LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

CS4001NI Programming

Assessment Weightage & Type

50% Individual Coursework

2019-20 Autumn

Student Name: Aashman Uprety

London Met ID: 19031231

College ID: NP01NT4A190066

Assignment Due Date: 2020/06/05

Assignment Submission Date: 2020/06/05

Word Count: 5343

Table of Contents

Table of figures

List of tables

1. Introduction

The project has four classes namely StaffHire, FullTimeStaffHire, PartTimeStaffHire and INGNepal where StaffHire is the parent class. Class INGNepal is made to make a GUI for the program where we have implemented actionlistener.

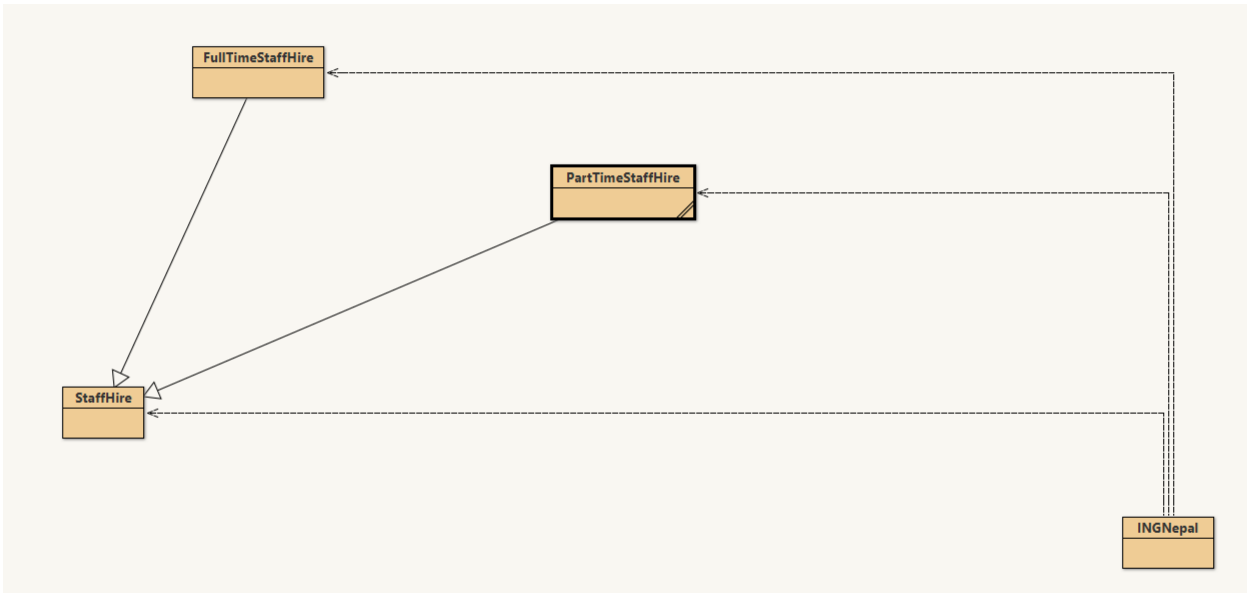
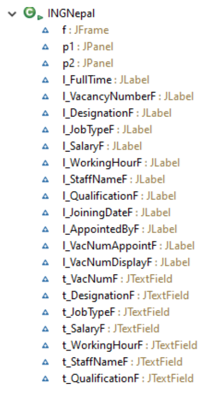


Figure 1: Overall Class Diagram for the project

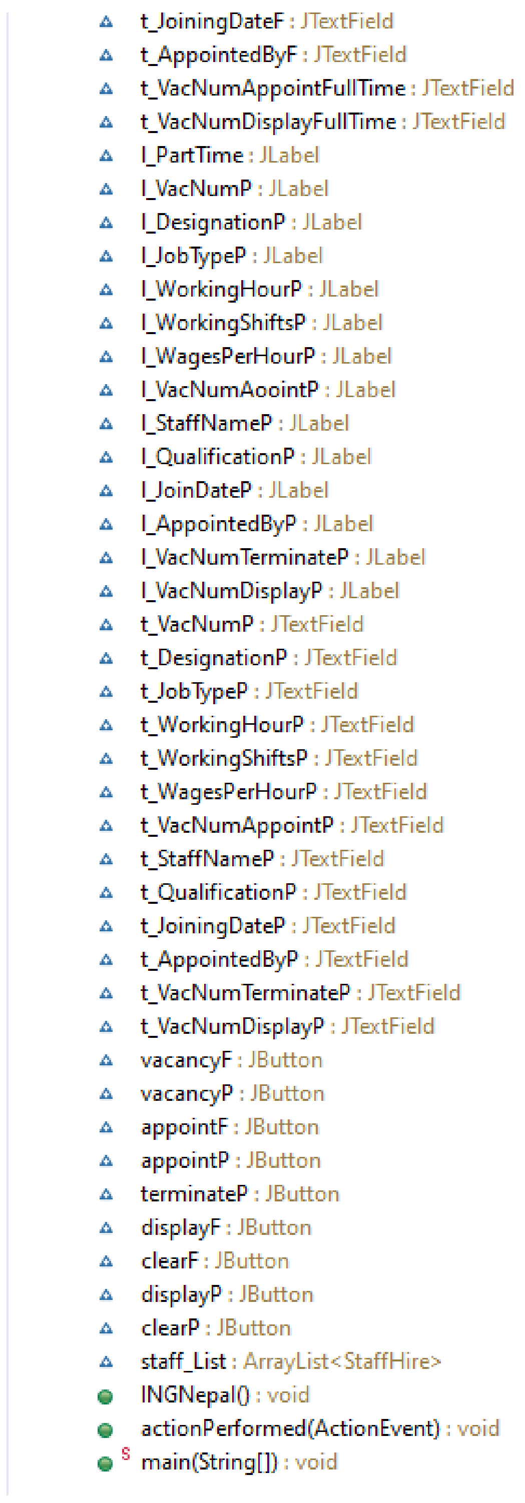
2. Class Diagram

2.1. Class Diagram for INGNepal class



```
INGNepal
    f : JFrame
    p1 : JPanel
    p2 : JPanel
    l_FullTime : JLabel
    l_VacancyNumberF : JLabel
    l_DesignationF : JLabel
    l_JobTypeF : JLabel
    l_SalaryF : JLabel
    l_WorkingHourF : JLabel
    l_StaffNameF : JLabel
    l_QualificationF : JLabel
    l_JoiningDateF : JLabel
    l_AppointedByF : JLabel
    l_VacNumAppointF : JLabel
    l_VacNumDisplayF : JLabel
    t_VacNumF : JTextField
    t_DesignationF : JTextField
    t_JobTypeF : JTextField
    t_SalaryF : JTextField
    t_WorkingHourF : JTextField
    t_StaffNameF : JTextField
    t_QualificationF : JTextField
```

Figure 2: Class Diagram for INGNepal(PART-1)



```
t_JoiningDateF : JTextField
t_AppointedByF : JTextField
t_VacNumAppointFullTime : JTextField
t_VacNumDisplayFullTime : JTextField
l_PartTime : JLabel
l_VacNumP : JLabel
l_DesignationP : JLabel
l_JobTypeP : JLabel
l_WorkingHourP : JLabel
l_WorkingShiftsP : JLabel
l_WagesPerHourP : JLabel
l_VacNumAoointP : JLabel
l_StaffNameP : JLabel
l_QualificationP : JLabel
l_JoinDateP : JLabel
l_AppointedByP : JLabel
l_VacNumTerminateP : JLabel
l_VacNumDisplayP : JLabel
t_VacNumP : JTextField
t_DesignationP : JTextField
t_JobTypeP : JTextField
t_WorkingHourP : JTextField
t_WorkingShiftsP : JTextField
t_WagesPerHourP : JTextField
t_VacNumAppointP : JTextField
t_StaffNameP : JTextField
t_QualificationP : JTextField
t_JoiningDateP : JTextField
t_AppointedByP : JTextField
t_VacNumTerminateP : JTextField
t_VacNumDisplayP : JTextField
vacancyF : JButton
vacancyP : JButton
appointF : JButton
appointP : JButton
terminateP : JButton
displayF : JButton
clearF : JButton
displayP : JButton
clearP : JButton
staff_List : ArrayList<StaffHire>
INGNepal() : void
actionPerformed(ActionEvent) : void
S main(String[]) : void
```

Figure 3: Class Diagram for INGNepal class(PART-2)

