LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CS4001NA Programming (Computing Group)

Assessment Weightage & Type

50% Individual Coursework

Year and Semester

2020 Autumn

Student Name: Prashanna GC

Group: C12

London Met ID: 19031368

College ID: NP01CP4A190249

Assignment Due Date: 17th APR, 2020

Assignment Submission Date: 21th APR, 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.

Contents

LIST OF FIGURES 3
TABLES OF TABLE 4
INTRODUCTION 4
CLASS DIAGRAM 6
PSEUDOCODE: 7
1) METHOD 1 7
2) METHOD 2 19
3) METHOD 3 26
METHOD DESCRIPTION: 26
1) CREATE GUI FORM 26
2) ACTION PERFORMED METHOD 27
3) MAIN METHOD 27
TESTING 1 27
1) COMPILATION OF PROGRAM I N COMMAND PROMPT 27
TESTING 2 29
1) ADD VACANCY FOR FULL TIME STAFF HIRE 29
2) APPOINT FULL TIME STAFF 31
3) ADD VACANCY FOR PART TIME STAFF HIRE 33
4) APPOINT FULL TIME STAFF 34
5) TERMINATE PART TIME STAFF 37
TESTING 3 38
1) EMPTY FIELDS FOR FULLTIMESTAFFHIRE 38
2) INVALID VACANCY FOR FULLTIMESTAFFHIRE 39
3) EMPTY FIELDS APPOINTED FOR FULLTIMESTAFFHIRE 41
4) INVALID VACANCY APPOINTED FOR FULLTIMESTAFFHIRE 42
5) DIFFERENT / UN-ADDED VACANCY APPOINTED FOR FULLTIMESTAFFHIRE 44
6) EMPTY FIELDS FOR PARTTIMESTAFFHIRE 45
7) INVALID VACANCY FOR PARTTIMESTAFFHIRE 47
8) EMPTY FIELDS APPOINTED FOR PARTTIMESTAFFHIRE 48
9) INVALID VACANCY APPOINTED FOR PARTTIMESTAFFHIRE 50
10) DIFFERENT / UN-ADDED VACANCY APPOINTED FOR PARTTIMESTAFFHIRE 51
11) VACANCY FROM FULLTIMESTAFFHIRE APPOINTED IN PARTTIMESTAFFHIRE 53
12) VACANCY FROM PARTTIMESTAFFHIRE APPOINTED IN FULLTIMESTAFFHIRE 55
ERROR DETECTION - CORRECTION 57
1) LOGICAL ERROR 57

2) RUNTIME ERROR 58
3) SYNTAX ERROR 59
CONCLUSION 60
REFERENCES 61
APPENDIX 61
1) APPENDIX 1 61
CREATING ING CLASS 61
2) APPENDIX 2 76

LIST OF FIGURES

Figure 1: Compiling the program in cmd 27
Figure 2: Running the program in cmd 28
Figure 3: Result after running the program in cmd 28
Figure 4 : Inserting values in text field of Full Time Staff Hire 29
Figure 5: Inserted values and message box for Full Time Staff Hire 30
Figure 6: Vacancy added message box for Full Time Staff Hire 30
Figure 7: Inserting values to appoint staff for Full Time Staff Hire 31
Figure 8: Inserted values and message box Full Time Staff Hire 32
Figure 9: Staff appointed message box for Full Time Staff Hire 32
Figure 10: Inserting values to add vacancy in Part Time Staff Hire 33
Figure 11: Inserted values and vacancy added message box for Part Time Staff Hire 33
Figure 12: Vacancy added message box for Part Time Staff Hire 34
Figure 13: Inserting values to appoint staff in Part Time Staff Hire 35
Figure 14: Inserted values and message box for staff appoint in Part Time Staff Hire 35
Figure 15: Staff appointed message box for Part Time Staff Hire 36
Figure 16: Inserting values to terminate Part Time Staff Hire 37
Figure 17: Inserted values and message box of staff terminated in Part Time Staff Hire 37
Figure 18: Staff terminated message box 38
Figure 19: Empty fields for FullTime Staff Hire 39
Figure 20: Message box for empty vacancy in Full Time Staff Hire 39
Figure 21: Invalid vacancy for Full Time Staff Hire 40
Figure 22: Message box for invalid vacancy in Full Time Staff Hire 40
Figure 23: Empty fields appointed for Full Time Staff Hire 41
Figure 24: Message box for appointing empty fields in FullTimeStaffHire 41
Figure 25: Invalid fields appointed for Full Time Staff Hire 42
Figure 26: Invalid fields appointed and message box for Full Time Staff Hire 43
Figure 27: Message box for invalid fields appointed for Full Time Staff Hire 43
Figure 28: Different or Un-added vacancy appointed for Full Time Staff Hire 44
Figure 29: Message box for appointing un-added vacancy for Full Time Staff Hire 44
Figure 30: Empty fields for Part Time Staff Hire 46
Figure 31: Message box for empty fields in Part Time Staff Hire 46
Figure 32: Invalid vacancy for Part Time Staff Hire 47
Figure 33: Message box for Invalid vacancy in Part Time Staff Hire 47
Figure 34: Empty fields appointed for part Time Staff Hire 49
Figure 35: Message box for empty fields appointed for Part Time Staff Hire 49

Figure 36: Invalid fields appointed for Part Time Staff Hire 50
Figure 37: Message box for invalid fields appointed in Part Time Staff Hire 50
Figure 38: Different or Un-added vacancy appointed for Part Time Staff Hire 52
Figure 39: Message box for appointing un-added vacancy for Part Time Staff Hire 52
Figure 40: Inserting vacancy from Full Time Staff Hire in Part Time Staff Hire 53
Figure 41: Message box for inserting vacancy from Full Time Staff Hire in Part Time Staff Hire 54
Figure 42: Inserting vacancy from Part Time Staff Hire in Full Time Staff Hire 55
Figure 43: Inserting vacancy from Part Time Staff Hire in Full Time Staff Hire with the message box 56
Figure 44: Message box for inserting vacancy from Part Time Staff Hire in Full Time Staff Hire 56
..... Figure 45: Unsolved logical error 58
Figure 46: Solved logical error 58
Figure 47: Unsolved runtime error 59
Figure 48: Solved runtime error 59
Figure 49: Unsolved syntax error 60
Figure 50: Solved syntax error 60

TABLES OF TABLE

Table 1: Testing compilation of program in command prompt 29
Table 2: Testing add vacancy button for Full Time Staff Hire 31
Table 3: Testing appoint button for Full Time Staff Hire 32
Table 4: Testing add vacancy button for Part Time Staff Hire 34
Table 5: Testing appoint button for Part Time Staff Hire 36
Table 6: Testing terminate button 38
Table 7: Testing add vacancy button for empty fields in Full Time Staff Hire 39
Table 8: Testing add vacancy button for invalid fields in Full Time Staff Hire 41
Table 9: Testing appoint button for empty fields in Full Time Staff Hire 42
Table 10: Testing appointed button for invalid fields in Full Time Staff Hire 43
Table 11: Testing appoint button for un-added vacancy in Full Time Staff Hire 45
Table 12: Testing add vacancy button for empty fields in Part Time Staff Hire 47
Table 13: Testing add vacancy button for invalid fields in Part Time Staff Hire 48
Table 14: Testing appoint button for empty fields in Part Time Staff Hire 50
Table 15: Testing appointed button for invalid fields in Part Time Staff Hire 51
Table 16: Testing appoint button for un-added vacancy in Part Time Staff Hire 53
Table 17: Testing appoint button in Part Time Staff Hire by adding vacancy from Full Time Staff Hire 54
Table 18: Testing appoint button in Full Time Staff Hire by adding vacancy from Part Time Staff Hire 57

INTRODUCTION

Java is a high-level programming language developed by Sun Microsystems. It was originally designed for developing programs for set-top boxes and handheld devices, but later became a popular choice for creating web application. (Productions, 2020)

BlueJ is an application that allows you to make Java programs in easiest way as possible. It is simple, designed for teaching, interactive, portable, mature, and innovative. It has simpler interface than professional applications such as NetBeans or Eclipse. It is a popular textbook designed for teaching. It also allows us to interact with objects. It runs on Mac, Windows, and Linux etc. without installation from a USB. It has many other features that are not seen before in other IDEs. BlueJ was basically worked to help with user for education purpose on object oriented programming. (Gosling, 2019) The main objective of this coursework is to learn how make a proper GUI for a program.

According to our course work, we have four different classes of different variables. ING class, StaffHire class, FullTimeStaffHire class and PartTimeStaffHire class are the different classes created for this course work.
The ING class is the parent class. It consist of two methods where one of them consist of the GUI and the other method have the action for the implementation of buttons from the GUI.

The StaffHire class consists of three variables which are designation which has string data type, job_type which also has string data type and vacancy_number which has integer data type. It also has method like getDesignation, getJob_type, getVacancy_number that return the value to its corresponding variable. It also has method like setDesignation, setJob_type, setVacancy_number that assigns value to the variable.

The FullTimeStaffHire class consists of various attributes which are salary, Working_Hour, Staff_Name, Joining_Date, Qualification, Appointed_By and Joined. Each attribute have different data type. Salary and Working_Hour have integer data type and Joined have Boolean data type whereas other have string data type. It has methods like getSalary, getWorking_Hour, getStaff_Name, getJoining_Date, getQualification, getAppointed_By, getJoined that returns the value to its corresponding variable. It also has method like setSalary that assign salary if the condition is met, setworkinhour that assigns working hour if condition is met and display that display the information that are stated.

The PartTimeStaffHire class have attributes such as Working_Hour, WagesPerHour, StaffName, Joining_Date, Qualification, Appointed_By, Shifts, Joined and Terminated. Each attribute have different data type. It has method like getWagesPerHour, getStaff_Name, getJoining_Date, getQualfication, getAppointed_By, getShifts, getJoined, getTerminated that returns values to its corresponding variable. It also has methods like setWorkingShifts that assigns shift if the condition is met, Hire_partTimeSatff that takes data on the staff if the condition is met, terminatestaff that shows whether the staff is terminated or not and displaydetails that displays the information that are stated.

CLASS DIAGRAM

ING
- frame1 : JFrame - panel1 : JPanel - heading1 : JLabel - heading2 : JLabel - write1 : JLabel - write2 : JLabel - write3 : JLabel - write4 : JLabel - write5 : JLabel - write6 : JLabel - write7 : JLabel - write8 : JLabel - write9 : JLabel - write10 : JLabel -type1:JLabel -type2:JLabel -type3:JLabel -type4:JLabel -type5:JLabel -type6:JLabel -type7:JLabel -type8:JLabel -type9:JLabel -type10:JLabel -type11:JLabel -type12:JLabel -box1 : JTextField -box2 : JTextField -box4 : JTextField -box5 : JTextField -box6 : JTextField -box7 : JTextField -box9 : JTextField -box10 : JTextField -empty1 : JTextField -empty2 : JTextField

-empty4 : JTextField -empty5 : JTextField -empty6 : JTextField -empty7 : JTextField -empty9 : JTextField -empty10 : JTextField -empty11 : JTextField -empty12 : JTextField -box3 : JComboBox -empty3 : JComboBox - cmbYear : JComboBox -cmbYear2 : JComboBox - cmbMonth : JComboBox -cmbMonth2 : JComboBox - cmbDate : JComboBox -cmbDate2 : JComboBox -button1 : JButton -button2 : JButton -button3 : JButton -button4 : JButton -button5 : JButton -button6 : JButton -button7 : JButton
+ m1() : void + ActionPerformed() : void

PSEUDOCODE:

1) METHOD 1

```
START

        BUILD void m1()

                DO

                DECLARE CLASS VARIABLES

                JFrame frame1,
                JPanel panel1,
                JLabel
                heading1,heading2,write1,write2,write3,write4,write5,write6,write7,write8,
                write9,
                write10,type1,type2,type3,type4,type5,type6,type7,type8,type9,type10,type
                11,type12,
```

JTextFieldbox1,box2,box4,box5,box6,box7,box9,box10,empty1,empty2,empty4,empty5,empty6,empty7,empty9,empty10,empty11,empty12,
JComboBoxbox3,empty3,cmbYear,cmbMonth,cmbDay,cmbYear2,cmbMonth2,cmbDay2,
JButton button1, button2, button3, button4, button5, button6, button7

INITIALIZE JFrame frame1

INITIALIZE JPanel panel1

GIVE THE SIZE TO THE FRAME

GIVE THE LAYOUT TO THE FRAME

INITIALIZE JLabel heading1

GIVE THE SIZE AND POSITION

GIVE THE FONT

ADD (heading1) IN PANEL

INITIALIZE JLabel write1

GIVE THE SIZE AND POSITION

ADD (write1) IN PANEL

INITIALIZE JLabel write2

GIVE THE SIZE AND POSITION

ADD (write2) IN PANEL

INITIALIZE JLabel write3

GIVE THE SIZE AND POSITION

ADD (write3) IN PANEL

INITIALIZE JLabel write4

GIVE THE SIZE AND POSITION

ADD (write4) IN PANEL

INITIALIZE JLabel write5

GIVE THE SIZE (250,10,305,50)

ADD (write5) IN PANEL

INITIALIZE JLabel write6

GIVE THE SIZE AND POSITION

ADD (write6) IN PANEL

INITIALIZE JLabel write7

GIVE THE SIZE AND POSITION

ADD (write7) IN PANEL

INITIALIZE JLabel write8

GIVE THE SIZE AND POSITION)

ADD (write8) IN PANEL

INITIALIZE JLabel write9

GIVE THE SIZE AND POSITION

ADD (write9) IN PANEL

INITIALIZE JLabel write10

GIVE THE SIZE AND POSITION

ADD (write10) IN PANEL

CREATE THE NEW JTextField box1

GIVE THE SIZE AND POSITION

ADD (box1) IN PANEL

CREATE THE NEW JTextField box2

GIVE THE SIZE AND POSITION

ADD (box2) IN PANEL

CREATE THE NEW JComboBox box3 AND INITIALIZE THE VALUES

GIVE THE SIZE AND POSITION

ADD (box3) IN PANEL

CREATE THE NEW JTextField box4

GIVE THE SIZE AND POSITION

ADD (box4) IN PANEL

CREATE THE NEW JTextField box5

GIVE THE SIZE AND POSITION

ADD (box5) IN PANEL

CREATE THE NEW JTextField box6

GIVE THE SIZE AND POSITION

ADD (box6) IN PANEL

CREATE THE NEW JTextField box7

GIVE THE SIZE AND POSITION

ADD (box7) IN PANEL

CREATE THE NEW JComboBox cmbYear AND INITIALIZE THE VALUES

GIVE THE SIZE AND POSITION

ADD (cmbYear) IN PANEL

CREATE THE NEW JComboBox cmbMonth AND INITIALIZE THE VALUES

GIVE THE SIZE AND POSITION

ADD (cmbMonth) IN PANEL

CREATE THE NEW JComboBox cmbDay AND INITIALIZE THE VALUES

GIVE THE SIZE AND POSITION

ADD (cmbDay) IN PANEL

CREATE THE NEW JTextField box9

GIVE THE SIZE AND POSITION

ADD (box9) IN PANEL

CREATE THE NEW JTextField box10

GIVE THE SIZE AND POSITION

ADD (box10) IN PANEL

CREATE THE NEW JButton button1

GIVE THE SIZE AND POSITION

ADD ACTION LISTENER

ADD (button1) IN PANEL

CREATE THE NEW JButton button2

GIVE THE SIZE AND POSITION

ADD ACTION LISTENER

ADD (button2) IN PANEL

INITIALIZE JLabel heading2

GIVE THE SIZE AND POSITION

GIVE THE FONT

ADD (heading2) IN PANEL

INITIALIZE JLabel type1

GIVE THE SIZE AND POSITION

ADD (type1) IN PANEL

INITIALIZE JLabel type2

GIVE THE SIZE AND POSITION

ADD (type2) IN PANEL

INITIALIZE JLabel type3

GIVE THE SIZE AND POSITION

ADD (type3) IN PANEL

INITIALIZE JLabel type4

GIVE THE SIZE AND POSITION

ADD (type4) IN PANEL

INITIALIZE JLabel type5

GIVE THE SIZE (250,10,305,50)

ADD (type5) IN PANEL

INITIALIZE JLabel type6

GIVE THE SIZE AND POSITION

ADD (type6) IN PANEL

INITIALIZE JLabel type7

GIVE THE SIZE AND POSITION

ADD (type7) IN PANEL

INITIALIZE JLabel type8

GIVE THE SIZE AND POSITION)

ADD (type8) IN PANEL

INITIALIZE JLabel type9

GIVE THE SIZE AND POSITION

ADD (type9) IN PANEL

INITIALIZE JLabel type10

GIVE THE SIZE AND POSITION

ADD (type10) IN PANEL

INITIALIZE JLabel type11

GIVE THE SIZE AND POSITION

ADD (type11) IN PANEL

INITIALIZE JLabel type12

GIVE THE SIZE AND POSITION

ADD (type12) IN PANEL

CREATE THE NEW JTextField empty1

GIVE THE SIZE AND POSITION

ADD (empty1) IN PANEL

CREATE THE NEW JTextField empty2

GIVE THE SIZE AND POSITION

ADD (empty2) IN PANEL

CREATE THE NEW JComboBox empty3 AND INITIALIZE THE VALUES

GIVE THE SIZE AND POSITION

ADD (empty3) IN PANEL

CREATE THE NEW JTextField empty4

GIVE THE SIZE AND POSITION

ADD (empty4) IN PANEL

CREATE THE NEW JTextField empty5

GIVE THE SIZE AND POSITION

ADD (empty5) IN PANEL

CREATE THE NEW JTextField empty6

GIVE THE SIZE AND POSITION

ADD (empty6) IN PANEL

CREATE THE NEW JTextField empty7

GIVE THE SIZE AND POSITION

ADD (empty7) IN PANEL

CREATE THE NEW JComboBox cmbYear2 AND INITIALIZE THE VALUES

GIVE THE SIZE AND POSITION

ADD (cmbYear2) IN PANEL

CREATE THE NEW JComboBox cmbMonth2 AND INITIALIZE THE VALUES

GIVE THE SIZE AND POSITION

ADD (cmbMonth2) IN PANEL

CREATE THE NEW JComboBox cmbDay2 AND INITIALIZE THE VALUES

GIVE THE SIZE AND POSITION

ADD (cmbDay2) IN PANEL

CREATE THE NEW JTextField empty9

GIVE THE SIZE AND POSITION

ADD (empty9) IN PANEL

CREATE THE NEW JTextField empty10

GIVE THE SIZE AND POSITION

ADD (empty10) IN PANEL

CREATE THE NEW JTextField empty11

GIVE THE SIZE AND POSITION

ADD (empty11) IN PANEL

CREATE THE NEW JTextField empty12

GIVE THE SIZE AND POSITION

ADD (empty12) IN PANEL

CREATE THE NEW JButton button3

GIVE THE SIZE AND POSITION

ADD ACTION LISTENER

ADD (button3) IN PANEL

CREATE THE NEW JButton button4

GIVE THE SIZE AND POSITION

ADD ACTION LISTENER

ADD (button4) IN PANEL

CREATE THE NEW JButton button5

GIVE THE SIZE AND POSITION

ADD ACTION LISTENER

ADD (button5) IN PANEL

CREATE THE NEW JButton button6

```
            GIVE THE SIZE AND POSITION
            ADD ACTION LISTENER
            ADD (button6) IN PANEL

            CREATE THE NEW JButton button7
            GIVE THE SIZE AND POSITION
            ADD ACTION LISTENER
            ADD (button7) IN PANEL

            ADD PANEL IN FRAME
            GIVE FRAME RESIZABLE
            GIVE FRAME VISIBILITY

        ENDDO
END
```

