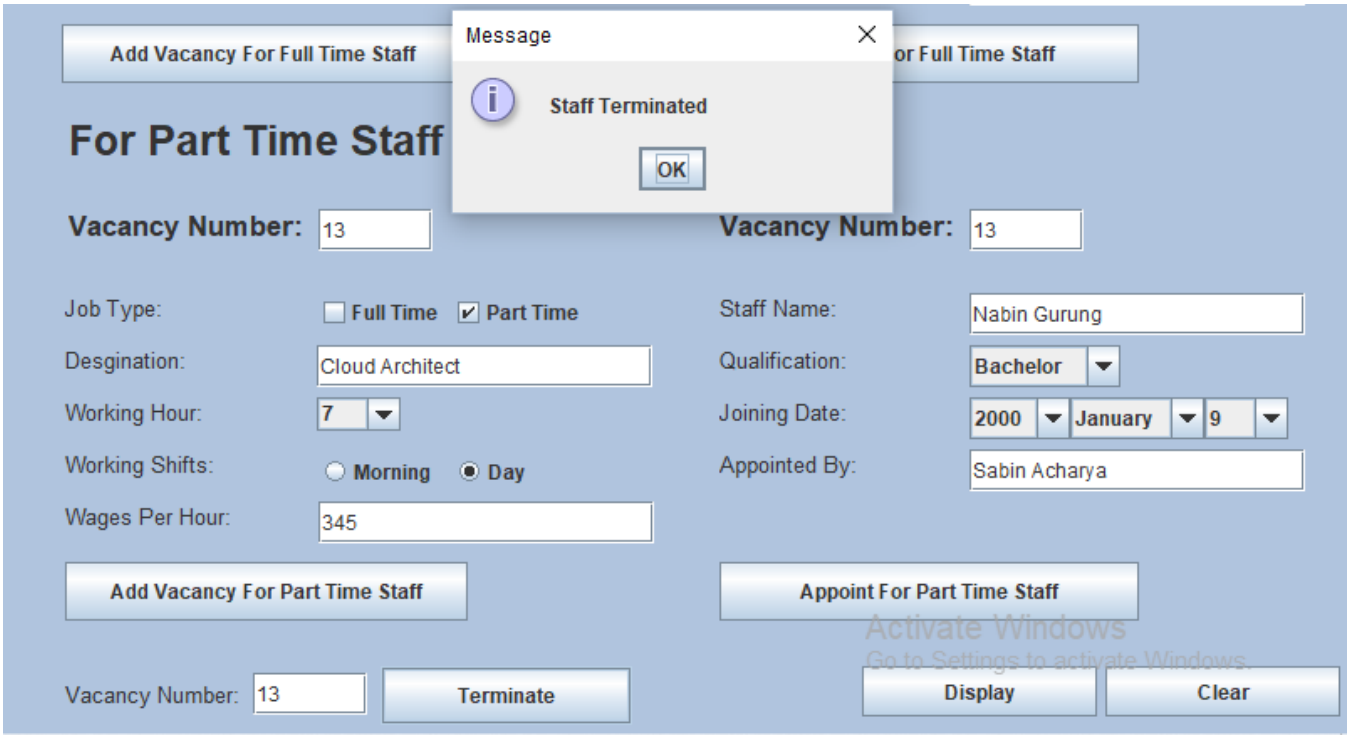

Figure 6: Screenshot of Terminating Staff

5.3.1 Test-3: Test for Dialog box message appearing when vacancy number is given which is not added.

Objective	Getting Dialog Box message that show "Invalid Vacancy Number, Please Try Again!"
Action	Different values are added to Full Time Staff by passing the values in textfields, CheckBox and ComboBox. - Vacancy Number: 12 - Staff Name: Nabin Gurung - Qualification: Bachelor - Wages Per Hour: 450 - Joining Date:1996/September/14 - Appointed By: Raj Thapa Magar
Expected Result	When the data is provided to textfield CheckBox and ComboBox, and "Appoint for Full Time Staff" button is clicked, Dialog Box message should appear, that tells "Invalid Vacancy Number, Please Try Again!"
Actual Result	When the data is provided to textfield CheckBox and ComboBox, and "Appoint for Full Time Staff" button is clicked, Dialog Box message appears, that tells "Invalid Vacancy Number, Please Try Again!"
Conclusion	Test is successful.

Table 9: Test for Dialog box message appearing when vacancy number is given which is not added.

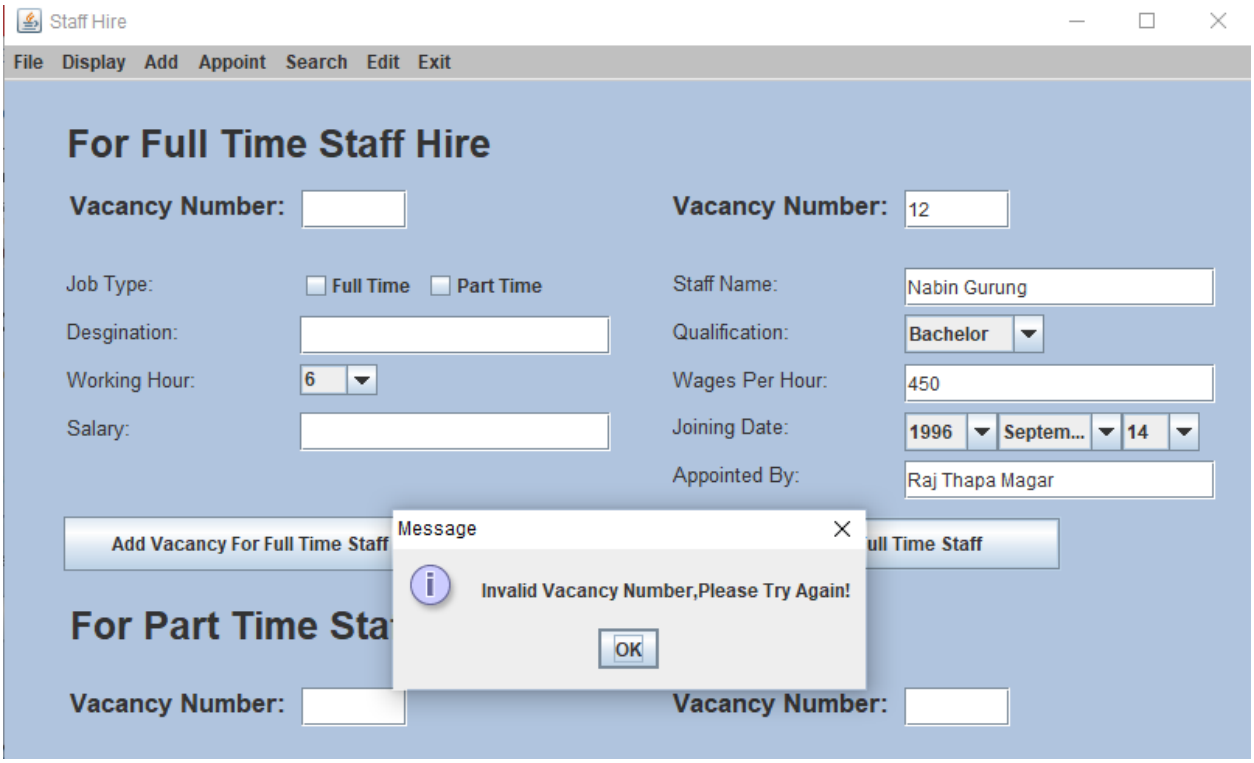

Figure 7: Screenshot of Dialog box message appearing when vacancy number is given which is not added.

5.3.2 Test-3: Test Dialog box message appearing when invalid type of input is given

Objective	Getting Dialog Box message that show "Invalid Input, Please Try Again!"
Action	Different values are added to Full Time Staff by passing the values in textfields, CheckBox and ComboBox. - Vacancy Number: ABCD - Job Type: Full Time - Designation: 123455 - Working Hours: 10 - Salary: ABCD
Expected Result	When the data is provided to textfield CheckBox and ComboBox, and "Add Vacancy for Full Time Staff" button is clicked, Dialog Box message should appear, that tells "Invalid Input, Please Try Again!"
Actual Result	When the data is provided to textfield CheckBox and ComboBox, and "Add Vacancy for Full Time Staff" button is clicked, Dialog Box message appears, that tells "Invalid Input, Please Try Again!"
Conclusion	Test is successful.

Table 10: Test of Dialog box message appearing when invalid type of input is given

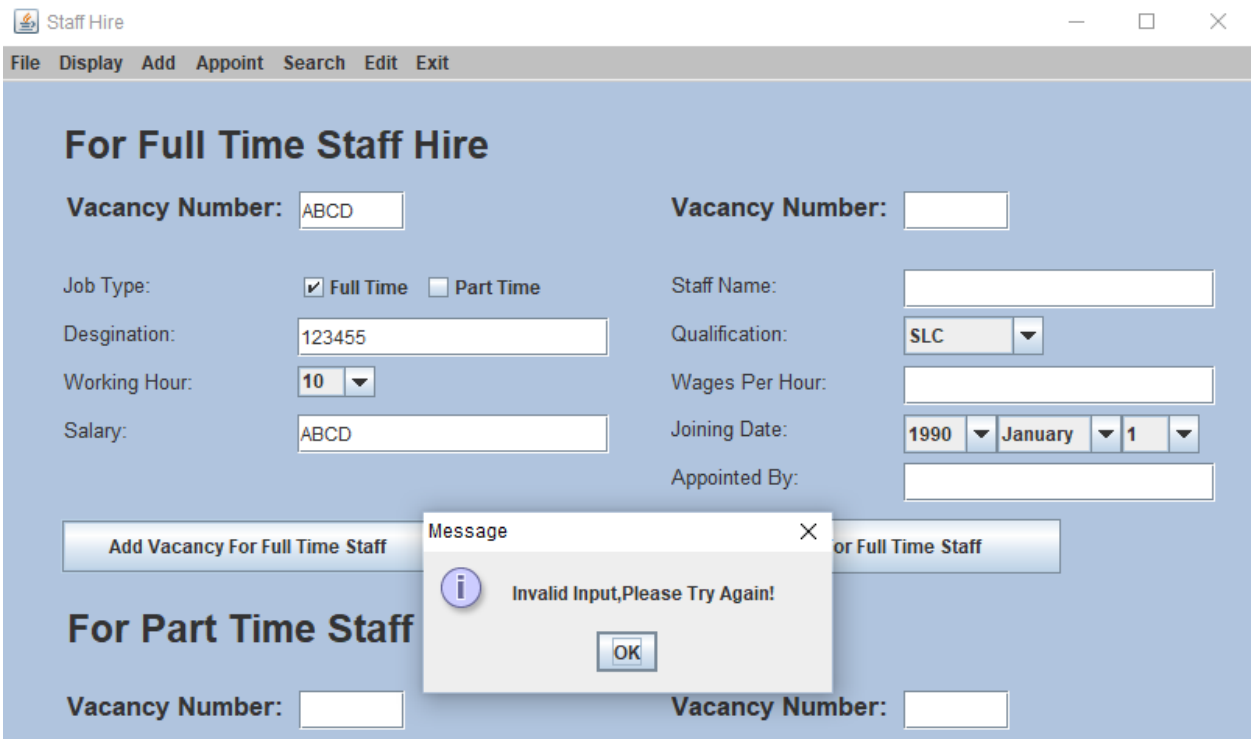

Figure 8: Screenshot of Dialog box message appearing when invalid type of input is given

5.3.3 Test-3: Test of Dialog box message appearing when vacancy number of Full Time Staff is added in Part Time Staff Hire.

Objective	Getting Dialog Box message that show “Not for Full Time Staff Hire”
Action	Different values are added to Part Time Staff by passing the values in textfields, CheckBox and ComboBox. - Vacancy Number: 11 - Job Type: Part Time - Designation: Manager - Working Hours: 10 - Working Shifts: Morning - Wages Per Hour: 345 Again, Different values are added to Full Time Staff by passing the values in textfields, CheckBox and ComboBox. - Vacancy Number: 11 - Staff Name: Nabin Gurung - Qualification: Bachelor - Wages Per Hour: 250 - Joining Date:1997/May/9 - Appointed By: Saroj Sapkota
Expected Result	When the data is provided to textfield CheckBox and ComboBox, and “Appoint for Full Time Staff” button is clicked ,Dialog Box message should appear, that show “Not for Full Time Staff Hire”
Actual Result	When the data is provided to textfield CheckBox and ComboBox, and “Appoint for Full Time Staff” button is clicked Dialog Box message appears, that show “Not for Full Time Staff Hire”
Conclusion	Test is successful.

Table 11: Test of Dialog box message appearing when vacancy number of Full Time Staff is added in Part Time Staff Hire.

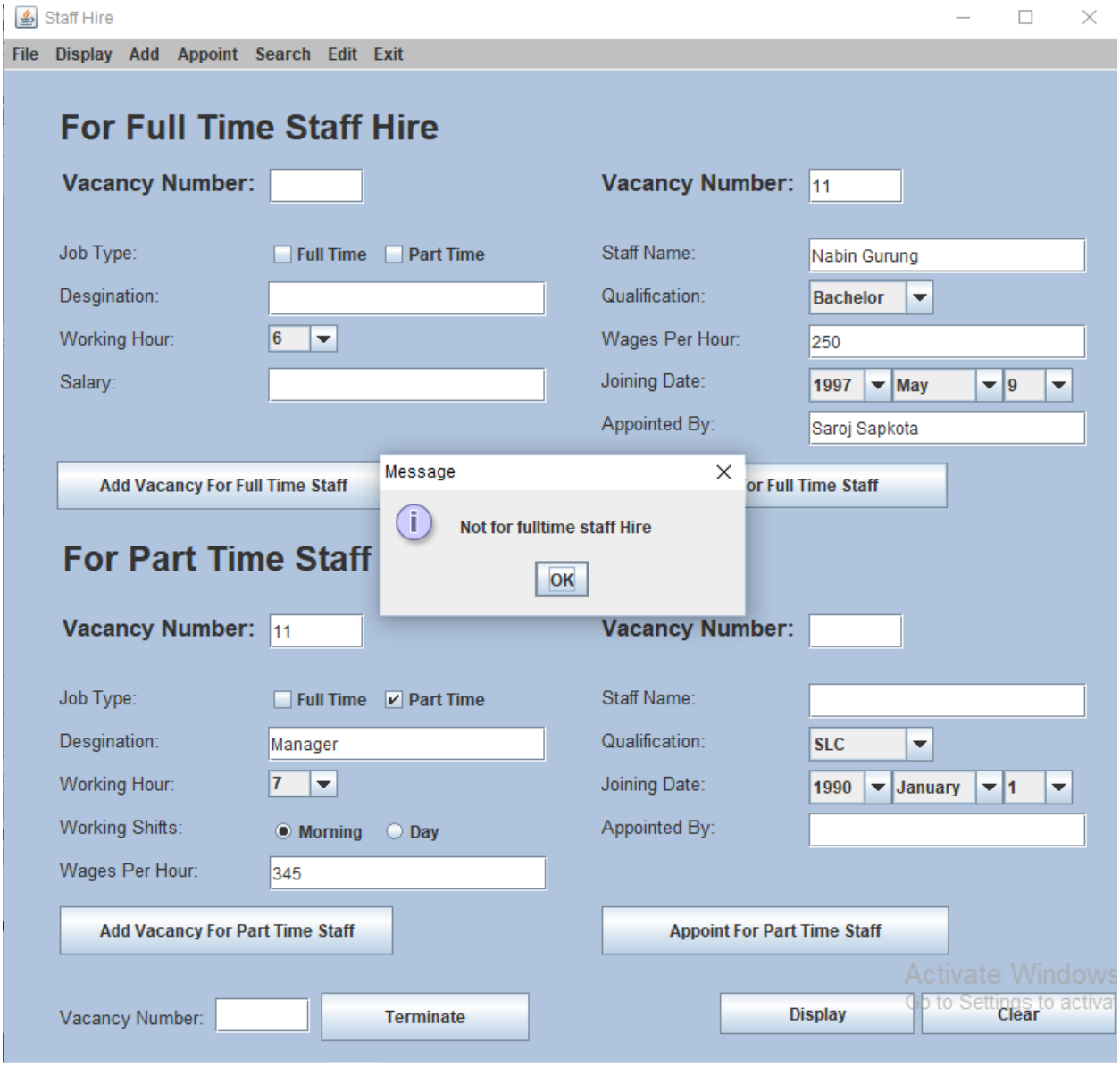

Figure 9: Screenshot of Dialog box message appearing when vacancy number of Full Time Staff is added in Part Time Staff Hire.

5.3.4 Test-3: Test for Dialog boxes appearing when vacancy number is tried to terminate, which is not added

Objective	Getting Dialog Box message that show "Invalid Vacancy Number, Please Try Again!"
Action	Vacancy Number, which is not added in Part Time Staff Class, is provided in Text Field - Vacancy Number: 11
Expected Result	When the data is provided to textfield, and "Terminate" button is clicked Dialog Box message should appear, that show "Invalid Vacancy Number, Please Try Again!"
Actual Result	When the data is provided to textfield, and "Terminate" button is clicked Dialog Box message appears, that show "Invalid Vacancy Number, Please Try Again!"
Conclusion	Test is successful.

Table 12: Test for Dialog box message appearing when vacancy number is tried to terminate, which is not added

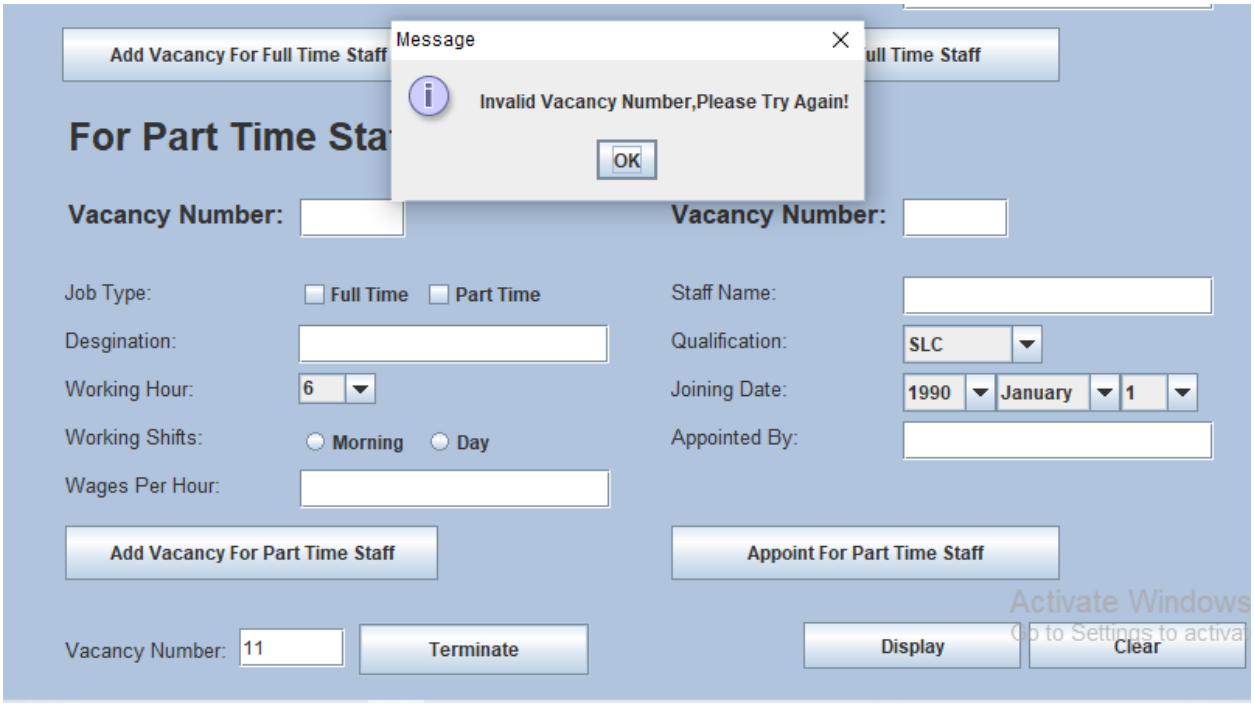

Figure 10: Screenshot of Dialog boxes appearing when vacancy number is tried to terminate, which is not added

5.3.5 Test-3: Test for Dialog box message appearing when invalid type of vacancy number is tried to terminate.

Objective	Getting Dialog Box message that show “Invalid Input, Please Try Again!”
Action	Vacancy Number, which is invalid is provided in Text Field. - Vacancy Number: 11asdasd
Expected Result	When the data is provided to textfield, and “Terminate” button is clicked Dialog Box message should appear, that show “Invalid Input, Please Try Again!””
Actual Result	When the data is provided to textfield, and “Terminate” button is clicked Dialog Box message appears, that show Invalid Input, Please Try Again!””
Conclusion	Test is successful.

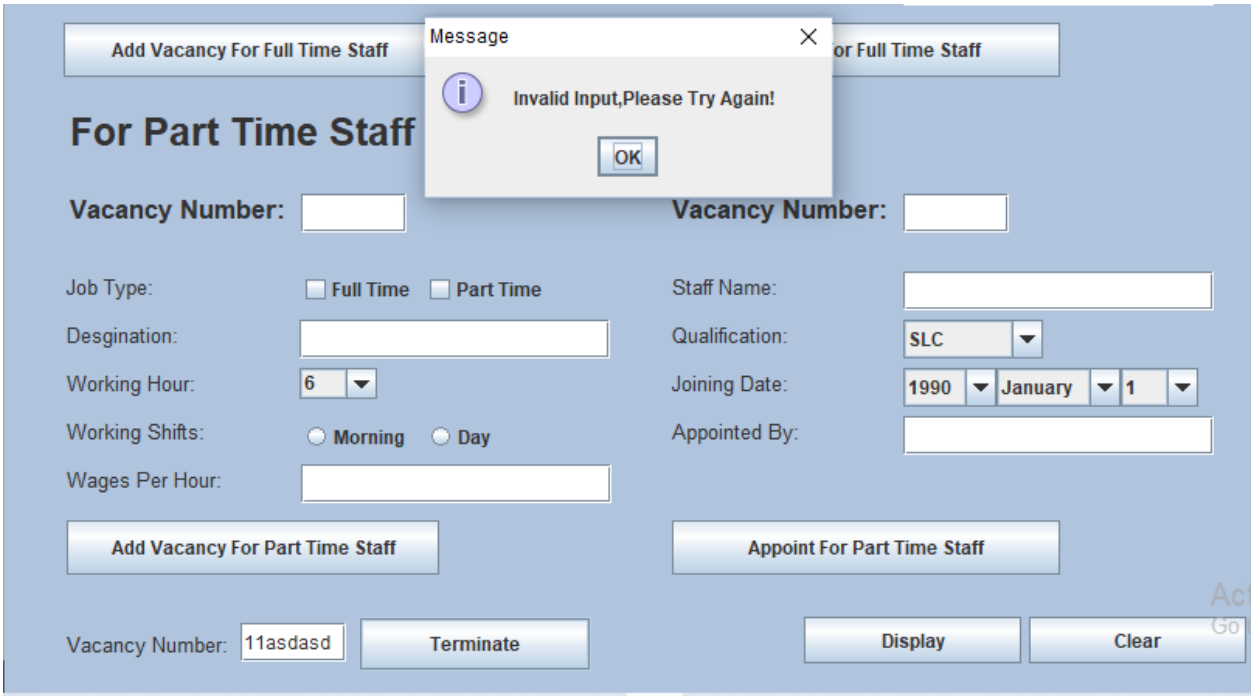

Figure 11: Screenshot of Dialog box message appearing when invalid type of vacancy number is tried to terminate.

