CS4001NI - PROGRAMMING

30% Individual Coursework

2020-21 Autumn

Student Name: Karsang Gurung

London Met ID: 19031333

College ID: NP01NT4A190138

Assignment Due Date: 5th June, 2020

Assignment Submission Date: 5th June, 2020

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

Contents

List of Tables

List of Figures

1. Introduction

A Graphical User Interface (GUI) is a computer program that allows the user to interact with computer system through commands, files and many other visual representations such as icons, menu, scroll bars, windows and dialog boxes. It allows the user to communicate by either moving the mouse or clicking the buttons. (Anon., 2004)

Figure 1: Graphical User Interface

This coursework solely depends upon making of the Graphical User Interface (GUI) for the previous coursework. The GUI made in this coursework is for hiring full time staffs and part time staffs, respectively. A GUI was made for a new class INGNepal that contains the information about the vacant posts and the staffs appointed to them. Here, the GUI contains Frames, Labels, Buttons and Text Fields. The IDE used for this particular coursework is the same one used for previous coursework i.e. Bluej which is very reliable and convenient for Java programming. Details of the GUI will also be further described here.

2. Class Diagram

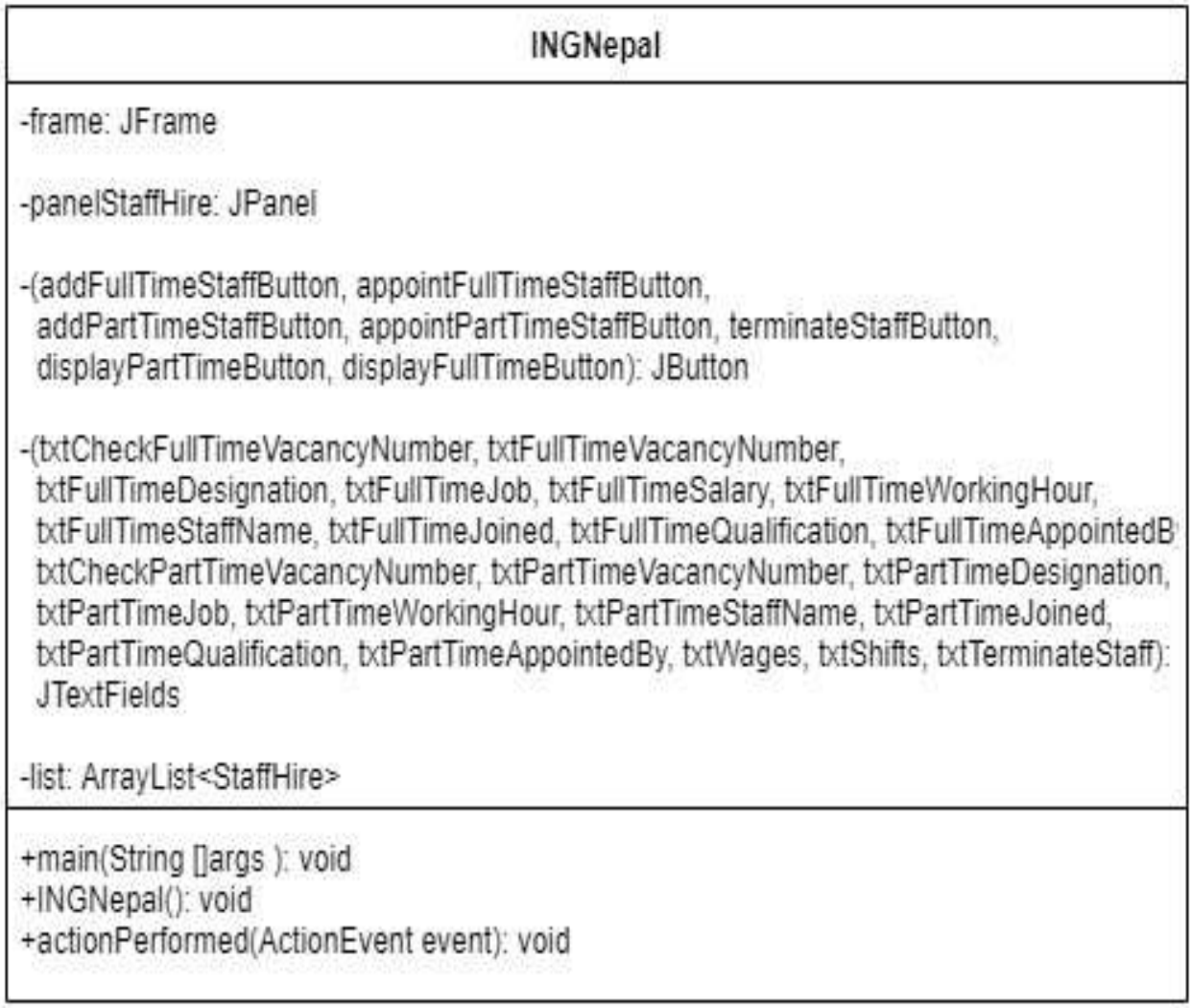


Figure 2: Class Diagram

3. Pseudocodes

```
public void
actionPerformed(ActionEvent event)
DO
  IF
AddFullTimeStaffHire  button is called
THEN
DO
      IF
         All the Text field are not filled
       THEN
         DO
         Display respective message in the dialog box
END IF

ELSE
DO
TRY
DO
         Use getText()method get the value of txtVacancy, convert it to whole
         number and set the value as vacancyNumber.
         Use getText()method get the value of txtDesignation and set the
         value as designation
         Use getText()method get the value of (String)Job and set the value
         as jobType
```

Use getText()method get the value of txtSalary,convert it to whole number and set the value as Salary

Use getText()method get the value of Working,convert it to whole number and set the value as WorkingHour
Put the value of Duplicate vacancy number as false

FOR each(unit) value in list

DO

IF vacancy number and the existing vacancy number is equal with the input value

THEN

DO

IF (unit) belongs to FullTimeStaffHire THEN

DO

Duplicate vacancy number is true

Break

END IF

END IF

END DO

IF (equals to false THEN

DO

Object is made of FullTime and is stored in the arraylist

Display Dialog box.

END IF

ELSE

DO

Display dialog box.

END ElSE

END DO

Catch(NumberFormatException exp)

DO

Display Dialog box.

END DO

END
ELSE
END IF

IF

AddPartTimeStaffHire button is called

THEN

DO

IF

All text field are not filled

THEN

DO

Display dialog box

END IF

ELSE

DO

TRY

DO

Use getText()method get the value of txtVacancy,convert it to whole number And set the value as vacancyNumber.

Use getText()method get the value of txtDesignation and set the value as designation

Use getText()method get the value of (String)Job and set the value as jobType

Use getText()method get the value of Working, convert it to whole number and set the value as WorkingHour

Use getText()method get the value of txtWagesPerHour,convert it to whole number and set the value as WagesPerHour.

Use getText()method get the value of txtShifts and set the value as Shifts
Put the value of Duplicate vacancy number as false **FOR** each(unit)

value in list

DO

IF vacancy number and the existing vacancy number is equal with the input value

THEN

DO

IF (unit) belongs to PartTimeStaffHire **THEN**

DO

Duplicate vacancy number is true

Break

END IF

END IF

END DO

IF equals to false **THEN**

DO

Object is made of PartTime and is stored in the arrayList Display

Dialog box.

END IF

ELSE

DO

Display the respective message in the Dialog box

END ELSE

END DO

Catch(NumberFormatException exp)

DO

Display Dialog box.
END DO

END
ELSE
END IF

IF

appointFullTimeStaffButton button is called

THEN

DO

IF

all text fields are not filled **THEN**

DO

Display respective message in the dialog box

END DO

ELSE

DO

TRY

DO

Use getText()method get the value of txtVacancy, convert it to whole number

Use getText()method get the value of txtStaffName and set the value as StaffName

Use getText()method get the value of txtJoiningDate and set the value as JoiningDate

Use getText()method get the value of txtQualification set the value as Qualification

Use getText()method get the value of txtAppointedby and set the value as Appointedby Put the value of count equals to one

FOR each(unit) value in list

DO

IF

vacancy number and the existing vacancy number is equal with the input value

THEN

DO

IF (unit) belongs to FullTimeStaffHire THEN **DO**

Object is made of FullTime and is stored in the arrayList

IF created object's value is true

THEN

DO

Display the respective message in the Dialog box.

Break

END IF

ELSE

DO

Created object is used to call the HireFullTimeStaff method of FullTimeStaffHire class by putting the same values in same order inside brackets.

Display the respective message in the Dialog box.

Break

END ELSE

END IF

ELSE

DO

IF value of count EQUALS to list size

THEN

DO

Display the respective message in Dialog box.

Break

END IF

END ELSE

END IF

ELSE

DO

IF value of count EQUALS to list size

THEN DO

Display Dialog box

Break

END IF

END ELSE

Increase the value of count by 1

END DO

END DO

Catch(NumberFormatException exp)

DO

Display Dialog box

END DO

END
ELSE
END IF

IF

appointPartTimeStaffButton button is called

THEN DO

IF all the text field are not empty

THEN

DO

Display dialog box

END IF

ELSE

DO

TRY

DO

Use getText()method get the value of txtVacancyand set the value as VacancyNumber

Use getText()method get the value of txtStaffName and set the value as Staff_Name

Use getText()method get the value of txtJoiningDate and set the value as JoiningDate

Use getText()method get the value of txtQualification and set the value as Qualification

Use getText()method get the value of txtAppointedby and set the value as Appointedby

Put the value of count equals to one

FOR each(unit) value in list

DO

IF vacancy number and the existing vacancy number is equal with the input value

THEN

DO

IF (unit) belongs to PartTimeStaffHire **THEN DO**

Object is made of PartTime and is stored in arraylist

IF the join value of created object is true

THEN

DO

Display the respective message in the Dialog box.