2) METHOD 2

```
START

        BUILD void actionPerformed (ActionEvent e)
            DO
            IF (button1)
                TRY
```

```
                    GIVE box1 as int data type
                    GIVE box2 as string data type
                    GIVE box3 as string data type
                    GIVE box4 as int data type
                    GIVE box5 as int data type
        FOR
        IF   (isDuplicateVacancy equals true)
        BREAK
        IF    (isDuplicateVacancy equals false)
                    CREATE FullTimeStaffHire full_time_obj;
                    ADD (full_time_obj) IN ARRAYLIST
                    DISPLAY THE DIALOG MESSAGE
        ELSE
                    DISPLAY THE DIALOG MESSAGE
                    CATCH (Exception ee)
                    DISPLAY THE DIALOG MESSAGE
    ENDDO
    DO
        IF (button3)
        TRY
                    GIVE empty1 as int data type
                    GIVE empty2 as string data type
                    GIVE empty3 as string data type
                    GIVE empty6 as int data type
                    GIVE empty5 as int data type
                    GIVE empty4 as string data type
```

```
        FOR
        IF   (isDuplicateVacancy equals true)
        BREAK
        IF    (isDuplicateVacancy equals false)
                CREATE PartTimeStaffHire part_time_obj;
                ADD (part_time_obj) IN ARRAYLIST
                DISPLAY THE DIALOG MESSAGE
        ELSE
                DISPLAY THE DIALOG MESSAGE
                CATCH (Exception ee)
                DISPLAY THE DIALOG MESSAGE
    ENDDO
    DO
        IF (button2)
        TRY
                GIVE box10 as int data type
                GIVE box6 as string data type
                GIVE box7 as string data type
                GIVE cmbYear as string data type
                GIVE cmbMonth as string data type
                GIVE cmbDay as string data type
                GIVE box9 as string data type

        FOR
        IF (vacancyFound equals true)
        IF (ob instanceof FullTimeStaffHire)
```

```
        CREATE FullTimeStaffHire ob;
        IF (h.getJoined () equals true)
                DISPLAY THE DIALOG MESSAGE
        ELSE
                CALL FullTimeStaff_Hire
                DISPLAY THE DIALOG MESSAGE
        BREAK
        ELSE
                DISPLAY THE DIALOG MESSAGE
        BREAK
        IF (vacancyFound equals false)
            DISPLAY THE DIALOG MESSAGE
        CATCH (Exception ee)
            DISPLAY THE DIALOG MESSAGE
    ENDDO
    DO
        IF (button4)
        TRY
            GIVE empty11 as int data type
            GIVE empty7 as string data type
            GIVE empty10 as string data type
            GIVE cmbYear2 as string data type
            GIVE cmbMonth2 as string data type
            GIVE cmbDay2 as string data type
            GIVE empty9 as string data type
```

```
        FOR
        IF (vacancyFound equals true)
        IF (ob_p instanceof PartTimeStaffHire)
        CREATE PartTimeStaffHire ob_p;
        IF (g.getJoined () equals true)
            DISPLAY THE DIALOG MESSAGE
        ELSE
            CALL Hire_PartTimeStaff
            DISPLAY THE DIALOG MESSAGE
        BREAK
        ELSE
            DISPLAY THE DIALOG MESSAGE
        BREAK
        IF (vacancyFound equals false)
            DISPLAY THE DIALOG MESSAGE
        CATCH (Exception ee)
            DISPLAY THE DIALOG MESSAGE
    ENDDO

    DO
        IF (button5)
        TRY
            GIVE empty12 as int data type
        FOR
        IF (vacancyFound equals true)
        IF (ob_t instanceof PartTimeStaffHire)
```

```
            CREATE PartTimeStaffHire ob_t;
            IF (l.getJoined () equals false)

                    DISPLAY THE DIALOG MESSAGE

            BREAK

            ELSE

            CALL Terminate_Staff

                    DISPLAY THE DIALOG MESSAGE

            BREAK

            ELSE

                    DISPLAY THE DIALOG MESSAGE

            BREAK

            IF (vacancyFound equals false)

                    DISPLAY THE DIALOG MESSAGE

            CATCH (Exception ee)

                    DISPLAY THE DIALOG MESSAGE

    ENDO


DO

            IF (button6)

                    GIVE box1 as “ ” value

                    GIVE box2 as “ ” value

                    GIVE box3 as INDEX “  0 ”

                    GIVE box4 as “ ” value

                    GIVE box5 as “ ” value

                    GIVE box6 as “ ” value

                    GIVE box7 as “ ” value
```

```
                GIVE cmbYear as INDEX "  0 "
                GIVE cmbMonth as INDEX "  0 "
                GIVE cmbDay as INDEX "  0 "
                GIVE box9 as " " value
                GIVE box10 as " " value
                GIVE empty1 as " " value
                GIVE empty2 as " " value
                GIVE empty3 as INDEX "  0 "
                GIVE empty4 as " " value
                GIVE empty5 as " " value
                GIVE empty6 as " " value
                GIVE empty7 as " " value
                GIVE cmbYear2 as INDEX "  0 "
                GIVE cmbMonth2 as INDEX "  0 "
                GIVE cmbDay2 as INDEX "  0 "
                GIVE empty9 as " " value
                GIVE empty10 as " " value
                GIVE empty11 as " " value
                GIVE empty12 as " " value
        ENDDO
        DO
            IF (button7)
            FOR
                IF (var instanceof FullTimeStaffHire)
                CREATE FullTimeStaffHire x;
                CALL x.Display();
```

```
                    IF (var instanceof PartTimeStaffHire)
                    CREATE PartTimeStaffHire z;
                    CALL z.Display();
            ENDDO
END
```

3) METHOD 3

```
START
BUILD static void main()
    DO
        CREATE ING b;
        CALL b.m1();
    ENDDO
END
```

METHOD DESCRIPTION:

1) CREATE GUI FORM

This method consists of all the elements required to make the GUI form. The elements JLabel, JTextField, JButton and JComboBox are used here. A frame is created where these elements are added. The size of the frame is initialized and the layout is set to zero. The buttons and the actionListener if each of the buttons is also added in the frame in this method.

2) ACTION PERFORMED METHOD

This method consists of actions of all the buttons added in the frame. The first Add button for FullTimeStaffHire adds the input values of the vacancy number, designation, job type, salary and working hour per day to the array list of StaffHire class. And, also the second Add button for PartTimeStaffHire adds the input values of the vacancy number, designation, job type, working hours per day, wages per hour and shift also to the array list of StaffHire class. Whereas the two Appoint button for both FullTimeStaffHire and PartTimeStaffHire appoints the appropriate staff from the array list respectively. A Terminate button is create for only PartTimeStaffHire which is used to terminate the staff appointed for PartTimeStaffHire. And, a Clear button to clear all the values entered in the text fields and a Display button to display all the information relating to the appropriate class in this method.

3) MAIN METHOD

The main method has an object of the main INGNepal class and is called here to run the codes.

TESTING 1

1) COMPILATION OF PROGRAM I N COMMAND PROMPT

Figure 1: Compiling the program in cmd

Figure 2: Running the program in cmd

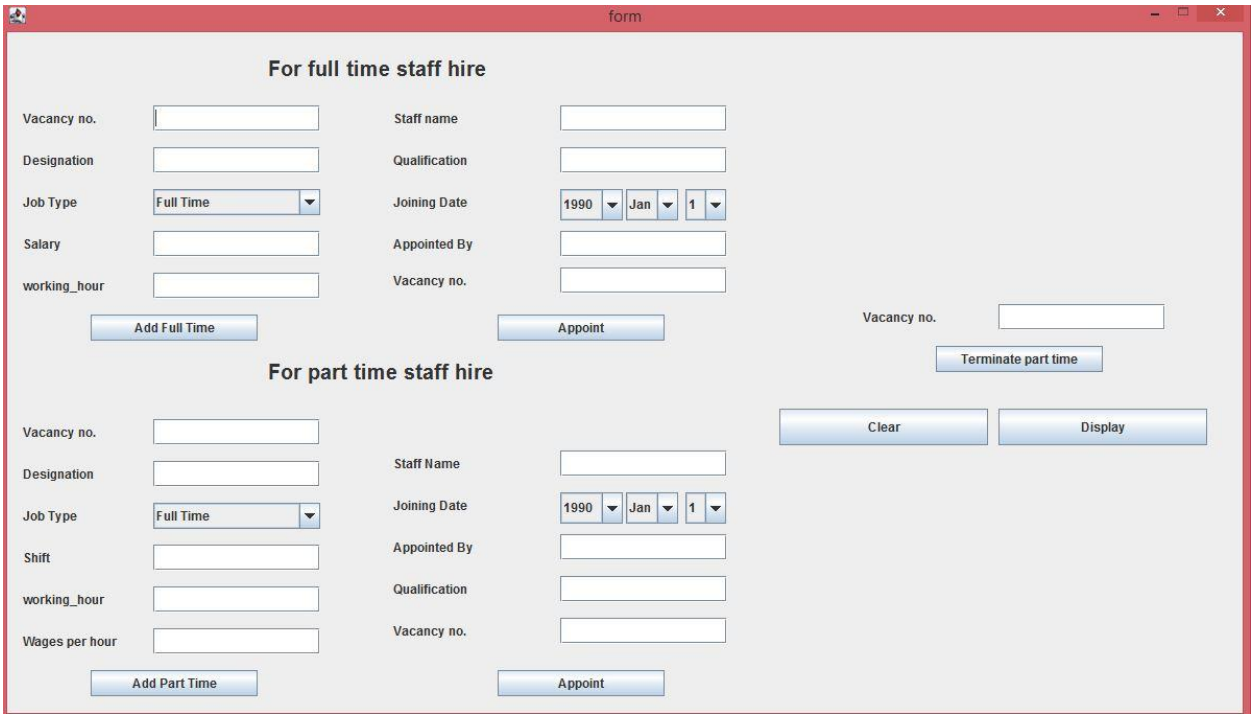


Figure 3: Result after running the program in cmd

TEST-1

Objective	To compile and run the program using command prompt

Action	The program was compiled and runned in command prompt
Expected Result	GUI form should be display.
Actual Result	GUI form was displayed.
Conclusion	Test pass.

Table 1: Testing compilation of program in command prompt

TESTING 2

1) ADD VACANCY FOR FULL TIME STAFF HIRE

Figure 4 : Inserting values in text field of Full Time Staff Hire

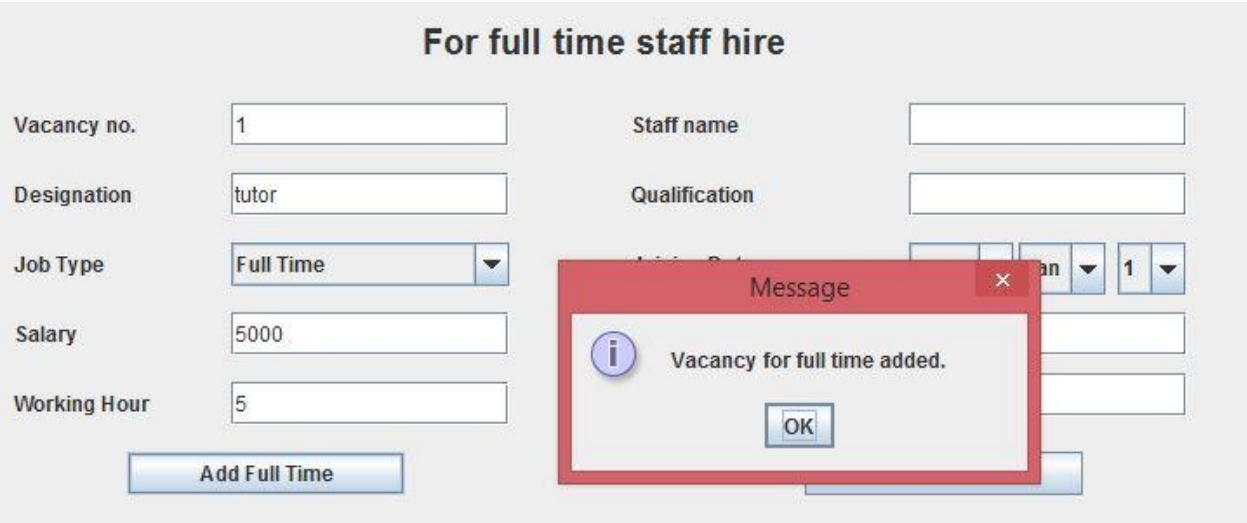


Figure 5: Inserted values and message box for Full Time Staff Hire

Figure 6: Vacancy added message box for Full Time Staff Hire

Test: 1

Objective	The input values of the vacancy number, designation, job type, salary and working hour per day are used to create a new object of type FullTimeStaffHire which is added to an array list of StaffHire class.
Action	The value for FullTimeStaffHire are assigned : Vacancy number : 1 Designation: " tutor " Job type: " Full Time " Salary: 5000 Working hour per day: 5

Expected Result	A pop up message box should appear informing about the vacancy for FullTimeStaffHire added in the array list.
Actual Result	A pop up message box appeared informing about the vacancy added for FullTimeStaffHire in the array list.
Conclusion	Test pass.

Table 2: Testing add vacancy button for Full Time Staff Hire

2) APPOINT FULL TIME STAFF

Figure 7: Inserting values to appoint staff for Full Time Staff Hire

Figure 8: Inserted values and message box Full Time Staff Hire

Figure 9: Staff appointed message box for Full Time Staff Hire

Test: 2

Objective	The input value of vacancy number is compared to the existing vacancy number in FullTimeStaffHire, and if valid vacancy number has been entered, it is used to appoint the appropriate staff from the list.
Action	The value for FullTimeStaffHire are assigned : Staff Name: " Prashan " Qualification: " graduate degree " Joining Date: " 1996-May-4 " Appointed By: " Mike " Vacancy Number: 1
Expected Result	A pop up message box should appear informing about the staff for FullTimeStaffHire from the array list has been hired.
Actual Result	A pop up message box appeared informing about the staff for FullTimeStaffHire from the array list has been hired.
Conclusion	Test pass.

Table 3: Testing appoint button for Full Time Staff Hire

3) ADD VACANCY FOR PART TIME STAFF HIRE

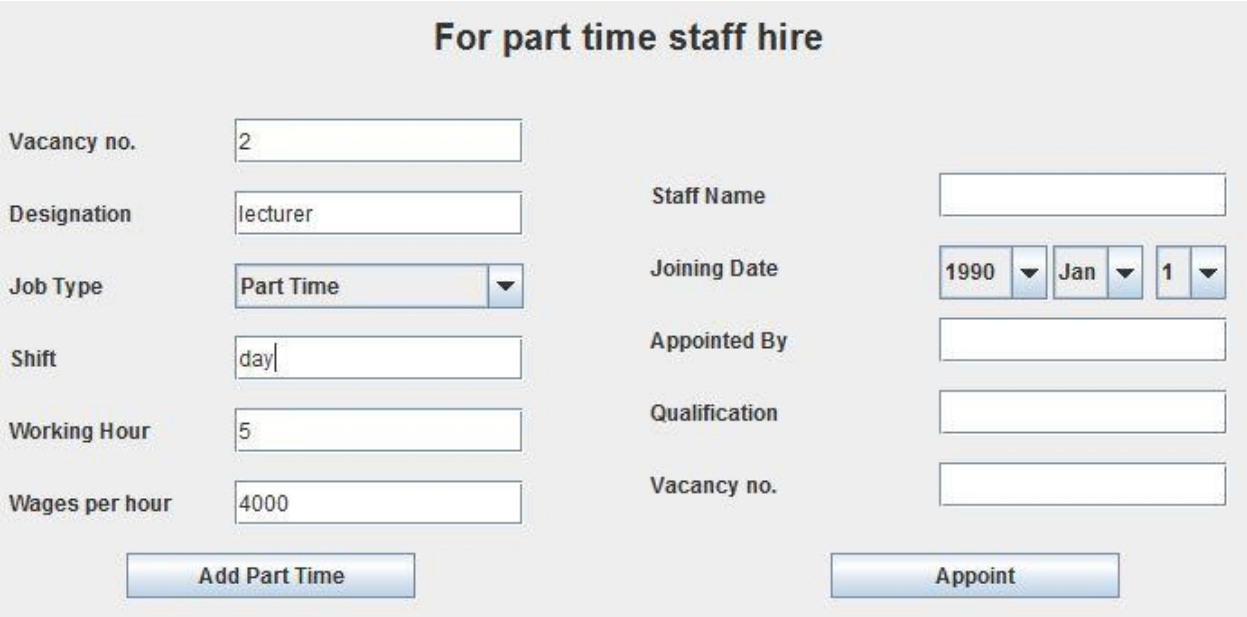


Figure 10: Inserting values to add vacancy in Part Time Staff Hire

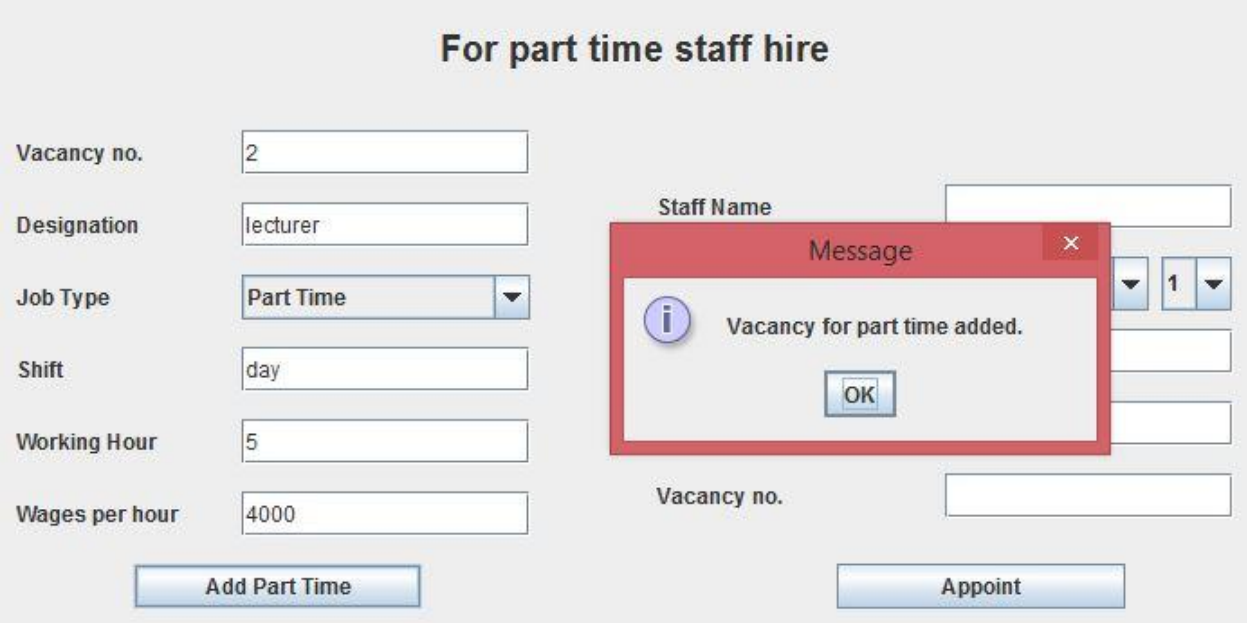


Figure 11: Inserted values and vacancy added message box for Part Time Staff Hire

Figure 12: Vacancy added message box for Part Time Staff Hire

Test: 3

Objective	The input values of the vacancy number, designation, job type, working hours per day, wages per hour and shift are used to create a new object of type PartTimeStaffHire which is added to an array list of StaffHire class.
Action	The value for PartTimeStaffHire are assigned : Vacancy number : 2 Designation: " lecturer " Job type: " Part Time " Working hour : 4000 Wages per hour: 5 Shift: " day "
Expected Result	A pop up message box should appear informing about the vacancy for PartTimeStaffHire added in the array list.
Actual Result	A pop up message box appeared informing about the vacancy added for PartTimeStaffHire in the array list.
Conclusion	Test pass.

