LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CS4001NI - Programming

Assessment Weightage & Type

30% Individual Coursework

Year and Semester

2019-20 Autumn

Student Name: Pratik Amatya

Group: C1

London Met ID: 19031389

College ID: NP01CP4A190024

Assignment Due Date: Friday 05 June 2020

Assignment Submission Date: Friday 05 June 2020

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

Table of Contents

List of Figures

List of Table

1. Introduction

The coursework was assigned to us in the week 20 belonging to the programing module. The first objective was to create the INGNepal class. A GUI application was to be created using AWT and Swing packages of Java. Different methods were used to store each GUI. Multiple GUI approach was decided to be used in the application.

1.1. Description of the project

INGNepal class was added to the previous course work. The essential packages for GUI components of the application were imported. Action Listener was implemented to the class to add functionality to the buttons. Each GUI has separate methods. There are total of four GUI. The main GUI consists of 3 buttons consisting of functionality such as to open part time staff GUI, full time staff GUI and to display all the stored records. The part time staff GUI and full-time staff GUI has sections to add vacancy and appoint staff as well. The part time staff GUI has an additional terminate button for terminating part time staff.

The methods storing GUI consists of components of Swing package such as JFrame, JLabel, JTextField, JButton, JComboBox, JSeparator, JOptionPane and SwingConstants. There are different methods for returning the attributes which stores value entered by the user. For Exception Handling, Try Catch statements are used to catch exceptions which occurs when values entered by the user is converted to suitable data types. JOptionPane is used to display suitable error message.

Each attribute has a Getter method which contains the try catch statements. This makes the code of the program easily understandable and less cluttered. Getter methods are called to access the values entered by the user in GUI.

1.2. Aim

The aim of this project is to add a class i.e. INGNepal to the previous project to make a GUI (Graphical User Interface) which stores the details of added vacancy and appointed staffs' details in an array list.

1.3. Use of the application

The use of this application is to add vacancy, hire staffs whose job type may be part time or full time. It stores the details of the staffs like staff name, joining date, qualification, etc.

1.4. Tools Used

Mainly two applications were used while doing this entire project.

- BlueJ – Text Editor for coding the java program
  It is an IDE for the Java programming language which is used for teaching and learning the concept of object-oriented programming. It was first developed by Michael Kölling and first released in 1999. (BlueJ, 2020)
- Draw.io – For creating the class diagram
  It is a free online diagram editor which is used to create UML, flowcharts, entity relation diagrams, etc.

2. Class Diagram

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modelling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram. (tutorialspoint, 2020)

The class diagram INGNepal class and relationship diagram is given in the next page:

2.1. Class Diagram of ING Nepal

ING Nepal
- mainFrame: JFrame - INGnepalMainFrameLabel: JLabel - mainFrameInfoLabel: JLabel - fullTimeStaffHireButton: JButton - partTimeStaffHireButton: JButton - displayMainFrameButton: JButton - framePartTimeStaffHire: JFrame - titlePartTimeLabel: JLabel - addVacancyPartTimeLabel: JLabel - vacancyNumberPartTimeLabel: JLabel - jobTypePartTimeLabel: JLabel - designationPartTimeLabel: JLabel - wagesPerHourPartTimeLabel: JLabel - workingHoursPerDayPartTimeLabel: JLabel - shiftPartTimeLabel: JLabel - appointPartTimeStaffHireLabel: JLabel - vacancyNumberAppointPartTimeLabel: JLabel - staffNameAppointPartTimeLabel: JLabel

- joiningDateAppointPartTimeLabel: JLabel

- appointedByAppointPartTimeLabel: JLabel

- qualificationAppointPartTimeLabel: JLabel

- savePartTimeButton: JButton

- clearPartTimeButton: JButton

- displayPartTimeButton: JButton

- terminatePartTimeButton: JButton

- appointPartTimeButton: JButton

- jobTypePartTimeComboBox: JComboBox<String>

- shiftPartTimeComboBox: JComboBox<String>

- joiningDateYearAppointPartTimeComboBox: JComboBox<String>

- joiningDateMonthAppointPartTimeComboBox: JComboBox<String>

- joiningDateDayAppointPartTimeComboBox: JComboBox<String>

- vacancyNumberPartTimeTextField: JTextField

- designationPartTimeTextField: JTextField

- wagesPerHourPartTimeTextField: JTextField

- workingHoursPerDayPartTimeTextField: JTextField

- vacancyNumberAppointPartTimeTextField: JTextField
- staffNameAppointPartTimeTextField: JTextField
- qualificationAppointPartTimeTextField: JTextField
- appointedByAppointPartTimeTextField: JTextField
- lineA: JSeperator
- frameTerminate: JFrame
- terminateLabel: JLabel
- terminateVacancyNumberTextField: JTextField
- terminateCancelButton: JButton
- terminateConfirmButton: JButton
- frameFullTimeStaffHire: JFrame
- titleFullTimeLabel: JLabel
- addVacancyFullTimeLabel: JLabel
- vacancyNumberFullTimeLabel: JLabel
- jobTypeFullTimeLabel: JLabel
- designationFullTimeLabel: JLabel
- salaryFullTimeLabel: JLabel
- workingHoursFullTimeLabel: JLabel
- appointFullTimeStaffHire: JLabel

- vacancyNumberAppointFullTimeLabel: JLabel

- staffNameAppointFullTimeLabel: JLabel

- joiningDateAppointFullTimeLabel: JLabel

- qualificationAppointFullTimeLabel: JLabel

- appointedByAppointFullTime: JLabel

- saveFullTimeButton: JButton

- clearFullTimeButton: JButton

- displayFullTimeButton: JButton

- appointFullTime: JButton

- jobTypeFullTimeComboBox: JComboBox<String>

- joiningDateYearAppointFullTimeComboBox: JComboBox<String>

- joiningDateMonthAppointFullTimeComboBox: JComboBox<String>

- joiningDateDayAppointFullTimeComboBox: JComboBox<String>

- vacancyNumberFullTimeTextField: JTextField

- designationFullTimeTextField: JTextField

- salaryFullTimeTextField: JTextField

- workingHoursFullTimeTextField: JTextField

- vacancyNumberAppointFullTimeTextField: JTextField
- staffNameAppointFullTimeTextField: JTextField
- qualificationAppointFullTimeTextField: JTextField
- appointedByAppointFullTimeTextField: JTextField
- lineB: JSeparator
- staffList: ArrayList<StaffHire>

+ INGNepal()
+ mainWIndowGUI(): void
+ PartTimeStaffHireGUI(): void
+ FullTimeStaffHireGUI(): void
+ terminateGUI(): void
+ resettingFieldsPartTime(): void
+ resettingFieldsFullTime(): void
+ getVacancyNumberPartTimeTextFied() :int
+ getDesignationPartTimeTextField(): String
+ getWagesPerHourPartTimeTextField(): int
+ getWorkingHoursPerDayPartTimeTextField(): int
+ getVacancyNumberAppointPartTimeTextField(): int

+ getStaffNameAppointPartTimeTextField(): String

+ getQualificationAppointPartTimeTextField(): String

+ getAppointedByAppointPartTimeTextField(): String

+ getJobTypePartTimeComboBox(): String

+ getShiftPartTimeComboBox(): String

+ getJoiningDateAppointPartTime(): String

+ getVacancyNumberFullTimeTextField(): int

+ getDesignationFullTimeTextField(): String

+ getWorkingHoursFullTimeTextField(): String

+ getSalaryFullTimeTextField(): int

+ getJobTypeFullTimeComboBox(): String

+ getVacancyNumberAppointFullTimeTextField(): int

+ getStaffNameAppointFullTimeTextField(): String

+ getQualificationAppointFullTimeTextField(): String

+ getAppointedByAppointFullTimeTextField(): String

+ getJoiningDateAppointFullTimeTextField(): String

+ getTerminateVacancyNumberTextField(): String

+ actionPerformed(e: ActionEvent): void

<u>+ main(args: String[.]) void</u>

Figure 1: Detailed Class Diagram of ING Nepal Class

2.2. Relationship Diagram

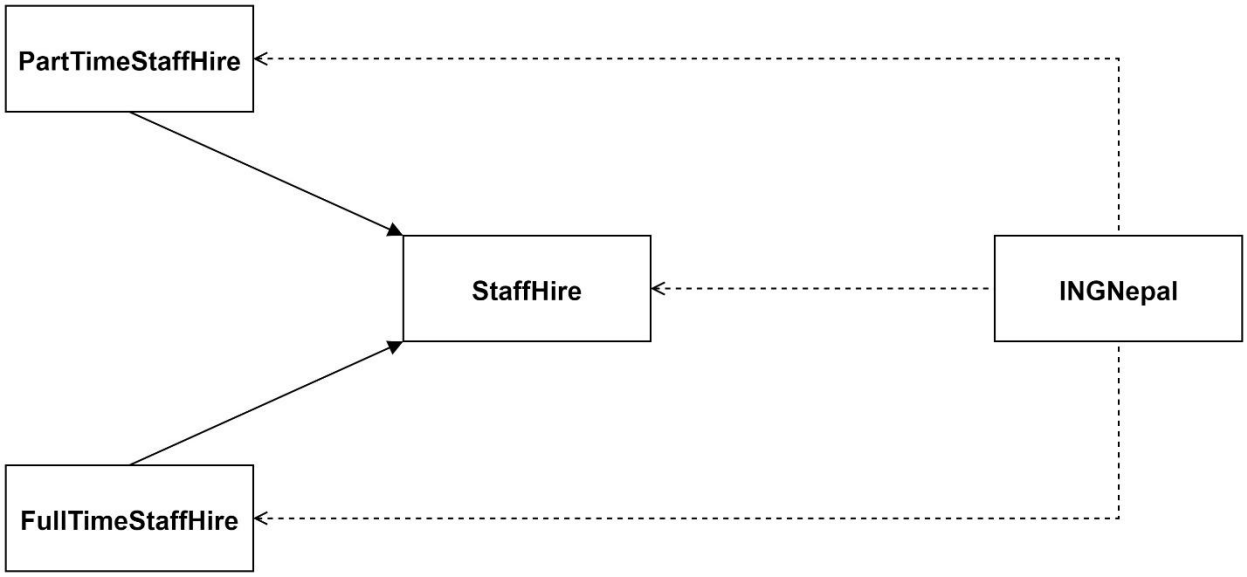

Figure 2: Relationship Diagram

3. Pseudocode

Pseudocode is an informal way of programming description that does not require any strict programming language syntax or underlying technology considerations. It is used for creating an outline or a rough draft of a program. (The Economics Times, 2020)

3.1. Pseudocode of INGNepal

```
DEFINE class INGNepal
    DO
    DEFINE no return type void INGNepal ()
    DO
        CALL mainWindowGUI ()
    END DO

    DEFINE no return type void mainWindowGUI ()
        DO
            CREATE staffList as an empty array list of string
            CREATE mainframe with JFrame having title “ING NEPAL”
            SET SIZE OF mainframe as (615,220)
            SET LAYOUT MANAGER OF mainFrame as FALSE
            SET RESIZABLE OF mainFrame as FALSE

            CREATE INGnepalMainFrameLabel with JLabel as “ING NEPAL”
            SET BOUNDS OF INGnepalMainFrameLabel as (250, 0, 200, 42)
            SET FONT SIZE OF INGnepalMainFrameLabel as 22f

            CREATE mainFrameInfoLabel with JLabel as "Please click one of
            the buttons below :"
            SET BOUNDS OF mainFrameInfoLabel as (200, 50, 400, 30)

            CREATE partTimeStaffHireButton with JButton as "Part Time Staff"
            SET BOUNDS OF partTimeStaffHireButton as (15, 110, 180, 50)
```

```
        ADD ACTION LISTENER TO partTimeStaffHireButton

        CREATE fullTimeStaffHireButton with JButton as "Full Time Staff"
        SET BOUNDS OF fullTimeStaffHireButton as (210, 110, 180, 50)
        ADD ACTION LISTENER TO fullTimeStaffHireButton

        CREATE displayMainFrameButton with JButton as "Display"
        SET BOUNDS OF displayMainFrameButton as (405, 110, 180, 50)
        ADD ACTION LISTENER TO displayMainFrameButton

        ADD INGnepalMainFrameLabel TO mainFrame
        ADD mainFrameInfoLabel TO mainFrame
        ADD partTimeStaffHireButton TO mainFrame
        ADD fullTimeStaffHireButton TO mainFrame
        ADD displayMainFrameButton TO mainframe

        SET mainFrame AS VISIBLE
    END DO
    DEFINE no return type void PartTimeStaffHireGUI ()
    DO
        CREATE framePartTimeStaffHire with JFrame having title “Part
        Time Staff”
        SET SIZE OF framePartTimeStaffHire as (610, 630)
        SET LAYOUT MANAGER OF framePartTimeStaffHire as FALSE
        SET RESIZABLE OF framePartTimeStaffHire as FALSE

        CREATE titlePartTimeLabel with JLabel as "For Part Time Staff"
        SET BOUNDS OF titlePartTimeLabel as (215, 0, 400, 30)
        SET FONT SIZE OF titlePartTimeLabel as 18f
```

CREATE addVacancyPartTimeLabel with JLabel as "Add Vacancy :"
SET BOUNDS OF addVacancyPartTimeLabel as (40, 40, 400, 30)
SET FONT SIZE OF addVacancyPartTimeLabel as 17f

CREATE vacancyNumberPartTimeLabel with JLabel as "Vacancy Number:"
SET BOUNDS OF vacancyNumberPartTimeLabel as (40, 90, 120, 30)

CREATE vacancyNumberPartTimeTextField with JTextField
SET BOUNDS OF vacancyNumberPartTimeTextField as (150, 90, 90, 30)

INITIALIZE jobTypeArray as array having items "Part Time" and "Full Time"
CREATE jobTypePartTimeComboBox with JComboBox having String Type array jobTypeArray
SET BOUNDS OF jobTypePartTimeComboBox as (440, 90, 120, 30)

CREATE designationPartTimeLabel with JLabel as "Designation:"
SET BOUNDS OF designationPartTimeLabel as (40, 140, 75, 30)

CREATE designationPartTimeTextField with JTextField
SET BOUNDS OF designationPartTimeTextField as (120, 140, 120, 30)

CREATE wagesPerHourPartTimeLabel with JLabel as "Wages Per Hour:"

SET BOUNDS OF wagesPerHourPartTimeLabel as

(370, 140, 100, 30)

CREATE wagesPerHourPartTimeTextField with JTextField

SET BOUNDS OF wagesPerHourPartTimeTextField as

(480, 140, 80, 30)

CREATE workingHoursPerDayPartTimeLabel with JLabel as

"Working Hours Per Day:"

SET BOUNDS OF workingHoursPerDayPartTimeLabel as

40, 190, 150, 30)

CREATE workingHoursPerDayPartTimeTextField with JTextField

SET BOUNDS OF workingHoursPerDayPartTimeTextField as
(180, 190, 60, 30)

CREATE shiftPartTimeLabel with JLabel as

"Shift:"

SET BOUNDS OF shiftPartTimeLabel as
(370, 190, 60, 30)

INITIALIZE shiftArray as array having items
"Morning", "Afternoon", "Evening", "Midnight"
CREATE shiftPartTimeComboBox with JComboBox having
String Type array shiftArray

SET BOUNDS OF shiftPartTimeComboBox as

(440, 90, 120, 30)

CREATE savePartTimeButton with JButton as "Save"

SET BOUNDS OF savePartTimeButton as (440, 240, 120, 40)

ADD ACTION LISTENER TO savePartTimeStaffHireButton

CREATE lineA with JSeparator

SET ORIENTATION AS Horizontal

SET BOUNDS OF lineA as (0, 300, 650, 1)

CREATE appointPartTimeStaffHireLabel with JLabel as

"Appoint Part Time Staff Hire :"

SET FONT SIZE OF appointPartTimeStaffHireLabel as 17f

SET BOUNDS OF appointPartTimeStaffHireLabel as
(40, 320, 300, 30)

CREATE vacancyNumberAppointPartTimeLabel with JLabel as

"Vacancy Number:"

SET BOUNDS OF vacancyNumberAppointPartTimeLabel as
(40, 370, 120, 30)

CREATE vacancyNumberAppointPartTimeTextField with
JTextField

SET BOUNDS OF vacancyNumberAppointPartTimeTextField as
(150, 370, 90, 30)

CREATE staffNameAppointPartTimeLabel with JLabel as

"Staff Name:"

SET BOUNDS OF staffNameAppointPartTimeLabel as
(370, 370, 90, 30)

CREATE staffNameAppointPartTimeTextField with JTextField

SET BOUNDS OF staffNameAppointPartTimeTextField as
(445, 370, 115, 30)

CREATE joiningDateAppointPartTimeLabel with JLabel as

"Joining Date:"

SET BOUNDS OF joiningDateAppointPartTimeLabel as
(40, 420, 100, 30)

INITIALIZE joiningDateYearArray as array having items
"YYYY", "2020", "2021", "2022", "2023", "2024" and "2025"
CREATE joiningDateYearAppointPartTimeComboBox with
JComboBox having String Type array joiningDateYearArray

SET BOUNDS OF joiningDateYearAppointPartTimeComboBox as

(125, 420, 60, 30)

INITIALIZE joiningDateMonthArray as array having items
"MM", "Jan", "Feb", "Mar", "Apr", "May", "Jun", "July", "Aug", "Sep",
"Oct", "Nov" and "Dec"

CREATE joiningDateMonthAppointPartTimeComboBox with
JComboBox having String Type array joiningDateMonthArray

SET BOUNDS OF joiningDateMonthAppointPartTimeComboBox
as (190, 420, 52, 30)

INITIALIZE joiningDateDayArray as array having items "DD", "01",
"02", "03", "04", "05", "06", "07", "08", "09", "10", "11", "12", "13",
"14", "15", "16", "17", "18", "19", "20", "21", "22", "23", "24", "25",
"26", "27", "28", "29", "30", "31" and "32"

CREATE joiningDateDayAppointPartTimeComboBox with
JComboBox having String Type array joiningDateDayArray

SET BOUNDS OF jjoiningDateDayAppointPartTimeComboBox as

(247, 420, 46, 30)

CREATE qualificationAppointPartTimeLabel with JLabel as

"Qualification:"

SET BOUNDS OF qualificationAppointPartTimeLabel as
(370, 420, 90, 30)

CREATE qualificationAppointPartTimeTextField with JTextField

SET BOUNDS OF qualificationAppointPartTimeTextField as (450, 420, 110, 30)

CREATE appointedByAppointPartTimeLabel with JLabel as

"Appointed By:"

SET BOUNDS OF appointedByAppointPartTimeLabel as (40, 470, 100, 30)

CREATE appointedByAppointPartTimeTextField with JTextField

SET BOUNDS OF appointedByAppointPartTimeTextField as (150, 470, 90, 30)

CREATE clearPartTimeButton with JButton as "Clear"

SET BOUNDS OF clearPartTimeButton as (40, 530, 126, 40)

ADD ACTION LISTENER TO clearPartTimeButton

CREATE displayPartTimeButton with JButton as "Display"

SET BOUNDS OF displayPartTimeButton as (176, 530, 126, 40)

ADD ACTION LISTENER TO displayPartTimeButton

CREATE terminatePartTimeButton with JButton as "Terminate"

SET BOUNDS OF terminatePartTimeButton as (312, 530, 126, 40)

ADD ACTION LISTENER TO terminatePartTimeButton

CREATE appointPartTimeButton with JButton as "Appoint"

SET BOUNDS OF appointPartTimeButton as (448, 530, 126, 40)

ADD ACTION LISTENER TO appointPartTimeButton

ADD titlePartTimeLabel **TO** framePartTimeStaffHire

ADD addVacancyPartTimeLabel **TO** framePartTimeStaffHire

ADD vacancyNumberPartTimeLabel **TO** framePartTimeStaffHire

ADD vacancyNumberPartTimeTextField **TO** framePartTimeStaffHire

ADD jobTypePartTimeLabel **TO** framePartTimeStaffHire

ADD jobTypePartTimeComboBox **TO** framePartTimeStaffHire

ADD designationPartTimeLabel **TO** framePartTimeStaffHire

ADD designationPartTimeTextField **TO** framePartTimeStaffHire

ADD wagesPerHourPartTimeLabel **TO** framePartTimeStaffHire

ADD wagesPerHourPartTimeTextField **TO** framePartTimeStaffHire

ADD workingHoursPerDayPartTimeLabel **TO** framePartTimeStaffHire

ADD workingHoursPerDayPartTimeTextField **TO** framePartTimeStaffHire

ADD shiftPartTimeLabel **TO** framePartTimeStaffHire

ADD shiftPartTimeComboBox **TO** framePartTimeStaffHire

ADD savePartTimeButton **TO** framePartTimeStaffHire

ADD lineA **TO** framePartTimeStaffHire

ADD appointPartTimeStaffHireLabel **TO** framePartTimeStaffHire

ADD vacancyNumberAppointPartTimeLabel **TO** framePartTimeStaffHire

ADD vacancyNumberAppointPartTimeTextField **TO** framePartTimeStaffHire

ADD staffNameAppointPartTimeLabel **TO** framePartTimeStaffHire

ADD staffNameAppointPartTimeTextField **TO** framePartTimeStaffHire

ADD joiningDateAppointPartTimeLabel **TO** framePartTimeStaffHire

ADD joiningDateYearAppointPartTimeComboBox **TO** framePartTimeStaffHire

ADD joiningDateMonthAppointPartTimeComboBox **TO** framePartTimeStaffHire

```
        ADD joiningDateDayAppointPartTimeComboBox TO
        framePartTimeStaffHire
        ADD qualificationAppointPartTimeLabel TO
        framePartTimeStaffHire
        ADD qualificationAppointPartTimeTextField TO
        framePartTimeStaffHire
        ADD appointedByAppointPartTimeLabel TO
        framePartTimeStaffHire
        ADD appointedByAppointPartTimeTextField TO
        framePartTimeStaffHire
        ADD clearPartTimeButton TO framePartTimeStaffHire
        ADD displayPartTimeButton TO framePartTimeStaffHire
        ADD terminatePartTimeButton TO framePartTimeStaffHire
        ADD appointPartTimeButton TO framePartTimeStaffHire

        SET framePartTimeStaffHire AS VISIBLE
END DO

DEFINE no return type void FullTimeStaffHireGUI ()
DO
        CREATE frameFullTimeStaffHire with JFrame having title “Full
        Time Staff"”
        SET SIZE OF mainframe as (610, 630)
        SET LAYOUT MANAGER OF frameFullTimeStaffHire as FALSE
        SET RESIZABLE OF frameFullTimeStaffHire as FALSE

        CREATE titleFullTimeLabel with JLabel as "For Full Time Staff"
        SET BOUNDS OF titleFullTimeLabel as (215, 0, 400, 30)
        SET FONT SIZE OF titleFullTimeLabel as 18f
```

CREATE addVacancyFullTimeLabel with JLabel as "Add Vacancy :"
SET BOUNDS OF addVacancyFullTimeLabel as (40, 40, 400, 30)
SET FONT SIZE OF addVacancyPartTimeLabel as 17f

CREATE vacancyNumberFullTimeLabel with JLabel as "Vacancy Number:"
SET BOUNDS OF vacancyNumberFullTimeLabel as (40, 90, 120, 30)

CREATE vacancyNumberFullTimeTextField with JTextField
SET BOUNDS OF vacancyNumberFullTimeTextField as (150, 90, 90, 30)

INITIALIZE jobTypeArray as array having items "Part Time" and "Full Time"
CREATE jobTypeFullTimeComboBox with JComboBox having String Type array jobTypeArray
SET BOUNDS OF jobTypeFullTimeComboBox as (440, 90, 120, 30)

CREATE designationFullTimeLabel with JLabel as "Designation:"
SET BOUNDS OF designationFullTimeLabel as (40, 140, 90, 30)

CREATE designationFullTimeTextField with JTextField
SET BOUNDS OF designationFullTimeTextField as (120, 140, 130, 30)

CREATE salaryFullTimeLabel with JLabel as "Salary:"
SET BOUNDS OF salaryFullTimeLabel as (370, 140, 50, 30)

CREATE salaryFullTimeTextField with JTextField

SET BOUNDS OF salaryFullTimeTextField as

(430, 140, 130, 30)

CREATE workingHoursFullTimeLabel with JLabel as

"Working Hours:"

SET BOUNDS OF workingHoursFullTimeLabel as

(40, 190, 120, 30)

CREATE workingHoursFullTimeTextField with JTextField

SET BOUNDS OF workingHoursFullTimeTextField as
(170, 190, 70, 30)

CREATE saveFullTimeButton with JButton as "Save"

SET BOUNDS OF saveFullTimeButton as (440, 240, 120, 40)

ADD ACTION LISTENER TO saveFullTimeButton

CREATE lineB with JSeparator

SET ORIENTATION AS Horizontal

SET BOUNDS OF lineB as (0, 300, 650, 1)

CREATE appointFullTimeStaffHireLabel with JLabel as

"Appoint Full Time Staff Hire :"

SET FONT SIZE OF appointFullTimeStaffHireLabel as 17f

SET BOUNDS OF appointFullTimeStaffHireLabel as
(40, 320, 300, 30)

CREATE vacancyNumberAppointFullTimeLabel with JLabel as

"Vacancy Number:"

SET BOUNDS OF vacancyNumberAppointFullTimeLabel as (40, 370, 120, 30)

CREATE vacancyNumberAppointFullTimeTextField with JTextField

SET BOUNDS OF vacancyNumberAppointFullTimeTextField as (150, 370, 90, 30)

CREATE staffNameAppointFullTimeLabel with JLabel as

"Staff Name:"

SET BOUNDS OF staffNameAppointFullTimeLabel as (370, 370, 90, 30)

CREATE staffNameAppointFullTimeTextField with JTextField

SET BOUNDS OF staffNameAppointFullTimeTextField as (445, 370, 115, 30)

CREATE joiningDateAppointFullTimeLabel with JLabel as

"Joining Date:"

SET BOUNDS OF joiningDateAppointFullTimeLabel as (40, 420, 100, 30)

INITIALIZE joiningDateYearArray as array having items "YYYY", "2020", "2021", "2022", "2023", "2024" and "2025"
CREATE joiningDateYearAppointFullTimeComboBox with JComboBox having String Type array joiningDateYearArray

SET BOUNDS OF joiningDateYearAppointFullTimeComboBox as

(125, 420, 60, 30)

INITIALIZE joiningDateMonthArray as array having items "MM", "Jan", "Feb", "Mar", "Apr", "May", "Jun", "July", "Aug", "Sep", "Oct", "Nov" and "Dec"

CREATE joiningDateMonthAppointFullTimeComboBox with JComboBox having String Type array joiningDateMonthArray

SET BOUNDS OF joiningDateMonthAppointFullTimeComboBox as (190, 420, 52, 30)

INITIALIZE joiningDateDayArray as array having items "DD", "01", "02", "03", "04", "05", "06", "07", "08", "09", "10", "11", "12", "13", "14", "15", "16", "17", "18", "19", "20", "21", "22", "23", "24", "25", "26", "27", "28", "29", "30", "31" and "32"

CREATE joiningDateDayAppointFullTimeComboBox with JComboBox having String Type array joiningDateDayArray

SET BOUNDS OF jjoiningDateDayAppointFullTimeComboBox as

(247, 420, 46, 30)

CREATE qualificationAppointFullTimeLabel with JLabel as

"Qualification:"

SET BOUNDS OF qualificationAppointFullTimeLabel as (370, 420, 90, 30)

CREATE qualificationAppointFullTimeTextField with JTextField

SET BOUNDS OF qualificationAppointFullTimeTextField as (450, 420, 110, 30)

CREATE appointedByAppointFullTimeLabel with JLabel as

"Appointed By:"

SET BOUNDS OF appointedByAppointFullTimeLabel as (40, 470, 90, 30)

CREATE appointedByAppointFullTimeTextField with JTextField

SET BOUNDS OF appointedByAppointFullTimeTextField as (130, 470, 110, 30)

CREATE clearFullTimeButton with JButton as "Clear"
SET BOUNDS OF clearFullTimeButton as (40, 530, 160, 40)
ADD ACTION LISTENER TO clearFullTimeButton

CREATE displayFullTimeButton with JButton as "Display"
SET BOUNDS OF displayFullTimeButton as (220, 530, 160, 40)
ADD ACTION LISTENER TO displayFullTimeButton

CREATE appointFullTimeButton with JButton as "Appoint"
SET BOUNDS OF appointFullTimeButton as (400, 530, 160, 40)
ADD ACTION LISTENER TO appointFullTimeButton

ADD titleFullTimeLabel **TO** frameFullTimeStaffHire
ADD addVacancyFullTimeLabel **TO** frameFullTimeStaffHire
ADD vacancyNumberFullTimeLabel **TO** frameFullTimeStaffHire
ADD vacancyNumberFullTimeTextField **TO** frameFullTimeStaffHire
ADD jobTypeFullTimeLabel **TO** frameFullTimeStaffHire
ADD jobTypeFullTimeComboBox **TO** frameFullTimeStaffHire
ADD designationFullTimeLabel **TO** frameFullTimeStaffHire
ADD designationFullTimeTextField **TO** frameFullTimeStaffHire
ADD salaryFullTimeLabel **TO** frameFullTimeStaffHire
ADD salaryFullTimeTextField **TO** frameFullTimeStaffHire
ADD workingHoursFullTimeLabel **TO** frameFullTimeStaffHire
ADD workingHoursFullTimeTextField **TO** frameFullTimeStaffHire
ADD saveFullTimeButton **TO** frameFullTimeStaffHire
ADD lineB **TO** frameFullTimeStaffHire
ADD appointFullTimeStaffHireLabel **TO** frameFullTimeStaffHire

```
        ADD vacancyNumberAppointFullTimeLabel TO
        frameFullTimeStaffHire
        ADD vacancyNumberAppointFullTimeTextField TO
        frameFullTimeStaffHire
        ADD staffNameAppointFullTimeLabel TO frameFullTimeStaffHire
        ADD staffNameAppointFullTimeTextField TO
        frameFullTimeStaffHire
        ADD joiningDateAppointFullTimeLabel TO frameFullTimeStaffHire
        ADD joiningDateYearAppointFullTimeComboBox TO
        frameFullTimeStaffHire
        ADD joiningDateMonthAppointFullTimeComboBox TO
        frameFullTimeStaffHire
        ADD joiningDateDayAppointFullTimeComboBox TO
        frameFullTimeStaffHire
        ADD qualificationAppointFullTimeLabel TO
        frameFullTimeStaffHire
        ADD qualificationAppointFullTimeTextField TO
        frameFullTimeStaffHire
        ADD appointedByAppointFullTimeLabel TO
        frameFullTimeStaffHire
        ADD appointedByAppointFullTimeTextField TO
        frameFullTimeStaffHire
        ADD clearFullTimeButton TO frameFullTimeStaffHire
        ADD displayFullTimeButton TO frameFullTimeStaffHire
        ADD appointFullTimeButton TO frameFullTimeStaffHire

        SET frameFullTimeStaffHire AS VISIBLE
    END DO
    DEFINE no return type void terminateGUI ()
    DO
```

CREATE frameTerminate with JFrame having title “Terminate Part Time Staff”

SET SIZE OF mainframe as (530, 230)

SET LAYOUT MANAGER OF frameTerminate as **FALSE**

SET RESIZABLE OF frameTerminate as **FALSE**

CREATE terminateLabel with JLabel as "Enter the vacancy number of the part time staff you want to terminate :"

SET BOUNDS OF terminateLabel as (15, 0, 600, 50)

SET FONT SIZE OF terminateLabel as 14f

CREATE terminateVacancyNumberTextField with JTextField

SET BOUNDS OF terminateVacancyNumberTextField as (15, 60, 480, 40)

CREATE terminateConfirmButton with JButton as "Terminate"

SET BOUNDS OF terminateConfirmButton as (272, 120, 200, 50)

ADD ACTION LISTENER TO terminateConfirmButton

CREATE terminateCancelButton with JButton as "Cancel"

SET BOUNDS OF terminateCancelButton as (57,120,200,50)

