**Module Code & Module Title**
**CS4001NI Programming**

**Assessment Weightage & Type**
**50% Individual Coursework**

**Year and Semester**
**2019-20 Autumn**

**Student Name: Anish Sherchan**
**Group: C2**
**London Met ID:19030714**
**College ID: NP01CP4A190171**
**Assignment Due Date: 2020/06/05**
**Assignment Submission Date: 2020/06/05**

## Contents

## Tables of figure

## a) Introduction

### I. JAVA

Java is a programming language and computing platform first released by Sun Microsystems in 1995. There are lots of applications and websites that will not work unless you have Java installed, and more are created every day. Java is fast, secure, and reliable. From laptops to datacenters, game consoles to scientific supercomputers, cell phones to the Internet, Java is everywhere! (JAVA, 2020)

### II. GUI (Graphical User Interface)

A GUI (graphical user interface) is a system of interactive visual components for computer software. A GUI displays objects that convey information, and represent actions that can be taken by the user. The objects change color, size, or visibility when the user interacts with them. GUI objects include icons, cursors, and buttons. These graphical elements are sometimes enhanced with sounds, or visual effects like transparency and drop shadows.

A GUI is considered to be more user-friendly than a text-based command-line interface, such as MS-DOS, or the shell of Unix-like operating systems. The GUI was first developed at Xerox PARC by Alan Kay, Douglas Engelbart, and a group of other researchers in 1981. Later, Apple introduced the Lisa computer with a GUI on January 19, 1983.

A GUI uses windows, icons, and menus to carry out commands, such as opening, deleting, and moving files. Although a GUI operating system is primarily navigated using a mouse, a keyboard can also be used via keyboard shortcuts or the arrow keys. As an example, if you wanted to open a program on a GUI system, you would move the mouse pointer to the program's icon and double-click it. (ComputerHopes, 2020)

### III. CUI (Character User Interface)

Short for character user interface or command-line user interface, CUI is a way for users to interact with computer programs. It works by allowing the user (client) to issue commands as one or more lines of text (referred to as command lines) to a program. Good examples CUIs are MS-DOS and the Windows Command Prompt. One of the CUI's uses is that it provides an easy way to implement programming scripts.

The command-line user interface was the primary method of communicating with a computer from the first machines and through the 1980s. Although it may still be accessed in today's operating systems, it is utilized far less due to the ease of use and familiarity of the GUI (graphical user interface). The CUI, however, is still preferred by many advanced end users as its features provide them with more comprehensive control over an operating system's functions. There are other command lines in addition to the ones mentioned above, namely, Terminal, and the Linux command line. (ComputerHopes, 2020)

## IV. Summary about project

This project was developed for an organization. This program helps in staff hiring process in any organization by the help of simple program and GUI (Graphical User Interface) that was developed in java platform called BlueJ and also can be developed in any other text editor's which supports and has JDK kits and other program's to read the developed java program.

This project consists of getter method, display method, Supper class, Derived Classes, Duplicate Vacancy number detector method, Staff Hiring method, Vacancy announcing method, terminating a staff method, etc. The getter method is required for taking in values from the users and to return the value. Here in this project many methods have their own specified methods and has a particular task and they perform those task whenever they are called. In this project there are 4 java files which are inter related with each other. The files are StaffHire(Parent class), FullTimeHire (Child Class), PartTimeHire (Child Class) and a GUI class which is INGNepal.

Display is just a simple code which displays the current status of the program and values that has been entered. Supper class or parental class is a class that is the parent class of the reaming child class. Derived class or a child class is a class that is derived from parental class.

## V. BlueJ

BlueJ is a Java integrated development environment which was solely designed for educational purposes and for small-scale software development to some extent. It is an IDE or Integrated development environment for doing JAVA programming. BlueJ was started by Michael Kolling and John Rosenberg in 1999 at Monash University, Australia. Presently this software was maintained by a team at King's College London, England, where Kolling works. A JDK (Java development kit) is required to run BlueJ. By the way, a JDK is a software development environment by which applets and JAVA applications can be developed. (BlueJ, 2020)

### VI. Things Used for designing GUI

There are many inbuilt things which has been used during the process of designing the developed GUI. The inbuilt package which is used for giving or creating colors for fonts, panels, frames are used in the GUI for giving a fine finish in color for frame, panel and fonts. The main thing which has been very helpful for designing this GUI was the set Bound method which helped me throughout the project for moving components on desired place throughout the frame. This set Bound method has all 4 parameters as Integer and takes the parameters as (X-axis, Y-axis, width, height). So this were all the things which has been used for designing the whole developed GUI by using the inbuilt packages in java.

## b) 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 modeling 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)

### I. INGNepal Class Diagram

The Class Diagram of class INGNepal is given below in new page.

| Class : INGNepal | |
|---|---|
| - cameVacancy: | INT |
| - cameSalary: | INT |
| - cameWorkinghr: | INT |
| - Selection: | String |
| - cameJob: | String |
| - cameShift: | String |
| - phoCheck: | Boolean |
| - checkIn | Boolean |
| - clls | Boolean |
| - anish | Boolean |
| - Frame | JFrame |
| - SubFrame | JFrame |
| - SubFrame1 | JFrame |
| - SubFrame2 | JFrame |
| - Subframe3 | JFrame |
| - CheckingFrame | JFrame |
| - TerminateFrame | JFrame |
| - Panelmainfu | JPanel |
| - Panelmain | JPanel |
| - PanelAppoint | JPanel |
| - PanelInfo | JPanel |
| - PanelTermi | JPanel |
| - VacFull | JButton |
| - AppFull | JButton |
| - Display | JButton |
| - Terminate | JButton |
| - OkBtn | JButton |
| - Cancel | JButton |
| - Partsub | |

| | |
|---|---|
| - Cleara | JButton |
| - clearB | JButton |
| - Add1 | JButton |
| - Add2 | JButton |
| - terminateBtn | JButton |
| - Checkcle | JButton |
| - Vacinput1 | JTextField |
| - DesigInput1 | JTextField |
| - JobInput1 | JTextField |
| - SalaryInput1 | JTextField |
| - WorkingIput1 | JTextField |
| - WagesInput1 | JTextField |
| - ShiftInpu1 | JTextField |
| - InsdieInput1 | JTextField |
| - InputVc | JTextField |
| - InputJB | JTextField |
| - Inputdesin | JTextField |
| - InputAppoint | JTextField |
| - InputShift | JTextField |
| - InputWages | JTextField |
| - InputSalary2 | JTextField |
| - InputTermi | JTextField |
| - InputFinal2 | JTextField |
| | JTextField |

| | |
|---|---|
| + vacancyReturn (): | INT |
| + workinghrReturn (): | INT |
| + salaryReturn (): | INT |
| + terminatereturn (): | INT |
| + QualificationReturn (): | String |
| + AppointedReturn(): | String |
| +JoiningReturn(): | String |
| + ShiftReturn(): | String |
| +:DesignationReturn(): | String |
| +StaffReturn(): | String |
| +WagesPerHrReturn(): | INT |
| + hire Part Time Staff (String staff Name, joining date, qualification, appointed by): | Void |
| + hire Full Time Staff (String staff Name, joining date, qualification, appointed by): | Void |
| + displayMet (): | Void |
| +main(): | Void |
| +VacAddmethod(): | Void |
| +AppointMet: | Void |
| +HireFull(): | Void |
| +RemoveGui(): | Void |
| +actionPerformed(): | Void |
| +mainTerminate(): | Void |
| +duplicStaff(): | Void |
| +Allstorage(): | Void |

*Figure 1*: Class Diagram of INGNepal

## c) Relation Diagram of project

Relations Diagrams are drawn to show all the different relationships between factors, areas, or processes. Why are they worthwhile? Because they make it easy to pick out the factors in a situation which are the ones which are driving many of the other symptoms or factors. (SkyMark, 2020)

The relation diagram of the project is given below. Here, In the relation diagram StaffHire is Parent Class where it has extended FullTimeHire and PartTimehire. Also INGNepal is related with all the classes associatively.

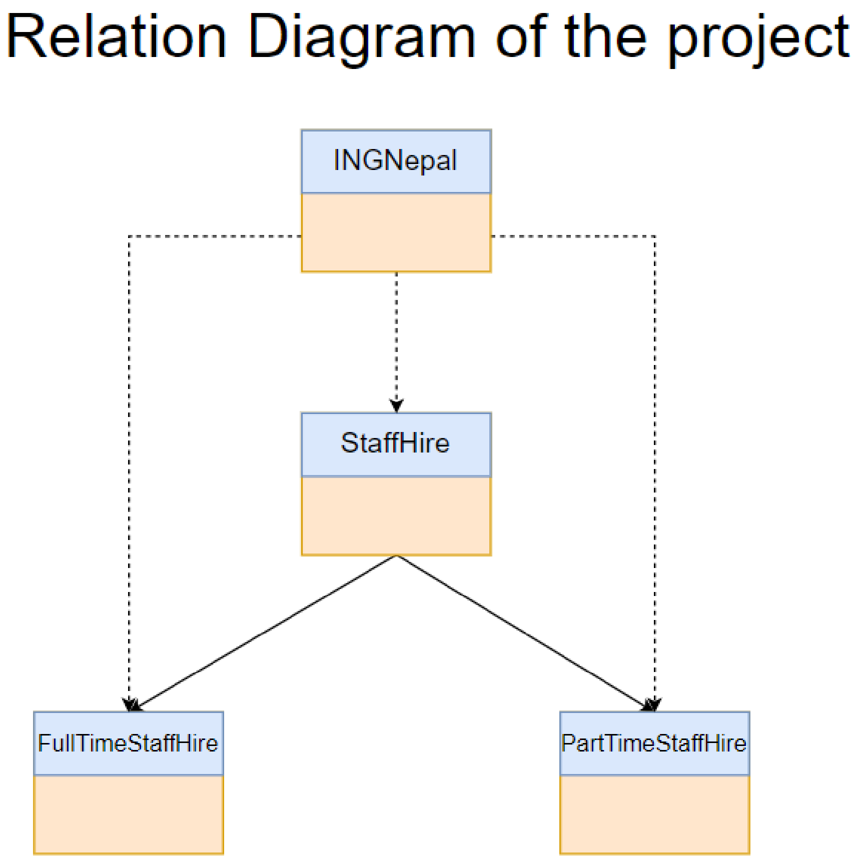


*Figure 2*: Relation Diagram

## d) Pseudocode for each method of INGNepal and short description.

### I. INGNepal (Class which relates all the classes associatively)

In this project INGNepal is a class which relates all the classes associatively and is the class which consists of main GUI for the project. This class consists of many methods which performs specific task and is very useful for proper functioning of the GUI. In this topic explanation about all the methods which are present in the INGNepal class and pseudocode of all the method will be described.

#### I. vacancyReturn()

The vacancyReturn() method is a method which is present in INGNepal class where it takes in the input from text area that is given by the user as string and is again converted into Integer by converting the string input to Integer. In this method user gives information about Vacancy number of staff. The Pseudocode of the given method is given below.

**Function**

```
public int vacancyReturn(){

CALL
    DO
        return Integer.parseInt(VacInput1.getText());

    END DO

END

 }
```

#### II. workHourReturn()

The workHourReturn() method is a method which is present in INGNepal class where it takes in the input from text area that is given by the user as string and is again converted into Integer by converting the string input to Integer. In this method user gives information about working hour of staff. The Pseudocode of the given method is given below.

```
Function

public int workHourReturn(){

CALL

    DO
        return Integer.parseInt(WorkHoInput1.getText());

    END DO

END

}
```

### III. salaryReturn()

The salaryReturn() method is a method which is present in INGNepal class where it takes in the input from text area that is given by the user as string and is again converted into Integer by converting the string input to Integer. In this method user gives information about Salary of fulltime Staff. The Pseudocode of the given method is given below.

```
Function
public int salaryReturn(){

CALL
    DO
            return Integer.parseInt(SalaryInput1.getText());
    END DO
END
```

}

## IV. wagePerHourReturn()

The wagePerHourReturn() method is a method which is present in INGNepal class where it takes in the input from text area that is given by the user as string and is again converted into Integer by converting the string input to Integer. In this method user gives information about wages per hour of part-time Staff. The Pseudocode of the given method is given below.

```
Function
public int wagePerHourReturn(){
        CALL
                DO
                        return Integer.parseInt(WageInput1.getText());

                END DO
        END
 }
```

## V. Terminateboxinret()

The TerminateboxinretReturn() method is a method which is present in INGNepal class where it takes in the input from text area that is given by the user as string and is again converted into Integer by converting the string input to Integer. In this method user gives information about Vacancy number of part-time Staff which has to be terminated . The Pseudocode of the given method is given below.

```
Function
public int Terminateboxinret(){
        CALL

        DO
                return Integer.parseInt(Terminateboxin.getText());

        END DO
END
}
```

## VI. designationReturn()

The designationReturn() method is a method which is present in INGNepal class where it takes in the input from text area that is given by the user as string and stores it and returns the value whenever it is called. In this method user gives information about Designation of staff. The Pseudocode of the given method is given below.

```
Function
public String designationReturn(){
    CALL
        DO
            return DegisinationInput1.getText();

        END DO
    END
}
```

## VII. shiftReturn()

The shiftReturn() method is a method which is present in INGNepal class where it takes in the input from text area that is given by the user as string and stores it and returns the value whenever it is called. In this method user gives information about shifts of staff. The Pseudocode of the given method is given below.

```
Function
public String shiftReturn(){
      CALL
        DO
            return ShiftInput1.getText();
        END DO
      END
 }
```

## VIII. staffNameReturn()

The staffNameReturn() method is a method which is present in INGNepal class where it takes in the input from text area that is given by the user as string and stores it and returns the value whenever it is called. In this method user gives information about Name of staff. The Pseudocode of the given method is given below.

```
Function
public String staffNameReturn(){
        CALL
                DO
                        return inputStaff.getText();

                END DO
        END
 }
```

## IX. appointedByReturn()

The appointedByReturn() method is a method which is present in INGNepal class where it takes in the input from text area that is given by the user as string and stores it and returns the value whenever it is called. In this method user gives information about Name of person who appointed the staff. The Pseudocode of the given method is given below.

```
Function
public String appointedByReturn(){
CALL
                DO

                        return inputdappoint.getText();
                END DO
        END
 }
```

## X. joinDateReturn()

The joinDateReturn() method is a method which is present in INGNepal class where it takes in the input from text area that is given by the user as string and stores it and returns the value whenever it is called. In this method user gives information about joining date of staff. The Pseudocode of the given method is given below.

```
Funcation
public String joinDateReturn(){
        CALL
                DO
                        return inputjdd.getText();
                END DO
        END
}
```

## XI. qualificationReturn()

The qualificationReturn() method is a method which is present in INGNepal class where it takes in the input from text area that is given by the user as string and stores it and returns the value whenever it is called. In this method user gives information about qualification of staff. The Pseudocode of the given method is given below.

```
Function
public String qualificationReturn(){
CALL
        DO
                return inputquliflication.getText();
        END DO

END
}
```

## XII. void main(String[ ] args)

This is the main method of the INGNepal class which consists (String [ ] args) as arguments and whenever the class gets compiled and starts to run the compiler looks for this method or a method which has parameters as (String[ ] args) and runs it. The pseudocode for this method is given below.

```
**Function**
public static void main(String[] args){
        DO
                Initialize obj1 as Object
                CALL main() as obj1.Main()
        END DO
End
}
```

### XIII. main()

This is the main GUI class or index GUI class of whole program. This is the class which gets called first whenever the whole project is compiled and executed. In this Class, A main frame is declared, 2 panels are declared and are moved in desired place with the help of set Bound method, also 6 buttons are kept in desired place using set Bound method. The pseudocode of the main() method is given below.

```
Function
private void main()
        DO
                Initialize Jframe as frame
                        Set Preferred Size of frame as (550,500)
                Initialize Color c as rgb (212,212,212)
                        Set Background color of frame as (c)
                        set resizable of frame as (False)
                        set frame layout as (null)

                Initialize JPanel as panel(null)
                        Set Bounds of panel as (1,1,550,130)
                Initialize Color C as rgb(17,164,242)
                        Set Background color of panel as (C)

                Initialize Jlable as label1(i)
                        Set font of label1 as (“Calibri”, Bold, size=70)
                        Set font color as (“Green”)
                        Set Bound of label1 as (240,30,30,70
ADD label1 in panel

Initialize Jlable as label2(ng)
    Set font of label2 as (“Calibri”, Bold, size=70)
    Set font color as (“WHITE”)
    Set Bound of label2 as (258,30,90,70)
ADD label2 in panel

Initialize Jlable as label3(Nepal Group)
    Set font of label3 as (“MONACO”, PLAIN, size=16)
    Set font color as (“Default”)
    Set Bound of label3 as (260,85,150,40)
ADD label3 in panel

Initialize MAIN JPanel as PanelMain(null)
    Set Bounds of PanelMain as (20,130,500,350)
    Set Background color of panel as (c)

Initialize JButton as VacFull(Vacancy for full time)
    Set Bound of VacFull(30,80,180,35)
    Set Font of Vacfull as (“Times New Roman”, Plain, size=16)
ADD Action listener in VacFull (“Calls GUI For Fulltime Vacancy”)
ADD VacFull in PanelMain

Initialize JButton as VacPart(Vacancy for Part time)
    Set Bound of VacPart(290,80,180,35)
    Set Font of VacPart as (“Times New Roman”, Plain, size=16)
ADD Action listener in VacPart (“Calls GUI for Parttime Vacancy”)
ADD VacPart in PanelMain

Initialize JButton as AppFull(Hire full time)
    Set Bound of AppFull(30,150,180,35)
    Set Font of Appfull as (“Times New Roman”, Plain, size=16)
ADD Action listener in AppFull (“Calls GUI For Hiring Fulltime staff”)
ADD AppFull in PanelMain
```

```
Initialize JButton as AppPart(Hire Part time)
    Set Bound of AppPart(290,150,180,35)
    Set Font of AppPart as (“Times New Roman”, Plain, size=16)
ADD Action listener in AppPart (“Calls GUI For Hiring Parttime staff”)
ADD AppPart in PanelMain

Initialize JButton as Display(Display)
    Set Bound of Display (30,220,180,35)
    Set Font of Display as (“Times New Roman”, Plain, size=16)
ADD Action listener in Display (“Displays All records of staff in console”)
ADD Diplay in PanelMain

Initialize JButton as terminate(Terminate)
    Set Bound of terminate (290,220,180,35)
    Set Font of terminate as (“Times New Roman”, Plain, size=16)
ADD Action listener in terminate (“Calls GUI of terminate”)
ADD terminate in PanelMain

ADD PanelMain in frame
    Pack frame
    Set default closed operation (Exit on close)
    Set frame visible as (true)
```

## XIV. VacAddmethod()

` This method is a method which is called from index GUI whenever user presses the button of fulltime vacancy or parttime vacancy in this method a GUI is made for announcing vacancy for full time and parttime. The pseudocode of this method is given below.

```
Function
Private void VacAddmethod()
    DO
```

```
**Initialize** Jframe as Subframe
        Set Preferred Size of frame as (550,6500)
**Initialize** Color c as rgb (212,212,212)
        Set Background color of frame as (c)
        set resizable of frame as (False)
        set frame layout as (null)

**Initialize** JPanel as FVpanel(null)
        Set Bounds of FVpanel as (1,1,550,130)
**Initialize** Color C as rgb(17,164,242)
        Set Background color of FVpanel as (C)

**Initialize** Jlable as label1(i)
        Set font of label1 as ("Calibri", Bold, size=70)
        Set font color as ("Green")
        Set Bound of label1 as (240,30,30,70
**ADD** label1 in FVpanel

**Initialize** Jlable as label2(ng)
        Set font of label2 as ("Calibri", Bold, size=70)
        Set font color as ("WHITE")
        Set Bound of label2 as (258,30,90,70)
**ADD** label2 in FVpanel

**Initialize** Jlable as label3(Nepal Group)
        Set font of label3 as ("MONACO", PLAIN, size=16)
        Set font color as ("Default")
        Set Bound of label3 as (260,85,150,40)
**ADD** label3 in FVpanel

**Initialize** MAIN JPanel as panelmainfu(null)
        Set Bounds of panelmainfu as (20,130,500,470)
        Set Background color of panelmainfu as (c)

**Initialize** Jlable as titlelabel("Add Vacancy")
```

```
        Set font of label3 as (“Calibri”, PLAIN, size=30)
        Set font color as (“Black”)
        Set Bound of label3 as (150,0,380,50)
**ADD** titlelabel in panelmainfu

**Initialize** JLabel lbl1(“Vacancy Number : “)
        Set Bound of lbl1 as (20,90,160,30)
        Set font of lbl1 as (“Calibri” ,Plain, size=20)
        Set font color(Black)
**ADD** lbl1 in panelmainfu

**Initialize** JtextField as VacInput1
        Set Font of VacInput1 as (“Calibri”, plain, 20)
        Set Bound of VacInput (20,80,160,30)
**ADD** VacInput in panelmainfu

**Initialize** JLabel lbl2(“Designation : “)
        Set Bound of lbl2 as (20,140,160,30)
        Set font of lbl2 as (“Calibri” ,Plain, size=20)
        Set font color(Black)
**ADD** lbl2 in panelmainfu

**Initialize** JtextField as DesignationInput1
        Set Font of DesignationInput1 as (“Calibri”, plain, 20)
        Set Bound of DesignationInput (200,130,160,30)
**ADD** DesignationInput in panelmainfu

**Initialize** JLabel lbl3(“JobType : “)
        Set Bound of lbl3 as (20,190,160,30)
        Set font of lbl3 as (“Calibri” ,Plain, size=20)
        Set font color(Black)
**ADD** lbl3 in panelmainfu

**Initialize** JtextField as JobInput1
        Set Font of JobInput1 as (“Calibri”, plain, 20)
        Set Bound of JobInput (200,180,160,30)
        Set editable as (False)
**ADD** JobInput in panelmainfu

**Initialize** JLabel lbl4("Salary : ")
        Set Bound of lbl4 as (20,240,160,30)
        Set font of lbl4 as ("Calibri" ,Plain, size=20)
        Set font color(Black)
**ADD** lbl4 in panelmainfu

**Initialize** JtextField as SalaryInput1
        Set Font of SalaryInput1 as ("Calibri", plain, 20)
        Set Bound of SalaryInput (200,230,160,30)
**ADD** SalaryInput in panelmainfu

**Initialize** JLabel lbl5("Working Hour : ")
        Set Bound of lbl5 as (20,290,160,30)
        Set font of lbl5 as ("Calibri" ,Plain, size=20)
        Set font color(Black)
**ADD** lbl5 in panelmainfu

**Initialize** JtextField as WorkHOInput1
        Set Font of WorkHoInput1 as ("Calibri", plain, 20)
        Set Bound of WorkHoInput (200,280,160,30)
**ADD** WorkHoInput in panelmainfu

**Initialize** JLabel lbl7("Shift : ")
        Set Bound of lbl7 as (20,340,160,30)
        Set font of lbl7 as ("Calibri" ,Plain, size=20)
        Set font color(Black)
**ADD** lbl7 in panelmainfu

**Initialize** JtextField as ShiftInput1
        Set Font of ShiftInput1 as ("Calibri", plain, 20)
        Set Bound of ShiftInput (200,330,160,30)
        ADD ShiftInput in panelmainfu

        Initialize JButton as OKbtn(Add Vacancy)
            Set Bound of OKbtn(100,420,120,40)
            Set Font of OKbtn as (“Times New Roman”, Plain, size=16)
        ADD Action listener in oktn (“Adds information about vacancy in array list”)
        ADD OKbtn in panelmainfu

        Initialize JButton as cancel(Clear)
            Set Bound of cancel (250,420,120,40)
            Set Font of cancel as (“Times New Roman”, Plain, size=16)
        ADD Action listener in cancel (“Clears all text area ”)
        ADD cancel in panelmainfu

        ADD panelfu in Subframe
            Pack Subframe
            Set default closed operation (Exit on close)
            Set Subframe visible as (true)

        IF (Selection equals Full Time)
            ADD  lbl4
            ADD SalaryInput1
        END

        IF (Selection equals Part Time)
            ADD lbl6
            ADD WagesInput1
            ADD lbl7
            ADD Shift Input1
        END

END DO

}
```

**END**

## XV. AppointMet()

This is the method which get executed whenever user presses the hire button of Index GUI. In this GUI, a label of title Vacancy Number is kept and a textfield is kept alongside it to check the validity of the Vacancy Number. The pseudocode of the method is given below.

```
Function
Private void AppointMet()
        DO
                Initialize Jframe as Subframe2
                        Set Preferred Size of Subframe2 as (550,420)
                Initialize Color c as rgb (212,212,212)
                        Set Background color of Subframe2 as (c)
                        set resizable of Subframe2 as (False)
                        set Subframe2 layout as (null)

                Initialize JPanel as FVpanel(null)
                        Set Bounds of FVpanel as (1,1,550,120)
                Initialize Color C as rgb(17,164,242)
                        Set Background color of FVpanel as (C)

                Initialize Jlable as label1(i)
                        Set font of label1 as (“Calibri”, Bold, size=70)
                        Set font color as (“Green”)
                        Set Bound of label1 as (210,30,30,70
                ADD label1 in FVpanel

                Initialize Jlable as label2(ng)
                        Set font of label2 as (“Calibri”, Bold, size=70)
                        Set font color as (“WHITE”)
                        Set Bound of label2 as (223,30,90,70)
                ADD label2 in FVpanel
```

```
Initialize Jlable as label3(Nepal Group)
    Set font of label3 as ("MONACO", PLAIN, size=16)
    Set font color as ("Default")
    Set Bound of label3 as (220,85,150,40)
ADD label3 in FVpanel

Initialize MAIN JPanel as panelappoint (null)
    Set Bounds of panelappoint as (20,130,500,470)
    Set Background color of panelappoint as (c)

Initialize Jlable as titlelabel("Confirm Vacancy No.")
    Set font of titlelabel as ("Calibri", PLAIN, size=30)
    Set font color as ("Black")
    Set Bound of titlelabel as (150,0,380,50)
ADD titlelabel in panelappoint

Initialize JLabel lbl1("Vacancy Number : ")
    Set Bound of lbl1 as (20,110,250,20)
    Set font of lbl1 as ("Calibri" ,Plain, size=25)
    Set font color(Black)
ADD lbl1 in panelappoint

Initialize JtextField as InsideVac
    Set Font of InsideVac as ("Calibri", plain, 20)
    Set Bound of InsideVac (20,80,160,30)
ADD InsideVac in panelappoint

IF (Selection equals Full Time)
Initialize Jbutton fullsubmit (Check)
    Set Bounds (180, 180, 120, 40)
    ADD fullsubmit in panelappoint
    ADD fullsubmit Action listener (This checks for any other duplicate Vacancy
                                                  number)
END
```

```
                IF (Selection equals Part Time)
                        Initialize Jbutton Partsubmit (Check)
                        Set Bounds (180, 180, 120, 40)
                        ADD Partsubmit in panelappoint
                        ADD Partsubmit Action listener (This checks for any other duplicate
                                                                        Vacancy number)
                END

                ADD panelFV in Subframe2
                        Pack Subframe2
                        Set default closed operation (Exit on close)
                        Set Subframe2 visible as (true)


        END DO
}

END
```

## XVI. HireFinal()

This is the method which gets executed whenever user confirms the vacancy number. This method consists of A GUI which consists of many Labels and text area also it consists of 2 buttons on it. The pseudocode of the method is given below.

```
Function
Public void HireFullTime()
        DO
                Initialize Jframe as SubFrame1
                        Set Preferred Size of Subframe2 as (750,450)
                Initialize Color c as rgb (212,212,212)
                        Set Background color of Subframe2 as (c)
                        set resizable of Subframe2 as (False)
                        set Subframe2 layout as (null)

                Initialize JPanel as FVpanel(null)
        Set Bounds of FVpanel as (1,1,550,120)
**Initialize** Color C as rgb(17,164,242)
        Set Background color of FVpanel as (C)

**Initialize** Jlable as label1(i)
        Set font of label1 as (“Calibri”, Bold, size=70)
        Set font color as (“Green”)
        Set Bound of label1 as (210,30,30,70
**ADD** label1 in FVpanel

**Initialize** Jlable as label2(ng)
        Set font of label2 as (“Calibri”, Bold, size=70)
        Set font color as (“WHITE”)
        Set Bound of label2 as (223,30,90,70)
**ADD** label2 in FVpanel

**Initialize** Jlable as label3(Nepal Group)
        Set font of label3 as (“MONACO”, PLAIN, size=16)
        Set font color as (“Default”)
        Set Bound of label3 as (220,85,150,40)
**ADD** label3 in FVpanel

**Initialize** MAIN JPanel as panelInfo (null)
        Set Bounds of panelinfo as (10,140,710,375)
        Set Background color of panelinfo as (c)

**Initialize** Jlable as titlelabel(“Hire Staff”)
        Set font of titlelabel as (“Calibri”, PLAIN, size=30)
        Set font color as (“Black”)
        Set Bound of titlelabel as (265,125,250,40)
**ADD** titlelabel in panelinfo

**Initialize** JtextField as lblvacancy
        Set Font of lblvacancy as (“Calibri”, plain, 20)
        Set Bound of lblvacancy (20,80,160,30)
```

```
ADD lblvacancy in panelinfo

Initialize JtextField as lblJobtype
    Set Font of lblJobtype as ("Calibri", plain, 20)
    Set Bound of lblJobtype (20,160,160,30)
ADD lblJobtype in panelinfo

Initialize JtextField as lblDesigna
    Set Font of lblDesigna as ("Calibri", plain, 20)
    Set Bound of lblDesigna (20,200,160,30)
ADD lblDesigna in panelinfo

Initialize JtextField as lblstaffName
    Set Font of lblstaffName as ("Calibri", plain, 20)
    Set Bound of lblstaffName (20,240,160,30)
ADD lblstaffName in panelinfo
Initialize JtextField as lblquali
    Set Font of lblquali as ("Calibri", plain, 20)
    Set Bound of lblquali (20,290,160,30)
ADD lblquali in panelinfo

Initialize JtextField as lblworkHour
    Set Font of lblworkHour as ("Calibri", plain, 20)
    Set Bound of lblworkHour (20,330,160,30)
ADD lblworkHour in panelinfo

Initialize JtextField as lbljoinDate
    Set Font of lbljoinDate as ("Calibri", plain, 20)
    Set Bound of lbljoinDate (20,330,160,30)
ADD lbljoinDate in panelinfo

Initialize JtextField as lblshift
    Set Font of lblshift as ("Calibri", plain, 20)
    Set Bound of lblshift (20,370,160,30)
ADD lblshift in panelinfo
```

```
Initialize JtextField as lblwageper
        Set Font of lblwageper as (“Calibri”, plain, 20)
        Set Bound of lblwageper (20,420,160,30)
ADD lblwageper in panelinfo

Initialize JtextField as lblsalary
        Set Font of lblwageper as (“Calibri”, plain, 20)
        Set Bound of lblwageper (20,420,160,30)
ADD lblsalary in panelinfo

Initialize JtextField as inputvc
        Set Font of inputvc as (“Calibri”, plain, 20)
        Integer.toString(cameSalary);
        Set Bound of inputvc (20,80,160,30)
ADD inputvc in panelappoint

Initialize JtextField as inputworkHo
        Set Font of inputworkHo as (“Calibri”, plain, 20)
        Integer.toString(cameworkHo);
        Set Bound of inputworkHo (20,120,160,30)
ADD inputworkHo in panelappoint

Initialize JtextField as inputWages
        Set Font of inputWages as (“Calibri”, plain, 20)
        Integer.toString(cameWags);
        Set Bound of inputworkHo (20,160,160,30)
ADD inputWages in panelappoint

Initialize JtextField as inputjb
        Set Font of inputjb as (“Calibri”, plain, 20)
        Set Bound of inputjb (20,160,200,30)
ADD inputjb in panelappoint

Initialize JtextField as inputdesign
```

```
        Set Font of inputdesign as ("Calibri", plain, 20)
        Set Bound of inputdesign (20,240,200,30)
**ADD** inputdesign in panelappoint

Initialize JtextField as inputqualification
        Set Font of inputqualification as ("Calibri", plain, 20)
        Set Bound of inputqualification (20,290,200,30)
ADD inputqualification in panelappoint

Initialize JtextField as inputjdd
        Set Font of inputjdd as ("Calibri", plain, 20)
        Set Bound of inputjdd (20,340,200,30)
ADD inputjdd in panelappoint

Initialize JtextField as inputjapoint
        Set Font of inputjapoint as ("Calibri", plain, 20)
        Set Bound of inputjapoint (20,340,200,30)
ADD inputjapoint in panelappoint

Initialize JButton as OKbtn1(Hire)
        Set Bound of OKbtn(560,360,120,40)
        Set Font of OKbtn as ("Times New Roman", Plain, size=16)
ADD Action listener in oktn ("Adds information about vStaff in array list")
ADD OKbtn1 in panelappoint

Initialize JButton as cancel1(Clear)
        Set Bound of cancel (30,360,120,40)
        Set Font of cancel as ("Times New Roman", Plain, size=16)
ADD Action listener in cancel ("Clears all text area ")
ADD cancel1 in panelappo

IF (Selection equals Full Time)
        ADD lblsalary it in panelappoint
          inputsalary.setText(salar);
        ADD lnputsalary it in panelappoint
```

```
                END

                IF (Selection equals Part Time)
                        ADD lblshift it in panelappoint
                        ADD lblwages it in panelappoint
                        ADD Inputshift it in panelappoin
                        ADD Inputwages it in panelappoin
                END
                ADD panelapointin SubFrame1
                        Pack SubFrame1
                        Set default closed operation (Exit on close)
                        Set SubFrame1 visible as (true)



        END DO
}

END
```