Table 4: Testing add vacancy button for Part Time Staff Hire

4) APPOINT FULL TIME STAFF

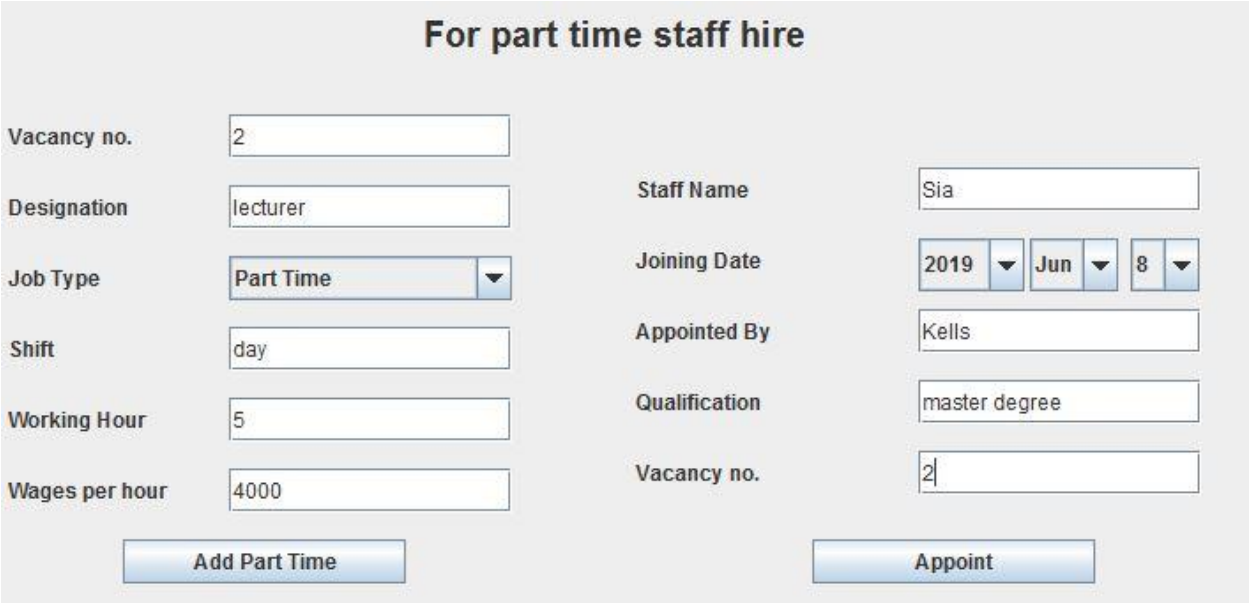


Figure 13: Inserting values to appoint staff in Part Time Staff Hire

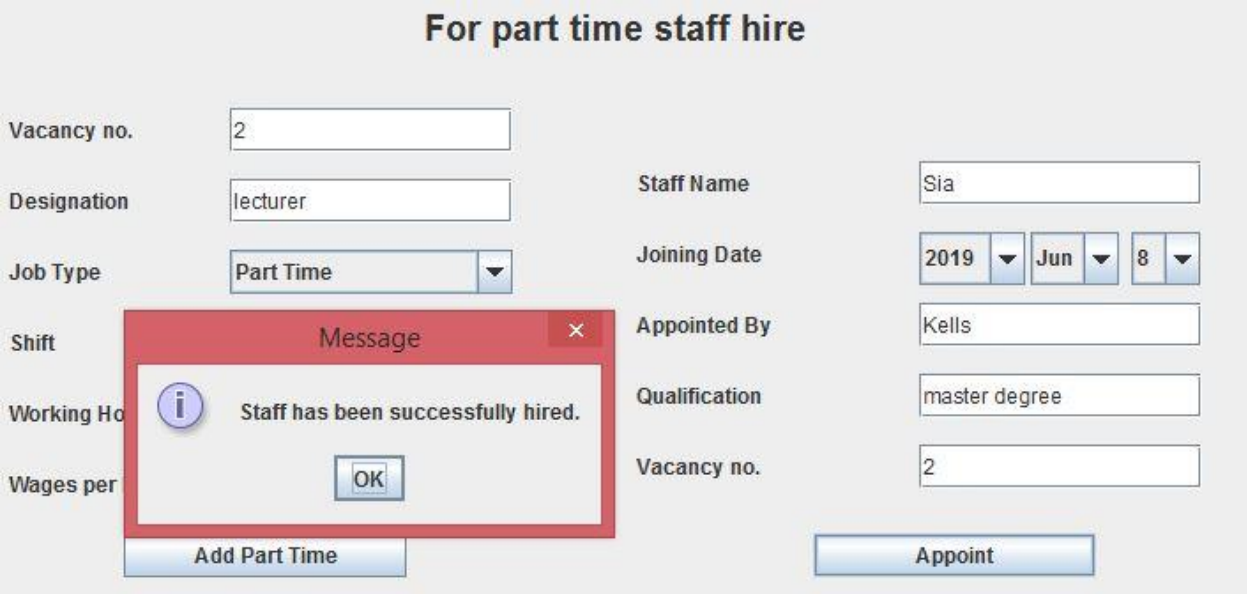


Figure 14: Inserted values and message box for staff appoint in Part Time Staff Hire

Figure 15: Staff appointed message box for Part Time Staff Hire

Test: 4

Objective	The input value of vacancy number is compared to the existing vacancy number in PartTimeStaffHire, and if valid vacancy number has been entered, it is used to appoint the appropriate staff from the list.
Action	The value for PartTimeStaffHire are assigned : Staff Name: " Sia " Qualification: " master degree " Joining Date: " 2019-Jun-8 " Appointed By: " Kells " Vacancy Number: 1
Expected Result	A pop up message box should appear informing about the staff for PartTimeStaffHire from the array list has been hired.
Actual Result	A pop up message box appeared informing about the staff for PartTimeStaffHire from the array list has been hired.
Conclusion	Test pass.

Table 5: Testing appoint button for Part Time Staff Hire

5) TERMINATE PART TIME STAFF

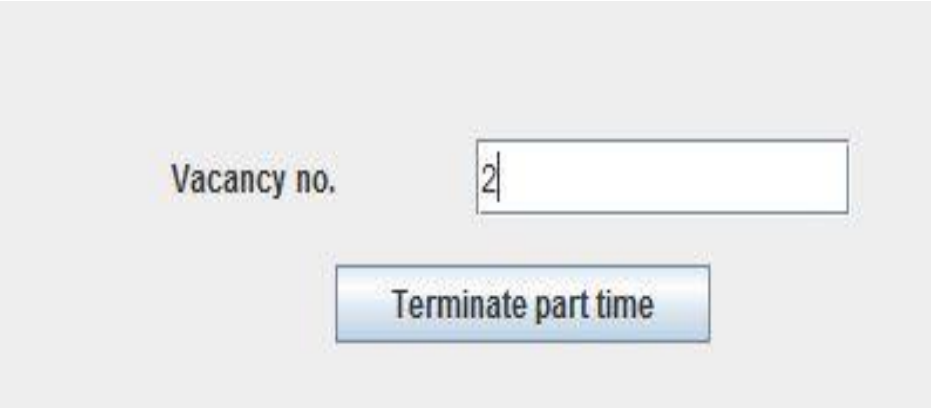


Figure 16: Inserting values to terminate Part Time Staff Hire

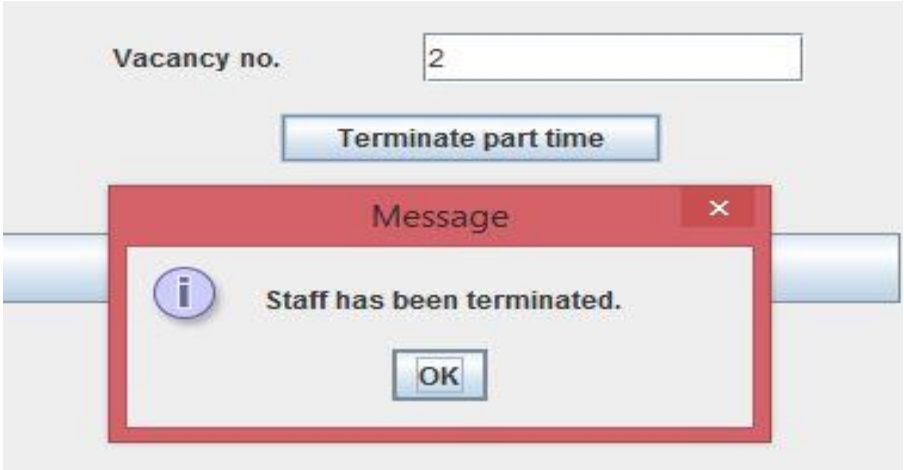


Figure 17: Inserted values and message box of staff terminated in Part Time Staff Hire

Figure 18: Staff terminated message box

Objective	The input value of the vacancy number is compared to the existing vacancy number in the list. If a valid value has been entered, it is used to terminate the appropriate part time staff from the array list of StaffHire.
Action	The vacancy number for PartTimeStaffHire is assigned : Vacancy Number: 2
Expected Result	A pop up message box should appear informing about the staff for PartTimeStaffHire from the array list has been terminated.
Actual Result	A pop up message box appeared informing about the staff for PartTimeStaffHire from the array list has been terminated.
Conclusion	Test pass.

Table 6: Testing terminate button

TESTING 3

1) EMPTY FIELDS FOR FULLTIMESTAFFHIRE

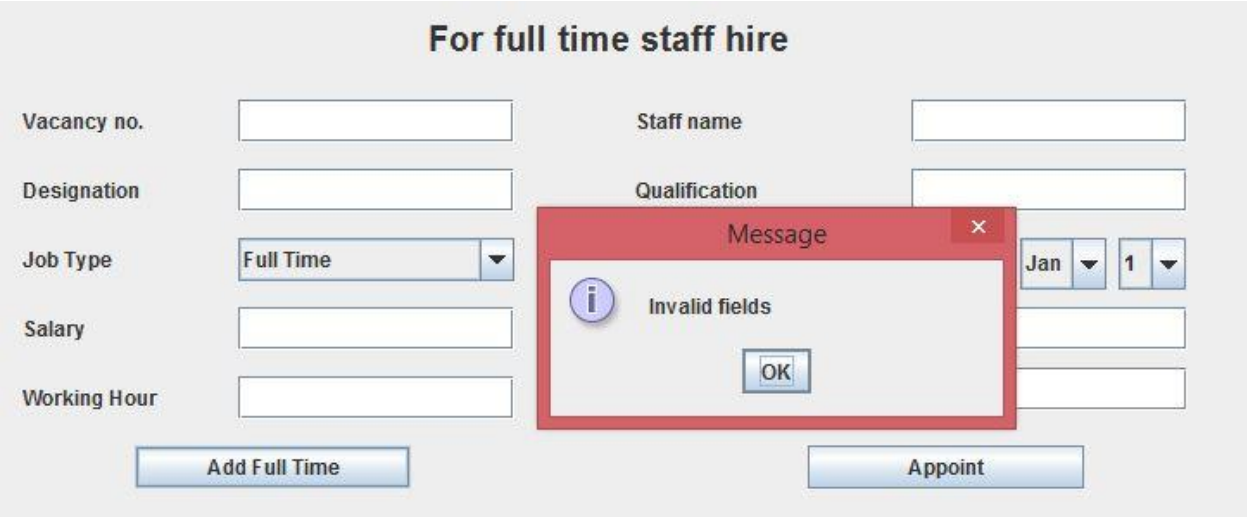


Figure 19: Empty fields for FullTime Staff Hire

Figure 20: Message box for empty vacancy in Full Time Staff Hire

Test-1

Objective	To get the appropriate dialog box when input fields for FullTimeStaffHire is left empty.
Action	The input fields for FullTimeStaffHire are assigned empty and vacancy is added.
Expected Result	A pop up message box should appear informing about the fields for FullTimeStaffHire to be invalid.
Actual Result	A pop up message box appeared informing about the fields for FullTimeStaffHire is invalid.
Conclusion	Test pass.

Table 7: Testing add vacancy button for empty fields in Full Time Staff Hire

2) INVALID VACANCY FOR FULLTIMESTAFFHIRE

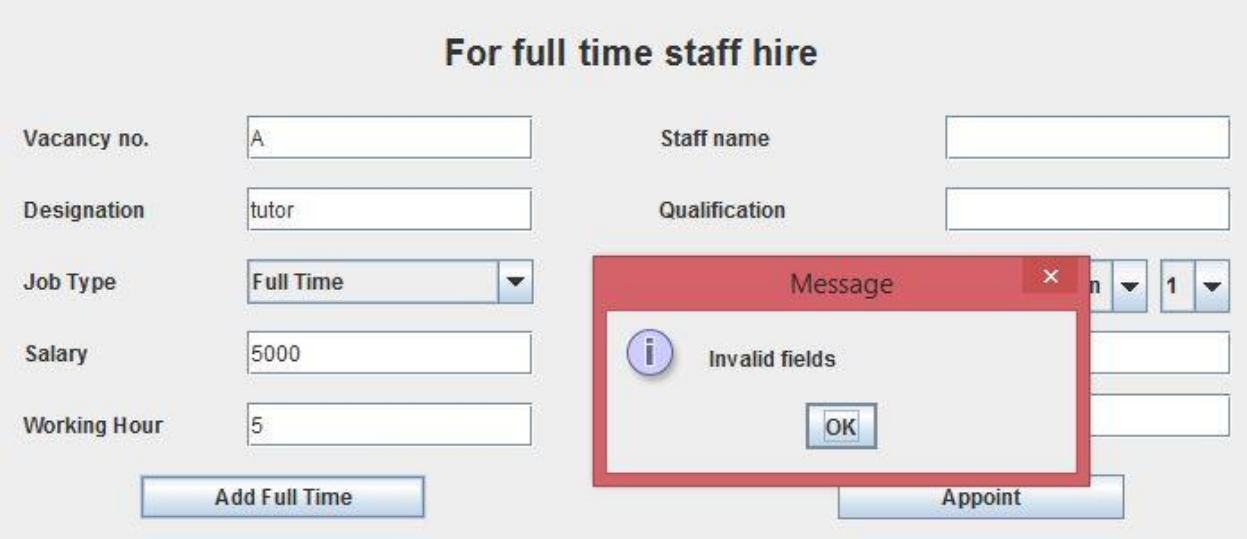


Figure 21: Invalid vacancy for Full Time Staff Hire

Figure 22: Message box for invalid vacancy in Full Time Staff Hire

Test-2

Objective	To get the appropriate dialog box when invalid vacancy is added for FullTimeStaffHire.
Action	Invalid vacancy for FullTimeStaffHire is assigned and vacancy is added.
Expected Result	A pop up message box should appear informing about the fields for FullTimeStaffHire to be invalid.

Actual Result	A pop up message box appeared informing about the fields for FullTimeStaffHire is invalid.
Conclusion	Test pass.

Table 8: Testing add vacancy button for invalid fields in Full Time Staff Hire

3) EMPTY FIELDS APPOINTED FOR FULLTIMESTAFFHIRE

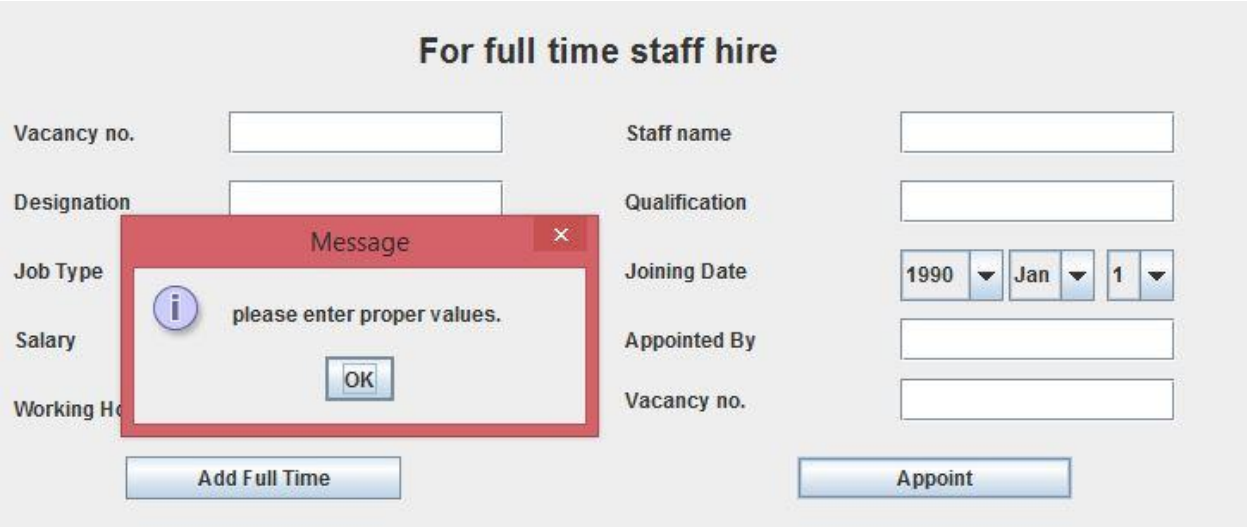


Figure 23: Empty fields appointed for Full Time Staff Hire

Figure 24: Message box for appointing empty fields in FullTimeStaffHire

Test-3

Objective	To get the appropriate dialog box when input fields for FullTimeStaffHire is left empty.
Action	The input fields for FullTimeStaffHire are assigned empty and vacancy is appointed.
Expected Result	A pop up message box should appear informing to enter proper values.
Actual Result	A pop up message box appeared informing to enter proper values.
Conclusion	Test pass.

Table 9: Testing appoint button for empty fields in Full Time Staff Hire

4) INVALID VACANCY APPOINTED FOR FULLTIMESTAFFHIRE

Figure 25: Invalid fields appointed for Full Time Staff Hire

Figure 26: Invalid fields appointed and message box for Full Time Staff Hire

Figure 27: Message box for invalid fields appointed for Full Time Staff Hire

Test-4

Objective	To get the appropriate dialog box when invalid vacancy is appointed for FullTimeStaffHire.
Action	Invalid vacancy for FullTimeStaffHire is assigned and vacancy is appointed.
Expected Result	A pop up message box should appear informing to enter proper values.
Actual Result	A pop up message box appeared informing to enter proper values.
Conclusion	Test pass.

Table 10: Testing appointed button for invalid fields in Full Time Staff Hire

5) DIFFERENT / UN-ADDED VACANCY APPOINTED FOR FULLTIMESTAFFHIRE

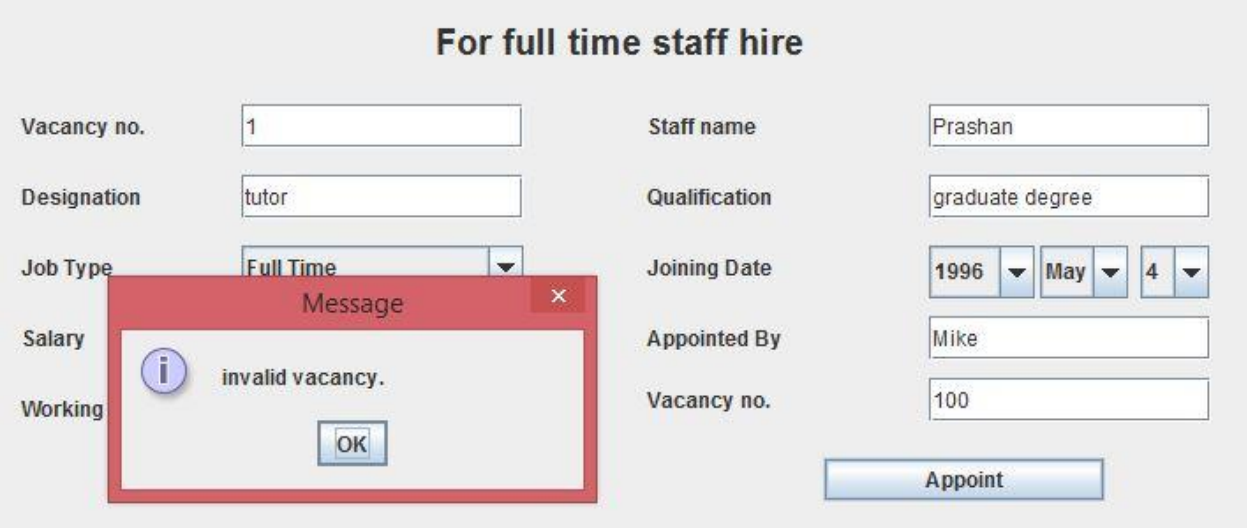


Figure 28: Different or Un-added vacancy appointed for Full Time Staff Hire

Figure 29: Message box for appointing un-added vacancy for Full Time Staff Hire

Test-5

Objective	To get the appropriate dialog box when un-added vacancy is appointed for FullTimeStaffHire
Action	Different vacancy for FullTimeStaffHire is assigned and vacancy is appointed.
Expected Result	A pop up message box should appear informing about invalid vacancy.
Actual Result	A pop up message box appeared informing about invalid vacancy.
Conclusion	Test pass.