Break

END IF

ELSE

DO

Created object is used to call the HirePartTimeStaff method of PartTimeStaffHire class by puting the same values in same order inside brackets.

Display the respective message inthe Dialog box.

Break

END ELSE

END IF

ELSE

DO

IF value of count EQUALS to list size

THEN

DO

Display the respective message in Dialog box.

Break

END IF

END ELSE

END IF

ELSE

DO

IF value of count EQUALS to list size

THEN

DO

Display the respective message in Dialog box

Break

END IF

END ELSE

Increase the value of count by 1

END DO

END DO

Catch(NumberFormatException exp)

DO

Display the respective message in Dialog box

END DO

END
ELSE
END IF

IF

Terminate button is called

THEN DO

IF all the text field are not empty

THEN

DO

Display respective message in the dialog box

END IF

ELSE

DO

TRY

DO

Use getText()method get the value of Vacancy convert it to whole number And set the value as VacancyNumber

Put the value of count equals to one

FOR each(unit) value in list

DO

IF vacancy number and the existing vacancy number is equal with the input value

THEN

DO

IF (unit) belongs to PartTimeStaffHire THEN

DO

Object is made of PartTime and is stored in arraylist

IF the join value of created object is true

THEN

DO

Display the respective message in Dialog box.

Break

END IF

ELSE

DO created object is used to call the TerminateStaff method of PartTimeStaffHire class.

Display the respective message in Dialog box.

Break

END ELSE

END IF

ELSE

DO

Display Dialog box.

Break

END ELSE

END IF

ELSE

DO

IF value of count EQUALS to list size

THEN

DO

Display the respective message in Dialog box.

Break

END IF

END ELSE

Increase the value of count by 1

END DO

END DO

Catch(NumberFormatException exp)

DO

Display the respective message in Dialog box

END DO

END
ELSE
END IF

IF

Display button is called

THEN DO

IF the list size is

zero

THEN

DO

Display the respective message in Dialog box

END IF

ELSE

DO

FOR each(unit) value in list

DO

IF (unit) belongs to FullTimeStaffHire **THEN DO**

Object is made of FullTime and is stored in arraylist

Created object is used to call the display method of class FullTimeStaffHire

END IF

END DO

END ELSE

END IF

IF

Display button is called

THEN DO

IF list size is

zero

THEN

DO

Display the respective message in Dialog box

END IF

ELSE

DO

FOR each(unit) value in list

DO

IF (unit) belongs to PartTimeStaffHire **THEN**

DO

Object is made of PartTime and is stored in arraylist

Created object is used to call the display method of class PartTimeStaffHire

END IF

END DO

END ELSE

END IF

IF

Clear button is called

THEN

DO

Use setText() method to set the value of txtVacancy and set the value as “ ”

Use setText() method to set the value of txtDesignation and set the value as “ ”

Use setText() method to set the value of txtSalary and set the value as “ ”

Use setText() method to set the value of txtWorking and set the value as “ ”

Use setText() method to set the value of txtVacancy and set the value as “ ”

Use setText() method to set the value of txtStaffName and set the value as “ ”

Use setText() method to set the value of txtQualification and set the value as “ ”

Use setText() method to set the value of txtJoiningDate and set the value as “ ”

Use setText() method to set the value of txtAppointedby and set the value as “ ”

END IF

IF

Clear button is called

THEN

DO

Use setText() method to set the value of txtVacancy and set the value as “ ”

Use setText() method to set the value of txtDesignation and set the value as “ ”

Use setText() method to set the value of txtWorking and set the value as “ ”

Use setText() method to set the value of txtWagesPerHour and set the value as “ ”

Use setText() method to set the value of txtShifts and set the value

as “ ”

Use setText() method to set the value of txtVacancy and set the value as “ ”

Use setText() method to set the value of txtStaffName and set the value as “ ”

Use setText() method to set the value of txtQualificationAreaB and set the value as “ ”

Use setText() method to set the value of txtJoiningDate and set the value as “ ”

Use setText() method to set the value of txtAppointedby and set the value as “ ”

Use setText() method to set the value of txtVacancy and set the value as “ ”

END IF

4. Method Description

Some of the methods used to build GUI are described in the table below:

Method	Description
public void actionPerformed()	This method controls all the actions of the codes. Inside this method, codes are written that are to be execute when action is occurred. Basically, it is responsible for all the actions pefomed in the GUI.
Public static void main()	This method is called to execute the program.
Public INGNepal()	This method is called to write the codes for GUI in it. The return type of this method is void.

Table 1: Method Description

5. Testing

5.1 Test that confirms program can be compiled and run using Command Prompt.

Objective	To compile and run the program using command prompt (cmd)
Action	Used command prompt (cmd) for the compilation and running the program
Expected result	Program must be compiled and run.
Actual result	Program was compiled and run
Conclusion	Test was successful.

Table 2: Test for compilation and running in cmd

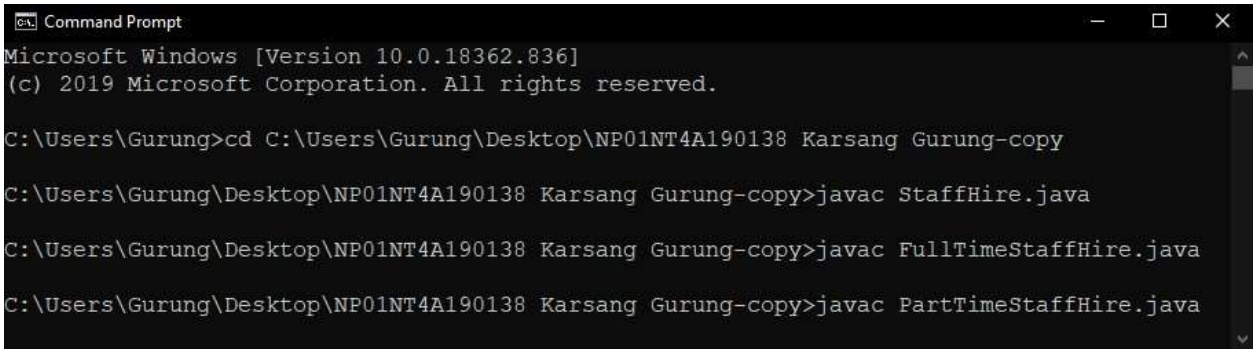


Figure 3: Compilation of GUI

Figure 4: Running the GUI

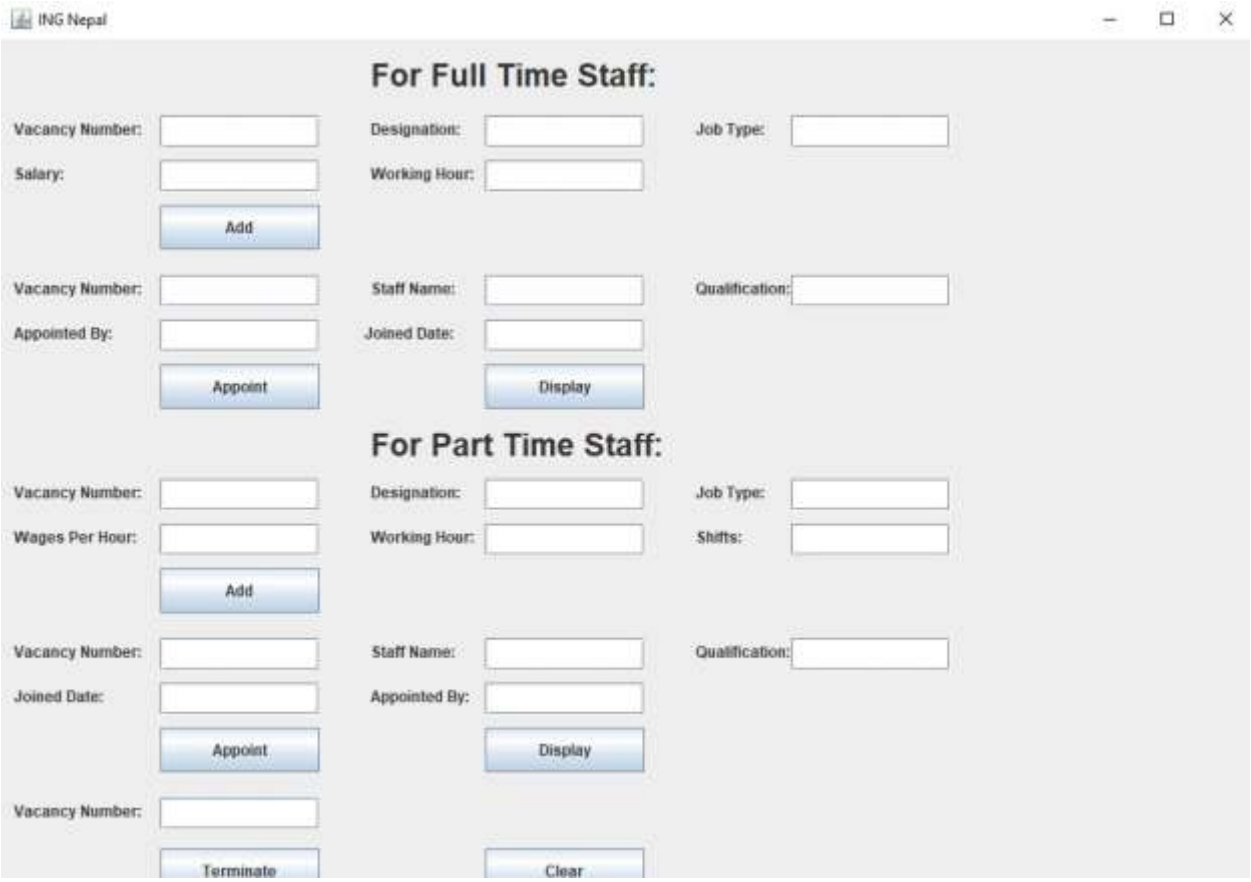


Figure 5: GUI after compiling and running id command prompt

5.2 Test for adding vacancy for full time and part time staff

5.2.1 Adding vacancy for Full Time Staff

Objective	To add vacancy for Full Time Staff

Action	Following details were filled to add vacancy for full time staff: • Vacancy Number=1 • Designation=Manager • Job Type=Full Time • Salary=45000 • Working Hour =8
Expected result	A dialog box with “Full Time staff has been added.” Should appear.
Actual result	New vacancy added.
Conclusion	Test was successful.