3. Pseudocode

3.1. Pseudocode for INGNepal class

```
    CLASS INGNepal IMPLEMENTS ActionListener INTERFACE

 DECLARE global variables

 CREATE ARRAYLIST staff_List having datatype StaffHire

 PUBLIC PROCEDURE INGNepal()

    CREATE new JFRAME frame having name “Staff Hire”

    SET LAYOUT for frame to null

    ADD frame to a CONTAINER container


    FOR FULL TIME STAFF

    CREATE new JPanel   p2

    SET BOUNDS for p2(5,5,975,410)

    SET LAYOUT for p2 to null

    SET BORDER for p2(BorderFactory.createLineBorder(Color.BLACK))

    ADD p2 to the container



   CREATE new JLabel l_ FullTime having name "**For Full Time Staffs**"

   SET FONTS for label_FullTime(new Font("Ariel", Font.PLAIN,35))

   SET BOUNDS for label_ FullTime (300,10,700,40)

   ADD l_FullTime to p2
```

```
Similarly CREATE JLABEL for l_VacancyNumberF,
l_DesignationF, l_SalaryF, l_WorkingHourF, l_JobTypeF,
l_StaffNameF, l_QualificationF, l_JoiningDateF, l_AppointedByF,
l_VacNumbF,l_VacNumAppointF, l_VacNumDisplayF
SET BOUNDS for these labels
ADD these labels to p2


 CREATE new JTextField t_VacNumF
 SET BOUNDS for JTextField t_VacNumbF(180,60,150,25)
 ADD JTextField text_VacancyNumberF to p2

 Similarly CREATE JTextField for t_DesignationF, t_JobTypeF, t_SalaryF,
 t_WorkingHourF, t_StaffNameF, t_QualificationF, t_JoiningDateF,
 t_AppointedByF, t_VacNumbAppointF, t_VacNumDisplayF
 SET BOUNDS for these textfields
 ADD these textfields to p2


 CREATE new JButton vacancyF having name “Add Vacancy”
 SET BOUNDS for vacancyF (5,240,325,50)
 SET FONTS for add_VacancyFullTime (new Font("Ariel", Font.PLAIN,25))
```

```
ADD ACTIONLISTENER to this button vacancyF

ADD vacancyF to p2


Similarly CREATE buttons for appointF, displaF, clearF

SET BOUNDS for these buttons

SET FONTS for these buttons

ADD ACTIONLISTENER to these buttons

ADD these buttons to p2


FOR PART TIME STAFF

CREATE new JPanel   p1

SET BOUNDS for p1(5,420,975,535)

SET LAYOUT for p1 to null

SET BORDER for p1(BorderFactory.createLineBorder(Color.BLACK))

ADD p1 to the container


CREATE new JLabel l_ PartTime having name "**For Part Time Staff**"


SET FONTS for l_PartTime(new Font("Ariel", Font.PLAIN,35))
SET BOUNDS for label_ PartTime (300,10,700,40)
ADD l_PartTime to p1

Similarly CREATE JLabel for l_VacNumbP,
l_DesignationP, l_WagesPerHourP, l_WorkingHourP, l_JobTypeP,
l_StaffNameP, l_QualificationP, l_WorkingShiftsP, l_JoiningDateP,
l_VacNumbAppointP, l_VacNumTerminateP,
l_VacNumbDisplayP, l_AppointedByP,
SET BOUNDS for these labels
ADD these labels to p1

CREATE new JTextField t_VacNumP
SET BOUNDS for JTextField text_VacNumP(180,60,150,25)
ADD JTextField t_VacNumbP to p1

Similarly CREATE JTextField for t_DesignationP, t_JobTypeP, t_WorkingHourP,
t_WagesPerHourP, t_WorkingShiftsP, t_VacancyNumberAppointP,
t_StaffNameP, t_QualificationP, t_JoiningDateP, t_AppointedByP
t_VacancyNumberTerminateP, t_VacancyNumberDisplayP
SET BOUNDS for these textfields
ADD these textfields to p1
```

```
CREATE new JButton vacancyP having name "Add Vacancy"
SET BOUNDS for vacancyP (5,275,325,50)
SET FONTS for vacancyP (new Font("Ariel", Font.PLAIN,25))
ADD ACTIONLISTENER to this button vcancyP
ADD vacancyP to p1


Similarly CREATE buttons for appointP, terminateP, displayP,
clearP
SET BOUNDS for these buttons
SET FONTS for these buttons
ADD ACTIONLISTENER to these buttons
ADD these buttons to p1


SET SIZE FOR frame (1000,1000)
SET VISIBLE for frame to true
```

```
    END PROCEDURE




    PUBLIC PROCEDURE actionPerformed(ActionEvent click)
        IF add_VacancyFullTime button is clicked
            TRY

     IF t_VacNumF or t_DesignationF or t_JobTypeF or t_SalaryF or t_WorkingHourF is EMPTY
          THROW NEW NumberFormatException with a message indicating text field is empty
     ENDIF

     CREATE a CHAR ARRAY char[]vac and STORE the input from text_VacancyNumberF
                    FOR (int i=0;i<vac.length;i++)
                        IF EVERYCHARACTER is digit
                   CONTINUE
                        ENDIF

             ELSE
                 THROW NEW NumberFormatException indicating the input value for vacancy number is invalid
             ELSEIF
                    ENDFOR

                DO SAME for checking the validity of salary and working hour

     CONVERT INPUT FROM t_VacNumF to an INT value and assign it to a

     VARIABLE vacancy_Number

             CONVERT INPUT FROM t_WorkingHourF to an INT value and assign it to a
     VARIABLE working_Hour

     CONVERT INPUT FROM t_SalaryF to an INT value and assign it to a
     VARIABLE salary

             DECLARE String designation equals to TEXT from t_DesignationF
             DECLARE String job_Type equals to TEXT from t_JobTypeF
      DECLARE boolean check is equals to false

      IF vacancancyNumber is equals to 0 or greater than 10000000
        THROW NEW NUMBER FORMAT EXCEPTION “It sounds unnatural for the
 vacancy number to be 0 or more than 1000000, isn't it?”
```

```
    ENDIF


          IF workingHour is smaller than 1 or greater than 24
              THROW NEW NUMBER FORMAT EXCEPTION "How can you work more
than 24 hours a day? Please enter valid working hour."
          ENDIF


          IF salary is less than 1000 or greater than 10000000
              THROW NEW NUMBER FORMAT EXCEPTION “The staff can have a
minimum salary of 1000 and maximum of 1000000. So, please check and enter.”
          ENDIF




          FOR (StaffHire staff:staff_List)
      IF staff instanceof FullTimeStaffHire
        CAST object of StaffHire as FullTimeStaffHire
 FullTimeStaffHire full is equals to (FullTimeStaffHire)staff
          IF (full.getVacancyNumber() is equals to vacancyNumber )
    SHOW JPANE MESSAGE “This vacancy number is already associated with another
full time staff. So, please add another vacancy number."


    SET check equals to true
```

```
            ENDIF
           ENDIF
         ENDFOR

      IF check is equals to false
         CREATE A NEW OBJECT OF FULL TIME STAFF HIRE with parameters
         (vacancyNumber,designation,jobType,salary,workingHour)



         ADD THIS OBJECT to the staff_List ARRAY LIST
         SHOW JPANE MESSAGE “A full time vacancy with vacancy number "" has been
added”

      ENDIF

            CATCH NUMBER FORMAT EXCEPTION
       SHOW CORRESPONDING EXCEPTION MESSAGE


         ENDIF

         IF appointF button is clicked
            TRY
```

```
IF INPUT from t_VacNumAppointF or t_StaffNameF or
t_QualificationF or t_JoiningDateF or t_AppointedByF is EMPTY

    THROW NEW NumberFormatException with a message indicating text field is empty

    IF displayP button is clicked
        TRY
    IF INPUT from t_VacNumDisplayP.getText() is EMPTY

            THROW NEW NumberFormatException with a message indictaing enter a vacancy number
    ENDIF

    IF vacancy_Number is zero or greater than 1000000
            THROW NEW NumberFormatException "The vacancy number cannot be zero or more than 1000000"
    ENDIF

    FOR (StaffHire staff:staff_List)
     IF staff instance of PartTimeStaffHire
    CAST object of PartTimeStaffHire as PartTimeStaffHire
```

```
PartTimeStaffHire part is equal to (PartTimeStaffHire)staff

        IF (part.getVacancyNumber() is equals to vacancyNumber)
    SHOW JPANE MESSAGE "Please check another window."
    SET check equals to true
    ENDIF
        ENDIF
    ENDFOR

IF clearP button is clicked
    set all JTextField to blank
```

4. Method Descriptions

4.1 ActionPerformed Method

Method signature :- public void actionPerformed(ActionEvent click)

The actionPerformed() method is called or invoked automatically whenever the component associated with it is clicked. As we have made a GUI for our class, we want some tasks to happen whenever the user clicks on various JButtons that we have made in our program. As we have implemeneted ActionListener interface in our INGNepal class we can register the component with the Listener and then override the actionPerformed() method. Furthermore, we can also use the anonymous class to implement the actionListener.

4.2 Main Method

Method signature :- public static void main(String args[])

In java, a main method is backbone to the program. So, we have made a main method at the last of the code making the object of INGNepal as “obj” and called the object. Public in the method signature indicates that the java runtime can execute this method. Static means JVM can load the class into memory and call the method. Void means main method doesnot return anything. String args[] means it is a java command line arguments. We could not have been able to access our GUI without making a main method.

4.3 getText() method

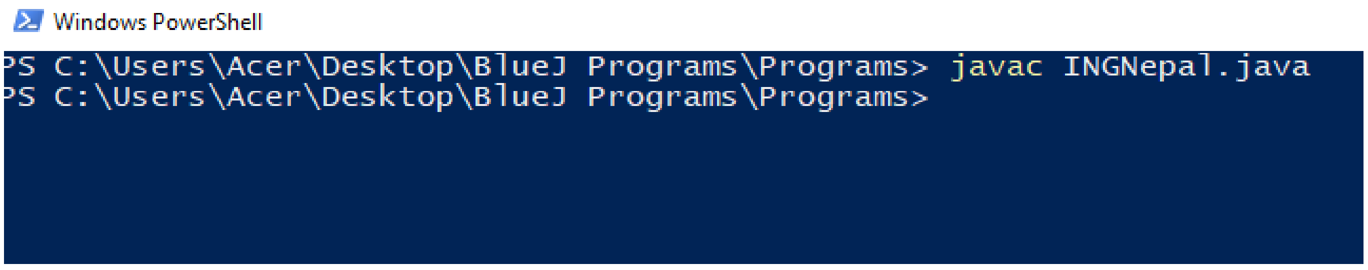
We have used getText() method in various parts of our program in order to extract the inputs that the user has given in the JTextFields area of the GUI and stire it in certain variables. This makes us easy in exception handling and controlling the flow of program as we come to know what has been inputted in the respective text fields.

5. Testing Section

5.1 Test 1

Test No.	1
Objective:	To test the program can be compiled and run using command prompt
Action:	1. The class was compiled using javac INGNepal.java 2. The compiled class was run using java INGNepal
Expected Result	The program should run and open a GUI
Actual Result	The program was run, and GUI was opened
Conclusion	The test was successful

Table 1: Test 1



```
Windows PowerShell
PS C:\Users\Acer\Desktop\BlueJ Programs\Programs> javac INGNepal.java
PS C:\Users\Acer\Desktop\BlueJ Programs\Programs>
```