Table 11: Testing appoint button for un-added vacancy in Full Time Staff Hire

6) EMPTY FIELDS FOR PARTTIMESTAFFHIRE

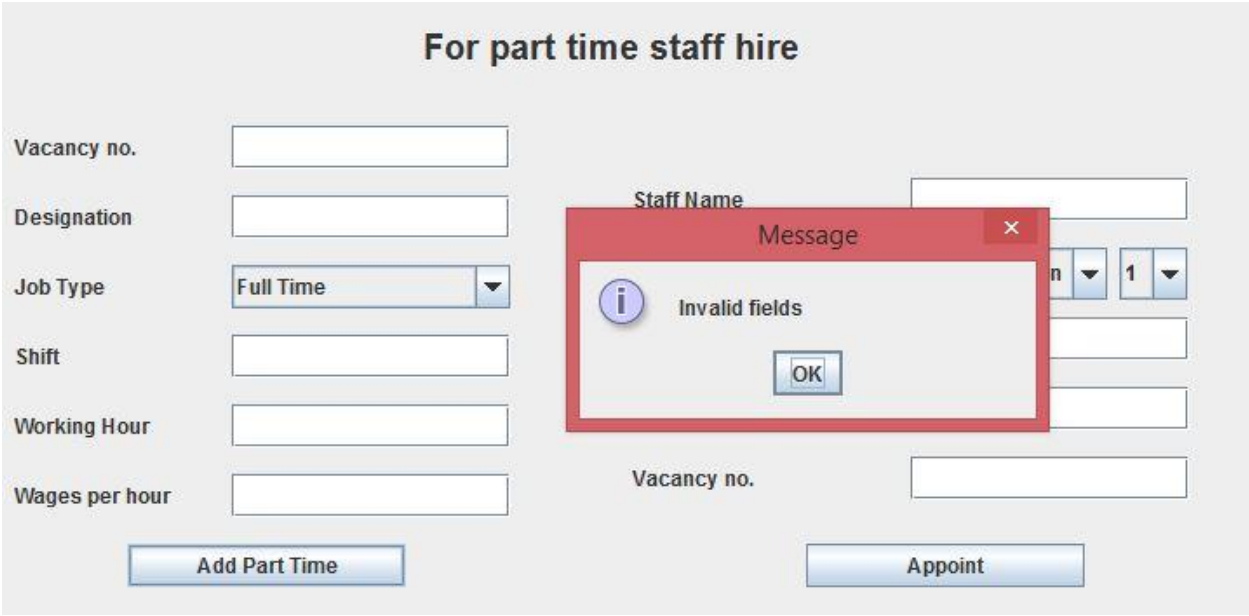


Figure 30: Empty fields for Part Time Staff Hire

Figure 31: Message box for empty fields in Part Time Staff Hire

Test-6

Objective	To get the appropriate dialog box when input fields for PartTimeStaffHire is left empty.
Action	The input fields for PartTimeStaffHire are assigned empty and vacancy is added.
Expected Result	A pop up message box should appear informing about the fields for PartTimeStaffHire to be invalid.

Actual Result	A pop up message box appeared informing about the fields for PartTimeStaffHire is invalid.
Conclusion	Test pass.

Table 12: Testing add vacancy button for empty fields in Part Time Staff Hire

7) INVALID VACANCY FOR PARTTIMESTAFFHIRE

Figure 32: Invalid vacancy for Part Time Staff Hire

Figure 33: Message box for Invalid vacancy in Part Time Staff Hire

Test-7

Objective	To get the appropriate dialog box when invalid vacancy is added for PartTimeStaffHire.
Action	Invalid vacancy for partTimeStaffHire is assigned and vacancy is added.
Expected Result	A pop up message box should appear informing about the fields for PartTimeStaffHire to be invalid.
Actual Result	A pop up message box appeared informing about the fields for PartTimeStaffHire is invalid.
Conclusion	Test pass.

Table 13: Testing add vacancy button for invalid fields in Part Time Staff Hire

8) EMPTY FIELDS APPOINTED FOR PARTTIMESTAFFHIRE

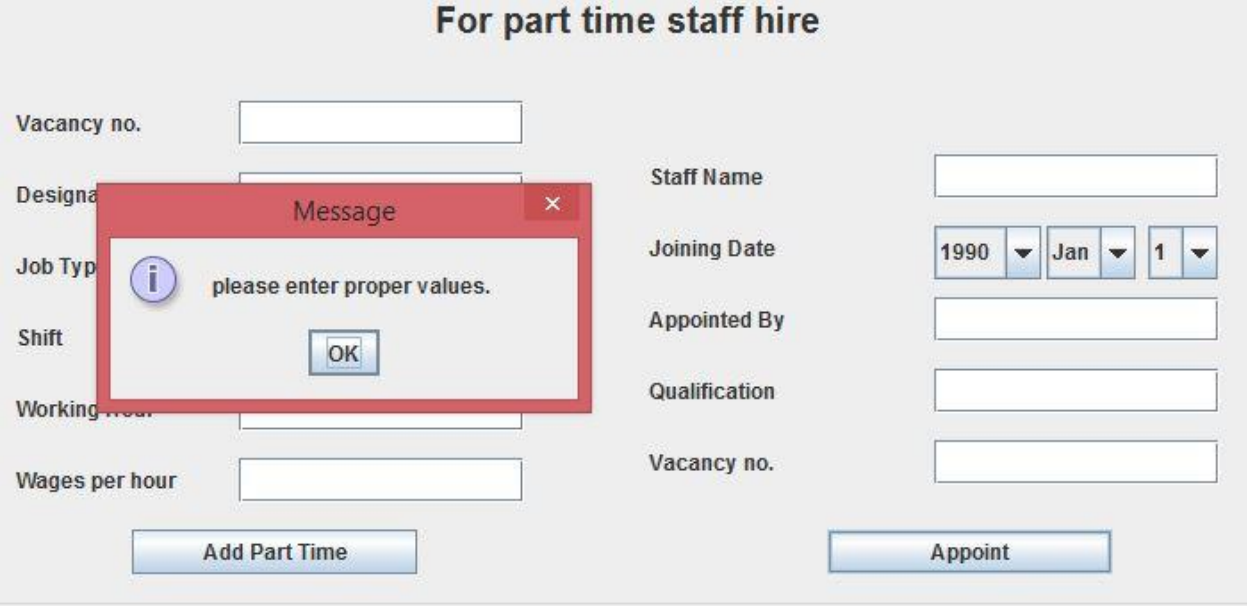


Figure 34: Empty fields appointed for part Time Staff Hire

Figure 35: Message box for empty fields appointed for Part Time Staff Hire

Test-8

Objective	To get the appropriate dialog box when input fields for PartTimeStaffHire is left empty.
Action	The input fields for PartTimeStaffHire are assigned empty and vacancy is appointed.
Expected Result	A pop up message box should appear informing to enter proper values.
Actual Result	A pop up message box appeared informing to enter proper values.

Conclusion	Test pass.

Table 14: Testing appoint button for empty fields in Part Time Staff Hire

9) INVALID VACANCY APPOINTED FOR PARTTIMESTAFFHIRE

Figure 36: Invalid fields appointed for Part Time Staff Hire

Figure 37: Message box for invalid fields appointed in Part Time Staff Hire

Test-9

Objective	To get the appropriate dialog box when invalid vacancy is appointed for PartTimeStaffHire.
Action	Invalid vacancy for PartTimeStaffHire is assigned and vacancy is appointed.
Expected Result	A pop up message box should appear informing to enter proper values.
Actual Result	A pop up message box appeared informing to enter proper values.
Conclusion	Test pass.

Table 15: Testing appointed button for invalid fields in Part Time Staff Hire

10) DIFFERENT / UN-ADDED VACANCY APPOINTED FOR PARTTIMESTAFFHIRE

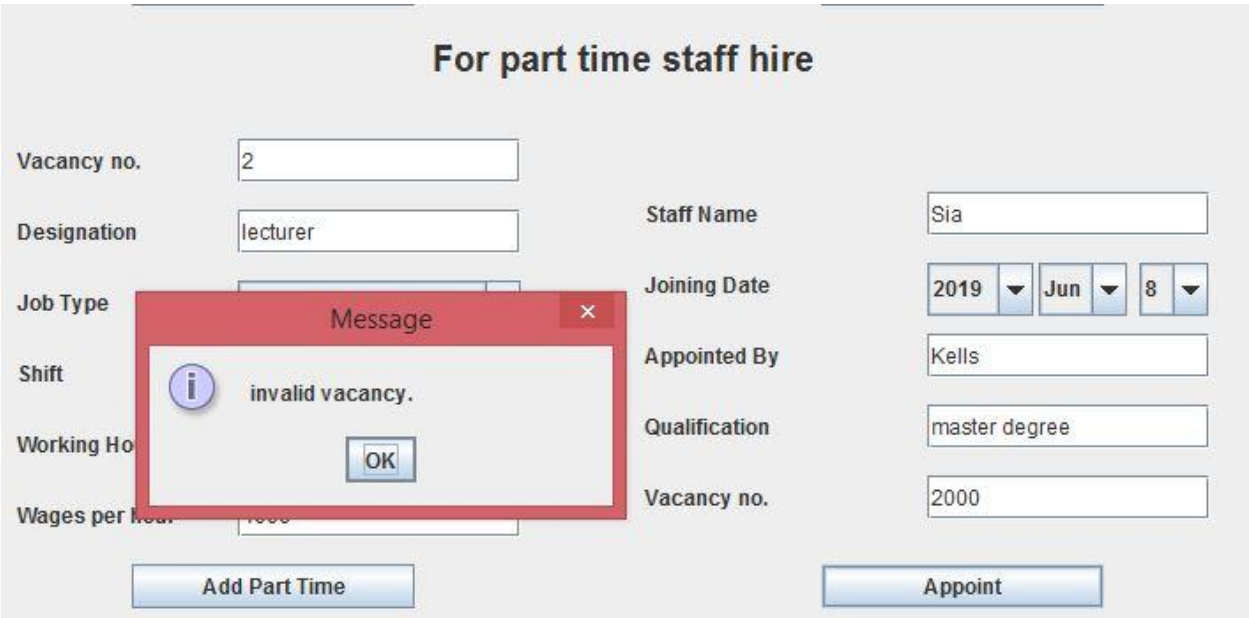


Figure 38: Different or Un-added vacancy appointed for Part Time Staff Hire

Figure 39: Message box for appointing un-added vacancy for Part Time Staff Hire

Test-10

Objective	To get the appropriate dialog box when un-added vacancy is appointed for PartTimeStaffHire
Action	Different vacancy for PartTimeStaffHire is assigned and vacancy is appointed.

Expected Result	A pop up message box should appear informing about invalid vacancy.
Actual Result	A pop up message box appeared informing about invalid vacancy.
Conclusion	Test pass.

Table 16: Testing appoint button for un-added vacancy in Part Time Staff Hire

11) VACANCY FROM FULLTIMESTAFFHIRE APPOINTED IN PARTTIMESTAFFHIRE

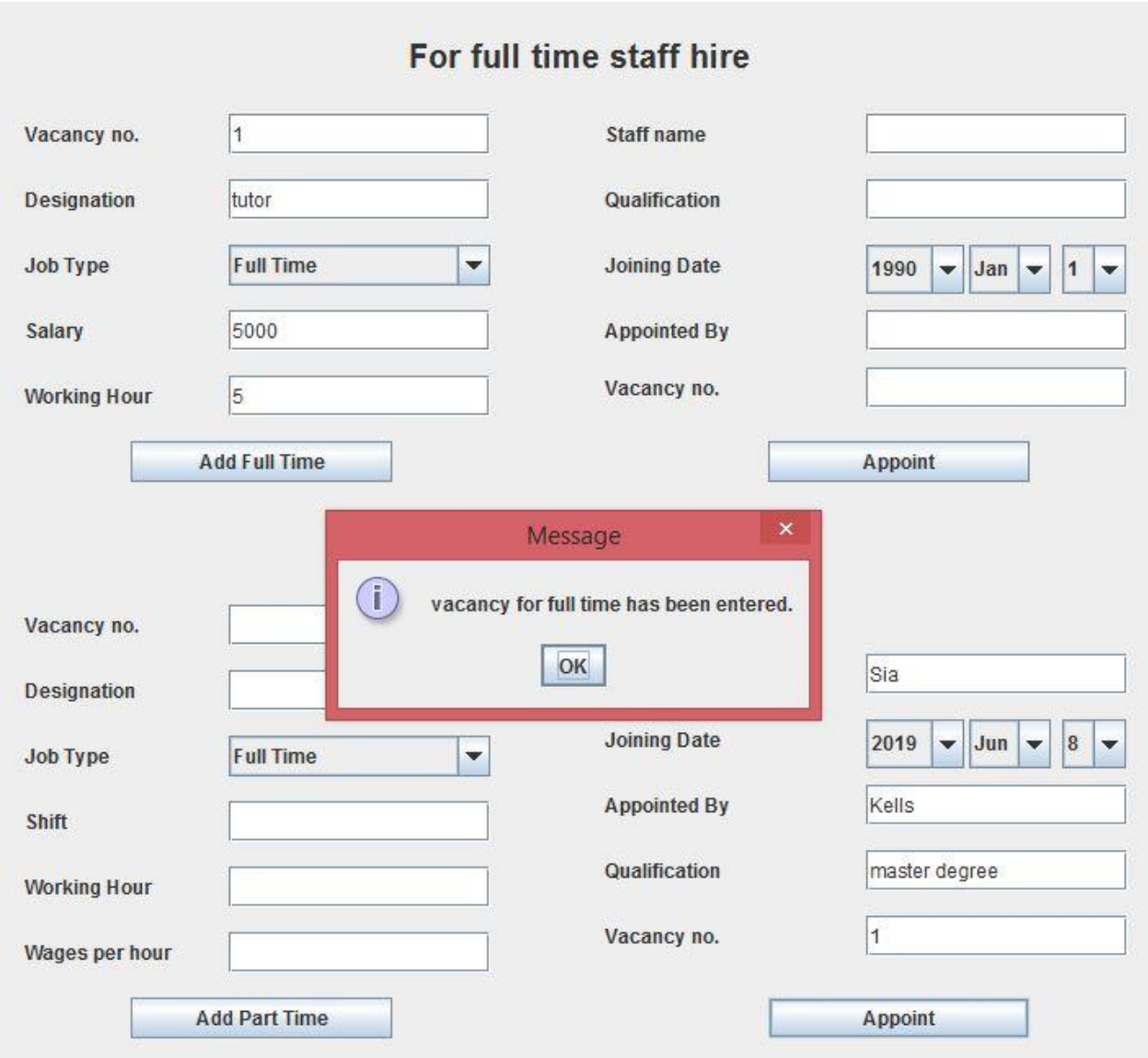


Figure 40: Inserting vacancy from Full Time Staff Hire in Part Time Staff Hire

Figure 41: Message box for inserting vacancy from Full Time Staff Hire in Part Time Staff Hire

Test-11

Objective	To get the appropriate dialog box when vacancy from FullTimeStaffHire is appointed in PartTimeStaffHire.
Action	Vacancy for FullTimeStaffHire is added and is appointed in PartTimeStaffHire.
Expected Result	A pop up message box should appear informing vacancy of different class has been entered.
Actual Result	A pop up message box appeared informing vacancy of different class has been entered.
Conclusion	Test pass.

Table 17: Testing appoint button in Part Time Staff Hire by adding vacancy from Full Time Staff Hire