Table 3: Addition of vacancy for full time staff

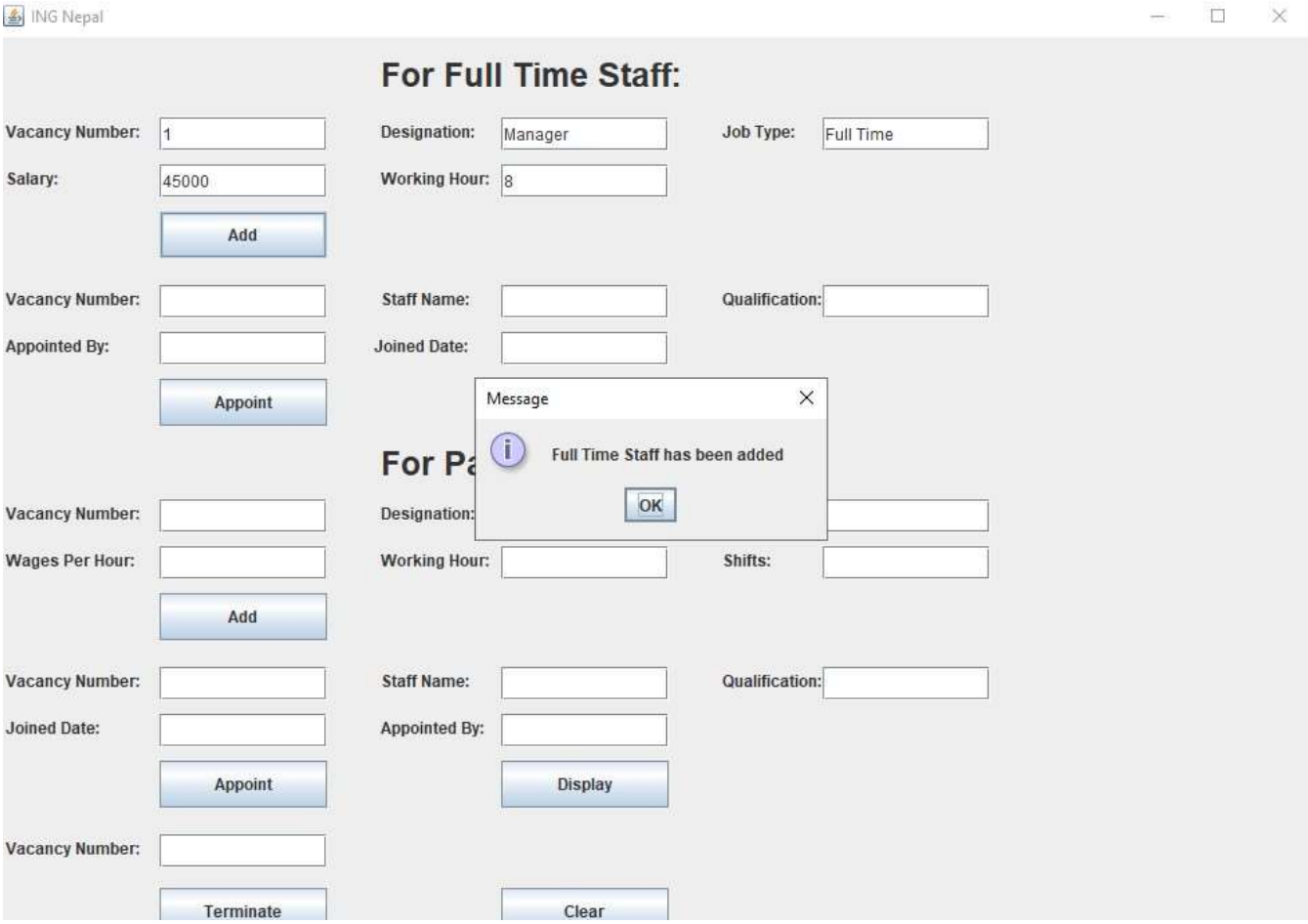


Figure 6: Adding vacancy for Full Time Staff

5.2.2 Adding vacancy for Part Time Staff

Objective	To add vacancy for Full Time Staff
Action	Following details were filled to add vacancy for full time staff: • Vacancy Number=1 • Designation=Managing Director • Job Type=Part Time • Wages Per Hour=900 • Working Hour =8 • Shifts=Day
Expected result	A dialog box with “Part Time staff has been added.” Should appear.
Actual result	New vacancy added.
Conclusion	Test was successful.

Table 4: Adding vacancy for Part Time Staff

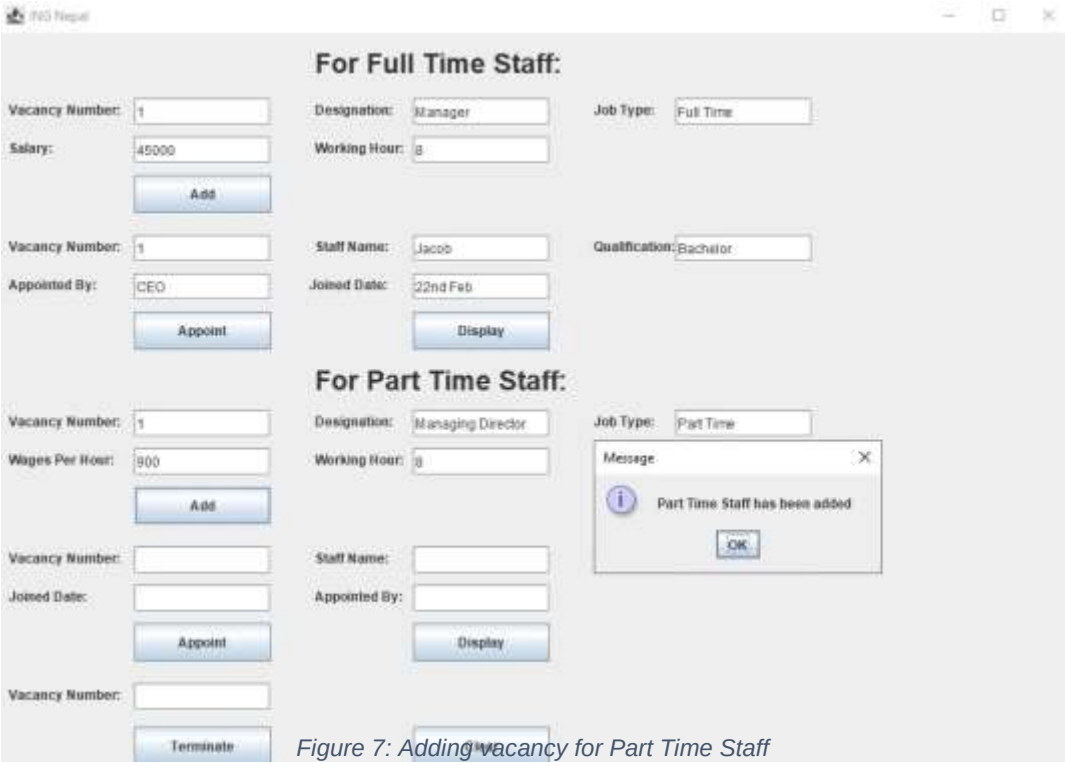


Figure 7: Adding vacancy for Part Time Staff

5.3 Test for appointing vacancy for Full Time Staff and Part Time Staff

5.3.1 Test for appointing vacancy for Full Time Staff

Objective	To appoint vacancy for Full Time Staff
Action	Following details were filled to add vacancy for full time staff: • Vacancy Number=1 • Staff Name=Jacob • Qualification=Bachelors • Appointed By=CEO • Joined Date=22nd Feb Appoint Full Time Staff Display Full Time Staff

Expected result	A dialog box with “Full Time staff has been appointed.” Should appear and displayed.
Actual result	Full Time Staff was appointed and displayed.
Conclusion	Test was successful.