## XVII. Terminate GUI

This is the method for terminating a staff and it consists of GUI containing 2 panels, label, text area and a button. The pseudocode of this method is given below.

```
Function
Public void HireFullTime()
        DO
                Initialize Jframe as TerminateFrame
                        Set Preferred Size of Subframe2 as (500,420)
                Initialize Color c as rgb (212,212,212)
                        Set Background color of TerminateFrame as (c)
                        set resizable of TerminateFrame as (False)
                        set TerminateFrame layout as (null)

                Initialize JPanel as FVpanel(null)
        Set Bounds of FVpanel as (1,1,550,120)
**Initialize** Color C as rgb(17,164,242)
        Set Background color of FVpanel as (C)

**Initialize** Jlable as label1(i)
        Set font of label1 as (“Calibri”, Bold, size=70)
        Set font color as (“Green”)
        Set Bound of label1 as (210,30,30,70
**ADD** label1 in FVpanel

**Initialize** Jlable as label2(ng)
        Set font of label2 as (“Calibri”, Bold, size=70)
        Set font color as (“WHITE”)
        Set Bound of label2 as (223,30,90,70)
**ADD** label2 in FVpanel

**Initialize** Jlable as label3(Nepal Group)
        Set font of label3 as (“MONACO”, PLAIN, size=16)
        Set font color as (“Default”)
        Set Bound of label3 as (220,85,150,40)
**ADD** label3 in FVpanel

**Initialize** MAIN JPanel as panelTerminate (null)
        Set Bounds of panelinfo as (10,140,710,375)
        Set Background color of panelinfo as (c)

**Initialize** Jlable as lblItem (“Vacancy no.”)
        Set font of lblItem as (“Calibri”, PLAIN, size=30)
        Set font color as (“Black”)
        Set Bound of lblItem as (15,110,250,40)
**ADD** lblItem in panelTerminate

**Initialize** JtextField as Terminateboxin
        Set Font of Terminateboxin as (“Calibri”, plain, 20)
        Set Bound of Terminateboxin (220,180,160,30)
```

```
            ADD Terminateboxin in panelTerminate

            Initialize JButton as Terminatebtn (Terminate)
                Set Bound of Terminatebtn (220,100,200,40)
                Set Font of Terminatebtn as (“Times New Roman”, Plain, size=16)
            ADD Action listener in Terminatebtn (“Deletes information about stafff”)
            ADD Terminatebtn in panelTerminate

            ADD panelTerminate TerminateFrame
                Pack TerminateFrame
                Set default closed operation (Exit on close)
                Set TerminateFrame visible as (true)



    END DO
}

END
```

## XVIII. DisplayMet()

This is the method which displays all the information about staff which has been hire and displays all the information about the vacancy available in console. The pseudocode of this method is given below

```
Function
Public void displayMet(){
    DO
        For( StaffHIre Temp : HireStaff){
            IF(temp instaceof Fulltimehire){
                Test = (FulltimeHire) temp
                System.out.println("Vacancy no. "+temp.getVacancyNo()+")
                 System.out.println("Designation : " + temp.getDesignation())
                 System.out.println("Job Type : "+ temp.getJobType());
                 System.out.println("Salary : "+ test.getsalary());
                 System.out.println("Work Hour : "+ test.getworkHour());
                System.out.println("Staff Name : "+ test.getstaffName());
                System.out.println("Join Date : "+ test.getjoinDate());
                System.out.println("Qualification : "+ test.getqualification());
                System.out.println("Appointed By : "+ test.getappointedBy())
                END
                 }
            END
        }

        For( StaffHIre Temp : HireStaff){
            IF(temp instaceof Parttimehire){
                PartTimeStaffHire test = (PartTimeStaffHire) temp;
                System.out.println("Vacancy no. "+temp.getVacancyNo()+);
                 System.out.println("Designation : " + temp.getDesignation());
                System.out.println("Job Type : "+temp.getJobType());
                System.out.println("Work Hour : "+test.getworkHour());
                System.out.println("Wage per Hour : "+ test.getwagePerHour());
                System.out.println("Shift : "+ test.getshifts());
                System.out.println("Staff Name : "+test.getstaffName());
                System.out.println("Join Date : "+test.getjoinDate());
                 System.out.println("Qualification : "+ test.getqualification());
                System.out.println("Appointed By : "+test.getappointedBy());

                END
           }
        END
         }
    END DO
}
END
```

## XIX. vacMemory()

It is a backend method which is created for cheking the empty text fields which are left by the user and is used mainly for creating or strong a Vacancy in an ArrayList. The pseudocode of this method is given below.

**Function**

```
private void vacMemory(){
DO
SET boolean flag as false
            If (Selection.equals("Full Time")) {
               if (VacInput1.getText().equals("") || DegisinationInput1.getText().equals("") ||
SalaryInput1.getText().equals("") || WorkHoInput1.getText().equals("")) {
                  DO
                  JOptionPane.showMessageDialog(SubFrame, "Please fill out all the text fields",
                  "Info", JOptionPane.INFORMATION_MESSAGE);
                 SET  flag=true;
                  END
               }
            END
            }
            else if(Selection.equals("Part Time")) {
               if (VacInput1.getText().equals("") || DegisinationInput1.getText().equals("") ||
WageInput1.getText().equals("") || ShiftInput1.getText().equals("")|| WorkHoInput1.getText().equals("")){
                           DO
                  JOptionPane.showMessageDialog(SubFrame, "Please fill out all the text fields",
                  "Info", JOptionPane.INFORMATION_MESSAGE);
                  SET flag=true;
                  END
               }
         END
            }
            if(flag==false)
                  DO
               try {
                  if (Selection.equals("Full Time")) {
                     FullTimeStaffHire fullObj = new FullTimeStaffHire(vacancyReturn(), designationReturn(),
Selection, salaryReturn(), workHourReturn());
                     HireStaff.add(fullObj);
                  }
```

```
                    if (Selection.equals("Part Time")) {
                       PartTimeStaffHire partObj = new PartTimeStaffHire(vacancyReturn(), designationReturn(),
Selection, workHourReturn(), wagePerHourReturn(), shiftReturn());
                       HireStaff.add(partObj);
                    }
                    END
                 } catch (NumberFormatException exe) {
                    JOptionPane.showMessageDialog(SubFrame,exe ,"Error!",
                    JOptionPane.WARNING_MESSAGE);
                 }
                    END
              }
          END
          }
END DO
```

## XX. Mainterminate()

This is the backend code which will run whenever a staff is needed to be terminated whenever the user gives the vacancy number of the staff which has to be terminated. Here in this method the checks all the records which are available in the array list and if the vacancy number matches it removes all the data of that staff. The pseudocode of this method is given below.

```
Function
public void mainTerminate(){
DO
          SET boolean recordFound = false;
          try {
          for (StaffHire ter : HireStaff){
          DO
             if (ter instanceof PartTimeStaffHire) {
                 PartTimeStaffHire pt = (PartTimeStaffHire) ter;
                 if (ter.getVacancyNo() == Terminateboxinret()){
                    if (pt.getjoined() == true) {
                             SET recordFound = true
                    pt.terminate
                        } else {
                        DO
                    JOptionPane.showMessageDialog(frame, "No Staff has been appointed in order to
terminate", "Error!", JOptionPane.INFORMATION_MESSAGE);
                  SET   clls = false;
                  }
                  END
                }
        END
            }
        }
        if (recordFound==false && clls==true){
        DO
           JOptionPane.showMessageDialog(frame,"No Valid record found for
termination","Error!",JOptionPane.WARNING_MESSAGE);
        }}catch (Exception aa) {
           JOptionPane.showMessageDialog(null,aa, "Error",JOptionPane.WARNING_MESSAGE);
        END DO
        }
        END
     }
END DO
```

## XXI. DuplicStaff()

This is the backend method which is responsible for checking duplicate Staff or vacancy number. In this method the main work is the check for any repeating vacancy number in an array list. The pseudocode of the method is given below.

```
Function
private void duplicStaff(){
DO
    SET boolean anis= false;
    try {
```

```
    DO

      for (StaffHire temp : HireStaff) {      //Iterating through the arraylist
        if (temp.getVacancyNo() == vacancyReturn())              JOptionPane.showMessageDialog(frame,
"This vacancy no. has already been used", "Error!",
    JOptionPane.INFORMATION_MESSAGE);

          SET  anis = true;
           break;
        }
        END
        }

      if(anis) { //This code will run if the user input vacancy no. is not equal to the vacancy no. present in the
arraylist and values are not being added to the arraylist for the first time


      } else {
            DO
        vacMemory();
        SubFrame.dispose();
    }} catch (Exception e) {
       if       (VacInput1.getText().equals("")       ||       DegisinationInput1.getText().equals("")       ||
SalaryInput1.getText().equals("") || WorkHoInput1.getText().equals("")) {
       vacMemory();
            END
       }
       else {
          JOptionPane.showMessageDialog(frame,e,"Error!",JOptionPane.ERROR_MESSAGE);

       }
    }
    END DO

 }
```

**END**

## XXII. AllStrorage()

This is the method which stores or which is called while hiring a fulltime or parttime staff. Mainly in this method the method checks for any matching vacancy number and if it gets matches it hires the staff and stores all the information on an array list else an error message is shown to user. The pseudocode of the method is given below.

**Function**

```
public void AllStorage(){
DO
    if(Selection.equals("Full Time")){
        DO
      for(StaffHire obj:HireStaff){
        if(obj instanceof FullTimeStaffHire){
        DO
            FullTimeStaffHire h = (FullTimeStaffHire) obj;
            if(h.getVacancyNo()==cameVacancy) {
            if(h.getjoined()==false){
           DO
        h.fullhire(staffNameReturn(),joinDateReturn(),qualificationReturn(),appointedByReturn())
              JOptionPane.showMessageDialog(SubFrame1,"Staff has been Hired!");
            }
        END
           }
        END
         }
        END DO
      }
    }
    if (Selection=="Part Time"){
      for(StaffHire obj:HireStaff){
        DO
          if(obj instanceof PartTimeStaffHire){
            PartTimeStaffHire h = (PartTimeStaffHire) obj;
            if(h.getVacancyNo()==cameVacancy) {
            if(h.getjoined()==false){
                h.partTimehire(staffNameReturn(),joinDateReturn(),qualificationReturn(),appointedByRet
        urn())
               JOptionPane.showMessageDialog(SubFrame1,"Staff has been Hired!");
            }
                END
           }

                END

          }
        }

                END

      }
        END
   }
END DO
```

## XXIII. ActionPerformed()

This is a implemented method from java package where all the action which are performed on any components by the user gets a particular task or calls any method which is need to make each component working. The pseudocode of this method is given below.

```
Function
@Override
        public void actionPerformed(ActionEvent e)throws ConcurrentModificationException {
            if (e.getSource()==VacFull){
               Selection = "Full Time";
               VacAddmethod();
             END
            }
```

```
if (e.getSource()==VacPart){
    Selection = "Part Time";
    VacAddmethod();
                  END
}


if (e.getSource()==AppFull){
    Selection = "Full Time";
    AppointMet();
  END

}
if (e.getSource()==AppPart){
    Selection = "Part Time";
    AppointMet();
       END

}
if (e.getSource()==CheckCLe){
    CheckingFrame.dispose();
                END

}
if (e.getSource()==terminate){
  RemoveGui();
   END

}
if (e.getSource()==TerminateBtn){
    if (Terminateboxin.getText().equals("")){
     DO
       JOptionPane.showMessageDialog(TerminateFrame,"Please    fill    out    the    Text
Field","Info",JOptionPane.INFORMATION_MESSAGE);
          }else {
            mainTerminate();
          }
         END

        }
        if(e.getSource()==display){
          displayMet();
               END

        }
        if (e.getSource()==Okbtn){
          duplicStaff();
        }
              END

        if(e.getSource()==cancel){
          VacInput1.setText("");
         ShiftInput1.setText("");
          WorkHoInput1.setText("");
          DegisinationInput1.setText("");
          WageInput1.setText("");
          SalaryInput1.setText("");
               END

        }
        if(e.getSource()==Clear1){
          inputdappoint.setText("");
          inputquliflication.setText("");
          inputStaff.setText("");
          inputjdd.setText("");
        }
        if(e.getSource()==Save1){
          if
```

```
(inputStaff.getText().equals("")||inputquliflication.getText().equals("")||inputdappoint.getText().equals("")||inputj
dd.getText().equals("")){
                    JOptionPane.showMessageDialog(SubFrame1,"Please fill out all the text
fields.","ERROR",JOptionPane.ERROR_MESSAGE);
                END

                 }else {
                    SubFrame1.dispose();
                    AllStorage();
                END

                }
                 }

              if (e.getSource()==Fullsubmit){
                 try {
                    boolean checkIsTrue = false;
                    for (StaffHire temp : HireStaff) {       //iterates within the arraylist
                       if (temp.getVacancyNo() == Integer.parseInt(InsideVac.getText()) &&
temp.getJobType().equals("Full Time"))
                          checkIsTrue = true;
                          cameVacancy = temp.getVacancyNo();
                          cameDesignation = temp.getDesignation();
                          camsJob = temp.getJobType();
                          FullTimeStaffHire ftsh = (FullTimeStaffHire) temp;
                         cameSalary = ftsh.getsalary();
                          cameWorkho = ftsh.getworkHour();
                          JustCheck();
                       }
                    }
                    if (checkIsTrue == false)
                       System.out.println("No Record Found in Full Time");
                       JOptionPane.showMessageDialog(SubFrame2, "No vacancy found", "Message",
JOptionPane.ERROR_MESSAGE);
                    }
```

```
          }catch(Exception exe){
            if(InsideVac.getText().equals("")){
              JOptionPane.showMessageDialog(SubFrame1,"Please fill out the Text Field","Info",JOptionPane.ERROR_MESSAGE);
            }
            else{
              JOptionPane.showMessageDialog(SubFrame1,exe);
            }
          }
        }
            if ( checkIsTrue == false){   //If the user input vacancy no. is not equals to the vacancy no. in the arraylist, a message box will be shown
              JOptionPane.showMessageDialog(SubFrame2,"No vacancy found" ,"Message",JOptionPane.ERROR_MESSAGE);
            }
          }catch(Exception exe){
            if(InsideVac.getText().equals("")){
              JOptionPane.showMessageDialog(SubFrame1,"Please fill out the Text Field","Info",JOptionPane.ERROR_MESSAGE);
            }
            else{
              JOptionPane.showMessageDialog(SubFrame1,exe);
            }
          }
        }
        if(e.getSource()==CheckBtn){
          SubFrame2.dispose();
          if (Selection=="Full Time") {
            for (StaffHire temp : HireStaff) {
              if (temp instanceof FullTimeStaffHire)
                FullTimeStaffHire ft = (FullTimeStaffHire) temp;
                if (ft.getVacancyNo()==cameVacancy) {
                  if (ft.getjoined() == false) {
                    HireFinal();
                    CheckingFrame.dispose();
```

```
                              END

                          } else {
                              JOptionPane.showMessageDialog(frame, "Staff has already been appointed for this
vacancy", "Info", JOptionPane.INFORMATION_MESSAGE);
                              CheckingFrame.dispose();
                          }

                        }
              END

                      }
              END
                  }

              END
               }
               if (Selection=="Part Time") {
                  for (StaffHire temp : HireStaff) {
                      if (temp instanceof PartTimeStaffHire) {
                          PartTimeStaffHire pt = (PartTimeStaffHire) temp;
                          if (pt.getVacancyNo()==cameVacancy) {
                          if (pt.getjoined() == false) {
                              HireFinal();
                              CheckingFrame.dispose();
                                    END
                   } else {

                          JOptionPane.showMessageDialog(frame, "Staff has already been appointed for this
vacancy", "Info", JOptionPane.INFORMATION_MESSAGE);

                        }
              END
```

```
            END
                    }
            END

                  }
            END
                }
             }

            END
          }

            END

            END

        }

    END DO
```

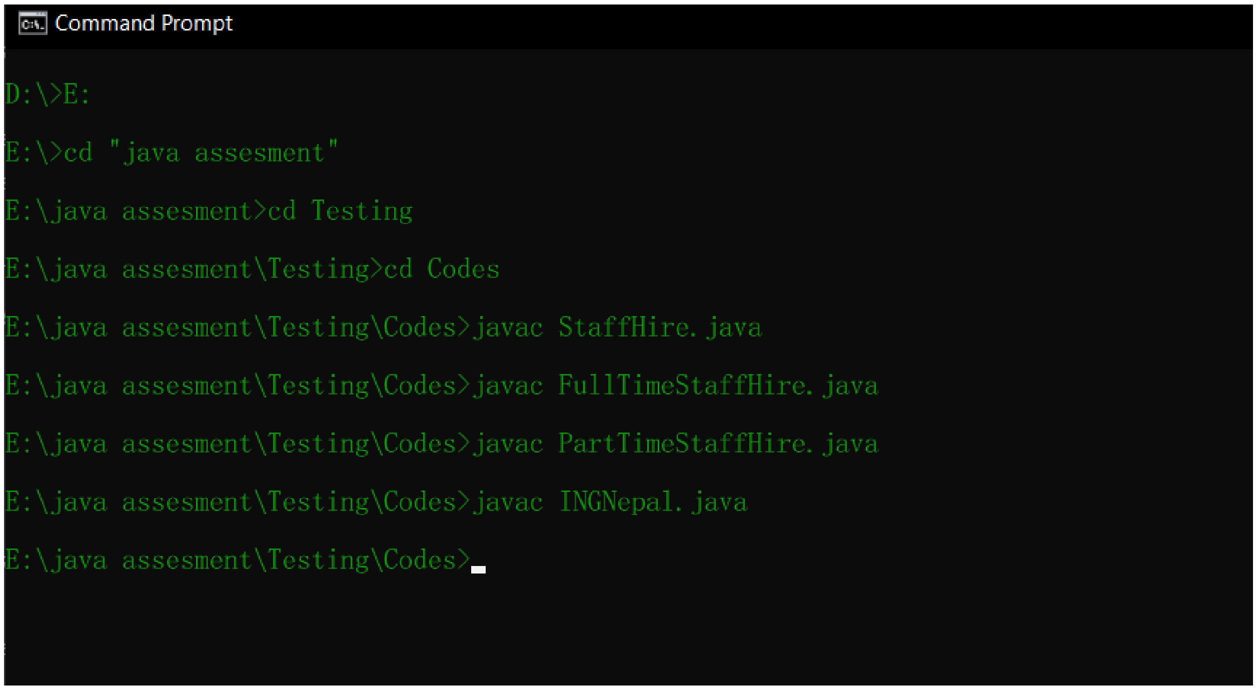
II.

## e) Testing

### I. Testing 1 (The project must get compiled using command prompt.)

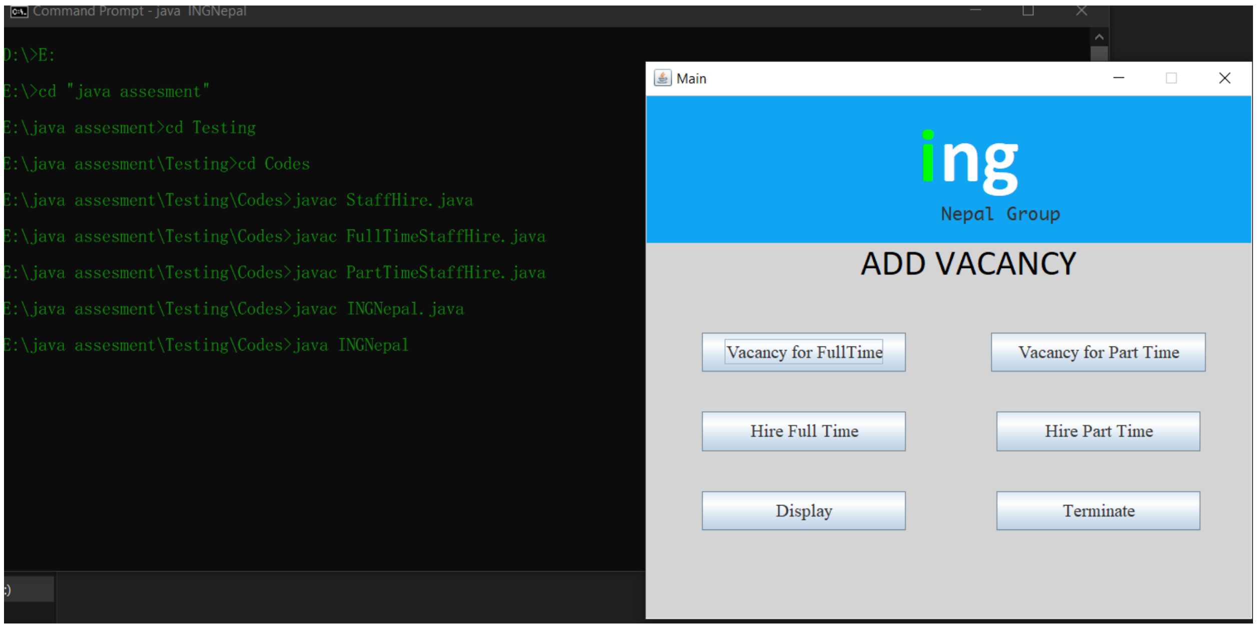
| | |
|---|---|
| Objective: | The project must get compiled using command prompt. |

| | |
|---|---|
| Action: | Firstly, the command prompt is opened from start option. After command prompt is opened the select default drive is D and is changed to E drive where my code is placed by using “E:” syntax. After changing the directory, we must go inside the folders where all my code is saved for testing and we change director by using syntax “cd Testing” and we now we are at the folder were all my codes are placed. Still we need to go inside one more file and we get into it by using same cd syntax “cd Codes” this is a file which is inside the testing files. Now we start compiling all 4 java files firstly the StaffHire file is compiled by using syntax”javac StaffHire.java”, and we again compile FullTimeStaffHire as same as we did for Staffhire, also same process for PartTimeStaffHire and INGNepal. Now after compiling all 4 classes we can now finally start the program by starting the INGNepal by using syntax “java INGNepal”. |
| Expected result: | After following all the action given above and after compiling all the java file, INGNepal must get executed using command propt. |
| Actual result: | The result of following test was as same as what was expected as INGNepal got executed in command prompt. |
| Conclusion: | The test which was carried out was a success. |



```
Command Prompt

D:\>E:

E:\>cd "java assesment"

E:\java assesment>cd Testing

E:\java assesment\Testing>cd Codes

E:\java assesment\Testing\Codes>javac StaffHire.java

E:\java assesment\Testing\Codes>javac FullTimeStaffHire.java

E:\java assesment\Testing\Codes>javac PartTimeStaffHire.java

E:\java assesment\Testing\Codes>javac INGNepal.java

E:\java assesment\Testing\Codes>
```

*Figure 3*: Testing 1 (Compiling all the java files)



```
Command Prompt - java INGNepal

D:\>E:
E:\>cd "java assesment"
E:\java assesment>cd Testing
E:\java assesment\Testing>cd Codes
E:\java assesment\Testing\Codes>javac StaffHire.java
E:\java assesment\Testing\Codes>javac FullTimeStaffHire.java
E:\java assesment\Testing\Codes>javac PartTimeStaffHire.java
E:\java assesment\Testing\Codes>javac INGNepal.java
E:\java assesment\Testing\Codes>java INGNepal
```

*Figure 4*: Starting INGNepal using command prompt

## II. Testing 2

| | |
|---|---|
| Objective: | Announcing vacancy for full time and part time ,hire staff for both full time and part time and also terminating the hired part-time staff. |
| Action: | Firstly, the INGNepal is started now we choose the option “Vacancy for full time” where we are going to announce the vacancy for full time as vacancy number 1, designation Manager, salary as 100k and working hour 10. Now the message stating “Vacancy has been added” must be seen.<br>Again, we go to the index GUI and now we choose the option “Vacancy for Part time” where we are going to announce the vacancy for Part time as vacancy number 2, designation Cook, wages as 1k and working hour 4. Now the message stating “Vacancy has been added” must be seen.<br>Now we again go back to index GUI and now we hire staff for both part time and full time here if we go in fulltime hire firstly we need to confirm vacancy number and we hire fulltime staff as name Anish Sherchan, qualification “MIT”, and joining date as 2020/04/05. Now again we perform all the same process for parttime and we put vacancy number as 2 and name as Manish Bomzon. After all the task has been performed a message stating staff has been hire must be shown for each hiring.<br>Finally, we go to terminate option and we provide a valid vacancy number of part time to terminate staff. |
| Expected result: | After following all the action given above, Vacancy must be announced, Staff must be hired and part time staff must get terminated when we provide a valid vacancy number. |
| Actual result: | The result of following test was as same as what was expected as vacancy must be announced, staff got hired and part time staff got terminated. |
| Conclusion: | The test which was carried out was a success. |

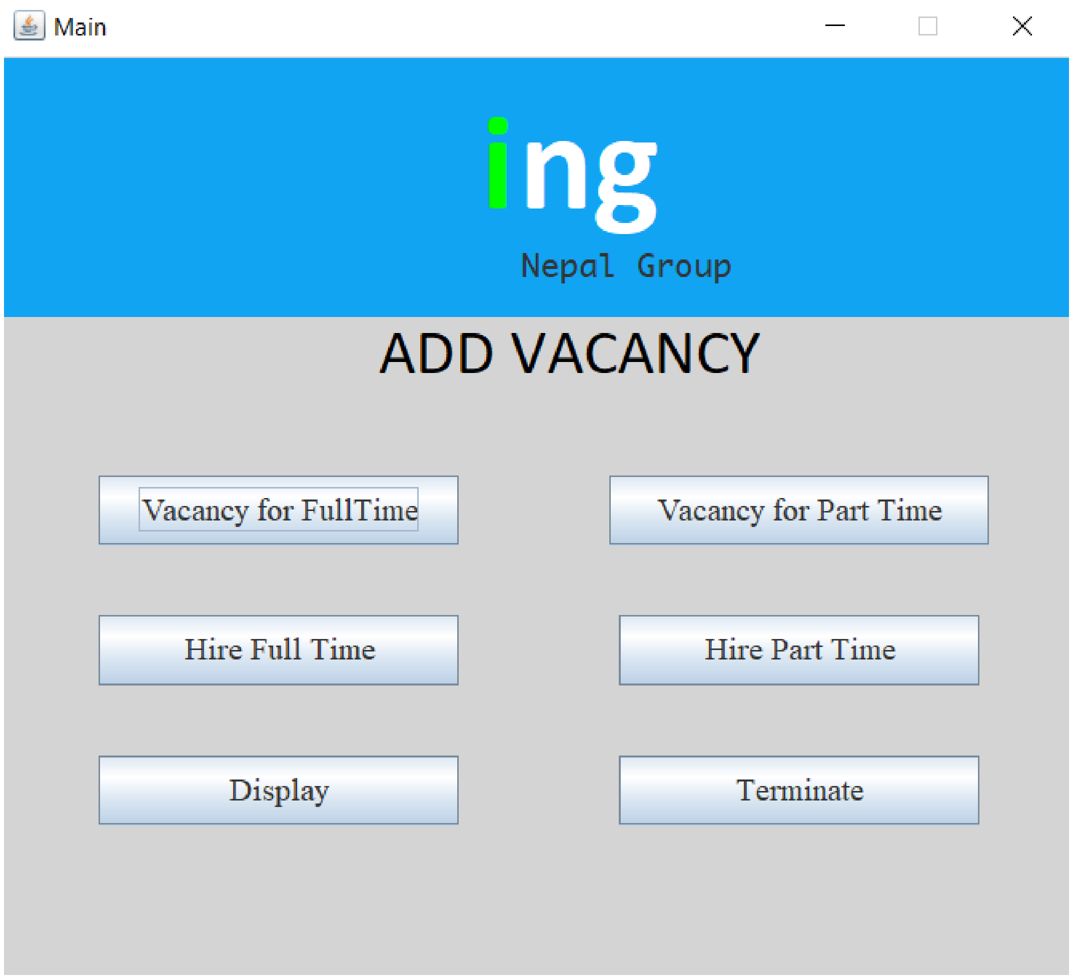
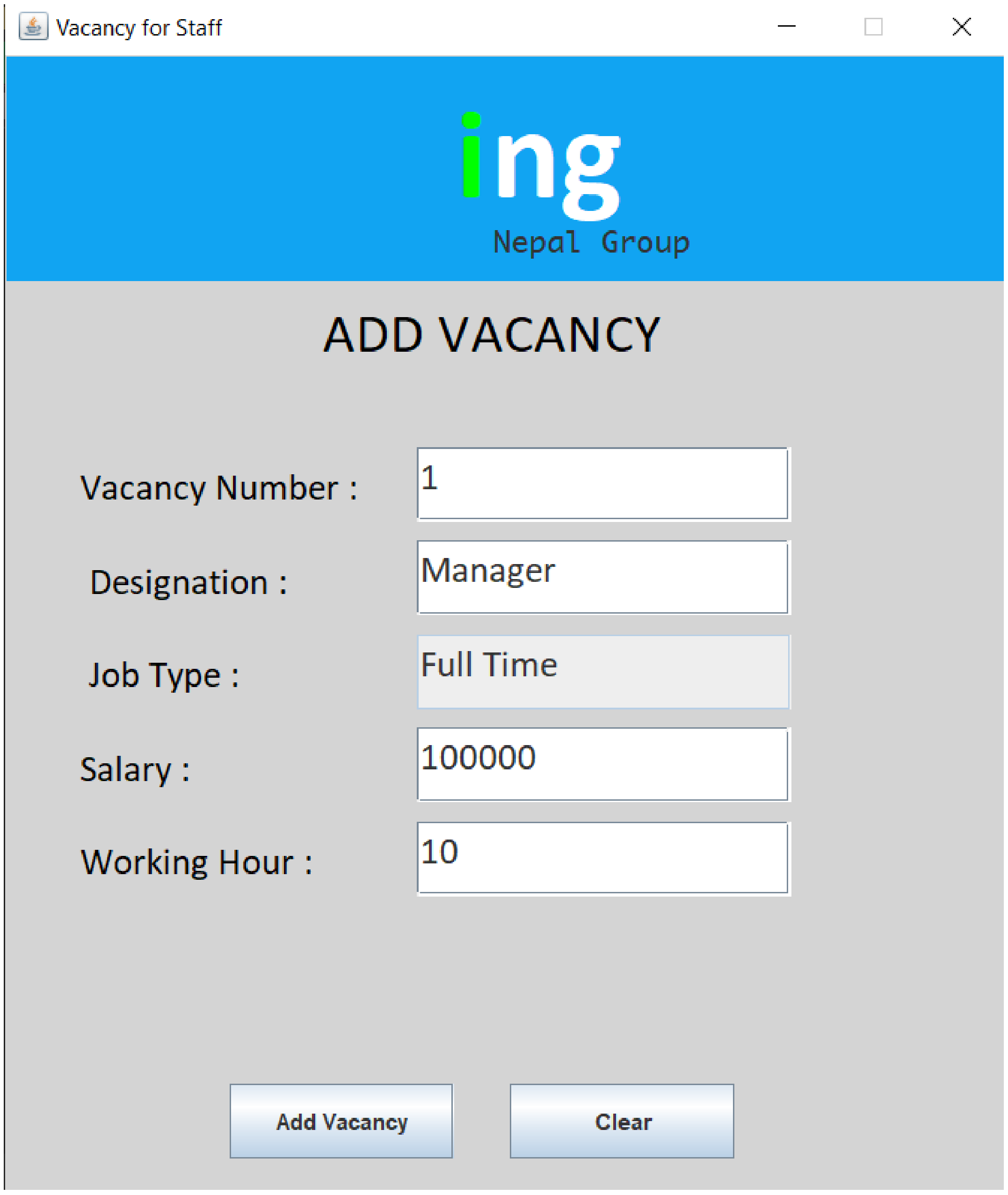