12) VACANCY FROM PARTTIMESTAFFHIRE APPOINTED IN FULLTIMESTAFFHIRE

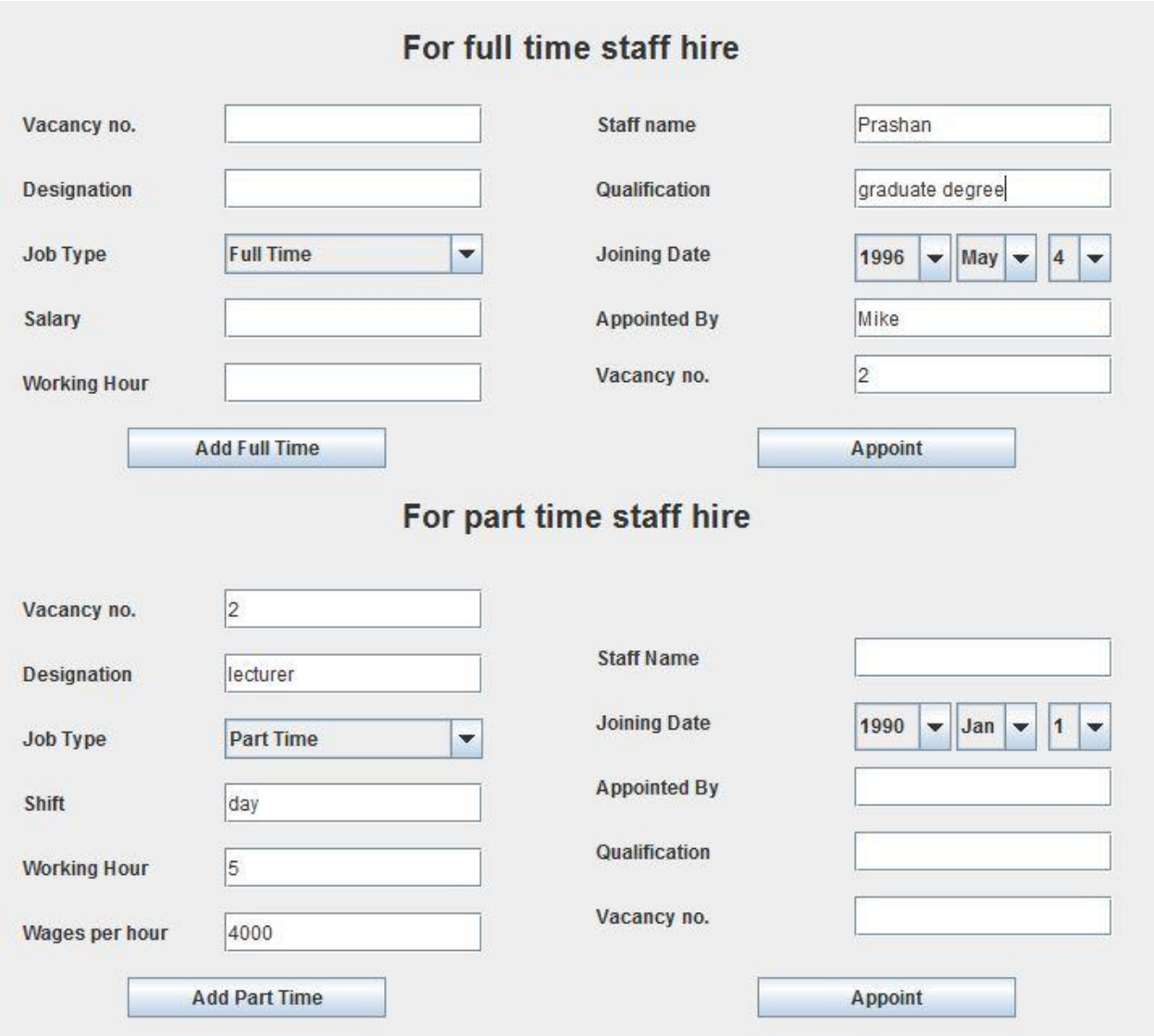


Figure 42: Inserting vacancy from Part Time Staff Hire in Full Time Staff Hire

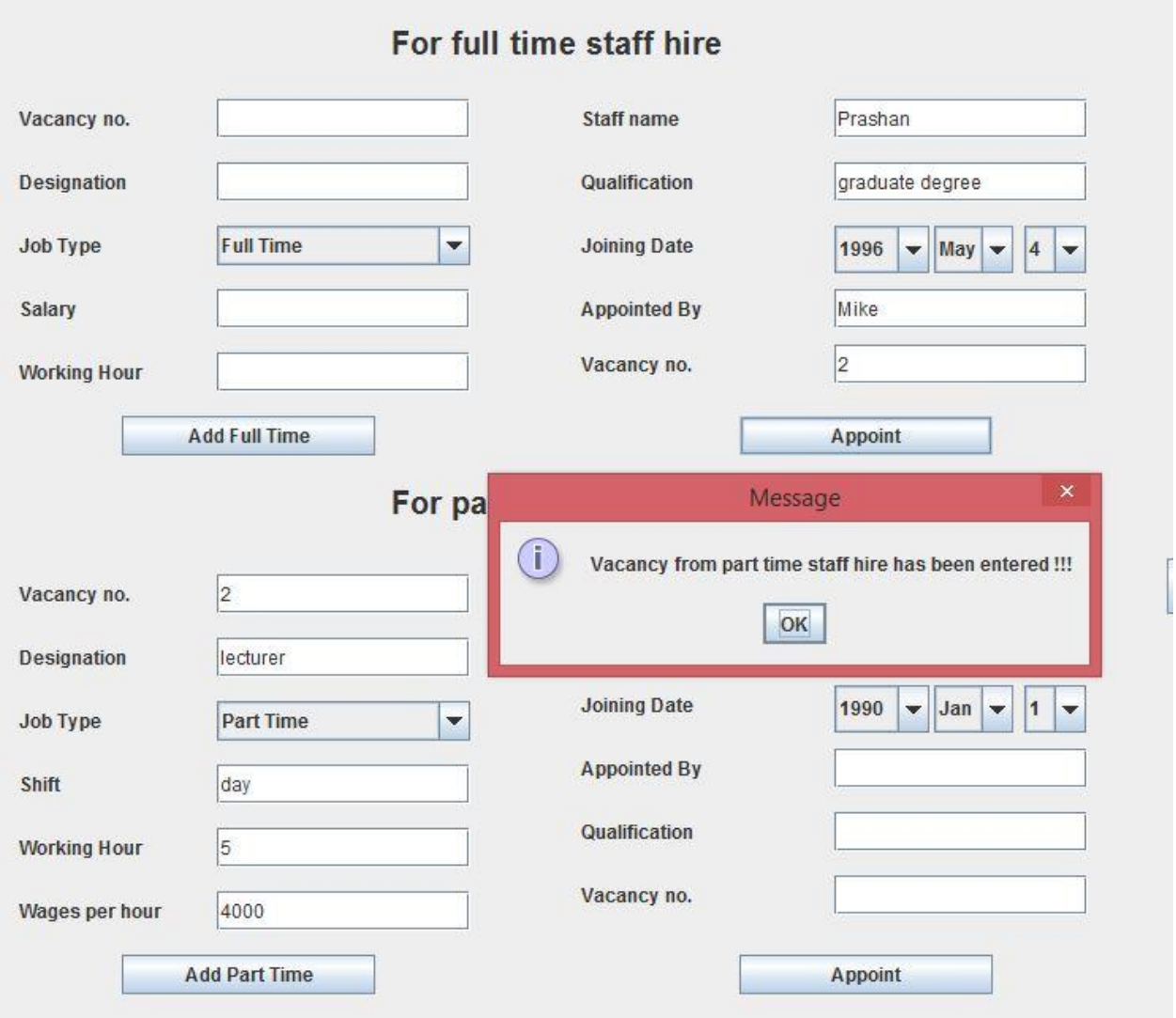


Figure 43: Inserting vacancy from Part Time Staff Hire in Full Time Staff Hire with the message box

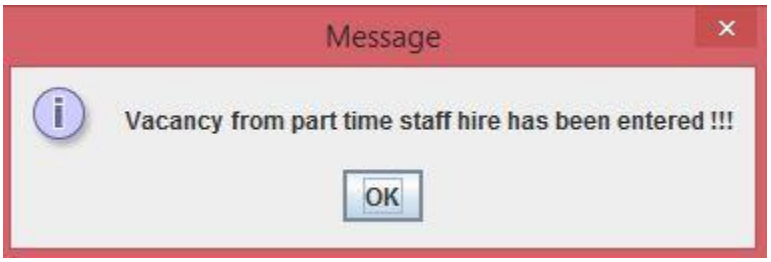


Figure 44: Message box for inserting vacancy from Part Time Staff Hire in Full Time Staff Hire

Test-12

Objective	To get the appropriate dialog box when vacancy from PartTimeStaffHire is appointed in FullTimeStaffHire.
Action	Vacancy for PartTimeStaffHire is added and is appointed in FullTimeStaffHire.
Expected Result	A pop up message box should appear informing vacancy of different class has been entered.
Actual Result	A pop up message box appeared informing vacancy of different class has been entered.
Conclusion	Test pass.

Table 18: Testing appoint button in Full Time Staff Hire by adding vacancy from Part Time Staff Hire

ERROR DETECTION - CORRECTION

There are three types of error in Java and they are:

Logical Error

Runtime Error

Syntax Error

1) LOGICAL ERROR

A logic error or logical error is a mistake in a program's source code that results in incorrect or unexpected behavior. It is a type of runtime error that may simply produce the wrong output or may cause a program to crash while running. (Productions, 2020)

The error in this program was because of logical error. The error is very simple but very hard to find. The error is I used single"=" while comparing isDuplicateVacancy a Boolean value instead of using"==". I was able to solve this error and get the output I expected.

```
if(isDuplicateVacancy=false)
{
    FullTimeStaffHire full_time_obj = new FullTimeStaffHire(vacancy,designation,job_type,salary,working_hour);
     array_list.add(full_time_obj);
     JOptionPane.showMessageDialog(frame1,"Vacancy for full time added.");
}

else
{
    JOptionPane.showMessageDialog(frame1,"Vacancy for full time is already added.");

}
```

Figure 45: Unsolved logical error

The logical error was solved by comparing the isDuplicateVacancy with proper symbol (==).

```
if(isDuplicateVacancy==false)
{
    FullTimeStaffHire full_time_obj = new FullTimeStaffHire(vacancy,designation,job_type,salary,working_hour);
     array_list.add(full_time_obj);
     JOptionPane.showMessageDialog(frame1,"Vacancy for full time added.");
}

else
{
    JOptionPane.showMessageDialog(frame1,"Vacancy for full time is already added.");

}
```

Figure 46: Solved logical error

2) RUNTIME ERROR

A runtime error is a program error that occurs while the program is running. The term is often used in contrast to other types of program errors, such as syntax errors and compiles time errors. There are many different types of runtime errors. One example is a logic error, which produces the wrong output. (Productions, 2020)

The runtime error occurs while trying to execute the program. The error is because I filled the text field of salary in FullTimeStaffHire as string data type instead of int data type which provides a message box as invalid since try catch block is used to handle the exception.

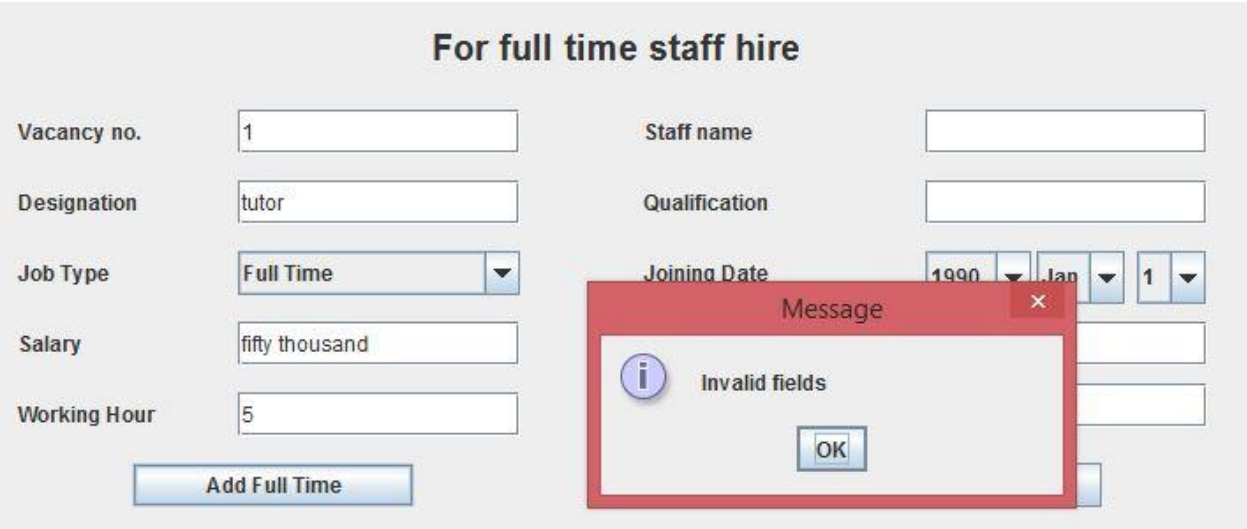


Figure 47: Unsolved runtime error

The runtime error was solved by providing proper int data type value instead of string data type value in the salary text field.

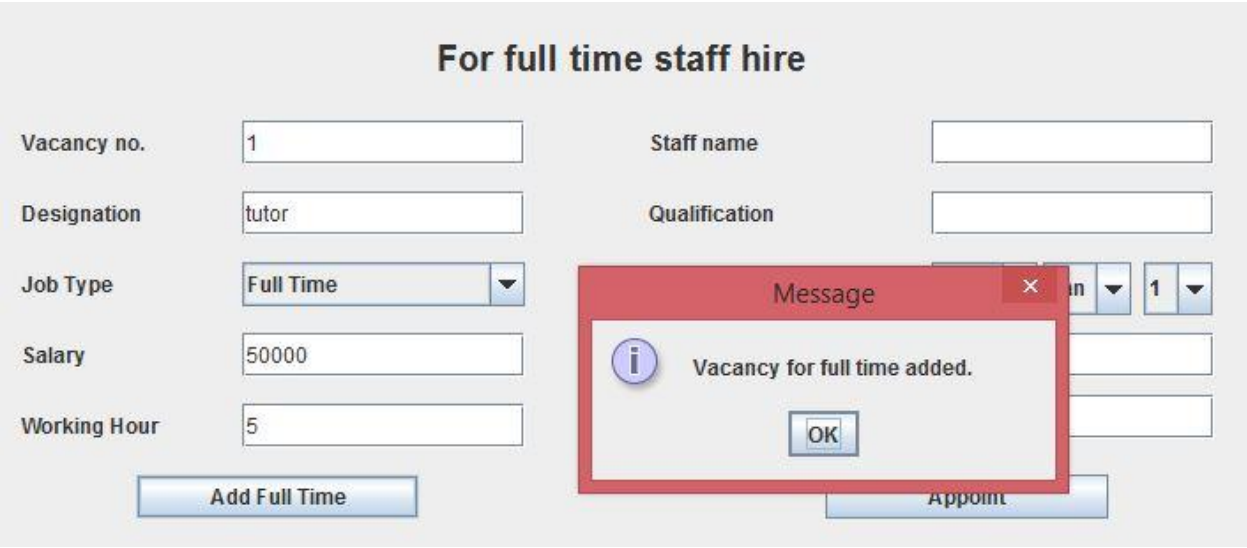


Figure 48: Solved runtime error

3) SYNTAX ERROR

A syntax error is an error in the source code of a program. Since computer programs must follow strict syntax to compile correctly, any aspects of the code that do not conform to the syntax of the programming language will produce a syntax error. (Productions, 2020)

During compiling of my program the error message shows a semi colon mark to be missing. I checked my program and I am able to find the error. The error is very minor, but also affects the program strongly.

```
button1=new JButton("Add Full Time");
button1.setBounds(80,270,160,25);
button1.addActionListener(this);
panel1.add(button1)

button2=new JButton("Appoint");
button2.setBounds(470,270,160,25);
button2.addActionListener(this);
panel1.add(button2)
```

Figure 49: Unsolved syntax error

The program had a syntax error was solved by filling the missing semi colon mark (;) at the end of the code.

```
button1=new JButton("Add Full Time");
button1.setBounds(80,270,160,25);
button1.addActionListener(this);
panel1.add(button1);

button2=new JButton("Appoint");
button2.setBounds(470,270,160,25);
button2.addActionListener(this);
panel1.add(button2);
```

Figure 50: Solved syntax error

CONCLUSION

This coursework was all about creating a graphical user interface (GUI) for a system that stores details of vacancy and hired staff details in the list. It was a quite difficult assignment

for me. It made me research about the project in various aspects. It was able to teach me about array list, event handling, object casting, different GUI components like frame, text fields, buttons labels etc. and so on.

I didn't find any problem during creating the GUI form. But I did find the action implementation for buttons in the form to be difficult. I faced with different kinds of errors and exceptions in this assignment but with the help of the internet, online class reference videos, tutors and friends, I was able to complete this assignment and also able to learn about various kinds of exception handlings. I finally completed the coursework but wasn't able to submit in the original submission date. I took a lot of time for this assignment since I had a lot of confusions with this work and now submitting the work in the extended submission date.

REFERENCES

Anon. (2019) *w3school* [Online]. Available from: https://www.w3schools.com/java/java_intro.asp.
Productions, S. (2020) *TechTerm* [Online]. Available from: https://techterms.com/definition/runtime_error.
Productions, S. (2020) *TechTerm* [Online]. Available from: https://techterms.com/definition/syntax_error.
Productions, S. (2020) *tectTerm* [Online]. Available from: https://techterms.com/definition/logical_error.