ADD ACTION LISTENER TO terminateCancelButton

ADD terminateLabel **TO** frameFullTimeStaffHire

ADD terminateVacancyNumberTextField **TO** frameFullTimeStaffHire

ADD terminateConfirmButton **TO** frameFullTimeStaffHire

ADD terminateCancelButton **TO** frameFullTimeStaffHire

SET frameTerminate **AS VISIBLE**

END DO

```
DEFINE no return type void resettingFieldsPartTime()
DO
    SET vacancyNumberPartTimeTextField AS ""
    SET designationPartTimeTextField AS “”
    SET wagesPerHourPartTimeTextField AS “”
    SET workingHoursPerDayPartTimeTextField AS “”
    SET vacancyNumberAppointPartTimeTextField AS “”
    SET staffNameAppointPartTimeTextField AS “”
    SET qualificationAppointPartTimeTextField AS “”
    SET appointedByAppointPartTimeTextField AS “”

    SET jobTypePartTimeComboBox AS ITEM HAVING INDEX 0
    SET shiftPartTimeComboBox AS ITEM HAVING INDEX 0
    SET joiningDateYearAppointPartTimeComboBox AS ITEM
    HAVING INDEX 0
    SET joiningDateMonthAppointPartTimeComboBox AS ITEM
    HAVING INDEX 0
    SET joiningDateDayAppointPartTimeComboBox AS ITEM
    HAVING INDEX 0
END DO
DEFINE no return type void resettingFieldsFullTime()
DO
    SET vacancyNumberFullTimeTextField AS ""
    SET designationFullTimeTextField AS “”
    SET workingHoursFullTimeTextField AS “”
    SET salaryFullTimeTextField AS “”
    SET vacancyNumberAppointFullTimeTextField AS “”
    SET staffNameAppointFullTimeTextField AS “”
    SET qualificationAppointFullTimeTextField AS “”
    SET appointedByAppointFullTimeTextField AS “”
```

```
        SET jobTypeFullTimeComboBox AS ITEM HAVING INDEX 0
        SET joiningDateYearAppointFullTimeComboBox AS ITEM
        HAVING INDEX 0
        SET joiningDateMonthAppointFullTimeComboBox AS ITEM
        HAVING INDEX 0
        SET joiningDateDayAppointFullTimeComboBox AS ITEM
        HAVING INDEX 0
END DO
DEFINE int returntype getVacancyNumberPartTimeTextField()
DO
        INTIALIZE vacancyNumPartTime as Int Type AS 0
        IF vacancyNumberPartTimeTextField.getText() EQUALS “”:
                DISPLAY ErrorMessage
        ELSE:
                TRY
                        vacancyNumPartTime =
                        vacancyNumberPartTimeTextField.getText() TO
                        INT;
                         IF (vacancyNumPartTime == 0):
                                DISPLAY ErrorMessage
                        ELSE IF (vacancyNumPartTime<0):
                                DISPLAY ErrorMessage
                CATCH (NumberFormatException e):
                        DISPLAY ErrorMessage
        RETURN vacancyNumPartTime
END DO
DEFINE String returntype getDesignationPartTimeTextField()
DO
        INTIALIZE designationPartTime as String Type AS
        designationPartTimeTextField.getText()
        IF designationPartTimeTextField.getText() EQUALS “”:
```

```
            DISPLAY ErrorMessage
        RETURN designationPartTime
    END DO
    DEFINE int returntype getWagesPerHourPartTimeTextField()
    DO
        INTIALIZE wagesPerHourPartTime as int Type AS 0
        IF wagesPerHourPartTimeTextField.getText() EQUALS “”:
            DISPLAY ErrorMessage
        ELSE:
            TRY
                wagesPerHourPartTime =
                wagesPerHourPartTimeTextField.getText() TO
                INT;
                 IF (wagesPerHourPartTime == 0):
                    DISPLAY ErrorMessage
                ELSE IF (wagesPerHourPartTime<0):
                    DISPLAY ErrorMessage
            CATCH (NumberFormatException e):
                DISPLAY ErrorMessage
        RETURN wagesPerHourPartTime
    END DO
    DEFINE int returntype getWorkingHoursPerDayPartTimeTextField()
    DO
        INTIALIZE workingHoursPerDayPartTime as int Type AS 0
        IF workingHoursPerDayPartTimeTextField.getText() EQUALS “”:
            DISPLAY ErrorMessage
        ELSE:
            TRY
                workingHoursPerDayPartTime =
                workingHoursPerDayPartTimeTextField.getText() TO
                INT;
```

```
                    IF (workingHoursPerDayPartTime == 0):
                        DISPLAY ErrorMessage
                    ELSE IF (workingHoursPerDayPartTime <0):
                        DISPLAY ErrorMessage
            CATCH (NumberFormatException e):
                    DISPLAY ErrorMessage
        RETURN workingHoursPerDayPartTime
END DO
DEFINE int returntype getVacancyNumberAppointPartTimeTextField()
DO
        INTIALIZE vacancyNumAppointPartTime as Int Type AS 0
        IF vacancyNumberAppointPartTimeTextField.getText() EQUALS “”:
            DISPLAY ErrorMessage
        ELSE:
            TRY
                vacancyNumAppointPartTime =
                vacancyNumberAppointPartTimeTextField.getText()
                TO INT;
                IF (vacancyNumAppointPartTime == 0):
                    DISPLAY ErrorMessage
                ELSE IF (vacancyNumAppointPartTime <0):
                    DISPLAY ErrorMessage
            CATCH (NumberFormatException e):
                DISPLAY ErrorMessage
        RETURN vacancyNumAppointPartTime
END DO
DEFINE String returntype getStaffNameAppointPartTimeTextField()
DO
        INTIALIZE StaffNamePartTime as String Type AS
        staffNameAppointPartTimeTextField.getText()
        IF StaffNamePartTime EQUALS “”:
```

```
            DISPLAY ErrorMessage
        RETURN StaffNamePartTime
    END DO
    DEFINE String returntype getQualificationAppointPartTimeTextField()
    DO
        INTIALIZE QualicationPartTime as String Type AS
        qualificationAppointPartTimeTextField().getText()
        IF QualicationPartTime EQUALS “”:
            DISPLAY ErrorMessage
        RETURN QualicationPartTime
    END DO
    DEFINE String returntype getAppointedByAppointPartTimeTextField()
    DO
        INTIALIZE appointedByPartTime as String Type AS
        appointedByAppointPartTimeTextField().getText()
        IF appointedByAppointPartTimeTextField().getText() EQUALS “”:
            DISPLAY ErrorMessage
        RETURN appointedByPartTime
    END DO
    DEFINE String returntype getJobTypePartTimeComboBox()
    DO
        INTIALIZE jobTypePartTime as String Type AS
        jobTypePartTimeComboBox().getSelectedItem()
        IF jobTypePartTime EQUALS “Full Time”:
            DISPLAY ErrorMessage
        RETURN jobTypePartTime
    END DO
    DEFINE String returntype getShiftPartTimeComboBox()
    DO
        INTIALIZE shiftPartTime as String Type AS
        shiftPartTimeComboBox().getSelectedItem()
```

```
        RETURN shiftPartTime
END DO
DEFINE String returntype getJoiningDateAppointPartTime()
DO
        INTIALIZE joiningDate as String Type AS “”
        INTIALIZE joiningDateYearAppointPartTime as String Type AS
        joiningDateYearAppointPartTimeComboBox().getSelectedItem()
        INTIALIZE joiningDateMonthAppointPartTime as String Type AS
        joiningDateMonthAppointPartTimeComboBox().getSelectedItem()
        INTIALIZE joiningDateDayAppointPartTime as String Type AS
        joiningDateDayAppointPartTimeComboBox().getSelectedItem()

        IF joiningDateYearAppointPartTime = “YYYY” OR
        joiningDateMonthAppointPartTime=”MM”
        OR joiningDateDayAppointPartTime= “DD”:
                DISPLAY ErrorMessage
        ELSE:
                joiningDate = joiningDateYearAppointPartTime +
                joiningDateMonthAppointPartTime +
                joiningDateDayAppointPartTime
        RETURN joiningDate
END DO

DEFINE int returntype getVacancyNumberFullTimeTextField()
DO
        INTIALIZE vacancyNumFullTime as Int Type AS 0
        IF vacancyNumberFullTimeTextField.getText() EQUALS “”:
                DISPLAY ErrorMessage
        ELSE:
                TRY
```

```
                    vacancyNumFullTime =
                    vacancyNumberFullTimeTextField.getText() TO
                    INT;
                     IF (vacancyNumFullTime == 0):
                            DISPLAY ErrorMessage
                    ELSE IF (vacancyNumFullTime <0):
                            DISPLAY ErrorMessage
            CATCH (NumberFormatException e):
                    DISPLAY ErrorMessage
        RETURN vacancyNumFullTime
END DO
DEFINE String returntype getDesignationFullTimeTextField()
DO
        INTIALIZE designationFullTime as String Type AS
        designationFullTimeTextField.getText()
        IF designationFullTimeTextField.getText() EQUALS “”:
                DISPLAY ErrorMessage
        RETURN designationFullTime
END DO
                DEFINE int returntype
                getWorkingHoursFullTimeTextField ()
DO
        INTIALIZE workingHoursFullTime as Float Type AS 0.0f
        IF workingHoursFullTimeTextField.getText() EQUALS “”:
                DISPLAY ErrorMessage
        ELSE:
                TRY
                        workingHoursFullTime =
                                workingHoursFullTimeTextField.getText() TO
                                INT;
                         IF (workingHoursFullTime == 0):
```

```
                    DISPLAY ErrorMessage
                ELSE IF (workingHoursFullTime<0):
                    DISPLAY ErrorMessage
            CATCH (NumberFormatException e):
                ELSE IF (workingHoursFullTime<0):
                    DISPLAY ErrorMessage
        RETURN workingHoursFullTime
END DO
DEFINE int returntype getSalaryFullTimeTextField ()
DO
        INTIALIZE salaryFullTime as Float Type AS 0.0f
        IF salaryFullTimeTextField.getText() EQUALS “”:
            DISPLAY ErrorMessage
        ELSE:
            TRY
                salaryFullTime =
                salaryFullTimeTextField.getText() TO FLOAT;
                 IF (salaryFullTime == 0):
                    DISPLAY ErrorMessage
                ELSE IF (salaryFullTime <0):
                    DISPLAY ErrorMessage
            CATCH (NumberFormatException e):
                DISPLAY ErrorMessage
        RETURN salaryFullTime
END DO
DEFINE String returntype getJobTypeFullTimeComboBox()
DO
        INTIALIZE jobTypeFullTime as String Type AS
        jobTypeFullTimeComboBox().getSelectedItem()
        IF jobTypeFullTime EQUALS “Part Time”:
            DISPLAY ErrorMessage
        RETURN jobTypeFullTime
```

```
END DO

DEFINE int returntype getVacancyNumberAppointFullTimeTextField()
DO
    INTIALIZE vacancyNumAppointFullTime as Int Type AS 0
    IF vacancyNumberAppointFullTimeTextField.getText() EQUALS “”:
        DISPLAY ErrorMessage
    ELSE:
        TRY
            vacancyNumAppointFullTime =
            vacancyNumberAppointFullTimeTextField.getText()
            TO INT;
            IF (vacancyNumAppointFullTime == 0):
                DISPLAY ErrorMessage
            ELSE IF (vacancyNumAppointFullTime <0):
                DISPLAY ErrorMessage
        CATCH (NumberFormatException e):
            DISPLAY ErrorMessage
    RETURN vacancyNumAppointFullTime
END DO
DEFINE String returntype getStaffNameAppointFullTimeTextField()
DO
    INTIALIZE StaffNameFullTime as String Type AS
    staffNameAppointFullTimeTextField.getText()
    IF StaffNameFullTime EQUALS “”:
        DISPLAY ErrorMessage
    RETURN StaffNameFullTime
END DO
DEFINE String returntype getQualificationAppointFullTimeTextField()
DO
```

```
        INTIALIZE QualicationFullTime as String Type AS
        qualificationAppointFullTimeTextField().getText()
        IF QualicationFullTime EQUALS “”:
                DISPLAY ErrorMessage
        RETURN QualicationFullTime
END DO
DEFINE String returntype getAppointedByAppointFullTimeTextField()
DO
        INTIALIZE AppointedByFullTime as String Type AS
        appointedByAppointFullTimeTextField().getText()
        IF AppointedByFullTime.getText() EQUALS “”:
                DISPLAY ErrorMessage
        RETURN AppointedByFullTime
END DO
DEFINE String returntype getJoiningDateAppointFullTime()
DO
        INTIALIZE joiningDate as String Type AS “”

        INTIALIZE joiningDateYearAppointFullTime as String Type AS
        joiningDateYearAppointFullTimeComboBox().getSelectedItem()
        INTIALIZE joiningDateMonthAppointFullTime as String Type AS
        joiningDateMonthAppointFullTimeComboBox().getSelectedItem()
        INTIALIZE joiningDateDayAppointFullTime as String Type AS
        joiningDateDayAppointFullTimeComboBox().getSelectedItem()

        IF joiningDateYearAppointFullTime = “YYYY” OR
        joiningDateMonthAppointFullTime=”MM”
        OR joiningDateDayAppointFullTime= “DD”:
                DISPLAY ErrorMessage
        ELSE:
```

```
                joiningDate = joiningDateYearAppointFullTime +
                joiningDateMonthAppointFullTime +
                joiningDateDayAppointFullTime
        RETURN joiningDate
END DO
DEFINE int returntype getTerminateVacancyNumberTextField()
DO
        INTIALIZE terminateVacancyNumberPartTime as Int Type AS 0
        IF terminateVacancyNumberTextField.getText() EQUALS “”:
                DISPLAY ErrorMessage
        ELSE:
                TRY
                        terminateVacancyNumberPartTime =
                        terminateVacancyNumberTextField.getText()
                        TO INT;
                         IF (terminateVacancyNumberPartTime== 0):
                                DISPLAY ErrorMessage
                        ELSE IF (terminateVacancyNumberPartTime <0):
                                DISPLAY ErrorMessage
                CATCH (NumberFormatException e):
                        DISPLAY ErrorMessage
        RETURN terminateVacancyNumberPartTime
END DO

DEFINE no returntype actionPerformed(ActionEvent e)
DO
        IF Button clicked = clearPartTimeButton:
                CALL resettingFieldsPartTime ()

        IF Button clicked = clearFullTimeButton:
                CALL resettingFieldsFullTime()
```

```
IF Button clicked = partTimeStaffHireButton:
        CALL PartTimeStaffHireGUI()

IF Button clicked = fullTimeStaffHireButton:
        CALL FullTimeStaffHireGUI()

IF Button clicked = terminatePartTimeButton:
        CALL terminateGUI()

IF Button clicked = terminateCancelButton:
        CLOSE frameTerminate

IF Button clicked = displayMainFrameButton:
        INTIALIZE recordFound as Boolean Type AS false
         FOR (StaffHire variable:staffList):
                IF variable is INSTANCE OF PartTimeStaffHire:
                        PartTimeStaffHire object =
                        (PartTimeStaffHire) variable
                        CALL object.display();
                        SET recordFound AS true;
                 ELSE:
                        FullTimeStaffHire object =
                        (FullTimeStaffHire)variable
                        CALL object.display();
                        SET recordFound AS true;
        END FOR
        IF recordFound = false:
                DISPLAY ErrorMessage
```

```
IF Button clicked = displayPartTimeButton:
        INTIALIZE recordFound as Boolean Type AS false
         FOR (StaffHire variable:staffList):
                IF variable is INSTANCE OF PartTimeStaffHire:
                        PartTimeStaffHire object =
                        (PartTimeStaffHire) variable
                        CALL object.display();
                        SET recordFound AS true;
        END FOR
        IF recordFound = false:
                DISPLAY ErrorMessage

IF Button clicked = displayFullTimeButton:
        INTIALIZE recordFound as Boolean Type AS false
         FOR (StaffHire variable:staffList):
                IF variable is INSTANCE OF FullTimeStaffHire:
                        FullTimeStaffHire object =
                        (FullTimeStaffHire) variable
                        CALL object.display();
                        SET recordFound AS true;
        END FOR
        IF recordFound = false:
                DISPLAY ErrorMessage

IF Button clicked = terminateConfirmButton:
        INTIALIZE terminateVacancyNumberpt as int Type AS
        getTerminateVacancyNumberTextField()
        INTIALIZE recordFound as Boolean Type AS false
        IF terminateVacancyNumberpt IS NOT 0:
                FOR (StaffHire variable:staffList):
```

```
IF variable.getVacancyNumber() =
terminateVacancyNumberpt:
        SET vacancyNumFound AS true
        IF variable is INSTANCE OF
        PartTimeStaffHire:
                PartTimeStaffHire object =
                (PartTimeStaffHire) variable
                IF object.getTerminated () = FALSE
                AND object.getJoined () = TRUE:
                        CALL object.terminate ();
                        DISPLAY Message (“The staff
                        has been Terminated”)
                        BREAK
                ELSE IF object.getTerminated () =
                TRUE AND object.getJoined () =
                FALSE:
                        DISPLAY Message (“The staff
                        has already been Terminated”)
                        BREAK
                ELSE:
                        DISPLAY Message (“The Staff
                        hasn't been hired. Hence, there is
                        no staff to terminate.”)
                        BREAK
        ELSE:
        DISPLAY Message (“The vacancy number is
        not for Full Time Staff Hire”)
        BREAK
END FOR
IF vacancyNumFound = false:
        DISPLAY ErrorMessage
```

```
IF Button clicked = savePartTimeButton:
        INTIALIZE vacancyNumberpt as int Type AS
        getVacancyNumberPartTimeTextField()
        INTIALIZE designationpt as String Type AS
        getDesignationPartTimeTextField()
        INTIALIZE wagesPerHourpt as int Type AS
        getWagesPerHourPartTimeTextField()
        INTIALIZE workingHoursPerDaypt as int Type AS
        getWorkingHoursPerDayPartTimeTextField()
        INTIALIZE jobTypept as String Type AS
        getJobTypePartTimeComboBox()
        INTIALIZE shiftpt as String Type AS
        getShiftPartTimeComboBox()

        INTIALIZE duplicateVacancyNum as boolean Type AS
        false

        IF vacancyNumberpt IS GREATER THAN 0 AND
        designationpt IS NOT "" AND wagesPerHourpt S GREATER
        THAN 0 AND workingHoursPerDaypt S GREATER THAN
        AND jobTypept IS "Part Time":
        FOR (StaffHire variable:staffList):
                IF variable.getVacancyNumber() = vacancyNumberpt:
                        SET duplicateVacancyNum = true
                        BREAK
        END FOR
        IF duplicateVacancyNum IS false:
                PartTimeStaffHire obj= new
                PartTimeStaffHire(designationpt, jobTypept,
```

```
                vacancyNumberpt, workingHoursPerDaypt,
                wagesPerHourpt,shiftpt)
                staffList.add(obj)

                Display Message "Vacancy has been added"
            ELSE:
                Display Message "The entered vacancy number is
                already in the list"

        IF Button clicked = appointPartTimeButton:
            INTIALIZE vacancyNumberAppointpt as int Type AS
            getVacancyNumberAppointPartTimeTextField()
            INTIALIZE staffNamept as String Type AS
            getStaffNameAppointPartTimeTextField()
            INTIALIZE qualificationpt as String Type AS
            getQualificationAppointPartTimeTextField()
            INTIALIZE appointedBypt as String Type AS
            getAppointedByAppointPartTimeTextField()
            INTIALIZE joiningDatept as String Type AS
            getJoiningDateAppointPartTimeComboBox()

            INTIALIZE vacancyNumFound as boolean Type AS
            false


            IF vacancyNumberAppointpt IS GREATER THAN 0 AND
            staffNamept IS NOT “” AND qualificationpt IS NOT “” AND
            appointedBypt IS NOT "" AND joiningDatept IS NOT “”:
                FOR (StaffHire variable:staffList):
                    IF variable.getVacancyNumber() =
                    vacancyNumberAppointpt:
```

```
                    SET vacancyNumFound = true
                    IF variable IS INSTANCE OF
                    PartTimeStaffHire:
                        PartTimeStaffHire object=
                        (PartTimeStaffHire) variable
                        IF object.getJoined()=false:
                            object.hiring (staffNamept,
                            joiningDatept,
                            qualificationpt,
                            appointedBypt)
                            Display Message (“Staff
                            Hired”)
                            BREAK
                        ELSE IF object.getJoined()=true:
                            Display Message "A staff
                            has already been hired to
                            fill this vacancy no."
                    ELSE:
                        Display Message “Vacancy No.
                        Not for Part Time”
                END FOR
                IF vacancyNumFound=false:
                    Display Message “The inserted
                    vacancy number is invalid”

IF Button clicked = saveFullTimeButton:
    INTIALIZE vacancyNumberft as int Type AS
    getVacancyNumberFullTimeTextField()
    INTIALIZE designationft as String Type AS
    getDesignationFullTimeTextField()
```

```
INTIALIZE jobTypeft as String Type AS
getJobTypeFullTimeComboBox()
INTIALIZE salaryft as int Type AS
getSalaryFullTimeTextField()
INTIALIZE workingHoursft as int Type AS
getWorkingHoursFullTimeTextField()

INTIALIZE duplicateVacancyNum as boolean Type AS
false

IF vacancyNumberft IS GREATER THAN 0 AND salaryft IS
GREATER THAN 0 AND workingHoursft IS GREATER
THAN 0 AND jobTypeft IS "Full Time" AND designationft IS
NOT “”:
    FOR (StaffHire variable:staffList):
        IF variable.getVacancyNumber() =
        vacancyNumberft:
            SET duplicateVacancyNum = true
            BREAK
    END FOR
    IF duplicateVacancyNum IS false:
        FullTimeStaffHire obj= new
        FullTimeStaffHire(designationft, jobTypeft,
        vacancyNumberft, salaryft,
        workingHoursft)

        staffList.add(obj)

        Display Message "Vacancy has been added"
    ELSE:
```

```
                    Display Message "The entered vacancy
                    number is already in the list"

IF Button clicked = appointFullTimeButton:
    INTIALIZE vacancyNumberAppointft as int Type AS
    getVacancyNumberAppointFullTimeTextField()
    INTIALIZE staffNameft as String Type AS
    getStaffNameAppointFullTimeTextField()
    INTIALIZE qualificationft as float Type AS
    getQualificationAppointFullTimeTextField()
    INTIALIZE appointedByft as String Type AS
    getAppointedByAppointFullTimeTextField()
    INTIALIZE joiningDateft as String Type AS
    getJoiningDateAppointFullTimeComboBox()

    INTIALIZE vacancyNumFound as boolean Type AS
    false

    IF vacancyNumberAppointft IS GREATER THAN 0 AND
    staffNameft IS NOT “” AND qualificationft IS NOT “” AND
    appointedByft IS NOT "" AND joiningDateft IS NOT “”:
        FOR (StaffHire variable:staffList):
            IF variable.getVacancyNumber() =
            vacancyNumberAppointft:
                SET vacancyNumFound = true
                IF variable IS INSTANCE OF
                FullTimeStaffHire:
                    FullTimeStaffHire object=
                    (FullTimeStaffHire) variable
                    IF object.getJoined()=false:
```

```
                                        object.hiring (staffNameft,
                                        joiningDateft,
                                        qualificationft,
                                        appointedByft)
                                        Display Message (“Staff
                                        Hired”)
                                        BREAK
                                    ELSE IF object.getJoined()=false:
                                        Display Message "A staff
                                        has already been hired to
                                        fill this vacancy no."
                                ELSE:
                                    Display Message “Vacancy No.
                                    Not for Full Time”
                        END FOR
                    IF vacancyNumFound=false:
                        Display Message “The inserted
                        vacancy number is invalid”
            END DO

            DEFINE Static no returntype main(String [ ] args)
            DO
                CALL INGNepal ()
            END DO
```

4. Method Description

Method Description describes the purpose and functions of each method in the project. The method description of ING Nepal class are listed below:

4.1. Method Description of ING Nepal

S.No.	Method	Description
1	INGNepal()	• Calls the mainWindowGUI() method
2	mainWindowGUI()	• Method contains the Main GUI consisting of classes of java Swing such as Jlabel, JButton, JFrame and action listener.
3	PartTimeStaffHireGUI()	• Method contains the Part Time Staff GUI consisting of classes of java Swing such as Jlabel, JTextField, JButton, JFrame, JComboBox, JSeparator and action listener. • It allows the user to enter the values like vacancy number, wages per hour, etc.
4	FullTimeStaffHireGUI()	• Method contains the Full Time Staff GUI consisting of classes of java Swing such as Jlabel, JTextField, JButton, JFrame, JComboBox, JSeparator and action listener. • It allows the user to enter the values like vacancy number, salary, etc.
5	terminateGUI()	• Method contains the Terminate GUI consisting of classes of java Swing such as Jlabel, JTextField, JButton, JFrame and action listener. • It allows the user to enter the the vacancy number of the staff from the user that they want to terminate.

6	resettingFieldsPartTime()	• Mutator method that sets the default selection of JCombox and clears the text fields of the Part Time GUI.
7	resettingFieldsFullTime()	• Mutator method that sets the default selection of JCombox and clears the text fields in the Full Time GUI.
8	getVacancyNumberPartTime TextField()	• Accessor method that returns the value of vacancyNumberPartTimeTextField as Int type. • It contains try catch statement and shows appropriate message when unsuitable value is entered in the vacancyNumberPartTimeTextField.
9	getDesignationPartTimeTextF ield()	• Accessor method that returns the value of designationPartTimeTextField as String type. • It contains if statement which shows appropriate message when no value is entered in the designationPartTimeTextField.
10	getWagesPerHourPartTimeT extField()	• Accessor method that returns the value of wagesPerHourPartTimeTextField as int type. • It contains try catch statement and shows appropriate message when unsuitable value is entered in the wagesPerHourPartTimeTextField.
11	getVacancyNumberAppointP artTimeTextField()	• Accessor method that returns the value of vacancyNumberAppointPartTimeTextField as Int type. • It contains try catch statement and shows appropriate message when unsuitable value

		is entered in the vacancyNumberAppointPartTimeTextField.
12	getStaffNameAppointPartTimeTextField()	• Accessor method that returns the value of staffNameAppointPartTimeTextField as String type. • It contains if statement which shows appropriate message when no value is entered in the staffNameAppointPartTimeTextField.
13	getQualificationAppointPartTimeTextField()	• Accessor method that returns the value of qualificationAppointPartTimeTextField as String type. • It contains if statement which shows appropriate message when no value is entered in the qualificationAppointPartTimeTextField.
14	getAppointedByAppointPartTimeTextField()	• Accessor method that returns the value of appointedByAppointPartTimeTextField as String type. • It contains if statement which shows appropriate message when no value is entered in the appointedByAppointPartTimeTextField.
15	getJobTypePartTimeComboBox()	• Accessor method that returns the value of jobTypePartTimeComboBox that the user has choosen as String type. • It contains if statement which shows appropriate message when Full Time is choosen in the jobTypePartTimeComboBox.

16	getShiftPartTimeComboBox()	• Accessor method that returns the value of shiftPartTimeComboBox that the user has choosen as String type.
17	getJoiningDateAppointPartTime()	• Accessor method that returns the the joining date by concatenating the values of the joiningDateYearAppointPartTimeComboBox, joiningDateMonthAppointPartTimeComboBox and joiningDateDayAppointPartTimeComboBox that the user has choosen as String type. • It contains if statement which shows appropriate message when the user doesnot select from the joiningDateYearAppointPartTimeComboBox, joiningDateMonthAppointPartTimeComboBox or joiningDateDayAppointPartTimeComboBox.
18	getVacancyNumberFullTimeTextField()	• Accessor method that returns the value of vacancyNumberFullTimeTextField as Int type. • It contains try catch statement and shows appropriate message when unsuitable value is entered in the vacancyNumberFullTimeTextField.
19	getDesignationFullTimeTextField()	• Accessor method that returns the value of designationFullTimeTextField as String type. • It contains if statement and shows appropriate message when no value is entered in the designationFullTimeTextField.

20	getWorkingHoursFullTimeTextField()	• Accessor method that returns the value of workingHoursFullTimeTextField as int type. • It contains try catch statement and shows appropriate message when unsuitable value is entered in the workingHoursFullTimeTextField.
21	getSalaryFullTimeTextField()	• Accessor method that returns the value of salaryFullTimeTextField as int type. • It contains try catch statement and shows appropriate message when unsuitable value is entered in the salaryFullTimeTextField.
22	getJobTypeFullTimeComboBox()	• Accessor method that returns the value of jobTypeFullTimeComboBox that the user has choosen as String type. • It contains if statement which shows appropriate message when Part Time is choosen in the jobTypeFullTimeComboBox.
23	getVacancyNumberAppointFullTimeTextField()	• Accessor method that returns the value of vacancyNumberAppointFullTimeTextField as Int type. • It contains try catch statement and shows appropriate message when unsuitable value is entered in the vacancyNumberAppointFullTimeTextField.
24	getStaffNameAppointFullTimeTextField()	• Accessor method that returns the value of staffNameAppointFullTimeTextField as String type. • It contains if statement which shows appropriate message when no value is

		entered in the staffNameAppointFullTimeTextField.
25	getQualificationAppointFullTimeTextField()	• Accessor method that returns the value of qualificationAppointFullTimeTextField as String type. • It contains if statement which shows appropriate message when no value is entered in the qualificationAppointFullTimeTextField.
26	getAppointedByAppointFullTimeTextField()	• Accessor method that returns the value of appointedByAppointFullTimeTextField as String type. • It contains if statement which shows appropriate message when no value is entered in the appointedByAppointFullTimeTextField.
27	getJoiningDateAppointFullTimeTextField()	• Accessor method that returns the the joining date by concatenating the values of the joiningDateYearAppointFullTimeComboBox, joiningDateMonthAppointFullTimeComboBox and joiningDateDayAppointFullTimeComboBox that the user has choosen as String type. • It contains if statement which shows appropriate message when the user doesnot select from the joiningDateYearAppointFullTimeComboBox, joiningDateMonthAppointFullTimeComboBox or joiningDateDayAppointFullTimeComboBox.

28	getTerminateVacancyNumber TextField()	• Accessor method that returns the value of terminateVacancyNumberTextFieldas Int type. • It contains try catch statement and shows appropriate message when unsuitable value is entered in the terminateVacancyNumberTextField.
29	actionPerformed(ActionEvent e)	• It is invokes automatically just after the user performs an action. • It is used to add functionality to the JButton.
30	Static main(String [ ] args)	• Main method which calls the constructor of the class.

Table 1: Method Description of INGNepal

5. Testing

Testing is the activity to check whether the actual results match the expected results and to ensure that the software system is defect free. (Guru99, 2020)

5.1. Test 1 – To test that the program can be compiled and run using the command prompt

Test No:	1
Objective:	To test that the program can be compiled and run using the command prompt
Action:	• The command prompt is opened. • The directory is changed to the file location, • The java files are compiled using javac command. • The java file ING Nepal containing GUI is ran using “java INGNepal.java” command.
Expected Result:	The program files would be compiled, and the program would run.
Actual Result:	The program files were compiled and the program was ran.
Conclusion:	The test is successful.

Table 2: To test that the program can be compiled and run using the command prompt

Output Result:

```
Command Prompt
Microsoft Windows [Version 10.0.16299.1868]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\pra0219>D:

D:\>cd D:\JavaPrograms\2nd Semester Assignment\Finalize\Test 1

D:\JavaPrograms\2nd Semester Assignment\Finalize\Test 1>dir
 Volume in drive D has no label.
 Volume Serial Number is 2CFE-15C1

 Directory of D:\JavaPrograms\2nd Semester Assignment\Finalize\Test 1

05/26/2020  09:38 PM    <DIR>          .
05/26/2020  09:38 PM    <DIR>          ..
04/08/2020  02:16 PM             4,377 FullTimeStaffHire.java
04/09/2020  07:55 PM            53,303 INGNepal.java
04/08/2020  02:16 PM             4,779 PartTimeStaffHire.java
03/27/2020  08:23 PM             2,328 StaffHire.java
               4 File(s)         64,787 bytes
               2 Dir(s)  410,497,773,568 bytes free

D:\JavaPrograms\2nd Semester Assignment\Finalize\Test 1>
```

Figure 3: Screenshot of the files in the Directory containing the java files

```
Command Prompt
D:\JavaPrograms\2nd Semester Assignment\Finalize\Test 1>javac StaffHire.java

D:\JavaPrograms\2nd Semester Assignment\Finalize\Test 1>javac FullTimeStaffHire.java

D:\JavaPrograms\2nd Semester Assignment\Finalize\Test 1>javac PartTimeStaffHire.java

D:\JavaPrograms\2nd Semester Assignment\Finalize\Test 1>javac INGNepal.java

D:\JavaPrograms\2nd Semester Assignment\Finalize\Test 1>
```

Figure 4: Screenshot of compiling the java files

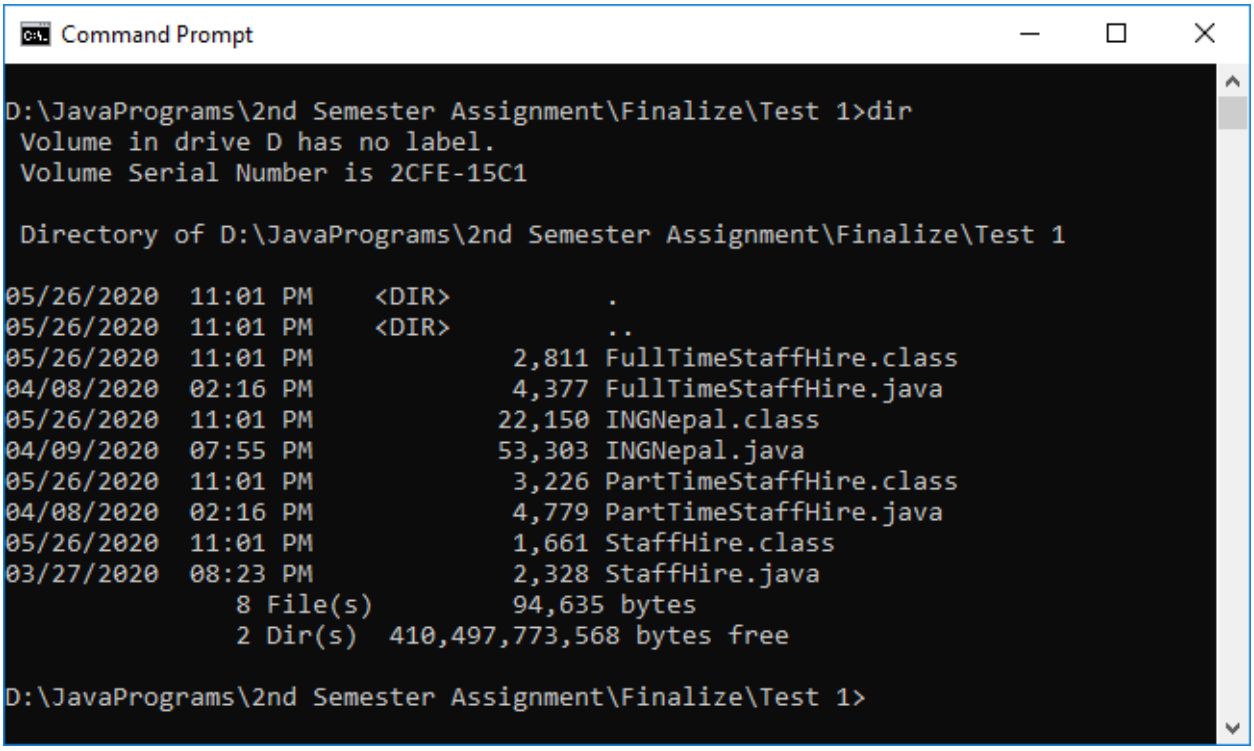

Figure 5: Screenshot of the files in the Directory containing the java files after they are compiled

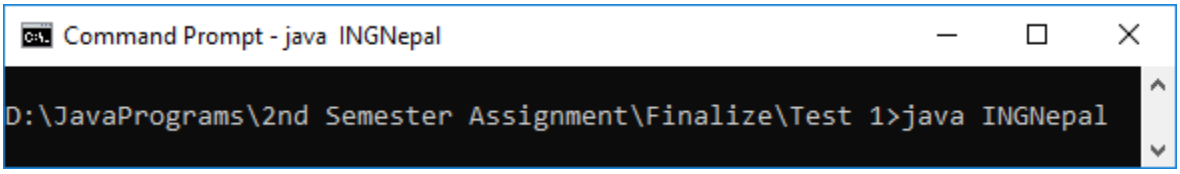

Figure 6: Screenshot of running the program

Figure 7: Screenshot of the program

5.2. Test 2

5.2.1. To add Vacancy for Full Time Staff

Test No:	2.1
Objective:	To add vacancy for Full Time Staff

Action:	• Following values are entered to the respective Text fields:
	  ○ vacancyNumberFullTimeTextField=”1”
	  ○ designationFullTimeTextField=” Teacher”
	  ○ workingHourFullTimeTextField=”8”
	  ○ salaryFullTimeTextField=”80000”
	• The Save button is clicked.
	• The Display button is clicked.