Table 5: Appointing vacancy for Full Time Staff

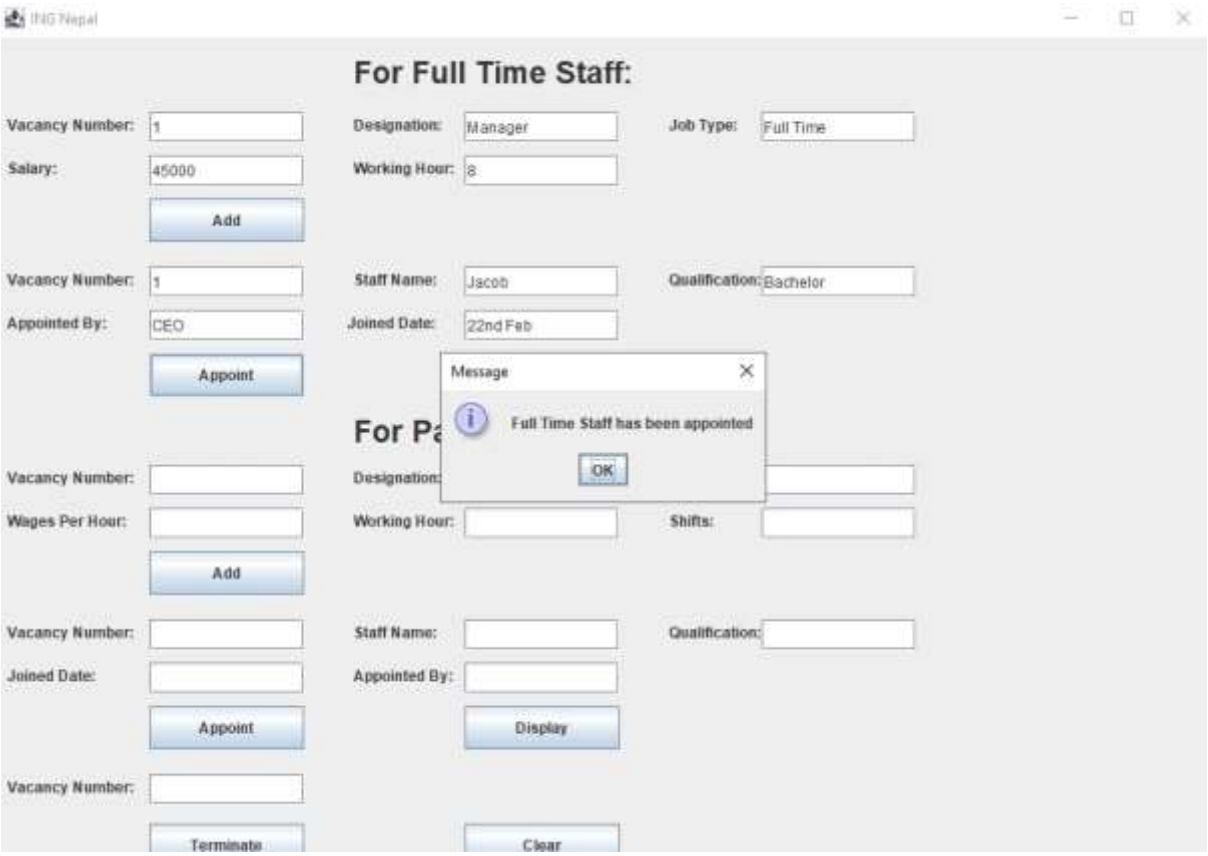


Figure 8: Appointing Full Time Staff

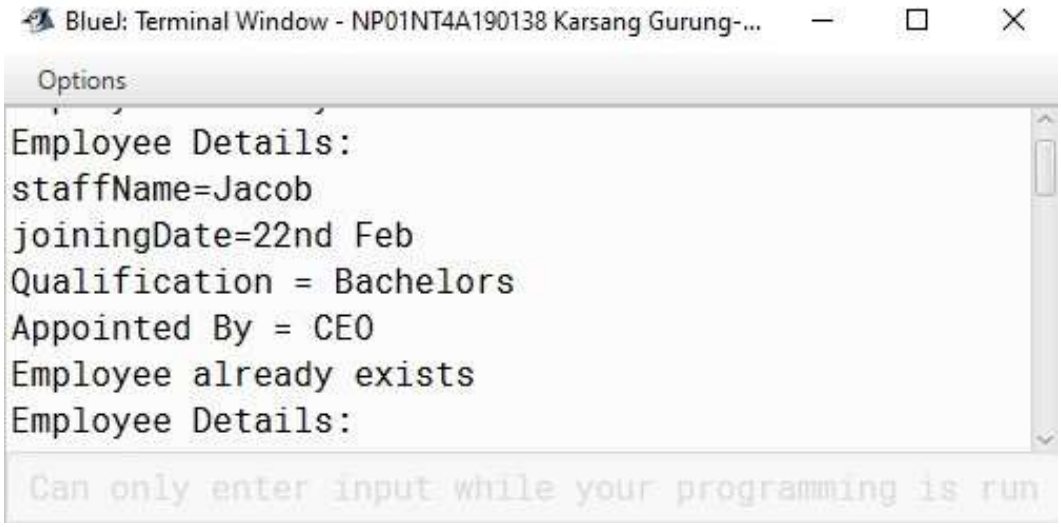


Figure 8: Displaying Full Time Staff

5.3.2 Test for appointing vacancy for Part Time Staff

Objective	To appoint vacancy for Part Time Staff
Action	Following details were filled to add vacancy for full time staff: • Vacancy Number=1 • Staff Name=Ozzy • Qualification=Ph.D • Appointed By=Co-Founder • Joined Date=5th November Appoint Full Time Staff Display Full Time Staff
Expected result	A dialog box with “Part Time staff has been appointed.” Should appear and displayed.
Actual result	Part Time Staff was appointed and displayed.
Conclusion	Test was successful.

Table 6: Appointing vacancy for Part Time Staff

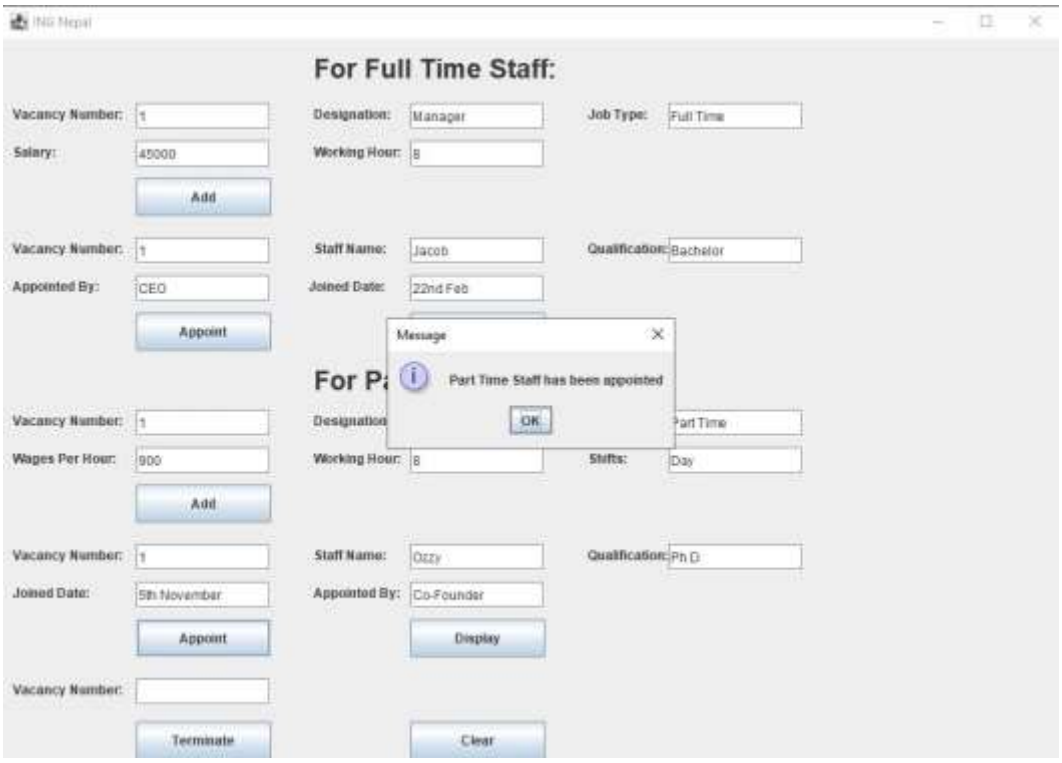


Figure 9: Appointing Part Time Staff

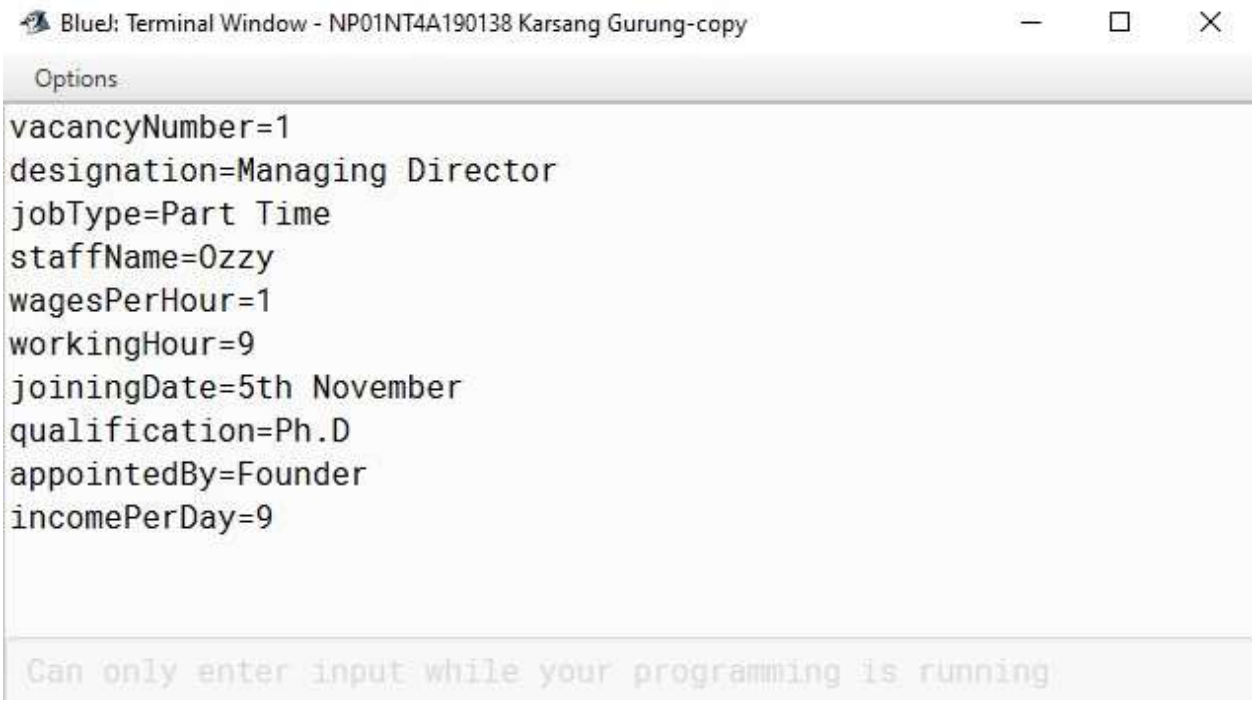


Figure 10: Displaying Part Time Staff

5.4 Test for appropriate dialog boxes

5.4.1 To test if the text fields can be empty

Objective	To show if text fields can be empty or not
Action	One of the text field was left empty
Expected result	Dialog boxes with “textfield cannot be empty” must be shown on the screen.
Actual result	Dialog box appeared on the screen.
Conclusion	Test was successful.