5.3.6 Test-3: Test for Dialog box message appearing when same vacancy Number is tried to terminate two times.

Objective	Getting Dialog Box message that show “Staff is already Terminated”
Action	Vacancy Number is provided to Text Field - Vacancy Number: 13
Expected Result	When the Vacancy Number is provided to textfield, and “Terminate” button is clicked two times, Dialog Box message should appear, that show “Staff is already Terminated.””
Actual Result	When the Vacancy Number is provided to textfield, and “Terminate” button is clicked two times, Dialog Box message appear, that show “Staff is already Terminated.””
Conclusion	Test is successful.

Table 13: Test for Dialog box message appearing when same vacancy Number is tried to terminate two times.

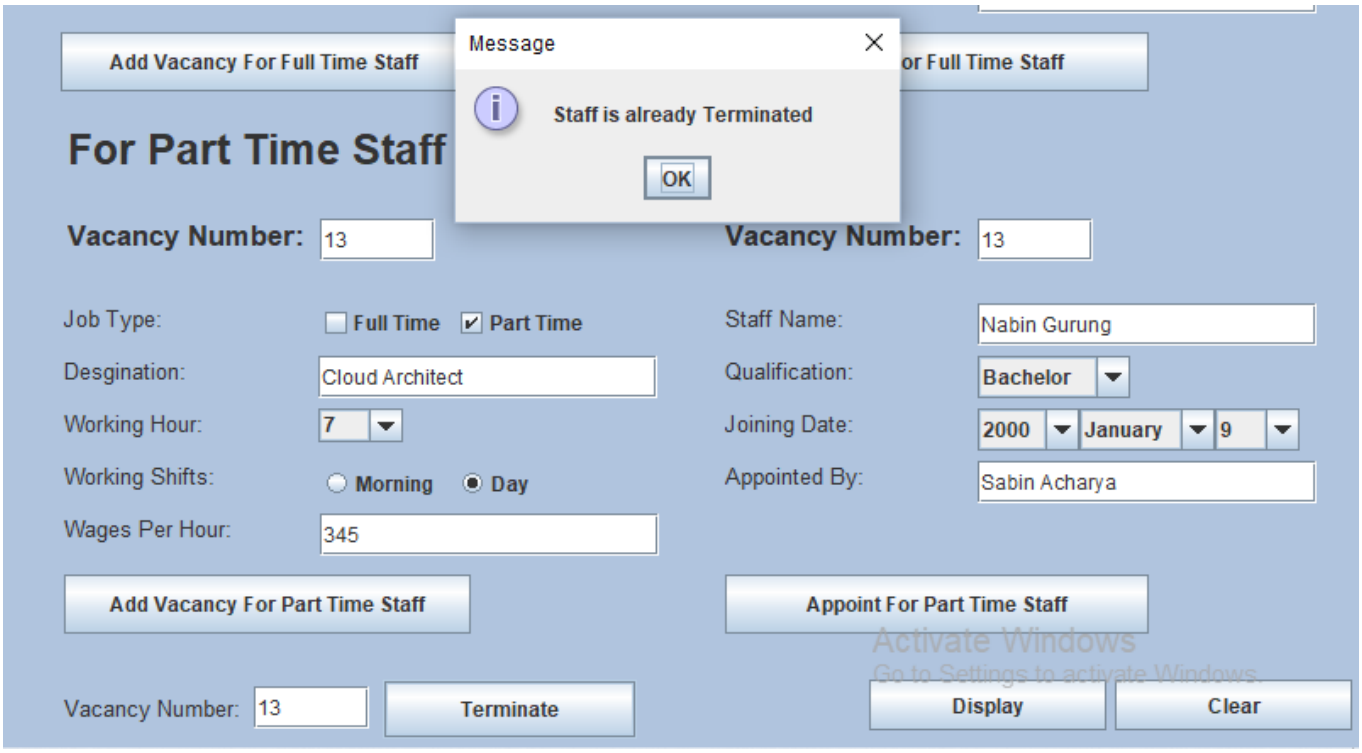

Figure 12: Test for Dialog box message appearing when same vacancy Number is tried to terminate two times.

6. Pseudo Code:

IMPORTING java swing

IMPORTING java awt

IMPORTING java arraylist

CREATE class INGNepal **IMPLEMENTS** ActionListener

DECLARE class variables

private String desigNationForFullTime,jobTypeForFullTime,staffNameForFullTime,joiningDateForFullTime,qualificationForFullTime,appointedByForFullTime,wagesPerHourForFullTime

private int vacNumForFullTime,salaryForFullTime,workingHourForFullTime

private String yearFullTime,monthFullTime,dayFullTime

private String desigNationForPartTime,jobTypeForPartTime,staffNameForPartTime,joiningDateForPartTime,qualificationForPartTime,appointedByForPartTime,radioButton_Morning,radioButton_Day

private int vacNumForPartTime,workingHourForPartTime,salaryForPartTime,wagesPerHourForPartTime

private String yearPartTime,monthPartTime,dayPartTime;

private String workingShiftsRadioButton

private String jobTypeCheckBox_FT,jobTypeCheckBox_PT

private JFrame frameStaffHire

private JPanel myPanel

private JTextField txtFieldForVacancyNum_FT, txtFieldForVacancyNum_2_FT,txtFieldForStaffNam_FT,txtFieldForDesignation_FT,txtFieldForWgsPerHr_FT,txtFieldForAppointedBy_FT,txtFieldForSalaryFT

private JTextField txtFieldForVacancyNum_PT,txtFieldForVacancyNum_2_PT,txtFieldForStaffNam_PT,txtFieldForDesignation_PT,txtFieldForWgsPerHr_PT,txtFieldForAppointedBy_PT,txtFieldForSalaryPT

private JCheckBox checkBoxFullTime_FT,checkBoxPartTime_FT,checkBoxFullTime_PT,checkBoxPartTime_PT

private JRadioButton radioButton_Morning_PT,radioButton_Day_PT

private JComboBox comboBoxWorkingHour_FT,comboBoxQualification_FT,comboBoxWorkingHour_PT,comboBoxQualification_PT,cmbYear_FT,cmbMonth_FT,cmbDay_FT,cmbYear_PT,cmbMonth_PT,cmbDay_PT

private JMenuBar menuBar

private JMenu fileMenu,exitMenu,searchMenu,aboutMenu,addMenu,appointMenu,displayMenu

private JMenuItem openMenuItem,saveMenuItem,clearMenuItem,exitMenuItem,fullMenuItem,partMenuItem,fullTimeToAppoint,partTimeToAppoint,fullTimeDisplayItem,partTimeDisplayItem

private JButton btnClear,btnDisplay,btnAppointFullTime,btnAddFullTime,btnAppointPartTime,btnAddPartTime,btnTerminate,btnTest

ArrayList <StaffHire> list = new **ArrayList** < StaffHire>

FUNCTION main (**String** [] args)

DO

    INGNepal object = **NEW INGNepal ()**

    CALLING METHOD myGUI ()

END DO

NON-PARAMETERIZED METHOD myGUI ()

    DO

frameStaffHire=new **JFrame**("Staff Hire")

CALL frameStaffHire.setVisible(true)

CALL frameStaffHire.setResizable(true)

CALL frameStaffHire.setBounds(293, 1, 820, 780)

```
JPanel myPanel=new JPanel()
CALL myPanel.setLayout(null)

Color cframe=new Color(176, 196, 222)
CALL myPanel.setBackground(cframe)

CALL frameStaffHire.add(myPanel)
CALL frameStaffHire.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)

menuBar=new JMenuBar()
CALL menuBar.setBackground(Color.lightGray)
CALL frameStaffHire.setJMenuBar(menuBar)


fileMenu=new JMenu("File")
CALL menuBar.add(fileMenu)

saveMenuItem=new JMenuItem("Save")
CALL fileMenu.add(saveMenuItem)

clearMenuItem=new JMenuItem("Clear")
CALL clearMenuItem.addActionListener(this)
CALL fileMenu.add(clearMenuItem)

displayMenu=new JMenu("Display")
menuBar.add(displayMenu)

fullTimeDisplayItem=new JMenuItem("Full Time")
fullTimeDisplayItem.addActionListener(this)
displayMenu.add(fullTimeDisplayItem)
```

```
partTimeDisplayItem=new JMenuItem("Part Time")
partTimeDisplayItem.addActionListener(this)
displayMenu.add(partTimeDisplayItem)

addMenu=new JMenu("Add")
CALL menuBar.add(addMenu)

fullMenuItem=new JMenuItem("Full Time")
CALL fullMenuItem.addActionListener(this)
CALL addMenu.add(fullMenuItem)

partMenuItem=new JMenuItem("Part Time")
CALL partMenuItem.addActionListener(this)
CALL addMenu.add(partMenuItem)

appointMenu=new JMenu("Appoint")
CALL menuBar.add(appointMenu)

fullTimeToAppoint=new JMenuItem("Full Time")
CALL fullTimeToAppoint.addActionListener(this)
CALL appointMenu.add(fullTimeToAppoint)

partTimeToAppoint=new JMenuItem("Part Time")
CALL partTimeToAppoint.addActionListener(this)
CALL appointMenu.add(partTimeToAppoint)

searchMenu=new JMenu("Search")
CALL  menuBar.add(searchMenu)
```

```
aboutMenu=new JMenu("Edit")
CALL menuBar.add(aboutMenu)

exitMenu=new JMenu("Exit")
CALL menuBar.add(exitMenu)


    JLabel lblVacancyNumber = new JLabel("Vacancy Number:")
    CALL lblVacancyNumber.setFont(new Font("Arial", Font.BOLD, 17))
    CALL lblVacancyNumber.setBounds(42, 70, 190, 20)
    CALL myPanel.add(lblVacancyNumber)

    JLabel lblVacancyNumber2 = new JLabel("Vacancy Number:")
    CALL lblVacancyNumber2.setFont(new Font("Arial", Font.BOLD, 17))
    CALL lblVacancyNumber2.setBounds(430, 70, 190, 20)
    CALL myPanel.add(lblVacancyNumber2)

    JLabel lblStaffhire = new JLabel("For Full Time Staff Hire")
    CALL lblStaffhire.setFont(new Font("Arial", Font.BOLD, 25))
    CALL lblStaffhire.setBounds(40, 35, 300, 20)
    CALL myPanel.add(lblStaffhire)

    JLabel lblStaffName = new JLabel("Staff Name:")
    CALL lblStaffName.setFont(new Font("Arial", Font.PLAIN, 13))
    CALL lblStaffName.setBounds(430, 120, 100, 20)
    CALL myPanel.add(lblStaffName)

    JLabel lblJobType = new JLabel("Job Type:")
    CALL lblJobType.setFont(new Font("Arial", Font.PLAIN, 13))
    CALL lblJobType.setBounds(40, 120, 100, 20)
```

```
CALL myPanel.add(lblJobType)

JLabel lblDesgination = new JLabel("Desgination:")
CALL lblDesgination.setFont(new Font("Arial", Font.PLAIN, 13))
CALL lblDesgination.setBounds(40, 151, 100, 20)
CALL myPanel.add(lblDesgination)

JLabel lblWorkingHour = new JLabel("Working Hour:")
CALL lblWorkingHour.setFont(new Font("Arial", Font.PLAIN, 13))
CALL lblWorkingHour.setBounds(40, 182, 100, 20)
CALL myPanel.add(lblWorkingHour)

JLabel lblWagesPerHour = new JLabel("Wages Per Hour:")
CALL lblWagesPerHour.setFont(new Font("Arial", Font.PLAIN, 13))
CALL lblWagesPerHour.setBounds(430, 182, 150, 20)
CALL myPanel.add(lblWagesPerHour)

JLabel lblQualification = new JLabel("Qualification:")
CALL lblQualification.setFont(new Font("Arial", Font.PLAIN, 13))
CALL lblQualification.setBounds(430, 151, 100, 20)
CALL myPanel.add(lblQualification)

JLabel lblJoiningDate = new JLabel("Joining Date:")
CALL lblJoiningDate.setFont(new Font("Arial", Font.PLAIN, 13))
CALL lblJoiningDate.setBounds(430, 213, 150, 18)
CALL myPanel.add(lblJoiningDate)

JLabel lblAppointedBy = new JLabel("Appointed By:")
CALL lblAppointedBy.setFont(new Font("Arial", Font.PLAIN, 13))
CALL lblAppointedBy.setBounds(430, 244, 150, 18)
```

```
CALL myPanel.add(lblAppointedBy)

JLabel lblSalary = new JLabel("Salary:")
CALL lblSalary.setFont(new Font("Arial", Font.PLAIN, 13))
CALL lblSalary.setBounds(40, 213, 100, 20)
CALL myPanel.add(lblSalary)

JLabel lblTitleParTime = new JLabel("For Part Time Staff Hire")
CALL lblTitleParTime.setFont(new Font("Arial", Font.BOLD, 25))
CALL lblTitleParTime.setBounds(42, 345, 300, 20)
CALL myPanel.add(lblTitleParTime)

JLabel lblVacancyNumberPT = new JLabel("Vacancy Number:")
CALL lblVacancyNumberPT.setFont(new Font("Arial", Font.BOLD, 17))
CALL lblVacancyNumberPT.setBounds(42, 390, 190, 20)
CALL  myPanel.add(lblVacancyNumberPT)

JLabel lblVacancyNumber2PT = new JLabel("Vacancy Number:")
CALL lblVacancyNumber2PT.setFont(new Font("Arial", Font.BOLD, 17))
CALL lblVacancyNumber2PT.setBounds(430, 390, 190, 20)
CALL myPanel.add(lblVacancyNumber2PT)

JLabel lblStaffNamePT = new JLabel("Staff Name:")
CALL lblStaffNamePT.setFont(new Font("Arial", Font.PLAIN, 13))
CALL lblStaffNamePT.setBounds(430, 440, 100, 20)
CALL myPanel.add(lblStaffNamePT)

JLabel lblJobTypePT = new JLabel("Job Type:")
CALL lblJobTypePT.setFont(new Font("Arial", Font.PLAIN, 13))
CALL lblJobTypePT.setBounds(40,440,100,20)
```

```
CALL myPanel.add(lblJobTypePT)

JLabel lblDesginationPT = new JLabel("Desgination:")
CALL lblDesginationPT.setFont(new Font("Arial", Font.PLAIN, 13))
CALL lblDesginationPT.setBounds(40, 471, 100, 20)
CALL myPanel.add(lblDesginationPT)

JLabel lblWorkingHourPT = new JLabel("Working Hour:")
CALL lblWorkingHourPT.setFont(new Font("Arial", Font.PLAIN, 13))
CALL lblWorkingHourPT.setBounds(40, 502, 100, 20)
CALL myPanel.add(lblWorkingHourPT)

JLabel lblWorkingShiftsPT = new JLabel("Working Shifts:")
CALL lblWorkingShiftsPT.setFont(new Font("Arial", Font.PLAIN, 13))
CALL lblWorkingShiftsPT.setBounds(40, 533, 100, 20)
CALL myPanel.add(lblWorkingShiftsPT)

JLabel lblWagesPerHourPT = new JLabel("Wages Per Hour:")
CALL lblWagesPerHourPT.setFont(new Font("Arial", Font.PLAIN, 13))
CALL lblWagesPerHourPT.setBounds(40, 564, 150, 20)
CALL myPanel.add(lblWagesPerHourPT)

JLabel lblQualificationPT = new JLabel("Qualification:")
CALL lblQualificationPT.setFont(new Font("Arial", Font.PLAIN, 13))
CALL lblQualificationPT.setBounds(430, 471, 100, 20)
CALL myPanel.add(lblQualificationPT)

JLabel lblJoiningDatePT = new JLabel("Joining Date:")
CALL  lblJoiningDatePT.setFont(new Font("Arial", Font.PLAIN, 13))
CALL lblJoiningDatePT.setBounds(430, 502, 150, 20)
```

```
CALL myPanel.add(lblJoiningDatePT)

JLabel lblAppointedByPT = new JLabel("Appointed By:")
CALL lblAppointedByPT.setFont(new Font("Arial", Font.PLAIN, 13))
CALL lblAppointedByPT.setBounds(430, 533, 150, 20)
CALL myPanel.add(lblAppointedByPT)

JLabel lblVacancyNumberTerminate = new JLabel("Vacancy Number:")
CALL lblVacancyNumberTerminate.setFont(new Font("Arial", Font.PLAIN, 13))
CALL lblVacancyNumberTerminate.setBounds(40, 670, 150, 20)
CALL myPanel.add(lblVacancyNumberTerminate)

txtFieldForVacancyNum_FT = new JTextField()
CALL txtFieldForVacancyNum_FT.setBounds(191, 70, 68, 25)
CALL myPanel.add(txtFieldForVacancyNum_FT)

txtFieldForVacancyNum_2_FT = new JTextField()
CALL txtFieldForVacancyNum_2_FT.setBounds(579, 70, 68, 25)
CALL myPanel.add(txtFieldForVacancyNum_2_FT)

txtFieldForStaffNam_FT = new JTextField()
CALL txtFieldForStaffNam_FT.setBounds(579, 120, 200, 25)
CALL myPanel.add(txtFieldForStaffNam_FT)

txtFieldForDesignation_FT = new JTextField()
CALL txtFieldForDesignation_FT.setBounds(190, 151, 200, 25)
CALL myPanel.add(txtFieldForDesignation_FT)

txtFieldForWgsPerHr_FT = new JTextField()
CALL txtFieldForWgsPerHr_FT.setBounds(579, 182, 200, 25)
```

```
CALL myPanel.add(txtFieldForWgsPerHr_FT)

txtFieldForAppointedBy_FT = new JTextField()
CALL txtFieldForAppointedBy_FT.setBounds(579, 244, 200, 25)
CALL myPanel.add(txtFieldForAppointedBy_FT)

txtFieldForSalaryFT = new JTextField()
CALL txtFieldForSalaryFT.setBounds(190, 213, 200, 25)
CALL myPanel.add(txtFieldForSalaryFT)

txtFieldForVacancyNum_PT = new JTextField()
CALL txtFieldForVacancyNum_PT.setBounds(191, 390, 68, 25)
CALL myPanel.add(txtFieldForVacancyNum_PT)

txtFieldForVacancyNum_2_PT = new JTextField()
CALL txtFieldForVacancyNum_2_PT.setBounds(579, 390, 68, 25)
CALL myPanel.add(txtFieldForVacancyNum_2_PT)

txtFieldForStaffNam_PT = new JTextField()
CALL txtFieldForStaffNam_PT.setBounds(579, 440, 200, 25)
CALL myPanel.add(txtFieldForStaffNam_PT)

txtFieldForDesignation_PT = new JTextField()
CALL txtFieldForDesignation_PT.setBounds(190, 471, 200, 25)
CALL myPanel.add(txtFieldForDesignation_PT)

txtFieldForWgsPerHr_PT = new JTextField()
CALL txtFieldForWgsPerHr_PT.setBounds(191, 564, 200, 25)
CALL myPanel.add(txtFieldForWgsPerHr_PT)
```

```
txtFieldForAppointedBy_PT = new JTextField()
CALL txtFieldForAppointedBy_PT.setBounds(579, 533, 200, 25)
CALL myPanel.add(txtFieldForAppointedBy_PT)

txtFieldToTerminate = new JTextField()
CALL txtFieldToTerminate.setBounds(152, 666, 68, 25)
CALL myPanel.add(txtFieldToTerminate)

checkBoxFullTime_FT = new JCheckBox("Full Time")
Color cfull1=new Color(176, 196, 222)
CALL checkBoxFullTime_FT.setBackground(cfull1)
CALL checkBoxFullTime_FT.setBounds(190, 120, 80, 23)
CALL myPanel.add(checkBoxFullTime_FT)

checkBoxPartTime_FT = new JCheckBox("Part Time")
Color cpart1=new Color(176, 196, 222)
CALL checkBoxPartTime_FT.setBackground(cpart1)
CALL checkBoxPartTime_FT.setBounds(270, 120, 80, 23)
CALL myPanel.add(checkBoxPartTime_FT)



checkBoxFullTime_PT = new JCheckBox("Full Time")
Color cfull2=new Color(176, 196, 222)
CALL checkBoxFullTime_PT.setBackground(cfull2)
CALL checkBoxFullTime_PT.setBounds(190, 440, 80, 23)
CALL myPanel.add(checkBoxFullTime_PT)

checkBoxPartTime_PT = new JCheckBox("Part Time")
Color cpart2=new Color(176, 196, 222)
```

```
CALL checkBoxPartTime_PT.setBackground(cpart2)
CALL checkBoxPartTime_PT.setBounds(270, 440, 80, 23)
CALL myPanel.add(checkBoxPartTime_PT)

radioButton_Morning_PT = new JRadioButton("Morning")
Color m2=new Color(176, 196, 222)
CALL radioButton_Morning_PT.setBackground(m2)
CALL radioButton_Morning_PT.setBounds(191, 533, 80, 27)
CALL myPanel.add(radioButton_Morning_PT)

radioButton_Day_PT = new JRadioButton("Day")
Color d2=new Color(176, 196, 222)
CALL radioButton_Day_PT.setBackground(d2)
CALL radioButton_Day_PT.setBounds(271, 533, 68, 27)
CALL myPanel.add(radioButton_Day_PT)

String workinghour[]= {"6","7","8","9","10","11","12","13","14"};
comboBoxWorkingHour_FT = new JComboBox<Object>(workinghour)
CALL comboBoxWorkingHour_FT.setBounds(190, 182, 50, 20)
CALL myPanel.add(comboBoxWorkingHour_FT)

String qualification[]= {"SLC","+2","Bachelor","Master"};
comboBoxQualification_FT = new JComboBox<Object>(qualification)
CALL comboBoxQualification_FT.setBounds(579, 150, 90, 25)
CALL myPanel.add(comboBoxQualification_FT)


String workinghourPT[]= {"6","7","8","9","10","11","12","13","14"};
comboBoxWorkingHour_PT = new JComboBox<Object>(workinghourPT)
CALL comboBoxWorkingHour_PT.setBounds(190, 502, 50, 20)
```

```
CALL myPanel.add(comboBoxWorkingHour_PT)


String qualificationPT[]= {"SLC","+2","Bachelor","Master"};
comboBoxQualification_PT = new JComboBox<Object>(qualificationPT)
CALL comboBoxQualification_PT.setBounds(579, 471, 90, 25)
CALL myPanel.add(comboBoxQualification_PT)


String year[]={"1990", "1991", "1992", "1993", "1994", "1995", "1996", "1997", "1998",
"1999", "2000", "2001", "2002", "2003", "2004", "2005", "2006", "2007", "2008", "2009",
"2010", "2011", "2012",
"2013","2014","2015","2016","2017","2018","2019","2020","2021"};
cmbYear_FT=new JComboBox<Object>(year)
CALL cmbYear_FT.setBounds(579, 213, 60, 25)
CALL myPanel.add(cmbYear_FT)


String
month[]={"January","February","March","April","May","June","July","August","Septembe
r","October","November","December"};
cmbMonth_FT=new JComboBox<Object>(month)
CALL cmbMonth_FT.setBounds(639, 213, 80, 25)
CALL myPanel.add(cmbMonth_FT)


String
day[]={"1","2","3","4","5","6","7","8","9","10","11","12","13","14","15","16","17","18","19","
20","21","22","23","24","25","26","27","28","29","30","31"};
cmbDay_FT=new JComboBox<Object>(day)
CALL cmbDay_FT.setBounds(719, 213, 50, 25)
CALL myPanel.add(cmbDay_FT)
```

```
String yearPT[]={"1990", "1991", "1992", "1993", "1994", "1995", "1996", "1997",
"1998", "1999", "2000", "2001", "2002", "2003", "2004", "2005", "2006", "2007", "2008",
"2009", "2010", "2011", "2012",
"2013","2014","2015","2016","2017","2018","2019","2020","2021"};

    cmbYear_PT=new JComboBox<Object>(yearPT)

    CALL cmbYear_PT.setBounds(579, 502, 60, 25)

    CALL myPanel.add(cmbYear_PT)


String
monthPT[]={"January","February","March","April","May","June","July","August","Septe
mber","October","November","December"};

    cmbMonth_PT=new JComboBox<Object>(monthPT)

    cmbMonth_PT.setBounds(639, 502, 80, 25)

    myPanel.add(cmbMonth_PT)


String
dayPT[]={"1","2","3","4","5","6","7","8","9","10","11","12","13","14","15","16","17","18","1
9","20","21","22","23","24","25","26","27","28","29","30","31"};

    cmbDay_PT=new JComboBox<Object>(dayPT)

    CALL cmbDay_PT.setBounds(719, 502, 50, 25)

    CALL myPanel.add(cmbDay_PT)


    btnClear = new JButton("Clear")

    CALL btnClear.addActionListener(this)

    CALL btnClear.setBounds(660, 662, 140, 30)

    CALL myPanel.add(btnClear)


    btnDisplay = new JButton("Display")

    CALL btnDisplay.setBounds(515, 662, 140, 30)

    CALL btnDisplay.addActionListener(this)

    CALL myPanel.add(btnDisplay)
```

```
    btnAddFullTime = new JButton("Add Vacancy For Full Time Staff")
    CALL btnAddFullTime.setBounds(38, 280,240, 35)
    CALL btnAddFullTime.addActionListener(this)
    CALL myPanel.add(btnAddFullTime)

    btnAddPartTime = new JButton("Add Vacancy For Part Time Staff")
    CALL btnAddPartTime.setBounds(40, 600, 240, 35)
    CALL btnAddPartTime.addActionListener(this)
    CALL myPanel.add(btnAddPartTime)

    btnAppointFullTime = new JButton("Appoint For Full Time Staff")
    CALL btnAppointFullTime.setBounds(430,280, 250, 35)
    CALL btnAppointFullTime.addActionListener(this)
    CALL myPanel.add(btnAppointFullTime)

    btnAppointPartTime = new JButton("Appoint For Part Time Staff")
    CALL btnAppointPartTime.setBounds(430, 600, 250, 35)
    CALL btnAppointPartTime.addActionListener(this)
    CALL myPanel.add(btnAppointPartTime)


    btnTerminate = new JButton("Terminate")
    CALL btnTerminate.addActionListener(this)
    CALL btnTerminate.setBounds(228, 662, 150, 35)
    CALL myPanel.add(btnTerminate)

END DO
```

```
OVERRIDE actionPerformed
DO
IF radioButton_Morning_PT.is Selected
        DO
                radioButton_Day_PT set to false
        END DO
IF radioButton_Day_PT.is Selected
        DO
                radioButton_Morning_PT set to false
        END DO

INITIALIZE workingShiftsRadioButton =  ""
IF radioButton_Morning_PT is Selected
        DO
                radioButton_Morning_PT  = “Morning”
        END DO
IF radioButton_Day_PT.is Selected
        DO
                radioButton_Day_PT= “Day”
        END DO

INITIALIZE  jobTypeCheckBox_FT = ""
        If checkBoxFullTime_FT is Selected
        DO
                checkBoxPartTime_FT set to false
        END DO

        If checkBoxPartTime_FT is Selected
        DO
                checkBoxFullTime_FT set to false
```

```
        END DO


        If checkBoxFullTime_FT is Selected
        DO
                jobTypeCheckBox_FT="Full Time";
        END DO

        If checkBoxPartTime_FT is Selected
        DO
                jobTypeCheckBox_FT="Part Time"
        END DO

INITIALIZE   jobTypeCheckBox_PT=""
        If checkBoxFullTime_PT is Selected
        DO
                checkBoxPartTime_PT set to false
        END DO

        If checkBoxPartTime_PT is Selected
        DO
                checkBoxFullTime_PT set to false
        END DO


        If checkBoxFullTime_PT is Selected
        DO
                jobTypeCheckBox_PT="Full Time";
        END DO
```

```
If checkBoxPartTime_PT is Selected
DO
        jobTypeCheckBox_PT="Part Time"
END DO

IF e.getSource() is equals to btnClear
        DO
                txtFieldForVacancyNum_FT set Text = ""
                txtFieldForStaffNam_FT set Text = ""
                txtFieldForDesignation_FT set Text = ""
                 txtFieldForStaffNam_FT set Text = ""
                 txtFieldForWgsPerHr_FT set Text = ""
                txtFieldForAppointedBy_FT set Text = ""
                 txtFieldForSalaryFT set Text = ""
                 txtFieldForVacancyNum_2_FT set Text = ""

                 radioButton_Morning_PT set Selected to "false"
                 radioButton_Day_PT set Selected to "false"

                 cmbYear_FT setSelectedIndex = 0
                 cmbMonth_FT setSelectedIndex = 0
                 cmbDay_FT setSelectedIndex = 0
                 comboBoxWorkingHour_FT setSelectedIndex = 0
                 comboBoxQualification_FT setSelectedIndex = 0

                checkBoxFullTime_FT setSelectedIndex = 0
                 checkBoxPartTime_FT setSelectedIndex = 0
                checkBoxFullTime_PT setSelectedIndex = 0
                checkBoxPartTime_PT setSelectedIndex = 0
```

```
txtFieldForVacancyNum_PT set Text =""
txtFieldForStaffNam_PT set Text =""
 txtFieldForDesignation_PT set Text =""
txtFieldForStaffNam_PT set Text =""
txtFieldForWgsPerHr_PT set Text =""
txtFieldForSalaryPT.setText("")
txtFieldForAppointedBy_PT.setText("")
 txtFieldForVacancyNum_2_PT.setText("")

cmbYear_PT setSelectedIndex = 0
cmbMonth_PT setSelectedIndex = 0
 cmbDay_PT setSelectedIndex = 0
comboBoxQualification_PT setSelectedIndex = 0
comboBoxWorkingHour_PT setSelectedIndex = 0
```