Figure 4: Screenshot for test 1

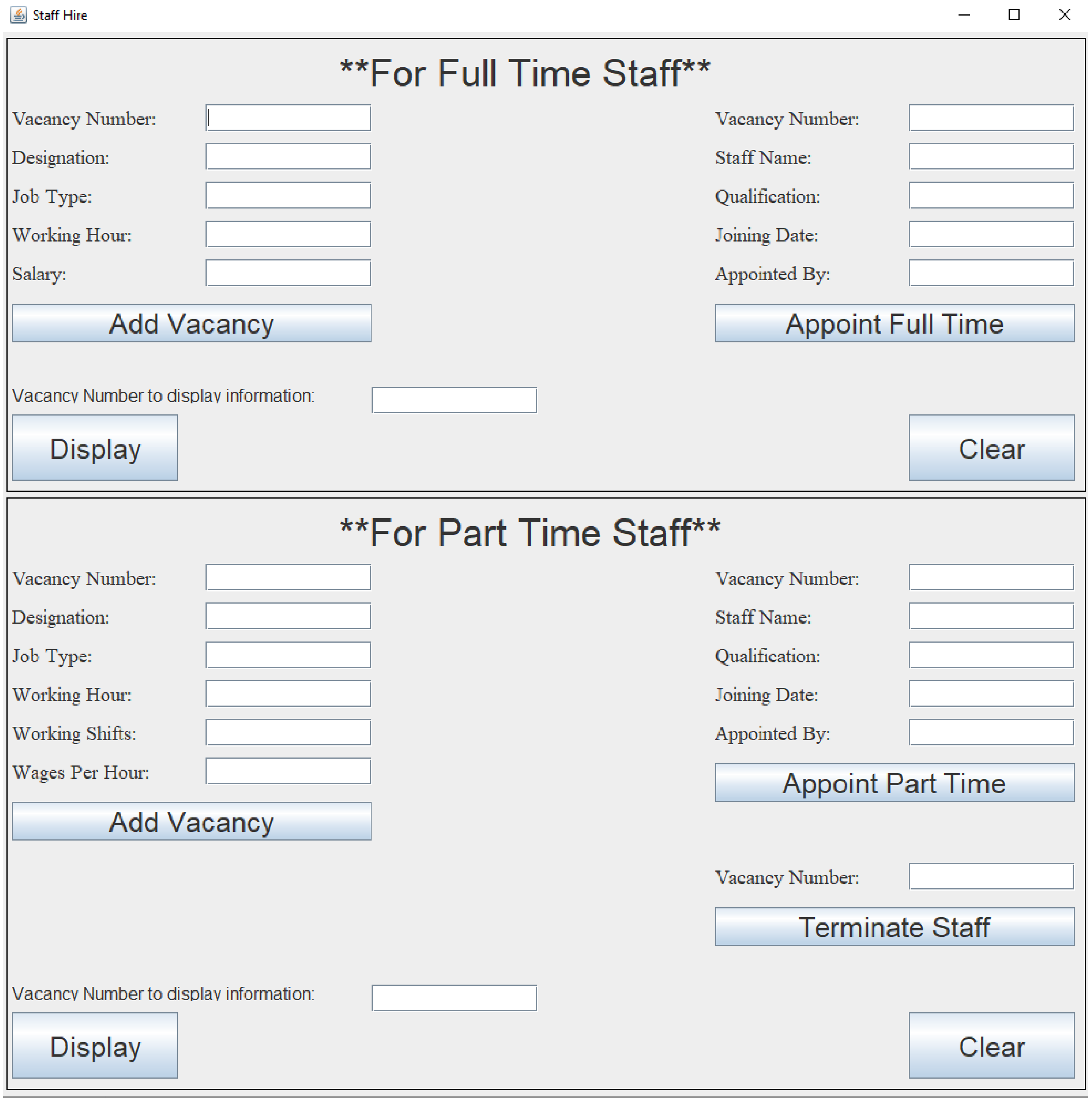


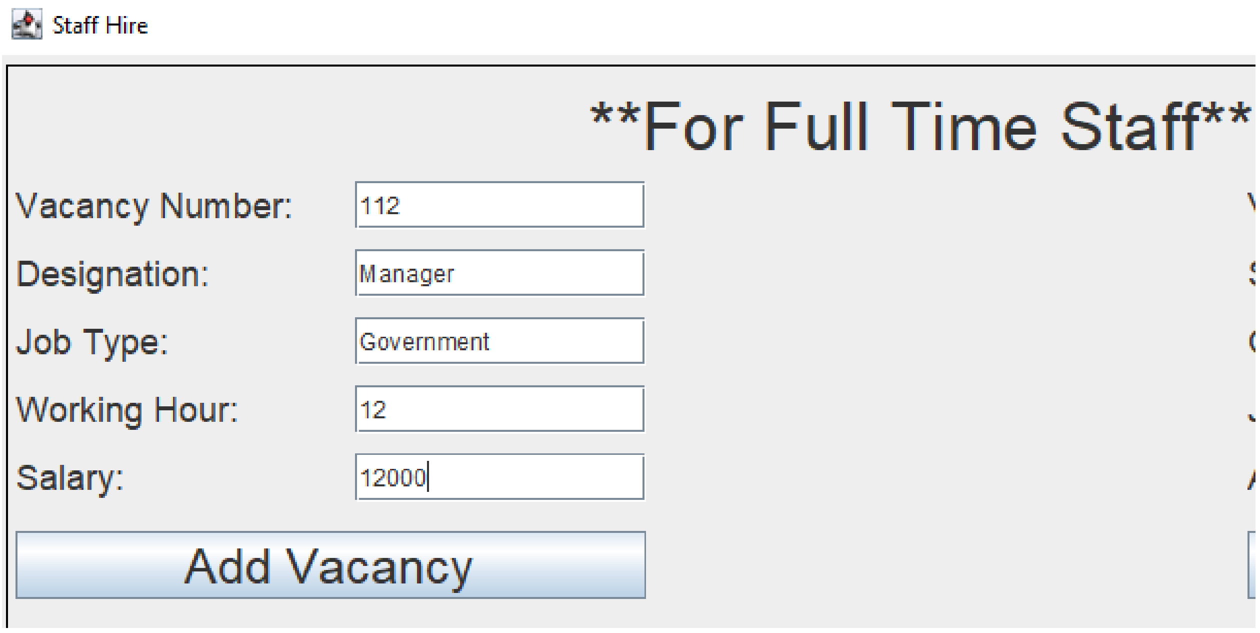
Figure 5: Screenshot for test 1

5.2 Test 2

5.2.1 Test 2(a)

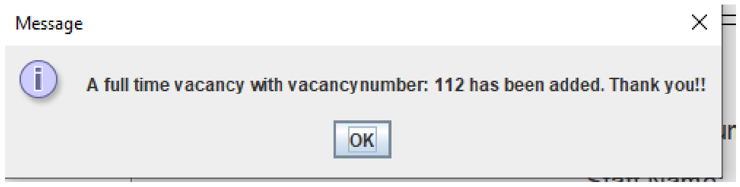
Test No.	2(a)
Objective:	To add vacancy for full time
Action:	1. All the text field of full time part time staff hire were filled with respective values. 2. Add Vacancy button was pressed.
Expected Result	The program should run and a dialog box should popup indicating the vacancy has been added.
Actual Result	The program was run, and dialog box was opened.
Conclusion	The test was successful

Table 2: Test 2(a)



Vacancy Number: 112
Designation: Manager
Job Type: Government
Working Hour: 12
Salary: 12000

Figure 6: Screenshot for test 2(a)



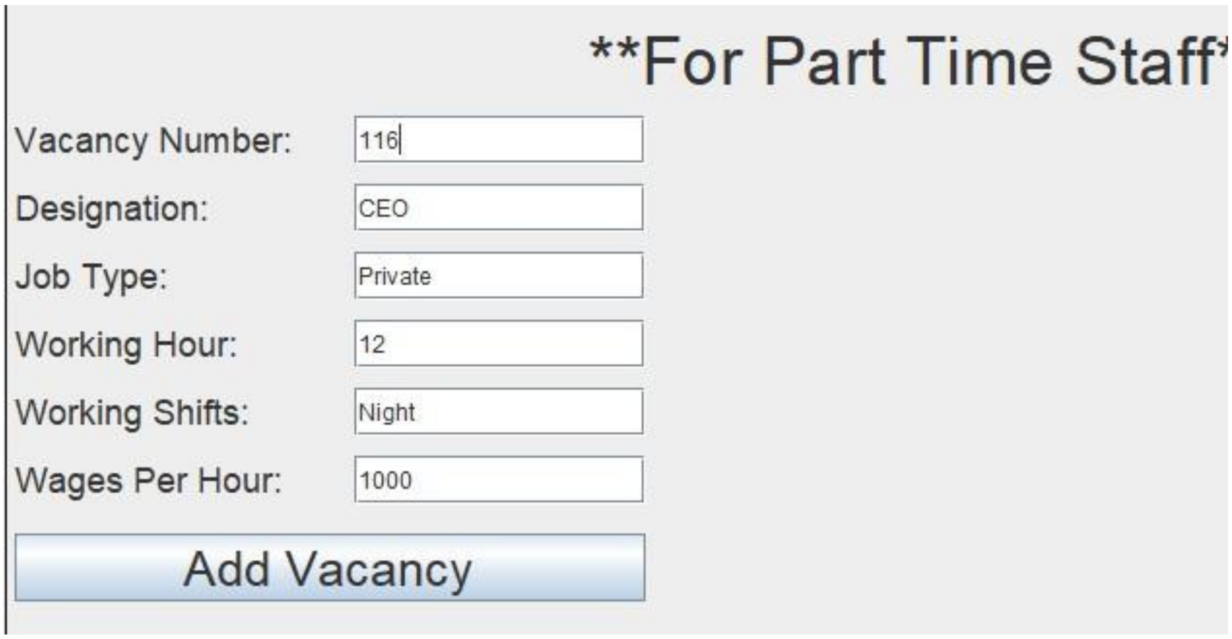
A full time vacancy with vacancynumber: 112 has been added. Thank you!!

Figure 7: Screenshot for test 2(a)

5.2.2 Test 2(b)

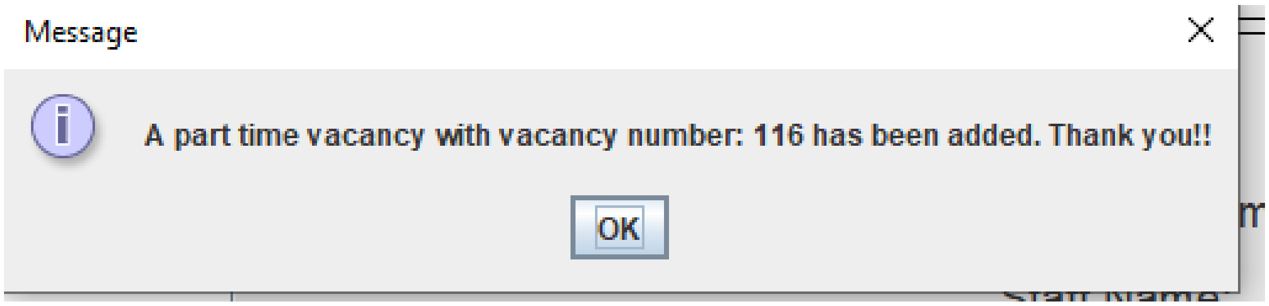
Test No.	2(b)
Objective:	To add vacancy for part time
Action:	1. All the text field of part time part time staff hire were filled with respective values. 2. Add Vacancy button was pressed.
Expected Result	The program should run and a dialog box should popup indicating the vacancy has been added.
Actual Result	The program was run, and dialog box was opened.
Conclusion	The test was successful

Table 3: Test 2(b)


**For Part Time Staff

Vacancy Number:	116
Designation:	CEO
Job Type:	Private
Working Hour:	12
Working Shifts:	Night
Wages Per Hour:	1000

Figure 8: Screenshot for test 2(b)


Message

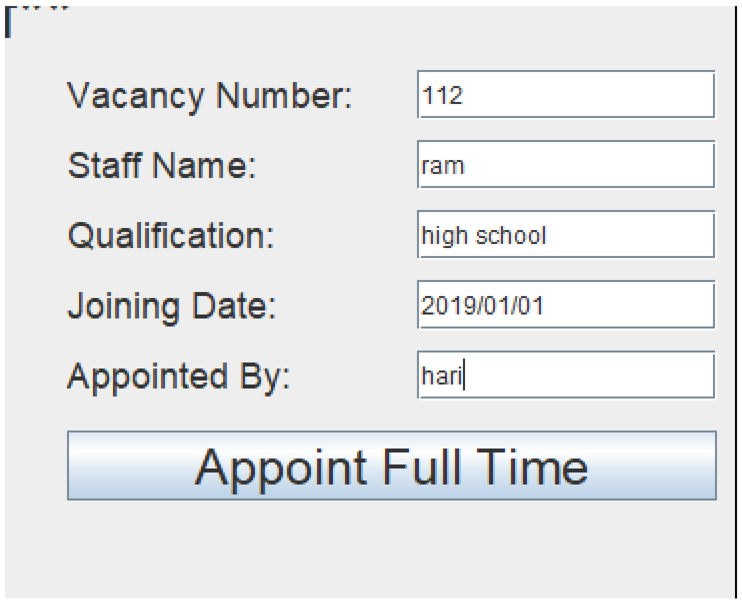
A part time vacancy with vacancy number: 116 has been added. Thank you!!