Table 7: Tesitng if textfields can be empty or not

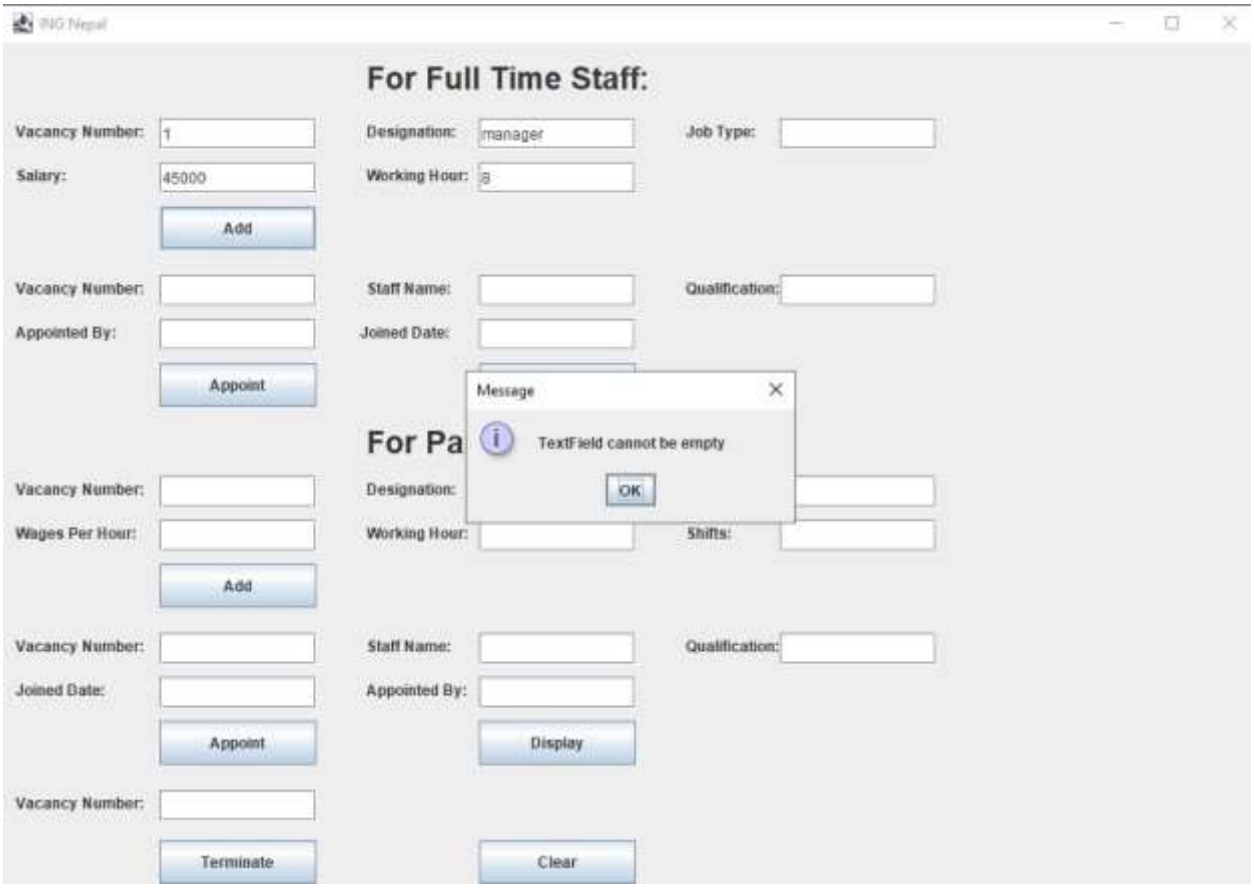


Figure 11: test for empty textfield

5.4.2 to test suitable value for vacancy number

Objective	To test suitable value for vacancy number
Action	Unsuitable for vacancy number was entered
Expected result	Dialog boxes with warning should appear on the screen
Actual result	Dialog box appeared on the screen.
Conclusion	Test was successful.

Table 8: testing suitable value for vacancy number

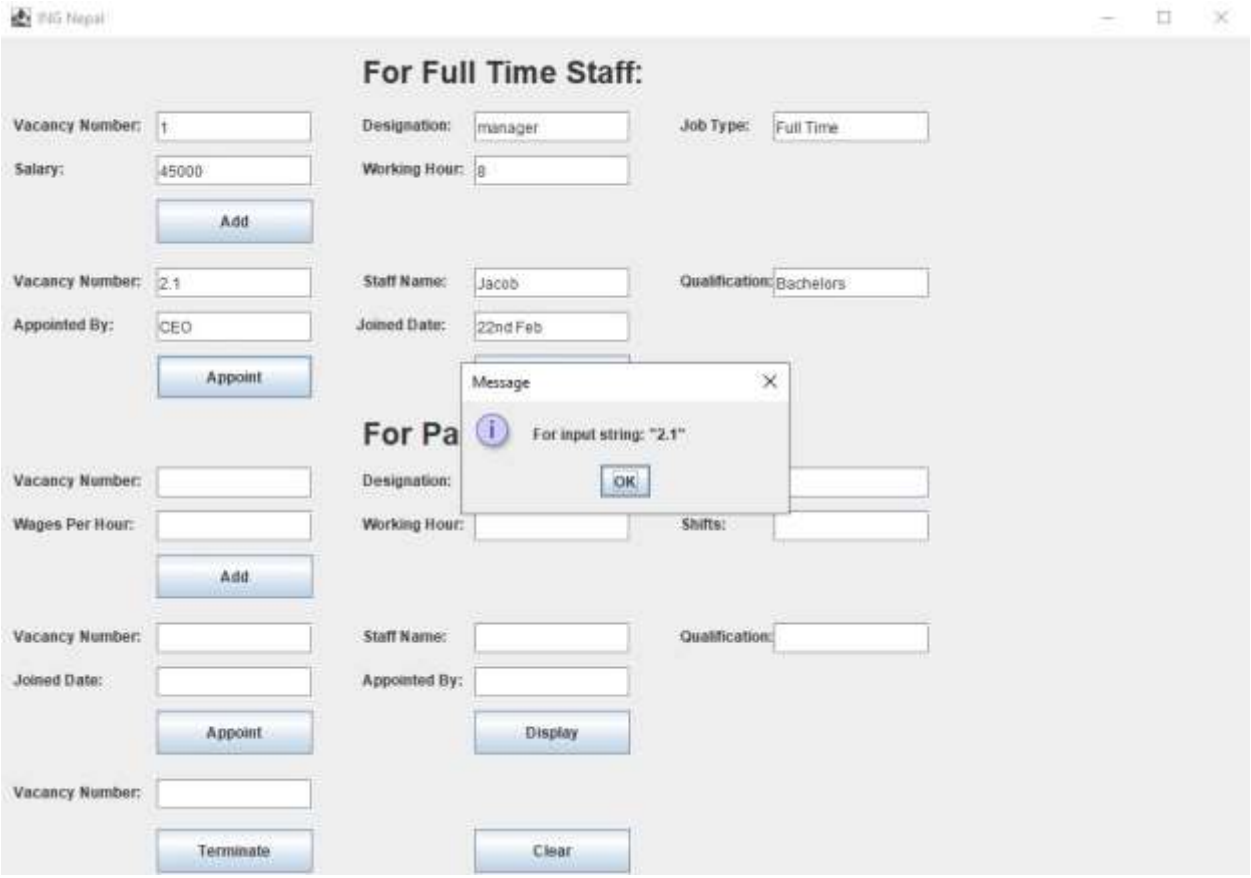
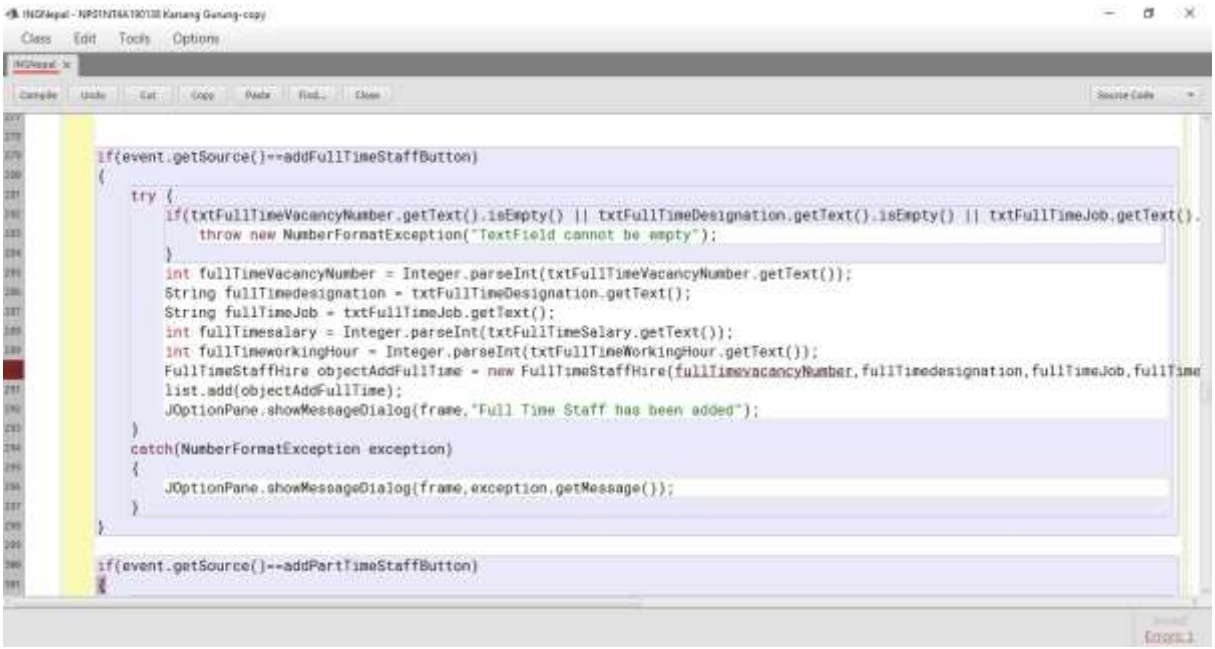