Expected Result:	The vacancy would be added.
Actual Result:	The vacancy was added.
Conclusion:	The test is successful.

Table 3: To Add Vacancy for Full Time Staff

Output Result:

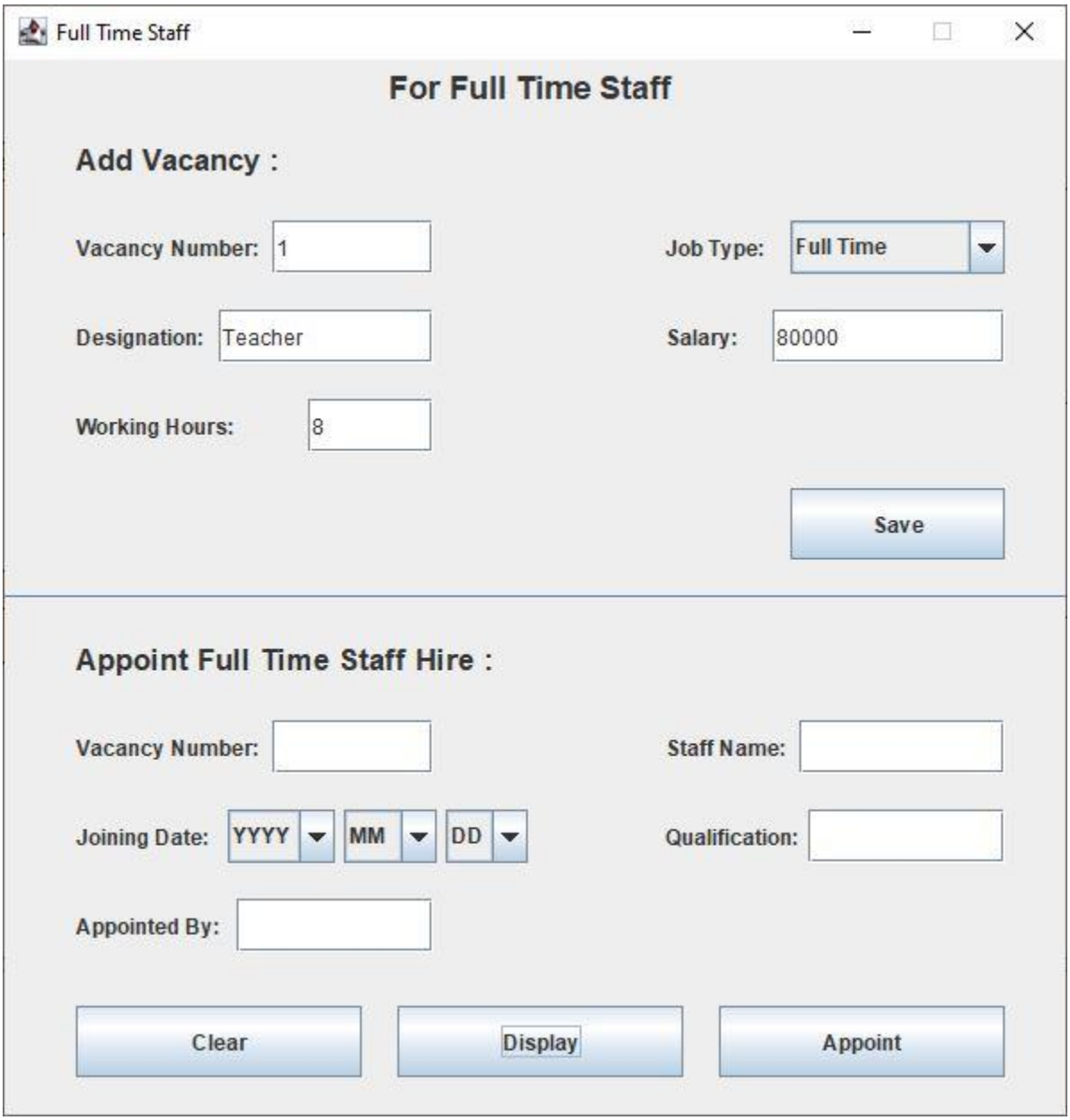

Figure 8: Screenshot of adding vacancy for Full Time Staff

Figure 9: Screenshot of message saying the vacancy has been added

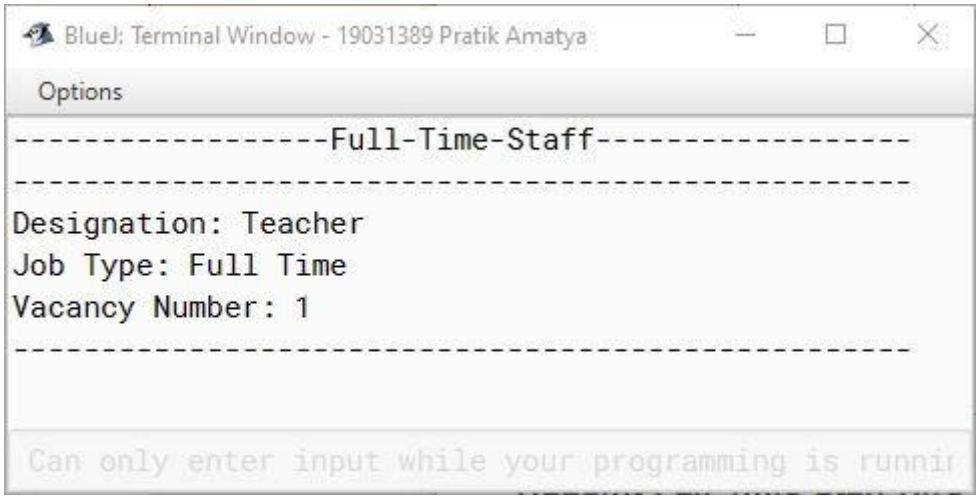

Figure 10: Displaying the vacancy for Full Time Staff

5.2.2. To add Vacancy for Part Time Staff

Test No:	2.2
Objective:	To add vacancy for Part Time Staff

Action:	• Following values are entered to the respective Text fields:
	  ○ vacancyNumberPartTimeTextField=”11”
	  ○ designationPartTimeTextField=” Janitor”
	  ○ workingHoursPerDayPartTimeTextField=”2”
	  ○ wagesPerHourTextField=”350”
	• Selecting shift as Morning from the Combo Box.
	• The Save button is clicked.
	• The Display button is clicked.

Expected Result:	The vacancy would be added.
Actual Result:	The vacancy was added.
Conclusion:	The test is successful.

Table 4: To add Vacancy for Part Time Staff

Output Result:

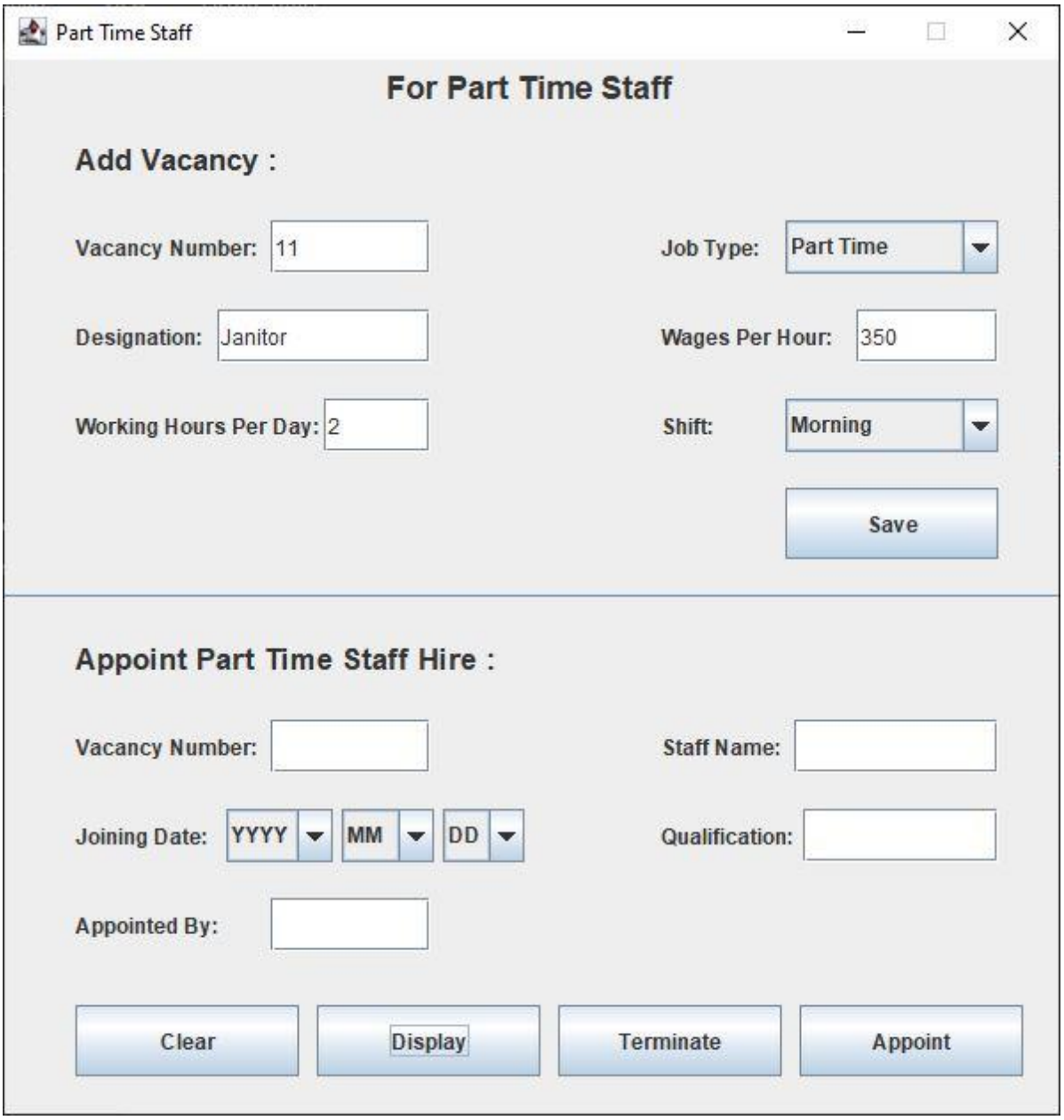

Figure 11: Screenshot of adding vacancy for Part Time Staff

Figure 12: Screenshot of message saying the vacancy has been added

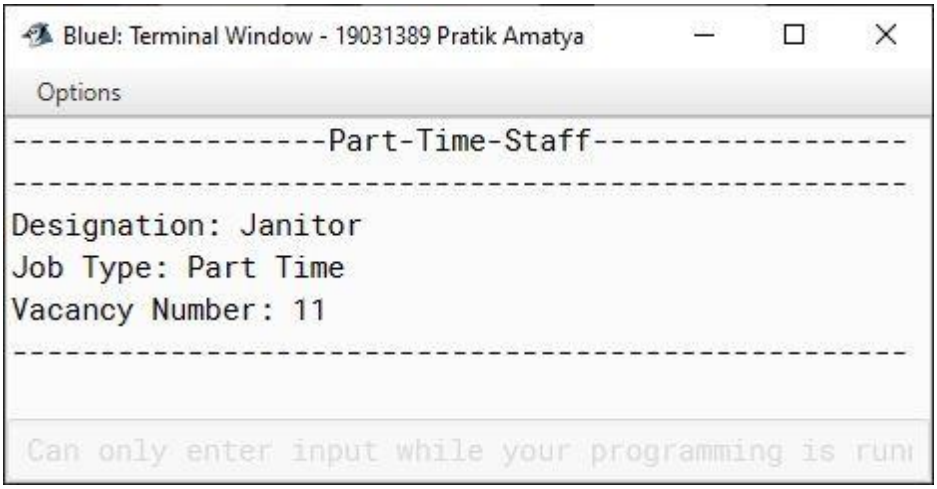

Figure 13: Displaying the vacancy for Full Time Staff

5.2.3. To appoint Full Time Staff

Test No:	2.3
Objective:	To appoint Full Time Staff

Action:	• Following values are entered to the respective Text fields:
	  ○ vacancyNumberAppointFullTimeTextField=”1”
	  ○ staffNameAppointFullTimeTextField=” Suman Subedi”
	  ○ qualificationAppointFullTimeTextField=” MBA”
	  ○ appointedByAppointFullTimeTextField=” Principal”
	• Selecting the Joining Date from the Combo Box i.e. Selecting year as 2020 month as Nov and day 20.
	• The Save button is clicked.
	• The Display button is clicked.

Expected Result:	The Full Time Staff would be appointed.
Actual Result:	The Full Time Staff was appointed.
Conclusion:	The test is successful.

Table 5: To appoint Full Time Staff

Output Result:

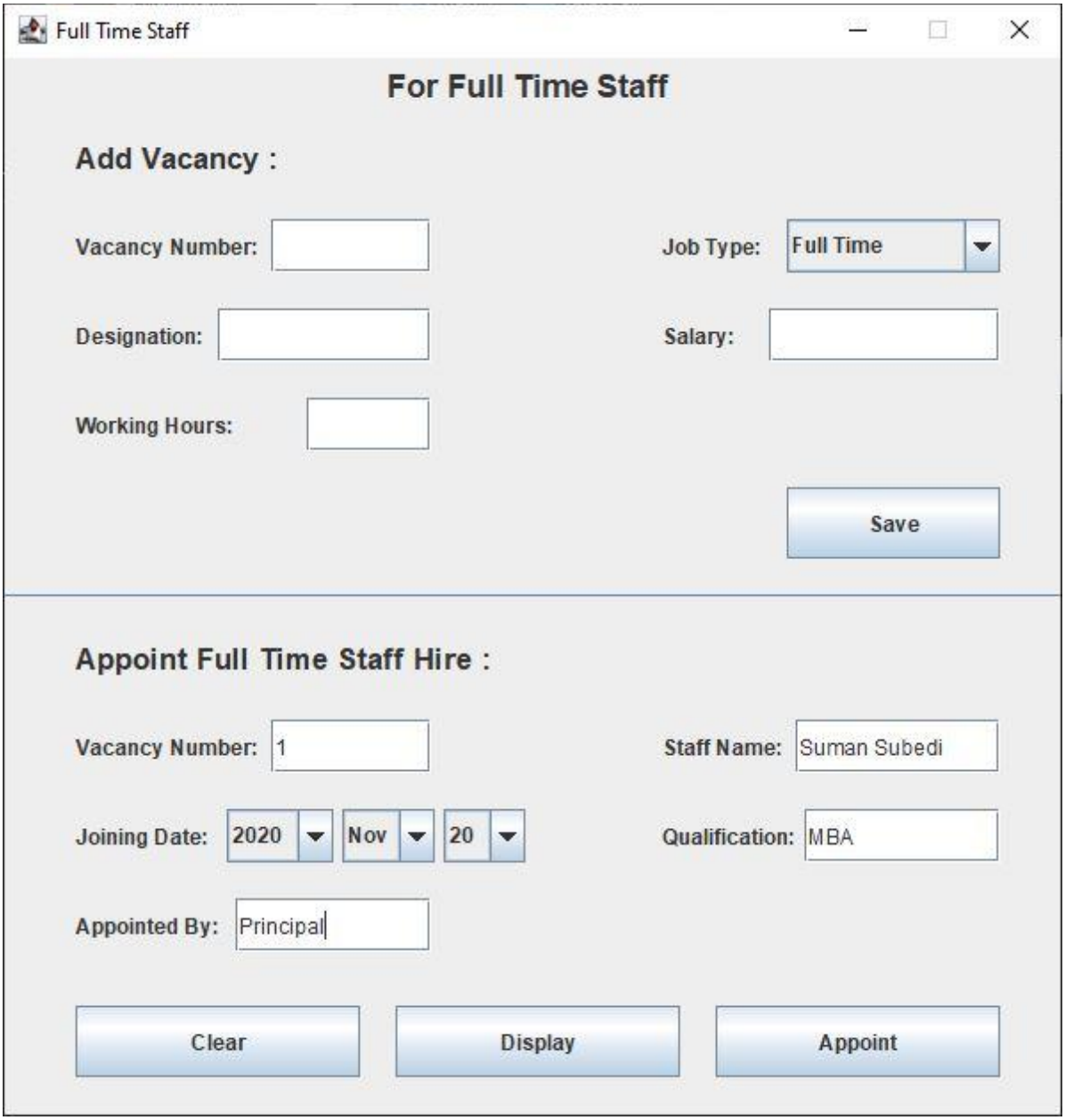

Figure 14: Screenshot of appointing Full Time Staff

Figure 15: Screenshot of message saying the staff has been appointed

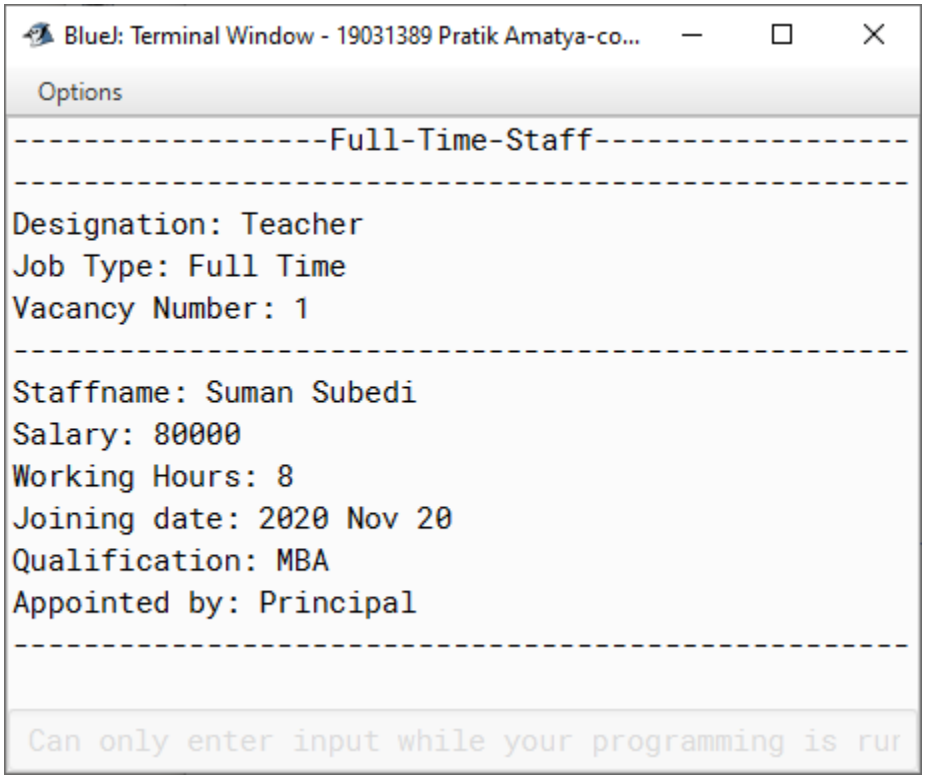

Figure 16: Displaying the details of Full Time Staff

5.2.4. To Appoint Part Time Staff

Test No:	2.4
Objective:	To appoint a Part Time Staff

Action:	• Following values are entered to the respective Text fields:
	  ○ vacancyNumberAppointPartTimeTextField=”11”
	  ○ staffNameAppointPartTimeTextField=” Badrinath Bhusal”
	  ○ qualificationAppointPartTimeTextField=” S.E.E”
	  ○ appointedByAppointPartTimeTextField=” Principal”
	• Selecting the Joining Date from the Combo Box i.e. Selecting year as 2020 month as Dec and day 20.
	• The Appoint button is clicked.
	• The Display button is clicked.

Expected Result:	The Part Time Staff would be appointed.
Actual Result:	The Part Time Staff was appointed.
Conclusion:	The test is successful.

Table 6: To appoint Part Time Staff

Output Result:

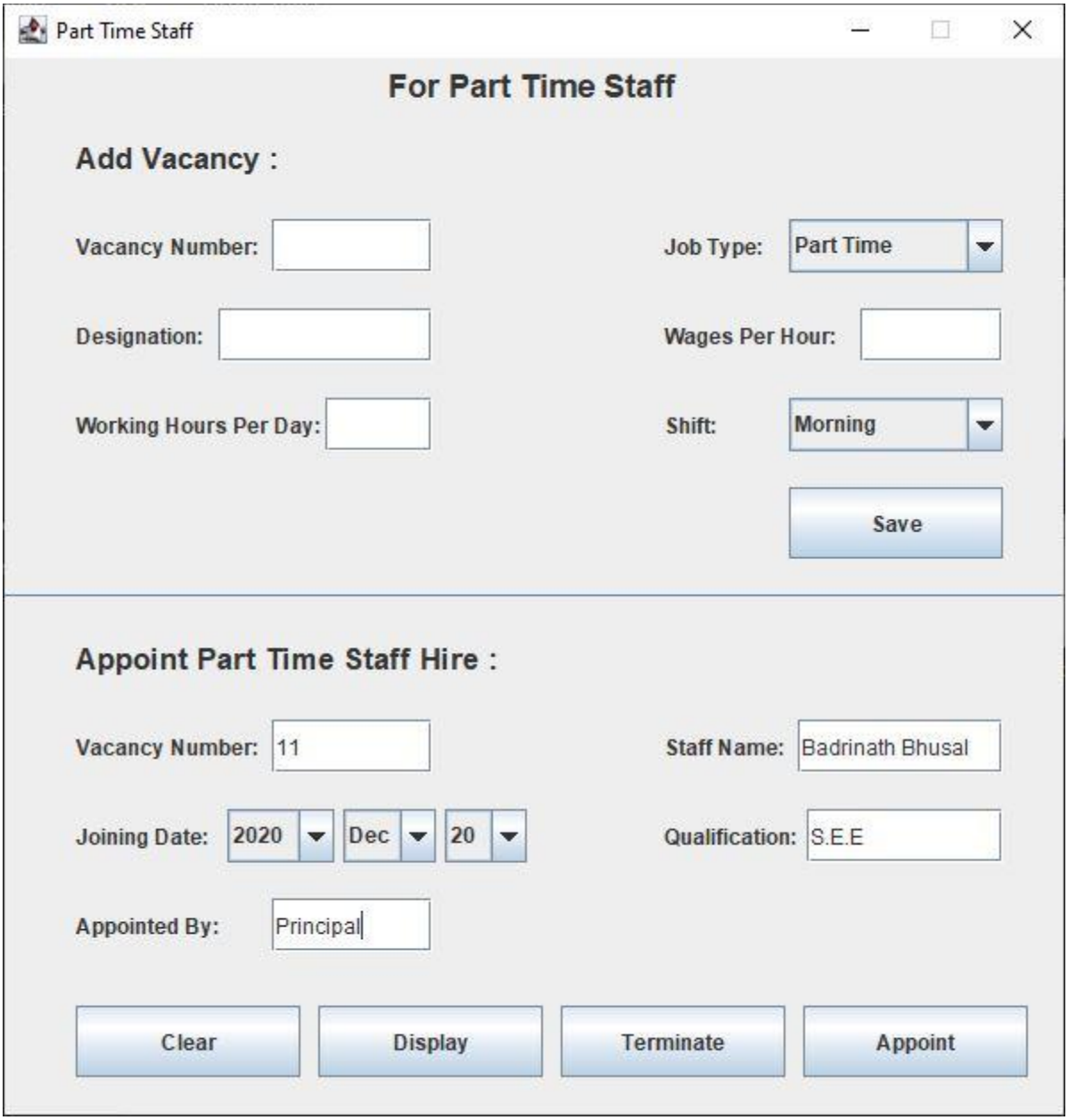

Figure 17: Screenshot of appointing Part Time Staff

Figure 18: Screenshot of message saying the staff has been appointed

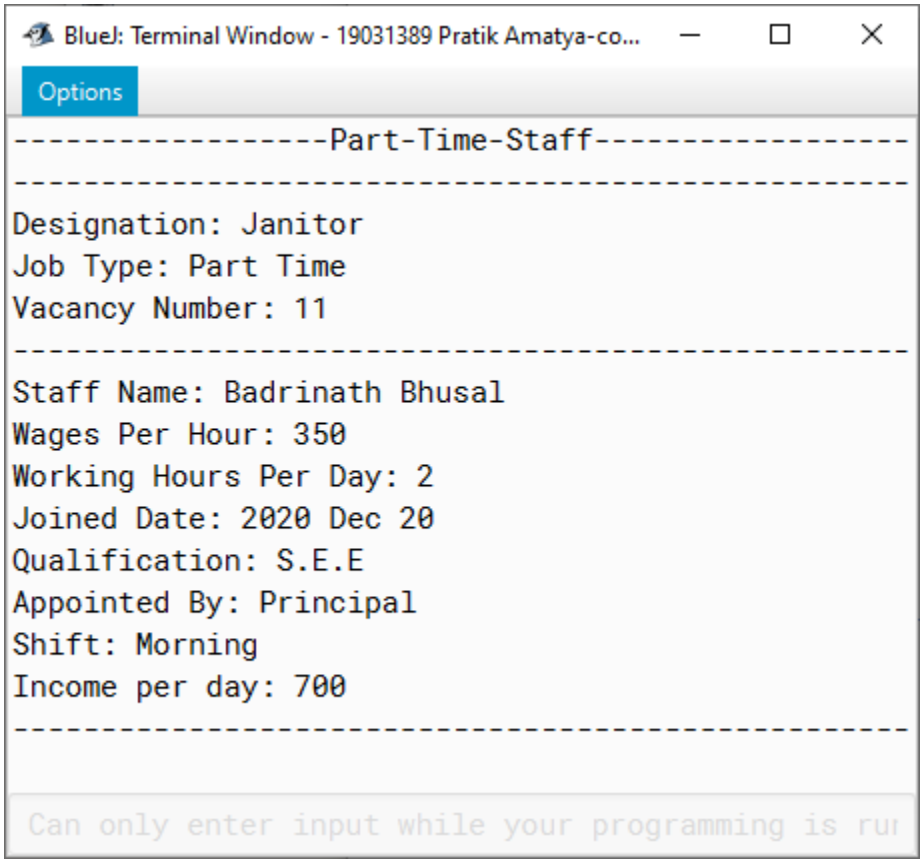

Figure 19: Displaying the details of Part Time Staff

5.2.5. To Terminate a Part Time Staff

Test No:	2.5
Objective:	To terminate a Part Time Staff
Action:	• The vacancy number of the concerned Part Time i.e. “11” was entered into the terminateVacancyNumberTextField. • The Terminate button is clicked. • The Display button is clicked.
Expected Result:	The Part Time Staff would be terminated.
Actual Result:	The Part Time Staff was terminated.
Conclusion:	The test is successful.

Table 7: To Terminate a Part Time Staff

Output Result:

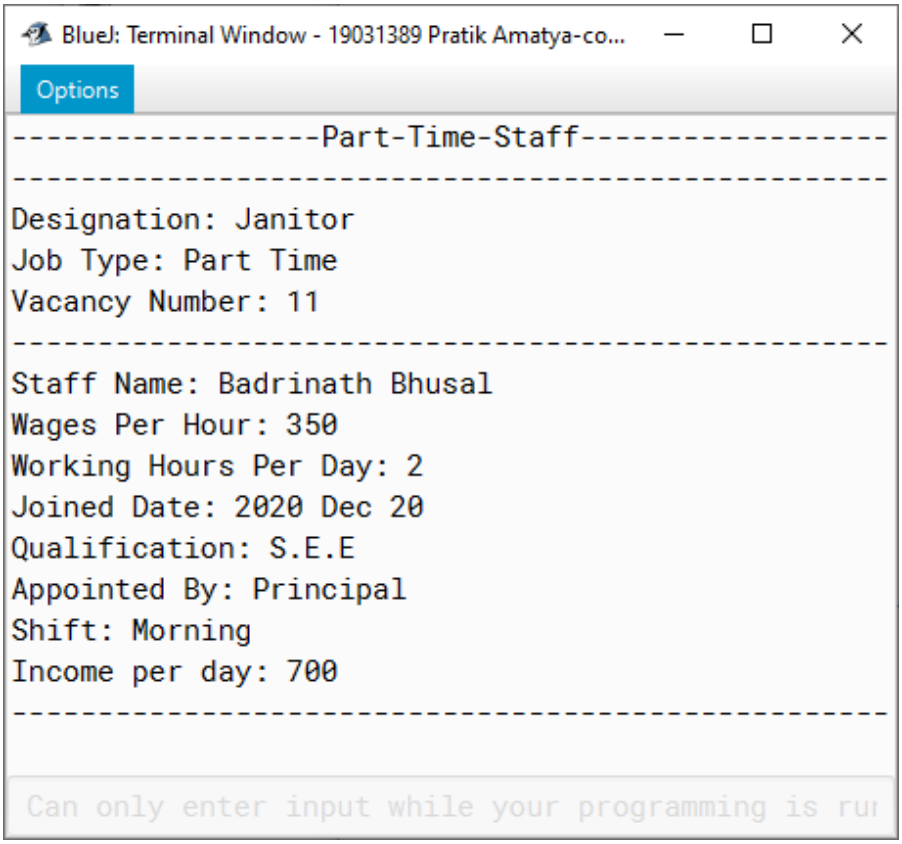

Figure 20: Displaying the details of the Part Time Staff before being terminated

Figure 21: Screenshot of Entering the Vacancy Number of Part Time Staff that is to be terminated

Figure 22: Screenshot of message saying the staff has been terminated

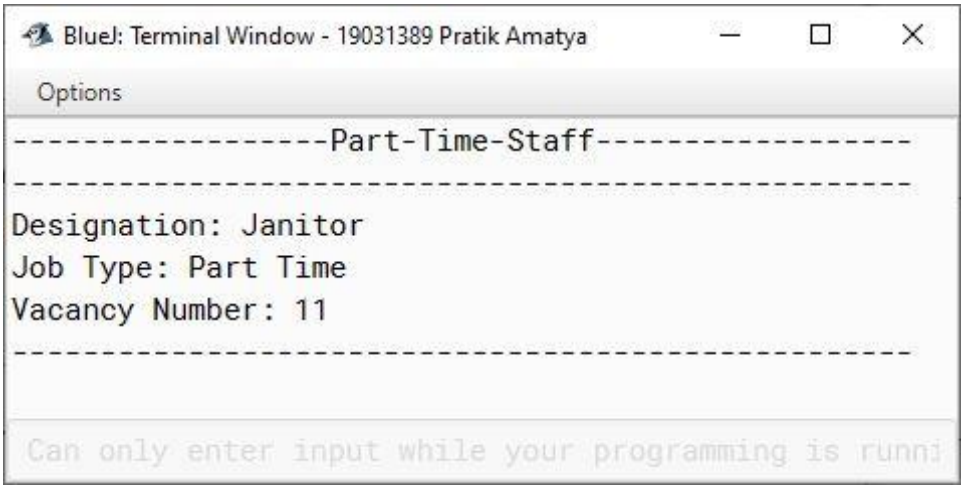

Figure 23: Displaying the details of the Part Time Staff after being terminated

5.3. Test 3 – To Test that appropriate dialog boxes appear when unsuitable values are entered for the vacancy number

Test No:	3
Objective:	To test that appropriate dialog boxes appear when unsuitable values are entered for the vacancy number
Action:	• All the appropriate values were entered into the text fields except for Vacancy Number in the add vacancy section for Full Time Staff. • Save button was clicked. • The number was entered in form of words in vacancyNumberFullTimeTextField in the add vacancy section for Full Time Staff. And the appropriate values were entered into the remaining text fields. • Save button was clicked. • The decimal number was entered in vacancyNumberFullTimeTextField in the add vacancy section for Full Time Staff. And the appropriate values were entered into the remaining text fields. • Save button was clicked. • The number 0 was entered in vacancyNumberFullTimeTextField in the add vacancy section for Full Time Staff. And the appropriate values were entered into the remaining text fields. • Save button was clicked. • The negative number -1 was entered in vacancyNumberFullTimeTextField in the add vacancy section for Full Time Staff. And the appropriate values were entered into the remaining text fields. • Save button was clicked. • The vacancy number 1 belonging to Part Time Staff was entered in the vacancyNumberTextField of Full Time Staff GUI to add vacancy for Full Time Staff. And the appropriate values were entered into the remaining text fields. • Save button was clicked.

	• The vacancy number 2 belonging to Part Time Staff was entered in the vacancy number textfield for appointing Part Time Staff in Full Time Staff GUI. • Appoint button was clicked.
Expected Result:	Appropriate dialog boxes would appear when unsuitable values are entered for the vacancy number.
Actual Result:	Appropriate dialog boxes appeared when unsuitable values were entered for the vacancy number.
Conclusion:	The test is successful.