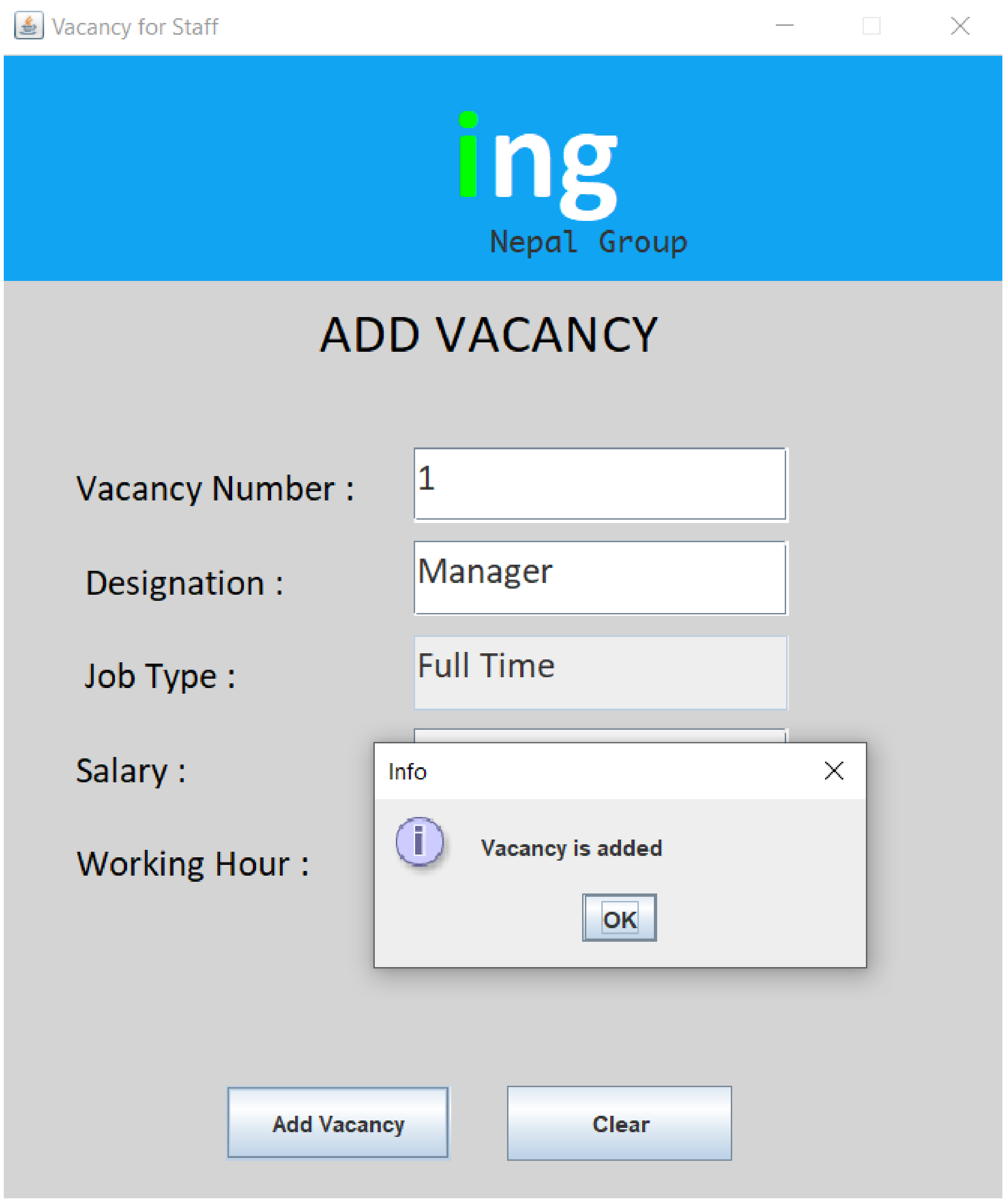
*Figure 5*: Opening Main GUI



ADD VACANCY

| | |
|---|---|
| Vacancy Number : | 1 |
| Designation : | Manager |
| Job Type : | Full Time |
| Salary : | 100000 |
| Working Hour : | 10 |

*Figure 6*: Announcing Full time Vacancy



ADD VACANCY

Vacancy Number : 1

Designation : Manager

Job Type : Full Time

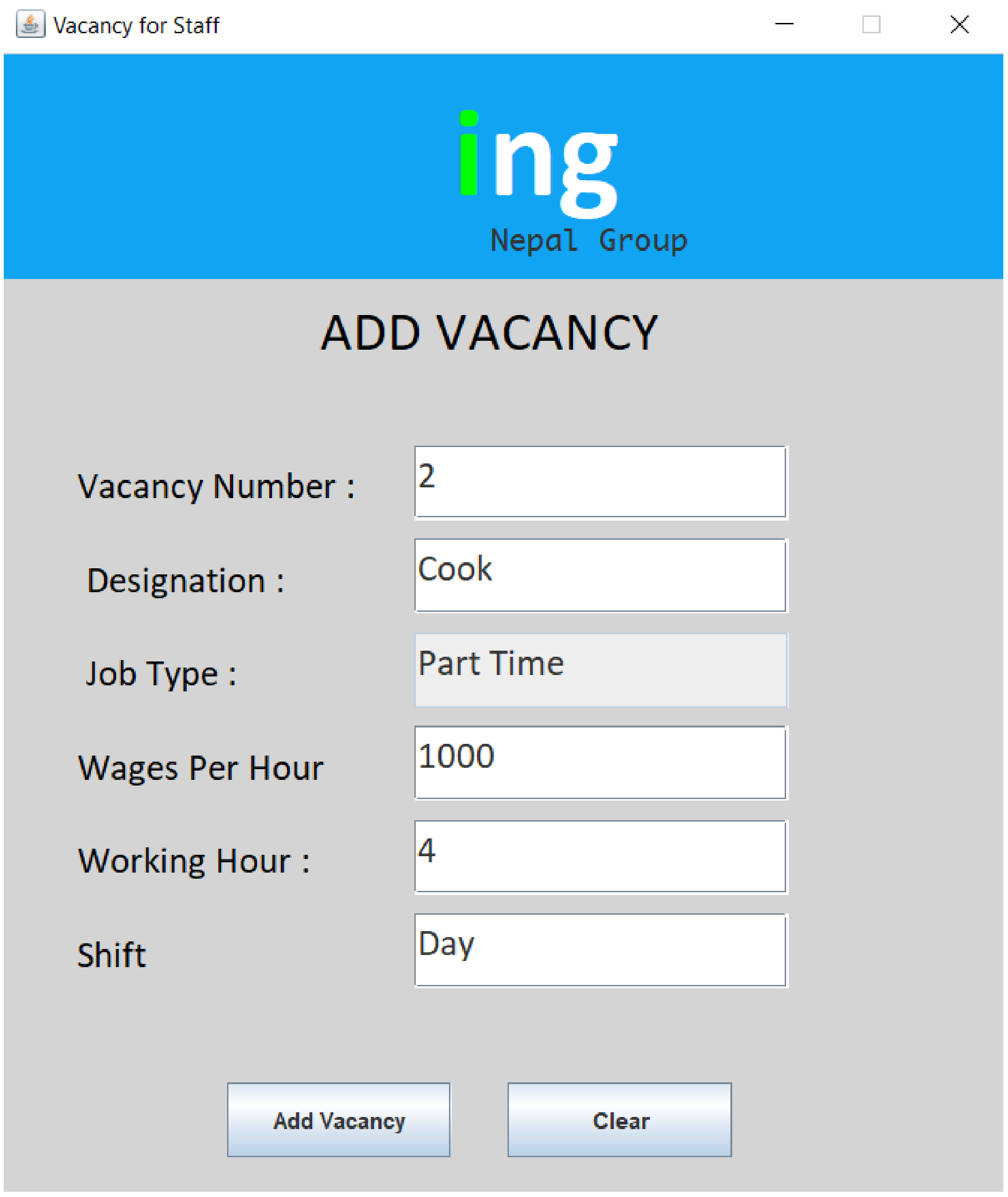
Salary :

Working Hour :

Info

Vacancy is added

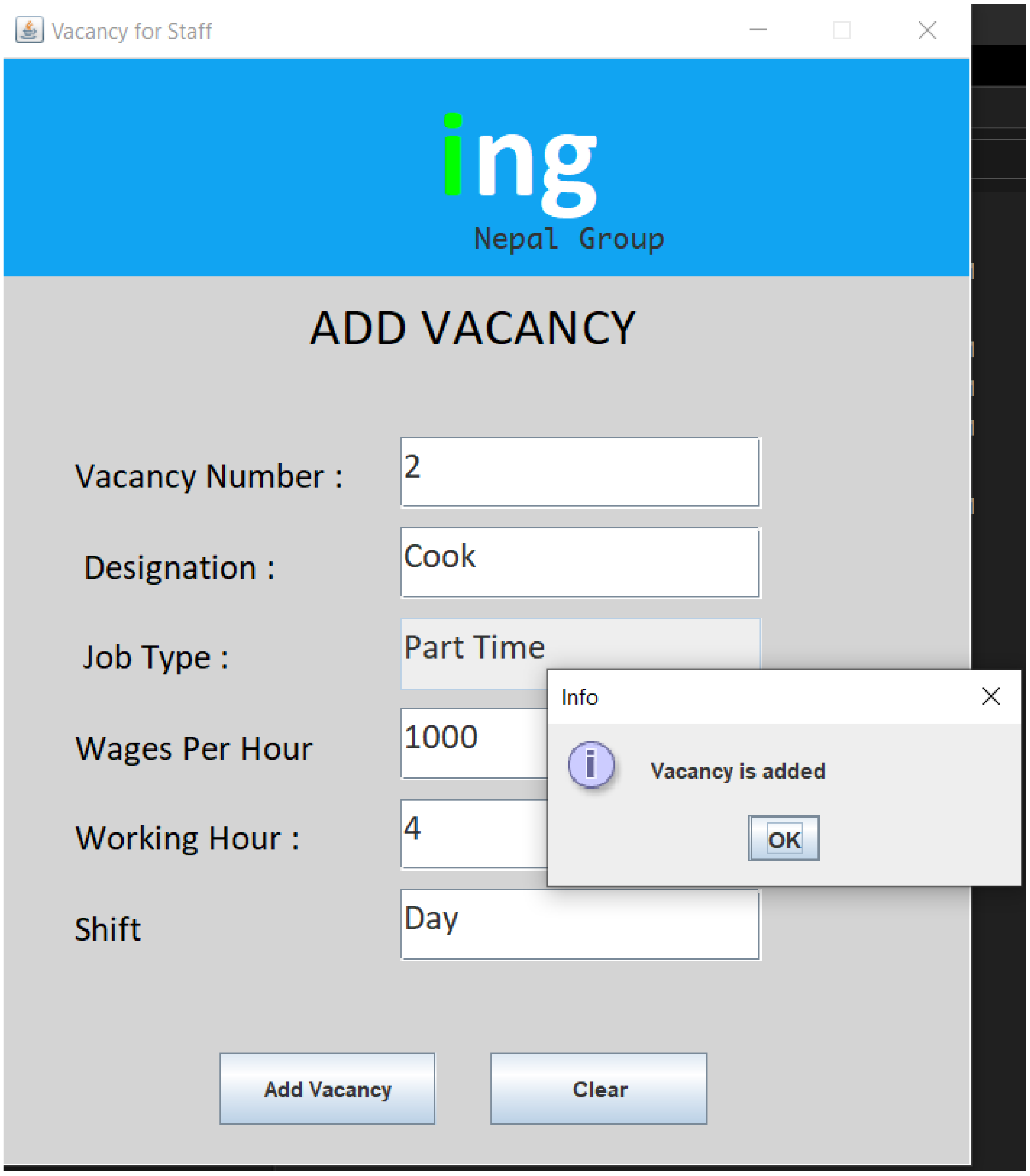
*Figure 7*: Full time Vacancy Confirmation



ADD VACANCY

| | |
|---|---|
| Vacancy Number : | 2 |
| Designation : | Cook |
| Job Type : | Part Time |
| Wages Per Hour | 1000 |
| Working Hour : | 4 |
| Shift | Day |

*Figure 8*: Part Time Vacancy

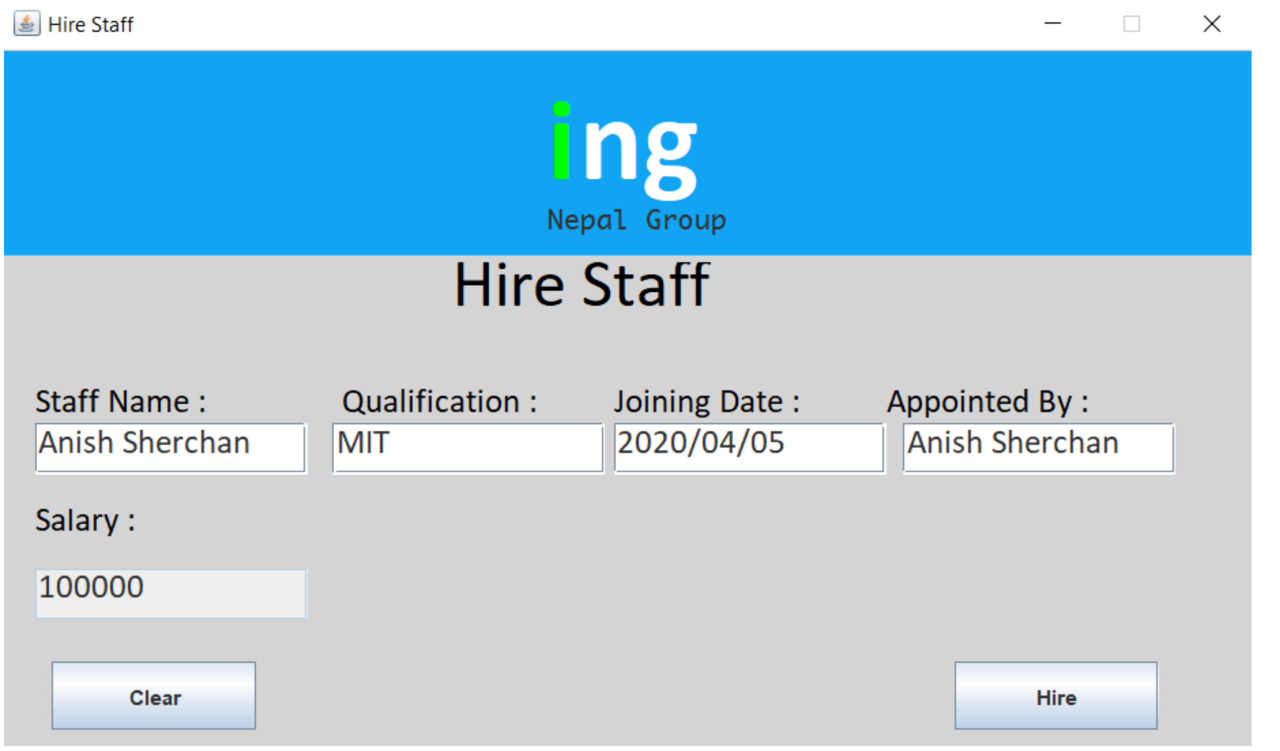


ADD VACANCY

| | |
|---|---|
| Vacancy Number : | 2 |
| Designation : | Cook |
| Job Type : | Part Time |
| Wages Per Hour | 1000 |
| Working Hour : | 4 |
| Shift | Day |

Vacancy is added

*Figure 9*: Part Time vacancy Confirmation



Hire Staff

Staff Name : Anish Sherchan

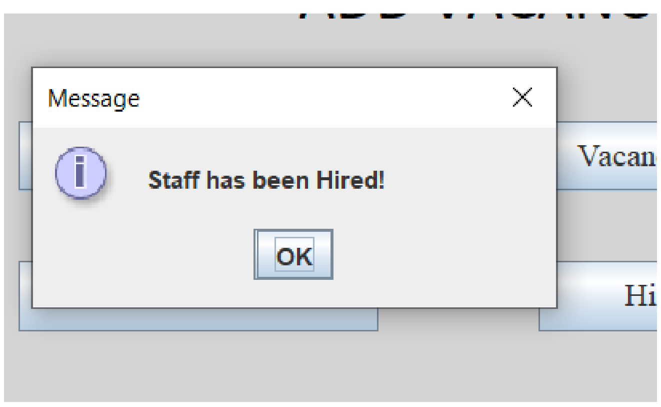
Qualification : MIT

Joining Date : 2020/04/05

Appointed By : Anish Sherchan

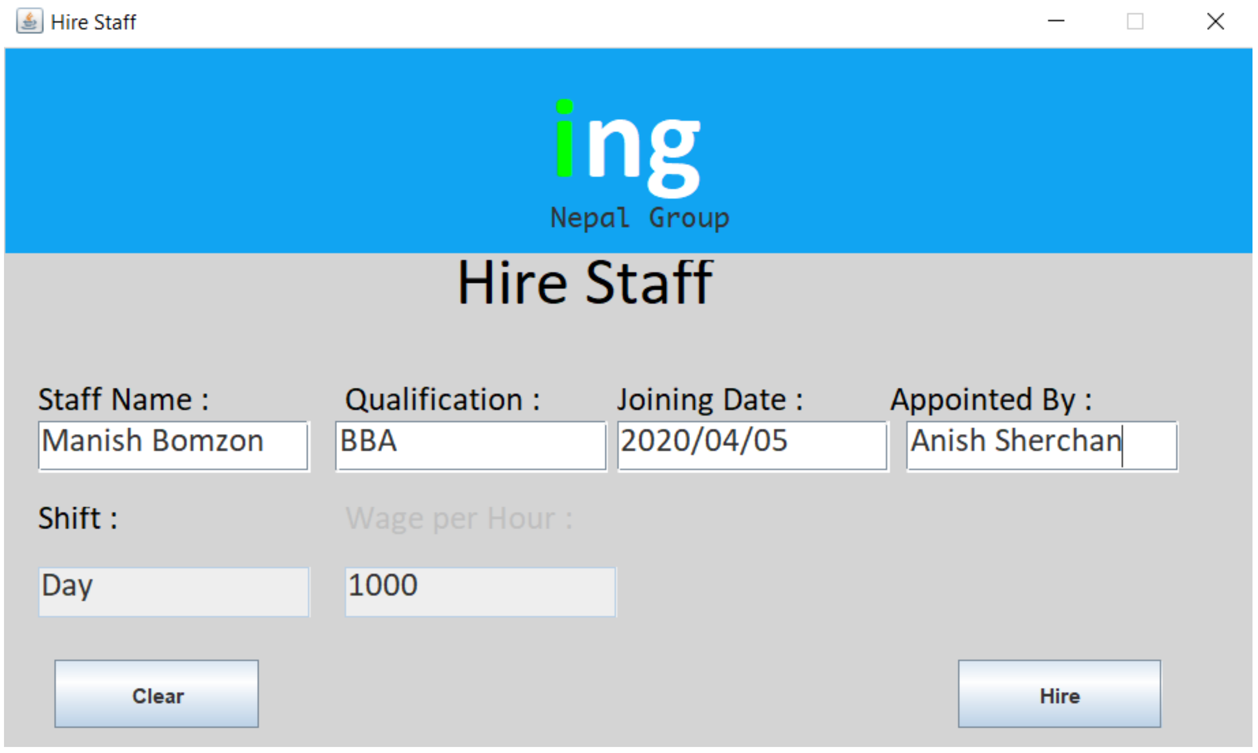
Salary : 100000

*Figure 10:* Hiring Full time staff



Staff has been Hired!

*Figure 11*: Staff Hire Confirmation



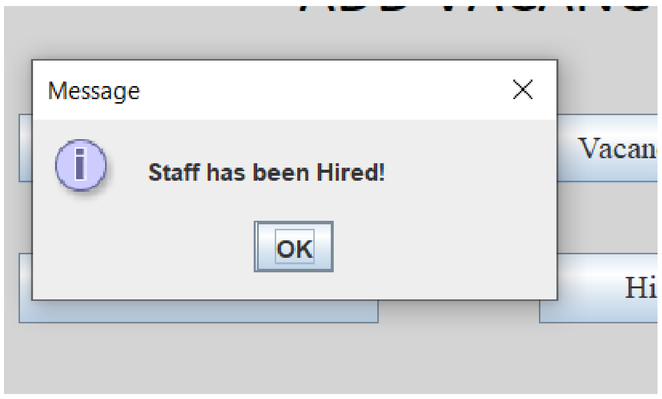
Hire Staff

ing
Nepal Group

Hire Staff

Staff Name : Manish Bomzon
Qualification : BBA
Joining Date : 2020/04/05
Appointed By : Anish Sherchan
Shift : Day
Wage per Hour : 1000

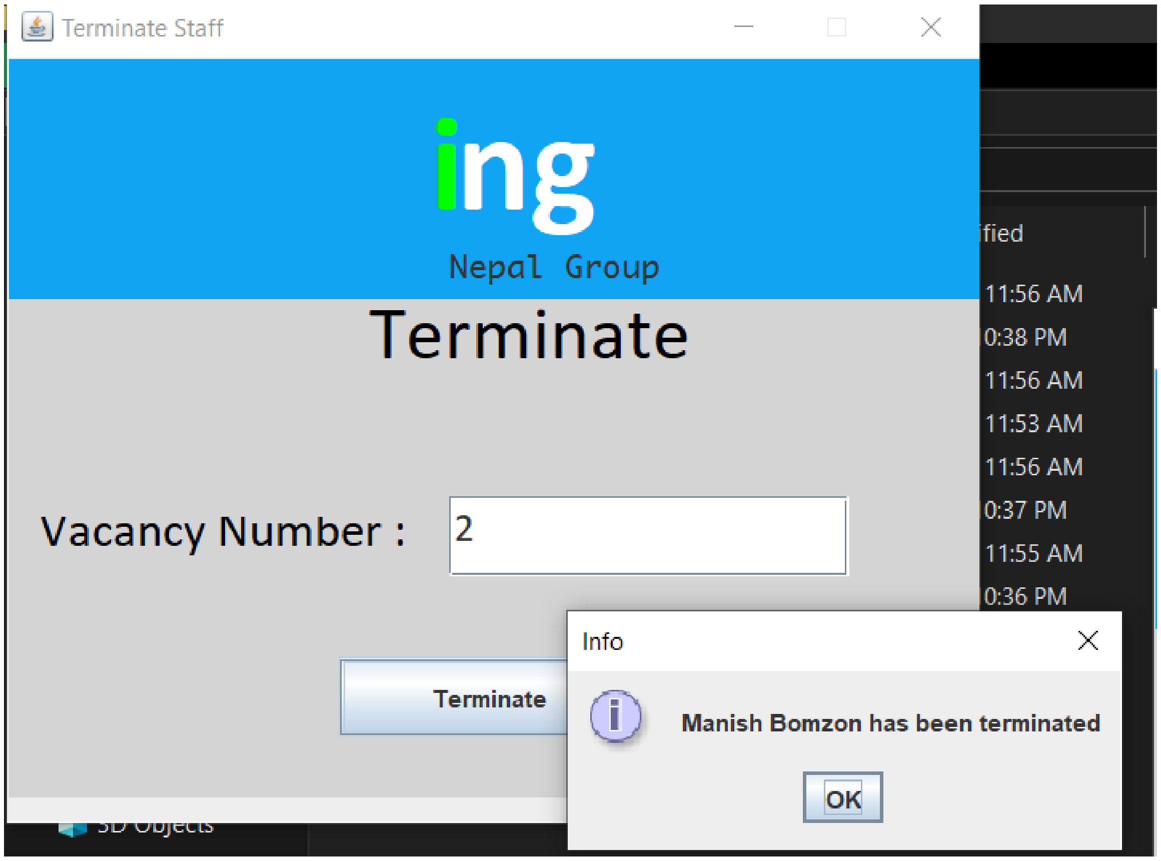
*Figure 12*: Hiring part time staff



Message

Staff has been Hired!

*Figure 13*: Confirmation Hire



Terminate

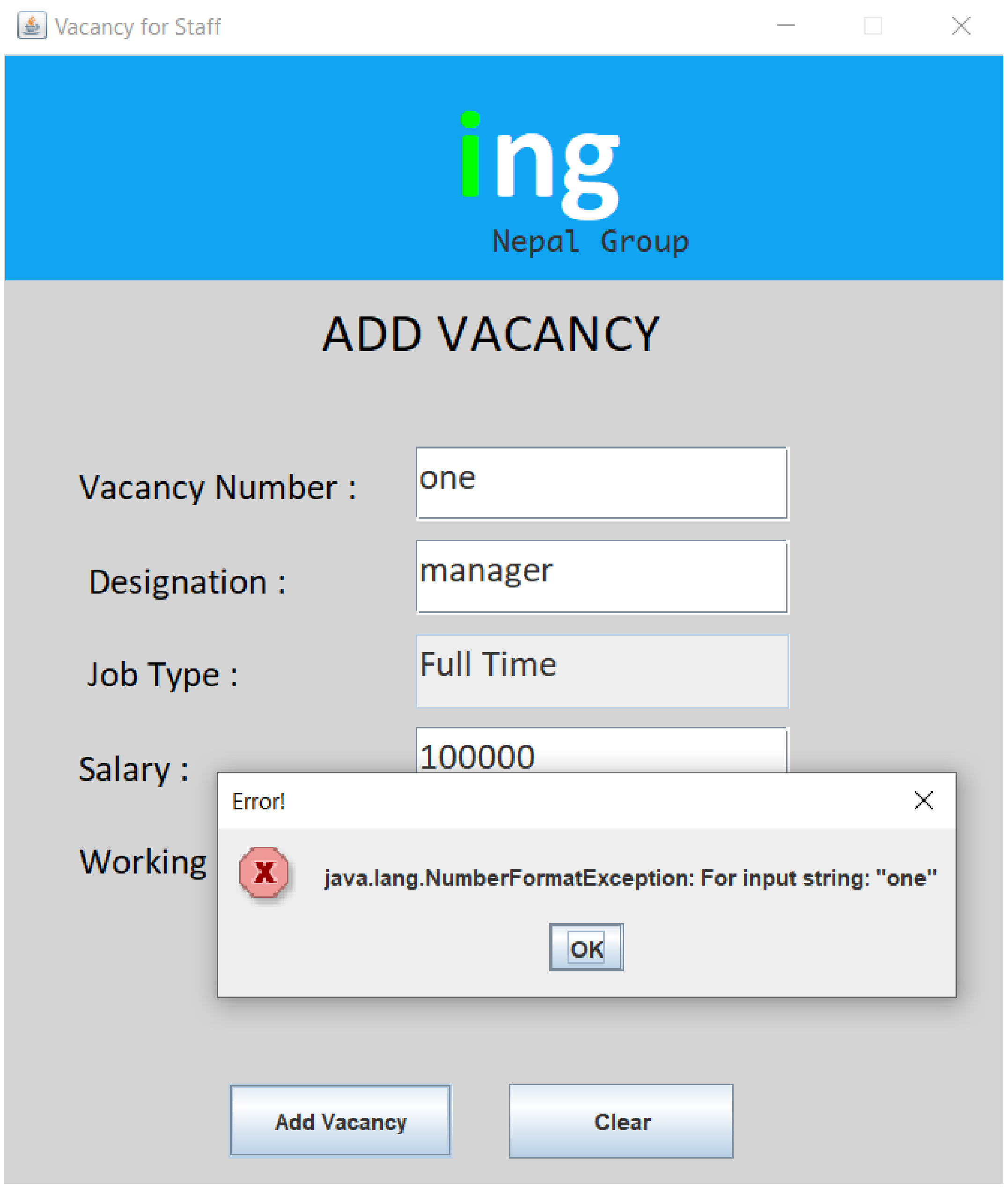
Vacancy Number : 2

Manish Bomzon has been terminated

*Figure 14*: Terminate Confirmation

## III. Testing 3

| | |
|---|---|
| Objective: | The program must throw an error message whenever we pass Letters or character instead of number in vacancy number text field. |
| Action: | Firstly, the INGNepal is started now we choose the option "Vacancy for full time" where we are going to announce the vacancy for full time as vacancy number "one", designation Manager, salary as 100k and working hour 10. Now the message stating "Vacancy has been added" must be seen. Again, we go to the index GUI and now we choose the option "Vacancy for Part time" where we are going to announce the vacancy for Part time as vacancy number "one", designation Cook, wages as 1k and working hour 4. Now the message stating "Vacancy has been added" must be seen. Finally, we go to terminate option and we provide vacancy number of part time to terminate staff again as one. |
| Expected result: | After following all the action given above, an error message must be shown Stating invalid input for INTEGER as STRING. |
| Actual result: | The result of following test was as same as what was expected, as an error message was shown. |
| Conclusion: | The test which was carried out was a success. |



ADD VACANCY

Vacancy Number : one

Designation : manager

Job Type : Full Time

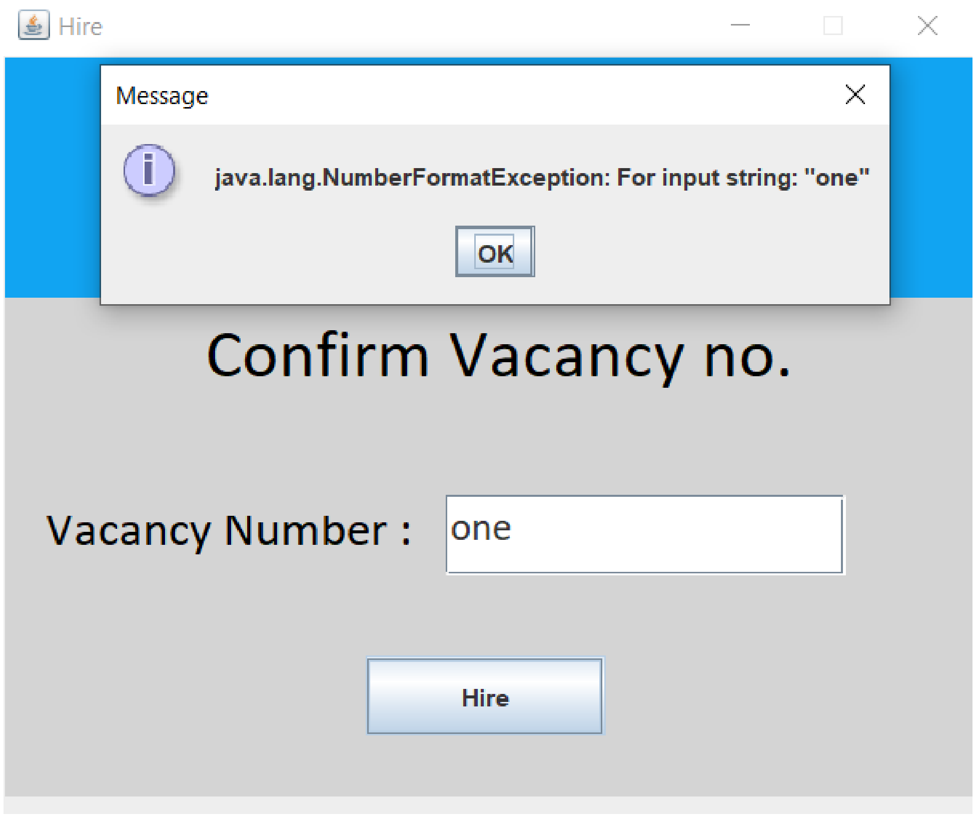
Salary : 100000

Working

Error!