Figure 9: Screenshot for test 2(b)

5.2.3 Test 2(c)

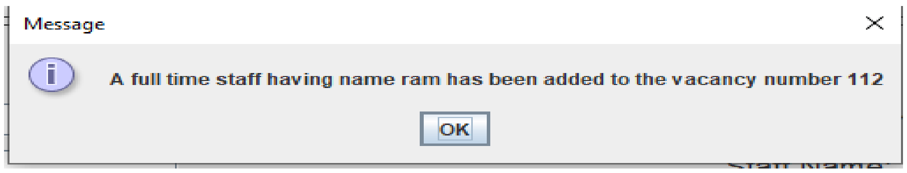
Test No.	2(c)
Objective:	To appoint full time
Action:	1. All the text field of full staff hire were filled with respective values. 2. Appoint button was pressed.
Expected Result	The program should run and a dialog box should popup indicating the vacancy has been appointed.
Actual Result	The program was run, and dialog box was opened.
Conclusion	The test was successful

Table 4: Test 2(c)



Vacancy Number: 112
Staff Name: ram
Qualification: high school
Joining Date: 2019/01/01
Appointed By: hari

Figure 10: Screenshot for test 2(c)



Message

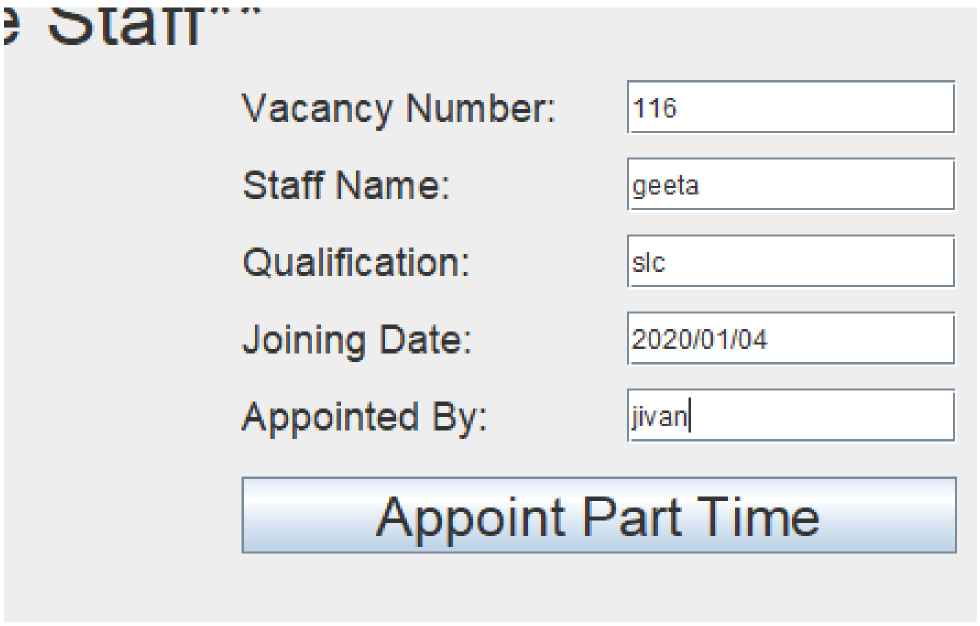
A full time staff having name ram has been added to the vacancy number 112

Figure 11: Screenshot for test 2(c)

5.2.4 Test 2(d)

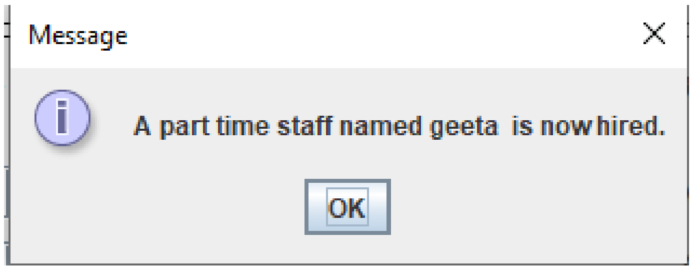
Test No.	2(d)
Objective:	To appoint part time staff
Action:	1. All the text field of part staff hire were filled with respective values. 2. Appoint button was pressed.
Expected Result	The program should run and a dialog box should popup indicating the vacancy has been appointed.
Actual Result	The program was run, and dialog box was opened.
Conclusion	The test was successful

Table 5: Test 2(d)



Vacancy Number: 116
Staff Name: geeta
Qualification: slc
Joining Date: 2020/01/04
Appointed By: jivan

Figure 12: Screenshot for test 2(d)



A part time staff named geeta is now hired.

Figure 13: Screenshot for test 2(d)

.

5.2.5 Test 2(e)

Test No.	2(e)
Objective:	To terminate part time staff
Action:	1. A hired part time staff vacancy number was entered 2. Terminate button was pressed.
Expected Result	The program should run and a dialog box should popup indicating the staff has been terminated.
Actual Result	The program was run, and dialog box was opened.
Conclusion	The test was successful

Table 6: Test 2(e)

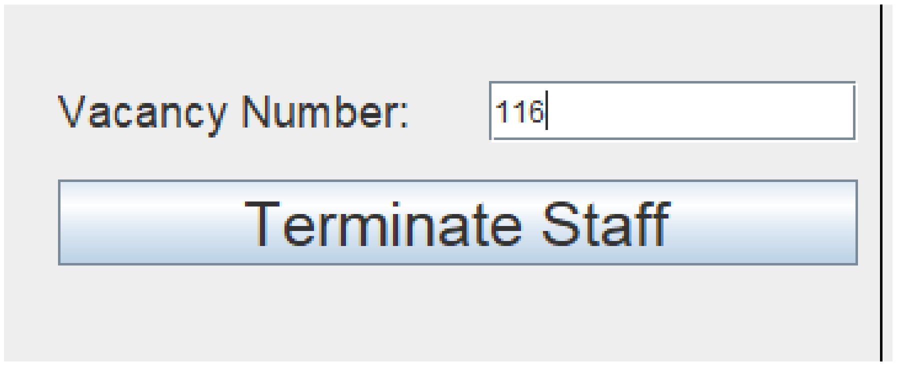
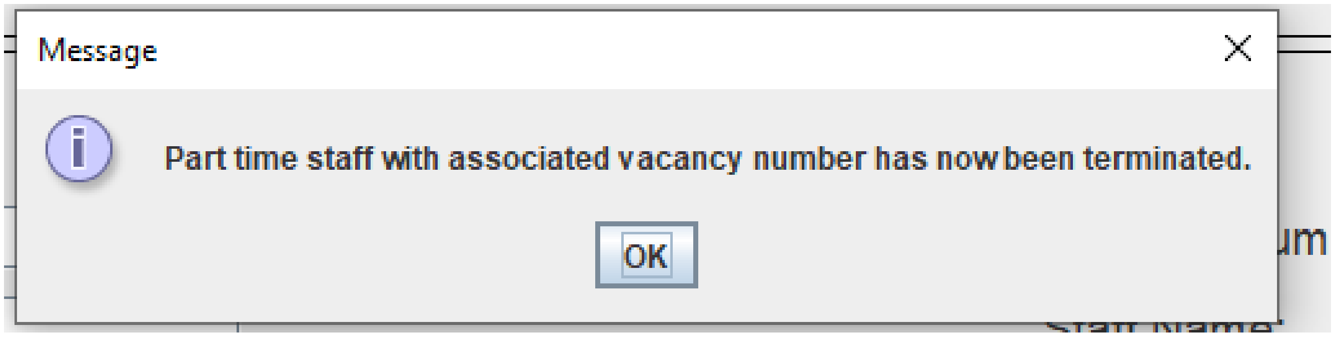


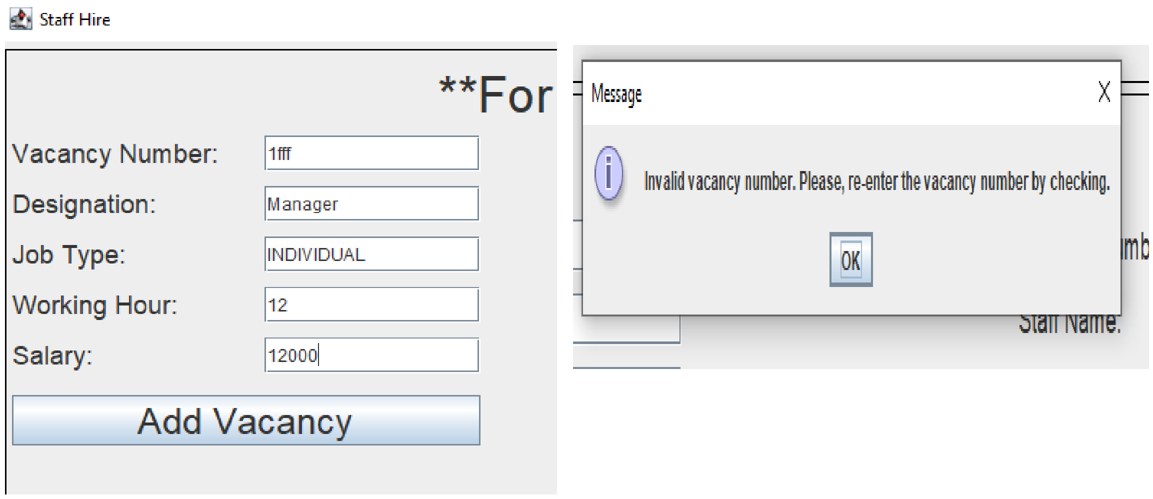
Figure 14: Screenshot for test 2(e)



Part time staff with associated vacancy number has now been terminated.

Figure 15: Screenshot for test 2(e)

5.3 Test 3


Staff Hire

**For

Vacancy Number: 1fff

Designation: Manager

Job Type: INDIVIDUAL

Working Hour: 12

Salary: 12000

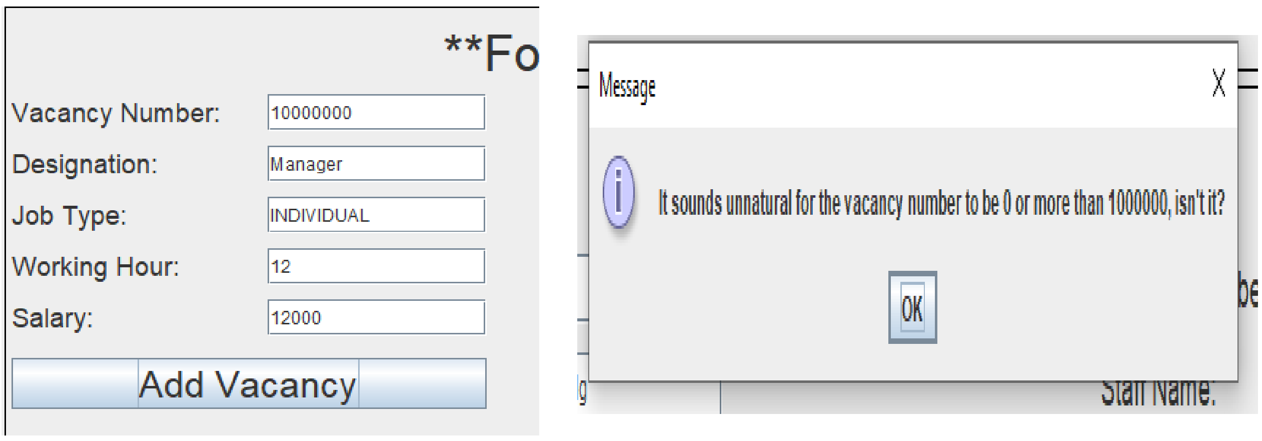
Add Vacancy

Message

Invalid vacancy number. Please, re-enter the vacancy number by checking.