Table 8: To Test that appropriate dialog boxes appear when unsuitable values are entered for the vacancy number

Output Result:

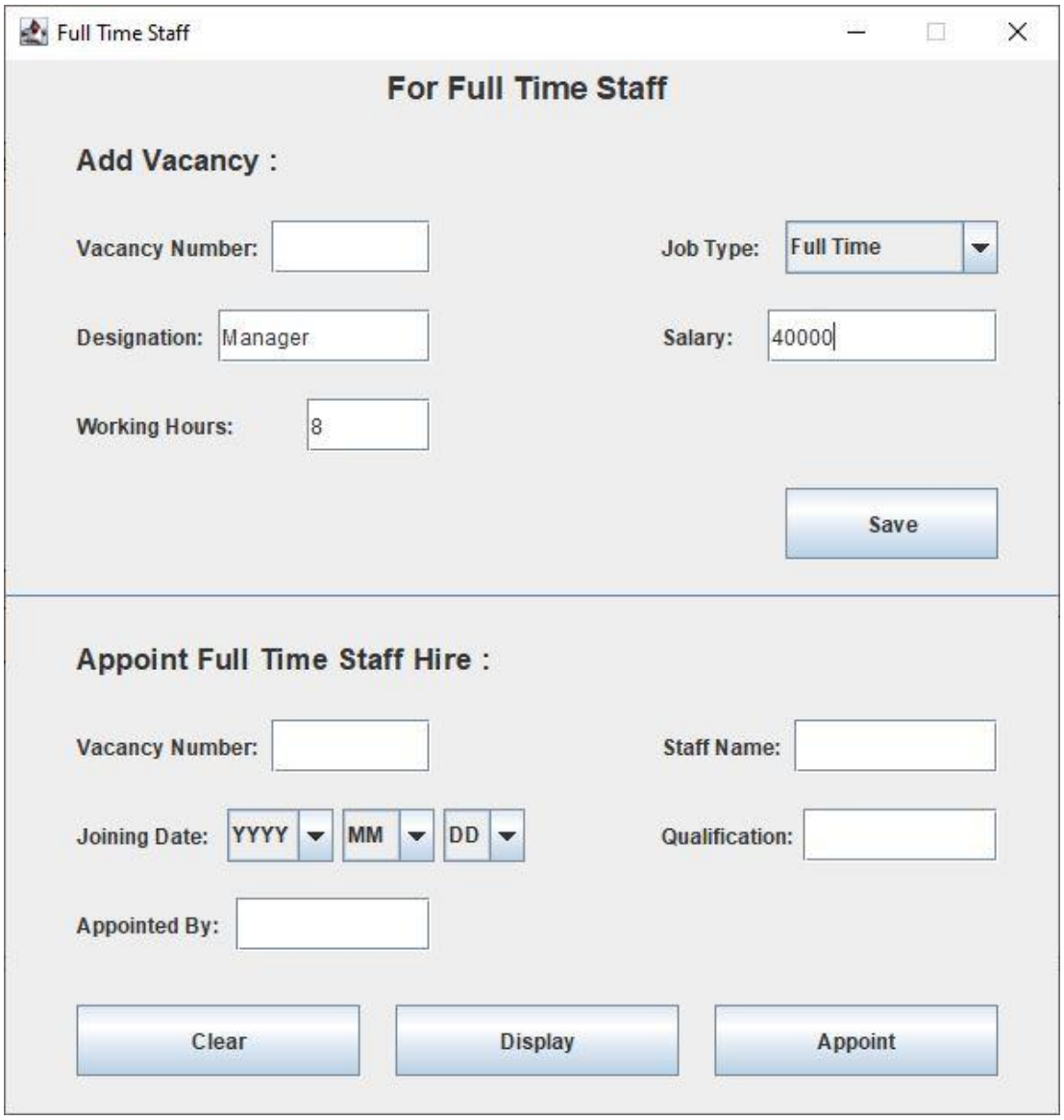

Figure 24: Screenshot of when the Vacancy Number Text Field is empty

Figure 25: Screenshot message saying vacancy number has to be entered in the Text Field

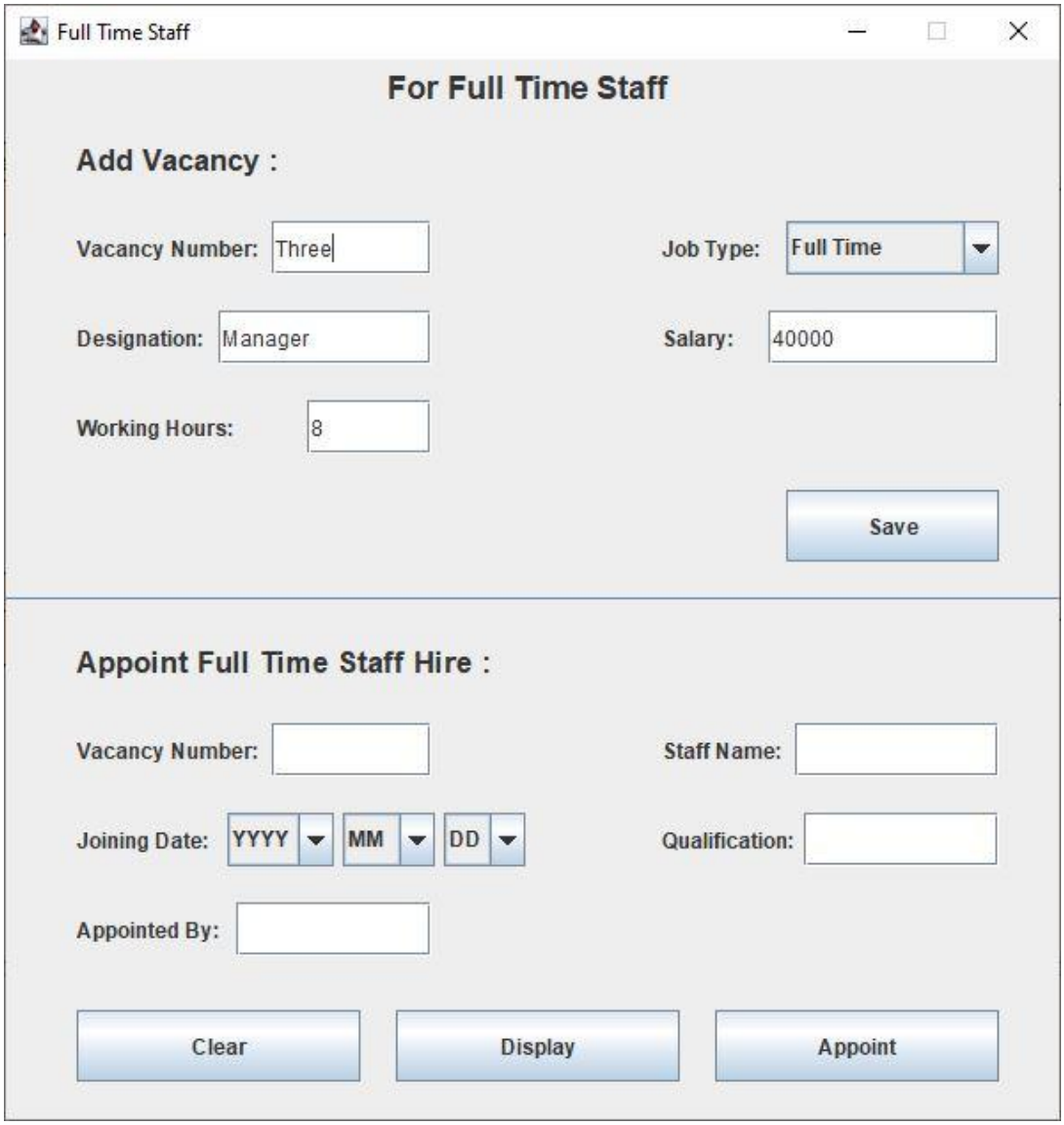

Figure 26: Screenshot of when word is entered in the Vacancy Number Text Field

Figure 27: Screenshot of message saying to enter whole numbers only in the Text Field

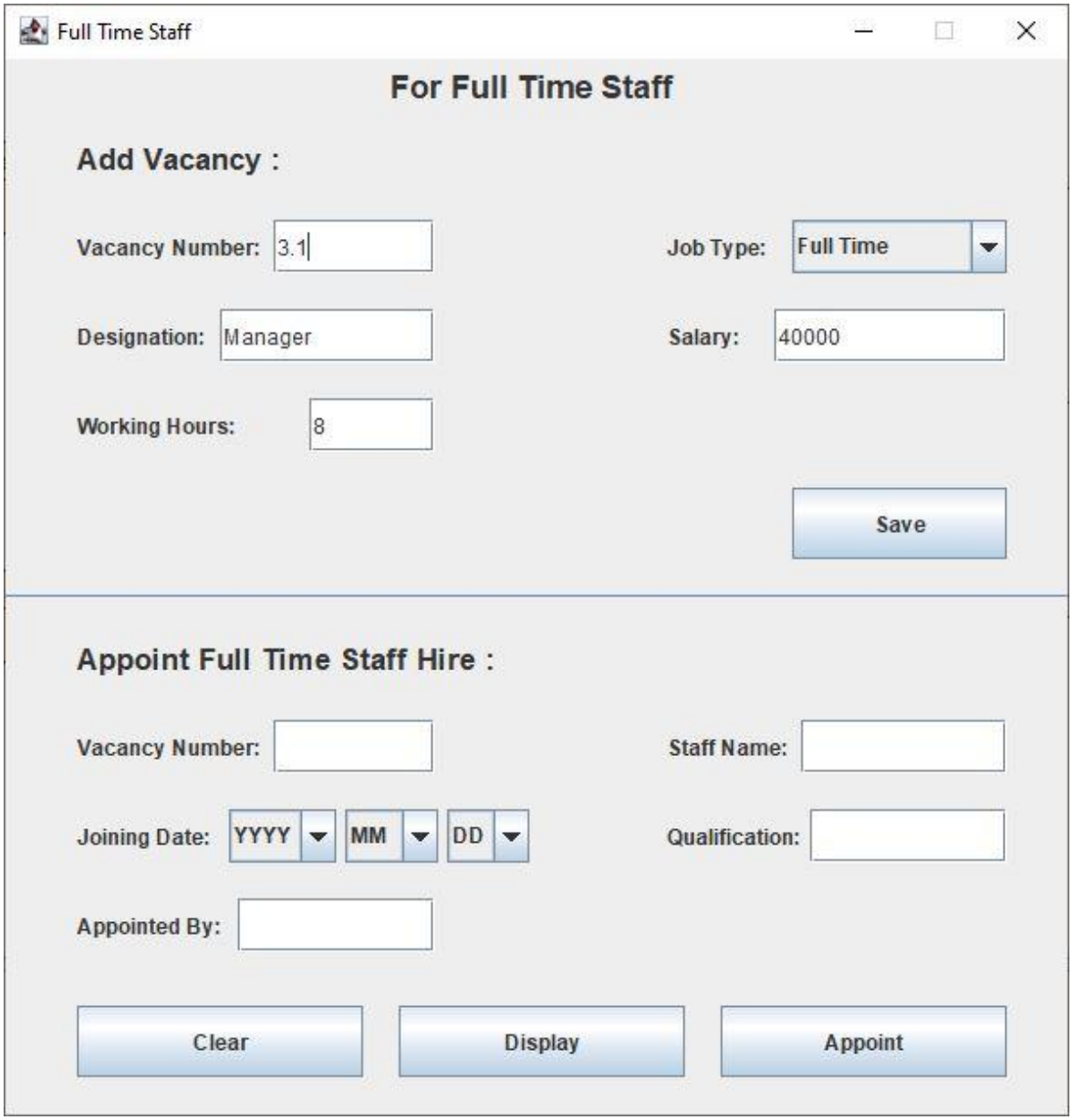

Figure 28: Screenshot of when Decimal value is entered in the Vacancy Number Text Field

Figure 29: Screenshot of message saying to enter whole numbers only in vacancy numbers

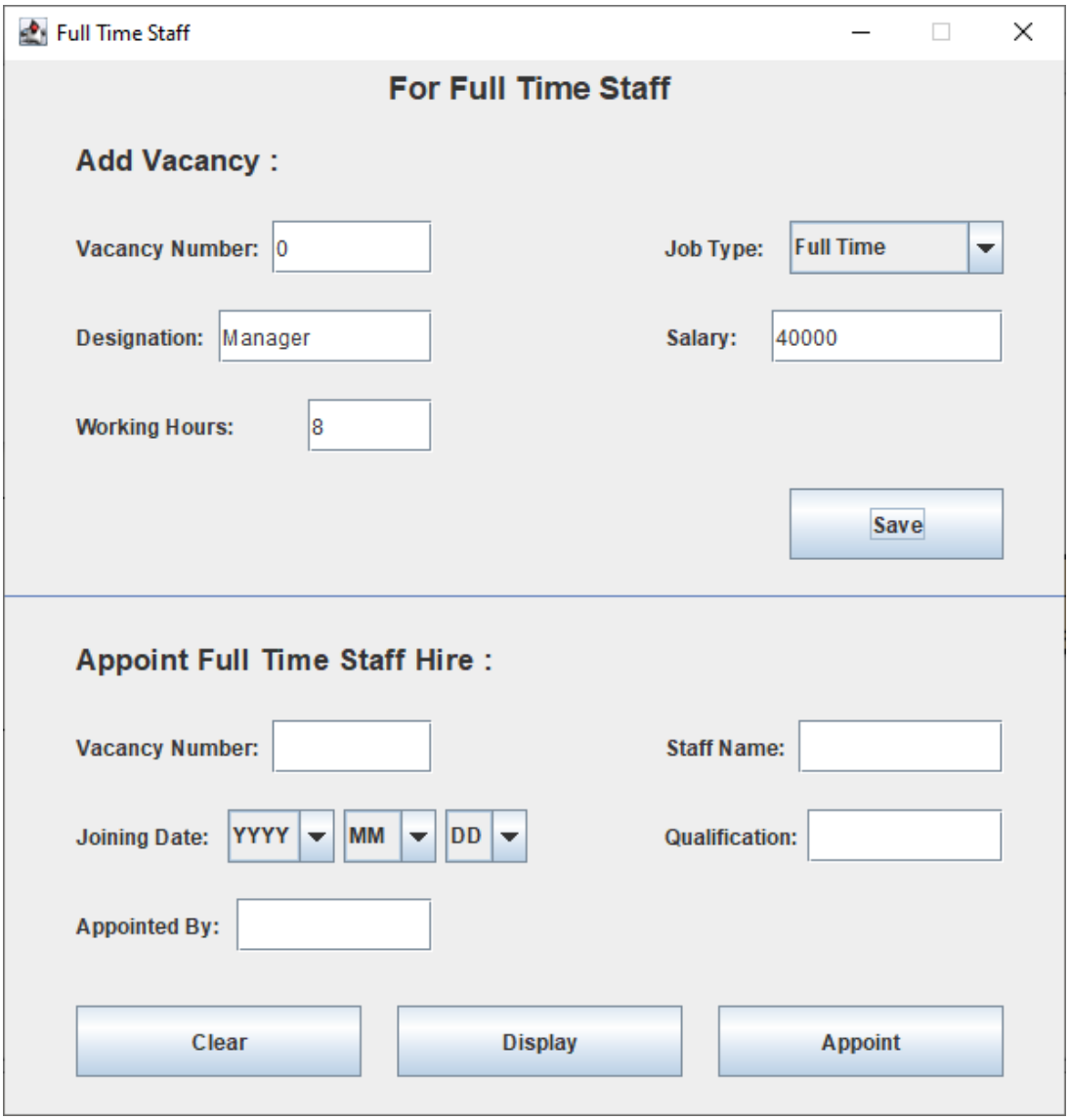

Figure 30: Screenshot of when number 0 is entered in the Vacancy Number Text Field

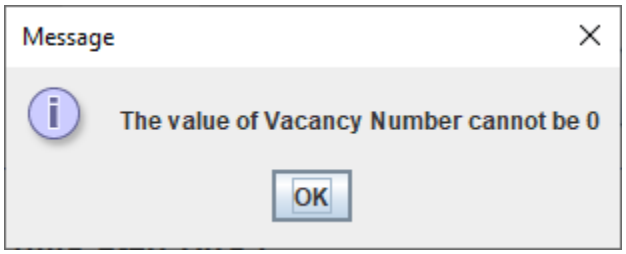

Figure 31: Screenshot of message saying value of vacancy number cannot be 0

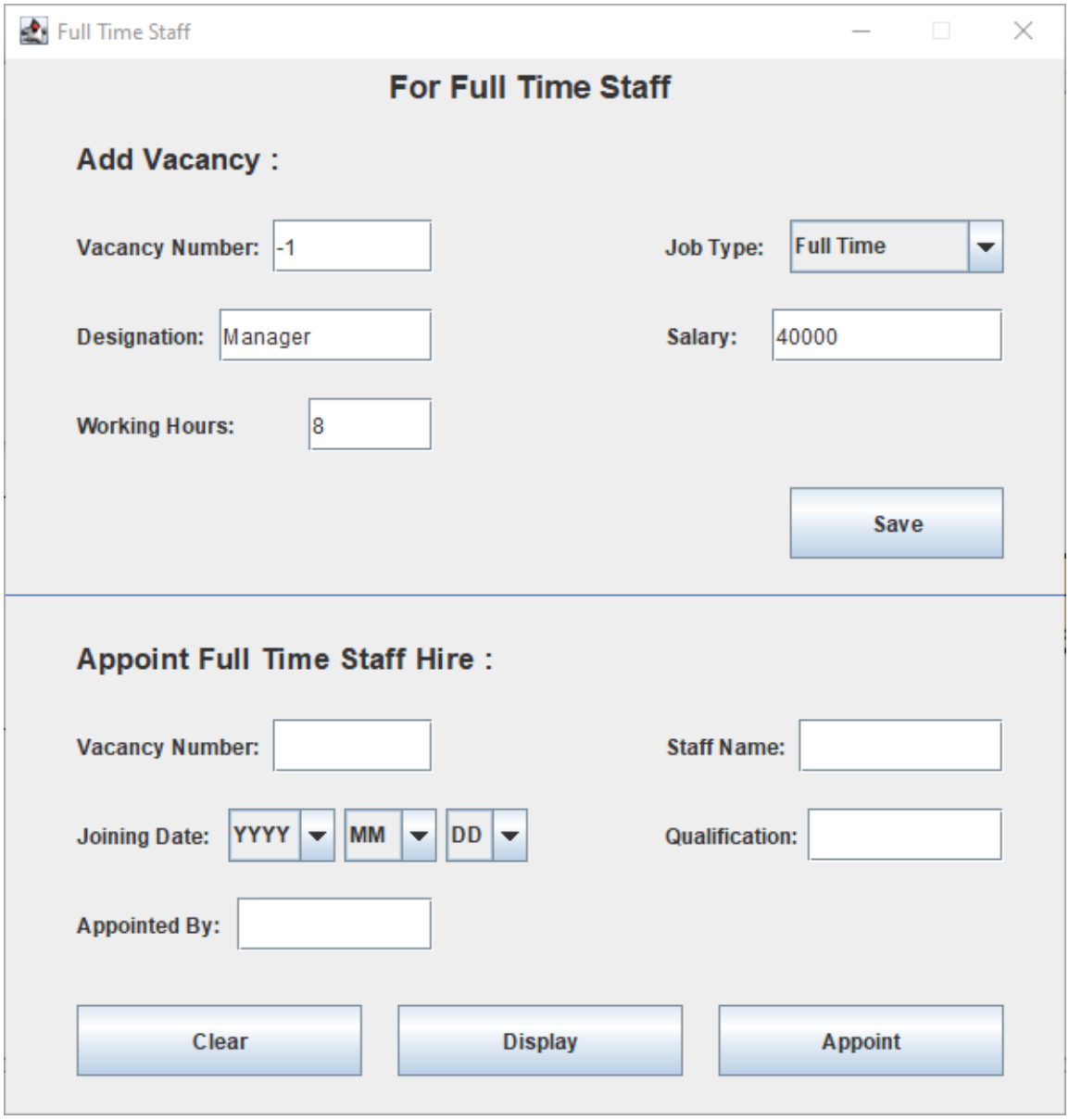

Figure 32: Screenshot of when Negative value is entered in the Vacancy Number Text Field

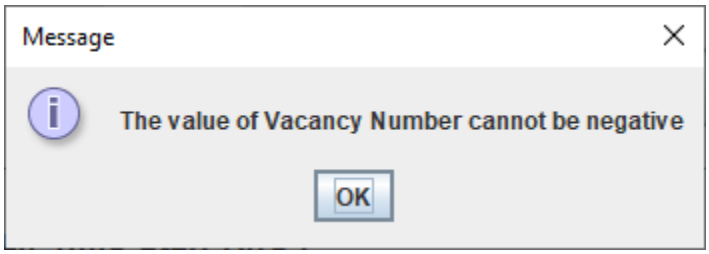

Figure 33: Screenshot of message saying value of vacancy number cannot be negative

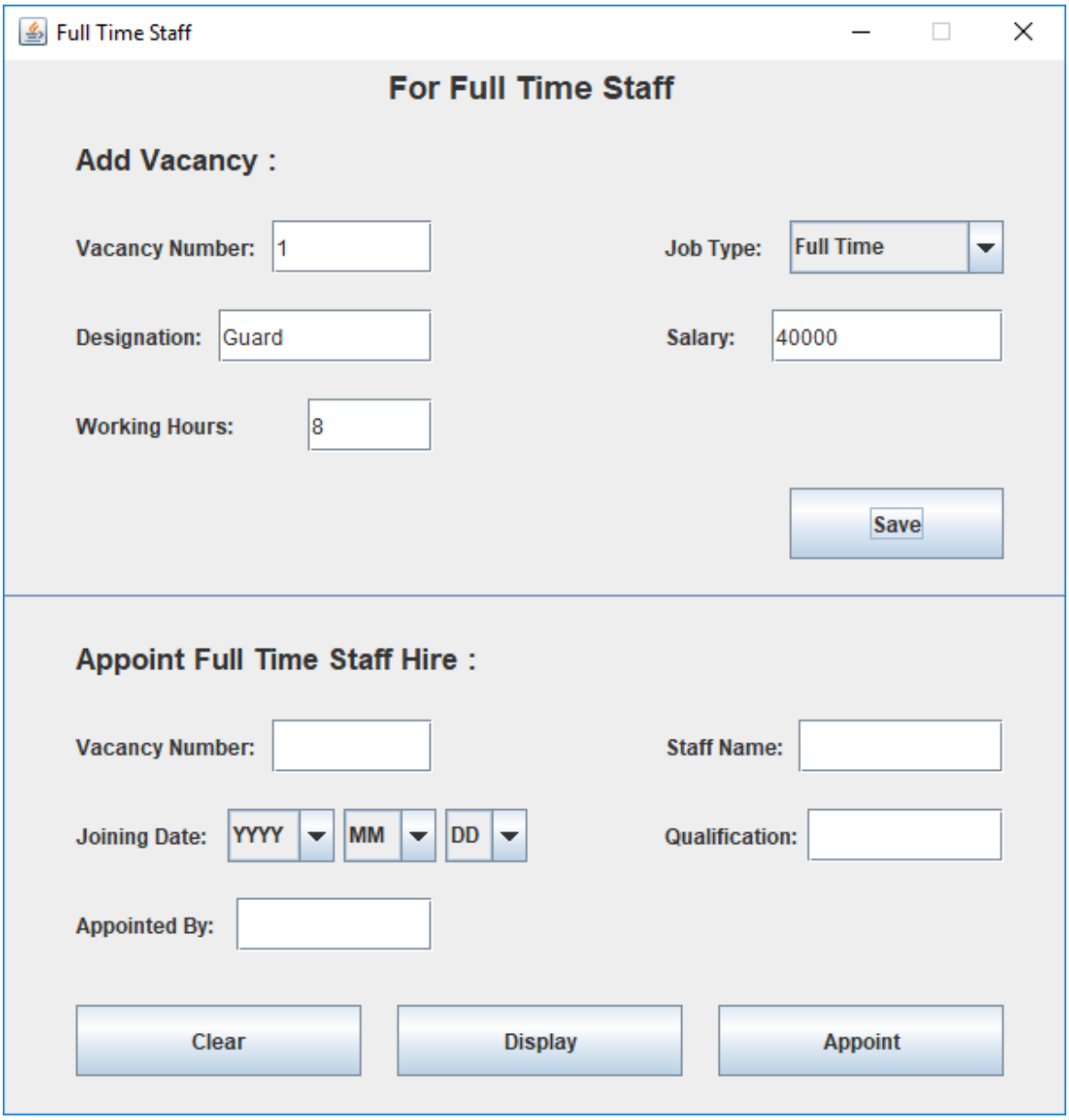

Figure 34: Screenshot of when vacancy number belonging to Part Time Staff is entered in the Vacancy Number Text Field of Full Time Staff GUI

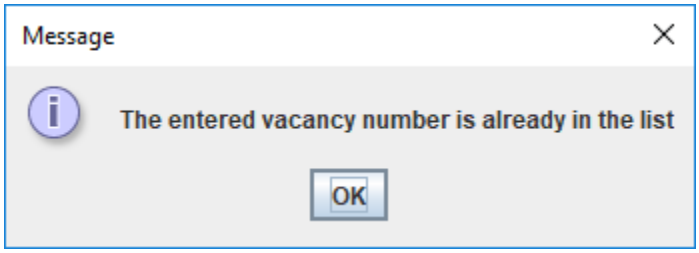

Figure 35: Screenshot of message saying value of vacancy number is aleady in the list

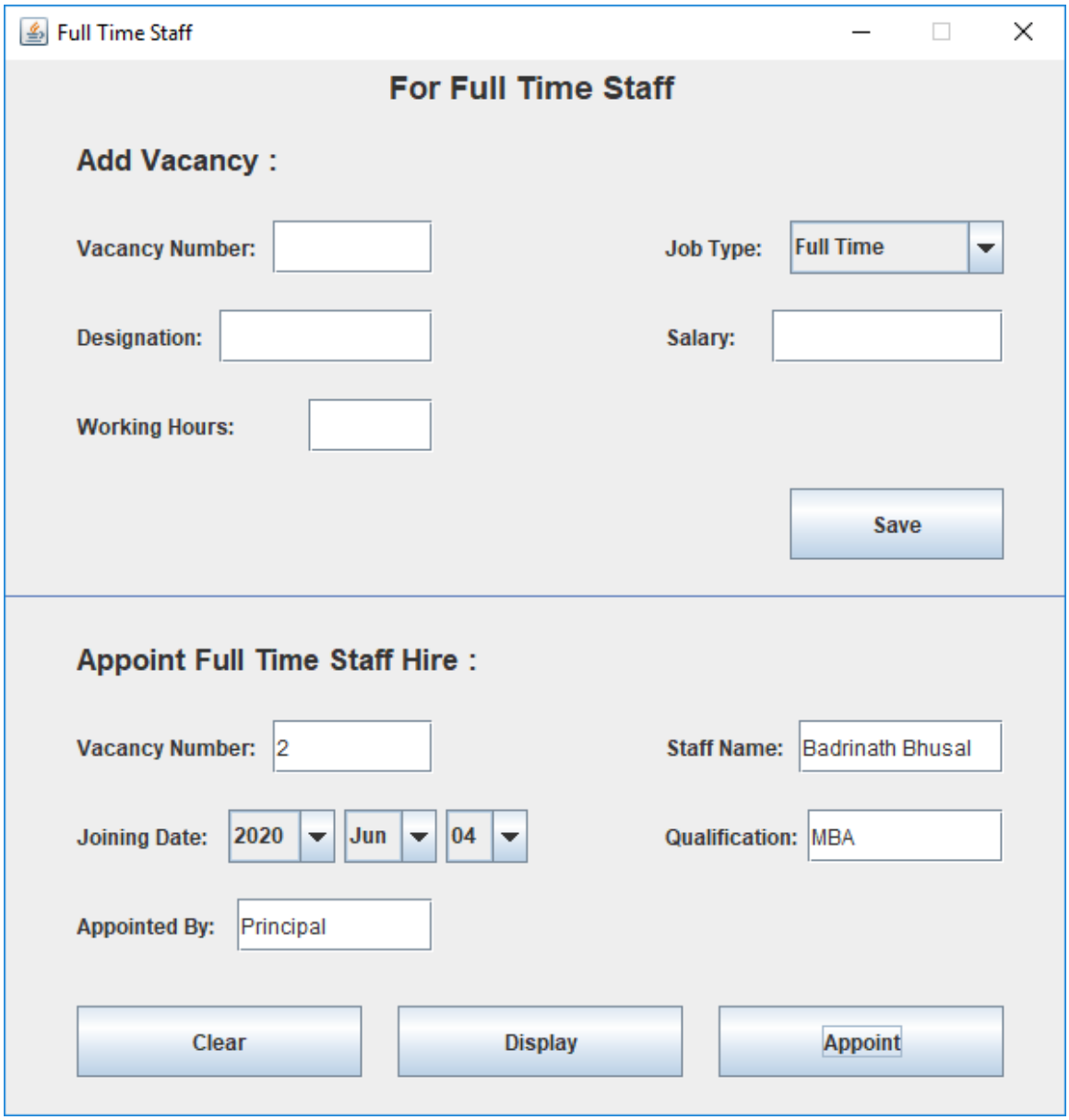

Figure 36: Screenshot of when vacancy number belonging to Part Time Staff is entered in the Vacancy Number Text Field of Full Time Staff GUI to appoint Part Time Staff

Figure 37: Screenshot of message saying value of vacancy number is not for Full Time Staff Hire

6. Error Handling

6.1. Error 1: Syntax Error

The case below is an example of syntax error. It usually occurs because of incorrect spelling or absence of variable declaration. Due to the missing parenthesis () in the event error message “cannot find symbol – variable getJobTypeFullTimeComboBox”. The missing parenthesis () causes “getJobTypeFullTimeComboBox” to be seen as variable which has not been initialized before instead of method. This is easily fixed by adding the missing brackets.

```
if (e.getSource() == saveFullTimeButton){
    boolean duplicateVacancyNum=false;

    String jobTypeft=getJobTypeFullTimeComboBox;
    int vacancyNumberft=getVacancyNumberFullTimeTextField();
    String designationft=getDesignationFullTimeTextField();
    int salaryft=getSalaryFullTimeTextField();
    int workingHoursft=getWorkingHoursFullTimeTextField();
```

Figure 38: Screenshot of the error 1

```
if (e.getSource() == saveFullTimeButton){
    boolean duplicateVacancyNum=false;

    String jobTypeft=getJobTypeFullTimeComboBox();
    int vacancyNumberft=getVacancyNumberFullTimeTextField();
    String designationft=getDesignationFullTimeTextField();
    int salaryft=getSalaryFullTimeTextField();
    int workingHoursft=getWorkingHoursFullTimeTextField();
```

Figure 39: Screenshot for the correction of the error 1

6.2. Error 2: Logical Error

The case below is an example of logic error. In the getter method, the variables are initialized as 0 or empty string depending upon the data type. In case of combo box, the value is initialized as the selected item by the user. Error message is displayed when the user enters an inappropriate value or does not enter at all. Even if error message is displayed, the initialized values of the variables which are obtained from their respective getter methods are passed. Hence, error message is displayed but the object is added to the list. Therefore, an If condition is added so that if any inappropriate value is present, the object made storing the values is not added to the array list

```java
if (e.getSource() == saveFullTimeButton){
    boolean duplicateVacancyNum=false;

    String jobTypeft=getJobTypeFullTimeComboBox();
    int vacancyNumberft=getVacancyNumberFullTimeTextField();
    String designationft=getDesignationFullTimeTextField();
    int salaryft=getSalaryFullTimeTextField();
    int workingHoursft=getWorkingHoursFullTimeTextField();

    // Iterates wthin the arraylist
    for (StaffHire obj:staffList){
        if(obj.getVacancyNumber() == vacancyNumberft){
            duplicateVacancyNum=true;
            break;
        }
    }
    if (duplicateVacancyNum==false){
        fullTimeStaff_obj= new FullTimeStaffHire(designationft,jobTypeft,vacancyNumberft , salaryft,workingHoursft);
        staffList.add(fullTimeStaff_obj);
        JOptionPane.showMessageDialog(frameFullTimeStaffHire,"Vacancy has been added");
    }else{// If the vacancy number is already in the list
        JOptionPane.showMessageDialog(frameFullTimeStaffHire, "The entered vacancy number is already in the list");
    }
}
```

Figure 40: Screenshot of the error 2

```java
if (e.getSource() == saveFullTimeButton){
    boolean duplicateVacancyNum=false;

    String jobTypeft=getJobTypeFullTimeComboBox();
    int vacancyNumberft=getVacancyNumberFullTimeTextField();
    String designationft=getDesignationFullTimeTextField();
    int salaryft=getSalaryFullTimeTextField();
    int workingHoursft=getWorkingHoursFullTimeTextField();

    if (vacancyNumberft >0 && salaryft > 0 && workingHoursft >0 && jobTypeft=="Full Time" && designationft != "") {
        // Iterates wthin the arraylist
        for (StaffHire obj:staffList){
            if(obj.getVacancyNumber() == vacancyNumberft){
                duplicateVacancyNum=true;
                break;
            }
        }
        if (duplicateVacancyNum==false){
            fullTimeStaff_obj= new FullTimeStaffHire(designationft,jobTypeft,vacancyNumberft , salaryft,workingHoursft);
            staffList.add(fullTimeStaff_obj);
            JOptionPane.showMessageDialog(frameFullTimeStaffHire,"Vacancy has been added");
        }else{// If the vacancy number is already in the list
            JOptionPane.showMessageDialog(frameFullTimeStaffHire, "The entered vacancy number is already in the list");
        }
    }
```

Figure 41: Screenshot for the correction of error 2

6.3. Error 3: Sematic error

The error has occurred because the data type of the variable “vacancyNumFound” is initialized as String but the value assigned is a Boolean value. The following error is corrected by changing the data type of “vacancyNumFound” to Boolean data type.

```
if (e.getSource() == terminateConfirmButton){
    int terminateVacancyNumberpt=getTerminateVacancyNumberTextField();
    String vacancyNumFound=false;
    if (terminateVacancyNumberpt != 0){
        // for-each loop
        for (StaffHire obj:staffList){
            if(obj.getVacancyNumber() == terminateVacancyNumberpt){
                vacancyNumFound=true;
```

Figure 42: Screenshot of the error 3

```
if (e.getSource() == terminateConfirmButton){
    int terminateVacancyNumberpt=getTerminateVacancyNumberTextField();
    boolean vacancyNumFound=false;
    if (terminateVacancyNumberpt != 0){
        // for-each loop
        for (StaffHire obj:staffList){
            if(obj.getVacancyNumber() == terminateVacancyNumberpt){
                vacancyNumFound=true;
```

Figure 43: Screenshot of the correction of error 3

7. Conclusion

In summary, a class INGNepal was added to the previous Coursework to add GUI (Graphical User Interface). The components of packages swing and awt were imported in INGNepal for the GUI elements. Multiple GUI approach was taken. Hence, Part Time Staff and Full Time Staff had separate GUI. Each of the GUI was stored in different methods. Different methods were assigned for accessing and storing the data entered by the user from the GUI application. The object containing values were stored in Array list. The Action Listener interface and Action Event were implemented in the class.
The actionPerformed() override method is used to add functionality to the buttons. Try and catch blocks were used to catch exceptions that might occur when incorrect data is entered by the user.