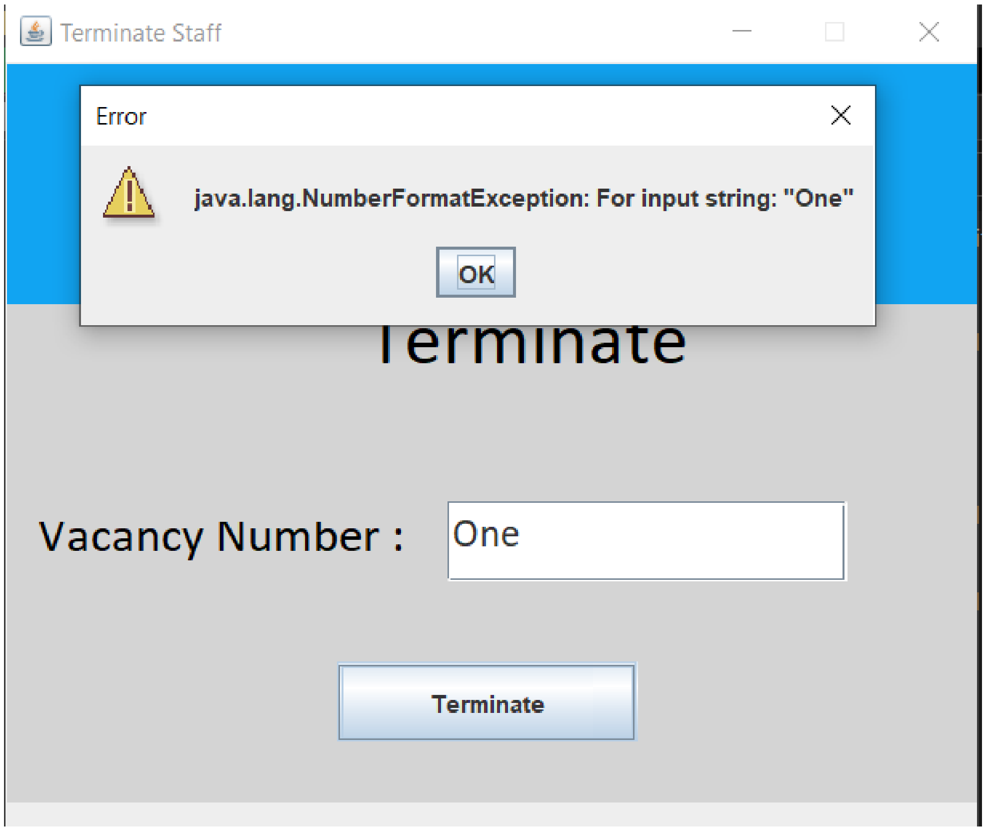
java.lang.NumberFormatException: For input string: "one"

*Figure 15*: Vacancy announcement



java.lang.NumberFormatException: For input string: "one"

*Figure 16*: Hire Staff



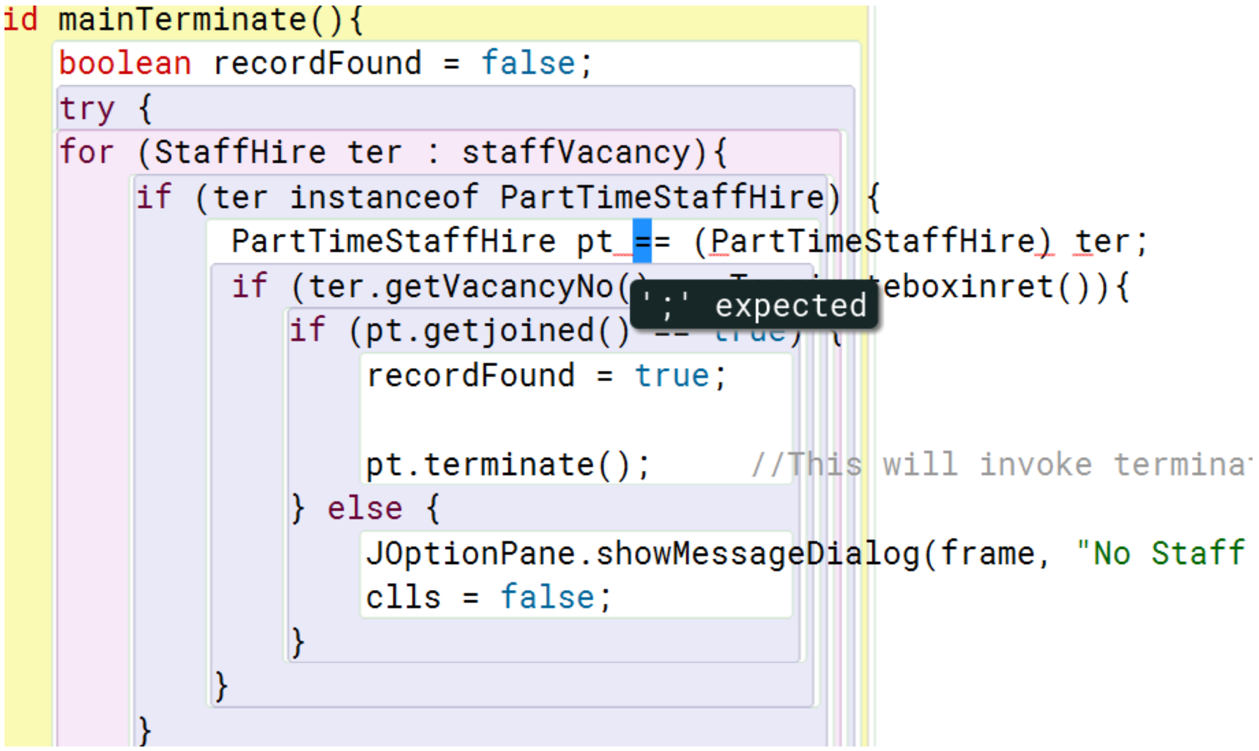
java.lang.NumberFormatException: For input string: "One"

*Figure 17*: Terminating for test 3

## f) Error detection and correction

## I. Error 1

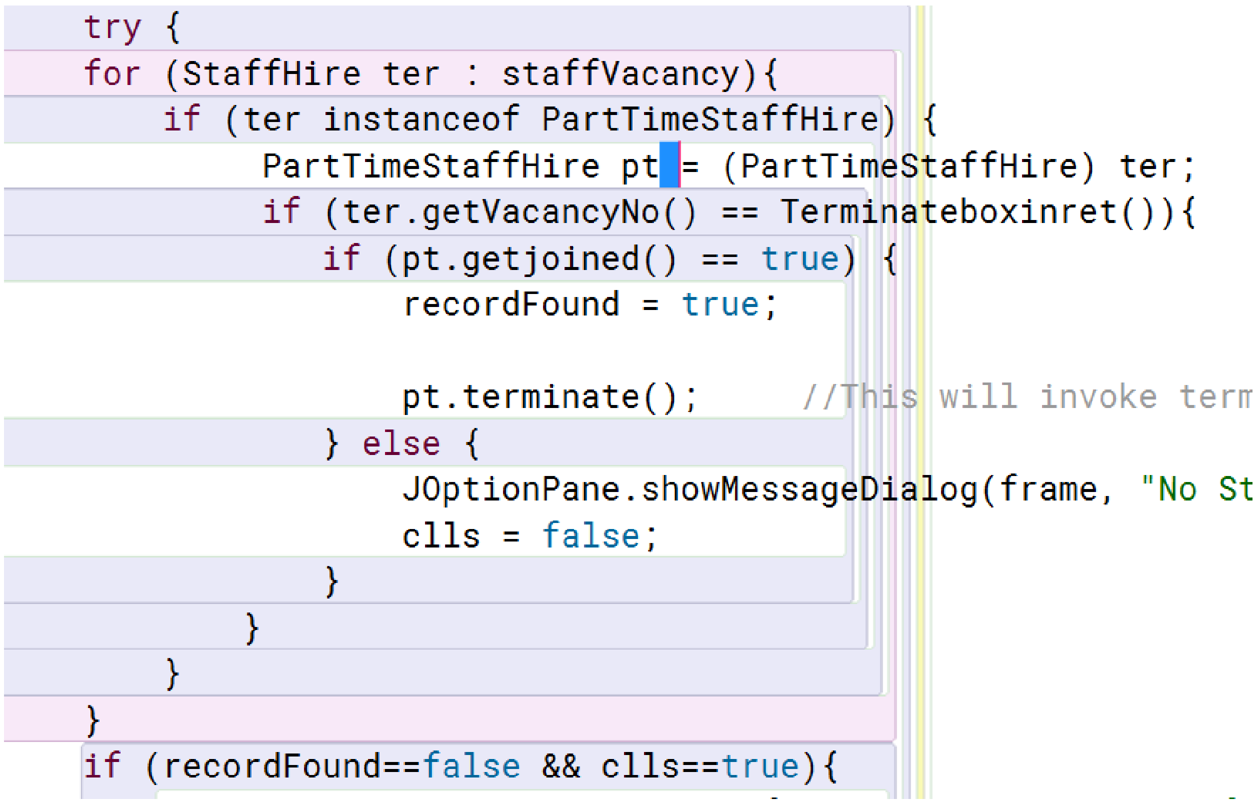
Here in first error the code “PartTimeStaffHire pt == (PartTimeStaffHire) ter ; ” has got an error due to which it is stopping the program from getting complied. The error was detected and was solved using many methods such as searching answers on books and on the internet and finally we knew that the problem was generated by “=” and was removed and complied. The photos of error detection and prevention are given below.



```
id mainTerminate(){
    boolean recordFound = false;
    try {
    for (StaffHire ter : staffVacancy){
        if (ter instanceof PartTimeStaffHire) {
            PartTimeStaffHire pt == (PartTimeStaffHire) ter;
            if (ter.getVacancyNo(          eboxinret()){
               if (pt.getjoined()
                   recordFound = true;

                   pt.terminate();      //This will invoke termina
               } else {
                   JOptionPane.showMessageDialog(frame, "No Staff
                   clls = false;
               }
            }
        }
```

';' expected

Figure 18: First Error

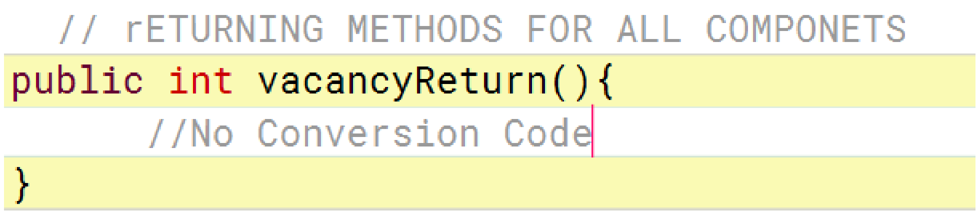


```
    try {
    for (StaffHire ter : staffVacancy){
        if (ter instanceof PartTimeStaffHire) {
             PartTimeStaffHire pt = (PartTimeStaffHire) ter;
             if (ter.getVacancyNo() == Terminateboxinret()){
                 if (pt.getjoined() == true) {
                     recordFound = true;

                     pt.terminate();      //This will invoke term
                 } else {
                     JOptionPane.showMessageDialog(frame, "No St
                     clls = false;
                 }
             }
        }
    }
    if (recordFound==false && clls==true){
```

*Figure 19*: Error correction 1

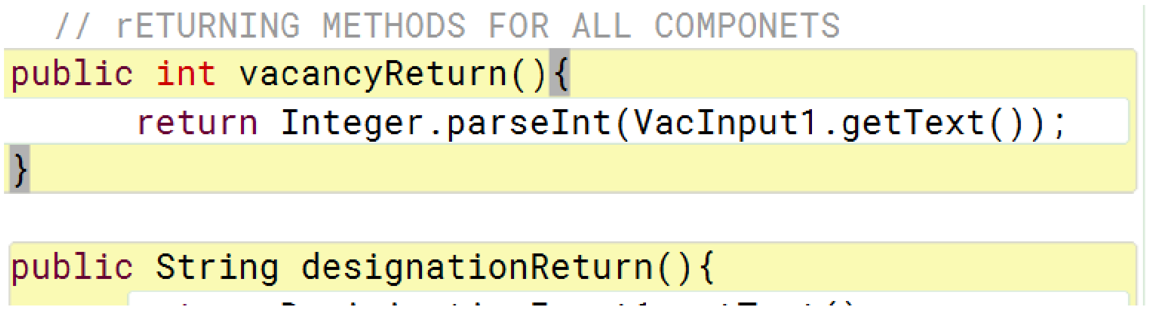
## II. Error 2

Here in second error the conversion was not applied due to which an exception was thrown. The error was detected and was solved using many methods such as searching answers on books and on the internet and finally we knew that the problem was generated by a missing code for conversion of data type and was removed and complied. The photos of error detection and prevention are given below.



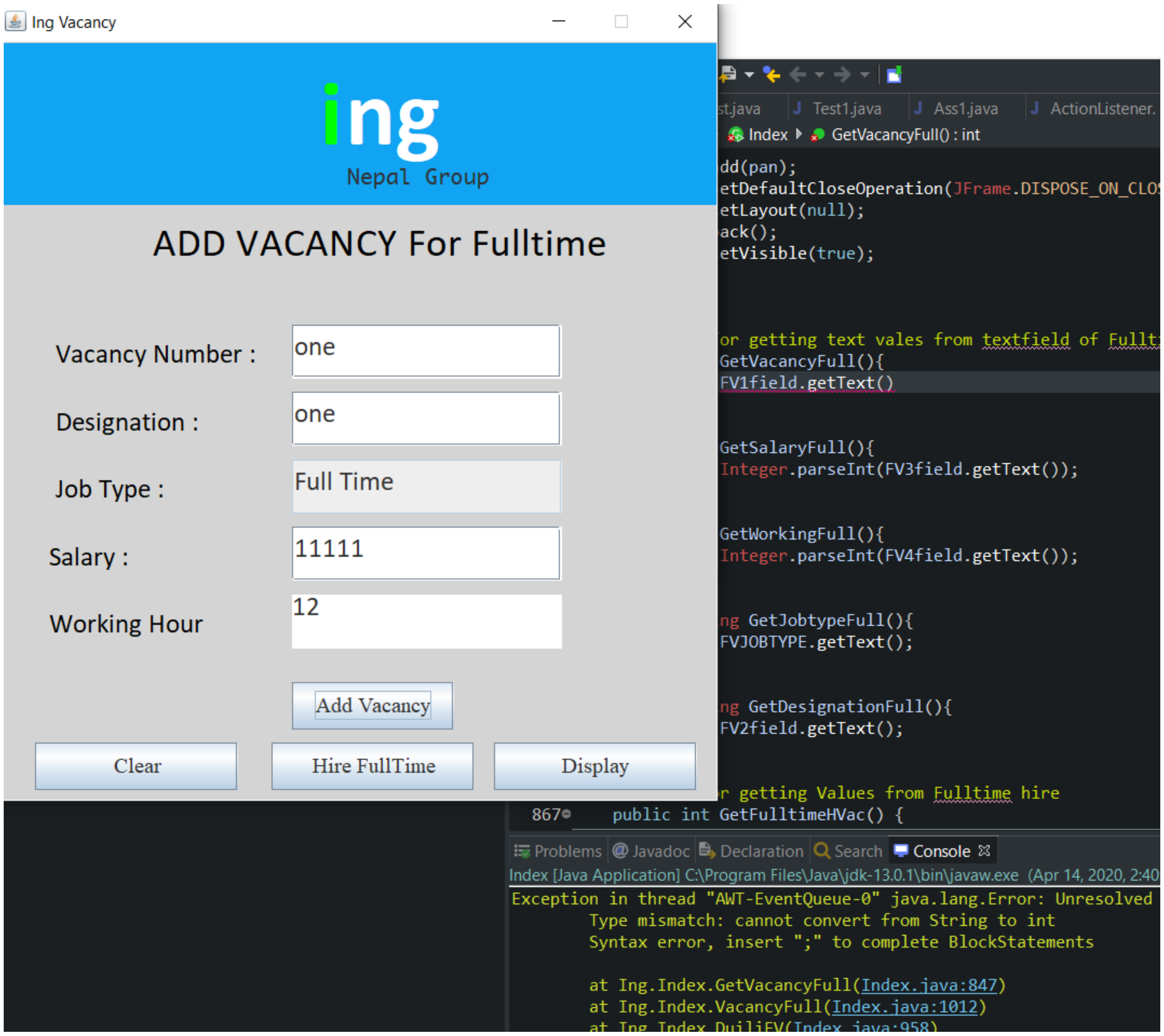
```
   // rETURNING METHODS FOR ALL COMPONETS
public int vacancyReturn(){
      //No Conversion Code
}
```

*Figure 20*: Error 2



```
  // rETURNING METHODS FOR ALL COMPONETS
public int vacancyReturn(){
      return Integer.parseInt(VacInput1.getText());
}

public String designationReturn(){
```

*Figure 21*: Error Correction



ADD VACANCY For Fulltime

| | |
|---|---|
| Vacancy Number : | one |
| Designation : | one |
| Job Type : | Full Time |
| Salary : | 11111 |
| Working Hour | 12 |

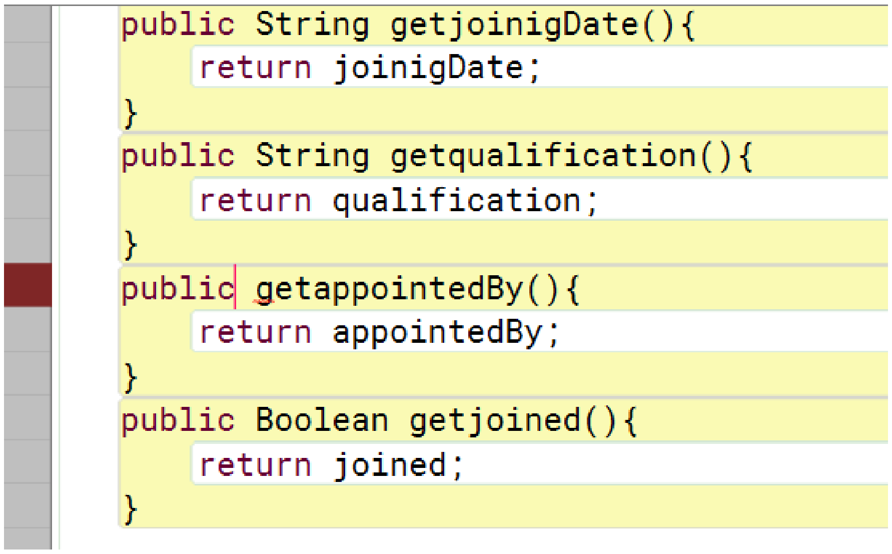
```
dd(pan);
etDefaultCloseOperation(JFrame.DISPOSE_ON_CLO
etLayout(null);
ack();
etVisible(true);

or getting text vales from textfield of Fullt
GetVacancyFull(){
FV1field.getText()

GetSalaryFull(){
Integer.parseInt(FV3field.getText());

GetWorkingFull(){
Integer.parseInt(FV4field.getText());

ng GetJobtypeFull(){
FVJOBTYPE.getText();

ng GetDesignationFull(){
FV2field.getText();

r getting Values from Fulltime hire
    public int GetFulltimeHVac() {
```

```
Index [Java Application] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe (Apr 14, 2020, 2:40
Exception in thread "AWT-EventQueue-0" java.lang.Error: Unresolved
	Type mismatch: cannot convert from String to int
	Syntax error, insert ";" to complete BlockStatements

	at Ing.Index.GetVacancyFull(Index.java:847)
	at Ing.Index.VacancyFull(Index.java:1012)
	at Ing.Index.DuiliFV(Index.java:958)
```

*Figure 22*: Evidence of error 2

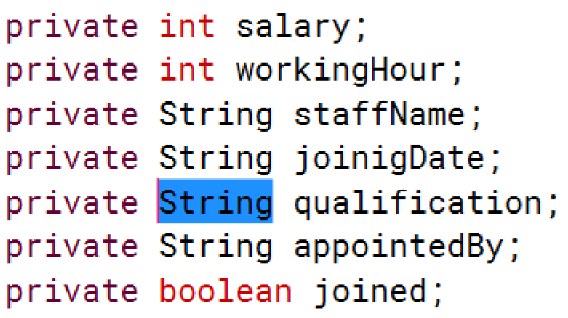
## III. Error 3

Here in error 3 we can see that there is error in code "public getappointedBy()" due to which the program is not able to get compiled the problem was detected during compiling the program and was again detected by checking the datatype in the private instance variable declaration side at the top and finally the problem was solved simply by adding the data type string to the missing place and was complied. The problem which was detected and solved with photo is given below.



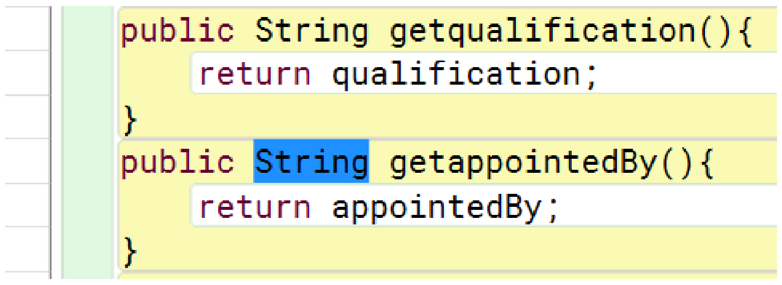
```
public String getjoinigDate(){
    return joinigDate;
}
public String getqualification(){
    return qualification;
}
public getappointedBy(){
    return appointedBy;
}
public Boolean getjoined(){
    return joined;
}
```

Figure 23: Error 3



```
private int salary;
private int workingHour;
private String staffName;
private String joinigDate;
private String qualification;
private String appointedBy;
private boolean joined;
```

*Figure 24*: Error 3 detection



```
public String getqualification(){
    return qualification;
}
public String getappointedBy(){
    return appointedBy;
}
```

*Figure 25* : Error 3 correction

## g) Conclusion

Java is a programming language and computing platform first released by Sun Microsystems in 1995. There are lots of applications and websites that will not work unless you have Java installed, and more are created every day. (JAVA, 2020) Or in other word we can say that java is an object oriented programming language and computing platform where we can deal with a programming problems in object oriented way and helps in developing a huge and complex programs in easy way possible. By the help of java this project where a user can use the program for hiring staff has been developed and was submitted to the classroom. The whole project was done in a Java platform provider called BlueJ. BlueJ is a Java integrated development environment designed for college and university students. (BlueJ, 2020)

Throughout the period for the development of the project I have learned many important aspects that were necessary for proper development of the project and it also helped me in self learning and realization process for recalling the things that I have heard and learned from the past. During the Development process I have gone through many kinds of problems and found the solution through many ways such as visiting library, accessing internets for educational use, learning much more things about Java from YouTube and all.

Also, During the development process, I came to knew that the knowledge that I have was very insufficient, so to cope with the requirements of my project development me along with my fellow mates started learning from our own mistakes and the main way of learning was group discussion and sharing of ideas and vision individual had. Another way of learning was through the help of elder brothers that were present in the college. Also throughout the learning process we can say that we had learned about GUI, CUI, Action Listener, Event Listener and many more which are required for GUI development. We also can say that we learned many more about method calling and so on. So through all my research and findings that I have done for this project finally has leaded me for the development of this project. And finally we can conclude that due to help of our teachers which are present in the college and help of the Bluej the project was finished.

## h) References

BlueJ, 2020. *BlueJ.* [Online] Available at: https://www.bluej.org/ [Accessed 09 04 2020].

ComputerHopes, 2020. *ComputerHopes.com.* [Online] Available at: https://www.computerhope.com/jargon/g/gui.htm [Accessed 09 04 2020].

JAVA, 2020. *Java.com.* [Online] Available at: https://java.com/en/download/faq/whatis_java.xml [Accessed 09 04 2020].

SkyMark, 2020. *SkyMark.* [Online] Available at: https://www.skymark.com/resources/tools/relations_diagram.asp [Accessed 10 04 2020].

tutorialspoint, 2020. *tutorialspoint.* [Online] Available at: https://www.tutorialspoint.com/uml/uml_class_diagram.htm [Accessed 09 04 2020].