Figure 12: test for suitable value

6. Error detection and correction

6.1 Syntax error

Syntax errors are the minor error that occurs during compilation of the program. These type of errors are occurred due to the mistake due to the camel casing difference or not ending the code in proper way. The error detection and correction is demonstrated below.



```
if(event.getSource()==addFullTimeStaffButton)
{
    try {
        if(txtFullTimeVacancyNumber.getText().isEmpty() || txtFullTimeDesignation.getText().isEmpty() || txtFullTimeJob.getText().
            throw new NumberFormatException("TextField cannot be empty");
        }
        int fullTimeVacancyNumber = Integer.parseInt(txtFullTimeVacancyNumber.getText());
        String fullTimedesignation = txtFullTimeDesignation.getText();
        String fullTimeJob = txtFullTimeJob.getText();
        int fullTimesalary = Integer.parseInt(txtFullTimeSalary.getText());
        int fullTimeworkingHour = Integer.parseInt(txtFullTimeWorkingHour.getText());
        FullTimeStaffHire objectAddFullTime = new FullTimeStaffHire(fullTimevacancyNumber,fullTimedesignation,fullTimeJob,fullTime
        list.add(objectAddFullTime);
        JOptionPane.showMessageDialog(frame,"Full Time Staff has been added");
    }
    catch(NumberFormatException exception)
    {
        JOptionPane.showMessageDialog(frame,exception.getMessage());
    }
}

if(event.getSource()==addPartTimeStaffButton)
{
```

Figure 13: Syntax error detection

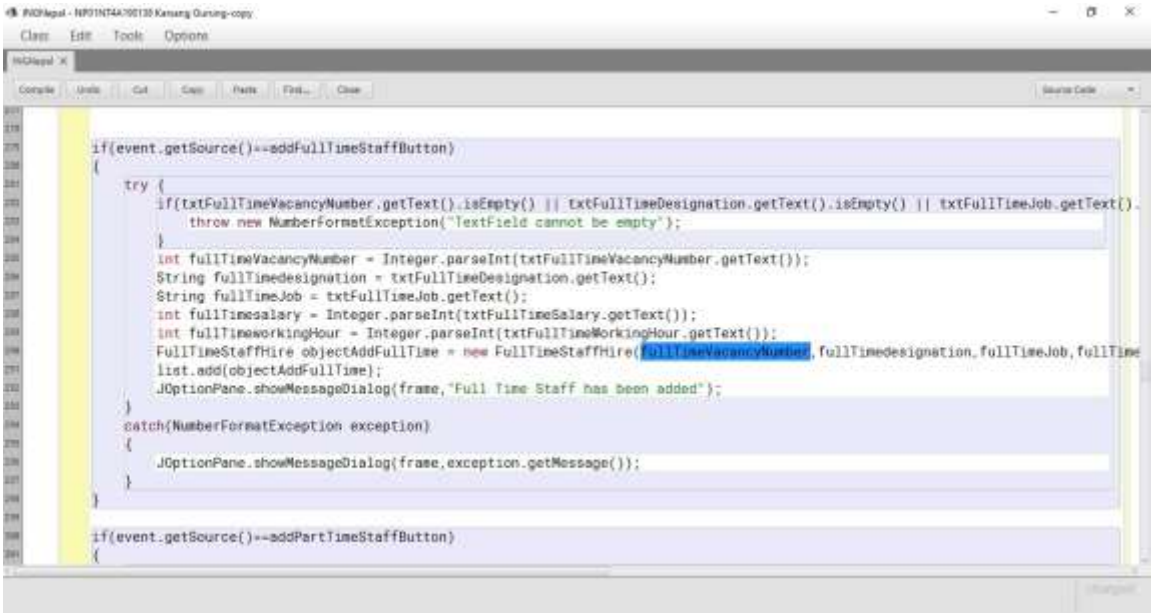


Figure 14: Syntax error correction

6.2 Semantic error

Semantic errors are the errors that occurs when the grammar of the code is correct but the use of it is not. It can also be the error in which code utilizes the variables that aren't instated.

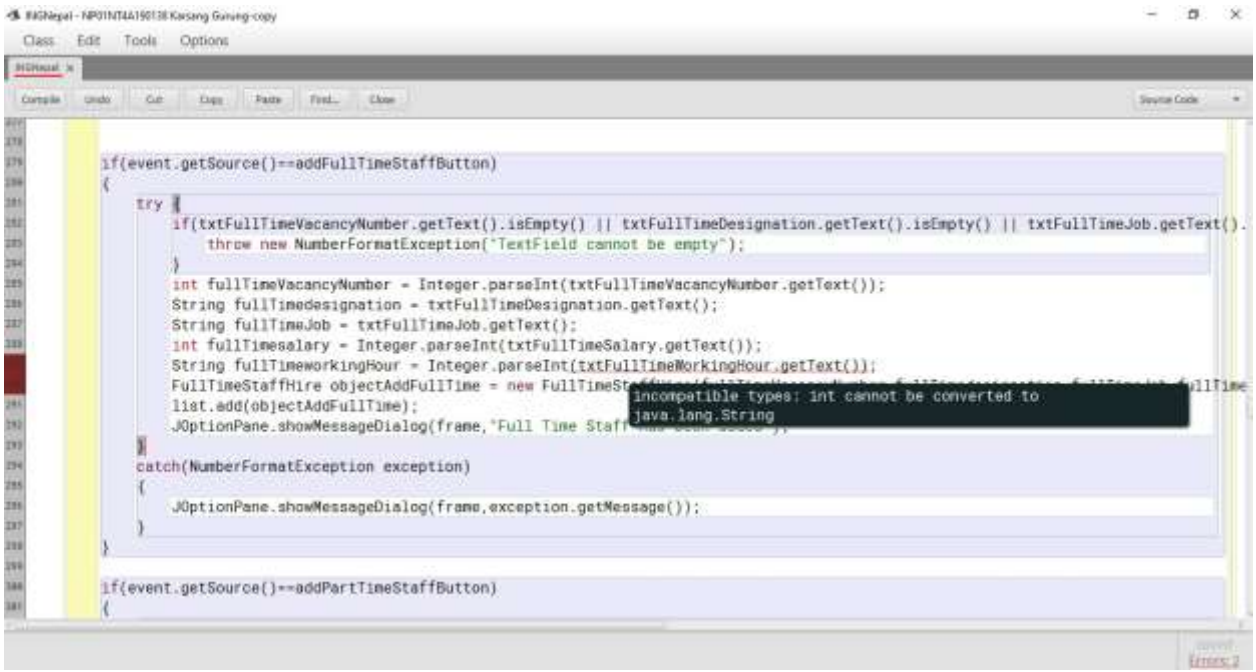
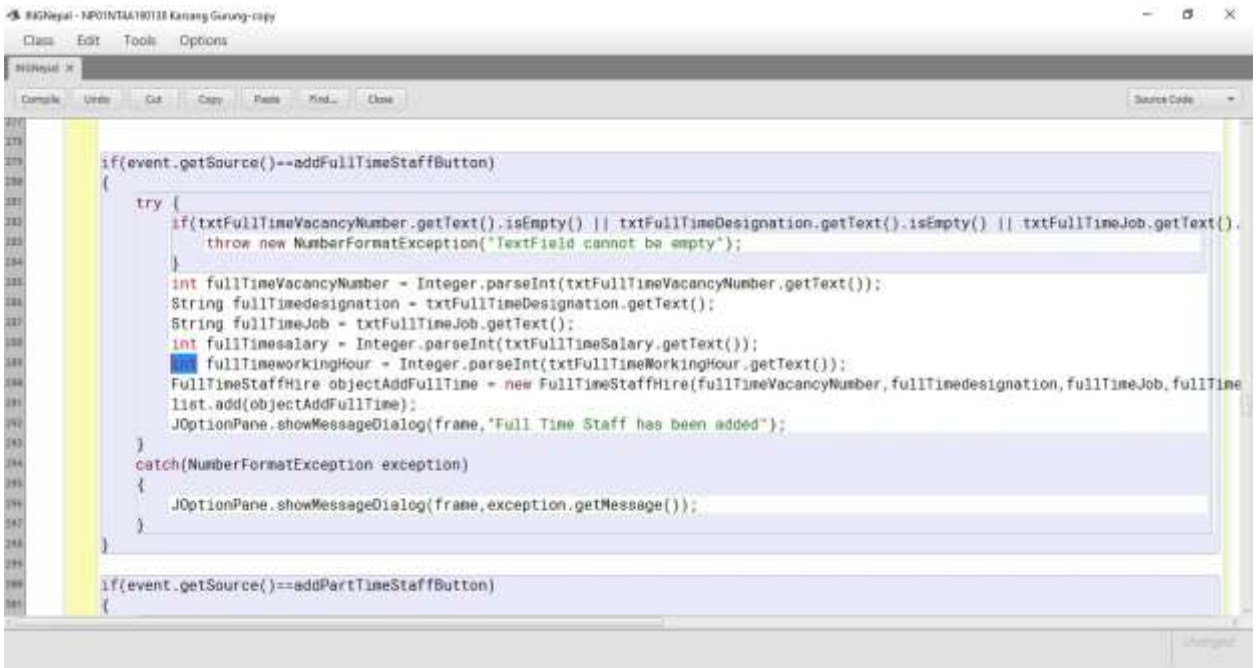