Many difficulties were faced doing this coursework. Adjusting to the online classes was a little troublesome. But due to the extraordinary effort from our tutorial teacher helped overcome every problem. Since being new to the topic, a lot of problems were faced when implementing the action listener and adding functionality to the buttons. But a lot research, videos uploaded by the teachers and feedback from our tutorial teacher helped a lot. Many concepts like down casting, action listener, etc were grasped and used in the coursework. The opportunity to design the GUI was quite exciting.

Draw.io was used for the class diagram and the coding was done in the text editor BlueJ. Though it was challenging, the coursework was completed and submitted on time. Many new concepts and topics were learned. Designing the GUI and developing the java program was fun and hectic at times as well. But overall it was a fun project to work on.

8. Bibliography

BlueJ. (2020) *About BlueJ* [Online]. Available from: https://www.bluej.org/about.html [Accessed 15 April 2020].

Guru99. (2020) *What is Software Testing_ Introduction, Definition, Basics & Types* [Online]. Available from: https://www.guru99.com/software-testing-introduction-importance.html [Accessed 11 April 2020].

The Economics Times. (2020) *What is Pseudocode_ Definition of Pseudocode, Pseudocode Meaning - The Economic Times* [Online]. Available from: https://economictimes.indiatimes.com/definition/pseudocode [Accessed 11 April 2020].

tutorialspoint. (2020) *UML - Class Diagram - Tutorialspoint* [Online]. Available from: https://www.tutorialspoint.com/uml/uml_class_diagram.htm [Accessed 11 April 2020].

9. Appendix 1

INGNepal:

```
/**
 * This application is an GUI aplication is designed to
 *  add vacancy and hire staff which may be a Full-time Staff or a Part-time Staff
 * @version 0.1
 * @author Pratik
 */

// Importing essential java packages
import javax.swing.JFrame;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.SwingConstants;
import javax.swing.JSeparator;
import javax.swing.JOptionPane;

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

import java.util.ArrayList;

public class INGNepal implements ActionListener{

    /**
     * MainFrame
     */

    private JFrame mainFrame;
    private JLabel INGnepalMainFrameLabel;
    private JLabel mainFrameInfoLabel;
    private JButton fullTimeStaffHireButton;
    private JButton partTimeStaffHireButton;
    private JButton displayMainFrameButton;
```

```
/**
 *  For Part Time Staff Hire
 * */

// JFrame
private JFrame framePartTimeStaffHire;

// JLabel
private JLabel titlePartTimeLabel;
private JLabel addVacancyPartTimeLabel;
private JLabel vacancyNumberPartTimeLabel;
private JLabel jobTypePartTimeLabel;
private JLabel designationPartTimeLabel;
private JLabel wagesPerHourPartTimeLabel;
private JLabel workingHoursPerDayPartTimeLabel;
private JLabel shiftPartTimeLabel;
private JLabel appointPartTimeStaffHireLabel;
private JLabel vacancyNumberAppointPartTimeLabel;
private JLabel staffNameAppointPartTimeLabel;
private JLabel joiningDateAppointPartTimeLabel;
private JLabel appointedByAppointPartTimeLabel;
private JLabel qualificationAppointPartTimeLabel;


// JButton
private JButton savePartTimeButton;
private JButton clearPartTimeButton;
private JButton displayPartTimeButton;
private JButton terminatePartTimeButton;
private JButton appointPartTimeButton;

// JComboBox
private JComboBox<String> jobTypePartTimeComboBox;
private JComboBox<String> shiftPartTimeComboBox;
private JComboBox<String> joiningDateYearAppointPartTimeComboBox;
private JComboBox<String> joiningDateMonthAppointPartTimeComboBox;
private JComboBox<String> joiningDateDayAppointPartTimeComboBox;
```

```
// JTextField
private JTextField vacancyNumberPartTimeTextField;
private JTextField designationPartTimeTextField;
private JTextField wagesPerHourPartTimeTextField;
private JTextField workingHoursPerDayPartTimeTextField;
private JTextField vacancyNumberAppointPartTimeTextField;
private JTextField staffNameAppointPartTimeTextField;
private JTextField qualificationAppointPartTimeTextField;
private JTextField appointedByAppointPartTimeTextField;

// JSeperator
private JSeparator lineA;

/**
 * Terminate Frame
*/
private JFrame frameTerminate;
private JLabel terminateLabel;
private JTextField terminateVacancyNumberTextField;
private JButton terminateCancelButton;
private JButton terminateConfirmButton;


/**
 *  For Full Time Staff Hire
 * */

// JFrame
private JFrame frameFullTimeStaffHire;

// JLabel
private JLabel titleFullTimeLabel;
private JLabel addVacancyFullTimeLabel;
private JLabel vacancyNumberFullTimeLabel;
private JLabel jobTypeFullTimeLabel;
```

```
private JLabel designationFullTimeLabel;
private JLabel salaryFullTimeLabel;
private JLabel workingHoursFullTimeLabel;
private JLabel appointFullTimeStaffHireLabel;
private JLabel vacancyNumberAppointFullTImeLabel;
private JLabel staffNameAppointFullTimeLabel;
private JLabel joiningDateAppointFullTimeLabel;
private JLabel qualificationAppointFullTimeLabel;
private JLabel appointedByAppointFullTimeLabel;

 // JButton
private JButton saveFullTimeButton;
private JButton clearFullTimeButton;
private JButton displayFullTimeButton;
private JButton appointFullTimeButton;

 // JComboBox
private JComboBox<String> jobTypeFullTimeComboBox;
private JComboBox<String> joiningDateYearAppointFullTimeComboBox;
private JComboBox<String> joiningDateMonthAppointFullTimeComboBox;
private JComboBox<String> joiningDateDayAppointFullTimeComboBox;

// JTextField
private JTextField vacancyNumberFullTimeTextField;
private JTextField designationFullTimeTextField;
private JTextField salaryFullTimeTextField;
private JTextField workingHoursFullTimeTextField;
private JTextField vacancyNumberAppointFullTimeTextField;
private JTextField staffNameAppointFullTimeTextField;
private JTextField qualificationAppointFullTimeTextField;
private JTextField appointedByAppointFullTimeTextField;

// JSeperator
private JSeparator lineB;

// Declaring arraylist
private ArrayList<StaffHire> staffList=new ArrayList<StaffHire>();
```

```
PartTimeStaffHire partTimeStaff_obj;
FullTimeStaffHire fullTimeStaff_obj;

// Constructor of the class
public INGNepal(){
    mainWindowGUI();
}

// Method containing GUI for the main window
public void mainWindowGUI(){
    // Initializing JFrame
    mainFrame = new JFrame("ING NEPAL");
    // Setting size of JFrame
    mainFrame.setSize(615,220);
    // Setting layout manager as null
    mainFrame.setLayout(null);
    // Setting frame as non resizable
    mainFrame.setResizable(false);

    // Initializing JLabel
    INGnepalMainFrameLabel = new JLabel("ING NEPAL");
    // Setting font for JLabel
    INGnepalMainFrameLabel.setFont(INGnepalMainFrameLabel.getFont().deriveFont(22f));
    // Setting bounds for JLabel
    INGnepalMainFrameLabel.setBounds(250,0,200,42);

    // Initializing JLabel
    mainFrameInfoLabel = new JLabel("Please click one of the buttons below :");
    // Setting bounds for JLabel
    mainFrameInfoLabel.setBounds(200,50,400,30);

    partTimeStaffHireButton = new JButton("Part Time Staff");
    partTimeStaffHireButton.setBounds(15,110,180,50);
    partTimeStaffHireButton.addActionListener(this);

    fullTimeStaffHireButton = new JButton("Full Time Staff");
```

```
        fullTimeStaffHireButton.setBounds(210,110,180,50);
        fullTimeStaffHireButton.addActionListener(this);

        displayMainFrameButton = new JButton("Display");
        displayMainFrameButton.setBounds(405,110,180,50);
        displayMainFrameButton.addActionListener(this);

        // Adding elements to the Frame
        mainFrame.add(INGnepalMainFrameLabel);
        mainFrame.add(mainFrameInfoLabel);
        mainFrame.add(partTimeStaffHireButton);
        mainFrame.add(fullTimeStaffHireButton);
        mainFrame.add(displayMainFrameButton);

        // Setting frame as visible
        mainFrame.setVisible(true);
    }

    // Method containing GUI for Part Time Staff
    public void PartTimeStaffHireGUI(){
        // Initializing JFrame
        framePartTimeStaffHire = new JFrame();
        framePartTimeStaffHire.setSize(610,630);
        framePartTimeStaffHire.setResizable(false);
        framePartTimeStaffHire.setLayout(null);
        framePartTimeStaffHire.setTitle("Part Time Staff");

        titlePartTimeLabel = new JLabel("For Part Time Staff");
        titlePartTimeLabel.setFont(titlePartTimeLabel.getFont().deriveFont(18f));
        titlePartTimeLabel.setBounds(215,0,400,30);

        addVacancyPartTimeLabel = new JLabel("Add Vacancy :");
        addVacancyPartTimeLabel.setFont(titlePartTimeLabel.getFont().deriveFont(17f));
        addVacancyPartTimeLabel.setBounds(40,40,400,30);

        vacancyNumberPartTimeLabel = new JLabel("Vacancy Number:");
        vacancyNumberPartTimeLabel.setBounds(40,90,120,30);
```

```
vacancyNumberPartTimeTextField = new JTextField();
vacancyNumberPartTimeTextField.setBounds(150,90,90,30);

jobTypePartTimeLabel = new JLabel("Job Type:");
jobTypePartTimeLabel.setBounds(370,90,60,30);

String jobTypeArray[] = {"Part Time","Full Time"} ;
jobTypePartTimeComboBox = new JComboBox<String>(jobTypeArray);
jobTypePartTimeComboBox.setBounds(440,90,120,30);

designationPartTimeLabel = new JLabel("Designation:");
designationPartTimeLabel.setBounds(40,140,75,30);

designationPartTimeTextField = new JTextField();
designationPartTimeTextField.setBounds(120,140,120,30);

wagesPerHourPartTimeLabel = new JLabel("Wages Per Hour:");
wagesPerHourPartTimeLabel.setBounds(370,140,100,30);

wagesPerHourPartTimeTextField = new JTextField();
wagesPerHourPartTimeTextField.setBounds(480,140,80,30);

workingHoursPerDayPartTimeLabel = new JLabel("Working Hours Per Day:");
workingHoursPerDayPartTimeLabel.setBounds(40,190,150,30);

workingHoursPerDayPartTimeTextField = new JTextField();
workingHoursPerDayPartTimeTextField.setBounds(180,190,60,30);

shiftPartTimeLabel = new JLabel("Shift:");
shiftPartTimeLabel.setBounds(370,190,60,30);

String shiftArray[] = {"Morning","Afternoon","Evening","Midnight"} ;
shiftPartTimeComboBox = new JComboBox<String>(shiftArray);
shiftPartTimeComboBox.setBounds(440,90,120,30);
shiftPartTimeComboBox.setBounds(440,190,120,30);
```

```
savePartTimeButton = new JButton("Save");
savePartTimeButton.setBounds(440,240,120,40);
savePartTimeButton.addActionListener(this);

// A horizontal line separating add vacancy and  appointing staff textfields
lineA = new JSeparator();
lineA.setOrientation(SwingConstants.HORIZONTAL);
lineA.setBounds(0,300,650,1);

appointPartTimeStaffHireLabel = new JLabel("Appoint Part Time Staff Hire :");
appointPartTimeStaffHireLabel.setBounds(40,320,300,30);
appointPartTimeStaffHireLabel.setFont(appointPartTimeStaffHireLabel.getFont().deriveFont(17f));

vacancyNumberAppointPartTimeLabel = new JLabel("Vacancy Number:");
vacancyNumberAppointPartTimeLabel.setBounds(40,370,120,30);

vacancyNumberAppointPartTimeTextField = new JTextField();
vacancyNumberAppointPartTimeTextField.setBounds(150,370,90,30);

staffNameAppointPartTimeLabel = new JLabel("Staff Name:");
staffNameAppointPartTimeLabel.setBounds(370,370,90,30);

staffNameAppointPartTimeTextField = new JTextField();
staffNameAppointPartTimeTextField.setBounds(445,370,115,30);

joiningDateAppointPartTimeLabel = new JLabel("Joining Date:");
joiningDateAppointPartTimeLabel.setBounds(40,420,100,30);

String joiningDateYearArray[] = {"YYYY","2020","2021","2022","2023","2024","2025"};
joiningDateYearAppointPartTimeComboBox = new JComboBox<String>(joiningDateYearArray);
joiningDateYearAppointPartTimeComboBox.setBounds(125, 420, 60, 30);

String
joiningDateMonthArray[]={"MM","Jan","Feb","Mar","Apr","May","Jun","July","Aug","Sep","Oct","Nov","Dec"}
;
joiningDateMonthAppointPartTimeComboBox = new JComboBox<String>(joiningDateMonthArray);
joiningDateMonthAppointPartTimeComboBox.setBounds(190,420,52,30);
```

```
        String[] joiningDateDayArray = {"DD","01", "02", "03", "04", "05", "06", "07", "08", "09", "10", "11",
"12", "13", "14", "15", "16", "17", "18", "19", "20", "21", "22", "23", "24", "25", "26", "27", "28", "29", "30",
"31", "32"};
        joiningDateDayAppointPartTimeComboBox = new JComboBox<String>(joiningDateDayArray);
        joiningDateDayAppointPartTimeComboBox.setBounds(247,420,46,30);

        qualificationAppointPartTimeLabel = new JLabel("Qualification:");
        qualificationAppointPartTimeLabel.setBounds(370,420,90,30);

        qualificationAppointPartTimeTextField = new JTextField();
        qualificationAppointPartTimeTextField.setBounds(450,420,110,30);

        appointedByAppointPartTimeLabel = new JLabel("Appointed By:");
        appointedByAppointPartTimeLabel.setBounds(40,470,100,30);

        appointedByAppointPartTimeTextField = new JTextField();
        appointedByAppointPartTimeTextField.setBounds(150,470,90,30);

        clearPartTimeButton = new JButton("Clear");
        clearPartTimeButton.setBounds(40,530,126,40);
        clearPartTimeButton.addActionListener(this);

        displayPartTimeButton = new JButton("Display");
        displayPartTimeButton.setBounds(176,530,126,40);
        displayPartTimeButton.addActionListener(this);

        terminatePartTimeButton = new JButton("Terminate");
        terminatePartTimeButton.setBounds(312,530,126,40);
        terminatePartTimeButton.addActionListener(this);

        appointPartTimeButton = new JButton("Appoint");
        appointPartTimeButton.setBounds(448,530,126,40);
        appointPartTimeButton.addActionListener(this);

        // Adding components to the frame
```

```
    framePartTimeStaffHire.add(titlePartTimeLabel);
    framePartTimeStaffHire.add(addVacancyPartTimeLabel);
    framePartTimeStaffHire.add(vacancyNumberPartTimeLabel);
    framePartTimeStaffHire.add(vacancyNumberPartTimeTextField);
    framePartTimeStaffHire.add(jobTypePartTimeLabel);
    framePartTimeStaffHire.add(jobTypePartTimeComboBox);
    framePartTimeStaffHire.add(designationPartTimeLabel);
    framePartTimeStaffHire.add(designationPartTimeTextField);
    framePartTimeStaffHire.add(wagesPerHourPartTimeLabel);
    framePartTimeStaffHire.add(wagesPerHourPartTimeTextField);
    framePartTimeStaffHire.add(workingHoursPerDayPartTimeLabel);
    framePartTimeStaffHire.add(workingHoursPerDayPartTimeTextField);
    framePartTimeStaffHire.add(shiftPartTimeLabel);
    framePartTimeStaffHire.add(shiftPartTimeComboBox);
    framePartTimeStaffHire.add(savePartTimeButton);
    framePartTimeStaffHire.add(lineA);
    framePartTimeStaffHire.add(appointPartTimeStaffHireLabel);
    framePartTimeStaffHire.add(vacancyNumberAppointPartTimeLabel);
    framePartTimeStaffHire.add(vacancyNumberAppointPartTimeTextField);
    framePartTimeStaffHire.add(staffNameAppointPartTimeLabel);
    framePartTimeStaffHire.add(staffNameAppointPartTimeTextField);
    framePartTimeStaffHire.add(joiningDateAppointPartTimeLabel);
    framePartTimeStaffHire.add(joiningDateYearAppointPartTimeComboBox);
    framePartTimeStaffHire.add(joiningDateMonthAppointPartTimeComboBox);
    framePartTimeStaffHire.add(joiningDateDayAppointPartTimeComboBox);
    framePartTimeStaffHire.add(qualificationAppointPartTimeLabel);
    framePartTimeStaffHire.add(qualificationAppointPartTimeTextField);
    framePartTimeStaffHire.add(appointedByAppointPartTimeLabel);
    framePartTimeStaffHire.add(appointedByAppointPartTimeTextField);
    framePartTimeStaffHire.add(clearPartTimeButton);
    framePartTimeStaffHire.add(displayPartTimeButton);
    framePartTimeStaffHire.add(terminatePartTimeButton);
    framePartTimeStaffHire.add(appointPartTimeButton);

    // Setting frame as visible
    framePartTimeStaffHire.setVisible(true);
  }
```

```
// method containing GUI for Full Time Staff
public void FullTimeStaffHireGUI(){
    frameFullTimeStaffHire = new JFrame();
    frameFullTimeStaffHire.setSize(610,630);
    frameFullTimeStaffHire.setResizable(false);
    frameFullTimeStaffHire.setLayout(null);
    frameFullTimeStaffHire.setTitle("Full Time Staff");

    titleFullTimeLabel = new JLabel("For Full Time Staff");
    titleFullTimeLabel.setFont(titleFullTimeLabel.getFont().deriveFont(18f));
    titleFullTimeLabel.setBounds(215,0,400,30);

    addVacancyFullTimeLabel = new JLabel("Add Vacancy :");
    addVacancyFullTimeLabel.setFont(titleFullTimeLabel.getFont().deriveFont(17f));
    addVacancyFullTimeLabel.setBounds(40,40,400,30);

    vacancyNumberFullTimeLabel = new JLabel("Vacancy Number:");
    vacancyNumberFullTimeLabel.setBounds(40,90,120,30);

    vacancyNumberFullTimeTextField = new JTextField();
    vacancyNumberFullTimeTextField.setBounds(150,90,90,30);

    jobTypeFullTimeLabel = new JLabel("Job Type:");
    jobTypeFullTimeLabel.setBounds(370,90,60,30);

    String jobTypeArray[] = {"Full Time","Part Time"} ;
    jobTypeFullTimeComboBox = new JComboBox<String>(jobTypeArray);
    jobTypeFullTimeComboBox.setBounds(440,90,120,30);

    designationFullTimeLabel = new JLabel("Designation:");
    designationFullTimeLabel.setBounds(40,140,90,30);

    designationFullTimeTextField = new JTextField();
    designationFullTimeTextField.setBounds(120,140,120,30);

    salaryFullTimeLabel = new JLabel("Salary:");
```

```
salaryFullTimeLabel.setBounds(370,140,50,30);

salaryFullTimeTextField = new JTextField();
salaryFullTimeTextField.setBounds(430,140,130,30);

workingHoursFullTimeLabel = new JLabel("Working Hours:");
workingHoursFullTimeLabel.setBounds(40,190,120,30);

workingHoursFullTimeTextField = new JTextField();
workingHoursFullTimeTextField.setBounds(170,190,70,30);

saveFullTimeButton = new JButton("Save");
saveFullTimeButton.setBounds(440,240,120,40);
saveFullTimeButton.addActionListener(this);

// Horizontal line
lineB = new JSeparator();
lineB.setOrientation(SwingConstants.HORIZONTAL);
lineB.setBounds(0,300,650,1);

appointFullTimeStaffHireLabel = new JLabel("Appoint Full Time Staff Hire :");
appointFullTimeStaffHireLabel.setBounds(40,320,300,30);
appointFullTimeStaffHireLabel.setFont(appointFullTimeStaffHireLabel.getFont().deriveFont(17f));

vacancyNumberAppointFullTImeLabel = new JLabel("Vacancy Number:");
vacancyNumberAppointFullTImeLabel.setBounds(40,370,120,30);

vacancyNumberAppointFullTimeTextField = new JTextField();
vacancyNumberAppointFullTimeTextField.setBounds(150,370,90,30);

staffNameAppointFullTimeLabel = new JLabel("Staff Name:");
staffNameAppointFullTimeLabel.setBounds(370,370,90,30);

staffNameAppointFullTimeTextField = new JTextField();
staffNameAppointFullTimeTextField.setBounds(445,370,115,30);

joiningDateAppointFullTimeLabel = new JLabel("Joining Date:");
```

```
        joiningDateAppointFullTimeLabel.setBounds(40,420,100,30);

        String joiningDateYearArray[] = {"YYYY","2020","2021","2022","2023","2024","2025"};
        joiningDateYearAppointFullTimeComboBox = new JComboBox<String>(joiningDateYearArray);
        joiningDateYearAppointFullTimeComboBox.setBounds(125, 420, 60, 30);

        String
joiningDateMonthArray[]={"MM","Jan","Feb","Mar","Apr","May","Jun","July","Aug","Sep","Oct","Nov","Dec"}
;
        joiningDateMonthAppointFullTimeComboBox = new JComboBox<String>(joiningDateMonthArray);
        joiningDateMonthAppointFullTimeComboBox.setBounds(190,420,52,30);


        String[] joiningDateDayArray = {"DD","01", "02", "03", "04", "05", "06", "07", "08", "09", "10", "11",
"12", "13", "14", "15", "16", "17", "18", "19", "20", "21", "22", "23", "24", "25", "26", "27", "28", "29", "30",
"31", "32"};
        joiningDateDayAppointFullTimeComboBox = new JComboBox<String>(joiningDateDayArray);
        joiningDateDayAppointFullTimeComboBox.setBounds(247,420,46,30);

        qualificationAppointFullTimeLabel = new JLabel("Qualification:");
        qualificationAppointFullTimeLabel.setBounds(370,420,90,30);

        qualificationAppointFullTimeTextField = new JTextField();
        qualificationAppointFullTimeTextField.setBounds(450,420,110,30);

        appointedByAppointFullTimeLabel = new JLabel("Appointed By:");
        appointedByAppointFullTimeLabel.setBounds(40,470,90,30);

        appointedByAppointFullTimeTextField = new JTextField();
        appointedByAppointFullTimeTextField.setBounds(130,470,110,30);

        clearFullTimeButton = new JButton("Clear");
        clearFullTimeButton.setBounds(40,530,160,40);
        clearFullTimeButton.addActionListener(this);

        displayFullTimeButton = new JButton("Display");
        displayFullTimeButton.setBounds(220,530,160,40);
```

```
displayFullTimeButton.addActionListener(this);

appointFullTimeButton = new JButton("Appoint");
appointFullTimeButton.setBounds(400,530,160,40);
appointFullTimeButton.addActionListener(this);

// Adding components to the frame
frameFullTimeStaffHire.add(titleFullTimeLabel);
frameFullTimeStaffHire.add(addVacancyFullTimeLabel);
frameFullTimeStaffHire.add(vacancyNumberFullTimeLabel);
frameFullTimeStaffHire.add(vacancyNumberFullTimeTextField);
frameFullTimeStaffHire.add(jobTypeFullTimeLabel);
frameFullTimeStaffHire.add(jobTypeFullTimeComboBox);
frameFullTimeStaffHire.add(designationFullTimeLabel);
frameFullTimeStaffHire.add(designationFullTimeTextField);
frameFullTimeStaffHire.add(salaryFullTimeLabel);
frameFullTimeStaffHire.add(salaryFullTimeTextField);
frameFullTimeStaffHire.add(workingHoursFullTimeLabel);
frameFullTimeStaffHire.add(workingHoursFullTimeTextField);
frameFullTimeStaffHire.add(saveFullTimeButton);
frameFullTimeStaffHire.add(lineB);
frameFullTimeStaffHire.add(appointFullTimeStaffHireLabel);
frameFullTimeStaffHire.add(vacancyNumberAppointFullTImeLabel);
frameFullTimeStaffHire.add(vacancyNumberAppointFullTimeTextField);
frameFullTimeStaffHire.add(staffNameAppointFullTimeLabel);
frameFullTimeStaffHire.add(staffNameAppointFullTimeTextField);
frameFullTimeStaffHire.add(joiningDateAppointFullTimeLabel);
frameFullTimeStaffHire.add(joiningDateYearAppointFullTimeComboBox);
frameFullTimeStaffHire.add(joiningDateMonthAppointFullTimeComboBox);
frameFullTimeStaffHire.add(joiningDateDayAppointFullTimeComboBox);
frameFullTimeStaffHire.add(qualificationAppointFullTimeLabel);
frameFullTimeStaffHire.add(qualificationAppointFullTimeTextField);
frameFullTimeStaffHire.add(appointedByAppointFullTimeLabel);
frameFullTimeStaffHire.add(appointedByAppointFullTimeTextField);
frameFullTimeStaffHire.add(clearFullTimeButton);
frameFullTimeStaffHire.add(displayFullTimeButton);
frameFullTimeStaffHire.add(appointFullTimeButton);
```

```
        // frameFullTimeStaffHire.add();
        frameFullTimeStaffHire.setVisible(true);
    }

    // GUI for terminating Staff
    public void terminateGUI(){
        frameTerminate = new JFrame("Terminate Part Time Staff");
        frameTerminate.setSize(530,230);
        frameTerminate.setLayout(null);
        frameTerminate.setResizable(false);

        terminateLabel = new JLabel("Enter the vacancy number of the part time staff you want to terminate
:");
        terminateLabel.setFont(terminateLabel.getFont().deriveFont(14f));
        terminateLabel.setBounds(15,0,600,50);

        terminateVacancyNumberTextField = new JTextField();
        terminateVacancyNumberTextField.setBounds(15,60,480,40);

        terminateConfirmButton = new JButton("Terminate");
        terminateConfirmButton.setBounds(272,120,200,50);
        terminateConfirmButton.addActionListener(this);

        terminateCancelButton = new JButton("Cancel");
        terminateCancelButton.setBounds(57,120,200,50);
        terminateCancelButton.addActionListener(this);

        // Adding components to the frame
        frameTerminate.add(terminateLabel);
        frameTerminate.add(terminateVacancyNumberTextField);
        frameTerminate.add(terminateConfirmButton);
        frameTerminate.add(terminateCancelButton);

        // Setting frame as visible
        frameTerminate.setVisible(true);
    }
```

```
// Method for clearing textfields  and resetting combobox for Part time GUI
public void resettingFieldsPartTime(){
    // Clearing JTextField
    vacancyNumberPartTimeTextField.setText("");
    designationPartTimeTextField.setText("");
    wagesPerHourPartTimeTextField.setText("");
    workingHoursPerDayPartTimeTextField.setText("");
    vacancyNumberAppointPartTimeTextField.setText("");
    staffNameAppointPartTimeTextField.setText("");
    qualificationAppointPartTimeTextField.setText("");
    appointedByAppointPartTimeTextField.setText("");

    // Resetting the JComboBox
    jobTypePartTimeComboBox.setSelectedIndex(0);
    shiftPartTimeComboBox.setSelectedIndex(0);
    joiningDateYearAppointPartTimeComboBox.setSelectedIndex(0);
    joiningDateMonthAppointPartTimeComboBox.setSelectedIndex(0);
    joiningDateDayAppointPartTimeComboBox.setSelectedIndex(0);
}

// Method for clearing textfields  and resetting combobox for Full time GUI
public void resettingFieldsFullTime(){
    // Clearing JTextField
    vacancyNumberFullTimeTextField.setText("");
    designationFullTimeTextField.setText("");
    workingHoursFullTimeTextField.setText("");
    salaryFullTimeTextField.setText("");
    vacancyNumberAppointFullTimeTextField.setText("");
    staffNameAppointFullTimeTextField.setText("");
    qualificationAppointFullTimeTextField.setText("");
    appointedByAppointFullTimeTextField.setText("");

    // Resetting JComboBox
    jobTypeFullTimeComboBox.setSelectedIndex(0);
    joiningDateYearAppointFullTimeComboBox.setSelectedIndex(0);
    joiningDateMonthAppointFullTimeComboBox.setSelectedIndex(0);
```

```
        joiningDateDayAppointFullTimeComboBox.setSelectedIndex(0);
    }

    // Exception Handling for Part Time

    /**
     * @return the vacancyNumberPartTimeTextField
     *  it takes value from the textfield and returns an int vacancy number.
     */
    public int getVacancyNumberPartTimeTextField() {
        int vacancyNumPartTime=0;
        if (vacancyNumberPartTimeTextField.getText().equalsIgnoreCase("")){
            JOptionPane.showMessageDialog(framePartTimeStaffHire,"Please enter value in Vacancy
Number Text Field" );
        }else{
            try{
                vacancyNumPartTime = Integer.parseInt(vacancyNumberPartTimeTextField.getText());
                if (vacancyNumPartTime == 0){
                    JOptionPane.showMessageDialog(framePartTimeStaffHire,"The value of Vacancy Number
cannot be 0" );
                }
                else if(vacancyNumPartTime <0){
                    JOptionPane.showMessageDialog(framePartTimeStaffHire,"The value of Vacancy Number
cannot be negative" );
                }
            }
            catch (NumberFormatException e){
                JOptionPane.showMessageDialog(framePartTimeStaffHire,"Please enter whole numbers only
in Vacancy Number Text Field" );
            }
        }
        return vacancyNumPartTime;
    }

    /**
     * @return the designationPartTimeTextField
     * it takes value from the textfield and returns a string designation.
```

```
     */
    public String getDesignationPartTimeTextField() {
        String designationPartTime=designationPartTimeTextField.getText();
        if (designationPartTime.equalsIgnoreCase("")){
            JOptionPane.showMessageDialog(framePartTimeStaffHire,"Please enter value in Designation
Text Field");
        }
        return designationPartTime;
    }

    /**
     * @return the wagesPerHourPartTimeTextField
     * it takes value from the textfield and returns an int wages per hour.
     */
    public int getWagesPerHourPartTimeTextField() {
        int wagesPerHourPartTime = 0;
        if (wagesPerHourPartTimeTextField.getText().equalsIgnoreCase("")){
            JOptionPane.showMessageDialog(framePartTimeStaffHire,"Please enter value in Wages Per
Hour Text Field" );
        }else{
            try{
                wagesPerHourPartTime  = Integer.parseInt(wagesPerHourPartTimeTextField.getText());
                if (wagesPerHourPartTime == 0){
                    JOptionPane.showMessageDialog(framePartTimeStaffHire,"The value of Wages Per Hour
cannot be 0" );
                }
                else if(wagesPerHourPartTime <0){
                    JOptionPane.showMessageDialog(framePartTimeStaffHire,"The value of Wages Per Hour
cannot be negative" );
                }
            }
            catch (NumberFormatException e){
                JOptionPane.showMessageDialog(framePartTimeStaffHire,"Please enter whole numbers only
in Wages Per Hour Text Field" );
            }
        }
        return wagesPerHourPartTime;
```

```
    }

    /**
     * @return the workingHoursPerDayPartTimeTextField
     * it takes value from the textfield and returns an int working hours.
     */
    public int getWorkingHoursPerDayPartTimeTextField() {
        int workingHoursPerDayPartTime = 0;
        if (workingHoursPerDayPartTimeTextField.getText().equalsIgnoreCase("")){
            JOptionPane.showMessageDialog(framePartTimeStaffHire,"Please enter value in Working Hours
Per Day Text Field" );
        }else{
            try{
                workingHoursPerDayPartTime  =
Integer.parseInt(workingHoursPerDayPartTimeTextField.getText());
                if (workingHoursPerDayPartTime == 0){
                    JOptionPane.showMessageDialog(framePartTimeStaffHire,"The value of Working Hours Per
Day cannot be 0" );
                }
                else if(workingHoursPerDayPartTime <0){
                    JOptionPane.showMessageDialog(framePartTimeStaffHire,"The value of Working Hours Per
Day cannot be negative" );
                }
            }
            catch (NumberFormatException e){
                JOptionPane.showMessageDialog(framePartTimeStaffHire,"Please enter whole numbers only
in Working Hours Per Day Text Field" );
            }
        }
        return workingHoursPerDayPartTime;
    }

    /**
     * @return the vacancyNumberAppointPartTimeTextField
     * it takes value from the textfield and returns an int vacancy number.
     */
    public int getVacancyNumberAppointPartTimeTextField() {
```

```
        int vacancyNumAppointPartTime=0;
        if (vacancyNumberAppointPartTimeTextField.getText().equalsIgnoreCase("")){
            JOptionPane.showMessageDialog(framePartTimeStaffHire,"Please enter value in Vacancy
Number Text Field" );
        }else{
            try{
                vacancyNumAppointPartTime =
Integer.parseInt(vacancyNumberAppointPartTimeTextField.getText());
                if (vacancyNumAppointPartTime == 0){
                    JOptionPane.showMessageDialog(framePartTimeStaffHire,"The value of Vacancy Number
cannot be 0" );
                }
                else if(vacancyNumAppointPartTime <0){
                    JOptionPane.showMessageDialog(framePartTimeStaffHire,"The value of Vacancy Number
cannot be negative" );
                }
            }
            catch (NumberFormatException e){
                JOptionPane.showMessageDialog(framePartTimeStaffHire,"Please enter whole numbers only
in Vacancy Number Text Field" );
            }
        }
        return vacancyNumAppointPartTime;
    }

    /**
     * @return the staffNameAppointPartTimeTextField
     * it takes value from the textfield and returns a string staff name.
     */
    public String getStaffNameAppointPartTimeTextField() {
        String StaffNamePartTime=staffNameAppointPartTimeTextField.getText();
        if (StaffNamePartTime.equalsIgnoreCase("")){
            JOptionPane.showMessageDialog(framePartTimeStaffHire,"Please enter value in Staff Name
Text Field");
        }
        return StaffNamePartTime;
    }
```

```
    /**
     * @return the qualificationAppointPartTimeTextField
     * it takes value from the textfield and returns a string qualification.
     */
    public String getQualificationAppointPartTimeTextField() {
        String QualicationPartTime=qualificationAppointPartTimeTextField.getText();
        if (QualicationPartTime.equalsIgnoreCase("")){
            JOptionPane.showMessageDialog(framePartTimeStaffHire,"Please enter value in Qualification
Text Field");
        }
        return QualicationPartTime;
    }

    /**
     * @return the appointedByAppointPartTimeTextField
     * it takes value from the textfield and returns a string appointed by.
     */
    public String getAppointedByAppointPartTimeTextField() {
        String appointedByPartTime= appointedByAppointPartTimeTextField.getText();
        if (appointedByPartTime.equalsIgnoreCase("")){
            JOptionPane.showMessageDialog(framePartTimeStaffHire,"Please enter value in Appointed By
Text Field");
        }
        return appointedByPartTime;
    }

    /**
     * @return the jobTypePartTimeComboBox
     * it takes value from the combobox and returns a string job type.
     */
    public String getJobTypePartTimeComboBox() {
        String jobTypePartTime =(String)jobTypePartTimeComboBox.getSelectedItem();
        if (jobTypePartTime=="Full Time"){
            JOptionPane.showMessageDialog(framePartTimeStaffHire,"Please use the Full Time Staff
application to add vacancy for Full Time Staff");
        }
```

```
        return jobTypePartTime;
    }

    /**
     * @return the shiftPartTimeComboBox
     * it takes value from the combobox and returns a string shift.
     */
    public String getShiftPartTimeComboBox() {
        String shiftPartTime = (String) shiftPartTimeComboBox.getSelectedItem();
        return shiftPartTime;
    }

    /**
     * @return the joiningDateAppointPartTimeLabel
     * It joins the values from comboboxes and returns a string date.
     */
    public String getJoiningDateAppointPartTime() {
        String joiningDateYearAppointPartTime=(String)
joiningDateYearAppointPartTimeComboBox.getSelectedItem();
        String joiningDateMonthAppointPartTime=(String)
joiningDateMonthAppointPartTimeComboBox.getSelectedItem();
        String joiningDateDayAppointPartTime= (String)
joiningDateDayAppointPartTimeComboBox.getSelectedItem();

        String joiningDate="";

        if (joiningDateYearAppointPartTime == ("YYYY") || joiningDateMonthAppointPartTime == ("MM") ||
joiningDateDayAppointPartTime == ("DD") ){
            JOptionPane.showMessageDialog(framePartTimeStaffHire,"Please choose the Year, Month and
Day of the joining date");
        }else{
            joiningDate = joiningDateYearAppointPartTime + " " + joiningDateMonthAppointPartTime + " " +
joiningDateDayAppointPartTime;
        }

        return joiningDate;
    }
```

```
    // Exception Handling For Full Time Staff

    /**
     * @return the vacancyNumberFullTimeTextField
     * It takes the value from the textfield and returns an int vacancy number.
     */
    public int getVacancyNumberFullTimeTextField() {
        int vacancyNumFullTime=0;
        if (vacancyNumberFullTimeTextField.getText().equalsIgnoreCase("")){
            JOptionPane.showMessageDialog(frameFullTimeStaffHire,"Please enter value in Vacancy
Number Text Field" );
        }else{
            try{
                vacancyNumFullTime = Integer.parseInt(vacancyNumberFullTimeTextField.getText());
                if (vacancyNumFullTime == 0){
                    JOptionPane.showMessageDialog(frameFullTimeStaffHire,"The value of Vacancy Number
cannot be 0" );
                }
                else if(vacancyNumFullTime <0){
                    JOptionPane.showMessageDialog(frameFullTimeStaffHire,"The value of Vacancy Number
cannot be negative" );
                }
            }
            catch (NumberFormatException e){
                JOptionPane.showMessageDialog(frameFullTimeStaffHire,"Please enter whole numbers only
in Vacancy Number Text Field" );
            }
        }
        return vacancyNumFullTime;
    }

    /**
     * @return the designationFullTimeTextField
     * It takes the value from the textfield and returns a string designation.
     */
    public String getDesignationFullTimeTextField() {
```

```
        String DesignationFullTime=designationFullTimeTextField.getText();
        if (DesignationFullTime.equalsIgnoreCase("")){
            JOptionPane.showMessageDialog(frameFullTimeStaffHire,"Please enter value in Designation
Text Field");
        }
        return DesignationFullTime;
    }

    /**
     * @return the workingHoursFullTimeTextField
     * It takes the value from textfield and returns an int working hour.
     */
    public int getWorkingHoursFullTimeTextField() {
        int workingHoursFullTime = 0;
        if (workingHoursFullTimeTextField.getText().equalsIgnoreCase("")){
            JOptionPane.showMessageDialog(frameFullTimeStaffHire,"Please enter value in Working Hours
Text Field" );
        }else{
            try{
                workingHoursFullTime  = Integer.parseInt(workingHoursFullTimeTextField.getText());
                if (workingHoursFullTime == 0){
                    JOptionPane.showMessageDialog(frameFullTimeStaffHire,"The value of Working Hours
cannot be 0" );
                }
                else if(workingHoursFullTime <0){
                    JOptionPane.showMessageDialog(frameFullTimeStaffHire,"The value of Working Hours
cannot be negative" );
                }
            }
            catch (NumberFormatException e){
                JOptionPane.showMessageDialog(frameFullTimeStaffHire,"Please enter whole numbers only
in Working Hours Text Field" );
            }
        }
        return workingHoursFullTime;
    }
```

```
    /**
     * @return the salaryFullTimeTextField
     * It takes the value from salary and returns an int salary.
     */
    public int getSalaryFullTimeTextField() {
        int salaryFullTime = 0;
        if (salaryFullTimeTextField.getText().equalsIgnoreCase("")){
            JOptionPane.showMessageDialog(frameFullTimeStaffHire,"Please enter value in Salary Text
Field" );
        }else{
            try{
                salaryFullTime  = Integer.parseInt(salaryFullTimeTextField.getText());
                if (salaryFullTime == 0){
                    JOptionPane.showMessageDialog(frameFullTimeStaffHire,"The value of Salary cannot be 0"
);
                }
                else if(salaryFullTime <0){
                    JOptionPane.showMessageDialog(frameFullTimeStaffHire,"The value of Salary cannot be
negative" );
                }
            }
            catch (NumberFormatException e){
                JOptionPane.showMessageDialog(frameFullTimeStaffHire,"Please enter whole numbers only
in Salary Text Field" );
            }
        }
        return salaryFullTime;
    }

     /**
     * @return the jobTypeFullTimeComboBox
     * It takes the value from combobox and returns a string job type.
     */
    public String getJobTypeFullTimeComboBox(){
        String jobTypeFullTime =(String)jobTypeFullTimeComboBox.getSelectedItem();
        if (jobTypeFullTime=="Part Time"){
```

```
            JOptionPane.showMessageDialog(frameFullTimeStaffHire, "Please use the Part Time Staff
application to add vacancy for Part Time Staff");
        }
        return jobTypeFullTime;
    }

    /**
     * @return the vacancyNumberAppointFullTimeTextField
     * It takes the value from textfield and returns an int vacancy number.
     */
    public int getVacancyNumberAppointFullTimeTextField() {
        int vacancyNumAppointFullTime=0;
        if (vacancyNumberAppointFullTimeTextField.getText().equalsIgnoreCase("")){
            JOptionPane.showMessageDialog(frameFullTimeStaffHire,"Please enter value in Vacancy
Number Text Field" );
        }else{
            try{
                vacancyNumAppointFullTime =
Integer.parseInt(vacancyNumberAppointFullTimeTextField.getText());
                if (vacancyNumAppointFullTime == 0){
                    JOptionPane.showMessageDialog(frameFullTimeStaffHire,"The value of Vacancy Number
cannot be 0" );
                }
                else if(vacancyNumAppointFullTime <0){
                    JOptionPane.showMessageDialog(frameFullTimeStaffHire,"The value of Vacancy Number
cannot be negative" );
                }
            }
            catch (NumberFormatException e){
                JOptionPane.showMessageDialog(frameFullTimeStaffHire,"Please enter whole numbers only
in Vacancy Number Text Field" );
            }
        }
        return vacancyNumAppointFullTime;
    }

    /**
```

```
     * @return the staffNameAppointFullTimeTextField
     * It takes the value from the textfield and returns a string staff name.
     */
    public String getStaffNameAppointFullTimeTextField() {
        String StaffNameFullTime=staffNameAppointFullTimeTextField.getText();
        if (StaffNameFullTime.equalsIgnoreCase("")){
            JOptionPane.showMessageDialog(frameFullTimeStaffHire,"Please enter value in Staff Name Text
Field");
        }
        return StaffNameFullTime;
    }

    /**
     * @return the qualificationAppointFullTimeTextField
     * It takes the value from textfield and returns a string qualification.
     */
    public String getQualificationAppointFullTimeTextField() {
        String QualicationFullTime=qualificationAppointFullTimeTextField.getText();
        if (QualicationFullTime.equalsIgnoreCase("")){
            JOptionPane.showMessageDialog(frameFullTimeStaffHire,"Please enter value in Qualification
Text Field");
        }
        return QualicationFullTime;
    }

    /**
     * @return the appointedbyAppointFullTimeTextField
     * It takes the value from textfield and returns a string appointed by.
     */
    public String getAppointedbyAppointFullTimeTextField() {
        String AppointedByFullTime= appointedByAppointFullTimeTextField.getText();
        if (AppointedByFullTime.equalsIgnoreCase("")){
            JOptionPane.showMessageDialog(frameFullTimeStaffHire,"Please enter value in Appointed By
Text Field");
        }
        return AppointedByFullTime;
    }
```

```
    /**
     * @return the joiningDateAppointFullTimeLabel
     * It joins the values from comboboxes and returns a string date.
     */
    public String getJoiningDateAppointFullTime() {
        String joiningDateYearAppointFullTime=(String)
joiningDateYearAppointFullTimeComboBox.getSelectedItem();
        String joiningDateMonthAppointFullTime=(String)
joiningDateMonthAppointFullTimeComboBox.getSelectedItem();
        String joiningDateDayAppointFullTime= (String)
joiningDateDayAppointFullTimeComboBox.getSelectedItem();

        String joiningDate="";

        if (joiningDateYearAppointFullTime == ("YYYY") || joiningDateMonthAppointFullTime == ("MM") ||
joiningDateDayAppointFullTime == ("DD") ){
            JOptionPane.showMessageDialog(frameFullTimeStaffHire,"Please choose the Year, Month and
Day of the joining date");
        }else{
            joiningDate = joiningDateYearAppointFullTime + " " +joiningDateMonthAppointFullTime + " " +
joiningDateDayAppointFullTime;
        }
        return joiningDate;
    }

    /**
     * Exception Handling for Terminate GUI
     */

     /*
      * @return the terminateVacancyNumberTextField
      * It takes the value from the textfield and returns an int vacancy number.
      */
     public int getTerminateVacancyNumberTextField() {
        int terminateVacancyNumberPartTime=0;
        if (terminateVacancyNumberTextField.getText().equalsIgnoreCase("")){
```

```
            JOptionPane.showMessageDialog(frameTerminate,"Please enter value in Vacancy Number Text
Field" );
        }else{
            try{
                terminateVacancyNumberPartTime =
Integer.parseInt(terminateVacancyNumberTextField.getText());
                if (terminateVacancyNumberPartTime == 0){
                    JOptionPane.showMessageDialog(frameTerminate,"The value of Vacancy Number cannot
be 0" );
                }
                else if(terminateVacancyNumberPartTime <0){
                    JOptionPane.showMessageDialog(frameTerminate,"The value of Vacancy Number cannot
be negative" );
                }
            }
            catch (NumberFormatException e){
                JOptionPane.showMessageDialog(frameTerminate,"Please enter whole numbers only in
Vacancy Number Text Field" );
            }
        }
        return terminateVacancyNumberPartTime;
    }


    //An actionPerformed method performs certain action when a certain button is clicked
    @Override
    public void actionPerformed(ActionEvent e){
        // Adding functionality to the clear button of Part Time Staff Hire GUI
        if (e.getSource() == clearPartTimeButton){
            resettingFieldsPartTime();
        }

        // Adding functionality to the clear button of Full Time Staff Hire GUI
        if (e.getSource() == clearFullTimeButton){
            resettingFieldsFullTime();
        }
```

```
    // Calling the method containing the Part Time Staff Hire GUI when the Part Time Staff Button is
clicked
    if (e.getSource() == partTimeStaffHireButton){
        PartTimeStaffHireGUI();
    }

    // Calling the method containing the Full Time Staff Hire GUI when the Full Time Staff Button is
clicked
    if (e.getSource() == fullTimeStaffHireButton){
        FullTimeStaffHireGUI();
    }
    // Calling the method containing the Terminate Staff GUI when the Terminate Button is clicked
    if (e.getSource() == terminatePartTimeButton){
        terminateGUI();
    }

    // Closes the frame when cancel button is clicked
    if (e.getSource() == terminateCancelButton){
        frameTerminate.dispose();// Closes the frame
    }

    // Adding functionality to the display button of the main frame
    if (e.getSource() == displayMainFrameButton){
        boolean recordFound=false;
        // Iterates wthin the arraylist
        for (StaffHire obj:staffList){
            if (obj instanceof PartTimeStaffHire){// Checks whether the object is an instance of
PartTimeStaffHire subclass.
                partTimeStaff_obj= (PartTimeStaffHire)obj;
                // Calling the display method from the PartTimeStaffHire class
                partTimeStaff_obj.display();
                recordFound=true;
            }else{
                fullTimeStaff_obj= (FullTimeStaffHire)obj;
                // Calling the display method from the FullTimeStaffHire class
                fullTimeStaff_obj.display();
                recordFound=true;
```

```
            }
        }
        if (recordFound==false){//If no records are found
            JOptionPane.showMessageDialog(mainFrame,"No records found");
        }
    }

    // Adding functionality to the display button of the Part Time Staff Frame
    if (e.getSource() == displayPartTimeButton){
        boolean recordFound=false;
        // Iterates wthin the arraylist
        for (StaffHire obj:staffList){
            if (obj instanceof PartTimeStaffHire){// Checks whether the object is an instance of
PartTimeStaffHire subclass.
                partTimeStaff_obj= (PartTimeStaffHire)obj;
                // Calling the display method from the PartTimeStaffHire subclass
                partTimeStaff_obj.display();
                recordFound=true;
            }
        }
        if (recordFound==false){// If no records are found
            JOptionPane.showMessageDialog(framePartTimeStaffHire,"No records found of Part Time
Staffs");
        }
    }

    // Adding functionality to the display button of the Full Time Staff Frame
    if (e.getSource() == displayFullTimeButton){
        boolean recordFound=false;
        // Iterates wthin the arraylist
        for (StaffHire obj:staffList){
            if (obj instanceof FullTimeStaffHire){// Checks whether the object is an instance of
FullTimeStaffHire subclass.
                // Calling the display method from the FullTimeStaffHire class
                fullTimeStaff_obj = (FullTimeStaffHire)obj;
                fullTimeStaff_obj.display();
                recordFound=true;
```

```
            }
        }
        if (recordFound==false){// If no records are found
            JOptionPane.showMessageDialog(frameFullTimeStaffHire,"No records found of Full Time
Staffs");
        }
    }

    // Adding functionality to the confirm button of the Terminate GUI
    if (e.getSource() == terminateConfirmButton){
        int terminateVacancyNumberpt=getTerminateVacancyNumberTextField();
        boolean vacancyNumFound=false;
        if (terminateVacancyNumberpt > 0){
            // Iterates wthin the arraylist
            for (StaffHire obj:staffList){
                if(obj.getVacancyNumber() == terminateVacancyNumberpt){
                    vacancyNumFound=true;
                    if (obj instanceof PartTimeStaffHire){// Checks whether the object is an instance of
PartTimeStaffHire subclass.
                        partTimeStaff_obj = (PartTimeStaffHire)obj;
                        if (partTimeStaff_obj.getTerminated()==false && partTimeStaff_obj.getJoined()==true){
                            partTimeStaff_obj.terminate();
                            JOptionPane.showMessageDialog(frameTerminate, "The Staff has been
Terminated");
                            break;
                        }else if(partTimeStaff_obj.getTerminated()==true &&
partTimeStaff_obj.getJoined()==false){
                            JOptionPane.showMessageDialog(frameTerminate,"The staff has already been
terminated");
                            break;
                        }else{
                            JOptionPane.showMessageDialog(frameTerminate,"The Staff hasn't been hired.
Hence, there is no staff to terminate.");
                            break;
                        }
                    }else{
```

```
                    JOptionPane.showMessageDialog(frameTerminate,"Vacancy no."
+terminateVacancyNumberpt+" does not belong to anybody of the Part Time Staffs");
                }
            }
        }
        if (vacancyNumFound==false){// If no records are found
            JOptionPane.showMessageDialog(frameTerminate, "The inserted vacancy number is
invalid");
        }
    }
}

// Adding functionality to the save button of the Part Time Staff GUI
if (e.getSource() == savePartTimeButton){
    String jobTypept=getJobTypePartTimeComboBox();
    int vacancyNumberpt=getVacancyNumberPartTimeTextField();
    String designationpt=getDesignationPartTimeTextField();
    int wagesPerHourpt=getWagesPerHourPartTimeTextField();
    int workingHoursPerDaypt=getWorkingHoursPerDayPartTimeTextField();
    String shiftpt=getShiftPartTimeComboBox();

    boolean duplicateVacancyNum=false;
    if (vacancyNumberpt >0 && designationpt != "" && wagesPerHourpt >0 &&
workingHoursPerDaypt >0 && jobTypept=="Part Time") {
        // Iterates wthin the arraylist
        for (StaffHire obj:staffList){
            if(obj.getVacancyNumber() == vacancyNumberpt){
                duplicateVacancyNum=true;
                break;
            }
        }
        if (duplicateVacancyNum==false){
            partTimeStaff_obj= new
PartTimeStaffHire(designationpt,jobTypept,vacancyNumberpt,workingHoursPerDaypt,wagesPerHourpt,s
hiftpt);
            staffList.add(partTimeStaff_obj);
            JOptionPane.showMessageDialog(framePartTimeStaffHire,"Vacancy has been added");
```

```
            }else{// If the entered vacancy number is already in the arraylist
                JOptionPane.showMessageDialog(framePartTimeStaffHire, "The entered vacancy number is
already in the list");
            }
        }
    }

    // Adding functionality to the Appoint button of the Part Time Staff GUI
    if (e.getSource() == appointPartTimeButton){
        boolean vacancyNumFound=false;

        int vacancyNumberAppointpt=getVacancyNumberAppointPartTimeTextField();
        String staffNamept=getStaffNameAppointPartTimeTextField();
        String joiningDatept=getJoiningDateAppointPartTime();
        String qualificationpt=getQualificationAppointPartTimeTextField();
        String appointedBypt=getAppointedByAppointPartTimeTextField();

        if (vacancyNumberAppointpt >0 && staffNamept!="" && qualificationpt !="" && appointedBypt !=""
&& joiningDatept !="" ){
            // Iterates wthin the arraylist
            for (StaffHire obj:staffList){
                if(obj.getVacancyNumber() == vacancyNumberAppointpt){
                    vacancyNumFound=true;
                    if (obj instanceof PartTimeStaffHire){// Checks whether the object is an instance of
PartTimeStaffHire subclass.
                        partTimeStaff_obj = (PartTimeStaffHire)obj;
                        if (partTimeStaff_obj.getJoined()==false){
                            partTimeStaff_obj.hiring(staffNamept, joiningDatept, qualificationpt, appointedBypt);
                            JOptionPane.showMessageDialog(framePartTimeStaffHire, "Staff Hired");
                            break;
                        }else if(partTimeStaff_obj.getJoined()==true){
                            JOptionPane.showMessageDialog(framePartTimeStaffHire,"A staff has already been
hired to fill this vacancy no.");
                            break;
                        }
                    }else{//If the entered vacancy number is for the vacancy number for Full Time Staff
```

```
                    JOptionPane.showMessageDialog(framePartTimeStaffHire,"Vacancy no.
"+vacancyNumberAppointpt+" is not for Part Time Staff Hire");
                }
            }
        }
        if (vacancyNumFound==false){// If no records are found
            JOptionPane.showMessageDialog(framePartTimeStaffHire, "The inserted vacancy number is
invalid");
        }
    }
}

// Adding functionality to the save button of the Full Time Staff GUI
if (e.getSource() == saveFullTimeButton){
    boolean duplicateVacancyNum=false;

    String jobTypeft=getJobTypeFullTimeComboBox();
    int vacancyNumberft=getVacancyNumberFullTimeTextField();
    String designationft=getDesignationFullTimeTextField();
    int salaryft=getSalaryFullTimeTextField();
    int workingHoursft=getWorkingHoursFullTimeTextField();

    if (vacancyNumberft >0 && salaryft > 0 && workingHoursft >0 && jobTypeft=="Full Time" &&
designationft != "") {
        // Iterates wthin the arraylist
        for (StaffHire obj:staffList){
            if(obj.getVacancyNumber() == vacancyNumberft){
                duplicateVacancyNum=true;
                break;
            }
        }
        if (duplicateVacancyNum==false){
            fullTimeStaff_obj= new FullTimeStaffHire(designationft,jobTypeft,vacancyNumberft ,
salaryft,workingHoursft);
            staffList.add(fullTimeStaff_obj);
            JOptionPane.showMessageDialog(frameFullTimeStaffHire,"Vacancy has been added");
        }else{// If the vacancy number is already in the list
```

```
                JOptionPane.showMessageDialog(frameFullTimeStaffHire, "The entered vacancy number is
already in the list");
            }
        }
    }

    // Adding functionality to the Appoint button of the Full Time Staff GUI
    if (e.getSource() == appointFullTimeButton){
        boolean vacancyNumFound=false;

        int vacancyNumberAppointft = getVacancyNumberAppointFullTimeTextField();
        String staffNameft=getStaffNameAppointFullTimeTextField();
        String joiningDateft= getJoiningDateAppointFullTime();
        String qualificationft= getQualificationAppointFullTimeTextField();
        String appointedByft= getAppointedbyAppointFullTimeTextField();
        if (vacancyNumberAppointft >0 && staffNameft!="" && joiningDateft !="" && qualificationft !="" &&
appointedByft !=""){
            // Iterates wthin the arraylist
            for (StaffHire obj:staffList){
                if(obj.getVacancyNumber() == vacancyNumberAppointft){
                    vacancyNumFound=true;
                    if (obj instanceof FullTimeStaffHire){// Checks whether the object is an instance of
FullTimeStaffHire subclass.
                        fullTimeStaff_obj = (FullTimeStaffHire)obj;
                        if (fullTimeStaff_obj.getJoined()==false){
                            fullTimeStaff_obj.hiring(staffNameft, joiningDateft, qualificationft, appointedByft);
                            JOptionPane.showMessageDialog(frameFullTimeStaffHire, "Staff Hired");
                            break;
                        }
                        else if (fullTimeStaff_obj.getJoined()==true){
                            JOptionPane.showMessageDialog(frameFullTimeStaffHire,"A staff has already been
hired to fill this vacancy no.");
                            break;
                        }
                    }else{// If the vacancy number entered is not the vacancy number for Full Time Staff
                        JOptionPane.showMessageDialog(frameFullTimeStaffHire,"Vacancy no.
"+vacancyNumberAppointft+" is not for Full Time Staff Hire");
```

```
                }
            }
        }
        if (vacancyNumFound==false){ // If the vacancy number entered is not in the arraylist
            JOptionPane.showMessageDialog(frameFullTimeStaffHire, "The inserted vacancy number is
invalid");
        }
    }
  }
}

  // The Main method
  public static void main(String[] args){
    new INGNepal();
  }
}
```