IF e.getSource() is equals to clearMenuItem

DO

```
txtFieldForVacancyNum_FT set Text = ""
txtFieldForStaffNam_FT set Text = ""
txtFieldForDesignation_FT set Text = ""
 txtFieldForStaffNam_FT set Text = ""
 txtFieldForWgsPerHr_FT set Text = ""
txtFieldForAppointedBy_FT set Text = ""
 txtFieldForSalaryFT set Text = ""
 txtFieldForVacancyNum_2_FT set Text = ""

 radioButton_Morning_PT set Selected to “false”
 radioButton_Day_PT set Selected to “false”

 cmbYear_FT setSelectedIndex = 0
```

```
cmbMonth_FT setSelectedIndex = 0
cmbDay_FT setSelectedIndex = 0
comboBoxWorkingHour_FT setSelectedIndex = 0
comboBoxQualification_FT setSelectedIndex = 0

checkBoxFullTime_FT setSelectedIndex = 0
checkBoxPartTime_FT setSelectedIndex = 0
checkBoxFullTime_PT setSelectedIndex = 0
checkBoxPartTime_PT setSelectedIndex = 0

txtFieldForVacancyNum_PT set Text =""
txtFieldForStaffNam_PT set Text =""
txtFieldForDesignation_PT set Text =""
txtFieldForStaffNam_PT set Text =""
txtFieldForWgsPerHr_PT set Text =""
txtFieldForSalaryPT.setText("")
txtFieldForAppointedBy_PT.setText("")
txtFieldForVacancyNum_2_PT.setText("")

cmbYear_PT setSelectedIndex = 0
cmbMonth_PT setSelectedIndex = 0
cmbDay_PT setSelectedIndex = 0
comboBoxQualification_PT setSelectedIndex = 0
comboBoxWorkingHour_PT setSelectedIndex = 0
```

IF e.getSource() is equals to **fullMenuItem**

DO

TRY

VacNumForFullTime = getText (txtFieldForVacancyNum_FT) **CONVERT TO INTEGER**

DesigNationForFullTime = getText (txtFieldForDesignation_FT)

DECLARE String wh = getSelectedItem (comboBoxWorkingHour_FT) **CONVERT TO STRING**

workingHourForFullTime= wh **CONVERT TO INTEGER**

salaryForFullTime= getText (txtFieldForSalaryFT) **CONVERT TO INTEGER**

DECLARE AND INTIALIZE boolean duplicateVacancyNum is equals to “false”;

FOR (StaffHire var:list)

IF(var.getVacancyNumber() is equals to vacNumForFullTime)

DO

duplicateVacancyNum=true;

BREAK

END DO

IF(duplicateVacancyNum is equals to false)

DO

FullTimeStaffHire objectFullTime=**new** FullTimeStaffHire(vacNumForFullTime, desigNationForFullTime,jobTypeCheckBox_FT,salaryForFullTime,workingHour ForFullTime)

CALL list.add(objectFullTime)

JOptionPane.showMessageDialog(frameStaffHire,"*Vacancy has been added in Full Time.Thank You!*")

END DO

ELSE

JOptionPane.showMessageDialog(frameStaffHire,"*Vacancy Number you have entered is already added.Please input **new** vacancy number.* ")

CATCH(NumberFormatException exp){

JOptionPane.showMessageDialog(frameStaffHire,"*Invalid Input,Please Try Again!*")

END DO

IF e.getSource() is equals to **fullTimeToAppoint**

DO

TRY

VacNumForFullTime = getText (txtFieldForVacancyNum_2_FT) **CONVERT TO INTEGER**

staffNameForFullTime=getText (txtFieldForStaffNam_FT)

yearFullTime =getSelectedItem(cmbYear_FT)**CONVERT TO STRING**

monthFullTime =getSelectedItem(cmbMonth_FT)**CONVERT TO STRING**

dayFullTime = getSelectedItem(cmbDay_FT)**CONVERT TO STRING**

joiningDateForFullTime = yearFullTime+"**CONCATINATE**"+monthFullTime+"**CONCATINATE**"+dayFullTime

String wh=getSelectedItem(comboBoxWorkingHour_FT) **CONVERT TO STRING**

workingHourForFullTime=wh(**CONVERT TO INTEGER**)

qualificationForFullTime=getSelectedItem(comboBoxQualification_FT)**CONVERT TO STRING**

wagesPerHourForFullTime=getText(txtFieldForWgsPerHr_FT)

appointedByForFullTime=getText(txtFieldForAppointedBy_FT)

DECLARE AND INTIALIZE boolean foundVacancyNum is equals to false

FOR(StaffHire staffHire:list)

```
                IF(staffHire.getVacancyNumber() is equals to
vacNumForFullTime)
                DO
                foundVacancyNum is equals to true
                END DO
                IF(staffHire instanceof FullTimeStaffHire)
                    DO
                  FullTimeStaffHire h=(FullTimeStaffHire)staffHire
                  IF(h.isJoined() is equals to true)
                    DO
                    JOptionPane.showMessageDialog(frameStaffHire,"Staff has
been hired already!!")
                  ELSE

h.hireFullTimeStaff(staffNameForFullTime,joiningDateForFullTime,qualification
ForFullTime,appointedByForFullTime)
                    JOptionPane.showMessageDialog(frameStaffHire,"Staff has
been hired in Full Time.Thank You!!")
                  BREAK
                    ELSE
                  JOptionPane.showMessageDialog(frameStaffHire,"Not for
fulltime staff Hire")
                  BREAK
        END DO

          IF(!foundVacancyNum)
                DO
            JOptionPane.showMessageDialog(frameStaffHire,"Invalid Vacancy
Number,Please Try Again!")
                END DO

        CATCH(EXCEPTION e3)
```

DO

JOptionPane.showMessageDialog(frameStaffHire,"*Invalid Input,Please Try Again!*")

END DO

IF e.getSource() is equals to **partTimeToAppoint**

DO

TRY

VacNumForPartTime = getText (txtFieldForVacancyNum_2_PT) **CONVERT TO INTEGER**

staffNameForPartTime=getText (txtFieldForStaffNam_PT)

yearPartTime =getSelectedItem(cmbYear_PT)**CONVERT TO STRING**

monthPartTime =getSelectedItem(cmbMonth_PT)**CONVERT TO STRING**

dayPartTime = getSelectedItem(cmbDay_PT)**CONVERT TO STRING**

joiningDateForPartTime = yearPartTime+"**CONCATINATE**"+monthPartTime+"**CONCATINATE**"+dayPart Time

workingHourForPartTime=wh(**CONVERT TO INTEGER**)

qualificationForPartTime=getSelectedItem(comboBoxQualification_PT)**CONVE RT TO STRING**

radioButton_Morning=getText(radioButton_Morning_PT) **CONVERT TO STRING**

radioButton_Day=getText(radioButton_Day_PT) **CONVERT TO STRING**

String WagesPrHr= getText (txtFieldForWgsPerHr_PT) **CONVERT TO STRING**

wagesPerHourForPartTime=WagesPrHr(**CONVERT TO INTEGER**)

appointedByForPartTime=getText(txtFieldForAppointedBy_PT)

DECLARE AND INITIALIZE boolean foundVacancyNum is equals to false

FOR(StaffHire staffHire:list)

IF(staffHire.getVacancyNumber() is equals to vacNumForPartTime

```
DO
        foundVacancyNum is equals to true
END DO

        IF(staffHire instanceof PartTimeStaffHire)
          DO
        PartTimeStaffHire h2=(PartTimeStaffHire)staffHire
        IF(h2.isHasJoined() is equals to true)
                DO
JOptionPane.showMessageDialog(frameStaffHire,"Staff has been hired
already!!")
                END DO
        ELSE
                DO

h2.hirePartTimeStaff(staffNameForPartTime,joiningDateForPartTime,qualificati
onForPartTime,appointedByForPartTime)
JOptionPane.showMessageDialog(frameStaffHire,"Staff has been hired in Part
Time.Thank You!!")
         BREAK
                END DO

        ELSE
                DO
JOptionPane.showMessageDialog(frameStaffHire,"Not for Part time staff Hire")
        BREAK
                        END DO

    IF(!foundVacancyNum)
      DO
JOptionPane.showMessageDialog(frameStaffHire,"Invalid Vacancy
Number,Please Try Again!")
```

```
            END DO


            CATCH(Exception e3)
JOptionPane.showMessageDialog(frameStaffHire,"Invalid Input,Please Try
Again!")


IF (e.getSource() is equals to fullTimeDisplayItem)
            DO
      FOR(StaffHire staffHire:list)
                  IF(staffHire instanceof FullTimeStaffHire)
                        FullTimeStaffHire o1=(FullTimeStaffHire)staffHire
                        CALL o1.displayStaffHire()

      System.exit(0)
            END DO


IF (e.getSource() is equals to partTimeDisplayItem)
            DO
      FOR(StaffHire staffHire:list)
                  IF(staffHire instanceof PartTimeStaffHire)
                        PartTimeStaffHire o1=(PartTimeStaffHire)staffHire
                        CALL o1.displayStaffHire()

      System.exit(0)
            END DO
```

IF e.getSource() is equals to **partMenuItem**

DO

TRY

VacNumForPartTime = getText (txtFieldForVacancyNum_PT) **CONVERT TO INTEGER**

DesigNationForPartTime = getText (txtFieldForDesignation_PT)

String wh_P=getSelectedItem(comboBoxWorkingHour_PT) **CONVERT TO STRING**

workingHourForPartTime=wh_P(**CONVERT TO INTEGER**)

String WagesPrHr=getText(txtFieldForWgsPerHr_PT)

wagesPerHourForPartTime=WagesPrHr(**CONVERT TO INTEGER**)

DECLARE AND INTIALIZE boolean duplicateVacancyNumParTime is equals to false

FOR (StaffHire staffHire:list)

IF(var.getVacancyNumber() is equals to vacNumForPartTime)

DO

duplicateVacancyNumParTime=true

BREAK

END DO

IF(duplicateVacancyNumParTime is equals to false)

DO

PartTimeStaffHire obj=new PartTimeStaffHire(vacNumForPartTime, desigNationForPartTime,jobTypeCheckBox_PT,workingHourForPartTime,wagesPerHourForPartTime,workingShiftsRadioButton)

list.add(obj)

JOptionPane.showMessageDialog(frameStaffHire,"*Vacancy has been added in Part Time.Thank You!"*)

END DO

ELSE

DO

JOptionPane.showMessageDialog(frameStaffHire,"*Vacancy Number you have entered is already added.Please input new vacancy number.* ")

END DO

CATCH(**NumberFormatException** expe)

JOptionPane.showMessageDialog(frameStaffHire,"*Invalid Input!Please try again*")

IF e.getSource() is equals to btnAddFullTime

DO

TRY

VacNumForFullTime = getText (txtFieldForVacancyNum_FT) **CONVERT TO INTEGER**

DesigNationForFullTime = getText (txtFieldForDesignation_FT)

DECLARE String wh = getSelectedItem (comboBoxWorkingHour_FT) **CONVERT TO STRING**

workingHourForFullTime= wh **CONVERT TO INTEGER**

salaryForFullTime= getText (txtFieldForSalaryFT) **CONVERT TO INTEGER**

DECLARE AND INTIALIZE boolean duplicateVacancyNum is equals to “false”

FOR (StaffHire var:list)

IF(var.getVacancyNumber() is equals to vacNumForFullTime)

DO

duplicateVacancyNum=true

BREAK

END DO

IF(duplicateVacancyNum is equals to false)

DO

FullTimeStaffHire objectFullTime=**new** FullTimeStaffHire(vacNumForFullTime, desigNationForFullTime,jobTypeCheckBox_FT,salaryForFullTime,workingHour ForFullTime)

CALL list.add(objectFullTime)

JOptionPane.showMessageDialog(frameStaffHire,"*Vacancy has been added in Full Time.Thank You!*")

END DO

ELSE

JOptionPane.showMessageDialog(frameStaffHire,"*Vacancy Number you have entered is already added.Please input **new** vacancy number.* ")

CATCH(NumberFormatException exp){

JOptionPane.showMessageDialog(frameStaffHire,"*Invalid Input,Please Try Again!*")

END DO

IF e.getSource() is equals to **btnAddPartTime**

DO

TRY

VacNumForPartTime = getText (txtFieldForVacancyNum_PT) **CONVERT TO INTEGER**

DesigNationForPartTime = getText (txtFieldForDesignation_PT)

String wh_P=getSelectedItem(comboBoxWorkingHour_PT) **CONVERT TO STRING**

workingHourForPartTime=wh_P(**CONVERT TO INTEGER**)

String WagesPrHr=getText(txtFieldForWgsPerHr_PT)

wagesPerHourForPartTime=WagesPrHr(**CONVERT TO INTEGER**)

DECLARE AND INTIALIZE boolean duplicateVacancyNumParTime is equals to false

FOR (StaffHire staffHire:list)

IF(var.getVacancyNumber() is equals to vacNumForPartTime)

DO

duplicateVacancyNumParTime=true

BREAK

END DO

IF(duplicateVacancyNumParTime is equals to false)

DO

PartTimeStaffHire obj=new PartTimeStaffHire(vacNumForPartTime, desigNationForPartTime,jobTypeCheckBox_PT,workingHourForPartTime,wagesPerHourForPartTime,workingShiftsRadioButton)

list.add(obj)

JOptionPane.showMessageDialog(frameStaffHire,"*Vacancy has been added in Part Time.Thank You!"*)

END DO

ELSE

DO

JOptionPane.showMessageDialog(frameStaffHire,"*Vacancy Number you have entered is already added.Please input new vacancy number.* ")

END DO

CATCH(**NumberFormatException** expe)

JOptionPane.showMessageDialog(frameStaffHire,"*Invalid Input!Please try again*")

IF e.getSource() is equals to **btnAppointFullTime**

DO

TRY

VacNumForFullTime = getText (txtFieldForVacancyNum_2_FT) **CONVERT TO INTEGER**

staffNameForFullTime=getText (txtFieldForStaffNam_FT)

yearFullTime =getSelectedItem(cmbYear_FT)**CONVERT TO STRING**

monthFullTime =getSelectedItem(cmbMonth_FT)**CONVERT TO STRING**

dayFullTime = getSelectedItem(cmbDay_FT)**CONVERT TO STRING**

joiningDateForFullTime = yearFullTime+"**CONCATINATE**"+monthFullTime+"**CONCATINATE**"+dayFullTime

String wh=getSelectedItem(comboBoxWorkingHour_FT) **CONVERT TO STRING**

workingHourForFullTime=wh(**CONVERT TO INTEGER**)

qualificationForFullTime=getSelectedItem(comboBoxQualification_FT)**CONVERT TO STRING**

wagesPerHourForFullTime=getText(txtFieldForWgsPerHr_FT)

appointedByForFullTime=getText(txtFieldForAppointedBy_FT)

DECLARE AND INTIALIZE boolean foundVacancyNum is equals to false

FOR(StaffHire staffHire:list)

IF(staffHire.getVacancyNumber() is equals to vacNumForFullTime)

DO

foundVacancyNum is equals to true

END DO

IF(staffHire instanceof FullTimeStaffHire)

DO

FullTimeStaffHire h=(FullTimeStaffHire)staffHire

IF(h.isJoined() is equals to true)

DO

JOptionPane.showMessageDialog(frameStaffHire,"*Staff has been hired already!!"*)

ELSE

h.hireFullTimeStaff(staffNameForFullTime,joiningDateForFullTime,qualification ForFullTime,appointedByForFullTime)

JOptionPane.showMessageDialog(frameStaffHire,"*Staff has been hired in Full Time.Thank You!!*")

BREAK

ELSE

JOptionPane.showMessageDialog(frameStaffHire,"*Not for fulltime staff Hire"*)

BREAK

END DO

IF(!foundVacancyNum)

DO

JOptionPane.showMessageDialog(frameStaffHire,"*Invalid Vacancy Number,Please Try Again!*")

END DO

CATCH(**EXCEPTION** e3)

DO

JOptionPane.showMessageDialog(frameStaffHire,"*Invalid Input,Please Try Again!*")

END DO

IF e.getSource() is equals to **btnAppointPartTime**

DO

TRY

VacNumForPartTime = getText (txtFieldForVacancyNum_2_PT) **CONVERT TO INTEGER**

staffNameForPartTime=getText (txtFieldForStaffNam_PT)

yearPartTime =getSelectedItem(cmbYear_PT)**CONVERT TO STRING**

monthPartTime =getSelectedItem(cmbMonth_PT)**CONVERT TO STRING**

dayPartTime = getSelectedItem(cmbDay_PT)**CONVERT TO STRING**

joiningDateForPartTime = yearPartTime+"**CONCATINATE**"+monthPartTime+"**CONCATINATE**"+dayPart Time

workingHourForPartTime=wh(**CONVERT TO INTEGER**)

qualificationForPartTime=getSelectedItem(comboBoxQualification_PT)**CONVE RT TO STRING**

radioButton_Morning=getText(radioButton_Morning_PT) **CONVERT TO STRING**

radioButton_Day=getText(radioButton_Day_PT) **CONVERT TO STRING**

String WagesPrHr= getText (txtFieldForWgsPerHr_PT) **CONVERT TO STRING**

wagesPerHourForPartTime=WagesPrHr(**CONVERT TO INTEGER**)

appointedByForPartTime=getText(txtFieldForAppointedBy_PT)

DECLARE AND INITIALIZE boolean foundVacancyNum is equals to false

FOR(StaffHire staffHire:list)

IF(staffHire.getVacancyNumber() is equals to vacNumForPartTime

DO

foundVacancyNum is equals to true

END DO

IF(staffHire instanceof PartTimeStaffHire)

DO

PartTimeStaffHire h2=(PartTimeStaffHire)staffHire

IF(h2.isHasJoined() is equals to true)

DO

JOptionPane.showMessageDialog(frameStaffHire,"*Staff has been hired already!!*")

END DO

ELSE

DO

h2.hirePartTimeStaff(staffNameForPartTime,joiningDateForPartTime,qualificationForPartTime,appointedByForPartTime)

JOptionPane.showMessageDialog(frameStaffHire,"*Staff has been hired in Part Time.Thank You!!*")

BREAK

END DO

ELSE

DO

JOptionPane.showMessageDialog(frameStaffHire,"*Not for Part time staff Hire"*)