APPENDIX

1) APPENDIX 1

CREATING ING CLASS

```
import java.awt.Color;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;
import java.util.ArrayList;

class INGNepal implements ActionListener
  {
```

```
        private JFrame frame1;

      private JPanel panel1;

      private JLabel
heading1,heading2,write1,write2,write3,write4,write5,write6,write7,write8,write9,

write10,type1,type2,type3,type4,type5,type6,type7,type8,type9,type10,type11,type12;

      private JTextField box1,box2,box4,box5,box6,box7,box9,box10,

empty1,empty2,empty4,empty5,empty6,empty7,empty9,empty10,empty11,empty12;

        private JComboBox
box3,empty3,cmbYear,cmbMonth,cmbDay,cmbYear2,cmbMonth2,cmbDay2;

        private JButton button1,button2,button3,button4,button5,button6,button7;




        int vacancy;
        int vacancy_no;
        String designation;
        String job_type,cmb,cmb_part;
        int working_hour;
        int salary;
        String working_shift;
        int wages_per_hour;
        String staffName;
        String appointedBy;
        String qualification;
        String joiningDate,year,month,day,year2,month2,day2;

         ArrayList<StaffHire> array_list =new ArrayList<>();
        /*
         *
         *
         *
         *
         */



        public void m1(){
        frame1 = new JFrame("form");
        panel1=new JPanel();
        frame1.setBounds(100,20,1200,690);// Bounds helps to set the size and location
of the tab at once
        panel1.setLayout(null);
```

```
/*
 *
 *
 *
 * "Full Time Staff Hire"
 *
 *
 *
 */
heading1=new JLabel();
heading1.setText("For full time staff hire");
heading1.setBounds(250,10,305,50);
Font topic=new Font("Arial",Font.BOLD,20);
heading1.setFont(topic);
panel1.add(heading1);

/*
 *
 *
 * JLabel of Full Time Staff Hire
 *
 *
 */
write1=new JLabel();
write1.setText("Vacancy no.");
write1.setBounds(15,70,100,25);
panel1.add(write1);


write2=new JLabel();
write2.setText("Designation");
write2.setBounds(15,110,100,25);
panel1.add(write2);

write3=new JLabel();
write3.setText("Job Type");
write3.setBounds(15,150,100,25);
panel1.add(write3);

write4=new JLabel();
write4.setText("Salary");
write4.setBounds(15,190,100,25);
panel1.add(write4);

write5=new JLabel();
write5.setText("working_hour");
write5.setBounds(15,230,100,25);
panel1.add(write5);

write6=new JLabel();
```

```
write6.setText("Staff name");
write6.setBounds(370,70,100,25);
panel1.add(write6);

write7=new JLabel();
write7.setText("Qualification");
write7.setBounds(370,110,100,25);
panel1.add(write7);

write8=new JLabel();
write8.setText("Joining Date");
write8.setBounds(370,150,100,25);
panel1.add(write8);

write9=new JLabel();
write9.setText("Appointed By");
write9.setBounds(370,190,100,25);
panel1.add(write9);

write10=new JLabel();
write10.setText("Vacancy no.");
write10.setBounds(370,225,100,25);
panel1.add(write10);



/*
 *
 *
 *
 * JTextField of Full Time
 *
 *
 *
 */

box1=new JTextField();
box1.setBounds(140,70,160,25);
panel1.add(box1);

box2=new JTextField();
box2.setBounds(140,110,160,25);
panel1.add(box2);

/*
String cmb[]={"Full Time", "Part Time"};
box3= new JComboBox(cmb);
*/

box3 = new JComboBox<>(new String[] {"Full Time", "Part Time" });
box3.setBounds(140,150,160,25);
```

```
panel1.add(box3);

box4=new JTextField();
box4.setBounds(140,190,160,25);
panel1.add(box4);

box5=new JTextField();
box5.setBounds(140,230,160,25);
panel1.add(box5);

box6=new JTextField ();
box6.setBounds(530,70,160,25);
panel1.add(box6);

box7=new JTextField();
box7.setBounds(530,110,160,25);
panel1.add(box7);



cmbYear = new JComboBox<>(new String[] {"1990", "1991", "1992", "1993",
"1994", "1995", "1996", "1997", "1998", "1999", "2000", "2001", "2002", "2003", "2004",
"2005", "2006", "2007", "2008", "2009", "2010", "2011", "2012" ,
"2013","2014","2015","2016","2017","2018","2019","2020","2021"});
 cmbYear.setBounds(530,150,60,30);
 panel1.add(cmbYear);



 cmbMonth= new JComboBox<>(new String[]
{"Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sept","Oct","Nov","Dec"});
 cmbMonth.setBounds(593,150,50,30);
 panel1.add(cmbMonth);


cmbDay= new JComboBox<>(new String[]
{"1","2","3","4","5","6","7","8","9","10","11","12","13","14","15","16","17","18","19","20","21
","22","23","24","25","26","27","28","29","30","31"});
cmbDay.setBounds(650,150,39,30);
panel1.add(cmbDay);


box9=new JTextField();
box9.setBounds(530,190,160,25);
panel1.add(box9);

box10=new JTextField();
box10.setBounds(530,225,160,25);
panel1.add(box10);
 /*
 *
```

```
 *
 *
 *
 * Button of Full Time
 *
 *
 *
 */



button1=new JButton("Add Full Time");
button1.setBounds(80,270,160,25);
button1.addActionListener(this);
panel1.add(button1);

button2=new JButton("Appoint");
button2.setBounds(470,270,160,25);
button2.addActionListener(this);
panel1.add(button2);

/*
 *
 *
 *
 * Part Time Staff Hire
 *
 *on
 */

heading2=new JLabel();
heading2.setText("For part time staff hire");
heading2.setBounds(250,300,300,50);
Font topic2=new Font("Arial",Font.BOLD,20);
heading2.setFont(topic2);
panel1.add(heading2);
/*
 *
 *
 *
 *
 * Jlabel of part time
 *
 *
 */
type1=new JLabel();
type1.setText("Vacancy no.");
type1.setBounds(15,370,100,25);
panel1.add(type1);
```

```
type2=new JLabel();
type2.setText("Designation");
type2.setBounds(15,410,100,25);
panel1.add(type2);

type3=new JLabel();
type3.setText("Job Type");
type3.setBounds(15,450,100,25);
panel1.add(type3);

type4=new JLabel();
type4.setText("Shift");
type4.setBounds(15,490,100,25);
panel1.add(type4);

type5=new JLabel();
type5.setText("working_hour");
type5.setBounds(15,530,100,25);
panel1.add(type5);

type6=new JLabel();
type6.setText("Wages per hour");
type6.setBounds(15,570,100,25);
panel1.add(type6);

type7=new JLabel();
type7.setText("Staff Name");
type7.setBounds(370,400,100,25);
panel1.add(type7);

type8=new JLabel();
type8.setText("Joining Date");
type8.setBounds(370,440,100,25);
panel1.add(type8);

type9=new JLabel();
type9.setText("Appointed By");
type9.setBounds(370,480,100,25);
panel1.add(type9);



type10=new JLabel();
type10.setText("Qualification");
type10.setBounds(370,520,100,25);
panel1.add(type10);


type11=new JLabel();
type11.setText("Vacancy no.");
```

```
type11.setBounds(370,560,100,25);
panel1.add(type11);

type12=new JLabel();
type12.setText("Vacancy no.");
type12.setBounds(820,260,100,25);
panel1.add(type12);
/*
 *
 *
 *
 * JTextField of Part Time
 *
 *
 *
 */

empty1=new JTextField();
empty1.setBounds(140,370,160,25);
panel1.add(empty1);

empty2=new JTextField();
empty2.setBounds(140,410,160,25);
panel1.add(empty2);

/*empty3=new JTextField();
empty3.setBounds(140,450,160,25);
panel1.add(empty3);
*/


 empty3 = new JComboBox<>(new String[] {"Full Time", "Part Time" });
empty3.setBounds(140,450,160,25);
panel1.add(empty3);



empty4=new JTextField();
empty4.setBounds(140,490,160,25);
panel1.add(empty4);

empty5=new JTextField();
empty5.setBounds(140,530,160,25);
panel1.add(empty5);

empty6=new JTextField();
empty6.setBounds(140,570,160,25);
panel1.add(empty6);

empty7=new JTextField();
empty7.setBounds(530,400,160,25);
```

```
            panel1.add(empty7);
            /*
            empty8=new JTextField();
            empty8.setBounds(530,440,160,25);
            panel1.add(empty8);
            */
            cmbYear2 = new JComboBox<>(new String[] {"1990", "1991", "1992", "1993",
"1994", "1995", "1996", "1997", "1998", "1999", "2000", "2001", "2002", "2003", "2004",
"2005", "2006", "2007", "2008", "2009", "2010", "2011", "2012" ,
"2013","2014","2015","2016","2017","2018","2019","2020","2021"});
            cmbYear2.setBounds(530,440,60,30);
             panel1.add(cmbYear2);

            cmbMonth2= new JComboBox<>(new String[]
{"Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sept","Oct","Nov","Dec"});
             cmbMonth2.setBounds(593,440,50,30);
             panel1.add(cmbMonth2);

            cmbDay2= new JComboBox<>(new String[]
{"1","2","3","4","5","6","7","8","9","10","11","12","13","14","15","16","17","18","19","20","21
","22","23","24","25","26","27","28","29","30","31"});
            cmbDay2.setBounds(650,440,39,30);
            panel1.add(cmbDay2);


            empty9=new JTextField();
            empty9.setBounds(530,480,160,25);
            panel1.add(empty9);

            empty10=new JTextField();
            empty10.setBounds(530,520,160,25);
            panel1.add(empty10);

            empty11=new JTextField();
            empty11.setBounds(530,560,160,25);
            panel1.add(empty11);

            empty12=new JTextField();
            empty12.setBounds(950,260,160,25);
            panel1.add(empty12);

             /*
             *
             *
             *
             *
             * Button of  Part Time
             *
             *
             *
             */
```

```
button3=new JButton("Add Part Time");
button3.setBounds(80,610,160,25);
button3.addActionListener(this);
panel1.add(button3);

button4=new JButton("Appoint");
button4.setBounds(470,610,160,25);
button4.addActionListener(this);
panel1.add(button4);

button5=new JButton("Terminate part time");
button5.setBounds(890,300,160,25);
button5.addActionListener(this);
panel1.add(button5);


/*
 *
 *
 * Button at the last
 *
 *
 *
 */
button6=new JButton("Clear");
button6.setBounds(740,360,200,35);
button6.addActionListener(this);
panel1.add(button6);

button7=new JButton("Display");
button7.setBounds(950,360,200,35);
button7.addActionListener(this);
panel1.add(button7);



/*
 *
 *
 *
 *
 *
 */
//frame.setTitle("Form1");// gives title to the tab
//frame.setSize(600,460); // gives size of the tab
//frame.setLocation(450,100); //gives location of the tab
frame1.add(panel1);

frame1.setResizable(false);// cannot resize the tab when  false
frame1.setVisible(true); // making the tab visible
```

```
            frame1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

        public void actionPerformed(ActionEvent e) {
            if(e.getSource()==button1){

            try{
            vacancy=Integer.parseInt(box1.getText());
            designation=box2.getText();
            job_type=(box3.getSelectedItem()).toString();
            salary=Integer.parseInt(box4.getText());
            working_hour=Integer.parseInt(box5.getText());


            boolean isDuplicateVacancy=false;
            for(StaffHire var:array_list)
        {
            if(var.getVacancy_number()==vacancy){
                isDuplicateVacancy=true;
                break;
            }
        }
            if(isDuplicateVacancy==false)
            {
                FullTimeStaffHire full_time_obj = new
FullTimeStaffHire(vacancy,designation,job_type,salary,working_hour);
                 array_list.add(full_time_obj);
                 JOptionPane.showMessageDialog(frame1,"Vacancy for full time added.");
            }

            else
            {
                JOptionPane.showMessageDialog(frame1,"Vacancy for full time is already added.");

            }


        }catch(Exception ee)
        {
            JOptionPane.showMessageDialog(frame1,"Invalid fields");

        }
    }


            if(e.getSource()==button3){

            try{
            vacancy=Integer.parseInt(empty1.getText());
            designation=empty2.getText();
            cmb_part=(empty3.getSelectedItem()).toString();
```

```
        job_type=cmb_part;
        wages_per_hour=Integer.parseInt(empty6.getText());
        working_hour=Integer.parseInt(empty5.getText());
        working_shift=empty4.getText();

        boolean isDuplicateVacancy=false;
        for(StaffHire var:array_list)
     {
        if(var.getVacancy_number()==vacancy){
           isDuplicateVacancy=true;
           break;
        }
     }
        if(isDuplicateVacancy==false)
        {
            PartTimeStaffHire part_time_obj = new
PartTimeStaffHire(vacancy,designation,job_type,working_hour,wages_per_hour,working_s
hift);
            array_list.add(part_time_obj);
            JOptionPane.showMessageDialog(frame1,"Vacancy for part time added.");
        }
        else
        {
           JOptionPane.showMessageDialog(frame1,"Vacancy for part time is already
added.");

        }

     }catch(Exception ee)
     {
        JOptionPane.showMessageDialog(frame1,"Invalid fields");

     }
  }
   if(e.getSource()==button2){

  try
  {
       vacancy=Integer.parseInt(box10.getText());
        staffName=box6.getText();
        qualification=box7.getText();
        year=(cmbYear.getSelectedItem()).toString();
        month=(cmbMonth.getSelectedItem()).toString();
        day=(cmbDay.getSelectedItem()).toString();
        joiningDate=year+month+day;
        appointedBy=box9.getText();

        boolean vacancyFound=false;
        for(StaffHire ob:array_list){
        if(ob.getVacancy_number()==vacancy){
```

```
            vacancyFound=true;
        if(ob instanceof FullTimeStaffHire){
            FullTimeStaffHire h=(FullTimeStaffHire)ob;

            if(h.getJoined()==true){

                JOptionPane.showMessageDialog(frame1,"Staff has been already hired.");
   }
   else
   {
   h.FullTimeStaff_Hire(staffName,joiningDate,qualification,appointedBy);

   JOptionPane.showMessageDialog(frame1,"Staff has been successfully hired.");
   break;
   }
   }
   else
   {
   JOptionPane.showMessageDialog(frame1,"Vacancy from part time staff hire has been
entered !!!");
   break;
   }
   }
   }
   if(vacancyFound==false)
   {
   JOptionPane.showMessageDialog(frame1,"invalid vacancy.");
   }
   }


   catch(Exception ee)
   {
      JOptionPane.showMessageDialog(frame1,"please enter proper values.");
   }

   }
    if(e.getSource()==button4){

   try
   {
        vacancy=Integer.parseInt(empty11.getText());
         staffName=empty7.getText();
         qualification=empty10.getText();
         year2=(cmbYear2.getSelectedItem()).toString();
         month2=(cmbMonth2.getSelectedItem()).toString();
         day2=(cmbDay2.getSelectedItem()).toString();
         joiningDate=year2+month2+day2;

         appointedBy=empty9.getText();
```

```
        boolean vacancyFound=false;
        boolean Staff_nameFound=false;

        for(StaffHire ob_p:array_list){
        if(ob_p.getVacancy_number()==vacancy)
        {

           vacancyFound=true;
        if(ob_p instanceof PartTimeStaffHire){
           PartTimeStaffHire g=(PartTimeStaffHire)ob_p;

           if(g.getJoined()==true){

                JOptionPane.showMessageDialog(frame1,"Staff has been already hired.");
  }
  else
  {
  g.Hire_PartTimeStaff(staffName,joiningDate,qualification,appointedBy);

  JOptionPane.showMessageDialog(frame1,"Staff has been successfully hired.");
  break;
  }
  }
  else
  {
  JOptionPane.showMessageDialog(frame1,"vacancy for full time has been entered.");
  break;
  }
  }
  }
  if(vacancyFound==false)
  {
  JOptionPane.showMessageDialog(frame1,"invalid vacancy.");
  }
  }


  catch(Exception ee)
  {
     JOptionPane.showMessageDialog(frame1,"please enter proper values.");
  }

  }
   if(e.getSource()==button5){
        try
        {

       vacancy=Integer.parseInt(empty12.getText());

       boolean vacancyFound=false;
```

```
        for(StaffHire ob_t:array_list){

        if(ob_t.getVacancy_number()==vacancy){

           vacancyFound=true;
        if(ob_t instanceof PartTimeStaffHire){

          PartTimeStaffHire  l=(PartTimeStaffHire)ob_t;

           if(l.getJoined()==false){


                JOptionPane.showMessageDialog(frame1,"Staff has been already terminated.");
           break;
      }
   else {
       l.Terminate_Staff();
   JOptionPane.showMessageDialog(frame1,"Staff has been terminated.");
   break;
   }
   }
   else
   {
   JOptionPane.showMessageDialog(frame1,"Vacancy from full time staff hire has been
entered !!!");
   break;
   }
   }
   }

        if(vacancyFound==false)
   {
   JOptionPane.showMessageDialog(frame1,"invalid vacancy.");
   }
   }       catch(Exception ee){
                JOptionPane.showMessageDialog(frame1,"Please enter valid vacancy.");

        }
   }
   if(e.getSource()==button6){
                box1.setText("");
                box2.setText("");
                box3.setSelectedIndex(0);
                box4.setText("");
                box5.setText("");
                box6.setText("");
                box7.setText("");
                cmbYear.setSelectedIndex(0);
                cmbMonth.setSelectedIndex(0);
                cmbDay.setSelectedIndex(0);
                box9.setText("");
```

```
                box10.setText("");
                empty1.setText("");
                empty2.setText("");
                empty3.setSelectedIndex(0);
                empty4.setText("");
                empty5.setText("");
                empty6.setText("");
                empty7.setText("");
                cmbYear2.setSelectedIndex(0);
                cmbMonth2.setSelectedIndex(0);
                cmbDay2.setSelectedIndex(0);
                empty9.setText("");
                empty10.setText("");
                empty11.setText("");
                empty12.setText("");
            }
if(e.getSource()==button7)
{
   for(StaffHire var:array_list){ //iterating array list
   if(var instanceof FullTimeStaffHire){
      FullTimeStaffHire x=(FullTimeStaffHire)var;
      x.Display();
   }
   if(var instanceof PartTimeStaffHire){
      PartTimeStaffHire z=(PartTimeStaffHire)var;
      z.Display();
   }
}
}
}
public static void main(String[] args)
{
   INGNepal b= new INGNepal();
   b.m1();
 }
}
```

2) APPENDIX 2

PSEDOCODE:

CREATE class FullTimeStaffHire

DECLARE class variables

String designation, String job_type, int vacancy_number

```
FUNCTION getDesignation()
DO
return designation;
END DO
END FUNCTION
FUNCTION setDesignation(String designation)
DO
this.designation=designation
END DO
END FUNCTION
FUNCTION getJob_type()
DO
return Job_type
END DO
END FUNCTION
FUNCTION setJob_type(String job_type)
DO
this.job_type=job_type
END DO

END FUNCTION
FUNCTION getVancancy_number()
DO
return vacancy_number
END DO
END FUNCTION
```

FUNCTION setVacancy_number(int vancancy_number)

DO

this.vancancy_number=vacancy_number

END DO

END FUNCTION

FUNCTION

DO

PRINT designation, job_type, vacancy_number

END DO

END FUNCTION

CLASS: FullTimeStaffHire

CREATE class FullTimeStaffHire

DECLARE class variables int salary, int Working_Hour, String Staff_Name, String Joining_Date, String Qualification, String Appointed_By, boolean Joined

FUNCTION getsalary(int)

DO

return salary;

```
END DO

END FUNCTION

FUNCTION getWorking_Hour(int)

DO

return Working_Hour;

END DO

END FUNCTION

FUNCTION getStaff_Name (String)

DO

return Staff_Name

END DO

END FUNCTION

FUNCTION getJoining_Date (String)

DO

return Joining_Date;

END DO

END FUNCTION

FUNCTION getQualification (String)

DO

return Qualification


END DO

END FUNCTION

FUNCTION getAppointed_By (String)
```

```
DO
return Appointed_By
END DO
END FUNCTION
FUNCTION getJoined (boolean)
DO
return Joined;
END DO
END FUNCTION
FUNCTION setsalary (int salary)
DO
IF Joined equals to true
PRINT “salary is not changeable”;
ELSE this.salary=salary;
END IF
END DO
END FUNCTION
FUNCTION
setWorking_Hour (int Working_Hour)
DO
this.Working_Hour=Working_Hour;
END DO
END FUNCTION
FUNCTION
FullTimeStaff_Hire (String Staff_Name, String Joining_Date, String Qualification, String
```

```
Appointed_By)

DO

IF Joined equals to true

PRINT "Staff has already joined organization";

ELSE

this.Staff_Name=Staff_Name; this.Joining_Date=Joining_Date;
this.Qualification=Qualification;

this.Appointed_By=Appointed_By;

this.Joined=true;

END IF

END DO

END FUNCTION

FUNCTION Display ()

DO

CALLING Display () method of super class IF Joined=true

PRINT Staff_Name, salary, Working_Hour, Joining_Date, Qualification, Appointed_By;

END IF

END DO

END FUNCTION
```

CLASS: PartTimeStaffHire

```
CREATE class PartlTimeStaffHire
```

DECLARE class variables int Working_Hour, int WagesPerHour, String Staff_Name, String Joining_Date, String Qualification, String Appointed_By,String Shifts, boolean Joined, boolean Terminated

FUNCTION getWagesPerHour(int)

DO

return WagesPerHour;

END DO

END FUNCTION

FUNCTION getStaff_Name(int)

DO

return Staff_Name

END DO

END FUNCTION

FUNCTION getStaff_Name (String)

DO

return Staff_Name

END DO

END FUNCTION

FUNCTION getJoining_Date (String)

DO

return Joining_Date;

END DO

END FUNCTION

FUNCTION getQualification (String)

DO

```
return Qualification


END DO
END FUNCTION
FUNCTION getAppointed_By (String)
DO
return Appointed_By
END DO
END FUNCTION
FUNCTION getShifts ()
DO
return Shifts
END DO
END FUNCTION
FUNCTION getJoined()
DO
IF Joined equals to true
PRINT “shift is not changeable”;
ELSE this.Shifts=Shifts;
END IF
END DO
END FUNCTION
FUNCTION Hire_PartTimeStaff(String Staff_Name, String Joining_Date, String Qualification,
String Appointed_By)
DO
```

```
IF Joined=true


PRINT Staff_Name and Joining_Date;

ELSE this.Staff_Name=Staff_Name; this.Joining_Date= Joining_Date;


this. Qualification =Qualification; this.Appointed_By= Appointed_By; this. Joined =Joined;
this.Terminated= Terminated;

END IF

END DO

END FUNCTION

FUNCTION Terminate_Staff()

DO

IF Terminated=true

PRINT "The staff is already terminated";

ELSE this.Staff_Name=""; this.Joining_Date=""; this.Qualification=""; this.Appointed_By="";
this.Joined=false; this.Terminated=true;

END IF

END DO

END FUNCTION




FUNCTION Display_Details()

DO

CALLING Display() method of super class

IF joined=true

PRINT Staff_Name, WagesPerHour, Working_Hour, joining_Date, Qualification,
Appointed_By, income per day;
```

```
ENDIF

ENDDO

ENDFUNCTION
```

METHOD DESCRIPTION

StaffHire

getDesignation()

This method returns the value of designation in String data type. It is also known as getter method.

setDesignation()

This method calls makes it able to set the value for designation and it is also know as setter method.

getJob_type()

This method returns the value of job_type in String data type.

setJob_type()

This is a method used to assign or update the data on the variable job_type.

getVacancy_number()

This method returns the value of vacancy_number in int data type. The data stores in job_type is called from this method.

setVacancy_number()

Assigning values to the variable vacancy_no is possible by this method.

display()

This method is used to display certain data or information selected by the programmer.

FullTimeStaffHire

getsalary()

This method returns the value of salary in int data type.

get Working_Hour()

This method returns the value of Working_Hour in int data type..

getStaff_Name()

This method return the value of Staff_Hire i.e the name of the staff hired to it's corresponding variable.

getJoining_Date()

This method return the value of Joining_Date i.e the date in which the staff was hired to it's corresponding variable.

getQualification()

This method return the value of qualification i.e the level of qualification that the staff has to it's corresponding variable.

getAppointedBy()

This method return the value of appointedByi.e the person that appointed the staff to it's corresponding variable.

getJoined()

This method return the value of joined i.e the information on whether the staff has joined or not to the organization to it's corresponding variable.