## i) Appendix 1

```
    import javax.swing.BorderFactory;
    import javax.swing.BoxLayout;
    import javax.swing.ImageIcon;
    import javax.swing.JButton;
    import javax.swing.JFrame;
    import javax.swing.JLabel;
    import javax.swing.JOptionPane;
    import javax.swing.JPanel;
    import javax.swing.JScrollPane;
    import javax.swing.JTextField;
    import java.awt.BorderLayout;
    import java.awt.Color;
    import java.awt.Dimension;
    import java.awt.Font;.
    import java.awt.GridLayout;
    import java.awt.Image;
    import java.awt.Panel;
    import java.awt.event.ActionEvent;
    import java.awt.event.ActionListener;
    import java.util.ArrayList;
    import java.util.ConcurrentModificationException;
public class INGNepal implements ActionListener {
        //Declearing componets for creating GUI
        public JFrame frame;
        public JFrame SubFrame;
        public JFrame SubFrame1;
        public JFrame SubFrame2;
        public JFrame CheckingFrame;
        public JFrame TerminateFrame;
        public JFrame displayFrame;
        private String Selection;
        private String cameDesignation;
        private String camsJob;
        private String cameShift;
        private JPanel Panelmainfu;
        private JPanel Panelmain;
        private JPanel PanelAppoint;
        private JPanel PanelInfo;
        private JPanel PanelTerminate;
        private JButton VacFull;
        private JButton AppFull;
        private JButton VacPart;
        private JButton AppPart;
        private JButton display;
        private JButton terminate;
        private JButton Okbtn;
        private JButton cancel;
        private JButton Fullsubmit;
        private JButton Partsubmit;
        private JButton Clear1;
        private JButton Save1;
        private JButton CheckBtn;
        private JButton TerminateBtn;
        private JButton CheckCLe;
        private int cameVacancy;
        private int cameSalary;
        private int cameWorkho;
        private int cameWage;
        boolean phoCheck=false;
        boolean checkInside;
        boolean clls=true;
        private JTextField VacInput1,DegisinationInput1,JobInput1,SalaryInput1,WorkHoInput1,WageInput1,
         ShiftInput1,InsideVac,inputvc,inputjb,inputdesign,inputdappoint,inputshift,inputStaff,inputWorkho,
        inputjdd,inputquliflication,inputwage,inputsalary,Terminateboxin;
        Color c = new Color(212, 212, 212);
        ArrayList<StaffHire> staffVacancy = new ArrayList<>();
        //<Main method of INGNepal Class
```

```
public static void main(String[] args) {
    INGNepal obj1=new INGNepal();//Creating object of ing nepal class
    obj1.main();
}
//Main GUI or INDEX GUI
private void main() {
    frame=new JFrame("Main");//Declearing Frame
    frame.setPreferredSize(new Dimension(550,500));
    frame.getContentPane().setBackground(c);
    frame.setResizable(false);
    frame.setLayout(null);

    JPanel panel = new JPanel(null);//Declearing Panel
    panel.setBounds(1, 1, 550, 130);
    Color C = new Color(17, 164, 242);
    panel.setBackground(C);

    //text For panel
    JLabel lable1 = new JLabel("i");//Declearing lable
    lable1.setFont(new Font("Calibri", Font.BOLD, 70));
    lable1.setForeground(Color.GREEN);
    lable1.setBounds(240, 30, 30, 70);
    panel.add(lable1);

    JLabel lable2 = new JLabel("ng");//delcearing label
    lable2.setFont(new Font("Calibri", Font.BOLD, 70));
    lable2.setBounds(258, 30, 90, 70);
    lable2.setForeground(Color.WHITE);
    panel.add(lable2);

    JLabel lable3 = new JLabel("Nepal Group");//lable
    lable3.setFont(new Font("Monaco", Font.PLAIN, 16));
    lable3.setBounds(260, 85, 150, 40);
    panel.add(lable3);
    frame.add(panel);
```

```
Panelmain=new JPanel(null);//panel componets
Panelmain.setBounds(20, 130, 500, 350);
Panelmain.setBackground(c);

//Creating Jbuttons

JLabel titleLabel=new JLabel("ADD VACANCY");
titleLabel.setBounds(170,0,200,50);
titleLabel.setFont(new Font("Calibri", Font.PLAIN, 32));

titleLabel.setForeground(Color.BLACK);
VacFull = new JButton("Vacancy for FullTime");
VacFull.setBounds(30,80,180,35);
VacFull.setFont(new Font("Times New Roman",Font.PLAIN,16));
AppFull  = new JButton("Hire Full Time");
AppFull.setBounds(30,150,180,35);
AppFull.setFont(new Font("Times New Roman",Font.PLAIN,16));
VacPart = new JButton("Vacancy for Part Time");
VacPart.setBounds(285,80,190,35);
VacPart.setFont(new Font("Times New Roman",Font.PLAIN,16));
AppPart = new JButton("Hire Part Time");
AppPart.setBounds(290,150,180,35);
AppPart.setFont(new Font("Times New Roman",Font.PLAIN,16));
display = new JButton("Display");
display.setBounds(30,220,180,35);
display.setFont(new Font("Times New Roman",Font.PLAIN,16));
terminate = new JButton("Terminate");
terminate.setBounds(290,220,180,35);
terminate.setFont(new Font("Times New Roman",Font.PLAIN,16));
//ading button
Panelmain.add(titleLabel);
Panelmain.add(VacFull);
Panelmain.add(AppFull);
Panelmain.add(VacPart);
```

```
        Panelmain.add(AppPart);
        Panelmain.add(display);
        Panelmain.add(terminate);
        //actn lsn
        VacFull.addActionListener(this);
        VacPart.addActionListener(this);
        AppFull.addActionListener(this);
        AppPart.addActionListener(this);
        terminate.addActionListener(this);
        display.addActionListener(this);
        //Packing frame
        frame.add(Panelmain);
        frame.pack();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
        frame.setLocationRelativeTo(null);

    }
    //Adding vacancy for full time and part time
    private void VacAddmethod() {
        SubFrame=new JFrame("Vacancy for Staff");
        SubFrame.setLayout(null);
        SubFrame.setResizable(false);
        JLabel titleLabel=new JLabel("ADD VACANCY");
        titleLabel.setBounds(150,0,380,50);
        titleLabel.setFont(new Font("Calibri", Font.PLAIN, 30));
        titleLabel.setForeground(Color.BLACK);
        SubFrame.setPreferredSize(new Dimension(550,650));



        JPanel FVpanel = new JPanel(null);
        FVpanel.setBounds(1, 1, 550, 120);
        Color C = new Color(17, 164, 242);
        FVpanel.setBackground(C);
```

```
//text For panel
JLabel lable1 = new JLabel("i");
lable1.setFont(new Font("Calibri", Font.BOLD, 70));
lable1.setForeground(Color.GREEN);
lable1.setBounds(240, 30, 30, 70);
FVpanel.add(lable1);

JLabel lable2 = new JLabel("ng");
lable2.setFont(new Font("Calibri", Font.BOLD, 70));
lable2.setBounds(258, 30, 90, 70);
lable2.setForeground(Color.WHITE);
FVpanel.add(lable2);

JLabel lable3 = new JLabel("Nepal Group");
lable3.setFont(new Font("Monaco", Font.PLAIN, 16));
lable3.setBounds(260, 80, 150, 40);
FVpanel.add(lable3);
SubFrame.add(FVpanel);

//creaating jpanel
Panelmainfu=new JPanel(null);
Panelmainfu.setBackground(c);
Panelmainfu.setBounds(20, 130, 500, 470);

//ADDING lables
JLabel lbl1 =new JLabel("Vacancy Number : ");
lbl1.setBounds(20, 90, 160, 30);
lbl1.setFont(new Font("Calibri", Font.PLAIN,20));
lbl1.setForeground(Color.BLACK);

//TextFields
VacInput1=new JTextField(5);
VacInput1.setFont(new Font("Calibri",Font.PLAIN,20));
VacInput1.setBounds(200, 80, 200, 40);
```

```
//ADDIng lbl
JLabel lbl2 =new JLabel(" Designation : ");
lbl2.setBounds(20, 140, 160, 30);
lbl2.setFont(new Font("Calibri", Font.PLAIN,20));
lbl2.setForeground(Color.black);

//Addinfg designation textfield
DegisinationInput1=new JTextField(12);
DegisinationInput1.setFont(new Font("Calibri",Font.PLAIN,20));
DegisinationInput1.setBounds(200, 130, 200, 40);

//Addinf lable
JLabel lbl3 =new JLabel(" Job Type :");
lbl3.setBounds(20, 190, 160, 30);
lbl3.setFont(new Font("Calibri", Font.PLAIN,20));
lbl3.setForeground(Color.BLACK);

//adding TextField
JobInput1=new JTextField(15);
JobInput1.setText(Selection);
JobInput1.setFont(new Font("Calibri",Font.PLAIN,20));
JobInput1.setBounds(200, 180, 200, 40);
JobInput1.setEditable(false);

//adding label
JLabel lbl4 =new JLabel("Salary : ");
lbl4.setBounds(20, 240, 160, 30);
lbl4.setFont(new Font("Calibri", Font.PLAIN,20));
lbl4.setForeground(Color.black);

//ading text field
SalaryInput1=new JTextField(5);
SalaryInput1.setFont(new Font("Calibri",Font.PLAIN,20));
SalaryInput1.setFont(new Font("Calibri",Font.PLAIN,20));
SalaryInput1.setBounds(200, 230, 200, 40);
```

```
//adding lable
JLabel lbl5 =new JLabel("Working Hour :");
lbl5.setBounds(20, 290, 160, 30);
lbl5.setFont(new Font("Calibri", Font.PLAIN,20));
lbl5.setForeground(Color.black);

//adding gtext area
WorkHoInput1=new JTextField(4);
WorkHoInput1.setFont(new Font("Calibri",Font.PLAIN,20));
WorkHoInput1.setBounds(200, 280, 200, 40);

//adding lbl
JLabel lbl6=new JLabel("Wages Per Hour");
lbl6.setBounds(20, 240, 160, 30);
lbl6.setFont(new Font("Calibri", Font.PLAIN,20));
lbl6.setForeground(Color.black);

//add text field
WageInput1=new JTextField(10);
WageInput1.setBounds(200, 230, 200, 40);
WageInput1.setFont(new Font("Calibri",Font.PLAIN,20));

//Shift Label
JLabel lbl7=new JLabel("Shift");
lbl7.setBounds(20, 340, 160, 30);
lbl7.setFont(new Font("Calibri", Font.PLAIN,20));
lbl7.setForeground(Color.black);

//Text area for shift
ShiftInput1=new JTextField(10);
ShiftInput1.setFont(new Font("Calibri",Font.PLAIN,20));
ShiftInput1.setBounds(200, 330, 200, 40);

//Adding btn
```

```
Okbtn = new JButton("Add Vacancy");
Okbtn.setBounds(100, 420, 120, 40);
Okbtn.addActionListener(this);

//adding btn
cancel=new JButton("Clear");
cancel.setBounds(250, 420, 120, 40);
cancel.addActionListener(this);

//adding lbl in pnel
Panelmainfu.add(lbl1);
Panelmainfu.add(VacInput1);
Panelmainfu.add(lbl2);
Panelmainfu.add(DegisinationInput1);
Panelmainfu.add(lbl3);
Panelmainfu.add(JobInput1);
Panelmainfu.add(lbl5);
Panelmainfu.add(WorkHoInput1);
Panelmainfu.add(Okbtn);
Panelmainfu.add(cancel);
Panelmainfu.add(titleLabel);

SubFrame.getContentPane().setBackground(c);
SubFrame.add(Panelmainfu);
SubFrame.pack();
SubFrame.dispose();
SubFrame.setVisible(true);
//ifesdsee
if(Selection.equals("Full Time")){
   Panelmainfu.add(lbl4);
   Panelmainfu.add(SalaryInput1);
}
if(Selection.equals("Part Time")){
   Panelmainfu.add(lbl6);
   Panelmainfu.add(WageInput1);
```

```
            Panelmainfu.add(lbl7);
            Panelmainfu.add(ShiftInput1);
        }

    }
    //Hiring GUI for APPOINtment
    public void AppointMet() {
        SubFrame2=new JFrame("Hire");
        SubFrame2.setPreferredSize(new Dimension(500,420));
        SubFrame2.setLayout(null);
        SubFrame2.setResizable(false);

        //Design
        JPanel FVpanel = new JPanel(null);
        FVpanel.setBounds(1, 1, 500, 120);
        Color C = new Color(17, 164, 242);
        FVpanel.setBackground(C);

        //text For panel
        JLabel lable1 = new JLabel("i");
        lable1.setFont(new Font("Calibri", Font.BOLD, 70));
        lable1.setForeground(Color.GREEN);
        lable1.setBounds(210, 30, 30, 70);
        FVpanel.add(lable1);

        JLabel lable2 = new JLabel("ng");
        lable2.setFont(new Font("Calibri", Font.BOLD, 70));
        lable2.setBounds(223, 30, 90, 70);
        lable2.setForeground(Color.WHITE);
        FVpanel.add(lable2);

        //Text lable
        JLabel lable3 = new JLabel("Nepal Group");
        lable3.setFont(new Font("Monaco", Font.PLAIN, 16));
        lable3.setBounds(220, 85, 150, 40);
```

```
FVpanel.add(lable3);
SubFrame2.add(FVpanel);

//Addinf panel
PanelAppoint=new JPanel(null);
PanelAppoint.setBackground(c);
PanelAppoint.setBounds(1, 120, 500, 250);

//text label
JLabel titleLabel=new JLabel("Confirm Vacancy no.");
titleLabel.setBounds(100,15,330,40);
titleLabel.setFont(new Font("Calibri", Font.PLAIN, 35));
titleLabel.setForeground(Color.BLACK);

//Textarea
JLabel lbl1 =new JLabel("Vacancy Number : ");
lbl1.setBounds(20, 110, 250, 20);
lbl1.setFont(new Font("Calibri", Font.PLAIN,25));
lbl1.setForeground(Color.black);

InsideVac=new JTextField(5);
InsideVac.setFont(new Font("Calibri",Font.PLAIN,20));
InsideVac.setBounds(220, 100, 200, 40);

//ADDING Text are and label
PanelAppoint.add(titleLabel);
PanelAppoint.add(lbl1);
PanelAppoint.add(InsideVac);

//PAcking the frame
SubFrame2.add(PanelAppoint);
SubFrame2.pack();
SubFrame2.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
SubFrame2.setVisible(true);
```

```
        //if condition for jobtype
        if(Selection.equals("Full Time")){
            Fullsubmit=new JButton(" Check ");
            Fullsubmit.setBounds(180, 180, 120, 40);
            PanelAppoint.add(Fullsubmit);
            Fullsubmit.addActionListener(this);
        }

        //if condition for jobtype
        if(Selection.equals("Part Time")){
            Partsubmit=new JButton("Hire");
            Partsubmit.setBounds(180, 180, 120, 40);
            PanelAppoint.add(Partsubmit);
            Partsubmit.addActionListener(this);
        }
        //END
    }
    //Main APointe GUI
    public void HireFinal() {
        SubFrame1=new JFrame("Hire Staff");

        //Design
        JPanel panelparttime = new JPanel(null);
        panelparttime.setBounds(1, 1, 750, 120);
        Color C = new Color(17, 164, 242);
        panelparttime.setBackground(C);

        //text For panel
        JLabel lableparttime= new JLabel("i");
        lableparttime.setFont(new Font("Calibri", Font.BOLD, 70));
        lableparttime.setForeground(Color.GREEN);
        lableparttime.setBounds(320, 30, 30, 70);
        panelparttime.add(lableparttime);

        //Lable for panel
```

```
JLabel lable2fulltime = new JLabel("ng");
lable2fulltime.setFont(new Font("Calibri", Font.BOLD, 70));
lable2fulltime.setBounds(338, 30, 90, 70);
lable2fulltime.setForeground(Color.WHITE);
panelparttime.add(lable2fulltime);

//lable for panel
JLabel lable3fulltime = new JLabel("Nepal Group");
lable3fulltime.setFont(new Font("Monaco", Font.PLAIN, 16));
lable3fulltime.setBounds(320, 80, 150, 40);
panelparttime.add(lable3fulltime);
SubFrame1.add(panelparttime);

//Title for Panel
JLabel titleLabel=new JLabel("Hire Staff");
titleLabel.setBounds(265,125,250,40);
titleLabel.setFont(new Font("Calibri", Font.PLAIN, 40));
titleLabel.setForeground(Color.BLACK);
PanelInfo=new JPanel(null);
PanelInfo.setBounds(10, 140, 710, 375);
PanelInfo.setBackground(c);

//Jlable for ALl componets
JLabel lblvacancy = new JLabel("Vacancy Number:");
JLabel lbljobtype = new JLabel("Job Type:");
JLabel lblDesigna = new JLabel("Designation:");
JLabel lblstaffName = new JLabel("Staff Name :");
JLabel lblquali= new JLabel("Qualification :");
JLabel lblworkHour = new JLabel("Working Hour :");
JLabel lbljoinDate = new JLabel("Joining Date :");
JLabel lblappol = new JLabel("Appointed By :");
JLabel lblshift = new JLabel("Shift :");
JLabel lblwageper = new JLabel("Wage per Hour :");
JLabel lblsalary = new JLabel("Salary :");
```

```
//Converting String to integer
String salar = Integer.toString(cameSalary);
String wage = Integer.toString(cameWage);
String hour = Integer.toString(cameWorkho);
String vac = Integer.toString(cameVacancy);

//Text area
inputvc=new JTextField();
inputjb=new JTextField();
inputdesign=new JTextField();
inputStaff=new JTextField();
inputquliflication=new JTextField();
inputWorkho=new JTextField();
inputjdd=new JTextField();
inputdappoint=new JTextField();
inputshift=new JTextField();
inputwage=new JTextField();
inputsalary=new JTextField();

//displaying in text field
inputwage.setText(wage);
inputshift.setText(cameShift);
lblappol.setBounds(520, 200, 150, 20);
lblappol.setFont(new Font("Calibri", Font.PLAIN,20));
lblappol.setForeground(Color.BLACK);

//componets of all label and text area
lblstaffName.setBounds(20, 200, 150, 20);
lblstaffName.setFont(new Font("Calibri", Font.PLAIN,20));
lblstaffName.setForeground(Color.BLACK);

lblquali.setBounds(200, 200, 150, 20);
lblquali.setFont(new Font("Calibri", Font.PLAIN,20));
lblquali.setForeground(Color.BLACK);
```

```
lblworkHour.setBounds(520, 60, 150, 20);
lblworkHour.setFont(new Font("Calibri", Font.PLAIN,20));
lblworkHour.setForeground(Color.BLACK);

lbljoinDate.setBounds(360, 200, 150, 20);
lbljoinDate.setFont(new Font("Calibri", Font.PLAIN,20));
lbljoinDate.setForeground(Color.BLACK);

lblshift.setBounds(20, 270, 150, 20);
lblshift.setFont(new Font("Calibri", Font.PLAIN,20));
lblshift.setForeground(Color.BLACK);

lblwageper.setBounds(200, 270, 150, 20);
lblwageper.setFont(new Font("Calibri", Font.PLAIN,20));
lblwageper.setForeground(Color.BLACK);

lblsalary.setBounds(20, 270, 150, 20);
lblsalary.setFont(new Font("Calibri", Font.PLAIN,20));
lblsalary.setForeground(Color.BLACK);

//adding textarea in bound
inputdappoint.setBounds(530, 220, 160, 30);
inputdappoint.setFont(new Font("Calibri" , Font.PLAIN,20));
inputStaff.setBounds(20, 220, 160, 30);
inputStaff.setFont(new Font("Calibri" , Font.PLAIN,20));
inputquliflication.setBounds(195, 220, 160, 30);
inputquliflication.setFont(new Font("Calibri" , Font.PLAIN,20));
inputWorkho.setBounds(50000, 95, 160, 30);
inputWorkho.setFont(new Font("Calibri" , Font.PLAIN,20));
inputjdd.setBounds(360, 220, 160, 30);
inputjdd.setFont(new Font("Calibri" , Font.PLAIN,20));
inputshift.setBounds(20, 305, 160, 30);
inputshift.setFont(new Font("Calibri" , Font.PLAIN,20));
inputwage.setBounds(200, 305, 160,30);
inputwage.setFont(new Font("Calibri" , Font.PLAIN,20));
```

```
inputsalary.setBounds(20, 305, 160,30);
inputsalary.setFont(new Font("Calibri" , Font.PLAIN,20));

//Button on this gui
Save1=new JButton("Hire");
Clear1=new JButton("Clear");
Clear1.setBounds(30, 360, 120, 40);
Save1.setBounds(560, 360, 120, 40);

//adding it
PanelInfo.add(Save1);
PanelInfo.add(Clear1);
//action lsnr for buton
Clear1.addActionListener(this);
Save1.addActionListener(this);

PanelInfo.add(lblvacancy);
PanelInfo.add(lbljobtype);
PanelInfo.add(lblDesigna);
PanelInfo.add(lblappol);
PanelInfo.add(lblstaffName);
PanelInfo.add(lblquali);
PanelInfo.add(lblworkHour);
PanelInfo.add(lbljoinDate);

PanelInfo.add(inputvc);
PanelInfo.add(inputjb);
PanelInfo.add(inputdesign);
PanelInfo.add(inputdappoint);
PanelInfo.add(inputStaff);
PanelInfo.add(inputquliflication);
PanelInfo.add(inputWorkho);
PanelInfo.add(inputjdd);
PanelInfo.add(titleLabel);
```

```
        //if conditon for all
        if(Selection.equals("Full Time")){
            PanelInfo.add(lblsalary);
            inputsalary.setText(salar);
            PanelInfo.add(inputsalary);
        }
        if(Selection.equals("Part Time")){
            PanelInfo.add(lblshift);
            PanelInfo.add(lblwageper);
            PanelInfo.add(inputshift);
            PanelInfo.add(inputwage);
        }

        //setting editable
        inputvc.setEditable(false);
        inputjb.setEditable(false);
        inputdesign.setEditable(false);
        inputwage.setEditable(false);
        inputWorkho.setEditable(false);
        inputsalary.setEditable(false);
        inputshift.setEditable(false);

        //subframe
        SubFrame1.add(PanelInfo);
        SubFrame1.setPreferredSize(new Dimension(750,450));
        SubFrame1.getContentPane().setBackground(c);
        SubFrame1.pack();
        SubFrame1.setResizable(false);
        SubFrame1.dispose();
        SubFrame1.setVisible(true);
    }
    // termination gui
    public void RemoveGui(){
        TerminateFrame=new JFrame("Terminate Staff");
        TerminateFrame.setResizable(false);
```

```
JLabel titleLabel=new JLabel("Terminate");
titleLabel.setBounds(180,5,250,40);
titleLabel.setFont(new Font("Calibri", Font.PLAIN, 38));
titleLabel.setForeground(Color.black);
TerminateFrame.setPreferredSize(new Dimension(500,420));
TerminateFrame.setLayout(null);

//Design
JPanel FVpanel = new JPanel(null);
FVpanel.setBounds(1, 1, 500, 120);
Color C = new Color(17, 164, 242);
FVpanel.setBackground(C);

//text For panel
JLabel lable1 = new JLabel("i");
lable1.setFont(new Font("Calibri", Font.BOLD, 70));
lable1.setForeground(Color.GREEN);
lable1.setBounds(210, 30, 30, 70);
FVpanel.add(lable1);

JLabel lable2 = new JLabel("ng");
lable2.setFont(new Font("Calibri", Font.BOLD, 70));
lable2.setBounds(223, 30, 90, 70);
lable2.setForeground(Color.WHITE);
FVpanel.add(lable2);

JLabel lable3 = new JLabel("Nepal Group");
lable3.setFont(new Font("Monaco", Font.PLAIN, 16));
lable3.setBounds(220, 85, 150, 40);
FVpanel.add(lable3);
TerminateFrame.add(FVpanel);

//Pnale for terminate
PanelTerminate=new JPanel(null);
PanelTerminate.setBackground(c);
```

```
        PanelTerminate.setBounds(1, 120, 500, 250);

        //label for vacacny
        JLabel lblItem =new JLabel("Vacancy Number : ");
        lblItem.setBounds(15, 110, 280, 20);
        lblItem.setFont(new Font("Calibri", Font.PLAIN,25));
        lblItem.setForeground(Color.BLACK);

        //Button
        TerminateBtn=new JButton("Terminate");
        TerminateBtn.setBounds(165, 180, 150, 40);
        PanelTerminate.add(TerminateBtn);
        TerminateBtn.addActionListener(this);
        //Text field
        Terminateboxin=new JTextField();
        Terminateboxin.setBounds(220, 100, 200, 40);
        Terminateboxin.setFont(new Font("Calibri",Font.PLAIN,20));

        //adding all cmponets
        TerminateFrame.add(PanelTerminate);
        PanelTerminate.add(titleLabel);
        PanelTerminate.add(lblItem);
        PanelTerminate.add(Terminateboxin);

        //packng frame
        TerminateFrame.pack();
        TerminateFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        TerminateFrame.setVisible(true);
    }
    // display
    public void displayMet(){
        System.out.println("-------------------------------------------------------------Records----------------------------
-----------------------------------");
        for( StaffHire temp : staffVacancy){
            if(temp instanceof FullTimeStaffHire){
                FullTimeStaffHire test = (FullTimeStaffHire) temp;
                System.out.println("--------------------Vacancy no. "+temp.getVacancyNo()+"---------------------");
                System.out.println("Designation : " + temp.getDesignation());
                System.out.println("Job Type : "+ temp.getJobType());
                System.out.println("Salary : "+ test.getsalary());
                System.out.println("Work Hour : "+ test.getworkHour());
                System.out.println("Staff Name : "+ test.getstaffName());
                System.out.println("Join Date : "+ test.getjoinDate());
                System.out.println("Qualification : "+ test.getqualification());
                System.out.println("Appointed By : "+ test.getappointedBy());
                System.out.println("----------------------------------------------------------------------------");
            }
        }

        for( StaffHire temp : staffVacancy){
            if(temp instanceof PartTimeStaffHire){
                PartTimeStaffHire test = (PartTimeStaffHire) temp;
                System.out.println("--------------------Vacancy no. "+temp.getVacancyNo()+"---------------------");
                System.out.println("Designation : " + temp.getDesignation());
                System.out.println("Job Type : "+temp.getJobType());
                System.out.println("Work Hour : "+test.getworkHour());
                System.out.println("Wage per Hour : "+ test.getwagePerHour());
                System.out.println("Shift : "+ test.getshifts());
                System.out.println("Staff Name : "+test.getstaffName());
                System.out.println("Join Date : "+test.getjoinDate());
                System.out.println("Qualification : "+ test.getqualification());
                System.out.println("Appointed By : "+test.getappointedBy());
                System.out.println("----------------------------------------------------------------------------");
            }
        }
    }
    //It stores all the datas of VACANCY which are anounced
    private void vacMemory(){
        boolean flag = false;   //This flag was created in order to stop executing codes if any of the text
fields were empty
```

```
          if(Selection.equals("Full Time")) {
            if (VacInput1.getText().equals("") || DegisinationInput1.getText().equals("") ||
SalaryInput1.getText().equals("") || WorkHoInput1.getText().equals("")) {
              JOptionPane.showMessageDialog(SubFrame, "Please fill out all the text fields", "Info",
JOptionPane.INFORMATION_MESSAGE);
              flag=true;

            }
          }
          else if(Selection.equals("Part Time")) {
            if (VacInput1.getText().equals("") || DegisinationInput1.getText().equals("") ||
WageInput1.getText().equals("") || ShiftInput1.getText().equals("")|| WorkHoInput1.getText().equals("")){
              JOptionPane.showMessageDialog(SubFrame, "Please fill out all the text fields", "Info",
JOptionPane.INFORMATION_MESSAGE);
              flag=true;

            }
          }
          if(flag==false) {     //This code will only run if the value of flag is false.
            try {
              if (Selection.equals("Full Time")) {
                FullTimeStaffHire fullObj = new FullTimeStaffHire(vacancyReturn(), designationReturn(),
Selection, salaryReturn(), workHourReturn());
                staffVacancy.add(fullObj);
                JOptionPane.showMessageDialog(null,"Vacancy is added
","Info",JOptionPane.INFORMATION_MESSAGE);
              }
              if (Selection.equals("Part Time")) {
                PartTimeStaffHire partObj = new PartTimeStaffHire(vacancyReturn(),
designationReturn(), Selection, workHourReturn(), wagePerHourReturn(), shiftReturn());
                staffVacancy.add(partObj);
                JOptionPane.showMessageDialog(null,"Vacancy is added
","Info",JOptionPane.INFORMATION_MESSAGE);
              }
            } catch (NumberFormatException exe) {
                    JOptionPane.showMessageDialog(SubFrame,exe                    ,"Error!",
JOptionPane.WARNING_MESSAGE);
                }
            }
        }
    //actionlistner
    @Override
    public void actionPerformed(ActionEvent e)throws ConcurrentModificationException {
        if (e.getSource()==VacFull){
            Selection = "Full Time";
            VacAddmethod();

        }


        if (e.getSource()==VacPart){
            Selection = "Part Time";
            VacAddmethod();

        }


        if (e.getSource()==AppFull){
            Selection = "Full Time";
            AppointMet();
        }
        if (e.getSource()==AppPart){
            Selection = "Part Time";
            AppointMet();

        }
        if (e.getSource()==CheckCLe){
            CheckingFrame.dispose();

        }
        if (e.getSource()==terminate){
          RemoveGui();


        }
        if (e.getSource()==TerminateBtn){
            if (Terminateboxin.getText().equals("")){
                JOptionPane.showMessageDialog(TerminateFrame,"Please fill out the Text
Field","Info",JOptionPane.INFORMATION_MESSAGE);
            }else {
                mainTerminate();
            }
        }
        if(e.getSource()==display){
            displayMet();
        }
        if (e.getSource()==Okbtn){
            duplicStaff();
        }

        if(e.getSource()==cancel){
            VacInput1.setText("");
           ShiftInput1.setText("");
            WorkHoInput1.setText("");
            DegisinationInput1.setText("");
            WageInput1.setText("");
            SalaryInput1.setText("");
        }
        if(e.getSource()==Clear1){
            inputdappoint.setText("");
            inputquliflication.setText("");
            inputStaff.setText("");
            inputjdd.setText("");
        }
        if(e.getSource()==Save1){
            if
```

```
(inputStaff.getText().equals("")||inputquliflication.getText().equals("")||inputdappoint.getText().equals("")||inputj
dd.getText().equals("")){
                    JOptionPane.showMessageDialog(SubFrame1,"Please fill out all the text
fields.","ERROR",JOptionPane.ERROR_MESSAGE);
                }else {
                    SubFrame1.dispose();
                    AllStorage();
                }
                }

            if (e.getSource()==Fullsubmit){ //This code will be run when appointSubmit1 is pressed
                try {
                    boolean checkIsTrue = false;    //creating a flag which will check if the vacancy no. has already
been used in the arraylist
                    for (StaffHire temp : staffVacancy) {       //iterates within the arraylist
                        if (temp.getVacancyNo() == Integer.parseInt(InsideVac.getText()) &&
temp.getJobType().equals("Full Time")) {     //If the user input vacancy no. is equals to the vacancy no. in the
arraylist, it will open a new frame which will let to Okbtn staff to the vacancy
                            checkIsTrue = true;
                            cameVacancy = temp.getVacancyNo();
                            cameDesignation = temp.getDesignation();
                            camsJob = temp.getJobType();
                            FullTimeStaffHire ftsh = (FullTimeStaffHire) temp;
                           cameSalary = ftsh.getsalary();
                            cameWorkho = ftsh.getworkHour();
                            JustCheck();
                        }
                    }
                    if (checkIsTrue == false) {         //If the user input vacancy no. is not equals to the vacancy no.
in the arraylist, a message box will be shown
                        System.out.println("No Record Found in Full Time");
                        JOptionPane.showMessageDialog(SubFrame2, "No vacancy found", "Message",
JOptionPane.ERROR_MESSAGE);
                    }
                }catch(Exception exe){
```

```
            if(InsideVac.getText().equals("")){
                JOptionPane.showMessageDialog(SubFrame1,"Please fill out the Text Field","Info",JOptionPane.ERROR_MESSAGE);
            }
            else{
                JOptionPane.showMessageDialog(SubFrame1,exe);
            }
        }
    }
    if (e.getSource()==Partsubmit){
        try {
            boolean checkIsTrue = false;     //creating a flag which will check if the vacancy no. has already been used in the arraylist
            for(StaffHire temp: staffVacancy) {  //iterates within the arraylist
                if (temp.getVacancyNo() == Integer.parseInt(InsideVac.getText()) ) {   //If the user input vacancy no. is equals to the vacancy no. in the arraylist, it will open a new frame which will let to Okbtn staff to the vacancy
                    if (temp.getJobType()=="Part Time") {
                        checkIsTrue = true;
                        cameVacancy = temp.getVacancyNo();
                        cameDesignation = temp.getDesignation();
                        camsJob = temp.getJobType();
                        PartTimeStaffHire ptsh = (PartTimeStaffHire) temp;
                        cameWorkho = ptsh.getworkHour();
                        cameWage = ptsh.getwagePerHour();
                        cameShift = ptsh.getshifts();
                        JustCheck();
                    }
                }

            }
            if ( checkIsTrue == false){   //If the user input vacancy no. is not equals to the vacancy no. in the arraylist, a message box will be shown
                JOptionPane.showMessageDialog(SubFrame2,"No vacancy found" ,"Message",JOptionPane.ERROR_MESSAGE);
```

```
                    }
                }catch(Exception exe){
                    if(InsideVac.getText().equals("")){
                        JOptionPane.showMessageDialog(SubFrame1,"Please fill out the Text
Field","Info",JOptionPane.ERROR_MESSAGE);
                    }
                    else{
                        JOptionPane.showMessageDialog(SubFrame1,exe);
                    }
                }
            }
            if(e.getSource()==CheckBtn){
                SubFrame2.dispose();
                if (Selection=="Full Time") {
                    for (StaffHire temp : staffVacancy) {
                        if (temp instanceof FullTimeStaffHire) {  // This checks whether object is an instance of
FullTimeStaffHire subclass.
                            FullTimeStaffHire ft = (FullTimeStaffHire) temp;
                            if (ft.getVacancyNo()==cameVacancy) {
                                if (ft.getjoined() == false) {
                                    HireFinal();
                                    CheckingFrame.dispose();
                                } else {
                                    JOptionPane.showMessageDialog(frame, "Staff has already been appointed for this
vacancy", "Info", JOptionPane.INFORMATION_MESSAGE);
                                    CheckingFrame.dispose();
                                }
                            }
                        }
                    }
                }
                if (Selection=="Part Time") {
                    for (StaffHire temp : staffVacancy) {
                        if (temp instanceof PartTimeStaffHire) { // This checks whether object is an instance of
PartTimeStaffHire subclass.
```

```
                        PartTimeStaffHire pt = (PartTimeStaffHire) temp;
                        if (pt.getVacancyNo()==cameVacancy) {
                        if (pt.getjoined() == false) {
                            HireFinal();
                            CheckingFrame.dispose();
                        } else {
                          JOptionPane.showMessageDialog(frame, "Staff has already been appointed for this
vacancy", "Info", JOptionPane.INFORMATION_MESSAGE);
                        }
                        }
                      }
                    }
                  }
                }
              //
            }
            // rETURNING METHODS FOR ALL COMPONETS
          public int vacancyReturn(){
              return Integer.parseInt(VacInput1.getText());
          }

          public String designationReturn(){
              return DegisinationInput1.getText();
          }

          public int workHourReturn(){
              return Integer.parseInt(WorkHoInput1.getText());
          }

          public int salaryReturn(){
              return Integer.parseInt(SalaryInput1.getText());
          }

          public String shiftReturn(){
              return ShiftInput1.getText();
```

```
    }

    public int wagePerHourReturn(){
        return Integer.parseInt(WageInput1.getText());
    }

    public String staffNameReturn(){
        return inputStaff.getText();
    }

    public String appointedByReturn(){
        return inputdappoint.getText();
    }

    public String joinDateReturn(){


       return inputjdd.getText();
    }

    public String qualificationReturn(){
        return inputquliflication.getText();
    }

    public int Terminateboxinret(){
        return Integer.parseInt(Terminateboxin.getText());
    }
      //mAIN TERMINATE
    public void mainTerminate(){
            boolean recordFound = false;
            try {
            for (StaffHire ter : staffVacancy){
                if (ter instanceof PartTimeStaffHire) {
                    PartTimeStaffHire pt = (PartTimeStaffHire) ter;
                    if (ter.getVacancyNo() == Terminateboxinret()){
                      if (pt.getjoined() == true) {
```

```
                    recordFound = true;

                    pt.terminate();     //This will invoke terminate method in PartTime StaffHire method.
                } else {
                    JOptionPane.showMessageDialog(frame, "No Staff has been appointed in order to
terminate", "Error!", JOptionPane.INFORMATION_MESSAGE);
                    clls = false;
                }
            }
        }
    }
    if (recordFound==false && clls==true){
        JOptionPane.showMessageDialog(frame,"No Valid record found for
termination","Error!",JOptionPane.WARNING_MESSAGE);
    }}catch (Exception aa) {
        JOptionPane.showMessageDialog(null,aa, "Error",JOptionPane.WARNING_MESSAGE);
    }
}
 //CHECK FOR DUPLICATE VACANCY NUMBER
private void duplicStaff(){
    boolean anis= false;
     try {


        for (StaffHire temp : staffVacancy) {       //Iterating through the arraylist
            if (temp.getVacancyNo() == vacancyReturn()) {  //If the user input vacancy no. is equal to
the vacancy no. present in the arraylist, it will show a message and then the code will not run furthermore.
                JOptionPane.showMessageDialog(frame, "This vacancy no. has already been used",
"Error!", JOptionPane.INFORMATION_MESSAGE);

                anis = true;
                break;
            }

        }
```

```
                    if(anis) { //This code will run if the user input vacancy no. is not equal to the vacancy no. present
in the arraylist and values are not being added to the arraylist for the first time


                    } else {
                        vacMemory();
                        SubFrame.dispose();
                    }} catch (Exception e) {
                      if      (VacInput1.getText().equals("")      ||      DegisinationInput1.getText().equals("")      ||
SalaryInput1.getText().equals("") || WorkHoInput1.getText().equals("")) {
                      vacMemory();
                      }
                      else {
                          JOptionPane.showMessageDialog(frame,e,"Error!",JOptionPane.ERROR_MESSAGE);

                      }
                    }

            }
            //Method for hiring a full time staff or part time staffd
            public void AllStorage(){
                if(Selection.equals("Full Time")){
                    for(StaffHire obj:staffVacancy){
                        if(obj instanceof FullTimeStaffHire){
                            FullTimeStaffHire h = (FullTimeStaffHire) obj;
                            if(h.getVacancyNo()==cameVacancy) {
                                if(h.getjoined()==false){

h.fullhire(staffNameReturn(),joinDateReturn(),qualificationReturn(),appointedByReturn());
                                    JOptionPane.showMessageDialog(SubFrame1,"Staff has been Hired!");
                                }
                            }
                        }
                    }
                }
                if (Selection=="Part Time"){
                    for(StaffHire obj:staffVacancy){
                        if(obj instanceof PartTimeStaffHire){
                            PartTimeStaffHire h = (PartTimeStaffHire) obj;
                            if(h.getVacancyNo()==cameVacancy) {
                                if(h.getjoined()==false){

h.partTimehire(staffNameReturn(),joinDateReturn(),qualificationReturn(),appointedByReturn());
                                    JOptionPane.showMessageDialog(SubFrame1,"Staff has been Hired!");
                                }
                            }
                        }
                    }
                }
            }

            public void JustCheck(){
                for (StaffHire s : staffVacancy) {
                    if (s.getVacancyNo() == Integer.parseInt(InsideVac.getText())) {
                            CheckingFrame = new JFrame("Check");

                            //design
                            JPanel panel = new JPanel(null);
                            panel.setBounds(1, 1, 500, 130);
                            Color C = new Color(17, 164, 242);
                            panel.setBackground(C);

                            //text For panel
                            JLabel lable1 = new JLabel("i");
                            lable1.setFont(new Font("Calibri", Font.BOLD, 70));
                            lable1.setForeground(Color.GREEN);
                            lable1.setBounds(200, 30, 30, 70);
                            panel.add(lable1);
```

```
JLabel lable2 = new JLabel("ng");
lable2.setFont(new Font("Calibri", Font.BOLD, 70));
lable2.setBounds(218, 30, 90, 70);
lable2.setForeground(Color.WHITE);
panel.add(lable2);

JLabel lable3 = new JLabel("Nepal Group");
lable3.setFont(new Font("Monaco", Font.PLAIN, 16));
lable3.setBounds(216, 85, 150, 40);
panel.add(lable3);
CheckingFrame.add(panel);



JPanel panelck = new JPanel(null);
panelck.setBounds(1, 140, 500, 200);
JLabel titleLabel=new JLabel(" Please Confirm !!");
titleLabel.setBounds(100,0,300,40);
titleLabel.setFont(new Font("Calibri", Font.PLAIN, 38));
titleLabel.setForeground(Color.BLACK);
CheckingFrame.setLayout(null);
CheckingFrame.setPreferredSize(new Dimension(500,420));

panelck.setBackground(c);

CheckBtn = new JButton("Confirm");
CheckBtn.setBounds(100,130,120,40);
CheckBtn.addActionListener(this);
panelck.add(CheckBtn);

CheckCLe = new JButton("Exit");
CheckCLe.setBounds(299,130,110,40);
CheckCLe.addActionListener(this);
panelck.add(CheckCLe);
panelck.add(titleLabel);
```

```
                    CheckingFrame.getContentPane().setBackground(c);
                    CheckingFrame.add(panelck);
                    CheckingFrame.setVisible(true);


                    CheckingFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
                    CheckingFrame.pack();
                }
            }
        }
    }
```