OK

Figure 16: Showing errors in vacancy number


**Fo

Vacancy Number: 10000000

Designation: Manager

Job Type: INDIVIDUAL

Working Hour: 12

Salary: 12000

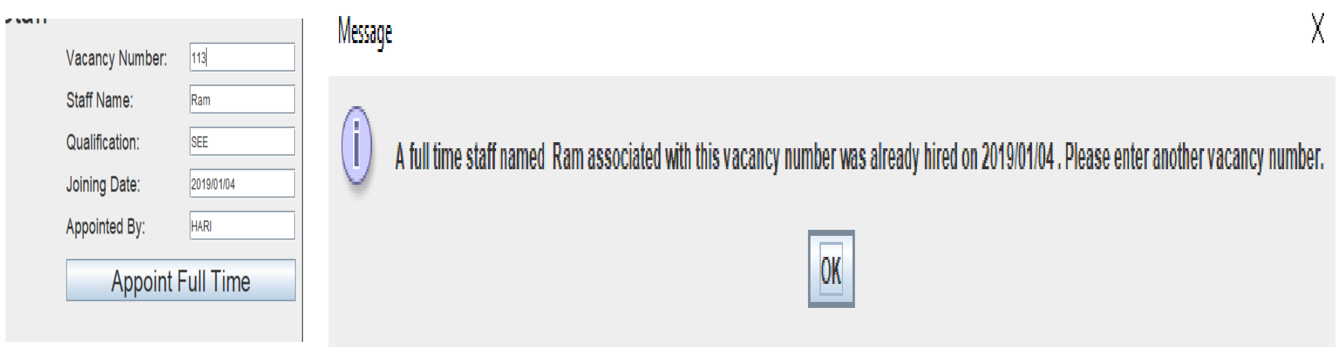
Add Vacancy

Message

It sounds unnatural for the vacancy number to be 0 or more than 1000000, isn't it?

OK

Figure 17: Showing errors in vacancy number


Vacancy Number: 113

Staff Name: Ram

Qualification: SEE

Joining Date: 2019/01/04

Appointed By: HARI

Appoint Full Time

Message

A full time staff named Ram associated with this vacancy number was already hired on 2019/01/04 . Please enter another vacancy number.

OK

Figure 18: Showing errors in vacancy number

6. Conclusion

At first, when I read the coursework I was unable to figure out from where to begin and what to do. Even after reading the question for five to six times it was really a hard task to shape my mind for starting the coursework. The hardest part for me was to do coding in the beginning. Then, I revised all the contents of lecture slide which gave me some what clear idea of writing the codes. After that, I started to write codes by making classes. It was really hard to understand the getter and setter method for me for which I consulted to Mr. Saroj Yadav, and got general idea of what it was and I even watched videos on YouTube and read some articles from W3Schools too. Moreover, because we had online classes, it was pretty hard to understand what the teacher is saying. Making a GUI using java was one of my dream and I was finally able to make it on my own and even test it. I had to encounter lots of difficulties like when I changed a single statement in code, the whole program will crash and I had to start it from the beginning. I got the general concepts of making a GUI using OOP. And, I was also able to know about how to handle errors i.e. exception handling.

7. Appendix

7.1 Code for StaffHire class

```
public class StaffHire
{
   protected int vacancyNumber; //encapsulating the sensitive data using protected
   protected String designation;
   protected String jobType;


public StaffHire(int vacancyNumber,String designation,String jobType) //making of
constructor to initialize the values
   {
      this.vacancyNumber=vacancyNumber;
      this.designation=designation;
```

```
        this.jobType=jobType;
    }

public int getVacancyNumber() { //getter method for vacancyNumber
    return vacancyNumber;
}
public void setVacancyNumber(int newVacancyNumber) { //setter method for
vacancyNumber
    this.vacancyNumber=newVacancyNumber;
}
public String getDesignation() { //getter method for designation
    return designation;
}
public void setDesignation(String newDesignation) {
    this.designation=newDesignation;
}
public String getJobType() {
    return jobType;
}
public void setJobType(String newJobType) {
    this.jobType=newJobType;
}
public void Display()
{
    System.out.println("The Vacancy number is: "+getVacancyNumber());
    System.out.println("The designation is: "+getDesignation());
    System.out.println("The job type is: "+getJobType());
}
}
```

7.2 Code for FullTimeStaffHire class

```
public class FullTimeStaffHire extends StaffHire

{

   private int salary;

   private int workingHour;

   private String staffName;

   private String joiningDate;

   private String qualification;

   private String appointedBy;

   private Boolean joined;


public FullTimeStaffHire(int salary,int workingHour,int vacancyNumber,String
designation,String jobType)

{

   super(vacancyNumber,designation,jobType);

   this.salary=salary;

   this.workingHour=workingHour;

   this.staffName="";

   this.joiningDate="";

   this.qualification="";

   this.appointedBy="";

   this.joined=false;

}


public void setSalary(int salary)

{

   this.salary=salary;

}

public int getSalary()
```

```
{
   return this.salary;
}
public void setWorkingHour(int workingHour)
{
   this.workingHour=workingHour;
}
public int getWorkingHour()
{
   return this.workingHour;
}
public void setStaffName(String staffName)
{
   this.staffName=staffName;
}
public String getStaffName()
{
   return this.staffName;
}
public void setJoiningDate(String joiningDate)
{
   this.joiningDate=joiningDate;
}
public String getJoiningDate()
{
   return this.joiningDate;
}
public void setQualification(String qualification)
```

```
{
    this.qualification=qualification;
}
public String getQualification()
{
    return this.qualification;
}
public void setAppointedBy(String appointedBy)
{
    this.appointedBy=appointedBy;
}
public String getAppointedBy()
{
    return this.appointedBy;
}
public void setJoined(Boolean joined)
{
    this.joined=joined;
}
public boolean getJoined()
{
    return this.joined;
}
public void newSalary(int salary)
{
    if (this.joined)
    {
        System.out.println("It is not possible to change the salary");
    }else
    {
        this.salary = salary;
    }
}
public void WorkingHour(int newWorkingHour)
{
    workingHour=newWorkingHour;
}
public void HireFullTimeStaff(String staffName,String joiningDate,String
qualification,String appointedBy)
{
    if (this.joined)
    {
        System.out.println("The name of the staff is :"+getStaffName());
        System.out.println("The date when the staff joined is :"+getJoiningDate());
    }else
    {
        this.staffName=staffName;
        this.joiningDate=joiningDate;
        this.qualification=qualification;
        this.appointedBy=appointedBy;
        this.joined=true;
    }

}
public void PartTimeStaff()
{
    super.Display();
    if(joined)
    {
        System.out.println("The staff name is: "+getStaffName());
        System.out.println("The salary of the staff is: "+getSalary());
        System.out.println("The working hour of the staff is: "+getWorkingHour());
        System.out.println("The joining date of staff is: "+getJoiningDate());
        System.out.println("The qualification of staff is: "+getQualification());
        System.out.println("The staff was appointed by: "+getAppointedBy());
    }



}



}
```

7.3 Code for PartTimeStaffHire class

```
public class PartTimeStaffHire extends StaffHire
{
   private int workingHour; //restricted access
   private int wagesPerHour;
   private String staffName;
   private String joiningDate;
   private String qualification;
   private String appointedBy;
   private String shifts;
   private Boolean joined;
   private Boolean terminated;
```

```
public PartTimeStaffHire(int vacancyNumber,String designation,String jobType,int
workingHour,int wagesPerHour,String shifts)
{
   super(vacancyNumber,designation,jobType); //calling from parent class constructor
using super keyword in java
   this.staffName=""; //initializing the values of instance variables using this keyword
   this.joiningDate="";
   this.qualification="";
   this.appointedBy="";
   this.joined=false;
   this.terminated=false;
   this.workingHour=workingHour;
   this.wagesPerHour=wagesPerHour;
   this.shifts=shifts;
}

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

```
{
   this.wagesPerHour=wagesPerHour;
}
public String getStaffName()
{
   return staffName;
}
public void setStaffName(String staffName)
{
   this.staffName=staffName;
}
public String getJoiningDate()
{
   return joiningDate;
}
public void setJoiningDate(String joiningDate)
{
   this.joiningDate=joiningDate;
}
public String getQualification()
{
   return qualification;
}
public void setQualification(String qualification)
{
   this.qualification=qualification;
}
public String getAppointedBy()
```

```
{
    return appointedBy;
}
public void setAppointedBy(String appointedBy)
{
    this.appointedBy=appointedBy;
}
public String getShifts()
{
    return shifts;
}
public void setShifts(String shifts)
{
    this.shifts=shifts;
}
public Boolean getJoined()
{
    return joined;
}
public void setJoined(Boolean joined)
{
    this.joined=joined;
}
public Boolean getTerminated()
{
    return terminated;
}
public void setTerminated(Boolean terminated)
```

```
{
    this.terminated=terminated;
}
public void WorkingShifts(String newWorkingShifts)
{
    if(joined)
    {
        System.out.println("It is not possible to change working hours");
    }else
    {
        setShifts(newWorkingShifts);
    }
}
public void HirePartTimeStaff(String staffName,String joiningDate,String
qualification,String appointedBy)
{
    if(joined)
    {
        System.out.println("The staff name is: "+getStaffName());
        System.out.println("The joining date of staff is: "+getJoiningDate());
    }else
    {
        this.staffName=staffName;
        this.joiningDate=joiningDate;
        this.qualification=qualification;
        this.appointedBy=appointedBy;
        this.joined=true;
        this.terminated=false;
    }
```

```
}
public void TerminateStaff()
{
    if(terminated)
    {
        System.out.println("The staff has been terminated already");
    }else
    {
        this.staffName="";
        this.joiningDate="";
        this.qualification="";
        this.appointedBy="";
        this.joined=false;
        this.terminated=true;
    }
}
public int IncomePerDay(int incomePerDay)
{
    incomePerDay=(wagesPerHour*workingHour);
    return incomePerDay;
}
public void partTimeStaffRequired()
{
    super.Display();
    if(joined)
    {
       System.out.println("The name of staff is: "+getStaffName());
```

```
        System.out.println("The per hour wage of staff is: "+getWagesPerHour());
        System.out.println("The working hour of staff is: "+getWorkingHour());
        System.out.println("The date when staff joined is: "+getJoiningDate());
        System.out.println("The qualification of staff is: "+getQualification());
        System.out.println("The staff was appointed by: "+getAppointedBy());
        System.out.println("The income of staff per day is: "+IncomePerDay(1956));
    }
}