10. Appendix 2

Introduction

The coursework was assigned to us on the week 8 belonging to the module Programming. The first objective of the coursework is to create a class StaffHire with two sub classes PartTimeStaffHire and FullTimeStaffHire using BlueJ. The classes consist of different methods witlh accessor methods for each attributes and a method in the class PartTimeStaffHire to terminate the staff as well.

Description of the project

The program consists of three classes: PartTimeStaffHire, FullTimeStaffHire and StaffHire. The StaffHire being the parent class consists of attributes like the vacancy number, designation and job type of the vacancy in the organization. The constructor method lets the user set values in these attributes. There are getter method for each attributes in order to make it accessible to other classes. And a display method which displays the attributes i.e. the designation, job type and vacancy number.

The FullTimeStaffHire contains details of the staff like salary, working hour, staff name, joining date, qualification, appointed by the status if the staff has joined or not. Salary needs to be updated as well if the staff wants their salary changed. Hence, a setter method is required to update it. The working hour may also be required to be changed. Hence, it also has a setter method. A method is also required to display the details stored. Each of these attributes have their accessor method.

The PartTimeStaffHire stores details of the part time staffs. Details like working hour, wages per hour , staff name, joining date, qualification, appointed by, shifts and the status if the staff has joined or not. A method is also created to terminate the staff. The shifts may be required to be changed. So a setter method is required. A method is created to display all the details.

Aim

The main aim of this project is to create a java program which has three classes staffHire, PartTimeStaffHire and FullTimeStaffHire with main purpose to hire staffs for an organization.

Use of the application

The application is used to hire staff whose job type may be part time or full time. The program must store details about the staff like name, joining date, qualification, etc.

Tools used

- Text editor for coding – BlueJ
- Program for class diagram – draw.io

Class Diagram

Class diagram is a static diagram which represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modelling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram. (Tutorialspoint, 2020)

The class diagram including all the classes is given in the next page.

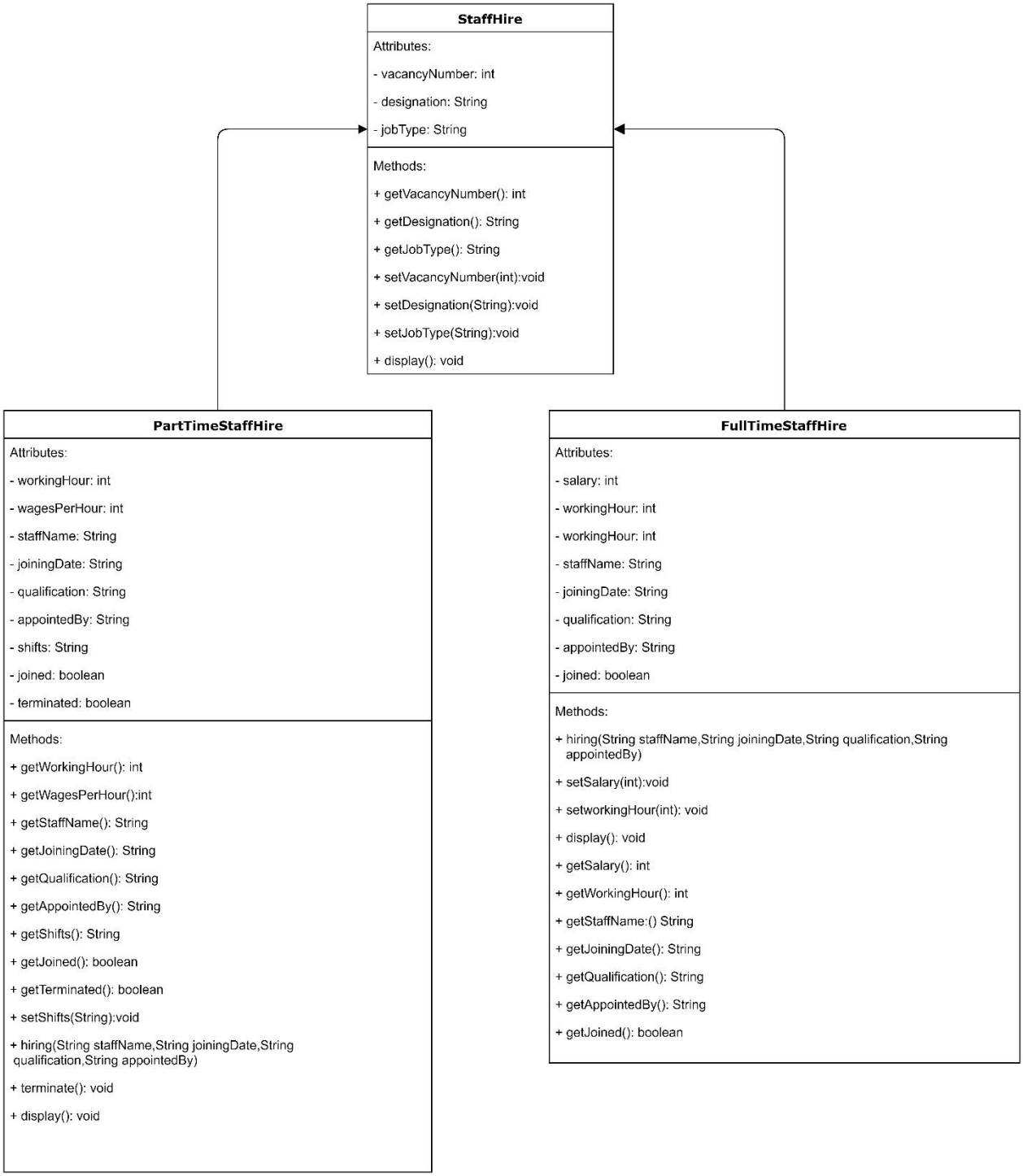

Figure 1: Class Diagram of StaffHire, FullTimeStaffHire and PartStaffHire

Pseudocode

Pseudocode is an informal way of programming description that does not require any strict programming language syntax or underlying technology considerations. It is used for creating an outline or a rough draft of a program. (The Economic Times, 2019)

Pseudocode for StaffHire

```
DEFINE class StaffHire
        DO

                INITIALIZE vacancyNumber as int type;
                INITIALIZE designation as String type;
                INITIALIZE jobType as String type;
            DEFINE StaffHire (String designation,String jobType, int vacancyNumber)
                DO

                        EXTRACT this.designation=designation;
                        EXTRACT this.jobtype=jobType;
                        EXTRACT this.vacancyNumber=vacancyNumber;
                END DO

            DEFINE int returntype getVacancyNumber()
                DO

                        EXTRACT "this.vacancyNumber"

                        RETURN "this.vacancyNumber"
                END DO

            DEFINE String returntype getDesignation()
                DO

                        EXTRACT "this.designation"
                        RETURN "this.designation"
                END DO

            DEFINE String returntype getJobType()
                DO
```

```
                    EXTRACT "this.jobType"
                    RETURN "this.jobType"
            END DO

    DEFINE no returntype void setVacancyNumber(int vacancyNumber)
                    DO

                    EXTRACT this.vacancyNumber=vacancyNumber;
                    END DO

    DEFINE no returntype void setDesignation(String designation)
                    DO

                      EXTRACT this.designation=designation;
                    END DO

   DEFINE no returntype void setJobType (String jobType)
                    DO

                          EXTRACT this.jobType=jobType;
                    END DO

      DEFINE no returntype void display()
            DO

                    PRINT "designation"

                    PRINT "jobType"

                    PRINT "vacancyNumber"
            END DO

END DO
```

Pseudocode for PartTimeStaffHire

```
DEFINE class PartTimeStaffHire
        DO
```

```
INITIALIZE workingHour as int type;
INITIALIZE wagesPerHour as int type;
INITIALIZE staffName as String type;
INITIALIZE joiningDate as String type;
INITIALIZE qualification as String type;
INITIALIZE appointedBy as String type;
INITIALIZE shifts as String type;
INITIALIZE joined as Boolean type;
INITIALIZE terminated as Boolean type;
DEFINE PartTimeStaffHire (int vacancyNumber, String designation, String jobType, int
workingHour, int wagesPerHour, String shifts)
DO

EXTRACT super (designation, jobType, vacancyNumber);
EXTRACT this.workingHour = workingHour;
EXTRACT this.wagesPerHour=wagesPerHour;
EXTRACT this.shifts=shifts;

EXTRACT staffName=" ";
EXTRACT joiningDate=" ";

EXTRACT qualification=" ";
EXTRACT appointedBy=" ";
EXTRACT this.joined=false;
EXTRACT this.terminated=false;
END DO

DEFINE int returntype getWorkingHour()
DO

EXTRACT "workingHour"
RETURN "workingHour"
END DO

DEFINE int returntype getWagesPerHour()
DO
```

```
                EXTRACT "wagesPerHour"
                RETURN "wagesPerHour"
            END DO

        DEFINE String returntype getStaffName()
            DO

                EXTRACT "staffName"
                RETURN "staffName"
            END DO

        DEFINE String returntype getJoiningDate()
            DO

                EXTRACT "joiningDate"
                RETURN "joiningDate"
            END DO

        DEFINE String returntype getQualification()
            DO

                EXTRACT "qualification"
                RETURN "qualification"
            END DO

DEFINE String returntype getAppointedBy()
            DO

                EXTRACT "appointedBy"
                RETURN "appointedBy"
            END DO

        DEFINE String returntype getShifts()
            DO

                EXTRACT "shifts"

                RETURN "shifts"
```

```
        END DO

    DEFINE boolean returntype getJoined()
        DO

            EXTRACT "joined"

            RETURN "joined"
        END DO

    DEFINE boolean returntype getTerminated()
        DO

            EXTRACT "terminated"
            RETURN "terminated"
        END DO

    DEFINE no returntype void setShifts(String shifts)
        IF (joined==false)
            DO

                EXTRACT this.shifts=shifts;
            END DO

        ELSE

            PRINT "Staff has already been hired.Shift cannot be changed"
        END DO

        END IF

DEFINE no returntype void hiring (String staffName,String joiningDate,String
qualification,String appointedBy)

        IF(joined==false)
            DO

                EXTRACT this.staffName=staffName;
                EXTRACT this.joiningDate=joiningDate
```

```
                    EXTRACT this.qualification=qualification;
                    EXTRACT this.appointedBy=appointedBy;
                    EXTRACT this.joined=true;

                    PRINT (“Staff Hired”);
            ELSE

            DO

                    PRINT (this.staffName+”has already joined

                    on”+this.joiningDate+”having”+this.qualification+”who was
appointed by”+this.appointedBy);

            END DO

            END IF

DEFINE no returntype void terminate()
        IF(terminated==false)
            DO

            EXTRACT this.staffName=" ";
            EXTRACT this.joiningDate=" ";
            EXTRACT this.qualification=" ";
            EXTRACT this.appointedBy=" ";
            EXTRACT this.joined=false;
            EXTRACT this.terminated=true;
            PRINT ("The staff has been terminated");
        ELSE

        DO

            PRINT ("The staff has already been terminated");
        END DO

        END IF
```

```
DEFINE no returntype void display()
        DO

                PRINT("------------------Full-Time-Staff------------------");
                 EXTRACT super.display();
          IF(joined==true);
                DO

                        INITIALIZE totalincome as int type;
                         EXTRACT totalincome=wagesPerHour*workingHour;
                        PRINT ("Staff Name: "+staffName);
                         PRINT ("Wages Per Hour: "+wagesPerHour);
                       PRINT ("Working Hours Per Day: "+workingHour);
                        PRINT ("Joined Date: "+joiningDate);
                        PRINT ("Qualification: "+qualification);
                        PRINT ("Appointed By: "+appointedBy);
                    PRINT ("Shift: "+shifts);
                         PRINT ("Income per day: "+totalincome);
                          PRINT ("--------------------------------------------------");
         END DO

        END IF

  END DO
```

Pseudocode for FullTimeStaffHire

```
DEFINE class FullTimeStaffHire
        DO

                INITIALIZE salary as int type;
                 INITIALIZE workingHour as int type;
                 INITIALIZE staffName as String type;
                 INITIALIZE joiningDate as String type;
                 INITIALIZE qualification as String type;
                 INITIALIZE appointedBy as String type;
                INITIALIZE joined as Boolean type;
```

```
DEFINE FullTimeStaffHire(int vacancyNumber, String jobType,String
designation,int salary,int workingHour)
DO
    EXTRACT super(designation,jobType,vacancyNumber);
    EXTRACT this.salary=salary;
    EXTRACT this.workingHour=workingHour;
    EXTRACT this.staffName="";
    EXTRACT this.joiningDate="";
    EXTRACT this.qualification="";
    EXTRACT this.appointedBy="";
    EXTRACT this.joined=false;
END DO

DEFINE no returntype void hiring (String staffName,String joiningDate,String
qualification,String appointedBy)
IF(joined==false)
DO
    EXTRACT this.staffName=staffName;
    EXTRACT this.joiningDate=joiningDate
    EXTRACT this.qualification=qualification;
    EXTRACT this.appointedBy=appointedBy;
    EXTRACT this.joined=true;

    PRINT (“Staff Hired”);
ELSE
DO
    PRINT ("The staff has already been hired whose name is
"+this.staffName+ " and the date that they joined
is"+this.joiningDate);
END DO
```

```
            END IF

DEFINE no returntype void setSalary(int salary)
            IF (this.joined==false)
            DO

                    EXTRACT this.salary=salary;
            ELSE

                    PRINT ("The staff has already joined. Therefore,you cannot salary

                  change.");

      END DO

            END IF

DEFINE no returntype void setWorkingHour(int workingHour)
            DO

              EXTRACT this.workingHour=workingHour;
            END DO

DEFINE no returntype void display()
      DO

            PRINT("------------------Part-Time-Staff------------------");
             EXTRACT super.display();
        IF(this.joined==true);
            DO

                    PRINT("Staffname: "+this.staffName);
                    PRINT ("Salary: "+this.salary);
                    PRINT ("Working Hours: "+this.workingHour);
                    PRINT ("Joining date: "+this.joiningDate);
                    PRINT ("Qualification: "+this.qualification);
                    PRINT ("Appointed by: "+this.appointedBy);
          PRINT ("--------------------------------------------------");
       END DO
```

```
        END IF

DEFINE int returntype getSalary()
            DO

                EXTRACT "salary"

                RETURN "salary"
            END DO

DEFINE int returntype getWorkingHour()
            DO

                EXTRACT "workingHour"
                RETURN "workingHour"
            END DO

DEFINE String returntype getStaffName()
            DO

                EXTRACT "staffName"
                RETURN "staffName"
            END DO

DEFINE String returntype getJoiningDate()
            DO

                EXTRACT "joiningDate"
                RETURN "joiningDate"
            END DO

DEFINE String returntype getQualification()
            DO

                EXTRACT "qualification"
                RETURN "qualification"
            END DO
```

DEFINE String returntype getAppointedBy()

DO

EXTRACT “appointedBy”

RETURN “appointedBy”

END DO

DEFINE boolean returntype getJoined()

DO

EXTRACT “joined”

RETURN “joined”

END DO

END DO

Method Description

Method description states the purpose and function of each method.