BREAK

END DO

IF(!foundVacancyNum)

DO

JOptionPane.showMessageDialog(frameStaffHire,"*Invalid Vacancy Number,Please Try Again!*")

END DO

CATCH(Exception e3)

JOptionPane.showMessageDialog(frameStaffHire,"*Invalid Input,Please Try Again!*")

IF (e.getSource() is equals to **btnTerminate**)

DO

TRY

int vacNumToTerminate=getText(txtFieldToTerminate) CONVERT TO INTEGER

FOR(StaffHire objectTerminate:list)

IF(objectTerminate.getVacancyNumber()is equals to vacNumToTerminate)

DO

IF(objectTerminate instanceof PartTimeStaffHire)

DO

PartTimeStaffHire ptsh=(PartTimeStaffHire)objectTerminate;

IF(ptsh.isTerminated()is equals to false)

DO

CALL ptsh.terminatePartTimeStaff()

JOptionPane.showMessageDialog(frameStaffHire, "*Staff Terminated*")

BREAK

END DO

ELSE

DO

JOptionPane.showMessageDialog(frameStaffHire*, "Staff is already Terminated*")

BREAK

END DO

END DO

ELSE IF(objectTerminate.getVacancyNumber()!=vacNumToTerminate)

JOptionPane.showMessageDialog(frameStaffHire, "*Invalid Vacancy Number,Please Try Again!*")

CATCH(Exception exp3) {

JOptionPane.showMessageDialog(frameStaffHire, "*Invalid Input,Please Try Again!*")

IF (e.getSource () is equals to **btnDisplay**)

DO

FOR (StaffHire Object2: list)

IF (Object2 instanceof FullTimeStaffHire)

DO

FullTimeStaffHire o1= (FullTimeStaffHire) Object2

CALL o1.displayStaffHire ()

END DO

IF (Object2 instanceof PartTimeStaffHire)

DO

PartTimeStaffHire o2= (PartTimeStaffHire) Object2

CALL o2.displayStaffHire ()

END DO

```
            System.exit (0)

END DO
```

7. Error:

7.1 Syntax Error:

Syntax errors are a basically a type of error which only occurs when the program is during compiling phase. Actually, it is error of a syntax, which might occur due to the missing symbols, improper indentation, invalid numbers, inputs etc. It is exactly inverse to the runtime error, which are not recognized until the program is running. Every Programming language has its own rules and method of coding. Java syntax is inconceivably more straightforward than the sentence structure of English or some other "regular" language yet it is likewise a lot stricter. Keep a comma separate from sentence in English just makes the author look messy. During the compilation of program that I have done, missing of symbol error occurs frequently, but I was able to detect it and solve it.

```
for(StaffHire object:list) {

                if(object.getVacancyNumber()==vacNumToTerminate) {

        if(object instanceof PartTimeStaffHire) {
                PartTimeStaffHire ptsh=(PartTimeStaffHire)object;
                if(ptsh.isTerminated()==false) {
                        ptsh.terminatePartTimeStaff();
                        JOptionPane.showMessageDialog(frameStaffHire, "Staff Terminated);
                        break;                                          unclosed string literal
        }
                else{
                JOptionPane.showMessageDialog(frameStaffHire, "Staff is already Terminated");
                        break;
        }

}
}else if(object.getVacancyNumber()!=vacNumToTerminate){
        JOptionPane.showMessageDialog(frameStaffHire, "Invalid Vacancy Number,Please Try Again!");

}
```

Figure 13: Syntax Error

```
for(StaffHire object:list) {

                if(object.getVacancyNumber()==vacNumToTerminate) {

        if(object instanceof PartTimeStaffHire) {
                PartTimeStaffHire ptsh=(PartTimeStaffHire)object;
                if(ptsh.isTerminated()==false) {
                        ptsh.terminatePartTimeStaff();
                        JOptionPane.showMessageDialog(frameStaffHire, "Staff Terminated");
                        break;
        }
                else{
                JOptionPane.showMessageDialog(frameStaffHire, "Staff is already Terminated");
                        break;
        }

}
}else if(object.getVacancyNumber()!=vacNumToTerminate){
        JOptionPane.showMessageDialog(frameStaffHire, "Invalid Vacancy Number,Please Try Again!");

}
```

Figure 14: After the error is solved

7.2 Runtime Error:

A runtime error is a programming mistake that happens while the program is in the running phase. Runtime error is an error that occur after the compilation of program. When runtime error are found after a program has been compiled. Runtime error are logic error, IO error, Encoding error, etc. I encountered this error while running the program. When user input wrong or invalid input, the program throws exception. It's really difficult to detect, but I did it. I used Try and Catch to handle that exception, so, whenever user inputs wrong value, the program will not terminate, but, shows message of "Invalid Input".

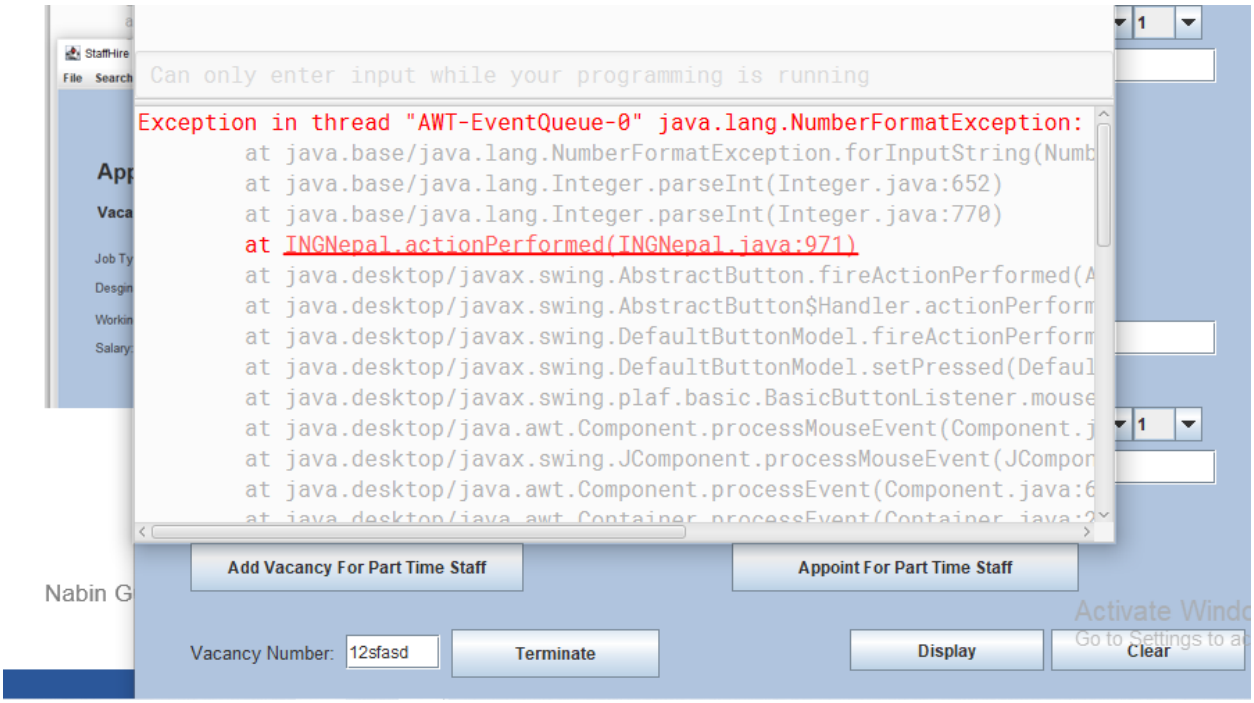

Figure 15: Run Time error

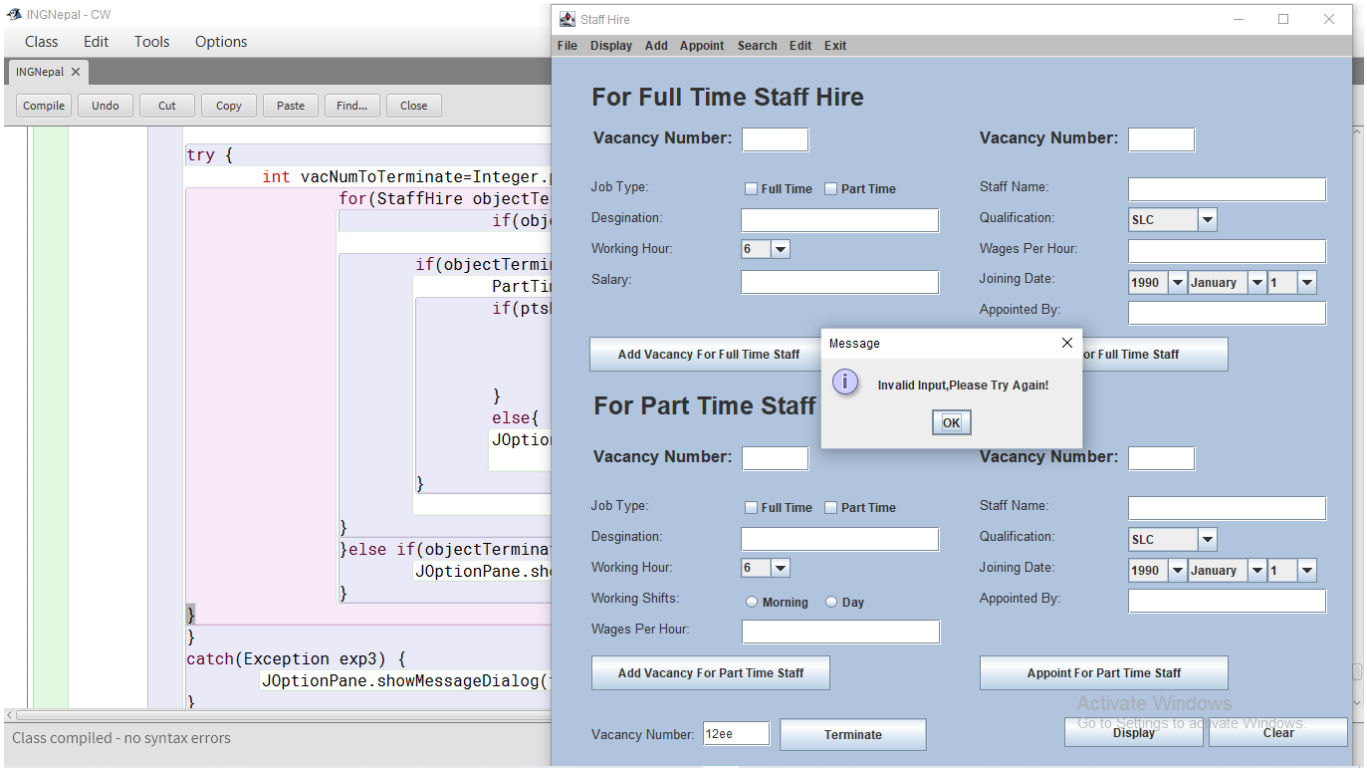

Figure 16: After the error is solved

7.3 Logical Error:

Logical Error in Java programming can be incredibly difficult to find considering the way that they don't reflect any sort of coding issue or a slip-up in the use of java language parts. Basi**CALL**y, program compiles and run as well, but it does the wrong thing. It just won't play out the program that you are expecting as your program to run. The code runs immaculately as made. These sorts of error may be the hardest to find. (dummies.com, 2020)

While doing program the outcome I get is not as I expected. The error is exceptionally basic however extremely elusive. Actually, I have to compare values by using two “==” operator, but mistakenly I used “=” which results me false input. Then, finally at last I found the error and resolved it.

```
ools    Options
Cut   Copy   Paste   Find...   Close

        boolean duplicateVacancyNumParTime=false;
            for(StaffHire var_P:list){
        if(var_P.getVacancyNumber()==vacNumForPartTime){
            duplicateVacancyNumParTime=true;
            break;
        }
    }
if(duplicateVacancyNumParTime=false){
    PartTimeStaffHire obj=new PartTimeStaffHire(vacNumForPartTime, desigNationForPartTime,jo
        list.add(obj);
    JOptionPane.showMessageDialog(frameStaffHire,"Vacancy has been added in Part Time.Thank
}else{
     JOptionPane.showMessageDialog(frameStaffHire,"Vacancy Number you have entered is alread
}
```

Figure 17: Logical Error

```
INGNepal - CW
Class   Edit   Tools   Options
INGNepal ×
Compile   Undo   Cut   Copy   Paste   Find...   Close

            boolean duplicateVacancyNumParTime=false;
                for(StaffHire var_P:list){
            if(var_P.getVacancyNumber()==vacNumForPartTime){
                duplicateVacancyNumParTime=true;
                break;
            }
        }
    if(duplicateVacancyNumParTime==false){
        PartTimeStaffHire obj=new PartTimeStaffHire(vacNumForPartTime, desigNationForPartTime
            list.add(obj);
        JOptionPane.showMessageDialog(frameStaffHire,"Vacancy has been added in Part Time.Tha
    }else{
         JOptionPane.showMessageDialog(frameStaffHire,"Vacancy Number you have entered is alr
    }

                }catch(NumberFormatException expe)      {
                        JOptionPane.showMessageDialog(frameStaffHire,"Invalid Input!Please tr
```