}
```

7.4 Code for INGNepal class

```
//* Needed import elements have been imported so as to make GUI
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
public class INGNepal implements ActionListener //As we will be needing to handle the
events in our GUI so we need to implement ActionListener in our class i.e INGNepal
{
    JFrame f;

    JPanel p1,p2 ; //making two separate panels for FullTimeStaffHire and
PartTimeStaffHire

    //JLabels and JTextfields for full time staff hire:

    //JLabels::::
```

```
    JLabel
l_FullTime,l_VacancyNumberF,l_DesignationF,l_JobTypeF,l_SalaryF,l_WorkingHourF,l
_StaffNameF,l_QualificationF,l_JoiningDateF,l_AppointedByF,l_VacNumAppointF,l_Va
cNumDisplayF; //"F" at the end of variable name denotes FullTimeStaffHire

    //JTextFields::::

    JTextField
t_VacNumF,t_DesignationF,t_JobTypeF,t_SalaryF,t_WorkingHourF,t_StaffNameF,t_Qu
alificationF,t_JoiningDateF,t_AppointedByF,t_VacNumAppointFullTime,t_VacNumDispl
ayFullTime;

    //JLabels and JTextFields for part time staff hire::::

    JLabel
l_PartTime,l_VacNumP,l_DesignationP,l_JobTypeP,l_WorkingHourP,l_WorkingShiftsP,l
_WagesPerHourP,l_VacNumAoointP,l_StaffNameP,l_QualificationP,l_JoinDateP,l_App
ointedByP,l_VacNumTerminateP,l_VacNumDisplayP; //"P" at the end of variable name
denotes PartTimeStaffHirE

    JTextField
t_VacNumP,t_DesignationP,t_JobTypeP,t_WorkingHourP,t_WorkingShiftsP,t_WagesP
erHourP,t_VacNumAppointP,t_StaffNameP,t_QualificationP,t_JoiningDateP,t_Appointe
dByP,t_VacNumTerminateP,t_VacNumDisplayP;

    //JButtons needed in GUI:

    JButton
vacancyF,vacancyP,appointF,appointP,terminateP,displayF,clearF,displayP,clearP;

    /*This is the functionality in Java through which we can access the child classes by
declaring the data type of arraylist as that of parent class. As our parent class is
StaffHire in this project,
     * we directly gave the data type as StaffHire instead of doing it two times by
declaring it once FullTimeStaffHire and PartTimeStaffHire
     */

   ArrayList<StaffHire> staff_List = new ArrayList<StaffHire>();

   public void INGNepal()
   {
      f = new JFrame("Staff Hire");
      f.setLayout(null);
      Container container = f.getContentPane();

      p2 = new JPanel();
      p2.setBounds(5,5,975,410);
      p2.setLayout(null);
      p2.setBorder(BorderFactory.createLineBorder(Color.BLACK));
      container.add(p2);

      l_FullTime = new JLabel("**For Full Time Staff**");
      l_FullTime.setFont(new Font("Ariel", Font.PLAIN,35));
      l_FullTime.setBounds(300,10,700,40);
      p2.add(l_FullTime);

      l_VacancyNumberF = new JLabel("Vacancy Number:");
      l_VacancyNumberF.setFont(new Font("TimesRoman", Font.PLAIN, 18));
      l_VacancyNumberF.setBounds(5,60,170,25);
      p2.add(l_VacancyNumberF);

      t_VacNumF = new JTextField();
```

```
t_VacNumF.setBounds(180,60,150,25);
p2.add(t_VacNumF);

l_DesignationF = new JLabel("Designation:");
l_DesignationF.setFont(new Font("TimesRoman", Font.PLAIN, 18));
l_DesignationF.setBounds(5,95,170,25);
p2.add(l_DesignationF);

t_DesignationF = new JTextField();
t_DesignationF.setBounds(180,95,150,25);
p2.add(t_DesignationF);

l_JobTypeF = new JLabel("Job Type:");
l_JobTypeF.setFont(new Font("TimesRoman", Font.PLAIN, 18));
l_JobTypeF.setBounds(5,130,170,25);
p2.add(l_JobTypeF);

t_JobTypeF = new JTextField();
t_JobTypeF.setBounds(180,130,150,25);
p2.add(t_JobTypeF);

l_WorkingHourF = new JLabel("Working Hour:");
l_WorkingHourF.setFont(new Font("TimesRoman", Font.PLAIN, 18));
l_WorkingHourF.setBounds(5,165,170,25);
p2.add(l_WorkingHourF);

t_WorkingHourF = new JTextField();
t_WorkingHourF.setBounds(180,165,150,25);
```

```
p2.add(t_WorkingHourF);

l_SalaryF = new JLabel("Salary:");
l_SalaryF.setFont(new Font("TimesRoman", Font.PLAIN, 18));
l_SalaryF.setBounds(5,200,170,25);
p2.add(l_SalaryF);

t_SalaryF = new JTextField();
t_SalaryF.setBounds(180,200,150,25);
p2.add(t_SalaryF);

l_VacNumAppointF = new JLabel("Vacancy Number:");
l_VacNumAppointF.setFont(new Font("TimesRoman", Font.PLAIN, 18));
l_VacNumAppointF.setBounds(640,60,170,25);
p2.add(l_VacNumAppointF);

t_VacNumAppointFullTime = new JTextField();
t_VacNumAppointFullTime.setBounds(815,60,150,25);
p2.add(t_VacNumAppointFullTime);

l_StaffNameF = new JLabel("Staff Name:");
l_StaffNameF.setFont(new Font("TimesRoman", Font.PLAIN, 18));
l_StaffNameF.setBounds(640,95,170,25);
p2.add(l_StaffNameF);

t_StaffNameF = new JTextField();
t_StaffNameF.setBounds(815,95,150,25);
p2.add(t_StaffNameF);
```

```
l_QualificationF = new JLabel("Qualification:");
l_QualificationF.setFont(new Font("TimesRoman", Font.PLAIN, 18));
l_QualificationF.setBounds(640,130,170,25);
p2.add(l_QualificationF);

t_QualificationF = new JTextField();
t_QualificationF.setBounds(815,130,150,25);
p2.add(t_QualificationF);

l_JoiningDateF = new JLabel("Joining Date:");
l_JoiningDateF.setFont(new Font("TimesRoman", Font.PLAIN, 18));
l_JoiningDateF.setBounds(640,165,170,25);
p2.add(l_JoiningDateF);

t_JoiningDateF = new JTextField();
t_JoiningDateF.setBounds(815,165,150,25);
p2.add(t_JoiningDateF);

l_AppointedByF = new JLabel("Appointed By:");
l_AppointedByF.setFont(new Font("TimesRoman", Font.PLAIN, 18));
l_AppointedByF.setBounds(640,200,170,25);
p2.add(l_AppointedByF);

t_AppointedByF = new JTextField();
t_AppointedByF.setBounds(815,200,150,25);
p2.add(t_AppointedByF);
```

```
        JLabel l_VacNumDisplayF = new JLabel("Vacancy Number to display
information:");
        l_VacNumDisplayF.setFont(new Font("Ariel", Font.PLAIN,16));
        l_VacNumDisplayF.setBounds(5,315,325,15);
        p2.add(l_VacNumDisplayF);

        t_VacNumDisplayFullTime = new JTextField();
        t_VacNumDisplayFullTime.setBounds(330,315,150,25);
        p2.add(t_VacNumDisplayFullTime);

        vacancyF = new JButton("Add Vacancy");
        vacancyF.setBounds(5,240,325,35);
        vacancyF.setFont(new Font("Ariel", Font.PLAIN,25));
        p2.add(vacancyF);
        vacancyF.addActionListener(this);

        appointF = new JButton("Appoint Full Time");
        appointF.setBounds(640,240,325,35);
        appointF.setFont(new Font("Ariel", Font.PLAIN,25));
        p2.add(appointF);
        appointF.addActionListener(this);

        displayF = new JButton("Display");
        displayF.setBounds(5,340,150,60);
        displayF.setFont(new Font("Ariel", Font.PLAIN,25));
        p2.add(displayF);
        displayF.addActionListener(this);