TESTING

1) FullTimeStaffHire:

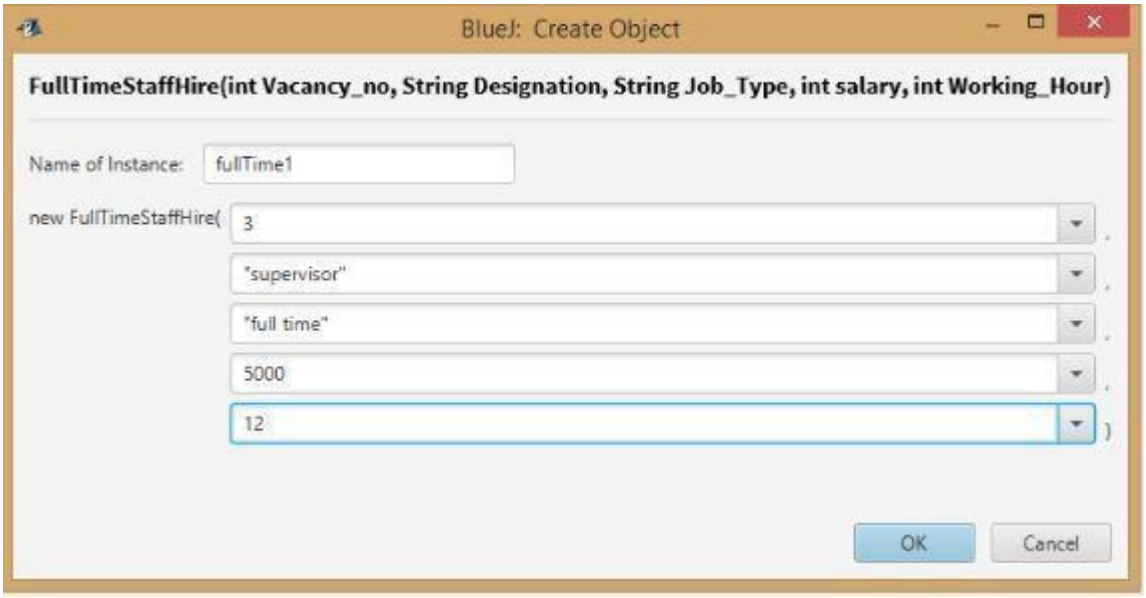


Figure 2 : creating object for FullTimeStaffHire

Figure 3: inspection for FullTimeStaffHire

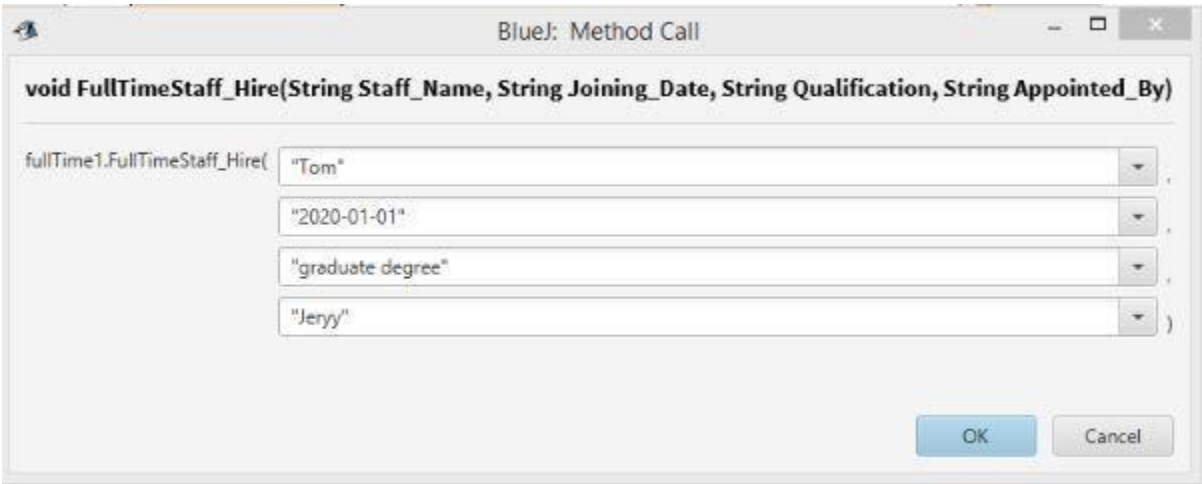


Figure 4 :method call for FullTimeStaffHire

Figure 5 : re-inspection for FullTimeStaffHire

Test 1

Objective	To inspect FullTimeStaffHire class, appoint value to hire full time staff and re-inspect the FullTimeStaffHire class

Action	The value assigned in FullTimestaffHire class are: Vacancy_no=3 Designation=”supervisor” Job-type=”full time” Salary=”5000” Working_Hour=12 Inspection of FullTimeStaffHire Void Hire is called staffName=”Tom” joiningDate=”2020-01-01” qualification=”graduate degree” appointedby=”Jerry” Re-inspect the FullTimeStaffHire
Expected Result	Entered data to be shown in inspection box
Actual Result	Entered data were shown in inspection box
Conclusion	Test pass

PartTimeStaffHire

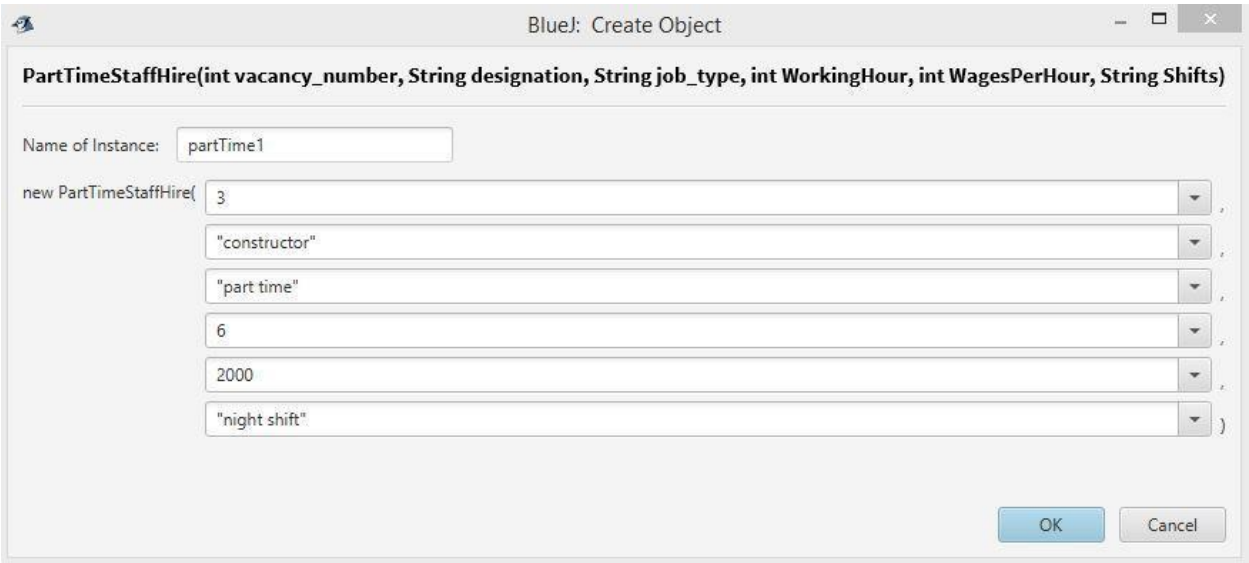


Figure 6: creating object for PartTimeStaffHire

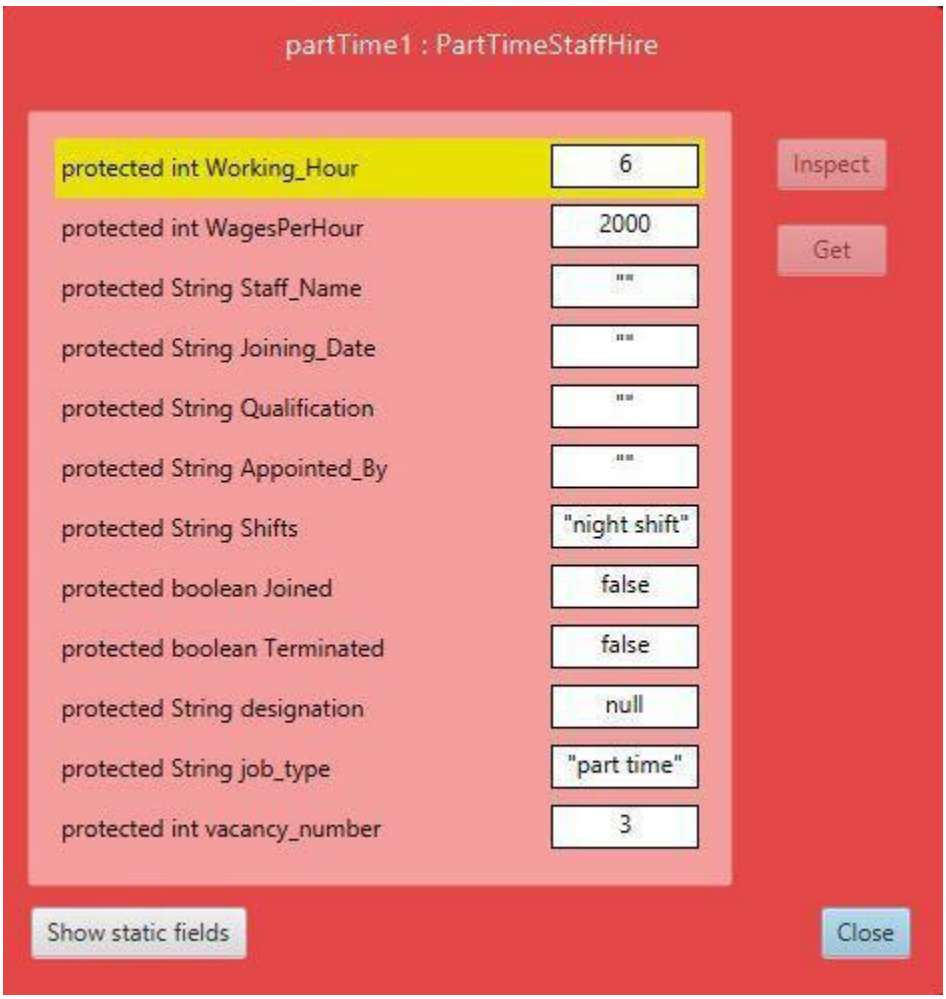
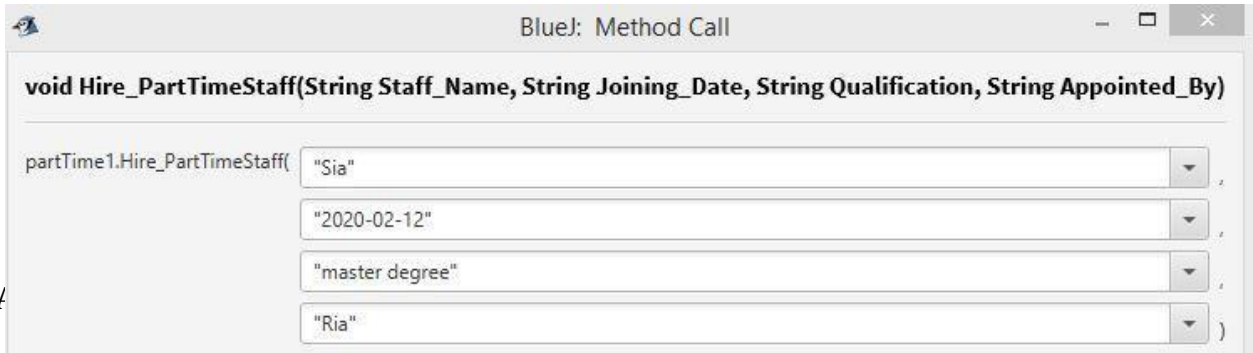


Figure 7: inspection for PartTimeStaffHire



PRA

Figure 8: method call for PartTimeStaffHire

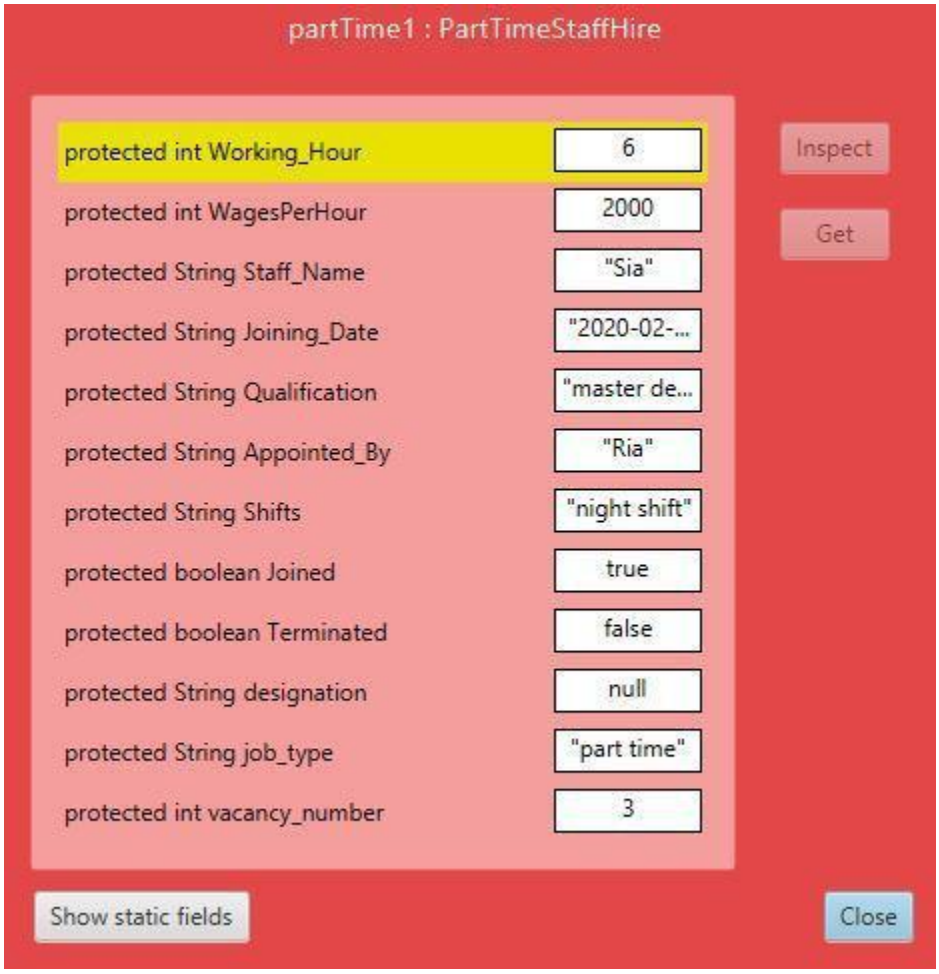


Figure 9: inspection for PartTimeStaffHire

Test 1

Objective	To inspect PartTimeStaffHire class, appoint value to hire part time staff and re-inspect the PartTimeStaffHire class
Action	The value assigned in PartTimestaffHire class are: Vacancy_no=3 Designation=”constructor” Job-type=”part time” Salary=”2000” Working_Hour=6 Inspection of PartTimeStaffHire Void Hire is called staffName=”Sia” joiningDate=”2020-02-12” qualification=”master degree” appointedby=”Ria” Re-inspect the PartTimeStaffHire
Expected Result	Entered data to be shown in inspection box
Actual Result	Entered data were shown in inspection box
Conclusion	Test pass

PartTimeStaffHire

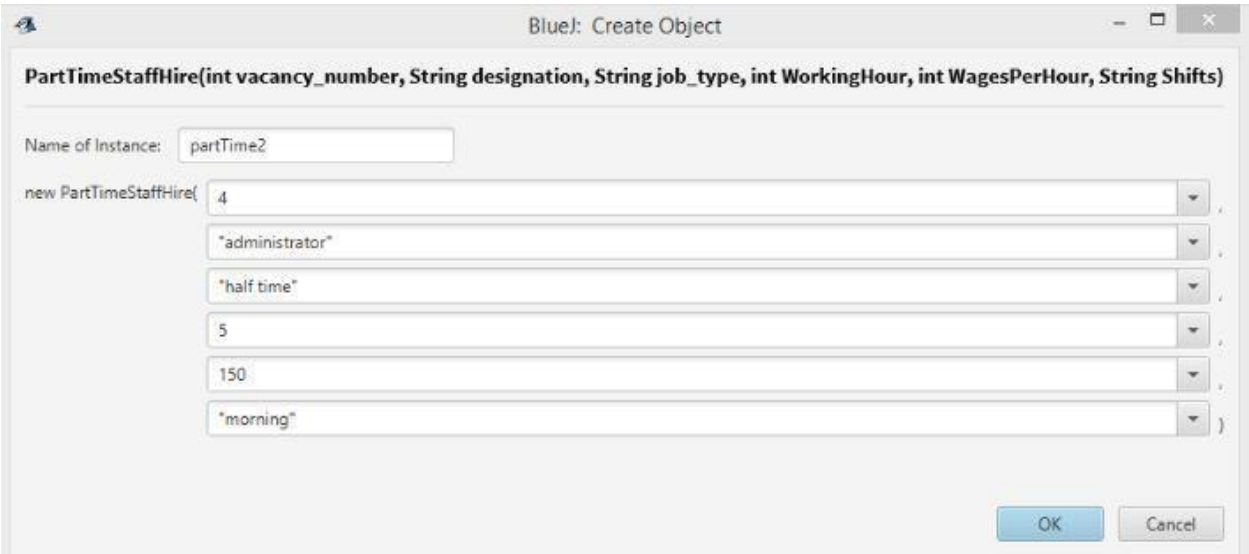


Figure 10: creating objectl for PartTimeStaffHire

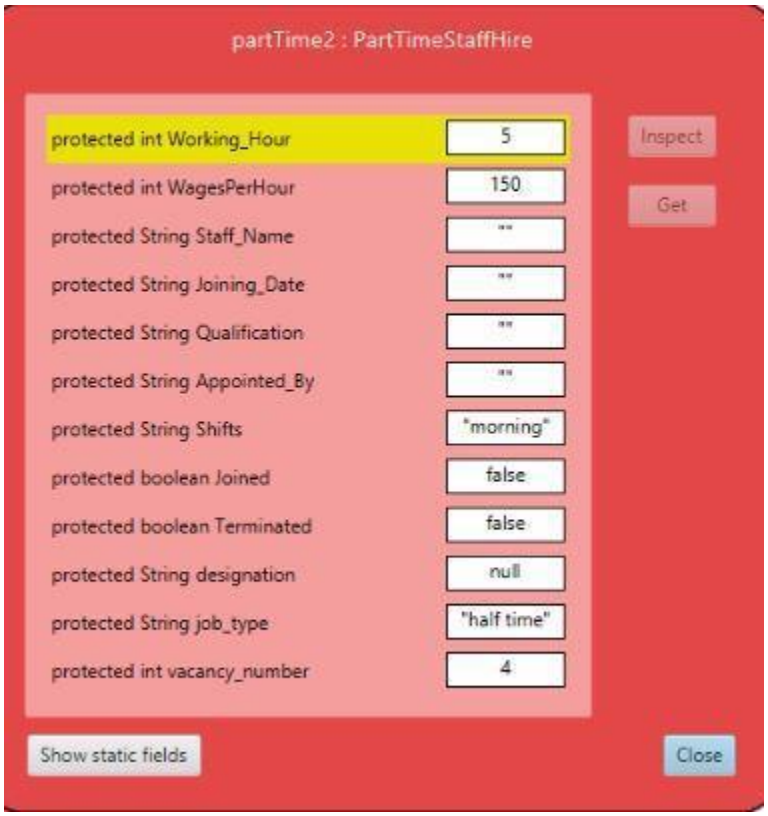


Figure 11: inspection for PartTimeStaffHire

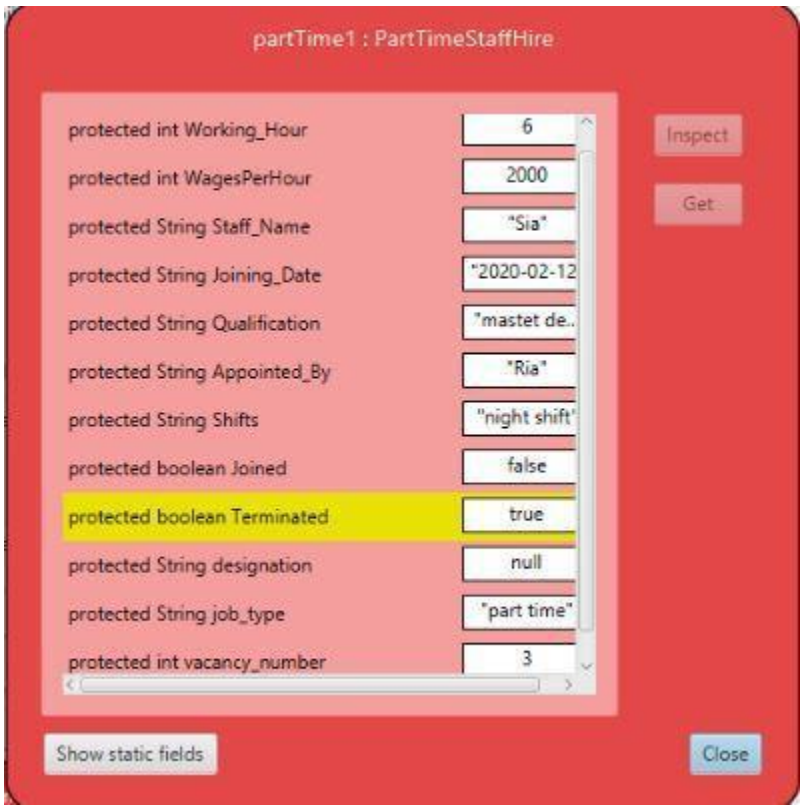


Figure 12: re-inspection for PartTimeStaffHire

Test 3

Objective	Inspect PartTimeStaffHire appoint value to terminate staff and re-nspect the PartTimeStaffHire class

Action	The PartTimeStaffHire is called with the following argument Vacancy_no=4 Designation=”administrator” Job_type=”half time” workingHour=5 wagesPerHour=1500 shifts=”morning” Inspection of the PartTimeStaffHire class Void terminateStaff is called with the following argument Re-inspecion of the class PartTimeStaffHire
Expected Result	Entered data to be shown in inspection box
Actual Result	Entered data were shown in inspection box
Conclusion	Test pass

4)

PartTimeStaffHire [Details]

```
BlueJ: Terminal Window - coursebook 3
Options
The designation isconstructorThe job type ispart timeThe vacancy  number3
Staff name: Sia
Wager per hour: 2000
Working hour: 6
Joining date: 2020-02-12
Qualification: master-degree
Appointed by: RiaIncome per day: 12000

Can only enter input while your programming is running
```

Figure 13: displaying PartTimeStaffHire

FullTimeStaffHire [Details]

```
BlueJ: Terminal Window - coursebook 3
Options
The designation issupervisorThe job type isfull timeThe vacancy  number3
The staff name is: Tom
The salary of the Tom is: 5000
The working hour of Tom is: 12
Tom has joined on: 2020-01-01
The qualification of Tom is: graduate degree
Tom is appointed by: Jerry

Can only enter input while your programming is running
```

Figure 14: displaying for FullTimeStaffHire

Test-4

Objectives	To display the details of PartTimeStaffHire class and FullTimeStaffHire class.
Action	Display details.
Expected Output	Should display the detail of Full and Part Time Staff Hire class

Actual Output	
Result	Test successful

Error Detection:

There are three types of error in Java and they are:

Logical Error

Runtime Error

Syntax Error

Logical Error:

A logic error or logical error is a mistake in a program's source code that results in incorrect or unexpected behavior. It is a type of runtime error that may simply produce the wrong output or may cause a program to crash while running. (Productions, 2020)

The error in this program was because of logical error. The error is very simple but very hard to find. The error is I used single”=” instead of using“==”.I was able to solve this error and get the output I expected.