Figure 18: After error is solved

8. Appendix 1:

```
//importing different awt,swing classes
import java.awt.Color;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JOptionPane;
import javax.swing.JPanel;

import java.awt.Font;
import javax.swing.JTextField;
```

```
import javax.swing.JButton;
import javax.swing.JCheckBox;

import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JComboBox;
import javax.swing.JRadioButton;

//importing array list class
import java.util.ArrayList;

//creating class and implementing ActionListener interface
public class INGNepal implements ActionListener
{

//creating different instance variables
   private String
desigNationForFullTime,jobTypeForFullTime,staffNameForFullTime,joiningDat
eForFullTime,qualificationForFullTime,appointedByForFullTime,wagesPerHour
ForFullTime;
private int vacNumForFullTime,salaryForFullTime,workingHourForFullTime;
private String yearFullTime,monthFullTime,dayFullTime;

private String
desigNationForPartTime,jobTypeForPartTime,staffNameForPartTime,joiningDa
teForPartTime,qualificationForPartTime,appointedByForPartTime,radioButton_
Morning,radioButton_Day;
private int
vacNumForPartTime,workingHourForPartTime,salaryForPartTime,wagesPerHo
urForPartTime;
private String yearPartTime,monthPartTime,dayPartTime;
```

```
private String workingShiftsRadioButton;

private String jobTypeCheckBox_FT,jobTypeCheckBox_PT;


//creating different instance variables for GUI

private JFrame frameStaffHire;

private JPanel myPanel;


private JTextField
txtFieldForVacancyNum_FT,txtFieldForVacancyNum_2_FT,txtFieldForStaffNa
m_FT,txtFieldForDesignation_FT,txtFieldForWgsPerHr_FT,txtFieldForAppointe
dBy_FT,txtFieldForSalaryFT;

private JTextField
txtFieldForVacancyNum_PT,txtFieldForVacancyNum_2_PT,txtFieldForStaffNa
m_PT,txtFieldForDesignation_PT,txtFieldForWgsPerHr_PT,txtFieldForAppointe
dBy_PT,txtFieldToTerminate;

private JCheckBox
checkBoxFullTime_FT,checkBoxPartTime_FT,checkBoxFullTime_PT,checkBo
xPartTime_PT;

private JRadioButton radioButton_Morning_PT,radioButton_Day_PT;

private JComboBox<Object>
comboBoxWorkingHour_FT,comboBoxQualification_FT,comboBoxWorkingHou
r_PT,comboBoxQualification_PT,cmbYear_FT,cmbMonth_FT,cmbDay_FT,cmb
Year_PT,cmbMonth_PT,cmbDay_PT;

private JMenuBar menuBar;

private JMenu
fileMenu,exitMenu,searchMenu,aboutMenu,addMenu,appointMenu,displayMen
u;

private JMenuItem
openMenuItem,saveMenuItem,clearMenuItem,exitMenuItem,fullMenuItem,part
MenuItem,fullTimeToAppoint,partTimeToAppoint,fullTimeDisplayItem,partTime
DisplayItem;
```

```
private JButton
btnClear,btnDisplay,btnAppointFullTime,btnAddFullTime,btnAppointPartTime,bt
nAddPartTime,btnTerminate,btnTest;


    //making array list of "StaffHire" type
    ArrayList <StaffHire> list=new ArrayList<StaffHire>();


    public static void main(String[] args)
    {
        INGNepal object=new INGNepal();
        object.myGUI();  //calling method
    }


    //creating non-parameterized constructor
    //creating GUI inside constructor
    public void myGUI()
    {
          //creating frame
        frameStaffHire=new JFrame("Staff Hire");
        frameStaffHire.setVisible(true);
        frameStaffHire.setResizable(true);
        frameStaffHire.setBounds(293, 1, 820, 780);


        JPanel myPanel=new JPanel();
        myPanel.setLayout(null);


        Color cframe=new Color(176, 196, 222);
        myPanel.setBackground(cframe);
```

```
frameStaffHire.add(myPanel);
frameStaffHire.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

//creating menubar
menuBar=new JMenuBar();
menuBar.setBackground(Color.lightGray);
frameStaffHire.setJMenuBar(menuBar);



fileMenu=new JMenu("File");
menuBar.add(fileMenu);

        saveMenuItem=new JMenuItem("Save");
        fileMenu.add(saveMenuItem);

        clearMenuItem=new JMenuItem("Clear");
        clearMenuItem.addActionListener(this);
        fileMenu.add(clearMenuItem);

displayMenu=new JMenu("Display");
menuBar.add(displayMenu);

  fullTimeDisplayItem=new JMenuItem("Full Time");
  fullTimeDisplayItem.addActionListener(this);
  displayMenu.add(fullTimeDisplayItem);

  partTimeDisplayItem=new JMenuItem("Part Time");
  partTimeDisplayItem.addActionListener(this);
```

```
    displayMenu.add(partTimeDisplayItem);

  addMenu=new JMenu("Add");
  menuBar.add(addMenu);

    fullMenuItem=new JMenuItem("Full Time");
    fullMenuItem.addActionListener(this);
    addMenu.add(fullMenuItem);

    partMenuItem=new JMenuItem("Part Time");
    partMenuItem.addActionListener(this);
    addMenu.add(partMenuItem);

  appointMenu=new JMenu("Appoint");
  menuBar.add(appointMenu);

    fullTimeToAppoint=new JMenuItem("Full Time");
    fullTimeToAppoint.addActionListener(this);
    appointMenu.add(fullTimeToAppoint);

    partTimeToAppoint=new JMenuItem("Part Time");
    partTimeToAppoint.addActionListener(this);
    appointMenu.add(partTimeToAppoint);

  searchMenu=new JMenu("Search");
  menuBar.add(searchMenu);

  aboutMenu=new JMenu("Edit");
```

```
        menuBar.add(aboutMenu);

        exitMenu=new JMenu("Exit");
        menuBar.add(exitMenu);


//creating different labels in full time staff hire portion
        JLabel lblVacancyNumber = new JLabel("Vacancy Number:");
        lblVacancyNumber.setFont(new Font("Arial", Font.BOLD, 17));
        lblVacancyNumber.setBounds(42, 70, 190, 20);
        myPanel.add(lblVacancyNumber);

        JLabel lblVacancyNumber2 = new JLabel("Vacancy Number:");
        lblVacancyNumber2.setFont(new Font("Arial", Font.BOLD, 17));
        lblVacancyNumber2.setBounds(430, 70, 190, 20);
        myPanel.add(lblVacancyNumber2);

        JLabel lblStaffhire = new JLabel("For Full Time Staff Hire");
        lblStaffhire.setFont(new Font("Arial", Font.BOLD, 25));
        lblStaffhire.setBounds(40, 35, 300, 20);
        myPanel.add(lblStaffhire);

        JLabel lblStaffName = new JLabel("Staff Name:");
        lblStaffName.setFont(new Font("Arial", Font.PLAIN, 13));
        lblStaffName.setBounds(430, 120, 100, 20);
        myPanel.add(lblStaffName);

        JLabel lblJobType = new JLabel("Job Type:");
```

```
lblJobType.setFont(new Font("Arial", Font.PLAIN, 13));
lblJobType.setBounds(40, 120, 100, 20);
myPanel.add(lblJobType);

JLabel lblDesgination = new JLabel("Desgination:");
lblDesgination.setFont(new Font("Arial", Font.PLAIN, 13));
lblDesgination.setBounds(40, 151, 100, 20);
myPanel.add(lblDesgination);

JLabel lblWorkingHour = new JLabel("Working Hour:");
lblWorkingHour.setFont(new Font("Arial", Font.PLAIN, 13));
lblWorkingHour.setBounds(40, 182, 100, 20);
myPanel.add(lblWorkingHour);

JLabel lblWagesPerHour = new JLabel("Wages Per Hour:");
lblWagesPerHour.setFont(new Font("Arial", Font.PLAIN, 13));
lblWagesPerHour.setBounds(430, 182, 150, 20);
myPanel.add(lblWagesPerHour);

JLabel lblQualification = new JLabel("Qualification:");
lblQualification.setFont(new Font("Arial", Font.PLAIN, 13));
lblQualification.setBounds(430, 151, 100, 20);
myPanel.add(lblQualification);

JLabel lblJoiningDate = new JLabel("Joining Date:");
lblJoiningDate.setFont(new Font("Arial", Font.PLAIN, 13));
lblJoiningDate.setBounds(430, 213, 150, 18);
myPanel.add(lblJoiningDate);
```

```
        JLabel lblAppointedBy = new JLabel("Appointed By:");
        lblAppointedBy.setFont(new Font("Arial", Font.PLAIN, 13));
        lblAppointedBy.setBounds(430, 244, 150, 18);
        myPanel.add(lblAppointedBy);

        JLabel lblSalary = new JLabel("Salary:");
        lblSalary.setFont(new Font("Arial", Font.PLAIN, 13));
        lblSalary.setBounds(40, 213, 100, 20);
        myPanel.add(lblSalary);

//creating different labels in part time staff hire portion
        JLabel lblTitleParTime = new JLabel("For Part Time Staff Hire");
        lblTitleParTime.setFont(new Font("Arial", Font.BOLD, 25));
        lblTitleParTime.setBounds(42, 345, 300, 20);
        myPanel.add(lblTitleParTime);

        JLabel lblVacancyNumberPT = new JLabel("Vacancy Number:");
        lblVacancyNumberPT.setFont(new Font("Arial", Font.BOLD, 17));
        lblVacancyNumberPT.setBounds(42, 390, 190, 20);
        myPanel.add(lblVacancyNumberPT);

        JLabel lblVacancyNumber2PT = new JLabel("Vacancy Number:");
        lblVacancyNumber2PT.setFont(new Font("Arial", Font.BOLD, 17));
        lblVacancyNumber2PT.setBounds(430, 390, 190, 20);
        myPanel.add(lblVacancyNumber2PT);

        JLabel lblStaffNamePT = new JLabel("Staff Name:");
```

```
lblStaffNamePT.setFont(new Font("Arial", Font.PLAIN, 13));
lblStaffNamePT.setBounds(430, 440, 100, 20);
myPanel.add(lblStaffNamePT);

JLabel lblJobTypePT = new JLabel("Job Type:");
lblJobTypePT.setFont(new Font("Arial", Font.PLAIN, 13));
lblJobTypePT.setBounds(40,440,100,20);
myPanel.add(lblJobTypePT);

JLabel lblDesginationPT = new JLabel("Desgination:");
lblDesginationPT.setFont(new Font("Arial", Font.PLAIN, 13));
lblDesginationPT.setBounds(40, 471, 100, 20);
myPanel.add(lblDesginationPT);

JLabel lblWorkingHourPT = new JLabel("Working Hour:");
lblWorkingHourPT.setFont(new Font("Arial", Font.PLAIN, 13));
lblWorkingHourPT.setBounds(40, 502, 100, 20);
myPanel.add(lblWorkingHourPT);

JLabel lblWorkingShiftsPT = new JLabel("Working Shifts:");
lblWorkingShiftsPT.setFont(new Font("Arial", Font.PLAIN, 13));
lblWorkingShiftsPT.setBounds(40, 533, 100, 20);
myPanel.add(lblWorkingShiftsPT);

JLabel lblWagesPerHourPT = new JLabel("Wages Per Hour:");
lblWagesPerHourPT.setFont(new Font("Arial", Font.PLAIN, 13));
lblWagesPerHourPT.setBounds(40, 564, 150, 20);
myPanel.add(lblWagesPerHourPT);
```

```
JLabel lblQualificationPT = new JLabel("Qualification:");
lblQualificationPT.setFont(new Font("Arial", Font.PLAIN, 13));
lblQualificationPT.setBounds(430, 471, 100, 20);
myPanel.add(lblQualificationPT);

JLabel lblJoiningDatePT = new JLabel("Joining Date:");
lblJoiningDatePT.setFont(new Font("Arial", Font.PLAIN, 13));
lblJoiningDatePT.setBounds(430, 502, 150, 20);
myPanel.add(lblJoiningDatePT);

JLabel lblAppointedByPT = new JLabel("Appointed By:");
lblAppointedByPT.setFont(new Font("Arial", Font.PLAIN, 13));
lblAppointedByPT.setBounds(430, 533, 150, 20);
myPanel.add(lblAppointedByPT);

JLabel lblVacancyNumberTerminate = new JLabel("Vacancy Number:");
lblVacancyNumberTerminate.setFont(new Font("Arial", Font.PLAIN, 13));
lblVacancyNumberTerminate.setBounds(40, 670, 150, 20);
myPanel.add(lblVacancyNumberTerminate);


//creating different text fields in full time staff hire portion

txtFieldForVacancyNum_FT = new JTextField();
txtFieldForVacancyNum_FT.setBounds(191, 70, 68, 25);
myPanel.add(txtFieldForVacancyNum_FT);
```

```
        txtFieldForVacancyNum_2_FT = new JTextField();
        txtFieldForVacancyNum_2_FT.setBounds(579, 70, 68, 25);
        myPanel.add(txtFieldForVacancyNum_2_FT);

        txtFieldForStaffNam_FT = new JTextField();
        txtFieldForStaffNam_FT.setBounds(579, 120, 200, 25);
        myPanel.add(txtFieldForStaffNam_FT);

        txtFieldForDesignation_FT = new JTextField();
        txtFieldForDesignation_FT.setBounds(190, 151, 200, 25);
        myPanel.add(txtFieldForDesignation_FT);

        txtFieldForWgsPerHr_FT = new JTextField();
        txtFieldForWgsPerHr_FT.setBounds(579, 182, 200, 25);
        myPanel.add(txtFieldForWgsPerHr_FT);

        txtFieldForAppointedBy_FT = new JTextField();
        txtFieldForAppointedBy_FT.setBounds(579, 244, 200, 25);
        myPanel.add(txtFieldForAppointedBy_FT);

        txtFieldForSalaryFT = new JTextField();
        txtFieldForSalaryFT.setBounds(190, 213, 200, 25);
        myPanel.add(txtFieldForSalaryFT);


//creating different text fields in part time staff hire portion

        txtFieldForVacancyNum_PT = new JTextField();
```

```
txtFieldForVacancyNum_PT.setBounds(191, 390, 68, 25);
myPanel.add(txtFieldForVacancyNum_PT);

txtFieldForVacancyNum_2_PT = new JTextField();
txtFieldForVacancyNum_2_PT.setBounds(579, 390, 68, 25);
myPanel.add(txtFieldForVacancyNum_2_PT);

txtFieldForStaffNam_PT = new JTextField();
txtFieldForStaffNam_PT.setBounds(579, 440, 200, 25);
myPanel.add(txtFieldForStaffNam_PT);

txtFieldForDesignation_PT = new JTextField();
txtFieldForDesignation_PT.setBounds(190, 471, 200, 25);
myPanel.add(txtFieldForDesignation_PT);

txtFieldForWgsPerHr_PT = new JTextField();
txtFieldForWgsPerHr_PT.setBounds(191, 564, 200, 25);
myPanel.add(txtFieldForWgsPerHr_PT);

txtFieldForAppointedBy_PT = new JTextField();
txtFieldForAppointedBy_PT.setBounds(579, 533, 200, 25);
myPanel.add(txtFieldForAppointedBy_PT);

txtFieldToTerminate = new JTextField();
txtFieldToTerminate.setBounds(152, 666, 68, 25);
myPanel.add(txtFieldToTerminate);
```

```
//creating different check boxes

    checkBoxFullTime_FT = new JCheckBox("Full Time");
    Color cfull1=new Color(176, 196, 222);
    checkBoxFullTime_FT.setBackground(cfull1);
    checkBoxFullTime_FT.setBounds(190, 120, 80, 23);
    myPanel.add(checkBoxFullTime_FT);

    checkBoxPartTime_FT = new JCheckBox("Part Time");
    Color cpart1=new Color(176, 196, 222);
    checkBoxPartTime_FT.setBackground(cpart1);
    checkBoxPartTime_FT.setBounds(270, 120, 80, 23);
    myPanel.add(checkBoxPartTime_FT);



    checkBoxFullTime_PT = new JCheckBox("Full Time");
    Color cfull2=new Color(176, 196, 222);
    checkBoxFullTime_PT.setBackground(cfull2);
    checkBoxFullTime_PT.setBounds(190, 440, 80, 23);
    myPanel.add(checkBoxFullTime_PT);

    checkBoxPartTime_PT = new JCheckBox("Part Time");
    Color cpart2=new Color(176, 196, 222);
    checkBoxPartTime_PT.setBackground(cpart2);
    checkBoxPartTime_PT.setBounds(270, 440, 80, 23);
    myPanel.add(checkBoxPartTime_PT);
```

```
//creating radiobutton

        radioButton_Morning_PT = new JRadioButton("Morning");
        Color m2=new Color(176, 196, 222);
        radioButton_Morning_PT.setBackground(m2);
        radioButton_Morning_PT.setBounds(191, 533, 80, 27);
        myPanel.add(radioButton_Morning_PT);

        radioButton_Day_PT = new JRadioButton("Day");
        Color d2=new Color(176, 196, 222);
        radioButton_Day_PT.setBackground(d2);
        radioButton_Day_PT.setBounds(271, 533, 68, 27);
        myPanel.add(radioButton_Day_PT);


//creating different Combo Boxes

        String workinghour[]= {"6","7","8","9","10","11","12","13","14"};
        comboBoxWorkingHour_FT = new JComboBox<Object>(workinghour);
        comboBoxWorkingHour_FT.setBounds(190, 182, 50, 20);
        myPanel.add(comboBoxWorkingHour_FT);

        String qualification[]= {"SLC","+2","Bachelor","Master"};
        comboBoxQualification_FT = new JComboBox<Object>(qualification);
        comboBoxQualification_FT.setBounds(579, 150, 90, 25);
        myPanel.add(comboBoxQualification_FT);
```

```
String workinghourPT[]= {"6","7","8","9","10","11","12","13","14"};
comboBoxWorkingHour_PT = new JComboBox<Object>(workinghourPT);
comboBoxWorkingHour_PT.setBounds(190, 502, 50, 20);
myPanel.add(comboBoxWorkingHour_PT);

String qualificationPT[]= {"SLC","+2","Bachelor","Master"};
comboBoxQualification_PT = new JComboBox<Object>(qualificationPT);
comboBoxQualification_PT.setBounds(579, 471, 90, 25);
myPanel.add(comboBoxQualification_PT);



String year[]={"1990", "1991", "1992", "1993", "1994", "1995", "1996",
"1997", "1998", "1999", "2000", "2001", "2002", "2003", "2004", "2005", "2006",
"2007", "2008", "2009", "2010", "2011", "2012",
"2013","2014","2015","2016","2017","2018","2019","2020","2021"};
cmbYear_FT=new JComboBox<Object>(year);
cmbYear_FT.setBounds(579, 213, 60, 25);
myPanel.add(cmbYear_FT);

String
month[]={"January","February","March","April","May","June","July","August","Se
ptember","October","November","December"};
cmbMonth_FT=new JComboBox<Object>(month);
cmbMonth_FT.setBounds(639, 213, 80, 25);
myPanel.add(cmbMonth_FT);

String
day[]={"1","2","3","4","5","6","7","8","9","10","11","12","13","14","15","16","17","1
8","19","20","21","22","23","24","25","26","27","28","29","30","31"};
cmbDay_FT=new JComboBox<Object>(day);
```

```
        cmbDay_FT.setBounds(719, 213, 50, 25);
        myPanel.add(cmbDay_FT);



        String yearPT[]={"1990", "1991", "1992", "1993", "1994", "1995", "1996",
"1997", "1998", "1999", "2000", "2001", "2002", "2003", "2004", "2005", "2006",
"2007", "2008", "2009", "2010", "2011", "2012",
"2013","2014","2015","2016","2017","2018","2019","2020","2021"};
        cmbYear_PT=new JComboBox<Object>(yearPT);
        cmbYear_PT.setBounds(579, 502, 60, 25);
        myPanel.add(cmbYear_PT);


        String
monthPT[]={"January","February","March","April","May","June","July","August","
September","October","November","December"};
        cmbMonth_PT=new JComboBox<Object>(monthPT);
        cmbMonth_PT.setBounds(639, 502, 80, 25);
        myPanel.add(cmbMonth_PT);


        String
dayPT[]={"1","2","3","4","5","6","7","8","9","10","11","12","13","14","15","16","17",
"18","19","20","21","22","23","24","25","26","27","28","29","30","31"};
        cmbDay_PT=new JComboBox<Object>(dayPT);
        cmbDay_PT.setBounds(719, 502, 50, 25);
        myPanel.add(cmbDay_PT);




//creating different Buttons


        btnClear = new JButton("Clear");
```

```
btnClear.addActionListener(this);
btnClear.setBounds(660, 662, 140, 30);
myPanel.add(btnClear);

btnDisplay = new JButton("Display");
btnDisplay.setBounds(515, 662, 140, 30);
btnDisplay.addActionListener(this);
myPanel.add(btnDisplay);

btnAddFullTime = new JButton("Add Vacancy For Full Time Staff");
btnAddFullTime.setBounds(38, 280,240, 35);
btnAddFullTime.addActionListener(this);
myPanel.add(btnAddFullTime);

btnAddPartTime = new JButton("Add Vacancy For Part Time Staff");
btnAddPartTime.setBounds(40, 600, 240, 35);
btnAddPartTime.addActionListener(this);
myPanel.add(btnAddPartTime);

btnAppointFullTime = new JButton("Appoint For Full Time Staff");
btnAppointFullTime.setBounds(430,280, 250, 35);
btnAppointFullTime.addActionListener(this);
myPanel.add(btnAppointFullTime);

btnAppointPartTime = new JButton("Appoint For Part Time Staff");
btnAppointPartTime.setBounds(430, 600, 250, 35);
btnAppointPartTime.addActionListener(this);
myPanel.add(btnAppointPartTime);
```

```
        btnTerminate = new JButton("Terminate");
        btnTerminate.addActionListener(this);
        btnTerminate.setBounds(228, 662, 150, 35);
        myPanel.add(btnTerminate);



    }



    @Override
    //Overriding actionPerformed method of ActionListener interface
    public void actionPerformed(ActionEvent e)
    {

        //when specific radio button "radioButton_Morning_PT" is selected,then
setting "radioButton_Day_PT" radio button to false
        if(radioButton_Morning_PT.isSelected()) {
                radioButton_Day_PT.setSelected(false);
        }
        if(radioButton_Day_PT.isSelected()) {
                radioButton_Morning_PT.setSelected(false);
        }

        //when specific radio button 'radioButton_Morning_PT" is selected,then
assigning "workingShiftsRadioButton" value to "workingShiftsRadioButton"
        workingShiftsRadioButton="";
        if(radioButton_Morning_PT.isSelected()) {
```

```
            workingShiftsRadioButton="Morning";
        }
        if(radioButton_Day_PT.isSelected()) {
            workingShiftsRadioButton="Day";
        }



        jobTypeCheckBox_FT="";
        if(checkBoxFullTime_FT.isSelected()) {
            checkBoxPartTime_FT.setSelected(false);
        }
        if(checkBoxPartTime_FT.isSelected()) {
            checkBoxFullTime_FT.setSelected(false);
        }

        if(checkBoxFullTime_FT.isSelected()) {
            jobTypeCheckBox_FT="Full Time";
        }
        if(checkBoxPartTime_FT.isSelected()) {
            jobTypeCheckBox_FT="Part Time";
        }



        jobTypeCheckBox_PT="";
        if(checkBoxFullTime_PT.isSelected()) {
            checkBoxPartTime_PT.setSelected(false);
```

```
        }
        if(checkBoxPartTime_PT.isSelected()) {
                checkBoxFullTime_PT.setSelected(false);
        }

        if(checkBoxFullTime_PT.isSelected()) {
                jobTypeCheckBox_PT="Full Time";
        }
        if(checkBoxPartTime_PT.isSelected()) {
                jobTypeCheckBox_PT="Part Time";
        }





      if(e.getSource()==btnClear) {
        //when "Clear" button is clicked,then all the provided values in radio
button,text field,check box and combo box are set to empty

          txtFieldForVacancyNum_FT.setText("");
          txtFieldForStaffNam_FT.setText("");
          txtFieldForDesignation_FT.setText("");
          txtFieldForStaffNam_FT.setText("");
          txtFieldForWgsPerHr_FT.setText("");
          txtFieldForAppointedBy_FT.setText("");
          txtFieldForSalaryFT.setText("");
          txtFieldForVacancyNum_2_FT.setText("");
```

```
radioButton_Morning_PT.setSelected(false);
radioButton_Day_PT.setSelected(false);

cmbYear_FT.setSelectedIndex(0);
cmbMonth_FT.setSelectedIndex(0);
cmbDay_FT.setSelectedIndex(0);
comboBoxWorkingHour_FT.setSelectedIndex(0);
comboBoxQualification_FT.setSelectedIndex(0);

checkBoxFullTime_FT.setSelected(false);
checkBoxPartTime_FT.setSelected(false);
checkBoxFullTime_PT.setSelected(false);
checkBoxPartTime_PT.setSelected(false);

txtFieldForVacancyNum_PT.setText("");
txtFieldForStaffNam_PT.setText("");
txtFieldForDesignation_PT.setText("");
txtFieldForStaffNam_PT.setText("");
txtFieldForWgsPerHr_PT.setText("");
txtFieldToTerminate.setText("");
txtFieldForAppointedBy_PT.setText("");
txtFieldForVacancyNum_2_PT.setText("");

cmbYear_PT.setSelectedIndex(0);
cmbMonth_PT.setSelectedIndex(0);
cmbDay_PT.setSelectedIndex(0);
comboBoxQualification_PT.setSelectedIndex(0);
comboBoxWorkingHour_PT.setSelectedIndex(0);
```

```
    }
    if(e.getSource()==clearMenuItem) {

      //when "Clear" Menu Item is clicked,then all the provided values in radio
button,text field,check box and combo box are set to empty

        txtFieldForVacancyNum_FT.setText("");
        txtFieldForStaffNam_FT.setText("");
        txtFieldForDesignation_FT.setText("");
        txtFieldForStaffNam_FT.setText("");
        txtFieldForWgsPerHr_FT.setText("");
        txtFieldForAppointedBy_FT.setText("");
        txtFieldForSalaryFT.setText("");
        txtFieldForVacancyNum_2_FT.setText("");

        radioButton_Morning_PT.setSelected(false);
        radioButton_Day_PT.setSelected(false);

        cmbYear_FT.setSelectedIndex(0);
        cmbMonth_FT.setSelectedIndex(0);
        cmbDay_FT.setSelectedIndex(0);
        comboBoxWorkingHour_FT.setSelectedIndex(0);
        comboBoxQualification_FT.setSelectedIndex(0);

        checkBoxFullTime_FT.setSelected(false);
        checkBoxPartTime_FT.setSelected(false);
        checkBoxFullTime_PT.setSelected(false);
        checkBoxPartTime_PT.setSelected(false);
```

```
            txtFieldForVacancyNum_PT.setText("");
            txtFieldForStaffNam_PT.setText("");
            txtFieldForDesignation_PT.setText("");
            txtFieldForStaffNam_PT.setText("");
            txtFieldForWgsPerHr_PT.setText("");
            txtFieldToTerminate.setText("");
            txtFieldForAppointedBy_PT.setText("");
            txtFieldForVacancyNum_2_PT.setText("");

            cmbYear_PT.setSelectedIndex(0);
            cmbMonth_PT.setSelectedIndex(0);
            cmbDay_PT.setSelectedIndex(0);
            comboBoxQualification_PT.setSelectedIndex(0);
            comboBoxWorkingHour_PT.setSelectedIndex(0);
        }

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

           //when "Full" Menu Item is clicked,then all the provided values in radio
button,text field,check box and combo box are set to empty

           try {

vacNumForFullTime=Integer.parseInt(txtFieldForVacancyNum_FT.getText());
            desigNationForFullTime=txtFieldForDesignation_FT.getText();
                              String
wh=(comboBoxWorkingHour_FT.getSelectedItem()).toString();
               workingHourForFullTime=Integer.parseInt(wh);
```

```
salaryForFullTime=Integer.parseInt(txtFieldForSalaryFT.getText());


boolean duplicateVacancyNum=false;
for(StaffHire var:list){
if(var.getVacancyNumber()==vacNumForFullTime){
duplicateVacancyNum=true;
break;
}
}
if(duplicateVacancyNum==false){
FullTimeStaffHire objectFullTime=new
FullTimeStaffHire(vacNumForFullTime,
desigNationForFullTime,jobTypeCheckBox_FT,salaryForFullTime,workingHour
ForFullTime);
list.add(objectFullTime);
JOptionPane.showMessageDialog(frameStaffHire,"Vacancy has been
added in Full Time.Thank You!");
}else{
JOptionPane.showMessageDialog(frameStaffHire,"Vacancy Number
you have entered is already added.Please input new vacancy number. ");
}


}catch(NumberFormatException exp){
JOptionPane.showMessageDialog(frameStaffHire,"Invalid Input,Please
Try Again!");
}


}


if(e.getSource()==fullTimeToAppoint) {
```

```
        //when "Appoint Full Time" button is clicked,then following things should
happen
         try {

                //getting and converting values of text field,combo box to string

vacNumForFullTime=Integer.parseInt(txtFieldForVacancyNum_2_FT.getText())
;
          staffNameForFullTime=txtFieldForStaffNam_FT.getText();
                        yearFullTime = (cmbYear_FT.getSelectedItem()).toString();
                        monthFullTime =
(cmbMonth_FT.getSelectedItem()).toString();
                        dayFullTime = (cmbDay_FT.getSelectedItem()).toString();

           joiningDateForFullTime =
yearFullTime+"/"+monthFullTime+"/"+dayFullTime;
                String
wh=(comboBoxWorkingHour_FT.getSelectedItem()).toString();
                //String wh = getSelectedItem(comboBoxWorkingHour_FT)
CONVERT TO STRING
           workingHourForFullTime=Integer.parseInt(wh);

qualificationForFullTime=(comboBoxQualification_FT.getSelectedItem()).toStri
ng();
           wagesPerHourForFullTime=txtFieldForWgsPerHr_FT.getText();
           appointedByForFullTime=txtFieldForAppointedBy_FT.getText();

          boolean foundVacancyNum=false;
          //iterating array list
```

```
        for(StaffHire staffHire:list) {
            if(staffHire.getVacancyNumber()==vacNumForFullTime) {
                foundVacancyNum=true;
                // using "instanceof" to check if object is in FullTimeStaffHire class
                if(staffHire instanceof FullTimeStaffHire) {
                    FullTimeStaffHire h=(FullTimeStaffHire)staffHire;
                    if(h.isJoined()==true){
                        JOptionPane.showMessageDialog(frameStaffHire,"Staff has
been hired already!!");
                    }else{

h.hireFullTimeStaff(staffNameForFullTime,joiningDateForFullTime,qualification
ForFullTime,appointedByForFullTime);
                        JOptionPane.showMessageDialog(frameStaffHire,"Staff has
been hired in Full Time.Thank You!!");
                        break;
                    }
                }else{
                    JOptionPane.showMessageDialog(frameStaffHire,"Not for
fulltime staff Hire");
                    break;
                }
            }

        }

        if(!foundVacancyNum){
            JOptionPane.showMessageDialog(frameStaffHire,"Invalid Vacancy
Number,Please Try Again!");
```

```
            }

            }catch(Exception e3) {
                JOptionPane.showMessageDialog(frameStaffHire,"Invalid
Input,Please Try Again!");
            }
        }

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

          try {

vacNumForPartTime=Integer.parseInt(txtFieldForVacancyNum_2_PT.getText()
);
                staffNameForPartTime=txtFieldForStaffNam_PT.getText();
                    yearPartTime = (cmbYear_PT.getSelectedItem()).toString();
                    monthPartTime = (cmbMonth_PT.getSelectedItem()).toString();
                    dayPartTime = (cmbDay_PT.getSelectedItem()).toString();
                joiningDateForPartTime =
yearPartTime+"/"+monthPartTime+"/"+dayPartTime;

qualificationForPartTime=(comboBoxQualification_PT.getSelectedItem()).toStri
ng();
                radioButton_Morning=(radioButton_Morning_PT.getText()).toString();
                radioButton_Day=(radioButton_Day_PT.getText()).toString();
                    String WagesPrHr=(txtFieldForWgsPerHr_PT.getText());
                wagesPerHourForPartTime=Integer.parseInt(WagesPrHr);
                appointedByForPartTime=txtFieldForAppointedBy_PT.getText();
```

```
            boolean foundVacancyNum=false;
            for(StaffHire staffHire:list) {
                if(staffHire.getVacancyNumber()==vacNumForPartTime) {
                    foundVacancyNum=true;
                    if(staffHire instanceof PartTimeStaffHire) {  //instanceof help to
check,if the object is present in FTSH
                        PartTimeStaffHire h2=(PartTimeStaffHire)staffHire;
                        if(h2.isHasJoined()==true){
                            JOptionPane.showMessageDialog(frameStaffHire,"Staff
has been hired already!!");
                        }else{

h2.hirePartTimeStaff(staffNameForPartTime,joiningDateForPartTime,qualificati
onForPartTime,appointedByForPartTime);
                            JOptionPane.showMessageDialog(frameStaffHire,"Staff has
been hired in Part Time.Thank You!!");
                            break;
                        }
                    }else{
                        JOptionPane.showMessageDialog(frameStaffHire,"Not for
Part time staff Hire");
                        break;
                    }
                }
            }



            if(!foundVacancyNum){
                JOptionPane.showMessageDialog(frameStaffHire,"Invalid
Vacancy Number,Please Try Again!");
            }
```

```
                }catch(Exception e3) {
                JOptionPane.showMessageDialog(frameStaffHire,"Invalid
Input,Please Try Again!");
            }
        }

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

          for(StaffHire staffHire:list) {
                    if(staffHire instanceof FullTimeStaffHire) {
                        FullTimeStaffHire o1=(FullTimeStaffHire)staffHire;
                        o1.displayStaffHire();
                    }
          }
          System.exit(0);
        }

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

          for(StaffHire staffHire:list) {
                    if(staffHire instanceof PartTimeStaffHire) {
                        PartTimeStaffHire o1=(PartTimeStaffHire)staffHire;
                        o1.displayStaffHire();
                    }
          }
          System.exit(0);
        }
```

```
    if(e.getSource()==partMenuItem) {

      try {

      vacNumForPartTime=Integer.parseInt(txtFieldForVacancyNum_PT.getT
ext());            //converting String value entered in text field into Integer
          desigNationForPartTime=txtFieldForDesignation_PT.getText();
              String
wh_P=(comboBoxWorkingHour_PT.getSelectedItem()).toString();
          workingHourForPartTime=Integer.parseInt(wh_P);
              String WagesPrHr=(txtFieldForWgsPerHr_PT.getText());
          wagesPerHourForPartTime=Integer.parseInt(WagesPrHr);


          boolean duplicateVacancyNumParTime=false;
             for(StaffHire staffHire:list){
          if(staffHire.getVacancyNumber()==vacNumForPartTime){
             duplicateVacancyNumParTime=true;
             break;
          }
             }
    if(duplicateVacancyNumParTime==false){
       PartTimeStaffHire obj=new PartTimeStaffHire(vacNumForPartTime,
desigNationForPartTime,jobTypeCheckBox_PT,workingHourForPartTime,wage
sPerHourForPartTime,workingShiftsRadioButton);
          list.add(obj);
       JOptionPane.showMessageDialog(frameStaffHire,"Vacancy has been
added in Part Time.Thank You!");
    }else{
        JOptionPane.showMessageDialog(frameStaffHire,"Vacancy Number
you have entered is already added.Please input new vacancy number. ");
```

```
        }

          }catch(NumberFormatException expe)    {
                        JOptionPane.showMessageDialog(frameStaffHire,"Invalid
Input!Please try again");
          }



        }

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

             try {

vacNumForFullTime=Integer.parseInt(txtFieldForVacancyNum_FT.getText());
                    desigNationForFullTime=txtFieldForDesignation_FT.getText();
                        String
wh=(comboBoxWorkingHour_FT.getSelectedItem()).toString();
                workingHourForFullTime=Integer.parseInt(wh);

salaryForFullTime=Integer.parseInt(txtFieldForSalaryFT.getText());


                  boolean duplicateVacancyNum=false;
                     for(StaffHire var:list){
                  if(var.getVacancyNumber()==vacNumForFullTime){
                     duplicateVacancyNum=true;
                     break;
                  }
               }
           if(duplicateVacancyNum==false){
```

```
            FullTimeStaffHire obj=new FullTimeStaffHire(vacNumForFullTime,
desigNationForFullTime,jobTypeCheckBox_FT,salaryForFullTime,workingHour
ForFullTime);

                list.add(obj);

            JOptionPane.showMessageDialog(frameStaffHire,"Vacancy has
been added in Full Time.Thank You!");

        }else{

             JOptionPane.showMessageDialog(frameStaffHire,"Vacancy Number
you have entered is already added.Please input new vacancy number. ");

        }


        }catch(NumberFormatException exp){

           JOptionPane.showMessageDialog(frameStaffHire,"Invalid
Input,Please Try Again!");

        }


       }


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


                        try {


       vacNumForPartTime=Integer.parseInt(txtFieldForVacancyNum_PT.getT
ext());            //converting String value entered in text field into Integer

                desigNationForPartTime=txtFieldForDesignation_PT.getText();

                 String
wh_P=(comboBoxWorkingHour_PT.getSelectedItem()).toString();

                workingHourForPartTime=Integer.parseInt(wh_P);

                 String WagesPrHr=(txtFieldForWgsPerHr_PT.getText());

                wagesPerHourForPartTime=Integer.parseInt(WagesPrHr);
```

```
                boolean duplicateVacancyNumParTime=false;
                    for(StaffHire var_P:list){
                if(var_P.getVacancyNumber()==vacNumForPartTime){
                    duplicateVacancyNumParTime=true;
                    break;
                }
            }
        if(duplicateVacancyNumParTime==false){
            PartTimeStaffHire obj=new PartTimeStaffHire(vacNumForPartTime,
desigNationForPartTime,jobTypeCheckBox_PT,workingHourForPartTime,wage
sPerHourForPartTime,workingShiftsRadioButton);
                list.add(obj);
            JOptionPane.showMessageDialog(frameStaffHire,"Vacancy has
been added in Part Time.Thank You!");
        }else{
             JOptionPane.showMessageDialog(frameStaffHire,"Vacancy Number
you have entered is already added.Please input new vacancy number. ");
        }


                        }catch(NumberFormatException expe)    {

        JOptionPane.showMessageDialog(frameStaffHire,"Invalid Input!Please
try again");
                        }


        }



        if(e.getSource()==btnAppointFullTime) {
```

```
            try {


vacNumForFullTime=Integer.parseInt(txtFieldForVacancyNum_2_FT.getText())
;
                    staffNameForFullTime=txtFieldForStaffNam_FT.getText();
                    yearFullTime = (cmbYear_FT.getSelectedItem()).toString();
                    monthFullTime = (cmbMonth_FT.getSelectedItem()).toString();
                    dayFullTime = (cmbDay_FT.getSelectedItem()).toString();


                joiningDateForFullTime =
yearFullTime+"/"+monthFullTime+"/"+dayFullTime;
                 String
wh=(comboBoxWorkingHour_FT.getSelectedItem()).toString();
                workingHourForFullTime=Integer.parseInt(wh);

qualificationForFullTime=(comboBoxQualification_FT.getSelectedItem()).toStri
ng();
                wagesPerHourForFullTime=txtFieldForWgsPerHr_FT.getText();
                appointedByForFullTime=txtFieldForAppointedBy_FT.getText();


            boolean foundVacancyNum=false;
            for(StaffHire obj:list) {
               if(obj.getVacancyNumber()==vacNumForFullTime) {
                  foundVacancyNum=true;
                  if(obj instanceof FullTimeStaffHire) {  //instanceof help to
check,if the object is present in FTSH
                     FullTimeStaffHire h=(FullTimeStaffHire)obj;
                     if(h.isJoined()==true){
                        JOptionPane.showMessageDialog(frameStaffHire,"Staff
has been hired already!!");
```

```
                    }else{

h.hireFullTimeStaff(staffNameForFullTime,joiningDateForFullTime,qualification
ForFullTime,appointedByForFullTime);
                      JOptionPane.showMessageDialog(frameStaffHire,"Staff has
been hired in Full Time.Thank You!!");
                      break;
                    }
                  }else{
                    JOptionPane.showMessageDialog(frameStaffHire,"Not for
fulltime staff Hire");
                    break;
                  }
                }

            }

            if(!foundVacancyNum){
                JOptionPane.showMessageDialog(frameStaffHire,"Invalid
Vacancy Number,Please Try Again!");

            }

            }catch(Exception e3) {
                JOptionPane.showMessageDialog(frameStaffHire,"Invalid
Input,Please Try Again!");
            }

        }
        if(e.getSource()==btnAppointPartTime) {
```

```
        try {

vacNumForPartTime=Integer.parseInt(txtFieldForVacancyNum_2_PT.getText()
);
            staffNameForPartTime=txtFieldForStaffNam_PT.getText();
                yearPartTime = (cmbYear_PT.getSelectedItem()).toString();
                monthPartTime = (cmbMonth_PT.getSelectedItem()).toString();
                dayPartTime = (cmbDay_PT.getSelectedItem()).toString();
            joiningDateForPartTime =
yearPartTime+"/"+monthPartTime+"/"+dayPartTime;

qualificationForPartTime=(comboBoxQualification_PT.getSelectedItem()).toStri
ng();
            radioButton_Morning=(radioButton_Morning_PT.getText()).toString();
            radioButton_Day=(radioButton_Day_PT.getText()).toString();
                String WagesPrHr=(txtFieldForWgsPerHr_PT.getText());
            wagesPerHourForPartTime=Integer.parseInt(WagesPrHr);
            appointedByForPartTime=txtFieldForAppointedBy_PT.getText();

            boolean foundVacancyNum=false;
            for(StaffHire obj:list) {
               if(obj.getVacancyNumber()==vacNumForPartTime) {
                  foundVacancyNum=true;
                  if(obj instanceof PartTimeStaffHire) {
                     PartTimeStaffHire h2=(PartTimeStaffHire)obj;
                     if(h2.isHasJoined()==true){
                        JOptionPane.showMessageDialog(frameStaffHire,"Staff
has been hired already!!");
```

```
                    }else{

h2.hirePartTimeStaff(staffNameForPartTime,joiningDateForPartTime,qualificati
onForPartTime,appointedByForPartTime);

                      JOptionPane.showMessageDialog(frameStaffHire,"Staff has
been hired in Part Time.Thank You!!");

                      break;

                    }

                  }else{

                    JOptionPane.showMessageDialog(frameStaffHire,"Not for
Part time staff Hire");

                    break;

                  }

                  }

            }



            if(!foundVacancyNum){

                JOptionPane.showMessageDialog(frameStaffHire,"Invalid
Vacancy Number,Please Try Again!");

            }



          }

            catch(Exception e3) {

                JOptionPane.showMessageDialog(frameStaffHire,"Invalid
Input,Please Try Again!");

            }


      }
```