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

```
clearF.setBounds(815,340,150,60);
clearF.setFont(new Font("Ariel", Font.PLAIN,25));
p2.add(clearF);
clearF.addActionListener(this);

p1 = new JPanel();
p1.setBounds(5,420,975,535);
p1.setLayout(null);
p1.setBorder(BorderFactory.createLineBorder(Color.BLACK));
container.add(p1);

l_PartTime = new JLabel("**For Part Time Staff**");
l_PartTime.setFont(new Font("Ariel", Font.PLAIN,35));
l_PartTime.setBounds(300,10,700,40);
p1.add(l_PartTime);

l_VacNumP = new JLabel("Vacancy Number:");
l_VacNumP.setFont(new Font("TimesRoman", Font.PLAIN, 18));
l_VacNumP.setBounds(5,60,170,25);
p1.add(l_VacNumP);

t_VacNumP = new JTextField();
t_VacNumP.setBounds(180,60,150,25);
p1.add(t_VacNumP);

l_DesignationP = new JLabel("Designation:");
l_DesignationP.setFont(new Font("TimesRoman", Font.PLAIN, 18));
l_DesignationP.setBounds(5,95,170,25);
```

```
p1.add(l_DesignationP);

t_DesignationP = new JTextField();
t_DesignationP.setBounds(180,95,150,25);
p1.add(t_DesignationP);

l_JobTypeP = new JLabel("Job Type:");
l_JobTypeP.setFont(new Font("TimesRoman", Font.PLAIN, 18));
l_JobTypeP.setBounds(5,130,170,25);
p1.add(l_JobTypeP);

t_JobTypeP = new JTextField();
t_JobTypeP.setBounds(180,130,150,25);
p1.add(t_JobTypeP);

l_WorkingHourP = new JLabel("Working Hour:");
l_WorkingHourP.setFont(new Font("TimesRoman", Font.PLAIN, 18));
l_WorkingHourP.setBounds(5,165,170,25);
p1.add(l_WorkingHourP);

t_WorkingHourP = new JTextField();
t_WorkingHourP.setBounds(180,165,150,25);
p1.add(t_WorkingHourP);

l_WorkingShiftsP = new JLabel("Working Shifts:");
l_WorkingShiftsP.setFont(new Font("TimesRoman", Font.PLAIN, 18));
l_WorkingShiftsP.setBounds(5,200,170,25);
p1.add(l_WorkingShiftsP);
```

```
t_WorkingShiftsP = new JTextField();
t_WorkingShiftsP.setBounds(180,200,150,25);
p1.add(t_WorkingShiftsP);

l_WagesPerHourP = new JLabel("Wages Per Hour:");
l_WagesPerHourP.setFont(new Font("TimesRoman", Font.PLAIN, 18));
l_WagesPerHourP.setBounds(5,235,170,25);
p1.add(l_WagesPerHourP);

t_WagesPerHourP = new JTextField();
t_WagesPerHourP.setBounds(180,235,150,25);
p1.add(t_WagesPerHourP);

l_VacNumAoointP = new JLabel("Vacancy Number:");
l_VacNumAoointP.setFont(new Font("TimesRoman", Font.PLAIN, 18));
l_VacNumAoointP.setBounds(640,60,170,25);
p1.add(l_VacNumAoointP);

t_VacNumAppointP = new JTextField();
t_VacNumAppointP.setBounds(815,60,150,25);
p1.add(t_VacNumAppointP);

l_StaffNameP = new JLabel("Staff Name:");
l_StaffNameP.setFont(new Font("TimesRoman", Font.PLAIN, 18));
l_StaffNameP.setBounds(640,95,170,25);
p1.add(l_StaffNameP);
```

```
t_StaffNameP = new JTextField();
t_StaffNameP.setBounds(815,95,150,25);
p1.add(t_StaffNameP);

l_QualificationP = new JLabel("Qualification:");
l_QualificationP.setFont(new Font("TimesRoman", Font.PLAIN, 18));
l_QualificationP.setBounds(640,130,170,25);
p1.add(l_QualificationP);

t_QualificationP = new JTextField();
t_QualificationP.setBounds(815,130,150,25);
p1.add(t_QualificationP);

l_JoinDateP = new JLabel("Joining Date:");
l_JoinDateP.setFont(new Font("TimesRoman", Font.PLAIN, 18));
l_JoinDateP.setBounds(640,165,170,25);
p1.add(l_JoinDateP);

t_JoiningDateP = new JTextField();
t_JoiningDateP.setBounds(815,165,150,25);
p1.add(t_JoiningDateP);

l_AppointedByP = new JLabel("Appointed By:");
l_AppointedByP.setFont(new Font("TimesRoman", Font.PLAIN, 18));
l_AppointedByP.setBounds(640,200,170,25);
p1.add(l_AppointedByP);

t_AppointedByP = new JTextField();
```

```
        t_AppointedByP.setBounds(815,200,150,25);
        p1.add(t_AppointedByP);

        l_VacNumTerminateP = new JLabel("Vacancy Number:");
        l_VacNumTerminateP.setFont(new Font("TimesRoman", Font.PLAIN, 18));
        l_VacNumTerminateP.setBounds(640,330,170,25);
        p1.add(l_VacNumTerminateP);

        t_VacNumTerminateP = new JTextField();
        t_VacNumTerminateP.setBounds(815,330,150,25);
        p1.add(t_VacNumTerminateP);

        JLabel l_VacNumDisplayP = new JLabel("Vacancy Number to display 
information:");
        l_VacNumDisplayP.setFont(new Font("Ariel", Font.PLAIN,16));
        l_VacNumDisplayP.setBounds(5,440,325,15);
        p1.add(l_VacNumDisplayP);

        t_VacNumDisplayP = new JTextField();
        t_VacNumDisplayP.setBounds(330,440,150,25);
        p1.add(t_VacNumDisplayP);

        vacancyP = new JButton("Add Vacancy");
        vacancyP.setBounds(5,275,325,35);
        vacancyP.setFont(new Font("Ariel", Font.PLAIN,25));
        p1.add(vacancyP);

        appointP = new JButton("Appoint Part Time");
        appointP.setBounds(640,240,325,35);
        appointP.setFont(new Font("Ariel", Font.PLAIN,25));
        p1.add(appointP);

        terminateP = new JButton("Terminate Staff");
        terminateP.setBounds(640,370,325,35);
        terminateP.setFont(new Font("Ariel", Font.PLAIN,25));
        p1.add(terminateP);

        displayP = new JButton("Display");
        displayP.setBounds(5,465,150,60);
        displayP.setFont(new Font("Ariel", Font.PLAIN,25));
        p1.add(displayP);

        clearP = new JButton("Clear");
        clearP.setBounds(815,465,150,60);
        clearP.setFont(new Font("Ariel", Font.PLAIN,25));
        p1.add(clearP);

        f.setSize(1000,1000);
        f.setVisible(true);

        vacancyP.addActionListener(this);
        appointP.addActionListener(this);
        terminateP.addActionListener(this);
        displayP.addActionListener(this);
        clearP.addActionListener(this);
    }
    public void actionPerformed(ActionEvent click)
    {
        if(click.getSource()==vacancyF)
        {
            try
            {
                if(t_VacNumF.getText().isEmpty() || t_DesignationF.getText().isEmpty() ||
t_JobTypeF.getText().isEmpty() || t_SalaryF.getText().isEmpty() ||
t_WorkingHourF.getText().isEmpty())
                {
                    throw new NumberFormatException("The text fields are empty. Please fill
the boxes with their respective fields.");

                }

                char[]vac = t_VacNumF.getText().toCharArray();
                for(int i=0;i<vac.length;i++)
                {
                    if(Character.isDigit(vac[i]))
                    {
                        continue;
                    }

                    else
                    {
                        throw new NumberFormatException("Invalid vacancy number. Please,
re-enter the vacancy number by checking.");

                    }
                }
```

```
char[]sal = t_SalaryF.getText().toCharArray();
for(int i=0;i<sal.length;i++)
{
   if(Character.isDigit(sal[i]))
   {
      continue;
   }

   else
   {
      throw new NumberFormatException("Invalid Salary.");

   }

}

char[]work_Hour = t_WorkingHourF.getText().toCharArray();
for(int i=0;i<work_Hour.length;i++)
{
   if(Character.isDigit(work_Hour[i]))
   {
      continue;
   }

   else
   {
      throw new NumberFormatException("The working hour is invalid.");
```

```
        }
    }

    int vacancyNumber = Integer.parseInt(t_VacNumF.getText());
    String designation = t_DesignationF.getText();
    String jobType = t_JobTypeF.getText();
    int salary = Integer.parseInt(t_SalaryF.getText());
    int workingHour = Integer.parseInt(t_WorkingHourF.getText());
    boolean check = false;

    if (vacancyNumber == 0 || vacancyNumber > 1000000)
    {

        throw new NumberFormatException("It sounds unnatural for the vacancy
number to be 0 or more than 1000000, isn't it?");

    }

    if (workingHour < 1 || workingHour > 24)
    {
        throw new NumberFormatException("How can you work more than 24
hours a day? Please enter valid working hour.");

    }

    if (salary < 1000 || salary > 1000000)
    {
```

```
                throw new NumberFormatException("The staff can have a minimum salary of 1000 and maximum of 1000000. So, please check and enter.");
            }


            for (StaffHire staff:staff_List)
            {
                if (staff instanceof FullTimeStaffHire)
                {
                    FullTimeStaffHire full = (FullTimeStaffHire)staff;
                     if ((full.getVacancyNumber() == vacancyNumber ))
                    {
                        JOptionPane.showMessageDialog(f,"This vacancy number is already associated with another full time staff. So, please add another vacancy number.");
                        check = true;
                    }


                }
            }


            if (check == false)
            {
                 FullTimeStaffHire full_Time = new FullTimeStaffHire(salary,workingHour,vacancyNumber,designation,jobType);
                 staff_List.add(full_Time);
                 JOptionPane.showMessageDialog(f,"A full time vacancy with vacancynumber: "+ vacancyNumber + " has been added. Thank you!!");


            }


        }
        catch(NumberFormatException exception)
        {
            JOptionPane.showMessageDialog(f,exception.getMessage());
        }
    }



    if (click.getSource() == appointF)
    {
        try
        {
            if(t_VacNumAppointFullTime.getText().isEmpty() ||
t_StaffNameF.getText().isEmpty() || t_QualificationF.getText().isEmpty() ||
t_JoiningDateF.getText().isEmpty() || t_AppointedByF.getText().isEmpty())
            {
                throw new NumberFormatException("The text fields are empty. Please fill
the textboxes with their respective fields..");

            }

            char[]vac = t_VacNumAppointFullTime.getText().toCharArray();
            for(int i=0;i<vac.length;i++)
            {
                if(Character.isDigit(vac[i]))
                {
                    continue;
                }
```

```
          else
          {
              throw new NumberFormatException("Invalid vacancy number. Please
enter a valid vacancy number by checking.");


          }
      }


      char[]staff_Name = t_StaffNameF.getText().toCharArray();
      for(int i = 0;i<staff_Name.length;i++)
      {
          if(Character.isDigit(staff_Name[i]) == false)
          {
              continue;
          }


          else
          {
              throw new NumberFormatException("From when did names start to
contain numbers ?? Please enter a string value in staff name field.");


          }


      }


      char[]appoint = t_AppointedByF.getText().toCharArray();
      for(int i = 0;i<appoint.length;i++)
      {
          if(Character.isDigit(appoint[i]) == false)
```

```
                {
                    continue;
                }

                else
                {
                    throw new NumberFormatException("From when did names start to
contain numbers ?? Please enter a string value in appointed by field.");

                }
            }

            int vacancy_Number = Integer.parseInt(t_VacNumAppointFullTime.getText());
            String name = t_StaffNameF.getText();
            String join = t_JoiningDateF.getText();
            String qualification = t_QualificationF.getText();
            String appointed_By = t_AppointedByF.getText();
            boolean check = false;

             if (vacancy_Number == 0 || vacancy_Number > 1000000)
            {

                throw new NumberFormatException("The vacancy number cannot be zero
or more than 1000000");

            }


            for (StaffHire staff:staff_List)
            {
                if (staff instanceof FullTimeStaffHire)
                {
                    FullTimeStaffHire full = (FullTimeStaffHire)staff;
                     if ((full.getVacancyNumber() == vacancy_Number ) && (full.getJoined()
== true))
                    {
                        JOptionPane.showMessageDialog(f,"A full time staff named " +
full.getStaffName() + " associated with this vacancy number was already hired on " +
full.getJoiningDate() + " . Please enter another vacancy number.");
                        check = true;


                    }
                    if ((full.getVacancyNumber() == vacancy_Number ) && (full.getJoined()
== false))
                    {
                        full.HireFullTimeStaff(name,join,qualification,appointed_By);
                        JOptionPane.showMessageDialog(f,"A full time staff having name " +
name +" has been added to the vacancy number " + vacancy_Number);
                        check = true;



                    }


                }
            }



            if (check == false)

                {
                    JOptionPane.showMessageDialog(f,"Please enter a valid vacancy
number.");
                }



            }


            catch(NumberFormatException ex)
            {
                JOptionPane.showMessageDialog(f,ex.getMessage());
            }

        }

        if(click.getSource()==displayF)
        {
            try
            {
                if(t_VacNumDisplayFullTime.getText().isEmpty())
                {
                    throw new NumberFormatException("Please enter a vacancy number.");

                }

                char[]vac = t_VacNumDisplayFullTime.getText().toCharArray();
                for(int i=0;i<vac.length;i++)
                {
                    if(Character.isDigit(vac[i]))
                    {
```

```
                continue;
            }


            else
            {
                throw new NumberFormatException("Please enter a valid vacancy
number.");


            }
        }