Figure 15: Semantic error detection



```
if(event.getSource()==addFullTimeStaffButton)
{
    try {
        if(txtFullTimeVacancyNumber.getText().isEmpty() || txtFullTimeDesignation.getText().isEmpty() || txtFullTimeJob.getText().
            throw new NumberFormatException("TextField cannot be empty");
        }
        int fullTimeVacancyNumber = Integer.parseInt(txtFullTimeVacancyNumber.getText());
        String fullTimedesignation = txtFullTimeDesignation.getText();
        String fullTimeJob = txtFullTimeJob.getText();
        int fullTimesalary = Integer.parseInt(txtFullTimeSalary.getText());
        int fullTimeworkingHour = Integer.parseInt(txtFullTimeWorkingHour.getText());
        FullTimeStaffHire objectAddFullTime = new FullTimeStaffHire(fullTimeVacancyNumber,fullTimedesignation,fullTimeJob,fullTime
        list.add(objectAddFullTime);
        JOptionPane.showMessageDialog(frame,"Full Time Staff has been added");
    }
    catch(NumberFormatException exception)
    {
        JOptionPane.showMessageDialog(frame,exception.getMessage());
    }
}

if(event.getSource()==addPartTimeStaffButton)
{
```

Figure 16: Semantic error correction

6.3 Run time error

Run time errors are those errors, which are occurred when the program is run. Such errors are occurred when the data types do given that are not support the input or the input is not given. A demonstration is shown below.

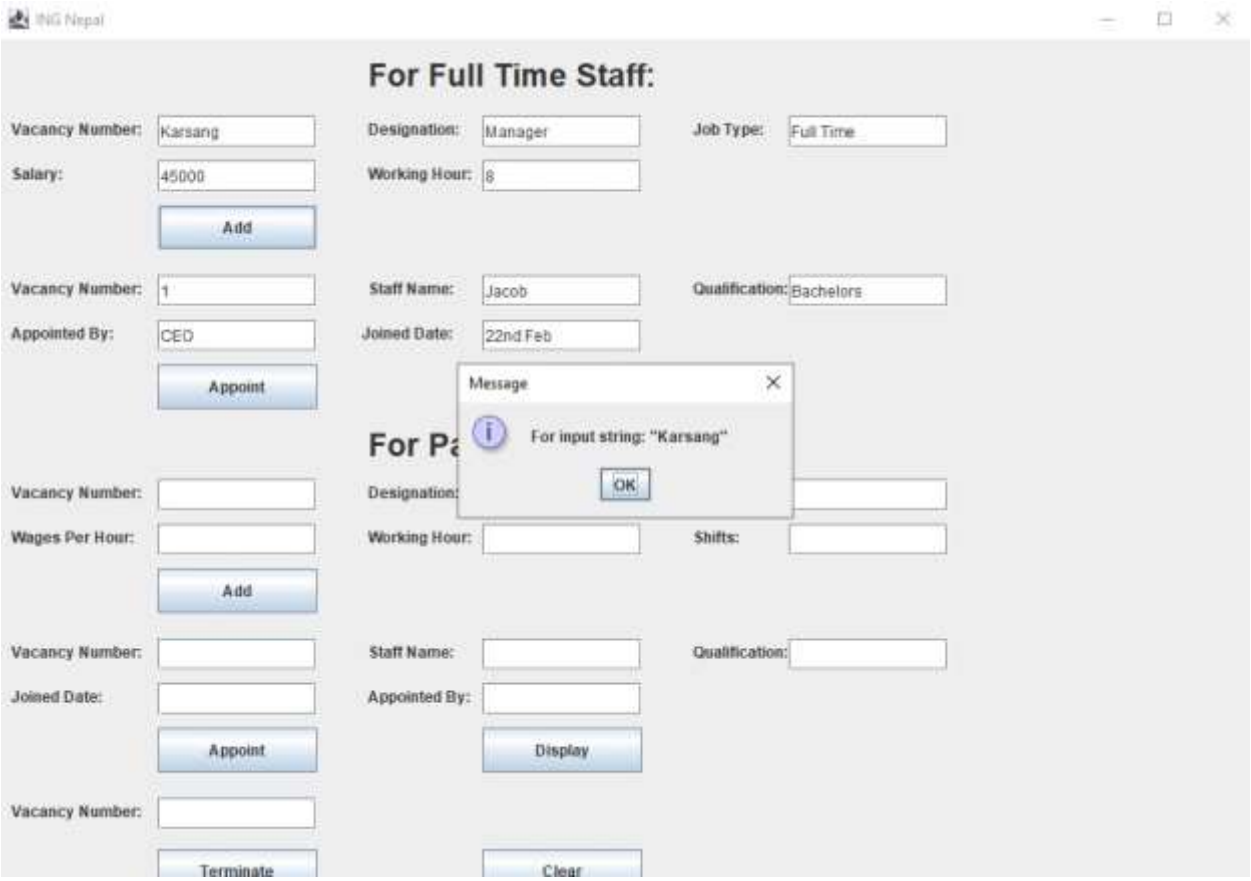


Figure 17: Run time error detection

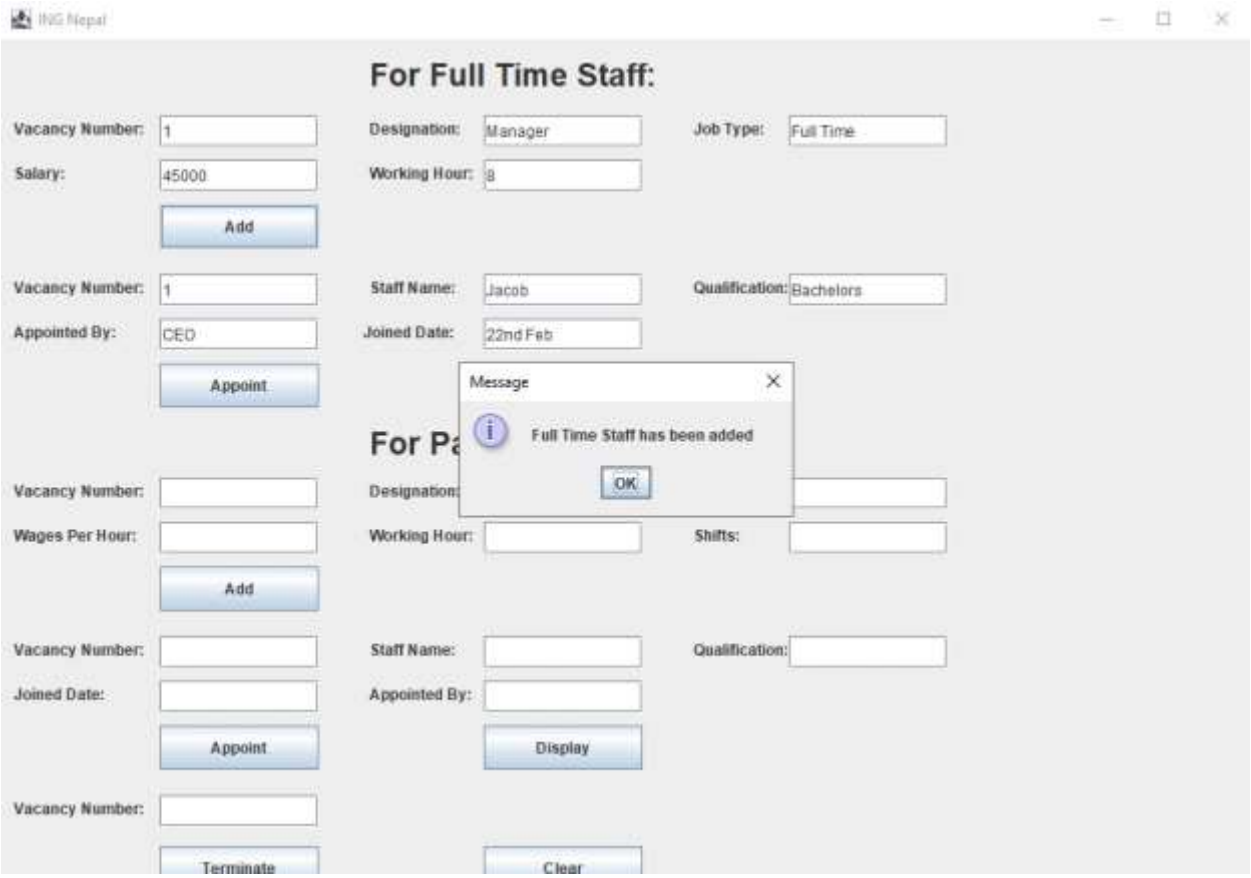


Figure 19: Run time error correction

7. Conclusion

Finally, the coursework has ended. This coursework was all about dealing with frame, panels, buttons and textfields to create a Graphical User Interface (GUI). Also, an opportunity to work with methods like actionPerformed and actionListener. This

coursework has been really helpful to learn about creating GUI and storing data in them. It has also been helpful in dealing with different types of errors while creating a program. But most importantly, this coursework has helped all of us individuals to work with event handling which might be very useful in upcoming days. Lastly, I'm very thankful to my module leader and teachers for helping and guiding through out the assessment.