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

try {
int
vacNumToTerminate=Integer.parseInt(txtFieldToTerminate.getText());

for(StaffHire objectTerminate:list) {

if(objectTerminate.getVacancyNumber()==vacNumToTerminate) {

if(objectTerminate instanceof
PartTimeStaffHire) {
PartTimeStaffHire
ptsh=(PartTimeStaffHire)objectTerminate;
if(ptsh.isTerminated()==false) {
ptsh.terminatePartTimeStaff();

JOptionPane.showMessageDialog(frameStaffHire, "Staff Terminated");

break;
}
else{
```

```
        JOptionPane.showMessageDialog(frameStaffHire, "Staff is already
Terminated");
                                                        break;
                                        }


                                }
                                }else
if(objectTerminate.getVacancyNumber()!=vacNumToTerminate){

        JOptionPane.showMessageDialog(frameStaffHire, "Invalid Vacancy
Number,Please Try Again!");


                                }
                }

                }

                catch(Exception exp3) {
                        JOptionPane.showMessageDialog(frameStaffHire, "Invalid
Input,Please Try Again!");
                }



        }

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

                //iterating arraylist
                //using instanceof to check the presence of object in list
```

```
                    for(StaffHire Object2:list) {
                        if(Object2 instanceof FullTimeStaffHire) {
                            FullTimeStaffHire
o1=(FullTimeStaffHire)Object2;
                            o1.displayStaffHire();

                        }
                        if(Object2 instanceof PartTimeStaffHire) {
                            PartTimeStaffHire
o2=(PartTimeStaffHire)Object2;
                            o2.displayStaffHire();
                    }

                    System.exit(0);
        }

        }

    }

    }
```

8 Appendix 2:

1. Class Diagram

StaffHire
-Vacancynumber:int -designation:String -jobType:String
+StaffHire (Vacancynumber:int, designation:String, jobType:String) +setVacancynumber(Vacancy number:int):void +setJobType(Job Type:String):void +setdesignation(Designation:String):void + getDesignation():String + getJobType():String + getVacancyNumber():int + display ():void

Table 14: Class diagram for Staff Hire

FullTimeStaffHire
-salary:float -workinghours:int -staffName:String -joiningDate:String -qualification:String -appointedBy:String -joined:Boolean
+FullTimeStaffHire(vacancyNumber:int, designation:String, jobType:String, salary:int, workingHour:int) (vacancyNumber: int, designation: String, jobType: String):super +getSalary():int +getworkinghours():int +getstaffName():String +getJoiningDate():String +getqualification():String +getappointedBy():String +getjoined():Boolean +set workinghours (new workinghours:int):void +FullTimeStaffHire(salary:int ,workingHour:String, staffName:String ,joiningDate: String,qualification: String,appointedBy: String):void + display ():void

Table 15: Class diagram for Full Time Staff Hire

PartTimeStaffHire
-workingHour:float -wagesPerHour:int -staffName:String -joiningDate:String -qualification:String -appointedBy:String -shifts:String -joined:Boolean -terminated:Boolean
+PartTimeStaffHire(vacancyNumber:int, designation:String, jobType:String, workingHour:int, wagesPerHour:int, shifts:String) + (vacancyNumber:int, designation:String, jobType:String): super + getwagesPerHour (): int + workingHour (): int +shifts (): String +staffName (): String +appointedBy (): String +qualification (): String +joiningDate (): String +Terminated (): String +hasJoined (): String +setshifts(shifts:String):void + hirePartTimeStaff(staffName:String, joiningDate: String, qualification: String, appointedBy: String):void +terminate(Terminate:Boolean):void +display(): void

Table 16: Class diagram for Part Time Staff Hire


StaffHire
-Vacancynumber:int
-designation:String
-jobType:String
+StaffHire (Vacancynumber:int, designation:String, jobType:String)
+setVacancynumber(Vacancy number:int):void
+setJobType(Job Type:String):void
+setdesignation(Designation:String):void
+ getDesignation():String
+ getJobType():String
+ getVacancyNumber():int
+ display ():void
FullTimeStaffHire
-salary:float
-workinghours:int
-staffName:String
-joiningDate:String
-qualification:String
-appointedBy:String
-joined:Boolean
+FullTimeStaffHire(vacancyNumber:int, designation:String, jobType:String, salary:int, workingHour:int)
(vacancyNumber: int, designation: String, jobType: String):super
+getSalary():int
+getworkinghours():int
+getstaffName():String
+getJoiningDate():String
+getqualification():String
+getappointedBy():String
+getjoined():Boolean
+set workinghours (new workinghours:int):void
+FullTimeStaffHire(salary:int ,workingHour:String, staffName:String ,joiningDate: String,qualification: String,appointedBy: String):void
+ display ():void
PartTimeStaffHire
-workingHour:float
-wagesPerHour:int
-staffName:String
-joiningDate:String
-qualification:String
-appointedBy:String
-shifts:String
-joined:Boolean
-terminated:Boolean
+PartTimeStaffHire(vacancyNumber:int, designation:String, jobType:String, workingHour:int, wagesPerHour:int, shifts:String)
+ (vacancyNumber:int, designation:String, jobType:String): super
+ getwagesPerHour (): int
+ workingHour (): int
+shifts (): String
+staffName (): String
+appointedBy (): String
+qualification (): String
+joiningDate (): String
+Terminated (): String
+hasJoined (): String
+setshifts(shifts:String):void
+ hirePartTimeStaff(staffName:String, joiningDate: String, qualification: String, appointedBy: String):void
+terminate(Terminate:Boolean):void
+display(): void

2. Pseudo Code

The pseudo code for different class is as follows:

2.1 For StaffHire:

```
CREATE StaffHire Class

DECLARE class variables

                String Designation, int Vacancy Number, String Job Type;

FUNCTION getDesignation()

      DO

     Return Designation;

END DO

FUNCTION setDesignation(String Designation)

       DO

this.Designation=Designation;

END DO


FUNCTION getVacancynumber ()

         DO

        Return Vacancynumber;

END DO

FUNCTION setVacancynumber(String Vacancynumber)

         DO

       this.Vacancynumber=Vacancynumber;

END DO


FUNCTION getJobType ()
```

```
    DO
RETURN JobType;
END DO
FUNCTION setJobType (String JobType)
    DO
this.JobType=JobType;
END DO


FUNCTION display ()
        DO
            PRINT this.getvancancy_number();
            PRINT this.getdesignation();
        PRINT this.getjob_type();
END DO
```

2.2 For FullTimeStaffHire:

```
CREATE StaffHire Class
DECLARE class variables
        String Designation, int Vacancy Number, String Job Type;
FUNCTION getDesignation()
        DO
    Return Designation;
END DO
```

```
FUNCTION setDesignation(String Designation)
        DO
this.Designation=Designation;
END DO


FUNCTION getVacancynumber ()
          DO
        Return Vacancynumber;
END DO
FUNCTION setVacancynumber(String Vacancynumber)
          DO
       this.Vacancynumber=Vacancynumber;
END DO


FUNCTION getJobType ()
      DO
RETURN JobType;
END DO
FUNCTION setJobType (String JobType)
       DO
this.JobType=JobType;
END DO
```

FUNCTION display ()

DO

PRINT this.getvancancy_number();

PRINT this.getdesignation();

PRINT this.getjob_type();

END DO

2.3 For PartTimeStaffHire:

CREATE class PartTimeStaffHire

DECLARE class variables

int salary,workingHour;

String staffName,joiningDate,qualification,appointedBy;

boolean hasJoined;

FUNCTION getWorkingHour (int)

DO

RETURN workingHour;

END DO

FUNCTION setWorkingHour(String WorkingHour) **DO**

this.workingHour=workingHour;

END DO

FUNCTION getStaffName (String)

DO

RETURN staffName;

END DO

```
        FUNCTION setStaffName(String
StaffName)  DO          this.staffName=staffName;
 END DO


  FUNCTION getJoiningDate (String)
        DO
       RETURN joiningDate;
  END DO


  FUNCTION setJoiningDate(String JoiningDate)
        DO
this.joiningDate=joiningDate;
END DO


   FUNCTION getQualification (String)
       DO
      RETURN qualification;
   END DO


   FUNCTION setQualification(String Qualification)
        DO
this.qualification=qualification;
END DO


   FUNCTION getAppointedBy (String)
        DO
```

```
            RETURN appointedBy;
  END DO

  FUNCTION setAppointedBy(String AppointedBy)
DO    this.appointedBy=appointedBy;
END DO
  FUNCTION getSalary (String)
          DO
        RETURN salary;
  END DO
  FUNCTION setSalary(String Salary)
            DO
this.salary=salary;
END DO

  FUNCTION isJoined (boolean)
              DO
              RETURN hasJoined;
  END DO
  FUNCTION setJoined(Boolean Joined)
              DO
this.joined=joined;
END DO
```

```
    FUNCTION setHasJoined(boolean
hasJoined)              DO
this.hasJoined=hasJoined;


     FUNCTION intSalary (int Salary)
              DO
         IF hasJoined equals to false PRINT “Staff has been already appointed.”;
                    ELSE this.workingHour = salary;
                  END DO
                   END FUNCTION
END DO


  FUNCTION int workingHour(int workingHour)
           DO
  IF hasJoined equals to false PRINT "Staff has been already appointed.";
          ELSE this.workingHour = workingHour;
          END IF
           END DO
            END FUNCTION
END DO