## j) Appendix 2

### 1. Introduction

#### I. JAVA

Java is a programming language and computing platform first released by Sun Microsystems in 1995. There are lots of applications and websites that will not work unless you have Java installed, and more are created every day. Java is fast, secure, and reliable. From laptops to datacenters, game consoles to scientific supercomputers, cell phones to the Internet, Java is everywhere! (Java, n.d.)

#### II. Summary About Project

This project was developed for an organization. This program helps in staff hiring process in any organization by the help of simple program that was developed in java platform called BlueJ and also can be developed in any other text editor's which supports and has JDK kits and other program's to read the developed java program.

This project consists of getter method, setter method, display, Supper class, Derived Classes, etc. The getter method is required for taking in values from the users and to return the value. The setter method is used for setting or editing the values which has already been given to the program in form of the object.

Display is just a simple code which displays the current status of the program and values that has been entered. Supper class or parental class is a class that is the parent class of the reaming child class. Derived class or a child class is a class that is derived from parental class.

## III. BlueJ

BlueJ is a Java integrated development environment which was solely designed for educational purposes and for small-scale software development to some extent. It is an IDE or Integrated development environment for doing JAVA programming. BlueJ was started by Michael Kolling and John Rosenberg in 1999 at Monash University, Australia. Presently this software was maintained by a team at King's College London, England, where Kolling works. A JDK(Java development kit) is required to run BlueJ. By the way, a JDK is a software development environment by which applets and JAVA applications can be developed. (DominateJava, n.d.)

## 1. Class Diagram

| Class : Staff Hire | |
|---|---|
| # vacancy Number: | INT |
| # Designation: | String |
| # Job Type : | String |
| + get Vacancy Number (): | INT |
| + get Designation (): | String |
| + get Job Type (): | String |
| + set Vacancy Number (INT vacancy Number): | Void |
| + set Designation (String designation): | Void |
| + set Job Type (String job Type): | Void |
| + display (): | Void |

Table 1: Class Diagram Staff Hire

## II. Full time staff Hire (Child Class)

| Class : Full Time Staff Hire | |
|---|---|
| - salary: | INT |
| - working Hour: | INT |
| - staff Name: | String |
| - joining Date: | String |
| - qualification: | String |
| - appointed by: | String |
| - joined: | Boolean |
| + get Salary (): | INT |
| + get Working Hour (): | INT |
| + get Staff Name (): | String |
| + get Joining Date (): | String |
| + get Qualification (): | String |
| + get Appointed By(): | String |
| + get Joined(): | Boolean |
| + set New Salary (int new Salary) : | Void |
| + setNewWorkingHour (int new WorkingHour) : | Void |
| + hire Full Time Staff (String staff Name, joining date, qualification, appointed by): | Void |
| + display (): | Void |

Table 2: Full time staff Hire

### III. Part Time Staff Hire (Child class)

| Class : Part Time Staff Hire | |
|---|---|
| - working hour: | INT |
| -wages per hour: | INT |
| -Staff Name: | String |
| -Joining Date: | String |
| -Qualification: | String |
| -Appointed by: | String |
| -Shifts: | String |
| -Joined: | Boolean |
| -Terminated: | Boolean |
| | INT |
| + get Working Hour (): | INT |
| + get Wages Per Hour (): | String |
| + get shifts (): | String |
| + get Staff Name (): | String |
| + get Joining Date (): | String |
| + get Qualification (): | String |
| + get Appointed by (): | Boolean |
| + get Joined (): | Boolean |
| + get Terminated (): | Void |
| + set Shifts (String new Shifts) : | Void |
| + set Working Hour(int new Working Hour): | Void |
| + setWagesPerHour (int new Wages Per Hour): | Void |
| +hire Part Time Staff (String staffName,String joiningDate,String qualification,String appointedBy): | Void |

| | |
|---|---|
| + terminate Staff ():<br>+ display(): | Void<br>Void |

Table 3: Class Part time Staff Hire

## 2. Pseudocode for the method of all used classes and short describtion.

### I. Staff Hire (Supper Class)

Staff Hire (Supper Class) helps user to take in information such as vacancy number, job type, designation, helps in creating object. And Finally helps to make a final output as report and displays it to user in informative way as possible.

#### a) Get Method

#### i. getVacancyNumber()

The get method on java is a method that returns the value of the private property or any private variable. Or in simply way we can say that " The Get method returns the variable value". In this class staffhire the method getVacancyNumber() uses the data type INT and is returned by the help of get method. And Pseudocode is given below.

```
Function

public int getVacancyNumber(){

   return vacancyNo;

END

}
```

#### ii. getDesignation()

The get method on java is a method that returns the value of the private property or any private variable. Or in simply way we can say that " The Get method returns the variable value". In this class staffhire the method getDesignation() uses the data type

String and is returned by the help of get method. And the Pseudocode for this method is given below.

```
Function

public String getDesignation(){
   return designation;
END

 }
```

iii. getJobType()

The get method on java is a method that returns the value of the private property or any private variable. Or in simply way we can say that " The Get method returns the variable value". In this class staffhire the method getJobType() uses the data type String and is returned by the help of get method. And Pseudocode for this method is given below.

```
Function

public String getJobType(){
   return jobType;
END

}
```

b) Void Set Method

i. setVacancyNumber(int vacancyNumber)

The set method on java is a method that sets the value of the private property or private variable and does not returns the value as it is void type. Or in simply way we can say that the set method sets the value of private porperty and does not returns it. In this class StaffHire the method setVacancyNumber(int vacancyNumber) uses data type

Integer to set the value of the private variable. The pseudocode for this method is given below.

```
Function
DO

public void setVacancyNumber(int vacancyNo){
   this.vacancyNo=vacancyNo;
          END DO
END
 }
```

ii. setDesignation(String designation)

The set method on java is a method that sets the value of the private property or private variable and does not returns the value as it is void type. Or in simply way we can say that the set method sets the value of private porperty and does not returns it. In this class StaffHire the method setDesignation(String designation) uses data type String to set the value of the private variable. The pseudocode for this method is given below.

```
Function
DO

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

iii. setJobType(String jobType)

The set method on java is a method that sets the value of the private property or private variable and does not returns the value as it is void type. Or in simply way we can say that the set method sets the value of private porperty and does not returns it. In this class StaffHire the method setJobType(String JobType) uses data type String to set the value of the private variable. The pseudocode for this method is given below.

```
Function

DO

public void setJobType(String jobType){

   this.jobType=jobType;

          END DO

END

 }
```

c) Display Method

i. displayInfo()

The Display method in java is a method which is void type and does not returns any value from the private property. The main purpose of creating this method called displayInfo() in class StaffHire is to show the information or vlaues that are stores in private and instance varible which are declared in the class StaffHire which is also a Super class of the whole program. The pseudocode for this method is given below.

```
Function

        DO

public void displayInfo(){

   System.out.println("Vacancy no:-"+getVacancyNumber());
System.out.println("Designation:-"+getDesignation());
```

```
        System.out.println("Job Type:-"+getJobType());
            END DO
    END
      }
```

## II. Part Time Staff Hire (Child Class)

Part Time Staff hire (Child class) helps user to takes information such as Vacancy number, Designation, Job type, working hour, Wages Per Hour, Shits and also helps in creating object. And finally helps to make a final report as output and displays it to user in informative way as possible.

### a) Get method

#### i. getworkingHour()

The get method on java is a method that returns the value of the private property or any private variable. Or in simply way we can say that " The Get method returns the variable value". In this class PartimeStaffHire the method getworkingHour() uses the data type Integer and is returned by the help of get method. And Pseudocode is given below.

```
CALL

public int getworkingHour(){
        DO
    return workingHour;
END DO
  }
```

#### ii. getwagePerHour()

The get method on java is a method that returns the value of the private property or any private variable. Or in simply way we can say that " The Get method returns the variable value". In this class ParttimeStaffHire the method getwagePerHour uses the data type Integer and is returned by the help of get method. And Pseudocode is given below.

```
CALL

public int getwagePerHour(){

DO

     return wagePerHour;

END DO

  }
```

iii. getstaffName()

The get method on java is a method that returns the value of the private property or any private variable. Or in simply way we can say that " The Get method returns the variable value". In this class PartimeStaffHire the method getstaffName() uses the data type String and is returned by the help of get method. And Pseudocode is given below.

```
CALL

public String getstaffName(){

DO

     return staffName;

END DO

  }
```

iv. getjoining_Date()

The get method on java is a method that returns the value of the private property or any private variable. Or in simply way we can say that " The Get method returns the variable value". In this class ParttimeStaffhire the method getjoining_Date() uses the data type String and is returned by the help of get method. And Pseudocode is given below.

```
        CALL
```

```
public String getjoining_Date(){
DO
      return joining_Date;
END DO
  }
```

v. getqulafication()

The get method on java is a method that returns the value of the private property or any private variable. Or in simply way we can say that " The Get method returns the variable value". In this class PartimeStaffhire the method getqulaofocation uses the data type String and is returned by the help of get method. And Pseudocode is given below.

```
CALL

public String getqulafication(){
DO
      return qulafication;
END DO
}
```

vi. getappontedBy()

The get method on java is a method that returns the value of the private property or any private variable. Or in simply way we can say that " The Get method returns the variable value". In this class ParttimeStaffhire the method getappontedBy() uses the data type String and is returned by the help of get method. And Pseudocode is given below.

```
CALL

public String getappointedBy(){
    DO

        return appointedBy;
```

```
            END DO

        }
```

vii. getshifts()

The get method on java is a method that returns the value of the private property or any private variable. Or in simply way we can say that " The Get method returns the variable value". In this class ParttimeStaffhire the method getshift() uses the data type String and is returned by the help of get method. And Pseudocode is given below.

```
CALL

private String getshifts(){

DO

    return shifts;

END DO

  }
```

viii. getterminated()

The get method on java is a method that returns the value of the private property or any private variable. Or in simply way we can say that " The Get method returns the variable value". In this class ParttimeStaffhire the method getterminated() uses the data type Boolean and is returned by the help of get method. And Pseudocode is given below.

```
CALL

private Boolean getterminated(){

DO

    return terminated;

END DO
```

```
  }
```

b) Void set Mehod

i. Setshifts(String shifts)

The set method on java is a method that sets the value of the private property or private variable and does not returns the value as it is void type. Or in simply way we can say that the set method sets the value of private porperty and does not returns it. In this class ParttimeStaffHire the method setShifts uses data type String to set the value of the private variable. The pseudocode for this method is given below.

```
CALL

Public void setShifts(String shifts){

    DO

    if(joined==false)

        DO

            if(shifts==shifts)

                DO

                    output("Shift is same")

                END DO

            else

                DO

                    output("Shift has been changed.")

                    this.shifts=shifts

                END DO

            END DO
```

**else**

```
                DO
                output("Shift cannot be changed.")
                END DO
        END DO
}
```

ii. setWagesPerHour(int new WagesPerHour)

The set method on java is a method that sets the value of the private property or private variable and does not returns the value as it is void type. Or in simply way we can say that the set method sets the value of private porperty and does not returns it. In this class PartimeStaffHire the method setWagesPerHour uses data type Integer to set the value of the private variable. The pseudocode for this method is given below.

```
CALL setWagesPerHour(int newWagesPerHour){
        DO
                if(joined==false)
                        DO
                                if(WagesPerHour=wagesPerHour)
                                        DO
                                                OUTPUT("Wages Per Hour has been changed.")
                                                this.wagesPerHour=wagesPerHour
                                        END DO
                                else
                                        DO
                                                Output("Wages Per Hour is same")
```

```
                    END DO
```

```
                END DO
        else
                DO
                        output(" Wages Per Hour cannot be changed.")
                END DO
    END DO
}
```

iii. setpartTimehire(String staffName,String joiningDate,String qualification)

The set method on java is a method that sets the value of the private property or private variable and does not returns the value as it is void type. Or in simply way we can say that the set method sets the value of private porperty and does not returns it. In this class ParttimeStaffhire the method getparttimehire uses data type String to set the value of the private variable. The pseudocode for this method is given below.

```
CALL

Public void setpartTimehire(String staffName,String joiningDate,String qualification,String appointedBy){
    DO
        if(joined==true)
                DO
                        OUTPUT(" getstaffName() +Staff cannot be hired+ getjoining_date()+ by +getappointedby")
                END DO
        else
```

```
DO
    this.staffName=staffName
```

```
                this.joining_Date=joining_Date
                this.qualification=qualification
                this.appointedBy=appointedBy
                joined=true
                terminated=false
                Output("staffName+ has already been hired + getjoing date()+
by + getappointed by")
            END DO
    END DO
}
```

c) Void setterminated method

The set method on java is a method that sets the value of the private property or private variable and does not returns the value as it is void type. Or in simply way we can say that the set method sets the value of private porperty and does not returns it. In this class ParttimeStaffHire the method setterminated() uses data type String to set the value of the private variable. The pseudocode for this method is given below.

```
CALL
Public void setterminate(){
DO
  if(joined==false)
      DO
          if(terminated==true)
              DO
```

```
                OUTPUT(“The staff’s record has already been terminated")
                END DO
 END DO
 else
      DO
           this.staffName=””;
           this.joining_Date=””
           this.qualification=””
           this.appointedBy=””
           joined=false
           terminated=true
           OUTPUT(“staffName+ has been terminated”)
      END DO
END DO
}
```

d) Display

i. displayInfo()

The Display method in java is a method which is void type and does not returns any value from the private property. The main purpose of creating this method called displayInfo() in class ParttimeStaffhire is to show the information or vlaues that are stores in private and instance varible which are declared in the class StaffHire which is also a Super class of the whole program. The pseudocode for this method is given below.

```
CALL

Public void displayInfo(){
    DO
        super.displayInfo()
        if(joined==true)
            DO
                OUTPUT("Staff Name : "+ staffName)
                OUTPUT("Wages Per Hour : "+ wagesPerHour)
                OUTPUT("Working Hour :"+ workingHour)
                OUTPUT("Joining Date :"+ joiningDate)
                OUTPUT("Qualification :"+ qualification)
                OUTPUT("Appointed By : "+ appointedBy)
                OUTPUT("Income Per Day : "+ (wagesPerHour*workingHour))
            END DO
        else
            DO
                OUTPUT("Staff has not been hired yet ")
            END DO
    END DO
}
```

## III. Full time staff hire

Full time staff Hire (Child Class) helps user to take in information such as vacancy number, job type, designation, salary, working hour and helps in creating object. And finally helps to make a final output as report and displays it to user in informative way as possible.

a) Get Method

i. getsalary()

The get method on java is a method that returns the value of the private property or any private variable. Or in simply way we can say that " The Get method returns the variable value". In this class Fulltimestaffhire the method getsalary() uses the data type Integer and is returned by the help of get method. And Pseudocode is given below.

```
FUNCTION

public    int    getsalary(){

     return salary;

END

  }
```

ii. getworkingHour()

The get method on java is a method that returns the value of the private property or any private variable. Or in simply way we can say that " The Get method returns the variable value". In this class Fulltimestaffhire the method getworkingHour() uses the data type Integer and is returned by the help of get method. And Pseudocode is given below.

```
FUNCTION

  public    int    getworkingHour(){

     return workingHour;

END

}
```

iii. getstaffName()

The get method on java is a method that returns the value of the private property or any private variable. Or in simply way we can say that '' The Get method returns the variable value". In this class Fulltimestaffhire the method getstaffName() uses the data type String and is returned by the help of get method. And Pseudocode is given below.

```
FUNCTION

  public String getstaffName(){

      return staffName;

END

  }
```

iv. getjoiningDate()

The get method on java is a method that returns the value of the private property or any private variable. Or in simply way we can say that " The Get method returns the variable value". In this class Fulltimestaffhire the method getjoiningDate() uses the data type String and is returned by the help of get method. And Pseudocode is given below.

```
FUNCTION

  public    String    getjoinigDate(){

      return joinigDate;

END

  }
```

v. getqualification()

The get method on java is a method that returns the value of the private property or any private variable. Or in simply way we can say that " The Get method returns the variable value". In this class Fulltimestaffhire the method getqualification() uses the data type String and is returned by the help of get method. And Pseudocode is given below.

```
FUNCTION

  public String getqualification(){
        return qualification;
```

```
END

  }
```

vi. getappontedBy()

The get method on java is a method that returns the value of the private property or any private variable. Or in simply way we can say that " The Get method returns the variable value". In this class Fulltimestaffhire the method getappontedBy() uses the data type String and is returned by the help of get method. And Pseudocode is given below.

```
FUNCTION

  public    String    getappointedBy(){

     return appointedBy;

END

  }
```

vii. getjoined()

The get method on java is a method that returns the value of the private property or any private variable. Or in simply way we can say that " The Get method returns the variable value". In this class Fulltimestaffhire the method getjoined() uses the data type Boolean and is returned by the help of get method. And Pseudocode is given below.

```
FUNCTION

  public Boolean getjoined(){

     return joined;

END

  }
```

b) Void Method

i. setsalary(int salary)

The set method on java is a method that sets the value of the private property or private variable and does not returns the value as it is void type. Or in simply way we can say that the set method sets the value of private porperty and does not returns it. In this class FulltimeStaffhire the method setsalary(int salary) uses data type Integer to set the value of the private variable. The pseudocode for this method is given below.

```
FUNCTION

Public void setsalary(int salary ){

DO

    if(joined==false)

        DO

            if(salary=salary)

                DO

                    Output(“The salary has been changed.”)

                    this.salary=salary

                END DO

        END DO

    ELSE

        DO

            OUTPUT(“It is not possible to change the salary”)

        END DO

end

}
```

ii. setWorkingHour(int newworkingHour)

The set method on java is a method that sets the value of the private property or private variable and does not returns the value as it is void type. Or in simply way we can say that the set method sets the value of private porperty and does not returns it. In this class FulltimeStaffhire the method setWorkinghour() uses data type Integer to set the value of the private variable. The pseudocode for this method is given below.

```
FUNCTION setWorkingHour(int newworkingHour){
    DO
            this.workingHour=WorkingHour
end DO
}
```

iii. setfullhire(String staffName, String joiningDate, String qualification)

The set method on java is a method that sets the value of the private property or private variable and does not returns the value as it is void type. Or in simply way we can say that the set method sets the value of private porperty and does not returns it. In this class FulltimeStaffhire the method setfullhire() uses data type String to set the value of the private variable. The pseudocode for this method is given below.

```
FUNCTION
Public void setfullhire(String staffName, String joiningDate, String qualification, String appointedBy){
    DO
        if(joined==false)
            DO
                this.staffName=staffName;
                this.joiningDate=joiningDate;
                this.qualification=qualification;
                this.appointedBy=appointedBy;
                joined=true
                Output("Staff has been hired")
```

```
                END DO

    END DO
else
DO
                    OUTPUT(“getstaffName()+ has already been hired on date
+getjoiningdate()”;
                END DO


end
}
```

## c) Display Void

### i. displayInfo()

The Display method in java is a method which is void type and does not returns any value from the private property. The main purpose of creating this method called displayInfo() in class FulltimeStaffhire is to show the information or vlaues that are stores in private and instance varible which are declared in the class StaffHire which is also a Super class of the whole program. The pseudocode for this method is given below.

```
FUNCTION displayInfo(){
    DO

        super.displayInfo()
        if(joined==true)
```

DO

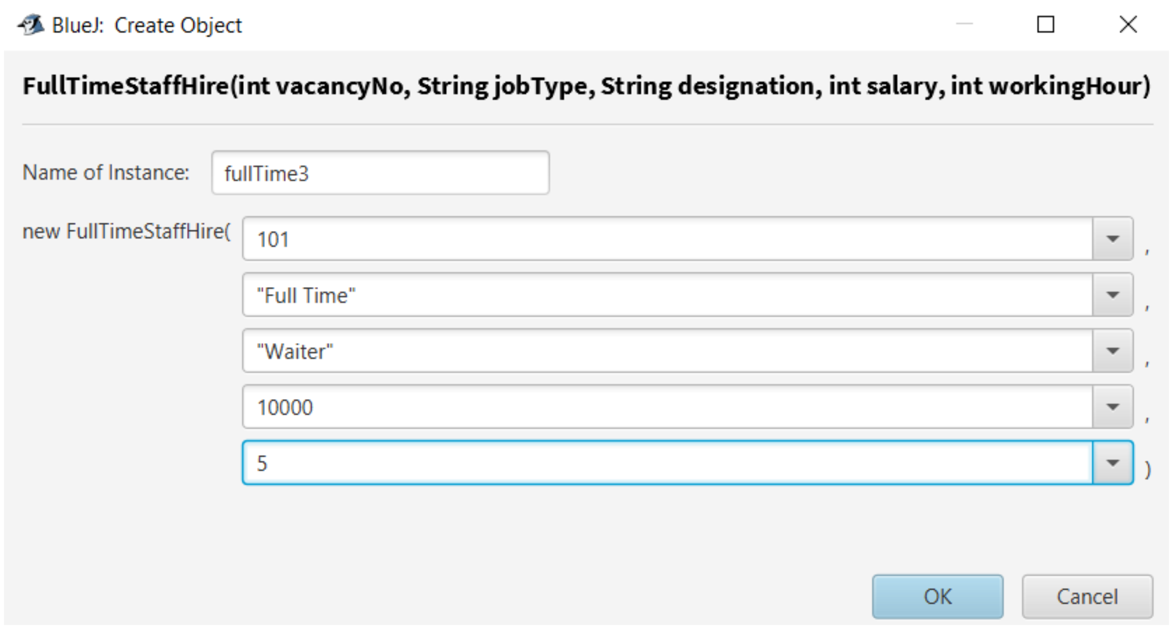
```
                    print  “Staff  Name  :  +staffName”
                    print “Salary : + salary”
                    print  “Working  Hour  :  +  workingHour”
                    print “Joining Date : + joiningDate” pirnt
                    “Qualification   :   +   qualification”   print
                    “Appointed By : + appointedBy”
                END DO

        Else

                DO

                OUTPUT(“Staff has been not hired yet.”)

                END DO

    END DO
end
}
```

## 3. Testing

## I. Test 1 (Displaying Details of FulltimeStaffhire)

| | |
|---|---|
| Objective: | Displaying Details which are stored in FullTimeStaffhire. |
| Action: | Firstly on class FulltimeStaffhire following value are inserted in the constructor. where VacancyNumber is set as 101, Jobtype is given as Full time, Designation is given as Waiter, Working hour isset as 5 hr and salary is set as 10000. After inserting all the values the object was created by inserting the value on the above constructor. After this process the process calling method called HireFulltimeStaff() is called and following values are inserted, staffName as Manish Bomzon, Qualification BBA, joining Date as 2019-09-09 and is appointed by Anish Sherchan. And Finally the Displaying method called displayInfo() was called. |
| Expected result: | After following all the action given ablove and after calling the method called displayInfo() of class Staffhire must show all the values which were stored in private instance variable staffName, Workinghour,joining date, salary, appointedBy and qualification. |
| Actual result: | The result of following test was as same as what was expected as all the value stored in private instance variable was shown. |
| Conclusion: | The test which was carried out was a sucess. |

Table 4: Test 1 (Displaying information)



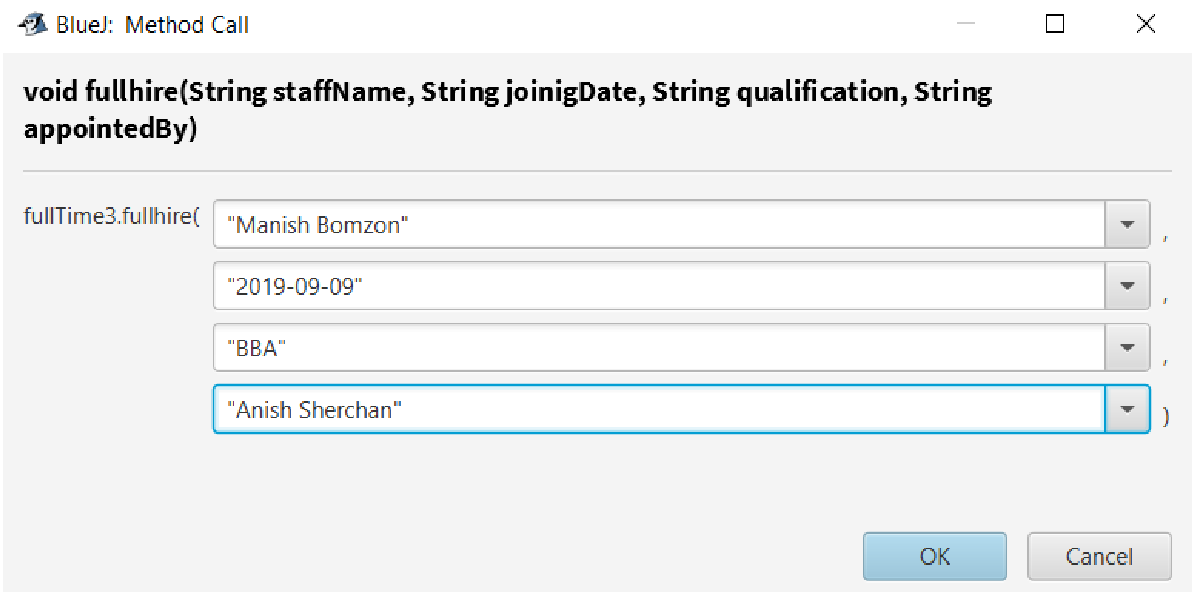
BlueJ: Create Object

**FullTimeStaffHire(int vacancyNo, String jobType, String designation, int salary, int workingHour)**

Name of Instance: fullTime3

new FullTimeStaffHire( 101 , "Full Time" , "Waiter" , 10000 , 5 )

Figure 1:Creating object of FullTimeStaffHire

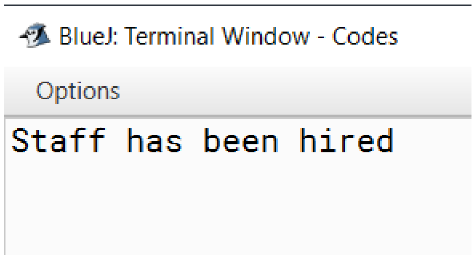


BlueJ: Method Call

**void fullhire(String staffName, String joinigDate, String qualification, String appointedBy)**

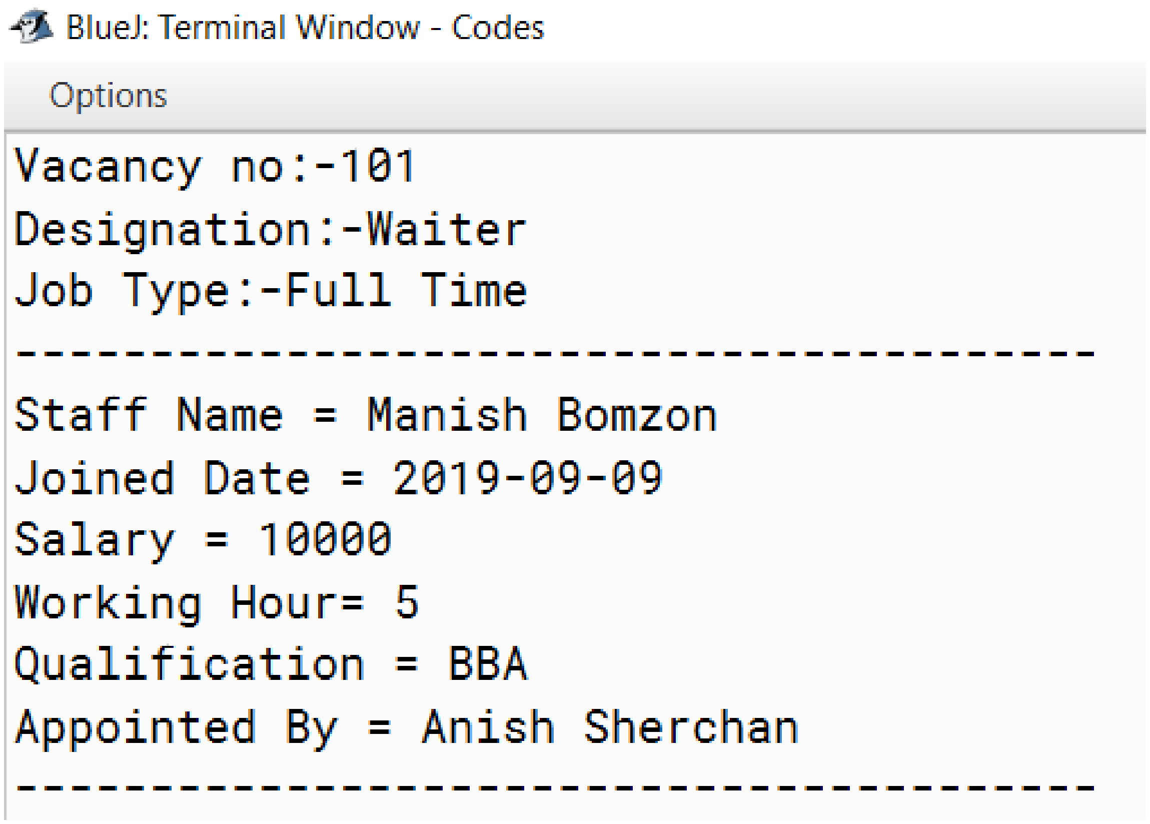
fullTime3.fullhire( "Manish Bomzon" , "2019-09-09" , "BBA" , "Anish Sherchan" )

*Figure 2:* Calling Method fullhire