Method Description of Class StaffHire

The method description of the class StaffHire are listed below:

1. getVacancyNumber()
   It is the getter method that returns the value of vacancyNumber that has int datatype when called. Getter method is used to make private variable accessible.

2. getDesignation()
   It is the getter method that returns the value of Designation as string datatype. It makes the private variable accessible.

3. getJobType()
   It is the getter method that returns the value of jobType as string datatype. It makes the private variable accessible.

4. setVacancyNumber(int vacancyNumber)
   It is setter which updates value of vacancyNumber having int datatype. Setter method is used to update the value of the variable which is declared private. It allows to set a new value of int vacancyNumber.
5. setDesignation(String designation)
   It is setter which updates value of designation having string datatype. Setter method is used to update the value of the variable which is declared private. It allows to set a new value of string designation.
6. setJobType(String jobType)
   It is setter which updates value of jobType having string datatype. Setter method is used to update the value of the variable which is declared private. It allows to set a new value of string jobType.
7. display()
   This method does not return any type. Hence, it is declared void. It is used to display the designation, jobType and vacancyNumber to the user.

Method Description of Class PartTimeStaffHire

The method description of the class PartTimeStaffHire are listed below:

1. getWorkingHour()
   It is the getter method that returns the value of workingHour that has int datatype when called. Getter method is used to make private variable accessible.

2. getWagesPerHour()
   It is the getter method that returns the value wagesPerHour as int datatype. It makes the private variable accessible.

3. getStaffName()
   It is the getter method that returns the value of staffName as string datatype. It makes the private variable accessible.

4. getJoiningDate()
   It is the getter method that returns the value of joiningDate as string datatype. It makes the private variable accessible.

5. getQualification()
   It is the getter method that returns the value of qualification as string datatype. It makes the private variable accessible.

6. getAppointedBy()
   It is the getter method that returns the value of appointedBy as string datatype. It makes the private variable accessible.

7. getShifts()
   It is the getter method that returns the value of shifts as string datatype. It makes the private variable accessible.

8. getJoined()
   It is the getter method that returns the value of joined as boolean datatype. It makes the private variable accessible.

9. getTerminated()
   It is the getter method that returns the value of terminated as boolean datatype. It makes the private variable accessible.

10. setShifts(String shifts)
    It is setter which updates value of shifts having string datatype. Setter method is used to update the value of the variable which is declared private. It first checks the status of Boolean joined. If the status is false then it allows to set a new value of string shifts. Else a message saying "Staff has already been hired. Shift cannot be changed." is shown in the screen.

11. hiring(String staffName,String joiningDate,String qualification,String appointedBy) This method is used to hire staff if he/she has not joined. It checks the status of joined. If false, it allows the user to set values of staffName, joiningDate, qualification, appointedBy and changes the status of joined to true. A message saying “Staff Hired” is also displayed.

    If the status joined is true, the suitable message is displayed.

12. terminate()

    The method checks the status of the terminated and if false sets the value of staffName, joiningDate, qualification,appointedBy to null , joined to false and terminated to true. The message "The staff has been terminated" is also displayed. If the status of terminated is true, the message "The staff has already been terminated" is displayed.

13. display()

    This method does not return any type. Hence, it is declared void. It is displays the display() method from the super class StaffHire. It also checks the status of the joined and if true, variable totalincome is declared which is int. It is the product of wagesPerHour and workingHour. And staffName, wagesPerHour, workingHour,joiningDate,qualification,appointedBy , totalincome and totalincome is also displayed with suitable message.

Method Description of Class FullTimeStaffHire

The method description of the class FullTimeStaffHire are listed below:

1. hiring(String staffName,String joiningDate,String qualification,String appointedBy) This method is used to hire staff if he/she has not joined. It checks the status of joined. If false, it allows the user to set values of staffName, joiningDate, qualification, appointedBy and changes the status of joined to true. A message saying “Staff Hired” is also displayed. If the status joined is true, the suitable message is displayed.

2. setSalary(int salary)

   It is setter which updates value of salary having int datatype. Setter method is used to update the value of the variable which is declared private. It first checks the status of Boolean joined. If the status is false then it allows to set a new value of int salary.

Else a message saying "The staff has already joined. Therefore,you cannot salary change." is shown in the screen.

3. setWorkingHour(int workingHour)

   It is setter which updates value of workingHour having int datatype. Setter method is used to update the value of the variable which is declared private. It allows to set a new value of int workingHour.

4. display()

   This method does not return any type. Hence, it is declared void. It is displays the display() method from the super class StaffHire. It also checks the status of the joined and if true, staffName, salary, workingHour, joiningDate, qualification, appointedBy is displayed with suitable message.

5. getSalary()

   It is the getter method that returns the value salary as int datatype. It makes the private variable accessible.

6. getWorkingHour()

   It is the getter method that returns the value of workingHour that has int datatype when called. Getter method is used to make private variable accessible.

7. getStaffName()

   It is the getter method that returns the value of staffName as string datatype. It makes the private variable accessible.

8. getJoiningDate()

   It is the getter method that returns the value of joiningDate as string datatype. It makes the private variable accessible.

9. getQualification()

   It is the getter method that returns the value of qualification as string datatype. It makes the private variable accessible.

10. getAppointedBy()

    It is the getter method that returns the value of appointedBy as string datatype. It makes the private variable accessible.

11. getJoined()

    It is the getter method that returns the value of joined as boolean datatype. It makes the private variable accessible.

Testing (Inspection)

Testing is the activity to check whether the actual results match the expected results and to ensure that the software system is defect free. (Guru99, 2020)

Test 1: To inspect FullTimeStaffHire Class, appoint the full time staff, and reinspect the FullTimeStaffHire Class

Test No.	**1**
Objective:	To inspect FullTimeStaffHire Class, appoint the full time staff, and reinspect the FullTimeStaffHire Class
Action:	➔ The FullTimeStaffHire is called with the following arguments: vacancyNumber=1 jobType="Full Time" designation="Teacher" salary="50000" workingHour="8" ➔ Inspection of the FullTimeStaffHire class. ➔ void hiring is called with the following arguments: staffName="Badrinath Bhusal" joiningDate="10 January 2020" qualification="10 years experience" apppointedBy="HR" ➔ Re-inspection of the FullTimeStaffHire class.
Expected Result:	The full time staff would be hired.
Actual Result:	The full time staff was appointed
Conclusion:	The test is successful.

Table 1: To inspect FullTimeStaffHire Class, appoint the full time staff, and reinspect the FullTimeStaffHire Class

Output result

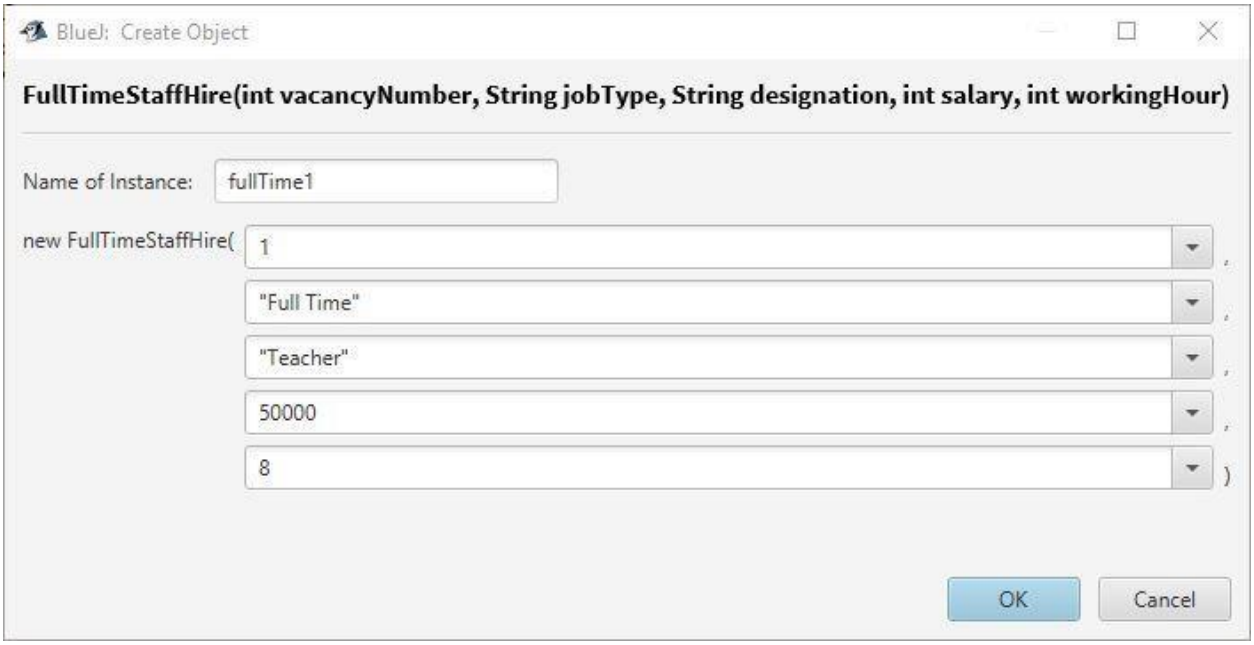

Figure 2: Screenshot of assigning the data in FullTimeStaffHire class

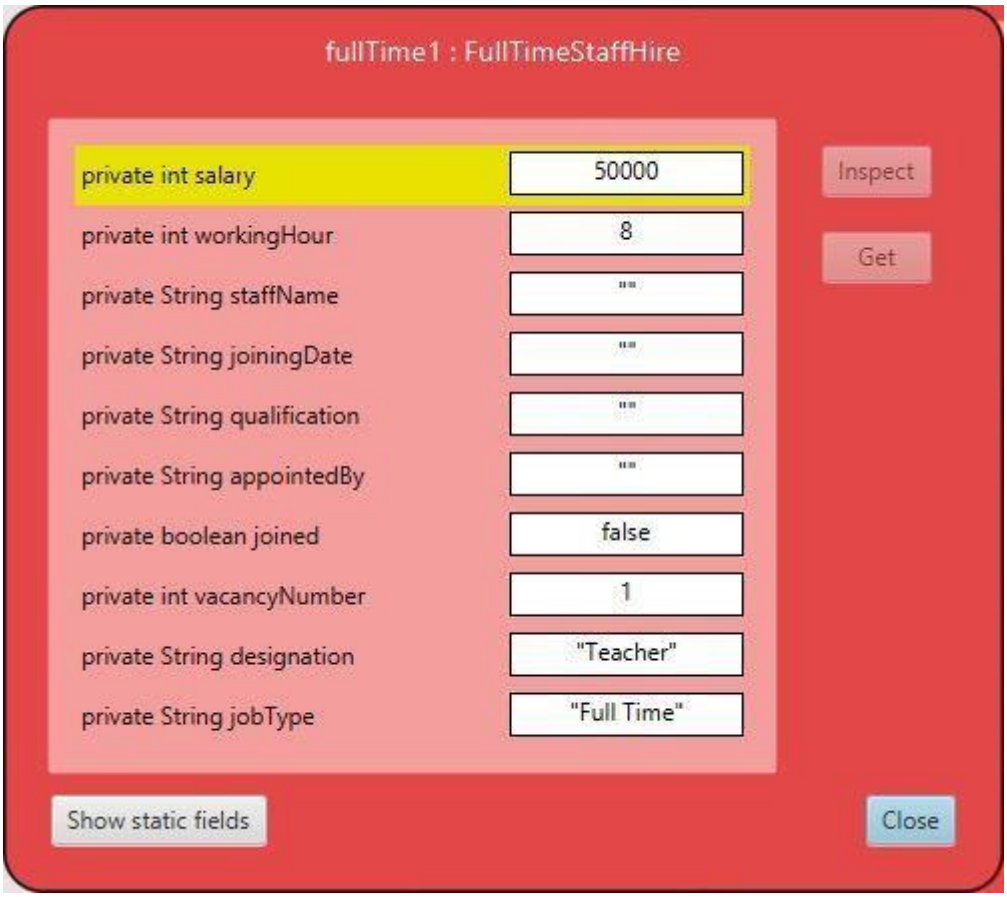

Figure 3: Screenshot of the inspection of the FullTimeStaffHire class

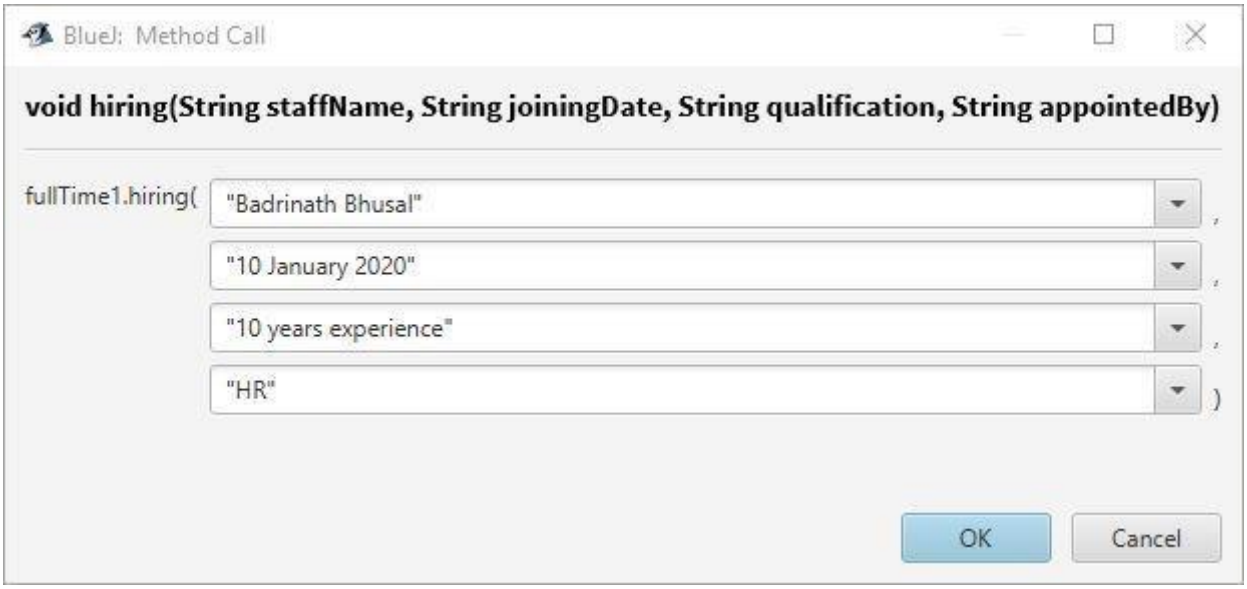

Figure 4: Screenshot of assigning the data in void hiring

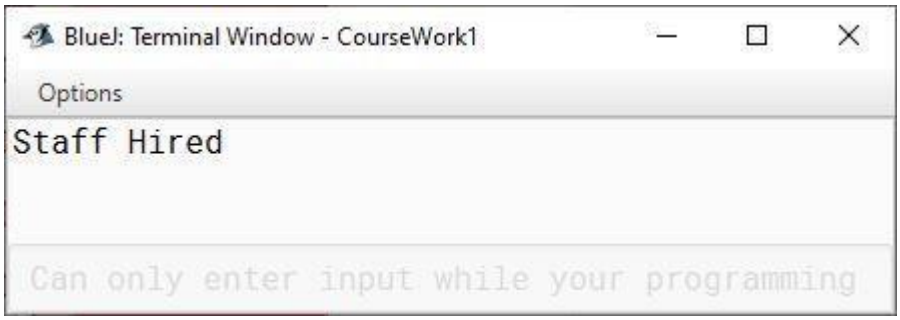

Figure 5: Screenshot of the message displayed after the data has been entered in void hiring

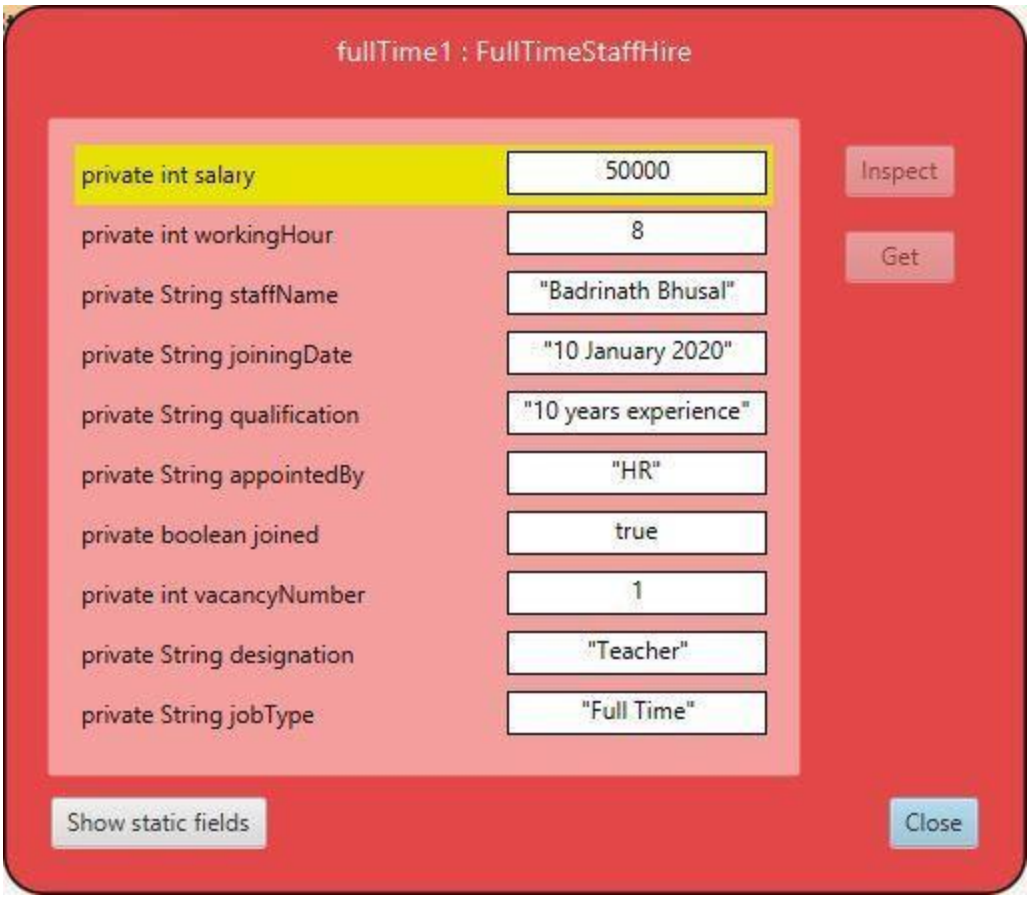

Figure 6: Screenshot of the re-inspection of the FullTimeStaffHire class

Test 2: To inspect PartTimeStaffHire Class, appoint part time staff, and reinspect the PartTimeStaffHire Class

Test No.	**2**
Objective:	To inspect PartTimeStaffHire Class, appoint part time staff, and reinspect the PartTimeStaffHire Class
Action:	➔ The PartTimeStaffHire is called with the following arguments: vacancyNumber=2 designation="Janitor" jobType="Part Time" workingHour=3 wagesPerHour=200 shifts=Day ➔ Inspection of the PartTimeStaffHire class. ➔ void hiring is called with the following arguments: staffName="Suman Subedi" joiningDate="8 Jan 2020" qualification="5 years experience" apppointedBy="HR" ➔ Re-inspection of the PartTimeStaffHire class.
Expected Result:	The part time staff would be hired
Actual Result:	The part time staff was appointed
Conclusion:	The test is successful.

Table 2: To inspect PartTimeStaffHire Class, appoint part time staff, and reinspect the PartTimeStaffHire Class

Output Result

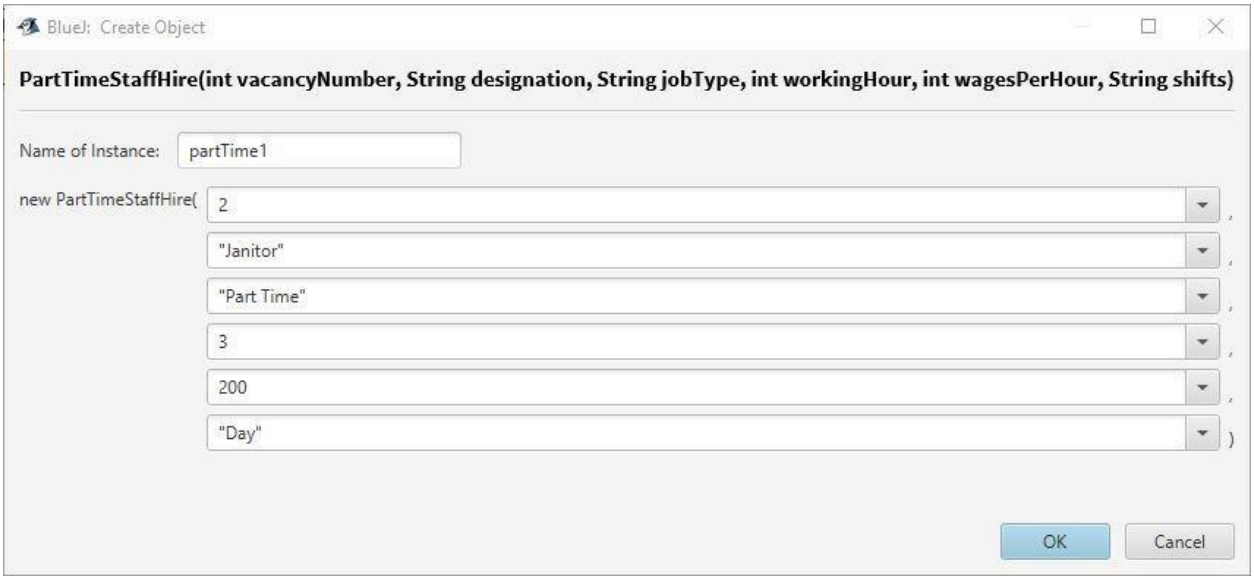

Figure 7: Assigning the data in the PartTimeStaffHire

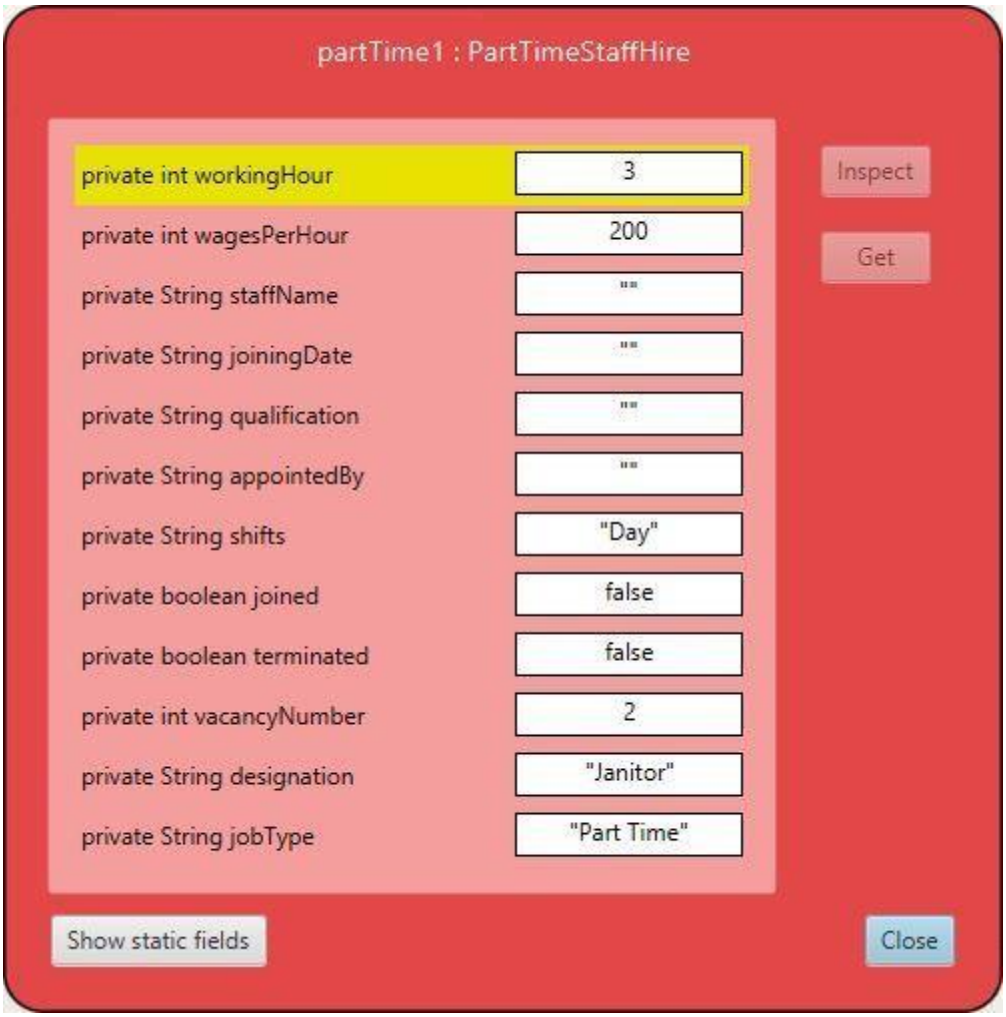

Figure 8: Screenshot of inspection of the PartTimeStaffHire

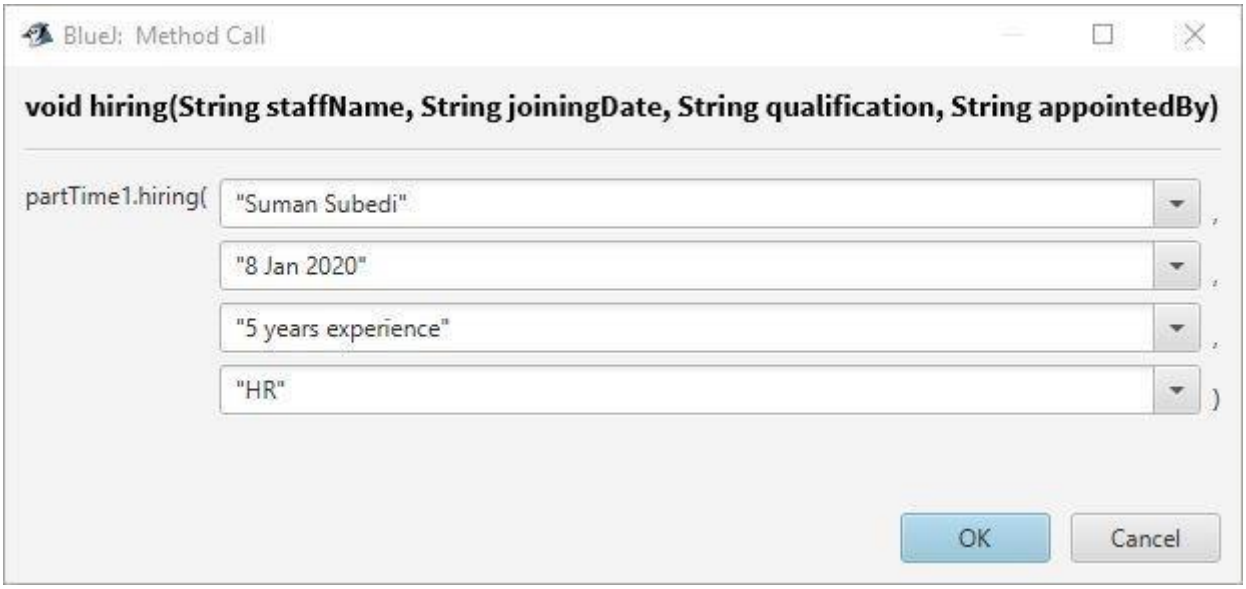

Figure 9: Assigning the data in void hiring

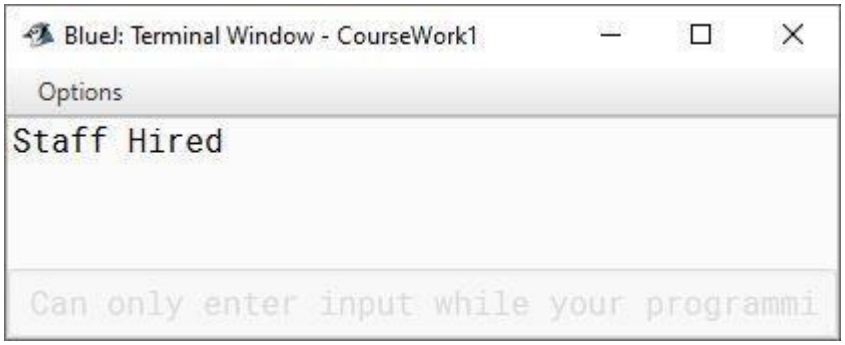

Figure 10: Screenshot of the message displayed after the data has been entered in void hiring

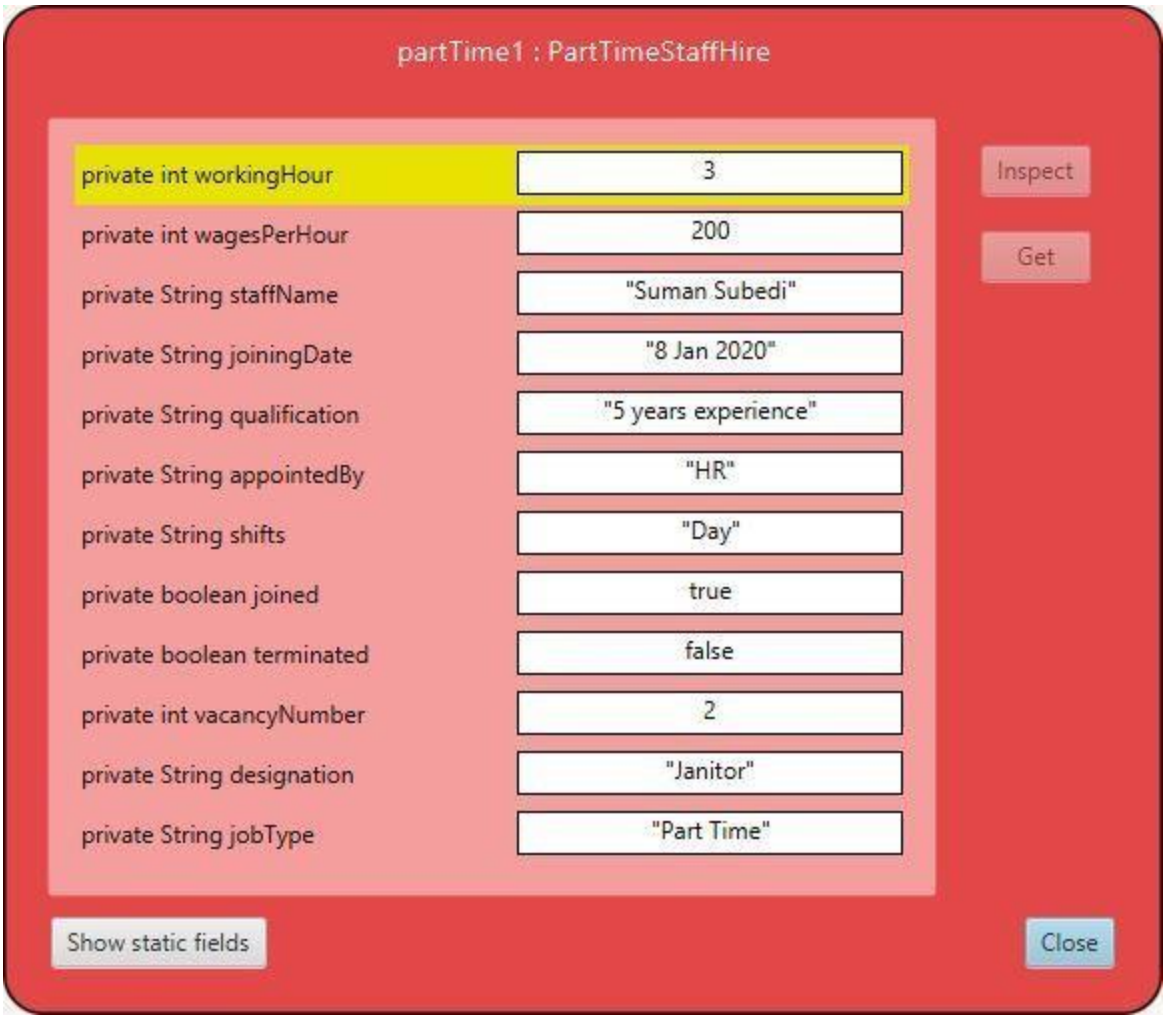

Figure 11: Screenshot of the re-inspection of the PartTimeStaffHire class

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

Test No.	**3**
Objective:	To inspect PartTimeStaffHire Class, change the termination status of a staff, and re-inspect the PartTimeStaffHire Class
Action:	➔ The following argument is called in the PartTimeStaffHire class: void terminate () ➔ Re-inspection of the PartTimeStaffHire class after the termination.
Expected Result:	The status of the PartTimeClass would be terminated
Actual Result:	Status of PartTimeStaffHire was changed
Conclusion:	The test is successful.