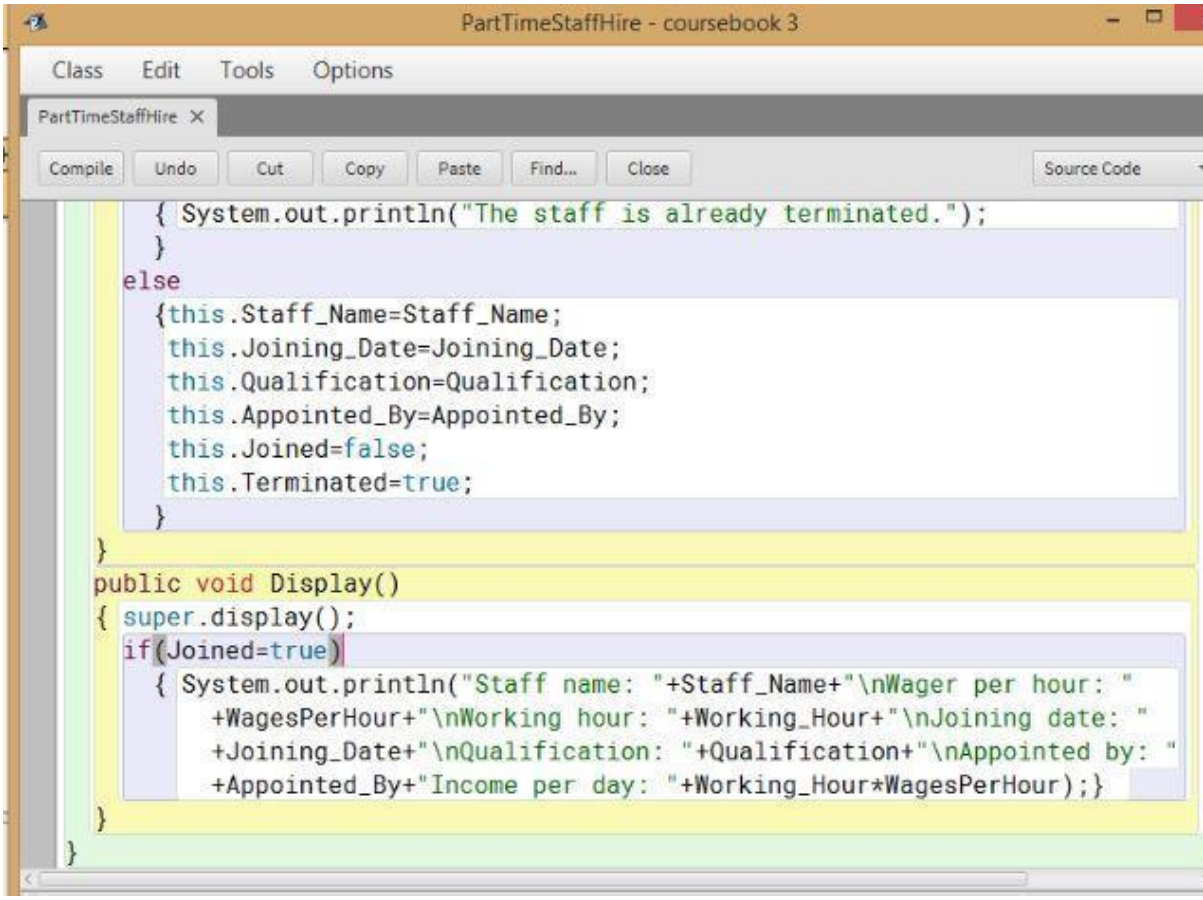


Figure 15: logical error

Runtime Error:

A runtime error is a program error that occurs while the program is running. The term is often used in contrast to other types of program errors, such as syntax errors and compile time errors. There are many different types of runtime errors. One example is a logic error, which produces the wrong output. (Productions, 2020)

The runtime error occurs while trying to execute the program. The error is because I didn't write the full time as string data type which I corrected by writing full time inside quotation mark.

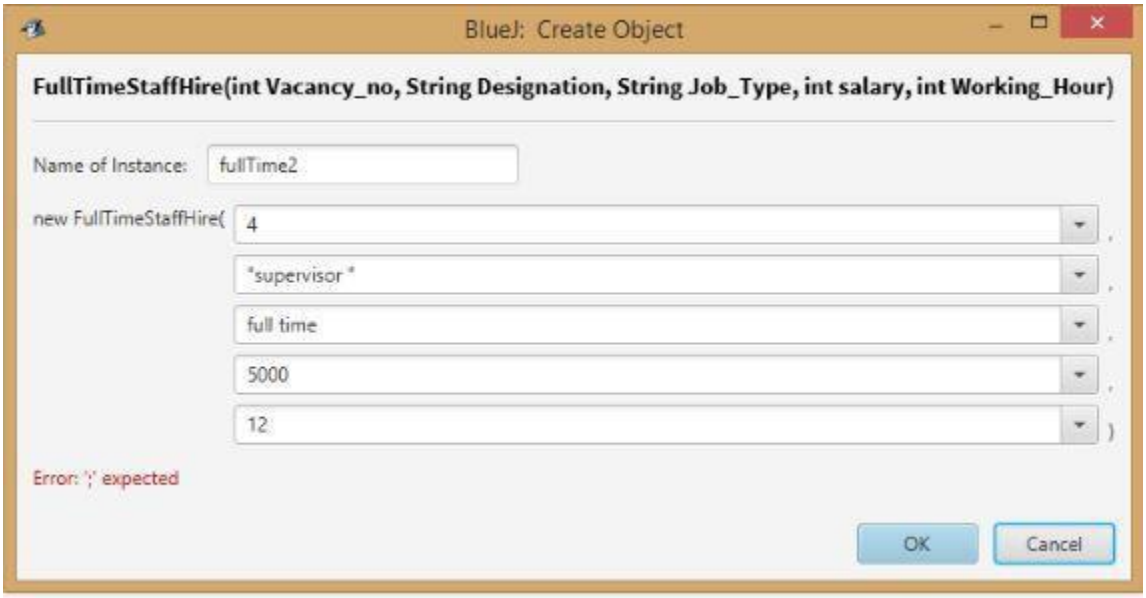


Figure 16: runtime error

Syntax Error:

A syntax error is an error in the source code of a program. Since computer programs must follow strict syntax to compile correctly, any aspects of the code that do not conform to the syntax of the programming language will produce a syntax error. (Productions, 2020)

During compiling of my program the error message shows a quotation mark to be missing. I checked my program and I am able to find the error. The error is very minor, but also affects the program strongly.

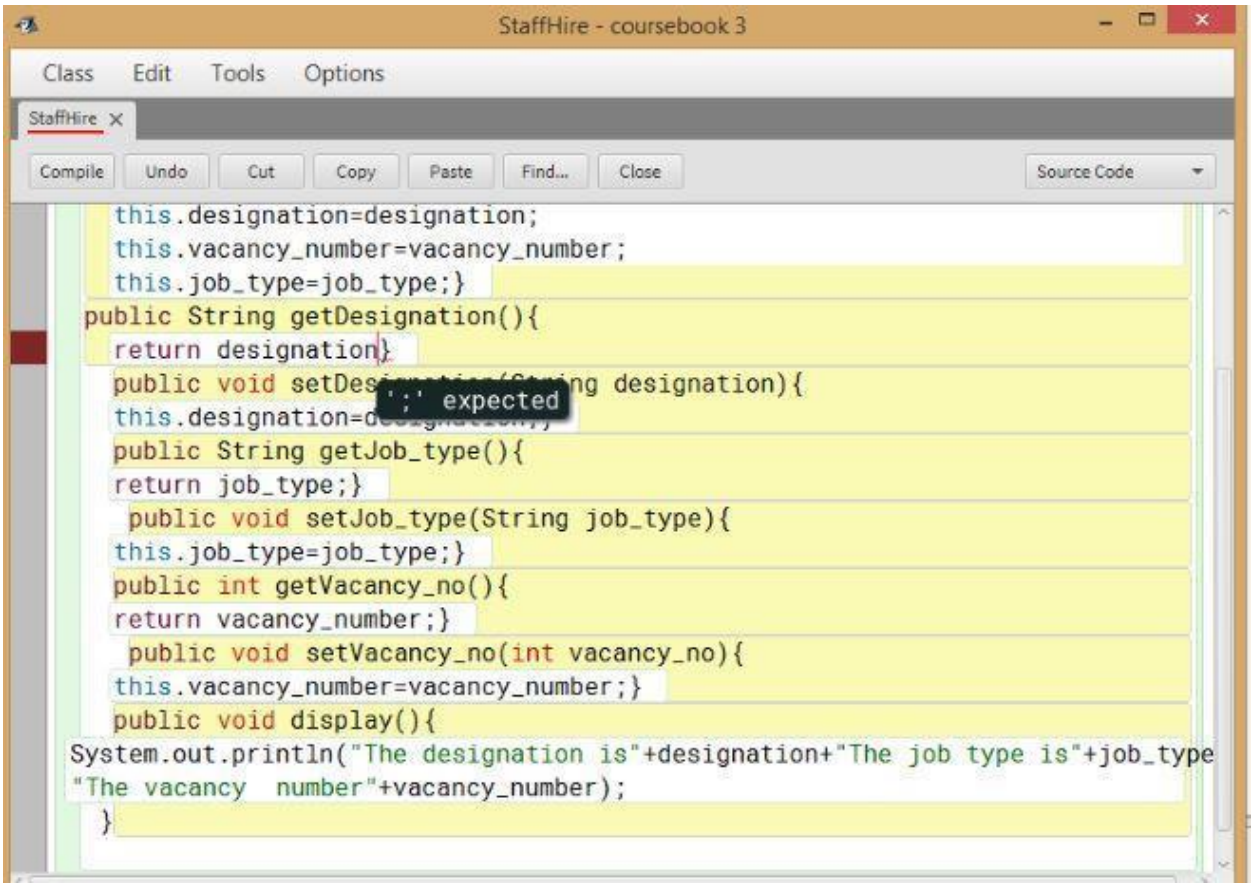


Figure 17: syntax error

Conclusion:

This coursebook was all about coding and running a certain programs while inserting various attributes. It was a quite difficult assignment for me. It made me researc about the project in various aspects. I took the help of the internet, friends and tutors in order to complete this assignment. Number of new things were known and a lot about coding was learned. While running my program I came with various error which made me research about the error and for its solution more and more. As a result a lot of new things were discovered which would surely help me in future.

Appendix:

Creating StaffHire class:

```
public class StaffHire
```

```
{

protected String designation; protected String job_type; protected int vacancy_number;

StaffHire(String designation,int vacancy_number,String job_type){
this.designation=designation; this.vacancy_number=vacancy_number;
this.job_type=job_type;}

public String getDesignation(){ return designation;}

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

public String getJob_type(){ return job_type;}

public void setJob_type(String job_type){ this.job_type=job_type;}

public int getVacancy_no(){ return vacancy_number;}

public void setVacancy_no(int vacancy_no){ this.vacancy_number=vacancy_number;}
public void display(){

System.out.println("The designation is"+designation+"The job type is"+job_type+ "The
vacancy number"+vacancy_number);

}



} public class StaffHire
```

```
{

protected String designation; protected String job_type; protected int vacancy_number;

StaffHire(String designation,int vacancy_number,String job_type){
this.designation=designation; this.vacancy_number=vacancy_number;
this.job_type=job_type;}

public String getDesignation(){ return designation;}

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

public String getJob_type(){ return job_type;}

public void setJob_type(String job_type){ this.job_type=job_type;}

public int getVacancy_no(){ return vacancy_number;}

public void setVacancy_no(int vacancy_no){ this.vacancy_number=vacancy_number;}
public void display(){

System.out.println("The designation is"+designation+"The job type is"+job_type+ "The
vacancy number"+vacancy_number);

}

}
```

Creating FullTimeStaffHireClass:

```
public class FullTimeStaffHire extends StaffHire{ private int salary;

private int Working_Hour; private String Staff_Name; private String Joining_Date; private String Qualification; private String Appointed_By; private boolean Joined;


FullTimeStaffHire(int Vacancy_no, String Designation, String Job_Type, int salary, int Working_Hour){

super(Designation,Vacancy_no,Job_Type); this.salary=salary; this.Working_Hour=Working_Hour; this.Staff_Name="";

this.Joining_Date=""; this.Qualification=""; this.Appointed_By=""; this.Joined=false;

}



int getsalary(){ return salary;

}
```

```
int getWorking_Hour(){ return Working_Hour;
}

String getStaff_Name(){ return Staff_Name;
}

String getJoining_Date(){ return Joining_Date;
}

String getQualification(){ return Qualification;
}

String getAppointed_By(){ return Appointed_By;
}

boolean getJoined(){ return Joined;
}

public void setSalary(int Salary){
```

```
if (Joined){

System.out.println("Salary of the"+ Staff_Name+ " is not changeable");

}

else{

this.salary=salary;

}

}


public void setWorkingHour(int Working_Hour){ this.Working_Hour=Working_Hour;

}


public void FullTimeStaff_Hire(String Staff_Name, String Joining_Date, String Qualification,
String Appointed_By){

if (Joined){

System.out.println(Staff_Name+" has already joined the organization on "+ Joining_Date);

}

else{ this.Staff_Name=Staff_Name; this.Joining_Date=Joining_Date;
this.Qualification=Qualification;


this.Appointed_By=Appointed_By; this.Joined=true;

}

}


public void Display(){
```

```
super.display();

if(Joined==true){

System.out.println("The staff name is: "+ Staff_Name); System.out.println("The salary of the
"+ Staff_Name + " is: "+ salary);

System.out.println("The working hour of "+ Staff_Name+ " is: "+ Working_Hour);
System.out.println(Staff_Name+" has joined on: "+ Joining_Date); System.out.println("The
qualification of "+Staff_Name+" is: "+Qualification); System.out.println(Staff_Name+" is
appointed by: "+Appointed_By);
}

}

}
```

Creating PartTimeStaffHire class:

```
public class PartTimeStaffHire extends StaffHire

{ protected int Working_Hour; protected int WagesPerHour; protected String Staff_Name;
protected String Joining_Date; protected String Qualification; protected String
Appointed_By; protected String Shifts; protected boolean Joined;



protected boolean Terminated;

PartTimeStaffHire(int vacancy_number,String designation,String job_type,int
WorkingHour,int WagesPerHour,String Shifts)

{

super(designation,vacancy_number,job_type); this.Working_Hour=WorkingHour;
this.WagesPerHour=WagesPerHour; this.Shifts=Shifts;

this.Staff_Name=""; this.Joining_Date=""; this.Qualification=""; this.Appointed_By="";
this.Joined=false; this.Terminated=false;
```

```
}
public int getWorkingHour(String Shifts)
{ if(Joined==true)
{
System.out.println("Shift: "+Shifts);
}
else
{
this.Shifts=Shifts; System.out.println("New shift: "+Shifts);
}
return Working_Hour;
```

```
}
public int getWagesPerHour()
{ return WagesPerHour;
}
public String getStaffName() { return Staff_Name;
}
public String getJoiningDate()
{ return Joining_Date;
}
public String getQualification() { return Qualification;
}
public String getAppointedbBy()
{ return Appointed_By;
}
public String getShifts() { return Shifts;
}
public boolean getJoined()
{ return Joined;
}
public boolean getTerminated() { return Terminated;
}
public void Hire_PartTimeStaff(String Staff_Name,String Joining_Date,
```

```
String Qualification,String Appointed_By)

{ if(Joined==true)

{ System.out.print("Staff name: "+Staff_Name+"\nJoin date"+Joining_Date);

}

else {this.Staff_Name=Staff_Name; this.Joining_Date=Joining_Date;
this.Qualification=Qualification;

this.Appointed_By=Appointed_By; this.Joined=true; this.Terminated=false;

}

}

public void Terminate_Staff()

{ if(Terminated==true)

{ System.out.println("The staff is already terminated.");

}

else {this.Staff_Name=Staff_Name; this.Joining_Date=Joining_Date;
this.Qualification=Qualification;

this.Appointed_By=Appointed_By; this.Joined=false; this.Terminated=true;

}

}
```

```
public void Display()

{ super.display(); if(Joined==true)

{ System.out.println("Staff name: "+Staff_Name+"\nWager per hour: "
+WagesPerHour+"\nWorking hour: "+Working_Hour+"\nJoining date: "
+Joining_Date+"\nQualification: "+Qualification+"\nAppointed by: "
+Appointed_By+"Income per day: "+Working_Hour*WagesPerHour);}

}

} public class PartTimeStaffHire

extends StaffHire

{ protected int Working_Hour; protected int WagesPerHour; protected String Staff_Name;
protected String Joining_Date; protected String Qualification; protected String
Appointed_By; protected String Shifts; protected boolean Joined; protected boolean
Terminated;

PartTimeStaffHire(int vacancy_number,String designation,String job_type,int
WorkingHour,int WagesPerHour,String Shifts)

{

super(designation,vacancy_number,job_type); this.Working_Hour=WorkingHour;
this.WagesPerHour=WagesPerHour; this.Shifts=Shifts;
```

```
this.Staff_Name=""; this.Joining_Date=""; this.Qualification=""; this.Appointed_By="";
this.Joined=false; this.Terminated=false;

}

public int getWorkingHour(String Shifts)

{ if(Joined==true)

{

System.out.println("Shift: "+Shifts);

}

else

{

this.Shifts=Shifts; System.out.println("New shift: "+Shifts);

}

return Working_Hour;

}

public int getWagesPerHour()

{ return WagesPerHour;

}

public String getStaffName() { return Staff_Name;

}

public String getJoiningDate()
```

```
{ return Joining_Date;

}

public String getQualification() { return Qualification;

}

public String getAppointedbBy()

{ return Appointed_By;

}

public String getShifts() { return Shifts;

}

public boolean getJoined() { return Joined;

}

public boolean getTerminated()

{ return Terminated;

}

public void Hire_PartTimeStaff(String Staff_Name,String Joining_Date, String
Qualification,String Appointed_By)

{ if(Joined==true)

{ System.out.print("Staff name: "+Staff_Name+"\nJoin date"+Joining_Date);

}

else

{this.Staff_Name=Staff_Name;

this.Joining_Date=Joining_Date;

this.Qualification=Qualification;
```

```
this.Appointed_By=Appointed_By; this.Joined=true; this.Terminated=false;

}

}

public void Terminate_Staff() { if(Terminated==true)

{ System.out.println("The staff is already terminated.");

}

else {this.Staff_Name=Staff_Name; this.Joining_Date=Joining_Date;
this.Qualification=Qualification;

this.Appointed_By=Appointed_By; this.Joined=false; this.Terminated=true;

}

}

public void Display()

{ super.display(); if(Joined==true)

{ System.out.println("Staff name: "+Staff_Name+"\nWager per hour: "
+WagesPerHour+"\nWorking hour: "+Working_Hour+"\nJoining date: "

+Joining_Date+"\nQualification: "+Qualification+"\nAppointed by: "

+Appointed_By+"Income per day: "+Working_Hour*WagesPerHour);}

} }
```