```
Staff has been hired
```

*Figure 3:* Message from the terminal



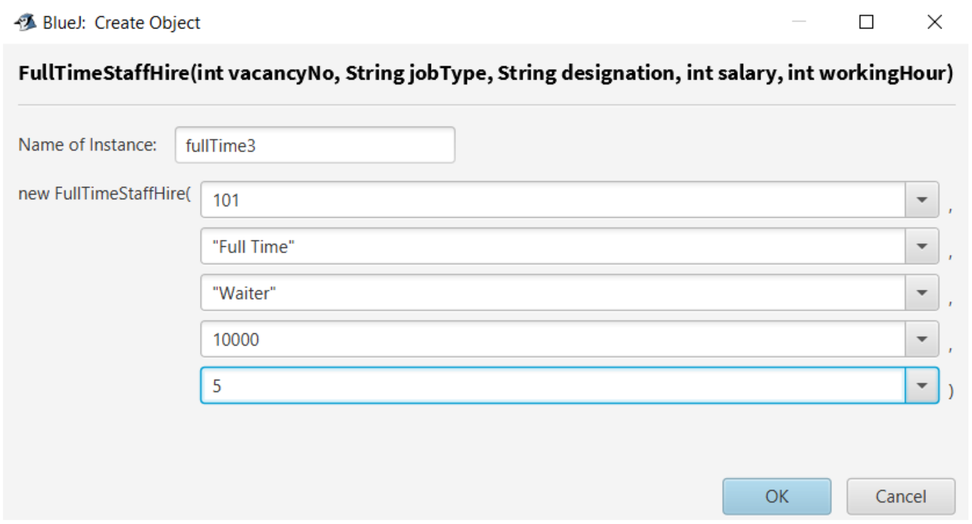
```
Vacancy no:-101
Designation:-Waiter
Job Type:-Full Time
----------------------------------------
Staff Name = Manish Bomzon
Joined Date = 2019-09-09
Salary = 10000
Working Hour= 5
Qualification = BBA
Appointed By = Anish Sherchan
----------------------------------------
```

*Figure 4:*Method DisplayInfo() displaying all info

## II. Test 2 (Inspecting FullTimeStaffHire)

| | |
|---|---|
| Objective: | Inspecting FullTimeStaffhire class and also appointing the Fulltimestaff and finally the class FulltimeStaffHire is re inspected. |
| Action: | Firstly on class FulltimeStaffhire following value are inserted in the constructor. where VacancyNumber is set as 101, Jobtype is given as Fulltime, Designation is given as Waiter, Working hour isset as 5 hr and salary is set as 10000. After inserting all the values the object was created by inserting the value on the above constructor. And was also Inspected. After this process the process calling method called HireFulltimeStaff() is called and following values are inserted, staffName as Manish Bomzon, Qualification BBA, joining Date as 2019-09-09 and is appointed by Anish Sherchan. And Finally the object was inspected again. |
| Expected result: | The method which has been used in FulltimeStaffHire must get the value which were inserted by the user and again it must assign the values to their respectuve private instance variables. |
| Actual result: | The result of following test was as same as what was expected as all the value stored in private instance variable. |
| Conclusion: | The test which was carried out was a success. |

*Table 5:* Inspection of class FulltimeStaff Hire

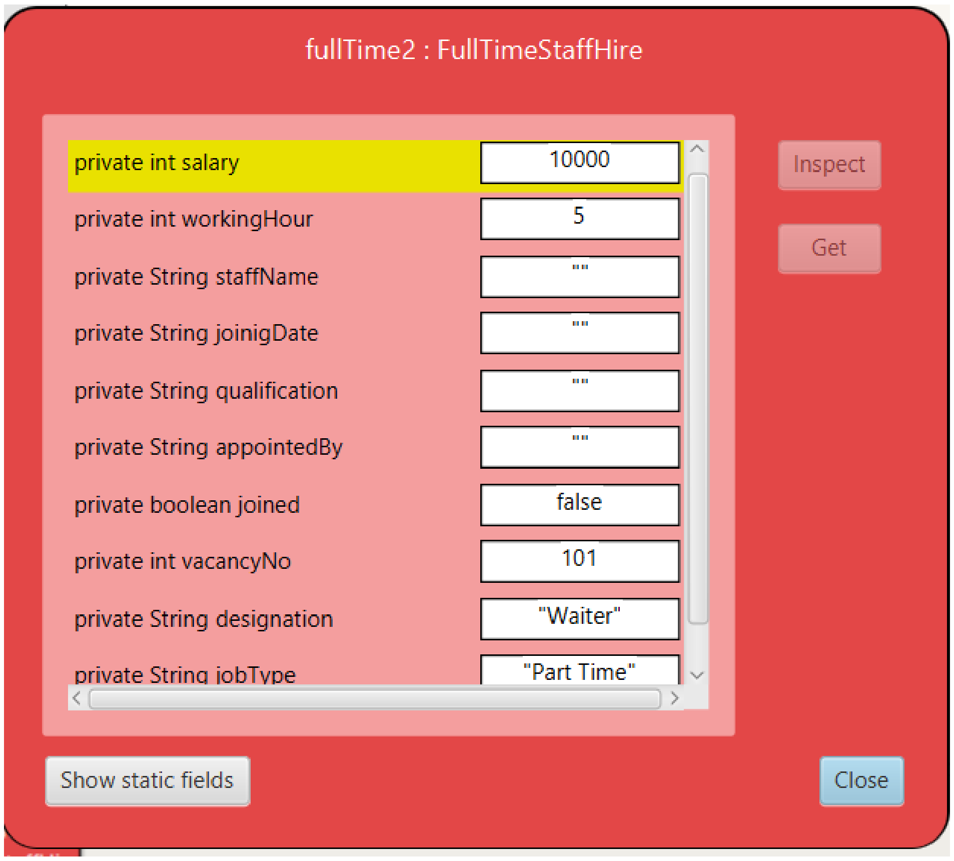


BlueJ: Create Object

**FullTimeStaffHire(int vacancyNo, String jobType, String designation, int salary, int workingHour)**

| | |
|---|---|
| Name of Instance: | fullTime3 |
| new FullTimeStaffHire( | 101 , |
| | "Full Time" , |
| | "Waiter" , |
| | 10000 , |
| | 5 ) |

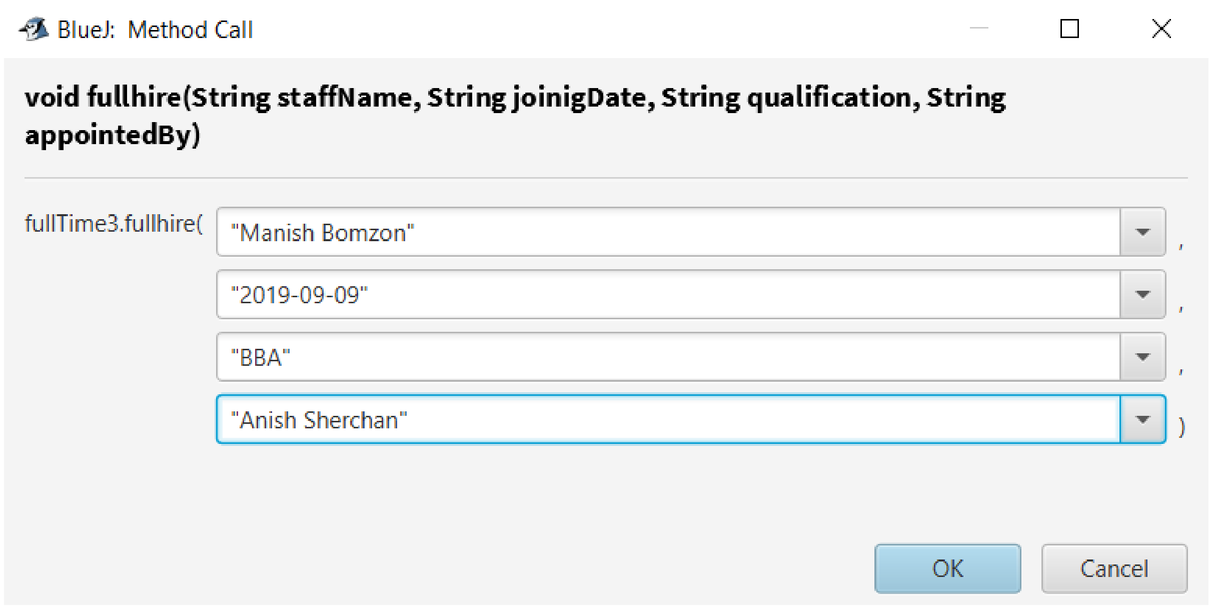
*Figure 5:*Creating object of fulltmestaffhire



fullTime2 : FullTimeStaffHire

| | |
|---|---|
| private int salary | 10000 |
| private int workingHour | 5 |
| private String staffName | "" |
| private String joinigDate | "" |
| private String qualification | "" |
| private String appointedBy | "" |
| private boolean joined | false |
| private int vacancyNo | 101 |
| private String designation | "Waiter" |
| private String jobType | "Part Time" |

*Figure 6*: Inspecting FullTimeStaffHire

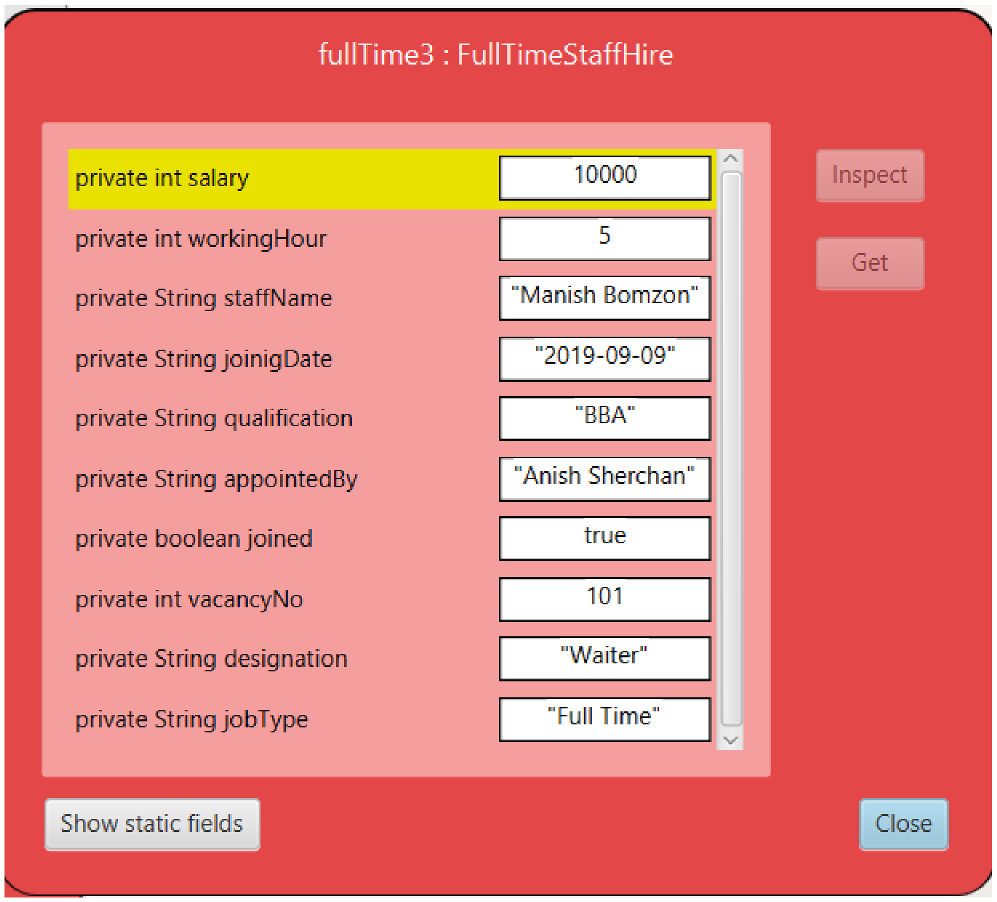


BlueJ: Method Call

**void fullhire(String staffName, String joinigDate, String qualification, String appointedBy)**

fullTime3.fullhire( "Manish Bomzon" , "2019-09-09" , "BBA" , "Anish Sherchan" )

*Figure 7:* Calling method fullhire()



fullTime3 : FullTimeStaffHire

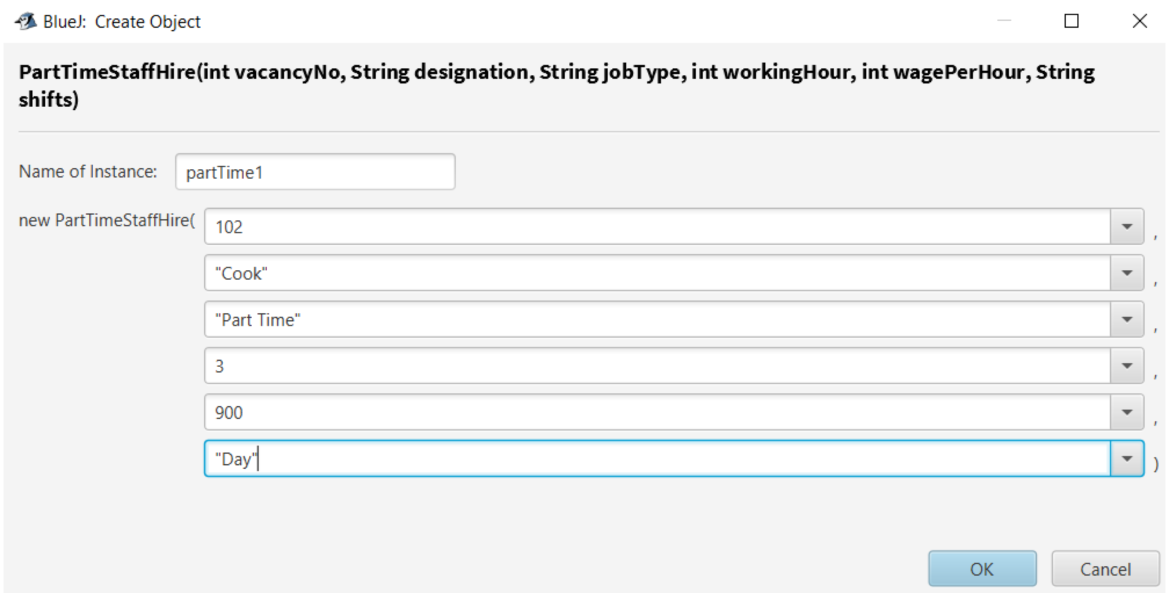
| | |
|---|---|
| private int salary | 10000 |
| private int workingHour | 5 |
| private String staffName | "Manish Bomzon" |
| private String joinigDate | "2019-09-09" |
| private String qualification | "BBA" |
| private String appointedBy | "Anish Sherchan" |
| private boolean joined | true |
| private int vacancyNo | 101 |
| private String designation | "Waiter" |
| private String jobType | "Full Time" |

*Figure 8:* Re inspecting The class FulltimeStaffHire

## III. Test 3 (Displaying the details which are stored in PartimeStaffHire)

| | |
|---|---|
| Objective: | Displaying Details which are stored in FullTimeStaffhire. |
| Action: | Firstly on class PartimeStaffhire following values are inserted in the constructor. where vacancyNo is set as 102, Designation is set as Cook, jobtype as Part time and working hour as 3 wagesPerHour as 900 and shits as Day<br>After inserting all the values the object was created by inserting the values on the above constructor. After this process the process of calling method called HireParttimeStaff() is called and following values as asked must be inserted and finally the displayInfo() was called. |
| Expected result: | After following all the action given above and after calling the method display info of class staffhire must show all the values which were stored in private instance variable staffName, Woringhour, Shift, WagesperHour, appointedBy and qualification. |
| Actual result: | The result of following test was as same as what was expected as all the value stored in private instance variable was shown. |
| Conclusion: | The test which was carried out was a success. |

*Table 6*: Displaying details on full time staffHire



BlueJ: Create Object

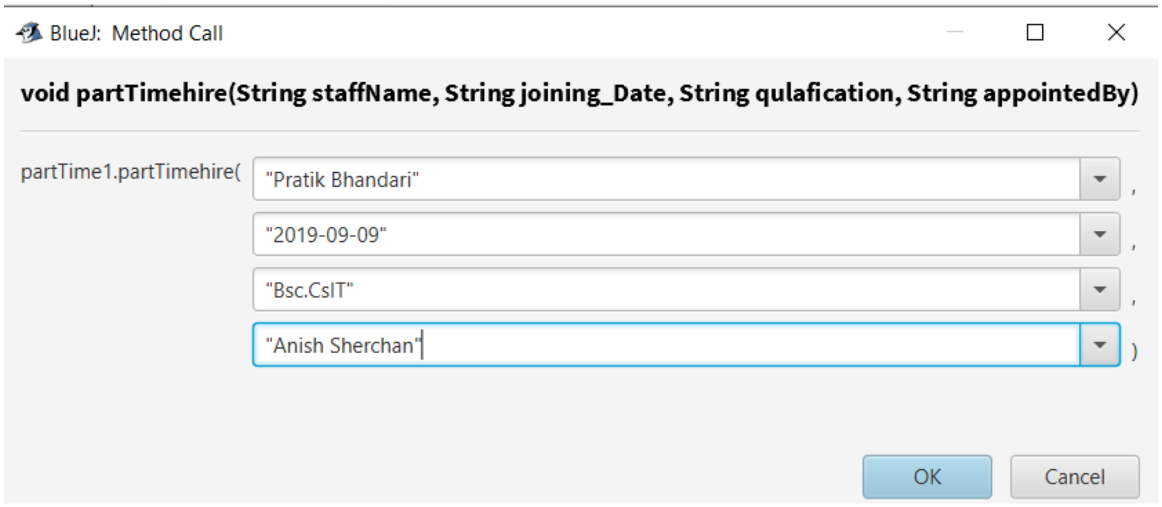
**PartTimeStaffHire(int vacancyNo, String designation, String jobType, int workingHour, int wagePerHour, String shifts)**

Name of Instance: partTime1

new PartTimeStaffHire( 102 , "Cook" , "Part Time" , 3 , 900 , "Day" )

OK Cancel

*Figure 9:* Creating object



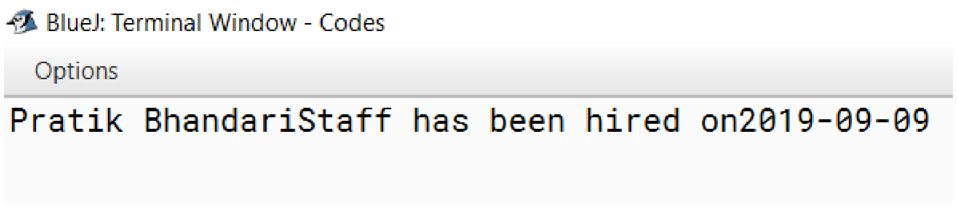
BlueJ: Method Call

**void partTimehire(String staffName, String joining_Date, String qulafication, String appointedBy)**

partTime1.partTimehire( "Pratik Bhandari" , "2019-09-09" , "Bsc.CsIT" , "Anish Sherchan" )

OK Cancel

*Figure 10*: Calling partTimehire()

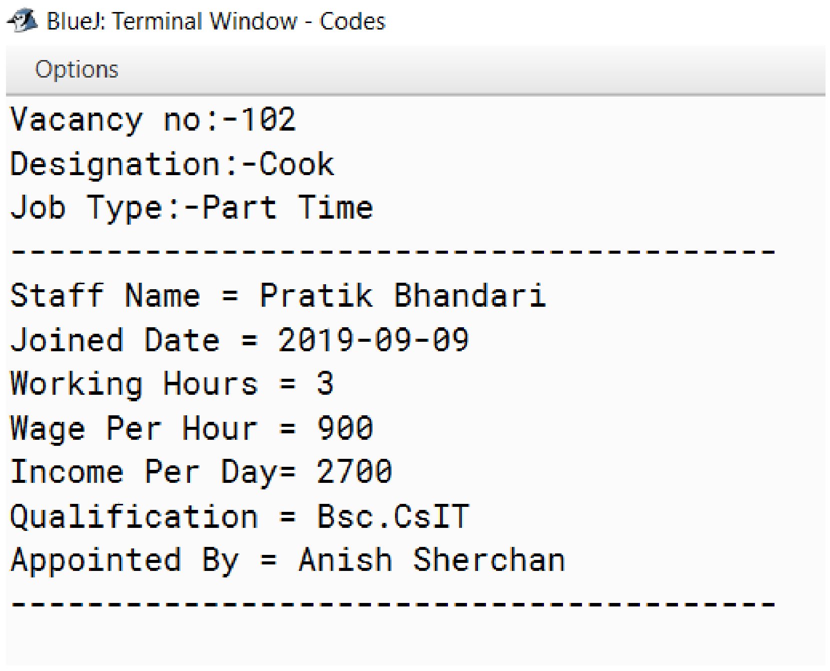


BlueJ: Terminal Window - Codes

Options

```
Pratik BhandariStaff has been hired on2019-09-09
```

*Figure 11*: 1st message of terminal


BlueJ: Terminal Window - Codes

Options

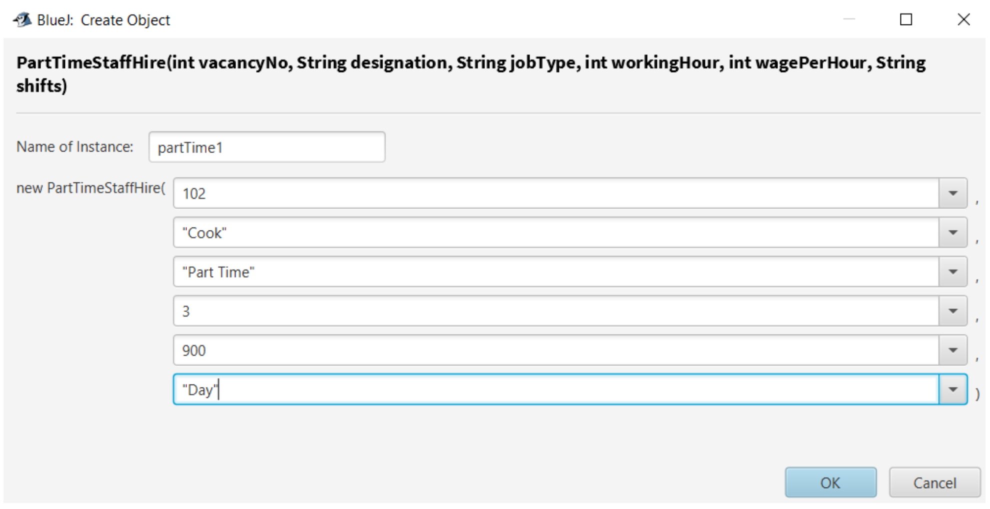
```
Vacancy no:-102
Designation:-Cook
Job Type:-Part Time
----------------------------------------
Staff Name = Pratik Bhandari
Joined Date = 2019-09-09
Working Hours = 3
Wage Per Hour = 900
Income Per Day= 2700
Qualification = Bsc.CsIT
Appointed By = Anish Sherchan
----------------------------------------
```

*Figure 12*: DisplayInfo()

## IV. Test 4 (Inspecting the class ParttimeStaffhire())

| | |
|---|---|
| Objective: | Inspecting PartTimeStaffhire class also appointing the partimestaff and finally the class PulltimeStaffHire is re inspected. |
| Action: | Firstly on class PulltimeStaffhire following value are inserted in the constructor. where VacancyNumber is set as 102, Jobtype is given as |

| | |
|---|---|
| | Parttime, Designation is given as Cook, Working hour is set as 2 hr and income is set as 900. After inserting all the values the object was created by inserting the value on the above constructor. And was also Inspected. After this process the process calling method called HireparttimeStaff() is called and following values are inserted, staffName as Pratik bhandari, Qualification Bsc.Csit, joining Date as 2019-09-09 and is appointed by Anish Sherchan. And Finally the object was inspected again. |
| Expected result: | The method which has been used in ParttimeStaffHire must get the value which were inserted by the user and again it must assign the values to their respectuve private instance variables. |
| Actual result: | The result of following test was as same as what was expected as all the value stored in private instance variable was shown. |
| Conclusion: | The test which was carried out was a success. |

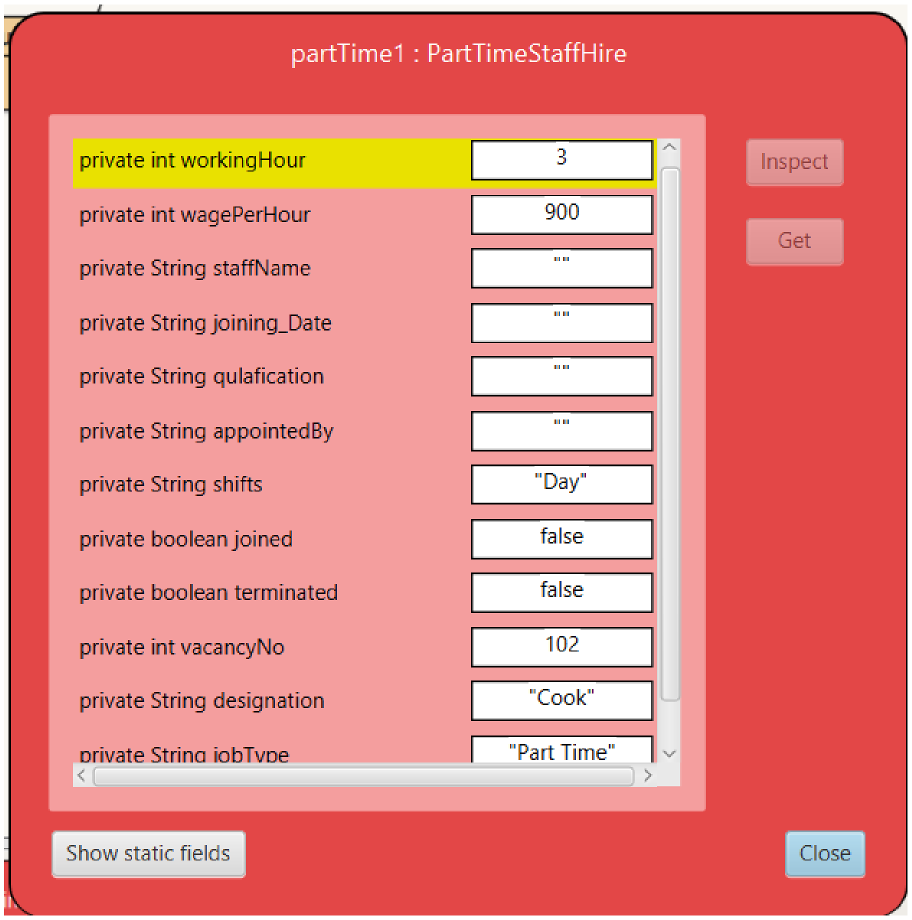


PartTimeStaffHire(int vacancyNo, String designation, String jobType, int workingHour, int wagePerHour, String shifts)

Name of Instance: partTime1

new PartTimeStaffHire( 102 , "Cook" , "Part Time" , 3 , 900 , "Day" )

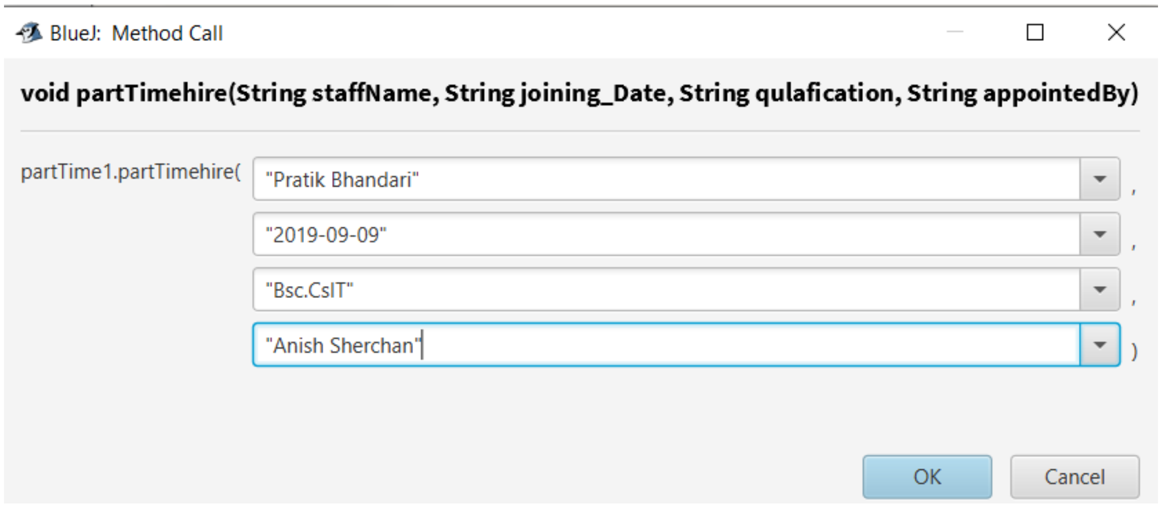
*Figure 13:* Creating object



partTime1 : PartTimeStaffHire

| | |
|---|---|
| private int workingHour | 3 |
| private int wagePerHour | 900 |
| private String staffName | "" |
| private String joining_Date | "" |
| private String qulafication | "" |
| private String appointedBy | "" |
| private String shifts | "Day" |
| private boolean joined | false |
| private boolean terminated | false |
| private int vacancyNo | 102 |
| private String designation | "Cook" |
| private String jobType | "Part Time" |

*Figure 14:* Inspecting PartTimestaffHire()

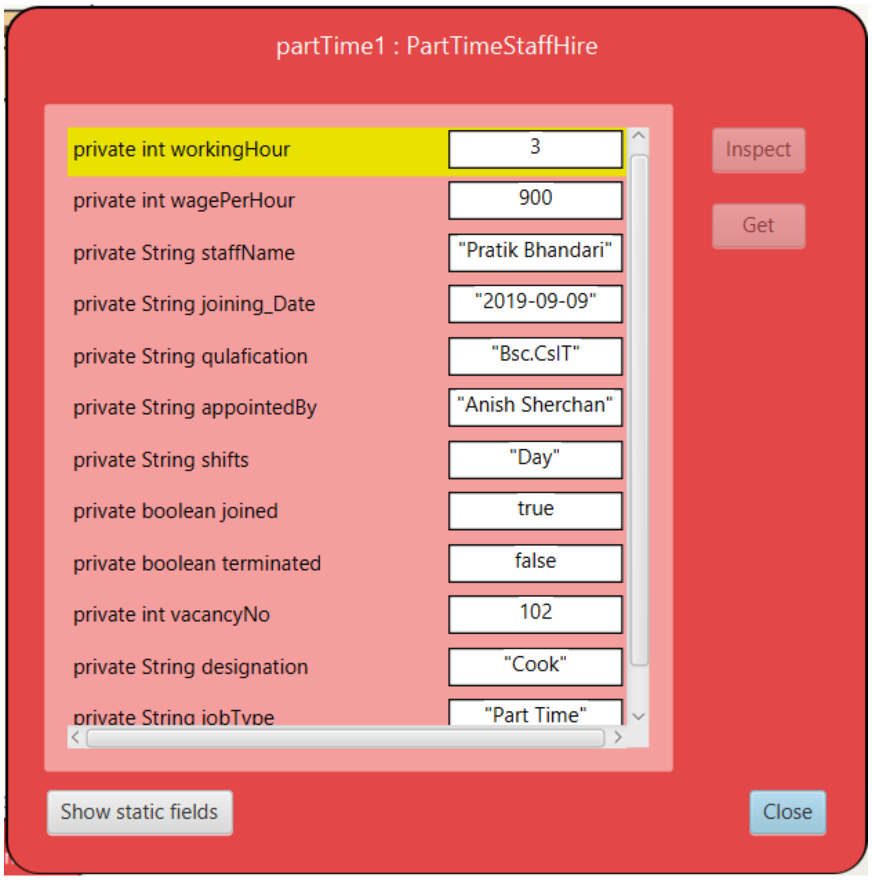


BlueJ: Method Call

**void partTimehire(String staffName, String joining_Date, String qulafication, String appointedBy)**