Table 3: To inspect PartTimeStaffHire Class, change the termination status of a staff, and re-inspect the PartTimeStaffHire Class

Output result

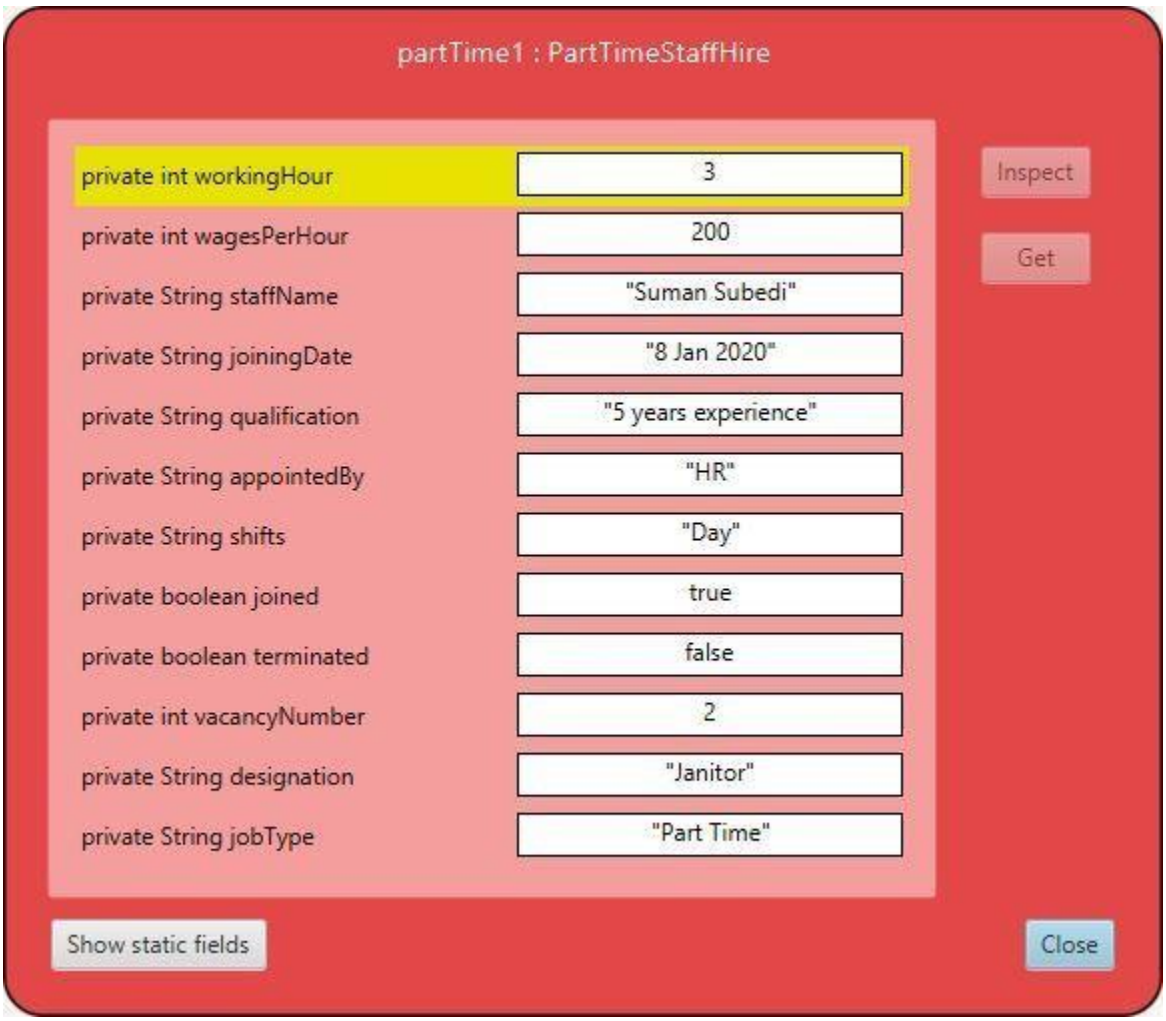

Figure 12: Inspection of the PartTimeStaffHire class before the termination

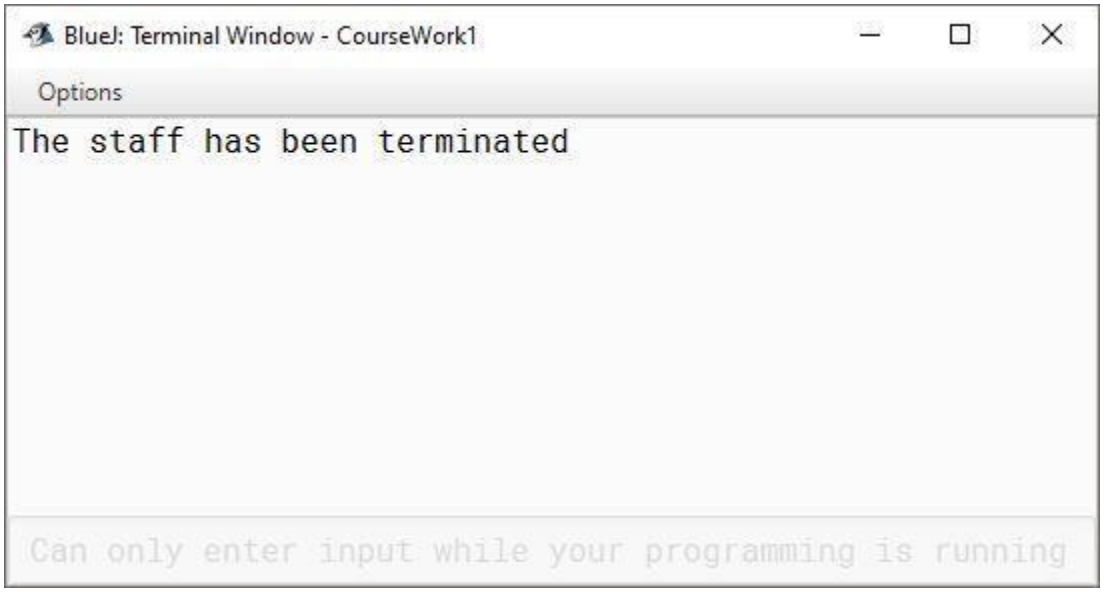

Figure 13: Display of the staff termination

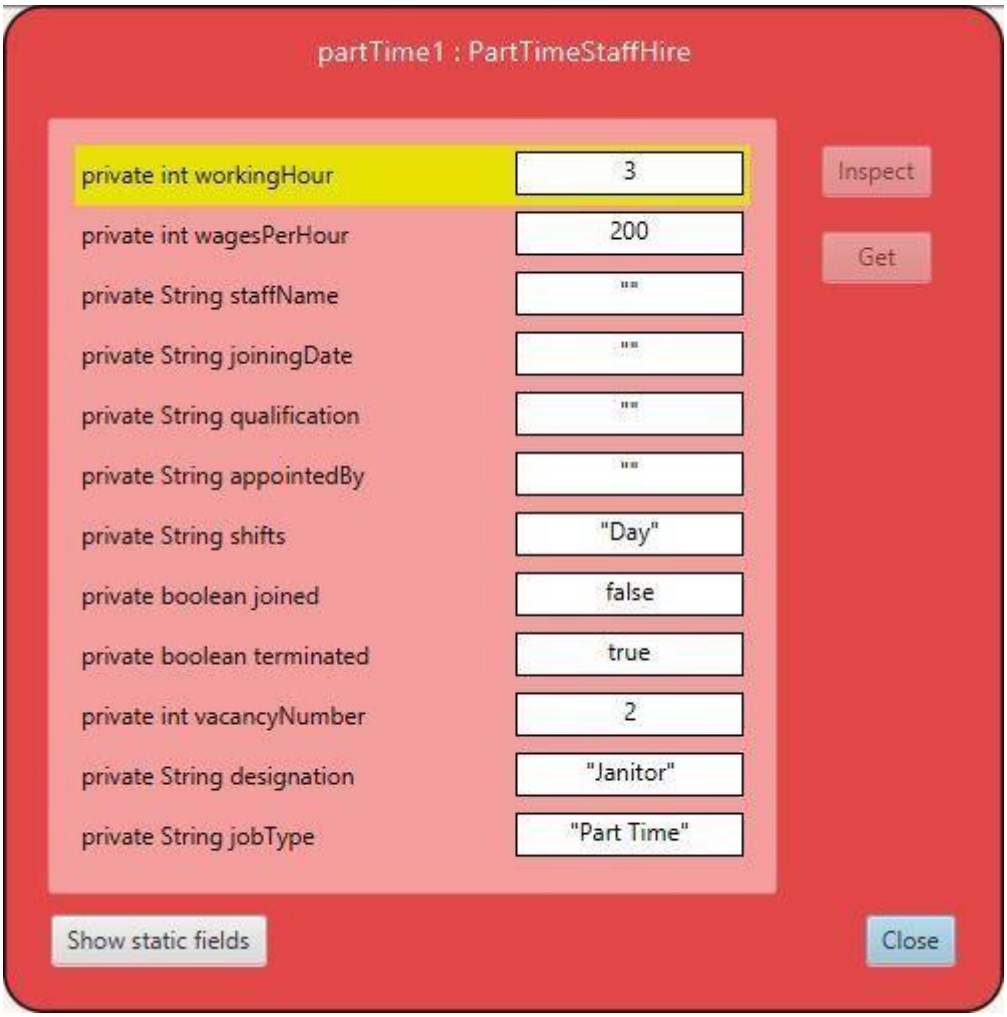

Figure 14: Inspection of PartTimeStaffHire class after the termination

Test 4: To display the detail of FullTimeStaffHire and PartTimeStaffHire Class

Test No.	**4**
Objective:	To display the detail of FullTimeStaffHire and PartTimeStaffHire Class
Action:	➔ The details of the FullTimeStaffHire class is displayed. ➔ The details of the PartTimeStaffHire class is displayed.
Expected Result:	The details of PartTimeStaffHire and FullTimeStaffHire would be displayed.
Actual Result:	The details of PartTimeStaffHire and FullTimeStaffHire was be displayed.
Conclusion:	The test is successful.

Table 4: To display the detail of FullTimeStaffHire and PartTimeStaffHire class

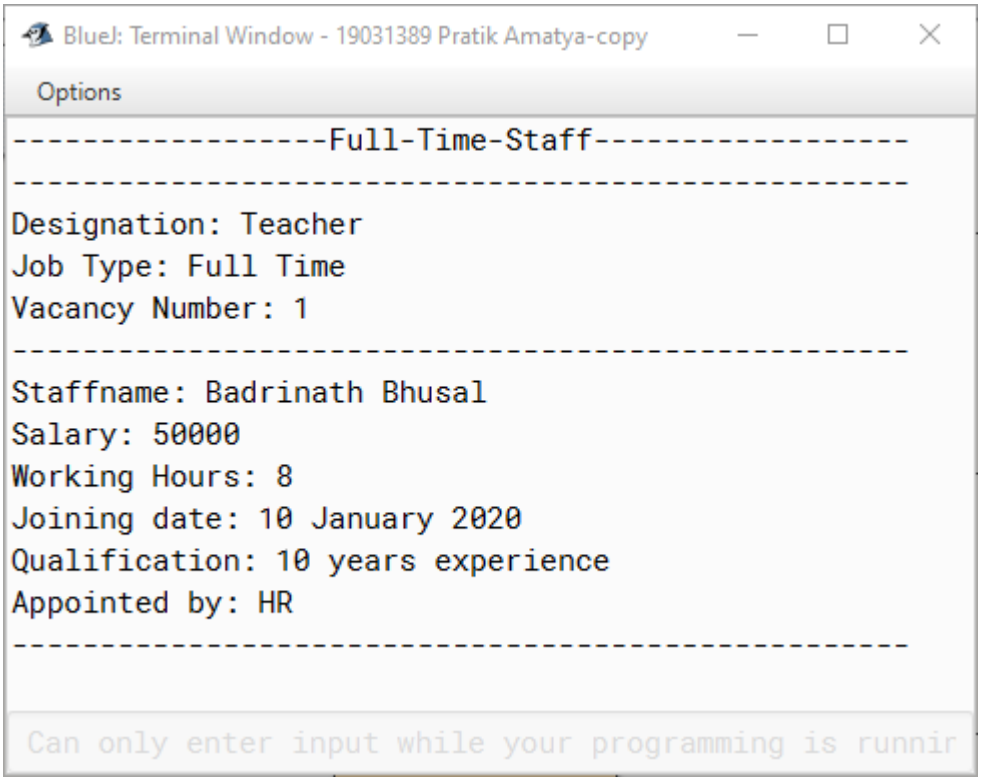

Figure 15: Displaying the details of the FullTimeStaffHire class

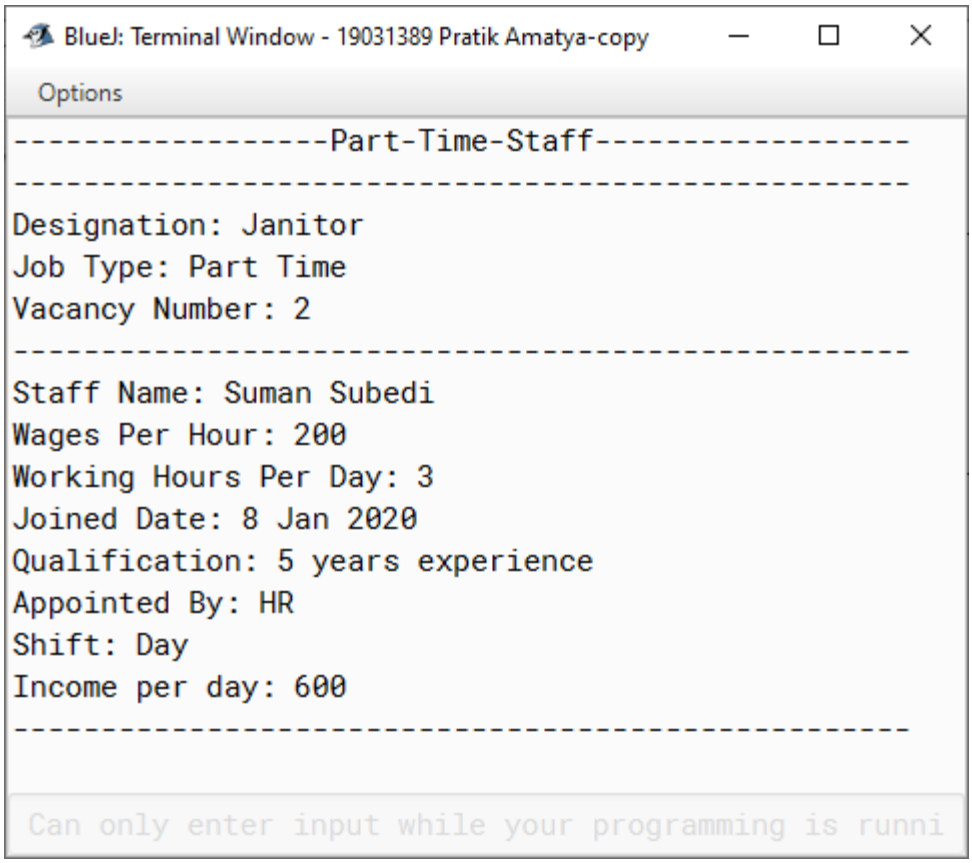

Figure 16: Displaying the details of the PartTimeStaffHire class

Errors

Error 1: Runtime Error

The error below occurs because the same method is repeated inside the method itself.

The error below shows java.lang.StackOverflow Error. This is fixed by adding super. Before the display() as the main intention to call the display() method from the parent table. Hence super.display() is used to displace the error.

```
public void display(){
    System.out.println("------------------Part-Time-Staff------------------");
    display();
    if (this.joined==true){
        int totalincome;
        totalincome=wagesPerHour*workingHour;
        System.out.println("Staff Name: "+staffName);
        System.out.println("Wages Per Hour: "+wagesPerHour);
        System.out.println("Working Hours Per Day: "+workingHour);
        System.out.println("Joined Date: "+joiningDate);
        System.out.println("Qualification: "+qualification);
        System.out.println("Appointed By: "+appointedBy);
        System.out.println("Shift: "+shifts);
        System.out.println("Income per day: "+totalincome);
        System.out.println("----------------------------------------------------");
    }
}
```

Figure 17: Screenshot of the error 1

```
public void display(){
    System.out.println("------------------Part-Time-Staff------------------");
    super.display();
    if (this.joined==true){
        int totalincome;
        totalincome=wagesPerHour*workingHour;
        System.out.println("Staff Name: "+staffName);
        System.out.println("Wages Per Hour: "+wagesPerHour);
        System.out.println("Working Hours Per Day: "+workingHour);
        System.out.println("Joined Date: "+joiningDate);
        System.out.println("Qualification: "+qualification);
        System.out.println("Appointed By: "+appointedBy);
        System.out.println("Shift: "+shifts);
        System.out.println("Income per day: "+totalincome);
        System.out.println("----------------------------------------------------");
    }
}
```

Figure 18: Screenshot of the correction for the error 1

Error 2: Syntax Error

This is an example of syntax error. A syntax error usually occurs when it unable to execute the program. It can occur due to incorrect spelling or absence of variable declaration which in case shows Cannot find symbol error. The error below has occurred as totalincome has not been declared before assigning its values. So, to fix this error, first the variable is declared as int variable and then only value is assigned.

```
public void display(){
    System.out.println("------------------Part-Time-Staff------------------");
    super.display();
    if (this.joined==true){
        totalincome=wagesPerHour*workingHour;
        System.out.println("Staff Name: "+staffName);
        System.out.println("Wages Per Hour: "+wagesPerHour);
        System.out.println("Working Hours Per Day: "+workingHour);
        System.out.println("Joined Date: "+joiningDate);
        System.out.println("Qualification: "+qualification);
        System.out.println("Appointed By: "+appointedBy);
        System.out.println("Shift: "+shifts);
        System.out.println("Income per day: "+totalincome);
        System.out.println("----------------------------------------------------");
    }
}
```

Figure 19: Screenshot of the error 2

```
public void display(){
    System.out.println("------------------Part-Time-Staff------------------");
    super.display();
    if (this.joined==true){
        int totalincome;
        totalincome=wagesPerHour*workingHour;
        System.out.println("Staff Name: "+staffName);
        System.out.println("Wages Per Hour: "+wagesPerHour);
        System.out.println("Working Hours Per Day: "+workingHour);
        System.out.println("Joined Date: "+joiningDate);
        System.out.println("Qualification: "+qualification);
        System.out.println("Appointed By: "+appointedBy);
        System.out.println("Shift: "+shifts);
        System.out.println("Income per day: "+totalincome);
        System.out.println("----------------------------------------------------");
    }
}
```

Figure 20: Screenshot of the correction for the error 2

Error 3: Logic Error

The error below is an example of logic error. No error is shown when complied. Here, the this.joined is declared false even if the staff has joined. Hence, when the display() method is called, the staff details is not shown as the status of joined is still false. Hence, the problem is fixed by changing the value of this.joined to true when staff is hired using hiring method.

```
public void hiring(String staffName,String joiningDate,String qualification,String appointedBy){
        if (this.joined==false){
            this.staffName=staffName;
            this.joiningDate=joiningDate;
            this.qualification=qualification;
            this.appointedBy=appointedBy;
            this.joined=false;
            System.out.println("Staff Hired");
        }else{
        System.out.println("The staff has already been hired whose name is "+this.staffName+ " and the date that they joined is"+this.joiningDate);
        }
}
```

Figure 21: Screenshot of the error 3

```
public void hiring(String staffName,String joiningDate,String qualification,String appointedBy){
        if (this.joined==false){
            this.staffName=staffName;
            this.joiningDate=joiningDate;
            this.qualification=qualification;
            this.appointedBy=appointedBy;
            this.joined=true;
            System.out.println("Staff Hired");
        }else{
        System.out.println("The staff has already been hired whose name is "+this.staffName+ " and the date that they joined is"+this.joiningDate);
        }
}
```

Figure 22: Screenshot of the correction for the error 3

Conclusion

In conclusion, three different classes were created. The StaffHire class is the parent class and the FullTimeStaff Hire and the PartTimeStaffHire are the sub classes. Each methods were assigned with different variable which were of different data types. This project helped knowledge on different methods like setter method and getter method and different types of classes.

While doing this coursework, I faced difficulties like while many errors occurring while coding and the topic was also new to me. I did not grasp the idea about the purpose and the way of using the types of method at once. I also had confusions regarding the pseudocode and the class diagram. But, in order to solve the problems and reduce the confusion, I consulted my tutor and researched on the Internet. Many YouTube videos with internet surfing were watched and done on java programming from the bottom level and to carry out this coursework. Going through the lecture slides helped to gain good knowledge about java. I also got to know about the methods which are used in the program and purpose of different type of it like Getter and Setter method.

The class diagram was done in draw.io and the coding in text editor BlueJ.

Although it was challenging and difficult at first, the coursework was fruitful. I also got to learned many new things and topics which I was not aware about. It was a great experience to develop a program using java, spending time on the study zone coding .I enjoyed the experience and value the knowledge that I learned from it.

Bibliography

Guru99. (2020) *What is Software Testing_ Introduction, Definition, Basics & Types* [Online]. Available from: https://www.guru99.com/software-testing-introductionimportance.html [Accessed 13 January 2020].

The Economic Times. (2019) *What is Pseudocode_ Definition of Pseudocode, Pseudocode Meaning - The Economic Times* [Online]. Available from: https://economictimes.indiatimes.com/definition/pseudocode [Accessed 9 January 2020].

Tutorialspoint. (2020) *UML - Class Diagram - Tutorialspoint* [Online]. Available from: https://www.tutorialspoint.com/uml/uml_class_diagram.htm [Accessed 7 January 2020].

Appendix

List of code

Code for Class StaffHire

```
/**
 * This application is designed to
 * represent Staff Hire,together with two subclasses to represent a Full-time Staff and a Part-
time Staff respectively
 * @version 0.1
 * @author Pratik
 */
//declaring the attributes name
public class StaffHire{
        private int vacancyNumber;//the unique number given to each vacancy
        private String designation;//string that contains the post
        private String jobType;//string containing whether the job type is fulltime or parttime
        //the constructor class of the class StaffHire
            public StaffHire(String designation,String jobType,int vacancyNumber){
                this.designation=designation;
                this.jobType=jobType;
                this.vacancyNumber=vacancyNumber;
            }
            //declaring getter method for all attributes
            public int getVacancyNumber(){
                return this.vacancyNumber;//The getvacancyNumber() method returns the value of
vacancy number as int datatype.
            }
            public String getDesignation(){
                return this.designation;//The getdesignation() method returns the value of designation
as string datatype.
            }
            public String getJobType(){
```

```
        return this.jobType;//The getjobType() method designation returns the value of job
type as string datatype.
    }
    //declaring setter method for all attributes
    public void setVacancyNumber(int vacancyNumber){
        this.vacancyNumber=vacancyNumber;
    }
    public void setDesignation(String designation){
        this.designation=designation;
    }
    public void setJobType(String jobType){
        this.jobType=jobType;
    }
    // displaying the designation,jobtype and the vacancy number
    public void display(){
        System.out.println("--------------------------------------------------");
        System.out.println("Designation: "+designation);
        System.out.println("Job Type: "+jobType);
        System.out.println("Vacancy Number: "+vacancyNumber);
        System.out.println("--------------------------------------------------");
    }
}
```

Code for Class FullTimeStaffHire

```
/**
 * This application is designed to
 * represent Staff Hire,together with two subclasses to represent a Full-time Staff and a Part-
time Staff respectively
 * @version 0.1
 * @author Pratik
 */
//declaring the attributes name
public class FullTimeStaffHire extends StaffHire{
	private int salary;
	private int workingHour;
	private String staffName;
	private String joiningDate;
	private String qualification;
	private String appointedBy;
	private boolean joined;
	//the constructor class of the class FullTimeStaffHire
	public FullTimeStaffHire(int vacancyNumber, String jobType,String designation,int
salary,int workingHour){
		super(designation,jobType,vacancyNumber);
		this.salary=salary;
		this.workingHour=workingHour;
		this.staffName="";
		this.joiningDate="";
		this.qualification="";
		this.appointedBy="";
		this.joined=false;
	}
	 //method to appoint the staff if the staff has not joined
	public void hiring(String staffName,String joiningDate,String qualification,String
appointedBy){
		if (this.joined==false){
		   this.staffName=staffName;
```

```
                this.joiningDate=joiningDate;
                this.qualification=qualification;
                this.appointedBy=appointedBy;
                this.joined=true;
                System.out.println("Staff Hired");
            }else{
            System.out.println("The staff has already been hired whose name is
"+this.staffName+ " and the date that they joined is"+this.joiningDate);
            }
        }
        //declaring setter method for salary
        public void setSalary(int salary){
            if (this.joined==false){
                this.salary=salary;
            }else{
                System.out.println("The staff has already joined. Therefore,you cannot salary
change.");
            }
        }
        //declaring setter method for workingHour
        public void setWorkingHour(int workingHour){
            this.workingHour=workingHour;
        }
        // displaying the designation,jobtype and the vacancy number from the superclass StaffHire
and staffName, salary,working hour,,joining date, qualification and appointed by from above if
joined is true
        public void display(){
            System.out.println("------------------Full-Time-Staff------------------");
            super.display();
            if (this.joined==true){
                System.out.println("Staffname: "+this.staffName);
                System.out.println("Salary: "+this.salary);
                System.out.println("Working Hour: "+this.workingHour);
                System.out.println("Joining date: "+this.joiningDate);
```

```
                System.out.println("Qualification: "+this.qualification);
                System.out.println("Appointed by: "+this.appointedBy);
                System.out.println("--------------------------------------------------");
            }
        }
        //declaring getter method for all attributes
        public int getSalary() {
            return salary; //The getSalary() method returns the value of salary as int datatype.
        }
        public int getWorkingHour() {
            return workingHour;//The getWorkingHour() method returns the value of working hour as
int datatype.
        }
        public String getStaffName() {
            return staffName; //The getStaffName() method returns the value of staff name as string
datatype.
        }
        public String getJoiningDate() {
            return joiningDate; //The getJoiningDate() method returns the value of joining date as
string datatype.
        }
        public String getQualification() {
            return qualification;//The getQualification() method returns the value of qualification as
string datatype.
        }
        public String getAppointedBy() {
            return appointedBy;//The getAppointedBy() method returns the value of Appointed By as
string datatype.
        }
        public boolean getJoined() {
            return joined;//The getJoined() method returns the value of joined as boolean datatype.
        }
}
```

Code for class PartTimeStaffHire

```
/**
 * This application is designed to
 * represent Staff Hire,together with two subclasses to represent a Full-time Staff and a Part-
time Staff respectively
 * @version 0.1
 * @author Pratik
 */
//declaring the attributes name
public class PartTimeStaffHire extends StaffHire{
    private int workingHour;
    private int wagesPerHour;
    private String staffName;
    private String joiningDate;
    private String qualification;
    private String appointedBy;
    private String shifts;
    private boolean joined;
    private boolean terminated;
    //the constructor class of the class PartTimeStaffHire
    public PartTimeStaffHire(int vacancyNumber,String designation,String jobType,int
workingHour,int wagesPerHour,String shifts){
        super(designation,jobType,vacancyNumber);
        this.workingHour=workingHour;
        this.wagesPerHour=wagesPerHour;
        this.shifts=shifts;
        staffName="";
        joiningDate="";
        qualification="";
        appointedBy="";
        this.joined=false;
        this.terminated=false;
    }
    //declaring getter method for all attributes
```

```
    public int getWorkingHour() {
        return workingHour ;//The getWorkingHour() method returns the value of working hour as
int datatype.
    }
    public int getWagesPerHour() {
        return wagesPerHour;//The getWagesPerHour() method returns the value of wages per
hour as int datatype.
    }
    public String getStaffName() {
        return staffName;//The getStaffName() method returns the value of staff Name as String
datatype.
    }
    public String getJoiningDate() {
     return joiningDate; //The getWorkingHour() method returns the value of working hour as int
datatype.
    }
    public String getQualification() {
     return qualification;//The getQualification() method returns the value of qualification as string
datatype.
    }
    public String getAppointedBy() {
     return appointedBy;//The getAppointed() method returns the value of appointed by  as string
datatype.
    }
    public String getShifts() {
     return shifts;//The getShifts() method returns the value of shifts as string datatype.
    }
    public boolean getJoined() {
     return joined;//The getJoined() method returns the value of joined as boolean datatype.
    }
    public boolean getTerminated() {
     return terminated;//The getTerminated() method returns the value of terminated as boolean
datatype.
    }
```

```
    //declaring setter method for shifts
    public void setShifts(String shifts){
        if (joined==false){
        this.shifts=shifts;//sets the new value to shifts if joined is false
        }else{
        System.out.println("Staff has already been hired.Shift cannot be changed.");
        }
    }
      //method to appoint the staff if the staff has not joined
    public void hiring(String staffName,String joiningDate,String qualification,String appointedBy){
            if (this.joined==false){
                    this.staffName=staffName;
                    this.joiningDate=joiningDate;
                    this.qualification=qualification;
                    this.appointedBy=appointedBy;
                    this.joined=true;
                    System.out.println("Staff Hired");
            }else{
                   System.out.print(this.staffName+" has already joined on
"+this.joiningDate+"having"+this.qualification+"who was appointed by"+this.appointedBy);
            }
    }
    //method to terminate the staff name if terminated is false
    public void terminate(){
        if (terminated==false){
            this.staffName="";
            this.joiningDate="";
            this.qualification="";
            this.appointedBy="";
            this.joined=false;
            this.terminated=true;
            System.out.println("The staff has been terminated");
        }else{
            System.out.println("The staff has already been terminated");
```

```
        }
    }
    // displaying the designation,jobtype and the vacancy number from the superclass StaffHire
and staff name, wagesperhour,workingHour,joiningDate,qualification,appointedBy and total
income from above if joined is true
    public void display(){
            System.out.println("------------------Part-Time-Staff------------------");
        super.display();
        if (this.joined==true){
            float totalincome;
            totalincome=wagesPerHour*workingHour;
            System.out.println("Staff Name: "+staffName);
            System.out.println("Wages Per Hour: "+wagesPerHour);
            System.out.println("Working Hour: "+workingHour);
            System.out.println("Joined Date: "+joiningDate);
            System.out.println("Qualification: "+qualification);
            System.out.println("Appointed By: "+appointedBy);
              System.out.println("Shift: "+shifts);
            System.out.println("Income per day: "+totalincome);
            System.out.println("--------------------------------------------------");
        }
    }
}
```