        FUNCTION hirePartTimeStaff (String StaffName, String JoiningDate, String
Qualification, String AppointedBy)
          DO
  IF Joined equals to false PRINT "Staff has been already appointed.";
```

ELSE this.staffName=staffName; this.joiningDate=joiningDate; this.qualification=qualification; this.appointedBy=appointedBy; this.hasjoined=true; this.Terminated=false;

END IF

END DO

END FUNCTION

END DO

FUNCTION displayStaffHire()

DO

CALLING Display () method of super class IF hasJoined=true PRINT StaffName, Salary, WorkingHour, JoiningDate, Qualification, AppointedBy, ;

END IF

END DO

END FUNCTION

END DO

3. Method Description

3.1 Method Description for StaffHire:

Method	**Description**
getDesignation	This method is used to return the Designation.
setDesignation	This method is used to set a new value for the Designation.
getJobType	This method is used to return the Job Type.

setJobType	This method is used to set a new value for the Job Type.
getVacancyNumber	This method is used to return the Vacancy Number.
setVacancyNumber	This method is used to set a new value for the Vacancy Number.
display	This method is used to display the attributes.

Table 17: Method Description for Staff Hire

3.2 Method Description for FullTimeStaffHire:

Method	**Description**
getWorkingHour	This method is used to return the WorkingHour.
setWorkingHour	This method is used to set a new value for the WorkingHour
getSalary	This method is used to return the Salary.
setSalary	This method is used to set a new value for the Salary
getStaffName	This method is used to return the StaffName.
setStaffName	This method is used to set a new value for the StaffName
getQualification	This method is used to return the Qualification.
setQualification	This method is used to set a new value for the Qualification.

getAppointedBy	This method is used to return the AppointedBy.
setAppointedBy	This method is used to set a new value for the AppointedBy
getJoiningDate	This method is used to return the JoiningDate.
setJoiningDate	This method is used to set a new value for the JoiningDate
setJoined	This method is used to set a new value for the Joined.

Table 18: Method Description for Full Time Staff Hire

3.3 Method Description for PartTimeStaffHire:

Method	Description
getWagesPerHour	This method is used to return the WagesPerHour.
setWagesPerHour	This method is used to set a new value for the WagesPerHour.
getWorkingHour	This method is used to return the WorkingHour.
setWorkingHour	This method is used to set a new value for the WorkingHour.
getStaffName	This method is used to return the StaffName.
setStaffName	This method is used to set a new value for the StaffName.
getAppointedBy	This method is used to return the AppointedBy.

setAppointedBy	This method is used to set a new value for the AppointedBy.
getJoiningDate	This method is used to return the JoiningDate.
getQualification	This method is used to return the Qualification.
setQualification	This method is used to set a new value for the Qualification
getShifts	This method is used to return the Shifts.
setShifts	This method is used to set a new value for the Shifts.
setHasJoined	This method is used to set a new value for the HasJoined.
setTerminated	This method is used to set a new value for the Terminated.

Table 19: Method Description for Part Time Staff Hire

4. Testing

4.1 Test 1-To Inspect FullTimeStaffHire Class, appoint the full time staff, and reinspect the FullTimeStaffHireClass

Test No.	1
Objectives	To Inspect FullTimeStaffHire Class, appoint the full time staff, and reinspect the FullTimeStaffHire Class
Action	Constructor is called. salary - input salary in int workingHour - input workingHour in int staffName - input staffName in String joiningDate - input joiningDate in String qualification - input qualification in String appointedBy - input appointedBy in String joined - input joined in boolean The object is re-inspected.
Excpected Output(Result)	Should show empty

Actual Output(Result)	Excepted result is displayed in fig 2 and 3.
Test Result	Test is Successful.

Table 20: To Inspect FullTimeStaffHire Class, appoint the full time staff, and reinspect the FullTimeStaffHire Class.

OUTPUT RESULT:

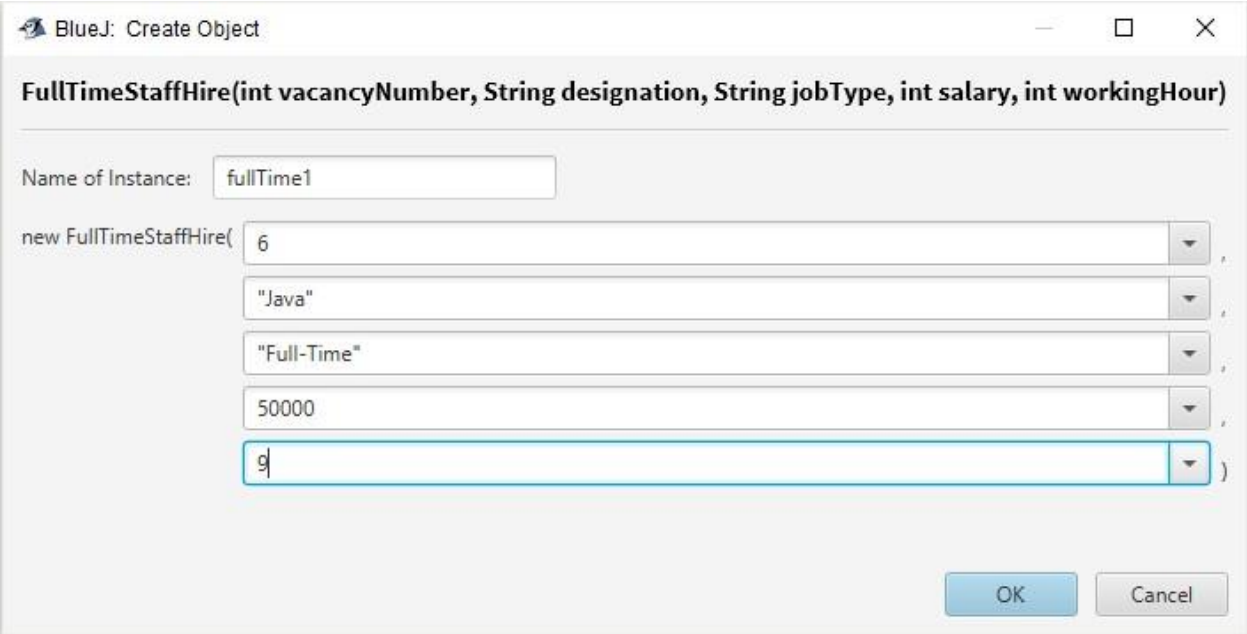

Figure 19: Screenshot of appointing staff in Full Time Staff Hire

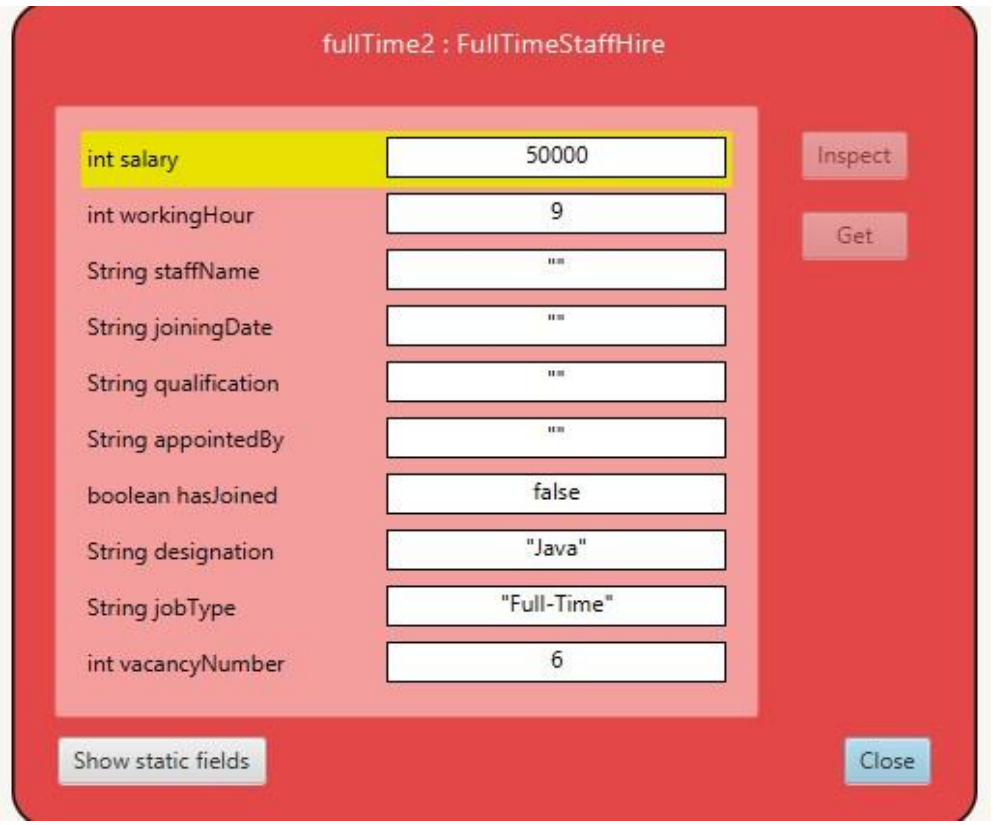

Figure 20: Screenshot of Inspection of FullTimeStaffHire

Figure 21: Screenshot of ReInspection of FullTimeStaffHire

4.2 Test 2- To Inspect PartTimeStaffHire Class, appoint part time staff, and reinspect the PartTimeStaffHire Class.

Test No.	2
Objectives	To Inspect PartTimeStaffHire Class, appoint part time staff, and reinspect the PartTimeStaffHire Class
Action	Constructor is called. workingHour -input workingHour in int wagesPerHour - input wagesPerHour in int staffName - input staffName in String joiningDate - input joiningDate in String qualification - input qualification in String appointedBy - input appointedBy in String shifts - input shifts in String joined - input joined in (boolean) terminated - input terminated in (boolean)The object is re-inspected.
Excpected Output(Result)	Should show empty.
Actual Output(Result)	Excepted result is displayed in fig 5 and 6.
Test Result	Test is Successful.

Table 21: To Inspect PartTimeStaffHire Class, appoint part time staff, and reinspect the PartTimeStaffHire Class

OUTPUT RESULT:

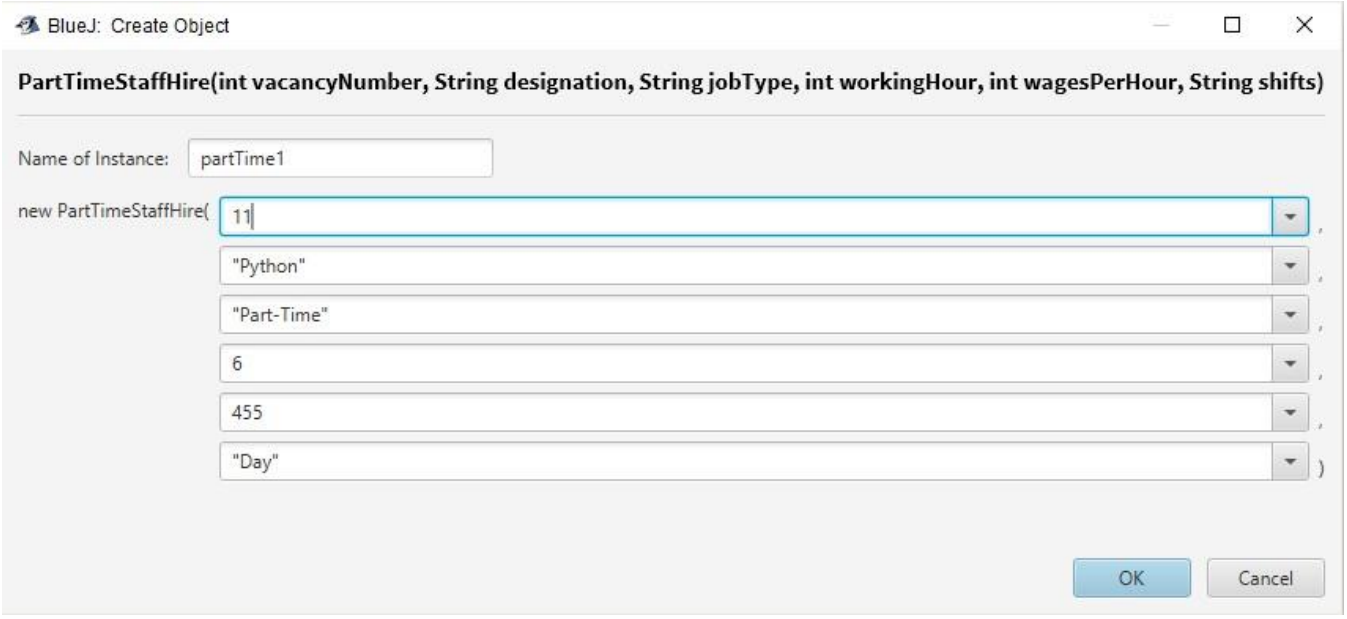

Figure 22: Screenshot of appointing staff in Part Time Staff Hire

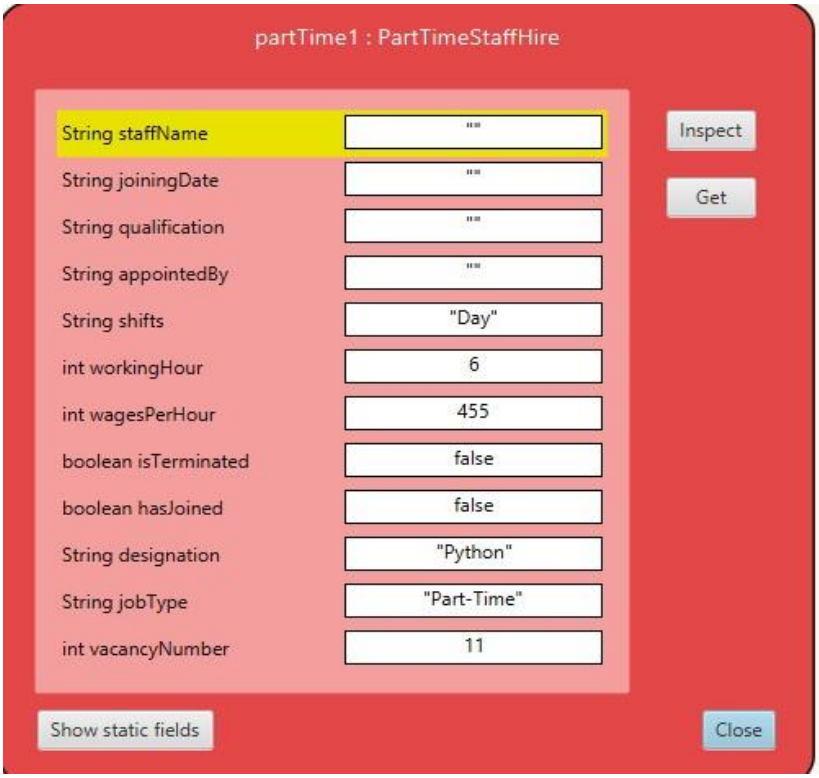

Figure 23: Screenshot of Inspection of PartTimeStaffHire

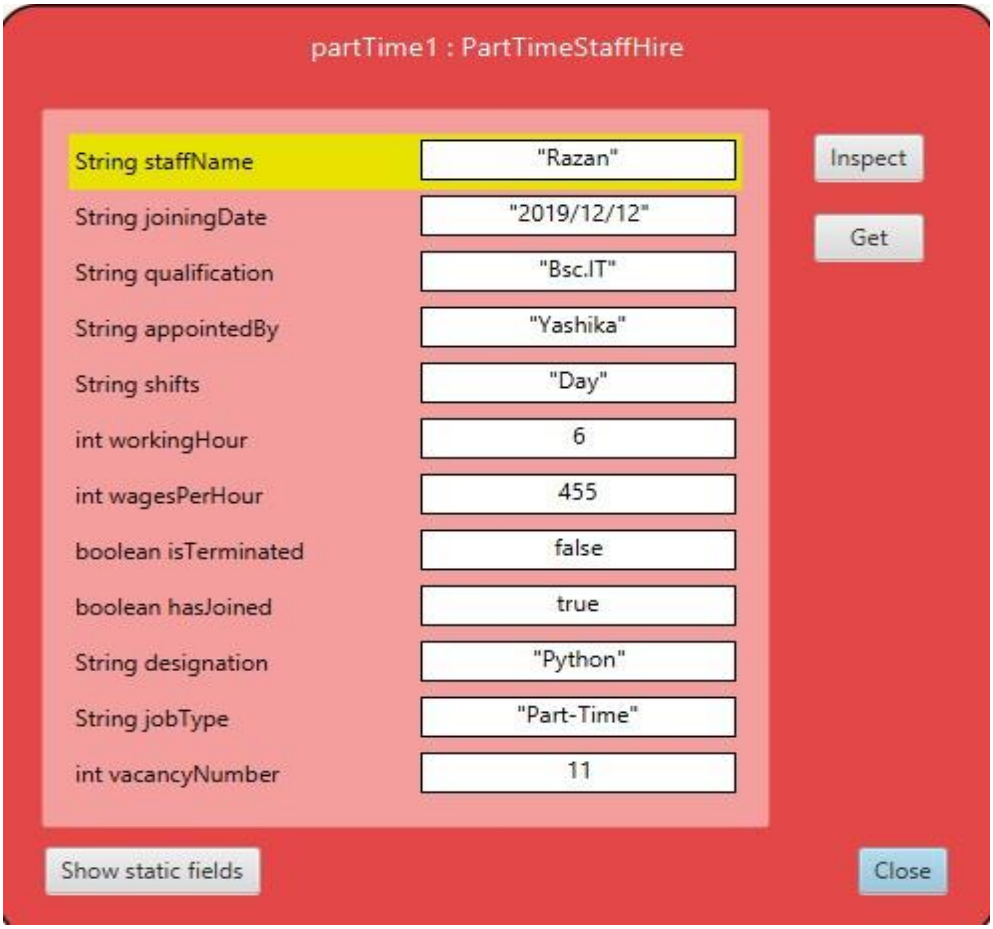

Figure 24: Screenshot of Re Inspect of PartTimeStaffHire

4.3 Test 3- To Inspect PartTimeStaffHire Class, change the termination status of a staff, and re-inspect the PartTimeStaffHire Class

Test No.	3
Objectives	To Inspect PartTimeStaffHire Class, change the termination status of a staff, and re-inspect the PartTimeStaffHire Class
Action	Constructor is called. Object is created. Void Terminated is called. The object is re-inspected.
Excpected Output(Result)	When inspecting the PartTimeStaffHire joined status of appointed staff should be true and terminated status is false. But when the staff is terminated the joined status should be false and terminated status should be true.
Actual Output(Result)	Excepted result is displayed in fig 7 and 8.
Test Result	Test is Successful.

Table 22: To Inspect PartTimeStaffHire Class, change the termination status of a staff, and re-inspect the PartTimeStaffHire Class.

OUTPUT RESULT:

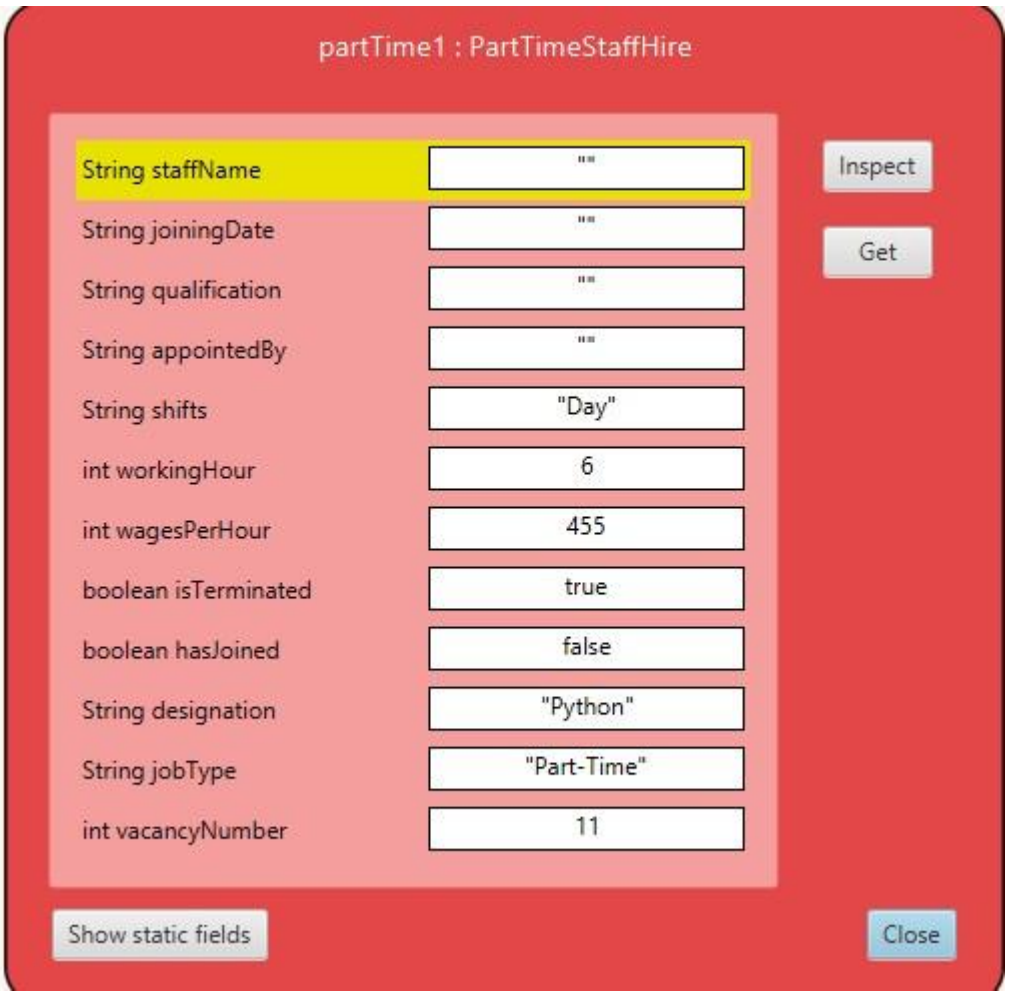

Figure 25: Screenshot of Inspection of PartTimeStaffHire after Termination

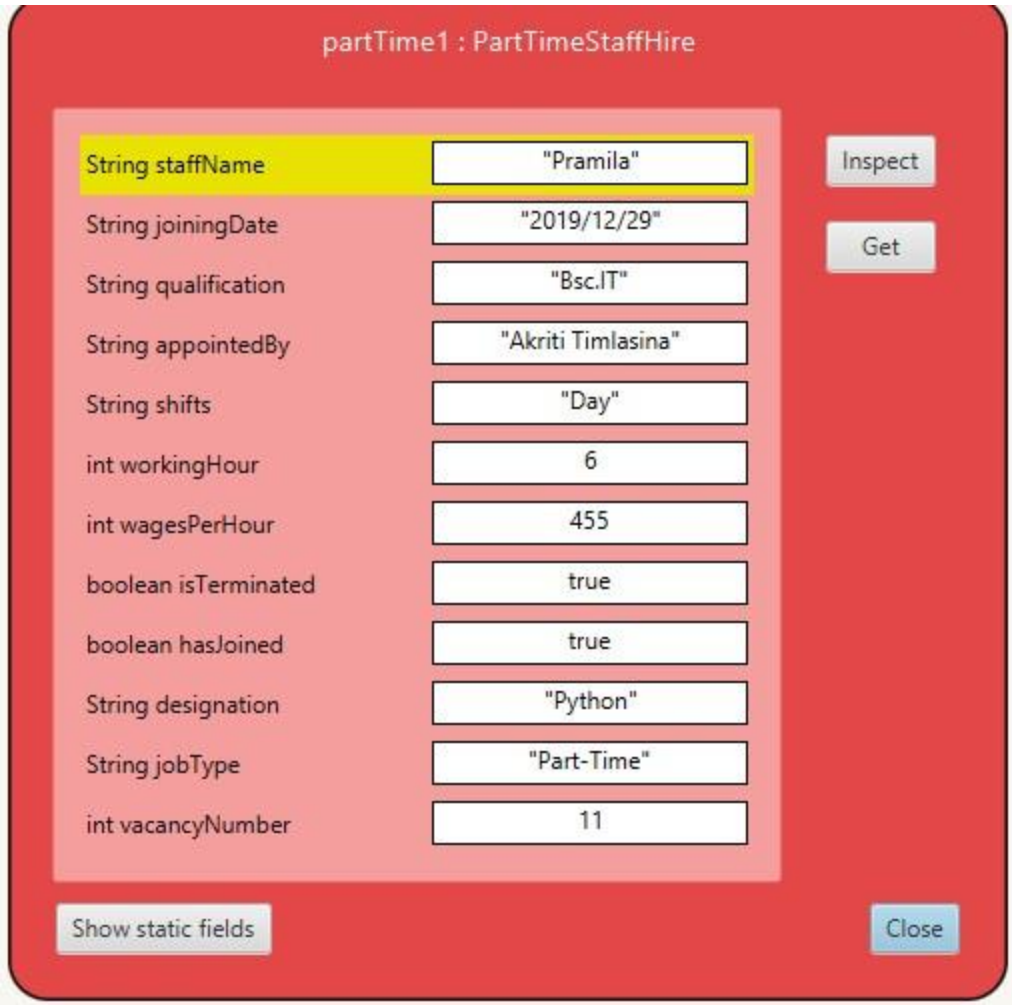

Figure 26: Screenshot of Reinspection of PartTimeStaffHire after Termination

4.4 Test 4- To Display the detail of FullTimeStaffHire and PartTimeStaffHire Class.

Test No.	4
Objectives	To Display the detail of FullTimeStaffHire and PartTimeStaffHire Class.
Action	Display details.
Expected Output(Result)	Should display the details of Full and Part Time Staff Hire Class.
Actual Output(Result)	Excepted result is displayed in fig 9 and 10.
Test Result	Test is Successful.

Table 23: To Display the detail of FullTimeStaffHire and PartTimeStaffHire Class.

OUTPUT RESULT:

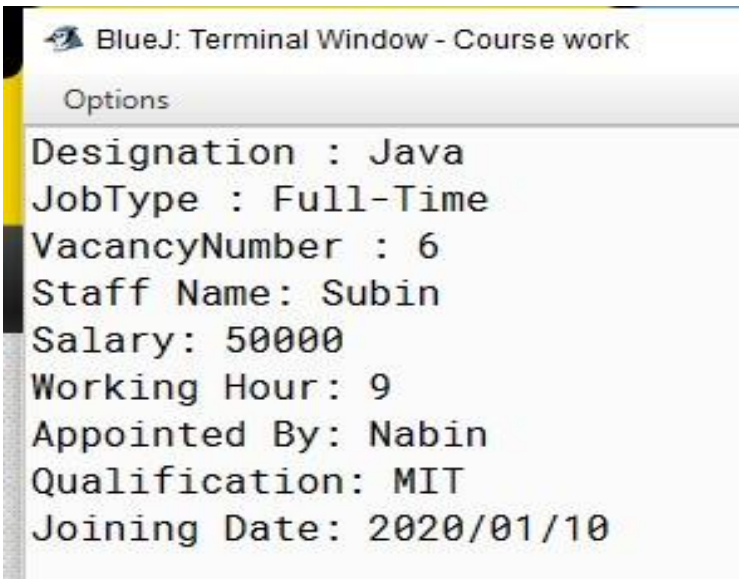

Figure 27: Screenshot of Display of Full Time Staff Hire