```
partTime1.partTimehire( "Pratik Bhandari" ,
                        "2019-09-09" ,
                        "Bsc.CsIT" ,
                        "Anish Sherchan" )
```

*Figure 15:* Calling Method partTimehire()



| | |
|---|---|
| private int workingHour | 3 |
| private int wagePerHour | 900 |
| private String staffName | "Pratik Bhandari" |
| private String joining_Date | "2019-09-09" |
| private String qulafication | "Bsc.CsIT" |
| private String appointedBy | "Anish Sherchan" |
| private String shifts | "Day" |
| private boolean joined | true |
| private boolean terminated | false |
| private int vacancyNo | 102 |
| private String designation | "Cook" |
| private String jobType | "Part Time" |

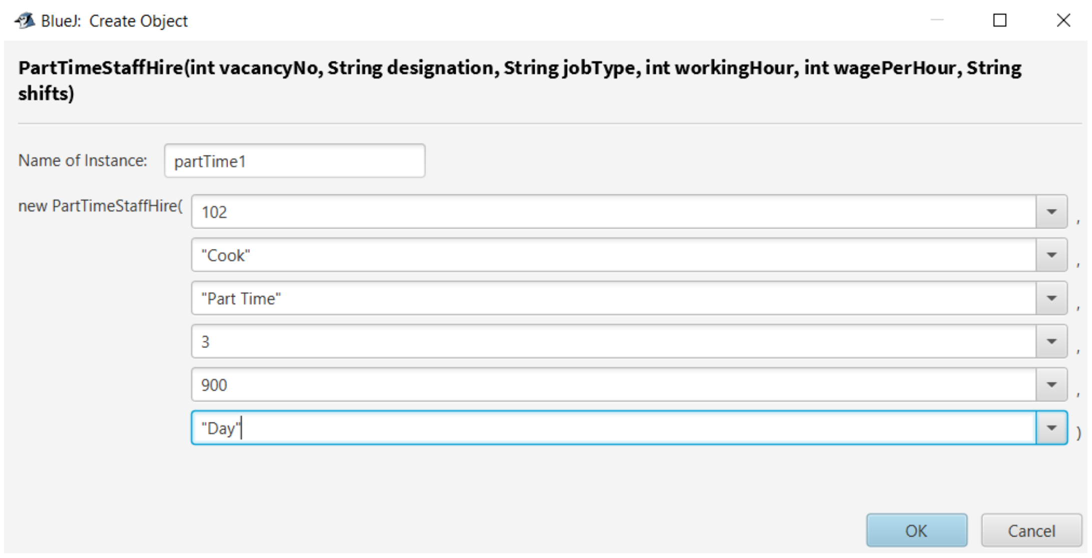
*Figure 16:* Re inspecting ParttimeHire()

## V. Test 5 ( Calling Method terminatestaff())

| | |
|---|---|
| Objective: | Inspecting PartTimeStaffhire class also appointing the partimestaff and finally terminate the appointed staff and also re inspecte the partTimeStaffhire. |
| Action: | Firstly on class PulltimeStaffhire following value are inserted in the constructor. where VacancyNumber is set as 102, Jobtype is given as Parttime, |

| | |
|---|---|
| | Designation is given as Cook, Working hour is set as 2 hr and income is set as 900. After inserting all the values the object was created by inserting the value on the above constructor. And was also Inspected. After this process the process calling method called HireparttimeStaff() is called and following values are inserted, staffName as Pratik bhandari, Qualification Bsc.Csit, joining Date as 2019-09-09 and is appointed by Anish Sherchan. The object was inspected again. And finally the appointed staff was terminated by call the method call terminateStaff(). |
| Expected result: | All the record of the staff who was terminated must be removed permanantly and must be empty for new staff. |
| Actual result: | The result of following test was as same as what was expected as all the value stored in private instance variables of the staff was removed. |
| Conclusion: | The test which was carried out was a success. |

*Table 7*: Calling method terminate Staff

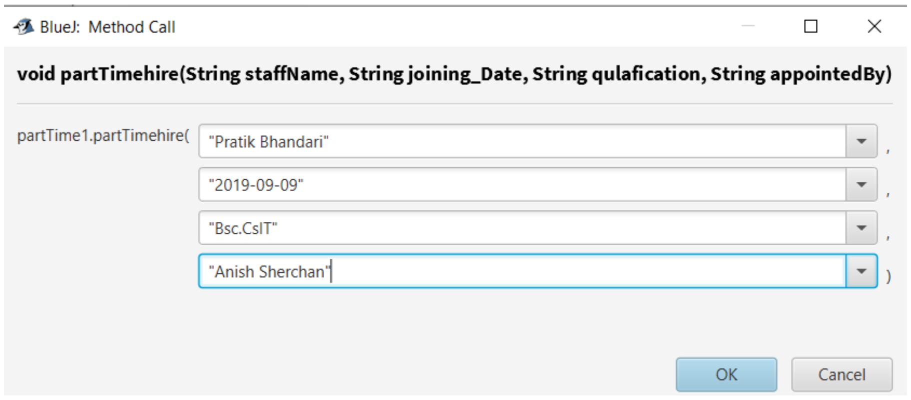

BlueJ: Create Object

PartTimeStaffHire(int vacancyNo, String designation, String jobType, int workingHour, int wagePerHour, String shifts)

Name of Instance: partTime1

new PartTimeStaffHire( 102 , "Cook" , "Part Time" , 3 , 900 , "Day" )

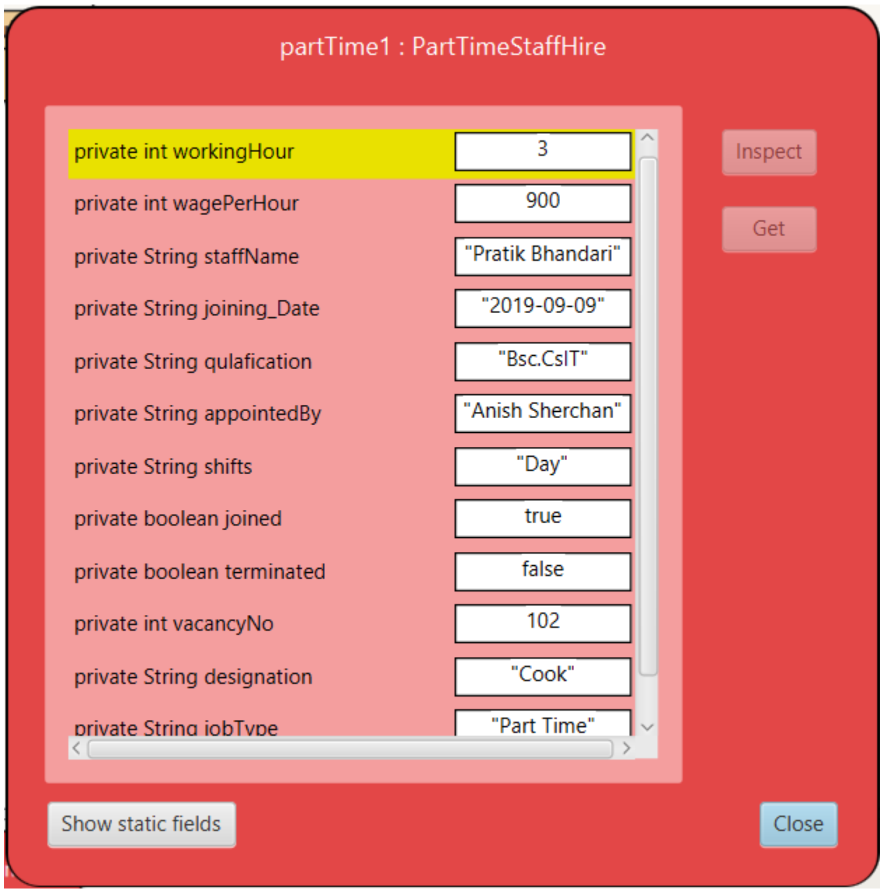
*Figure 17*: Creating Object


BlueJ: Method Call

void partTimehire(String staffName, String joining_Date, String qulafication, String appointedBy)

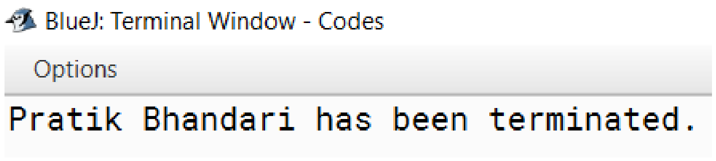
partTime1.partTimehire( "Pratik Bhandari" , "2019-09-09" , "Bsc.CsIT" , "Anish Sherchan" )

*Figure 18*: Calling Method partTImeStaffhire()



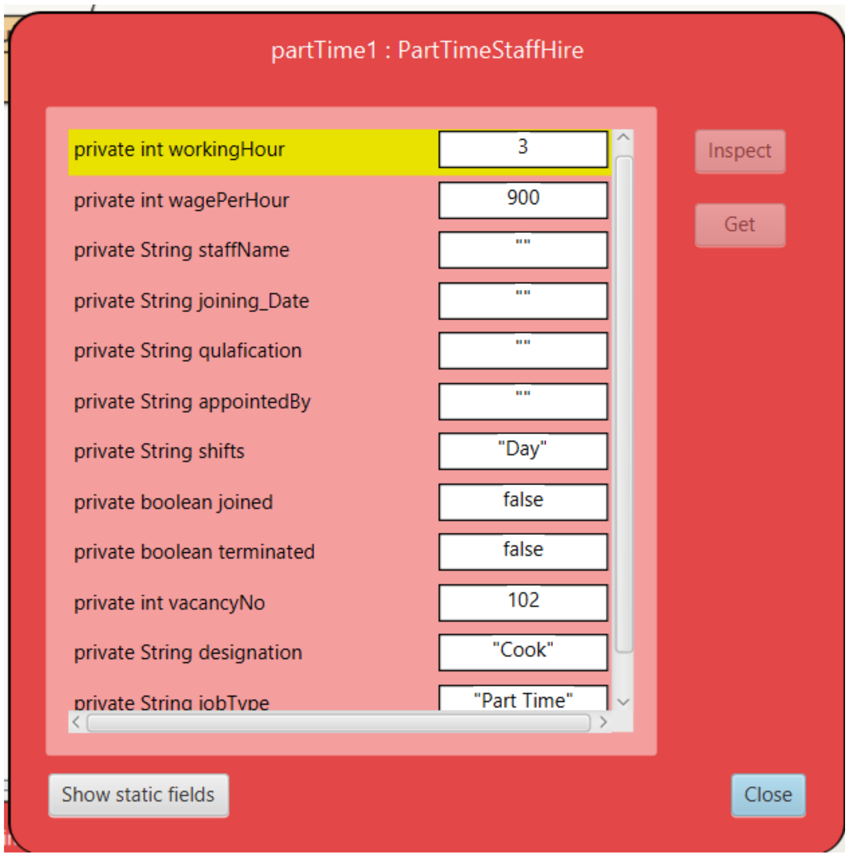
| | |
|---|---|
| private int workingHour | 3 |
| private int wagePerHour | 900 |
| private String staffName | "Pratik Bhandari" |
| private String joining_Date | "2019-09-09" |
| private String qulafication | "Bsc.CsIT" |
| private String appointedBy | "Anish Sherchan" |
| private String shifts | "Day" |
| private boolean joined | true |
| private boolean terminated | false |
| private int vacancyNo | 102 |
| private String designation | "Cook" |
| private String jobType | "Part Time" |

*Figure 19*: Inspecting the parttimestaffHire()



```
Pratik Bhandari has been terminated.
```

*Figure 20*: Terminating the appointed staff



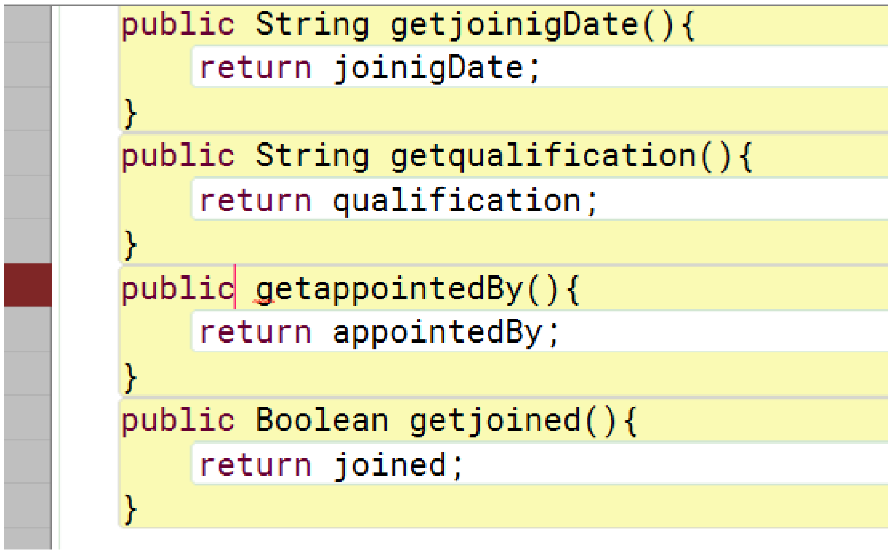
| | |
|---|---|
| private int workingHour | 3 |
| private int wagePerHour | 900 |
| private String staffName | "" |
| private String joining_Date | "" |
| private String qulafication | "" |
| private String appointedBy | "" |
| private String shifts | "Day" |
| private boolean joined | false |
| private boolean terminated | false |
| private int vacancyNo | 102 |
| private String designation | "Cook" |
| private String jobType | "Part Time" |

*Figure 21:* Re inspecting the parttimestaffhire()

## 4. Errors detection and Correction

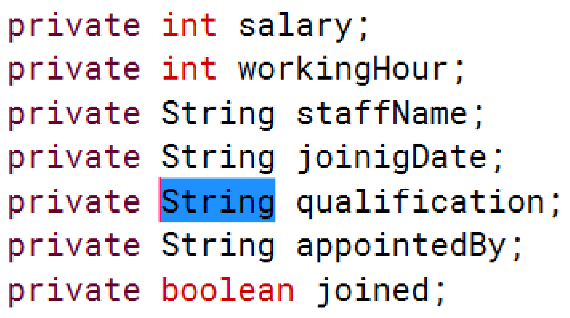
### I. Error 1 (Data type error)

Here in error 1 we can see that there is error in code “public getappointedBy()” due to which the program is not able to get compiled the problem was detected during compiling the program and was again detected by checking checking the datatype in the private instance variable decleration side at the top and finally the problem was solved simply by adding the data type string to the missing place and was complied. The problem which was detected and solved with photo is given below.



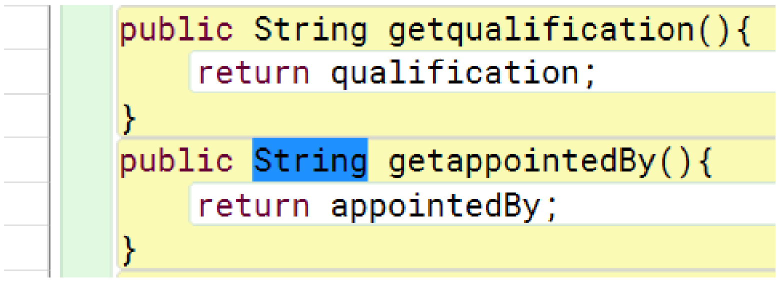
```
public String getjoinigDate(){
    return joinigDate;
}
public String getqualification(){
    return qualification;
}
public getappointedBy(){
    return appointedBy;
}
public Boolean getjoined(){
    return joined;
}
```

*Figure 22:* Error od Data Type



```
private int salary;
private int workingHour;
private String staffName;
private String joinigDate;
private String qualification;
private String appointedBy;
private boolean joined;
```

*Figure 23*: Error detection

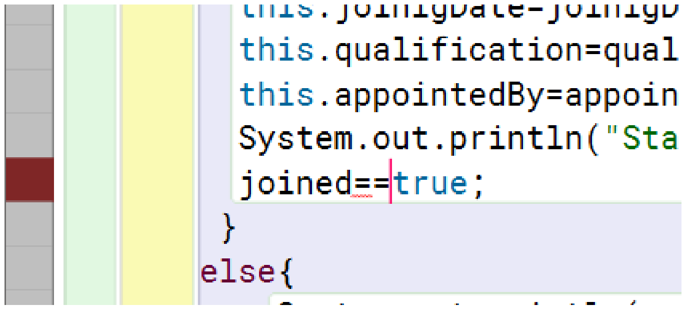


```
public String getqualification(){
    return qualification;
}
public String getappointedBy(){
    return appointedBy;
}
```

*Figure 24*: Error Prevention

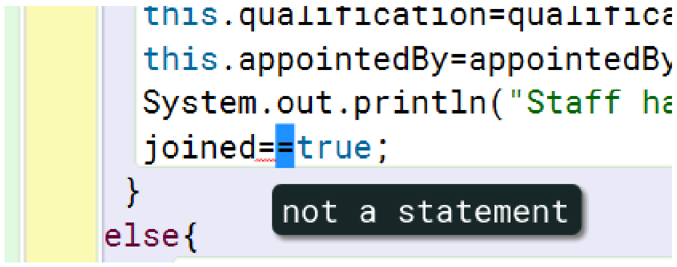
## II. Error 2

Here in first error the code “joined==true” has got an error due to which it is stopping the program from getting complied. The error was detected and was tried to solved using many method such as searching answers on books and on the internet and finally we knew that the problem was generated by “=” and was removed and complied. The photos of error detection and prevention are given below.



```
this.qualification=qual
this.appointedBy=appoin
System.out.println("Sta
joined==true;
}
else{
```

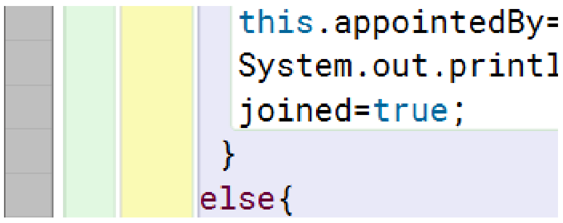
*Figure 25* :Error 2



```
this.qualification=qualifica
this.appointedBy=appointedBy
System.out.println("Staff ha
joined==true;
}
else{
```

not a statement

*Figure 26*: Detection of Error 2

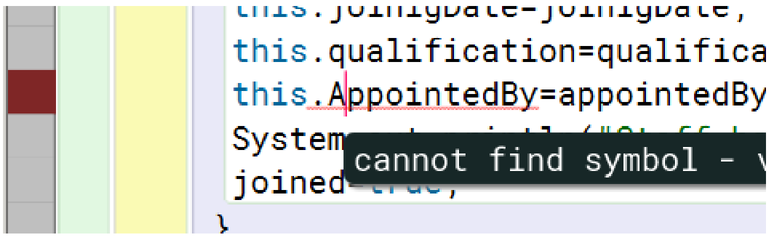


```
this.appointedBy=
System.out.printl
joined=true;
}
else{
```

*Figure 27*: Prevention of Error 2

### III. Error 3

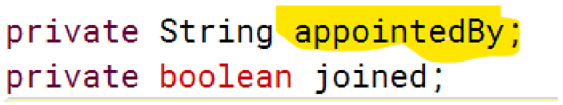
Here in Error 3 an error on the code “this.AppointedBy=appointedBy” stoped the program from getting complied and the error was detected. Latter on after checking the private instance variable we came to knew that there was a problem on this.AppointedBy as the actual variable name was appoitedBy and was solved simply by replacing the letter A with small a as shown in picture below.



```
this.qualification=qualifica
this.AppointedBy=appointedBy
System
joined
```

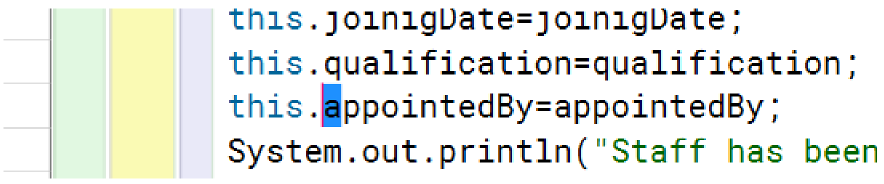
```
cannot find symbol -
```

*Figure 28*: Error 3



```
private String appointedBy;
private boolean joined;
```

*Figure 29*: Detection of Error 3



```
this.joinigDate=joinigDate;
this.qualification=qualification;
this.appointedBy=appointedBy;
System.out.println("Staff has been
```

*Figure 30*: Prevention of Error3

## 5. Conclusion

Java is a programming language and computing platform first released by Sun Microsystems in 1995. There are lots of applications and websites that will not work unless you have Java installed, and more are created every day. (Java, n.d.) Or in other word we can say that java is an object oriented programming language and computing platform where we can deal with a programming problems in object oriented way and helps in developing a huge and complex programs in easy way possible. By the help of java this project where a user can use the program for hiring staff has been developed and was submitted to the classroom. The whole project was done in a Java platform provider called BlueJ. BlueJ is a Java integrated development environment designed for college and university students (Anon., n.d.)

Throughout the period for the development of the project I have learned many important aspects that were necessary for proper development of the project and it also helped me in self learning and realization process for recalling the things that I have heard and learned from the past. During the Development process I have gone through many kinds of problems and found the solution through many ways such as visiting library, accessing internets for educational use, learning much more things about Java from YouTube and all.

Also, During the development process, I came to knew that the knowledge that I have was very insufficient, so to cope with the requirements of my project development me along with my fellow mates started learning from our own mistakes and the main way of learning was group discussion and sharing of ideas and vision individual had. Another way of learning was through the help of elder brothers that were present in the college. Also throughout the learning process we can say that we had learned about GUI, CUI, Action Listener, Event Listener and many more which are required for GUI development. We also can say that we learned many more about method calling and so on. So through all my research and findings that I have done for this project finally has leaded me for the development of this project. And finally we can conclude that due to help of our teachers which are present in the college and help of the Bluej the project was finished.

## 7. References

Anon., n.d. *DominateJava.* [Online]
[Accessed 2020].

Anon., n.d. *Javatpoint.* [Online]
Available at: https://www.java.com/en/java_in_action/bluej.jsp
[Accessed 9 1 2020].

DominateJava, n.d. *DominateJava.* [Online]
Available at: https://dominatejava.blogspot.com/2019/08/what-is-bluej.html
[Accessed 8 1 2020].

Java, n.d. *Java.* [Online]
Available at: https://java.com/en/download/faq/whatis_java.xml
[Accessed 8 1 2020].

## 8. Appendix

### I. StaffHire

```
/**
 * Write a description of class StaffHire here.
 *Write a description of
 * @author (Anish Sherchan)
 * @version (0.001)
 */
public class StaffHire
{
  private    int    vacancyNo;
  private String designation;
  private String jobType;
  public        StaffHire(int        vacancyNo,String        designation,String        jobType){
 this.vacancyNo=vacancyNo;
 this.designation=designation;
 this.jobType=jobType;
}
public int getVacancyNumber(){
    return vacancyNo;
 }
public void setVacancyNumber(int vacancyNo){
    this.vacancyNo=vacancyNo;
  }
public String getDesignation(){
    return designation;
  }
  public void setDesignation(String designation){
    this.designation=designation;
  }
  public    String    getJobType(){
    return jobType;
}
public void setJobType(String jobType){
    this.jobType=jobType;
  }
  public void displayInfo(){
    System.out.println("Vacancy  no:-"+getVacancyNumber());
     System.out.println("Designation:-"+getDesignation());
     System.out.println("Job Type:-"+getJobType());


    }
}
```

## II. FullTImeStaffHire

```
/**
 * Write a description of class FullTimeStaffHire here.
 *
 * @author (Anish Sherchan)
 * @version (0.001)
 */
public class FullTimeStaffHire extends StaffHire
{
    private    int    salary;
    private int workingHour;
    private   String   staffName;
    private   String   joinigDate;
    private  String  qualification;
    private  String  appointedBy;
    private boolean joined;
    public FullTimeStaffHire(int vacancyNo,String jobType,String designation,int salary,int
workingHour)
    {
        super(vacancyNo,designation,jobType);
        this.workingHour=workingHour;
        this.salary=salary;
        staffName="";

        joinigDate="";

        qualification="";

        appointedBy="";

        joined=false;


    }

    public int getsalary(){

       return salary;

   }

   public    int    getworkingHour(){

       return workingHour;

   }

   public String getstaffName(){

       return staffName;

   }

   public String getjoinigDate(){

       return joinigDate;

   }

   public String getqualification(){

       return qualification;

   }

   public String getappointedBy(){
        return appointedBy;
    }
    public Boolean getjoined(){
        return joined;
    }

    public   void   setsalary(int   salary){
        if(joined==false){
        this.salary=salary;
        System.out.println("Salary has been changed");
     }
     else{
       System.out.println("It is not possible to change the salary ");
     }
}
public          void          setWorkingHour(int          workingHour){
        this.workingHour=workingHour;
        }
    public void fullhire ( String staffName, String joinigDate, String qualification, String
appointedBy){
          if(joined==false){
          this.staffName=staffName;
          this.joinigDate=joinigDate;
          this.qualification=qualification;
          this.appointedBy=appointedBy;
          System.out.println("Staff has been hired");
          joined=true;
         }
        else{
           System.out.println( getstaffName() + " has already been hired on date
"+getjoinigDate());
         }
    }
        public void displayInfo(){
           super.displayInfo();
        if(joined==true){
           System.out.println("__________________________");
           System.out.println("Staff Name = " + staffName);
           System.out.println("Joined Date = " + joinigDate);
           System.out.println("Salary = " + salary);
           System.out.println("Working Hour= " + workingHour);
           System.out.println("Qualification = " + qualification );
           System.out.println("Appointed By = " + appointedBy);
           System.out.println("__________________________");
        }
        else{
            System.out.println("Staff has not been hired");
        }
    }


     }
```

## III. ParttimeStaffHire

```
/**
 * Write a description of class PartTimeStaffHire here.
 *
 * @author (Anish Sherchan)
 * @version (0.001)
 */
public class PartTimeStaffHire extends StaffHire
{
   private   int   workingHour;
   private  int  wagePerHour;
   private  String  staffName;
   private String joining_Date;
   private  String  qulafication;
   private String appointedBy;
   private String shifts;
```

```
  private    boolean    joined;

  private boolean terminated;


  public  PartTimeStaffHire(int  vacancyNo,  String  designation,  String  jobType,int
workingHour, int wagePerHour, String shifts)

  {

      super(vacancyNo,      designation,      jobType);

      this.workingHour=workingHour;

      this.wagePerHour=wagePerHour;

      this.shifts=shifts;

      staffName="";

      joining_Date="";

      qulafication="";

      appointedBy="";

      joined=       false;

      terminated=false;

  }

  public    int    getworkingHour(){

     return workingHour;

  }

  public   int   getwagePerHour(){

     return wagePerHour;

  }
  public String getstaffName(){
      return staffName;
  }
  public     String     getjoining_Date(){
      return joining_Date;
  }
  public     String     getqulafication(){
      return qulafication;
  }
  public     String     getappointedBy(){
      return appointedBy;
  }
  private    String    getshifts(){
      return shifts;
   }
  private Boolean getjoined(){
      return joined;
  }
  private    Boolean    getterminated(){
      return terminated;
  }

  public void setshifts( String shifts ){
```

```
 if(joined==false)
 {
     this.shifts=shifts;
 System.out.println("Shift has been changed.");
}
else
{
    System.out.println("Shift cannot be changed ");
}
}

public void setWagesPerHour(int wagePerHour){
if                                        (joined==false){
this.wagePerHour=wagePerHour;
System.out.println("Wages Per Hour has been changed.");
}
else{
System.out.println("Wages per hour cannot be changed");
}
}
public void setWorkingHour(int workingHour){
if                                    (joined==false){
this.workingHour=workingHour;
```

```
    System.out.println("Working Hour has been changed.");
    }
    }

   public void partTimehire( String staffName, String joining_Date, String qulafication,
String appointedBy){
       if(joined==false){
           this.staffName=staffName;
           this.joining_Date=joining_Date;
           this.qulafication=qulafication;
           this.appointedBy=appointedBy;
           joined=true;
           terminated=false;
           System.out.println(staffName+"Staff has been hired on"+getjoining_Date() );
        }
        else{
            System.out.println( getstaffName() + " has already been hired "+getjoining_Date()
+" by " +getappointedBy());
        }
   }

   public  void  terminate(){
       if( terminated==true){
           System.out.println("The Staff's record has already been terminated.");
        }
        else{
           System.out.println( staffName + " has been terminated.");
           this.staffName="";
           this.joining_Date="";
           this.qulafication="";
           this.appointedBy="";
           joined=false;
           terminated=true;
        }
   }

   public void displayInfo(){
       super.displayInfo();
       if(joined==true){
       System.out.println("___________________________");
           System.out.println("Staff       Name      =      "      +       staffName);
           System.out.println("Joined      Date      =      "      +      joining_Date);
           System.out.println("Working     Hours     =      "      +      workingHour);
           System.out.println("Wage    Per    Hour    =    "    +    wagePerHour);
           System.out.println("Income Per Day= " + (workingHour*wagePerHour));
            System.out.println("Qualification = " + qulafication );
            System.out.println("Appointed  By  =  "  +

            appointedBy); System.out.println("_______________");

        }

        else{

            System.out.println("Staff has not been hired yet.");

        }

    }

 }
```