        int vacancy_Number = Integer.parseInt(t_VacNumDisplayFullTime.getText());
        boolean check = false;


        for (StaffHire staff:staff_List)
        {
            if (staff instanceof FullTimeStaffHire)
            {
                FullTimeStaffHire full = (FullTimeStaffHire)staff;
                 if (full.getVacancyNumber() == vacancy_Number)
                {
                    full.Display();
                    check = true;
                    JOptionPane.showMessageDialog(f,"Information regarding the
entered vacancy number is displayed on the terminal window.");


                }


            }
```

```
                }

                if (check == false)
                {
                    JOptionPane.showMessageDialog(f,"Please enter a valid vacancy
number.");
                }

            }

            catch(NumberFormatException ex)
            {
                JOptionPane.showMessageDialog(f,ex.getMessage());
            }

        }

        if (click.getSource()==clearF)
        {

            t_VacNumF.setText("");
            t_DesignationF.setText("");
            t_JobTypeF.setText("");
            t_SalaryF.setText("");
            t_WorkingHourF.setText("");
            t_VacNumAppointFullTime.setText("");
            t_StaffNameF.setText("");
            t_QualificationF.setText("");
            t_JoiningDateF.setText("");
            t_AppointedByF.setText("");
            t_VacNumDisplayFullTime.setText("");
        }


         if(click.getSource()==vacancyP)
        {
            try
            {
                if(t_VacNumP.getText().isEmpty() || t_DesignationP.getText().isEmpty() ||
t_JobTypeP.getText().isEmpty() || t_WagesPerHourP.getText().isEmpty() ||
t_WorkingHourP.getText().isEmpty() || t_WorkingShiftsP.getText().isEmpty())
                {
                    throw new NumberFormatException("Text boxes can not be empty. Please
fill the boxes with appropriate texts.");


                }


                char[]vac = t_VacNumP.getText().toCharArray();
                for(int i=0;i<vac.length;i++)
                {
                    if(Character.isDigit(vac[i]))
                    {
                        continue;
                    }


                    else
                    {
                        throw new NumberFormatException("Invalid vacancy number.");
                    }
                }

                char[]wages = t_WagesPerHourP.getText().toCharArray();
                for(int i=0;i<wages.length;i++)
                {
                    if(Character.isDigit(wages[i]))
                    {
                        continue;
                    }

                    else
                    {
                        throw new NumberFormatException("Invalid wages per hour input
detected.");

                    }

                }

                char[]work_Hour = t_WorkingHourP.getText().toCharArray();
                for(int i=0;i<work_Hour.length;i++)
                {
                    if(Character.isDigit(work_Hour[i]))
                    {
                        continue;
                    }

                    else
```

```
                {
                    throw new NumberFormatException("Please enter a valid working
hour.");

                }
            }

            int vacancyNumber = Integer.parseInt(t_VacNumP.getText());
            String designation = t_DesignationP.getText();
            String jobType = t_JobTypeP.getText();
            int workingHour = Integer.parseInt(t_WorkingHourP.getText());
            int wagesPerHour = Integer.parseInt(t_WagesPerHourP.getText());
            String shifts = t_WorkingShiftsP.getText();
            boolean check = false;

             if (vacancyNumber == 0 || vacancyNumber > 1000000)
            {

                throw new NumberFormatException("The vacancy number cannot be zero
or more than 1000000");

            }

            if (workingHour < 1 || workingHour > 24)
            {
                throw new NumberFormatException("How can you work more than 24
hours a day as a day itself has 24 hours. The working hour per day is in range of 1-24");

            }
```

```
        if(wagesPerHour < 100 || wagesPerHour > 50000)
        {
            throw new NumberFormatException("The minimum wages per hour is 100 whereas maximum wages per hour is 50000 which makes a sense too. Re-enter wages per hour.");
        }

        for (StaffHire staff:staff_List)
        {
            if (staff instanceof PartTimeStaffHire)
            {
                PartTimeStaffHire part = (PartTimeStaffHire)staff;
                 if ((part.getVacancyNumber() == vacancyNumber ))
                {
                    JOptionPane.showMessageDialog(f,"A part time staffv with same vacancy number has already been added.So, please input another vacancy number.");
                    check = true;
                }

            }
        }

        if (check == false)
        {
            PartTimeStaffHire part_Time = new PartTimeStaffHire(vacancyNumber,designation,jobType,workingHour,wagesPerHour,shifts);
            staff_List.add(part_Time);
```

```
                JOptionPane.showMessageDialog(f,"A part time vacancy with vacancy number: " + vacancyNumber + " has been added. Thank you!!");
            }


         }


         catch(NumberFormatException exception)
         {
            JOptionPane.showMessageDialog(f,exception.getMessage());
         }
       }


       if (click.getSource() == appointP)
      {
         try
         {
            if(t_VacNumAppointP.getText().isEmpty() || t_StaffNameP.getText().isEmpty() || t_QualificationP.getText().isEmpty() || t_JoiningDateP.getText().isEmpty() || t_AppointedByP.getText().isEmpty())
            {
                throw new NumberFormatException("Empty TextBox Detected. Please make sure to enter Vacancy number, staff name, qualification,joining date and appointed by on their respective textbox.");


            }


            char[]vac = t_VacNumAppointP.getText().toCharArray();
            for(int i=0;i<vac.length;i++)
            {
                if(Character.isDigit(vac[i]))
```

```
          {
             continue;
          }

          else
          {
             throw new NumberFormatException("Please enter a valid vacancy
number.");

          }
       }

       char[]staff_Name = t_StaffNameP.getText().toCharArray();
       for(int i = 0;i<staff_Name.length;i++)
       {
          if(Character.isDigit(staff_Name[i]) == false)
          {
             continue;
          }

          else
          {
             throw new NumberFormatException("Please enter a valid staff
name.(Staff name cannot contain numeric values.)");

          }

       }
```

```
        char[]appoint = t_AppointedByP.getText().toCharArray();
        for(int i = 0;i<appoint.length;i++)
        {
          if(Character.isDigit(appoint[i]) == false)
          {
            continue;
          }

          else
          {
            throw new NumberFormatException("Please enter a valid value on
appointed by text field.(Name cannot contain numeric values.)");

          }

        }



        int vacancy_Number = Integer.parseInt(t_VacNumAppointP.getText());

         if (vacancy_Number == 0 || vacancy_Number > 1000000)
        {

          throw new NumberFormatException("The vacancy number cannot be zero
or more than 1000000");

        }

        String staffName = t_StaffNameP.getText();
```

```
            String joiningDate = t_JoiningDateP.getText();
            String qualification = t_QualificationP.getText();
            String appointedBy = t_AppointedByP.getText();
            boolean check = false;

            for (StaffHire staff:staff_List)
            {
                if (staff instanceof PartTimeStaffHire)
                {
                    PartTimeStaffHire part = (PartTimeStaffHire)staff;
                     if ((part.getVacancyNumber() == vacancy_Number ) &&
(part.getJoined() == true))
                    {
                        JOptionPane.showMessageDialog(f,"A part time staff " +
part.getStaffName() + " was already hired on " + part.getJoiningDate() + " . Please enter
another vacancy number");
                        check = true;
                    }
                    if ((part.getVacancyNumber() == vacancy_Number ) && (part.getJoined()
== false))
                    {

part.HirePartTimeStaff(staffName,joiningDate,qualification,appointedBy);
                        JOptionPane.showMessageDialog(f,"A part time staff named " +
staffName  + "  is now hired.");
                        check = true;


                    }
                }
            }

            if (check == false)
            {
                JOptionPane.showMessageDialog(f,"Please enter a valid vacancy
number.");
            }



        }

        catch(NumberFormatException ex)
        {
            JOptionPane.showMessageDialog(f,ex.getMessage());
        }

    }

    if (click.getSource() == terminateP)
    {
        try
        {
            if(t_VacNumTerminateP.getText().isEmpty())
            {
                throw new NumberFormatException("Please enter a vacancy number.");

            }
            char[]vac = t_VacNumTerminateP.getText().toCharArray();
            for(int i=0;i<vac.length;i++)
            {
                if(Character.isDigit(vac[i]))
                {
                    continue;
                }

                else
                {
                    throw new NumberFormatException("Please enter a valid vacancy
number.");

                }
            }

            int vacancy_Number = Integer.parseInt(t_VacNumTerminateP.getText());
            boolean check = false;

            for (StaffHire staff:staff_List)
            {
                if (staff instanceof PartTimeStaffHire)
                {
                    PartTimeStaffHire part = (PartTimeStaffHire)staff;
                     if ((part.getVacancyNumber() == vacancy_Number ) &&
(part.getTerminated() == true))
                    {
                    JOptionPane.showMessageDialog(f,"The part time staff associated 
with this vacancy number was already hired.");
                    check = true;
                }
                if ((part.getVacancyNumber() == vacancy_Number ) && 
(part.getTerminated() == false) && (part.getJoined() == true))
                {

                    JOptionPane.showMessageDialog(f,"Part time staff with associated 
vacancy number has now been terminated.");
                    part.TerminateStaff();
                    check = true;

                }

              }
            }



            if (check == false)
            {
              JOptionPane.showMessageDialog(f,"Please enter a valid vacancy 
number.");
            }



        }


        catch(NumberFormatException ex)
        {
            JOptionPane.showMessageDialog(f,ex.getMessage());
        }

    }

    if (click.getSource() == displayP)
    {
        try
        {
            if(t_VacNumDisplayP.getText().isEmpty())
            {
                throw new NumberFormatException("Please enter a vacancy number.");

            }

            char[]vac = t_VacNumDisplayP.getText().toCharArray();
            for(int i=0;i<vac.length;i++)
            {
                if(Character.isDigit(vac[i]))
                {
                    continue;
                }

                else
                {
                    throw new NumberFormatException("Invalid vacancy number.");

                }
            }


            int vacancy_Number = Integer.parseInt(t_VacNumDisplayP.getText());
            boolean check = false;


             if (vacancy_Number == 0 || vacancy_Number > 1000000)
            {


                throw new NumberFormatException("The vacancy number cannot be zero
    or more than 1000000");


            }


            for (StaffHire staff:staff_List)
            {
                if (staff instanceof PartTimeStaffHire)
                {
                    PartTimeStaffHire part = (PartTimeStaffHire)staff;
                     if (part.getVacancyNumber() == vacancy_Number)
                    {
                        part.Display();
                        check = true;
                        JOptionPane.showMessageDialog(f,"Please check another window.");
                    }



                }
            }
```

```
                if (check == false)
                {
                    JOptionPane.showMessageDialog(f,"Invalid vacancy number.");
                }

            }

            catch(NumberFormatException ex)
            {
                JOptionPane.showMessageDialog(f,ex.getMessage());
            }

        }

        if (click.getSource() == clearP)
        {
            t_VacNumP.setText("");
            t_DesignationP.setText("");
            t_JobTypeP.setText("");
            t_WorkingHourP.setText("");
            t_WorkingShiftsP.setText("");
            t_WagesPerHourP.setText("");
            t_VacNumAppointP.setText("");
            t_StaffNameP.setText("");
            t_QualificationP.setText("");
            t_JoiningDateP.setText("");
            t_AppointedByP.setText("");
            t_VacNumTerminateP.setText("");
            t_VacNumDisplayP.setText("");


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