```
Designation : Python
JobType : Part-Time
VacancyNumber : 11
Staff Name: Razan
Qualification: Bsc.IT
Wages per hour: 455
Working Hour: 6
Income per day: 2730
Appointed By: Yashika
Joining Date: 2019/12/12
```

Figure 28: Screenshot of Display of Part Time Staff Hire

5. Error Detection

There are 3 types of errors in Java and they are:

- Syntax Error
- Runtime Error
- Logical Error

5.1 Syntax Error

Syntax errors are a sort of compiler mistake which will be identified promptly if the software engineer attempts to change over his source code into a program. This is inverse to runtime blunders, which are not recognized until the program is really running. Java has its very own linguistic structure. For example, one rule of Java syntax is that all commands must end with a semicolon (;). (techwalla.com, 2020)

During compiling of my program I get error message. I checked my program and I am able to find the error. My error is nothing big, but the missing of simple of symbol (;), which I correct.

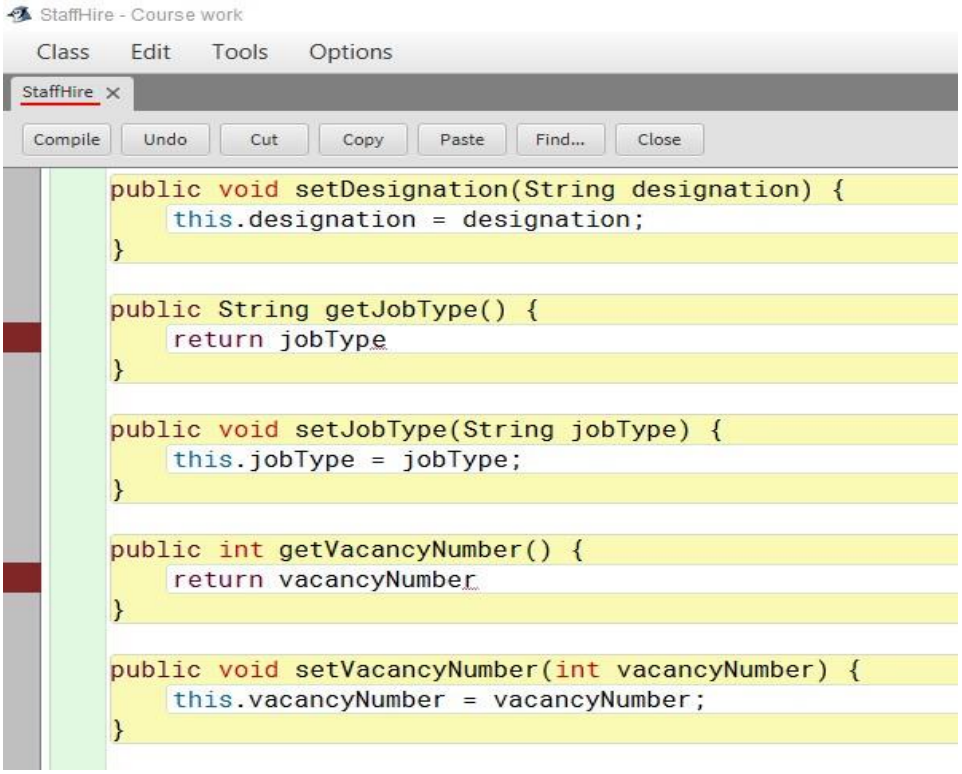

Figure 29: Syntax Error

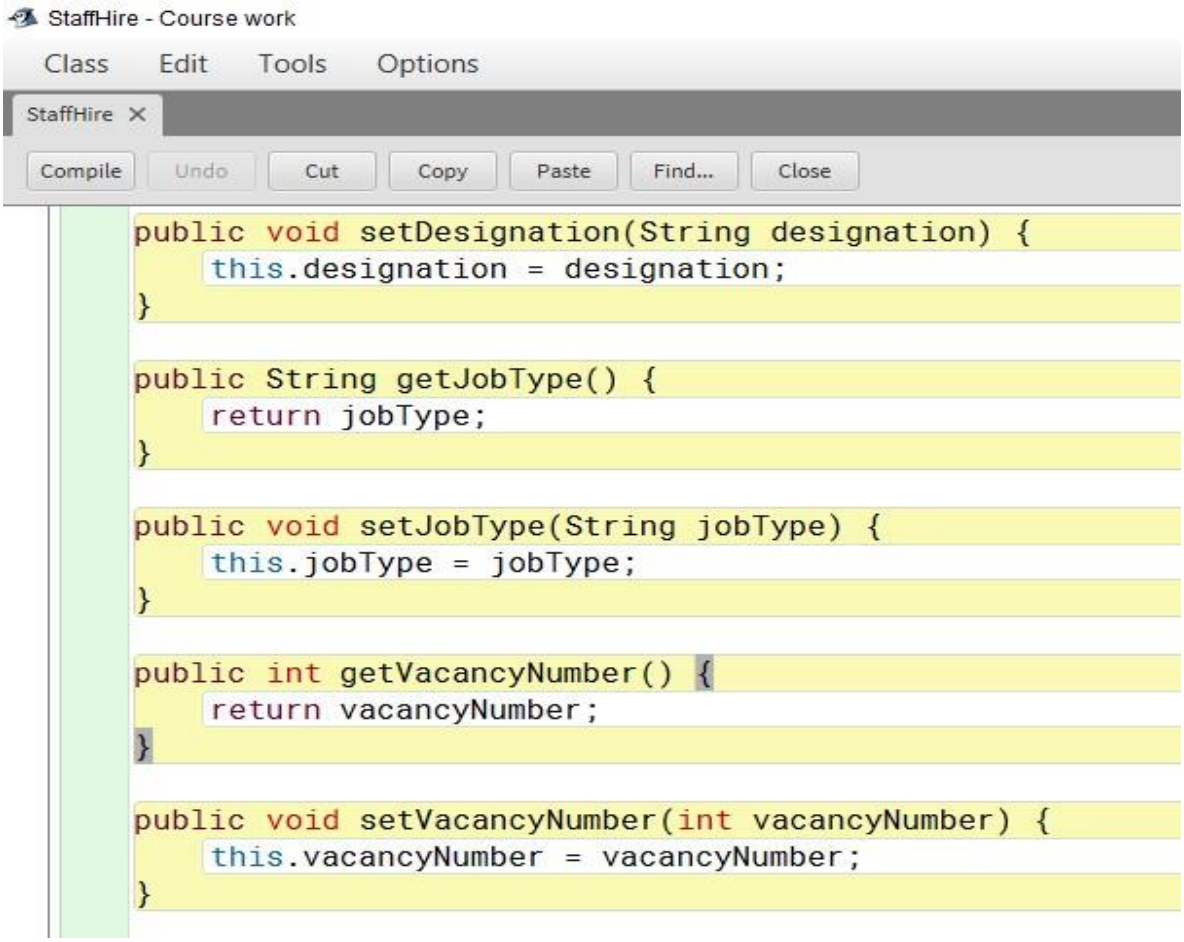

Figure 30: After solving the Syntax error

5.2 Runtime Error

A runtime blunder is a program mistake that happens while the program is running. Runtime blunders are ordinarily called alluded to as "bugs," and are frequently found during the investigating procedure, before the product is discharged. When runtime blunders are found after a program has been conveyed to the general population, engineers regularly discharge fixes, or refreshes and so forth.While trying to execute the program, I find the error message display. I tried to run the error with String in String value. The error is run time error which appears during the running of program. I solved it by adding String as shown in figure.

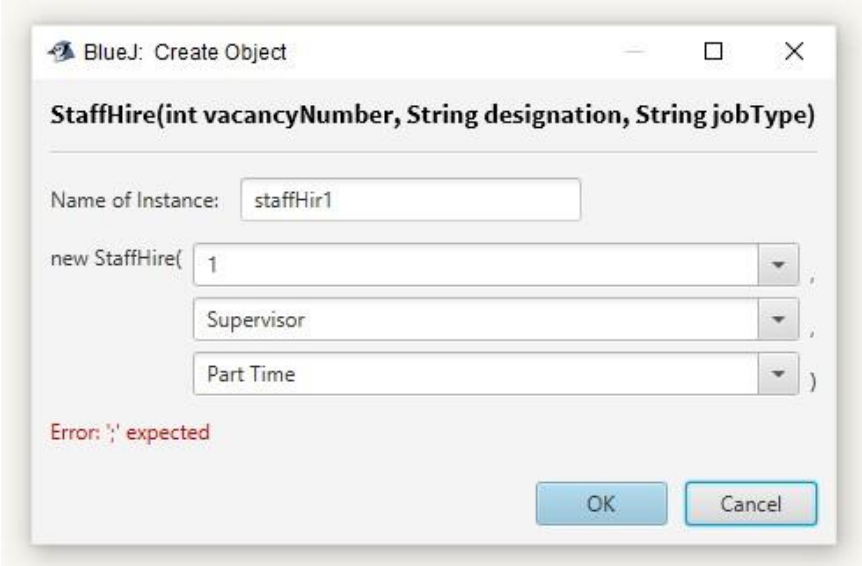

Figure 31: Runtime Error

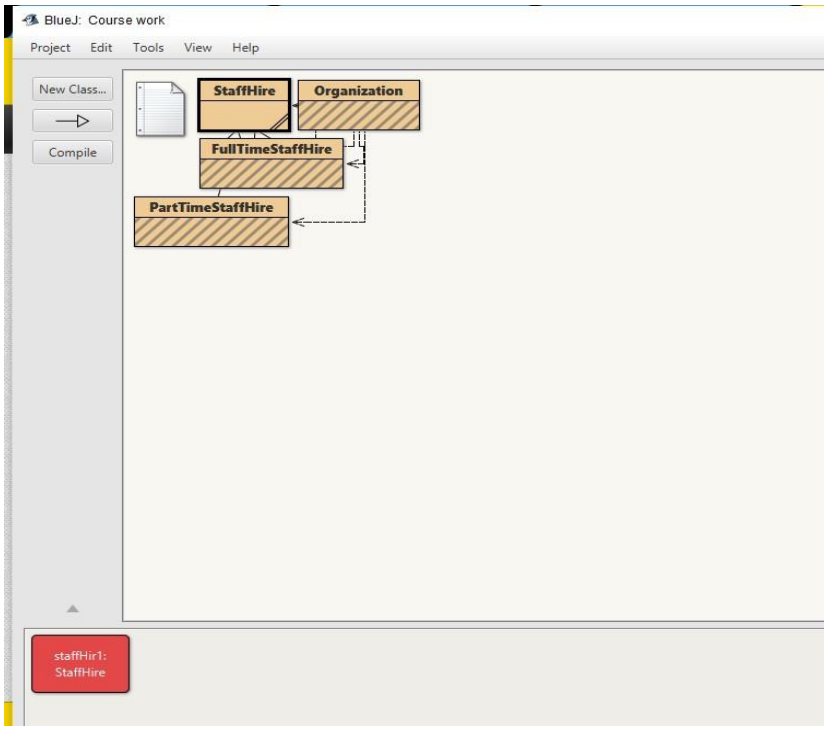

Figure 32: After solving the runtime error

5.3 Logical Error

Logical error in Java programming can be amazingly hard to discover in light of the fact that they don't mirror any kind of coding issue or a mistake in the utilization of java language components. It just won't play out the program that you are expecting as your program to run. The code runs flawlessly as composed. These types of error might be the hardest to discover.In this program I did not get the result as I expected because of the logical error that occurs.

The error is very simple but very hard to find. The error is I used single"=" instead of using "==".Then I finally able to solve my error and get the output I expected.

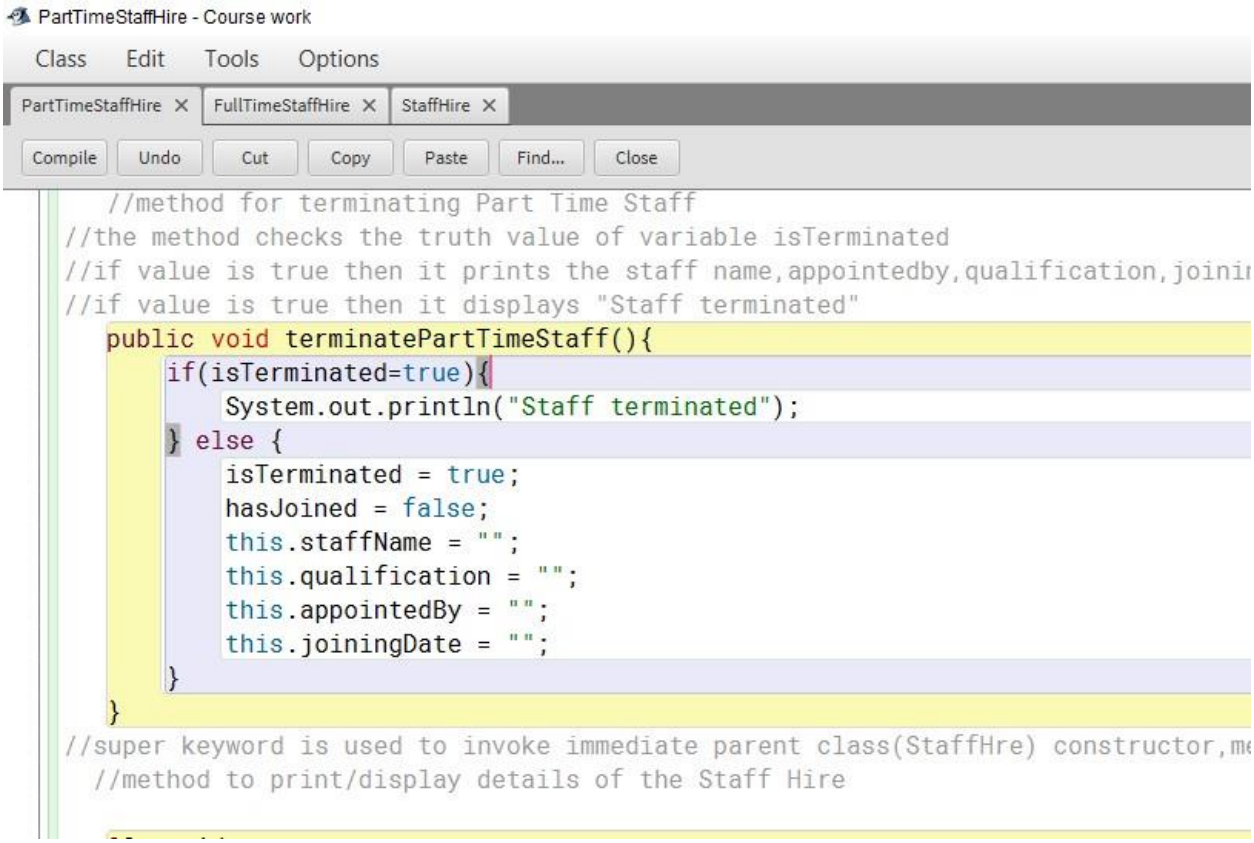

Figure 33: Logical Error

PartTimeStaffHire - Course work

Class Edit Tools Options

PartTimeStaffHire × FullTimeStaffHire × StaffHire ×

Compile Undo Cut Copy Paste Find... Close

```
   //method for terminating Part Time Staff
//the method checks the truth value of variable isTerminated
//if value is true then it prints the staff name,appointedby,qualificat
//if value is true then it displays "Staff terminated"
   public void terminatePartTimeStaff(){
       if(isTerminated==true){
           System.out.println("Staff terminated");
       } else {
           isTerminated = true;
           hasJoined = false;
           this.staffName = "";
           this.qualification = "";
           this.appointedBy = "";
           this.joiningDate = "";
       }
   }
//super keyword is used to invoke immediate parent class(StaffHre) const
  //method to print/display details of the Staff Hire
```

Figure 34: After the error is solved

6. Appendix

```
//Creation of class StaffHire
class StaffHire{

    //variable create
String designation,jobType;
int vacancyNumber;



//creation of constructor that takes paramaters of string, string and integer data type
public StaffHire(int vacancyNumber, String designation, String jobType) {
this.designation = designation;        this.jobType = jobType;        this.vacancyNumber =
vacancyNumber;

  }
//accessor method to get Designation
public String getDesignation() {
return designation;
  }
//setter method to set Designation     public void
setDesignation(String designation) {
this.designation = designation;
  }
//accessor method to get JobType
public String getJobType() {
return jobType;

  }
//setter method to set JobType     public
void setJobType(String jobType) {
this.jobType = jobType;
  }
```

```
//accessor method to get
VacancyNumber          public int
getVacancyNumber() {         return
vacancyNumber;
   }
//setter method to set VacancyNumber     public void
setVacancyNumber(int vacancyNumber) {
this.vacancyNumber = vacancyNumber;
   }
  //method to print/display details of the Staff Hire

   public void displayStaffHire(){

       System.out.println("Designation : "+ designation);
       System.out.println("JobType : "+jobType);
       System.out.println("VacancyNumber : "+vacancyNumber);
   }
}



//creation of new class FullTimeStaffHire that is a sub class of StaffHire class
class FullTimeStaffHire extends StaffHire {     int salary,workingHour;
   String staffName,joiningDate,qualification,appointedBy;
    boolean hasJoined;

//Constructor that takes parameters of String,String,Integer and int
//Constructor hasjoined value set to false



   public FullTimeStaffHire(int vacancyNumber, String designation, String jobType, int
salary, int workingHour) {         super(vacancyNumber, designation, jobType);
this.workingHour = workingHour;         this.salary = salary;
```

```
        staffName="";
qualification = "";
appointedBy = "";
joiningDate = "";
hasJoined = false;
    }

    //asscessor method to get
WorkingHour         public int
getWorkingHour() {         return
workingHour;
    }
    //refer to workingHour of a constructor.
public void setWorkingHour(int workingHour) {
this.workingHour = workingHour;
    }
    //asscessor method to get Salary
public int getSalary() {
        return salary;
    }

    public void setSalary(int salary) {
this.salary = salary;
    }
    //asscessor method to get
StaffName     public String
getStaffName() {         return
staffName;
    }

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

```
    }
    //asscessor method to get Qualification
public String getQualification() {
return qualification;
    }

    public void setQualification(String qualification) {
this.qualification = qualification;
    }
    //asscessor method to get AppointedBy
public String getAppointedBy() {
return appointedBy;
    }

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

     }
    //asscessor method to get
JoiningDate     public String
getJoiningDate() {         return
joiningDate;
    }

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

    public boolean isJoined() {
return hasJoined;
    }
    //asscessor method to get Joined
public void setJoined(boolean joined) {
this.hasJoined = joined;
```

```
    }

    public int salary(int salary){
if (hasJoined==false){
this.workingHour = salary;

}
els
e{
            System.out.println("Staff has been appointed already.");
            System.out.println();
        }
        return salary;
    }
     //the method checks the truth value of variable hasJoined
 //if value is false then it prints the workingHour
 //if value is false then it displays "Staff has been apointed already"
public int workingHour(int workingHour){        if (hasJoined==false){
this.workingHour = workingHour;
        }else {
            System.out.println("Staff has been appointed already.");
            System.out.println();
        }
        return workingHour;
    }
    //method for hiring Full Time Staff that takes parameters of  String and Boolean
 //the method checks the truth value of variable hasJoined
 //if value is false then it prints the staff name,appointedby,qualification,joiningdate
 //if value is false then it displays "Staff has been apointed already"


    public void hireFullTimeStaff(String staffName, String joiningDate, String qualification, String
appointedBy){        if (hasJoined == false){            this.staffName = staffName;
```

```
this.appointedBy = appointedBy;            this.qualification = qualification;
this.joiningDate = joiningDate;            this.hasJoined = true;
        } else {
            System.out.println("Staff has been appointed already.");
            System.out.println();
        }
    }
    //the method checks the truth value of variable hasJoined
 //if value is true then it prints the staff
name,salary,workingHour,appointedby,qualification,joiningdate


    @Override     public void
displayStaffHire(){         if
(hasJoined==true){
super.displayStaffHire();
 //super keyword is used to invoke immediate parent class(StaffHre) constructor,method.
   //method to print/display details of the Staff Hire


        System.out.println("Staff Name: " + staffName);
        System.out.println("Salary: " +salary);
        System.out.println("Working Hour: " + workingHour);
        System.out.println("Appointed By: " + appointedBy);
        System.out.println("Qualification: " + qualification);
        System.out.println("Joining Date: " + joiningDate);
        System.out.println();
        }
    }
}



//creation of new class PartTimeStaffHire that is a sub class of StaffHire class public class
PartTimeStaffHire extends StaffHire {
```

```
   String staffName,joiningDate,qualification,appointedBy,shifts;
    int workingHour,wagesPerHour;
   boolean isTerminated,hasJoined;

   //constructor with six parameters in which three are inherited from StaffHire superclass
public PartTimeStaffHire(int vacancyNumber, String designation, String jobType, int
workingHour, int wagesPerHour, String shifts) {         super(vacancyNumber, designation,
jobType);

       this.wagesPerHour =
wagesPerHour;         this.workingHour =
workingHour;         this.shifts = shifts;
staffName="";         appointedBy = "";
qualification = "";                 joiningDate
= "";         isTerminated = false;
hasJoined = false;

   }

   //accessor methods getter and
setter     public int
getWagesPerHour() {         return
wagesPerHour;
   }

   public void setWagesPerHour(int wagesPerHour) {
this.wagesPerHour = wagesPerHour;
   }
    public int getWorkingHour() {
       return workingHour;
   }

   public void setWorkingHour(int workingHour) {
this.workingHour = workingHour;
```

```
    }



    public String getStaffName() {
return staffName;
    }

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

     public String getAppointedBy() {
return appointedBy;
    }

    public void setAppointedBy(String appointedBy) {
this.appointedBy = appointedBy;
    }
    public String getJoiningDate() {
return joiningDate;
    }
     public String getQualification() {
        return qualification;
    }

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

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

```
    public String getShifts() {
return shifts;
    }

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

    public boolean isHasJoined() {
return hasJoined;
    }

    public void setHasJoined(boolean hasJoined) {
this.hasJoined = hasJoined;
    }
    public boolean isTerminated() {
return isTerminated;
     }

    public void setTerminated(boolean terminated) {
isTerminated = terminated;
    }

    // Setting condition methods:
    public void setNewShifts(String
shifts){        if (hasJoined != true){
this.shifts= shifts;
        } else {
            System.out.println("Staff has been already appointed.");
            System.out.println();
        }
```

```
   }
   //method for hiring Part Time Staff that takes parameters of  String and Boolean
 //the method checks the truth value of variable hasJoined
 //if value is false then it prints the staff name,appointedby,qualification,joiningdate
 //if value is false then it displays "Staff has been apointed already"     public
void hirePartTimeStaff(String staffName, String joiningDate, String qualification,
String appointedBy){        if (hasJoined == false){            this.staffName =
staffName;            this.qualification = qualification;
this.appointedBy = appointedBy;            this.joiningDate = joiningDate;
this.hasJoined = true;
       } else {
           System.out.println("Staff has been already appointed.");            System.out.println();
       }
   }
   //method for terminating Part Time Staff
 //the method checks the truth value of variable isTerminated
 //if value is true then it prints the staff name,appointedby,qualification,joiningdate
 //if value is true then it displays "Staff terminated"
public void terminatePartTimeStaff(){
if(isTerminated==true){
           System.out.println("Staff terminated");
       } else {
isTerminated = true;
hasJoined = false;
this.staffName = "";
this.qualification = "";
this.appointedBy = "";
this.joiningDate = "";
       }
   }
 //super keyword is used to invoke immediate parent class(StaffHre) constructor,method.
  //method to print/display details of the Staff Hire
```

```
    @Override     public void
displayStaffHire(){         if
(hasJoined==true){
super.displayStaffHire();
            System.out.println("Staff Name: " + staffName);
            System.out.println("Qualification: " + qualification);            System.out.println("Wages
per hour: " + wagesPerHour);
            System.out.println("Working Hour: " + workingHour);
            System.out.println("Income per day: "+ (workingHour*wagesPerHour));
            System.out.println("Appointed By: " + appointedBy);
            System.out.println("Joining Date: " + joiningDate);
            System.out.println();
        }
    }
}
```

9. Conclusion:

This coursework is all about java swings, array list, awt and many GUI components. We have to use previous Coursework to do this one. I started doing my coursework by making GUI design, I am little confused about the design of my GUI, so I researched and finally able to found the GUI I wished to make. Making GUI is not so hard for me, it’s really simple but takes a lot of time to build.

While finishing this coursework, I didn’t go through many troubles. Since GUI was easy for me I didn’t went through any trouble but while checking the proper functionality of the button in GUI, it really did not work as it needs to be The data's that are entered are not storing in arraylist. So, for the completion of the program I went through different researches such as articles, journals, books as well as different websites, I asked with my seniors, teachers and friends. So, through different help I finished program in no Time. While, doing this coursework I learned how to make a proper function able GUI, storing data’s in arraylist through GUI and many more things.

Actually, this coursework is really heavy, it contains lots of reports and writings. This coursework made me realize that when you work hard with smart mentality, I can finish as well as run the program as coursework requirement in a less time. I learned many new things from this program like, I learned to use Font, color classes of java that add beauty in my GUI. After running my program, I was successfully able to test the program in command prompt.

After finishing this coursework, I came to know that I wasn’t wasting my time doing only the homework stuffs but also it increased my skills in various way. I really enjoyed creating and designing GUI, it’s my favorite part during the

coursework .So, it was very fun doing this coursework which enhanced my performance very well.

References

Blue J, 2020. [Online]
Available at: https://www.bluej.org/about.html
[Accessed 15 4 2020].

Blue J, 2020. [Online]
Available at: https://www.bluej.org/
[Accessed 24 April 2020].

Computer Hope, 2020. [Online]
Available at: https://www.computerhope.com/jargon/m/microsoft-word.htm
[Accessed 15 4 2020].

Computer Hope, 2020. [Online]
Available at: https://www.computerhope.com/jargon/m/microsoft-word.htm
[Accessed 24 April 2020].

Draw.io, 2020. [Online]
Available at: https://drawio-app.com/zzz/about-us/
[Accessed 15 4 2020].

Microsoft.com, 2020. [Online]
Available at: https://www.microsoft.com/en-us/p/drawio-diagrams/9mvvszk43qqw?activetab=pivot:overviewtab
[Accessed 24 April 2020].