8. References

1. Anon., 2004. *Linux Information Project.* [Online] Available at: http://www.linfo.org/gui.html

2. Anon., n.d. [Online] Available at: https://www.omnisci.com/technical-glossary/graphical-user-interface

3. Anon., n.d. [Online] Available at: https://www.omnisci.com/technical-glossary/graphical-user-interface

9. Appendix

9.1 Staff Hire

```
public class StaffHire
{
   protected int vacancyNumber;
protected String designation;     protected
String jobType;

   public StaffHire(int vacancyNumber,String designation,String jobType)
   {
      this.vacancyNumber=vacancyNumber;
this.designation=designation;
      this.jobType=jobType;
   }
   public void setvacancyNumber(int vacancyNumber)//setter method
   {
      this.vacancyNumber=vacancyNumber;
   }
   public int getvacancyNumber()//getter method
   {
      return this.vacancyNumber;
   }
   public void setdesignation(String designation)//setter method
   {
      this.designation=designation;
   }
   public String getdesignation()//getter method
   {
      return this.designation;
   }
   public void setjobType(String jobType)//setter method
   {
      this.jobType=jobType;
   }
   public String getjobType()//getter method
   {
      return this.jobType;
```

```
    }
    public void display()
    {
        System.out.println("vacancyNumber="+getvacancyNumber());
        System.out.println("designation="+getdesignation());
        System.out.println("jobType="+getjobType());
    }
}
```

9.2 Full Time Staff Hire

```
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 vacancyNumber, String designation, String
        jobType, int salary, int workingHour)
           {
              super(vacancyNumber,designation,jobType);//calling superclass conductor
              this.salary=salary;
              this.workingHour=workingHour;//giving value to the empty string
        this.staffName="";        this.joiningDate="";        this.qualification="";
        this.appointedBy="";        this.joined=false;
           }

           public void setSalary(int salary)//setter method
           {
              this.salary=salary;
           }
           public int getSalary()//getter method
           {
              return this.salary;
           }
           public void setworkingHour(int workingHour)//setter method
           {
              this.workingHour=workingHour;
           }
           public int getworkingHour()//getter method
           {
              return this.workingHour;
           }
           public void setstaffName(String staffName)//setter method
           {
              this.staffName=staffName;
```

```
}
public String getstaffName()//getter method
{
    return this.staffName;
}
public void setjoiningDate(String joiningDate)//setter method
{
    this.joiningDate=joiningDate;
}
public String getjoiningDate()//getter method
{
    return this.joiningDate;
}
public void setqualfication(String qualification)//setter method
{
    this.qualification=qualification;
}
public String getqualification()//getter method
{
    return this.qualification;
}
public void setappointedBy(String appointedBy)//setter method
{
    this.appointedBy=appointedBy;
}
public String getappointedBy()//getter method
{
    return this.appointedBy;
}
public void setjoined(boolean joined)//setter method
{
    this.joined=joined;
}
public boolean getjoined()//getter
{
    return this.joined;
}
public int gerSalary() //method to set new salary
{
   return this.salary;
```

```
    }

    public void workingHours(int new_workinghour) //method to set new working
hour
    {
        this.workingHour=new_workinghour;
    }

    public void HirefulltimeStaff(String staffName, String joiningDate, String
qualification, String appointedBy) //method defination for joined display
    {
        if (this.joined)
        {
            System.out.println("Employee already exists");
            System.out.println("Employee Details:");
            System.out.println("staffName="+getstaffName());
            System.out.println("joiningDate="+getjoiningDate());
            System.out.println("Qualification = " + getqualification());
            System.out.println("Appointed By = " + getappointedBy());
        }
else
        {
            this.staffName=staffName;
this.joiningDate=joiningDate;
this.qualification=qualification;
this.appointedBy=appointedBy;
            this.joined=true;
        }
    }

    public void display()  //display method
    {
        super.display();
if (joined)
        {
            System.out.println("staffName"+getstaffName());
            System.out.println("salary"+getSalary());
            System.out.println("workingHour"+getworkingHour());
            System.out.println("joiningDate"+getjoiningDate());
            System.out.println("qualification"+getqualification());
```

```
            System.out.println("appointedBy"+getappointedBy());
        }
        else{
        System.out.println("No employee data");
        }
    }
}
```

9.3 Part Time Staff Hire

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

   public PartTimeStaffHire(int vacancyNumber,String designation,String
jobType,
   int workingHour,int wagesPerHour, String shifts)
   {
      super(vacancyNumber,designation,jobType);
this.workingHour=workingHour;        this.wagesPerHour=wagesPerHour;
      this.staffName="";
this.joiningDate="";
this.qualification="";
this.appointedBy="";
this.shifts="";        this.joined=false;
      this.terminated=false;
   }
   public void setworkingHour(int workingHour)//setter method
   {
      this.workingHour=workingHour;
   }
   public int getworkingHour()//getter Method
   {
      return this.workingHour=workingHour;
   }
   public void setwagesPerHour(int wagesPerHour)//setter method
   {
      this.wagesPerHour=wagesPerHour;
   }
```

```
public int getwagesPerHour()//getter method
{
    return this.wagesPerHour=wagesPerHour;
}
public void setstaffName(String staffName)//setter method
{
    this.staffName=staffName;
}
public String getstaffName()//getter method
{
    return this.staffName=staffName;
}
public void setjoiningDate(String joiningDate)//setter method
{
    this.joiningDate=joiningDate;
}
public String getjoiningDate()//getter method
{
    return this.joiningDate=joiningDate;
}
public void setqualification(String qualification)//setter method
{
    this.qualification=qualification;
}
public String getqualification()//getter method
{
    return this.qualification=qualification;
}
public void setappointedBy(String appointedBy)//setter method
{
    this.appointedBy=appointedBy;
}
public String getappointedBy()//getter method
{
    return this.appointedBy;
}
public void setShifts(String shifts)
{
if (this.joined == true) {
  System.out.println("The working shift cannot be changed");
```

```
    }
else {
      this.shifts = shifts;
    }

}
    public void setjoined(boolean joined)//setter method
    {
        this.joined=joined;
    }
    public boolean getjoined()//getter method
    {
        return this.joined;
    }
    public void setterminated(boolean terminated)//setter method
    {
        this.terminated=terminated;
    }
    public boolean getterminated()//getter method
    {
        return this.terminated;
    }


    public void PartTimeStaffHire(String staffName, String joiningDate,
String qualification, String appointedBy) //method defination for joined
display
    {
        if (joined)
        {
            System.out.println("Employee already exists");
            System.out.println("Employee Details:");
            System.out.println("staffName"+getstaffName());
            System.out.println("joiningDate"+getjoiningDate());
        }
else
        {
            this.staffName=staffName;
this.joiningDate=joiningDate;
```

```
this.qualification=qualification;
this.appointedBy=appointedBy;
        this.joined=true;
        this.terminated=false;
    }

  }

  public void terminateStaff() //method for terminating staff
  {
    if (terminated)
    {
        this.staffName = "";
this.joiningDate = "";
this.qualification = "";          this.joined
= false;
        this.terminated = true;
    }
else
    {
        System.out.println("Employee not found");
    }
  }

  public void display() //display method
  {
    super.display();
if (joined)
    {
        System.out.println("staffName="+getstaffName());
        System.out.println("wagesPerHour="+getwagesPerHour());
        System.out.println("workingHour="+getworkingHour());
        System.out.println("joiningDate="+getjoiningDate());
        System.out.println("qualification="+getqualification());
System.out.println("appointedBy="+getappointedBy());          int
incomePerDay = getwagesPerHour() * getworkingHour();
        System.out.println("incomePerDay="+incomePerDay);
    }
else
    {
```

```
        System.out.println("No employee data");
    }
    }
}
```

9.4 INGNepal

```
import javax.swing.*; import
java.awt.*; import java.awt.event.*;
import java.util.ArrayList; import
java.awt.Color; import java.awt.Font;
import java.awt.FontFormatException;
import java.awt.GraphicsEnvironment;
import java.io.File; import
java.io.IOException;
public class INGNepal implements ActionListener
{
    JFrame frame;

    JPanel panelStaffHire;

    JLabel
labelFullTimeTitle,labelVacancyNumber,labelDesignation,labelJob,labelSa
lary,labelWorkingHour,labelStaffName,labelJoined,labelQualification,label
AppointedBy,labelPartTimeTitle,labelWages,labelShifts,labelTerminateSta
ff;

    JTextField
txtCheckFullTimeVacancyNumber,txtFullTimeVacancyNumber,txtFullTime
Designation,txtFullTimeJob,txtFullTimeSalary,txtFullTimeWorkingHour,txt
FullTimeStaffName,txtFullTimeJoined,txtFullTimeQualification,txtFullTime
AppointedBy;
JTextField
txtCheckPartTimeVacancyNumber,txtPartTimeVacancyNumber,txtPartTim
eDesignation,txtPartTimeJob,txtPartTimeWorkingHour,txtPartTimeStaffNa
me,txtPartTimeJoined,txtPartTimeQualification,txtPartTimeAppointedBy,tx
tWages,txtShifts,txtTerminateStaff;

    JButton
```

```
addFullTimeStaffButton,appointFullTimeStaffButton,addPartTimeStaffButt
on,appointPartTimeStaffButton,terminateStaffButton,clearButton,displayP
artTimeButton,displayFullTimeButton;
   ArrayList<StaffHire> list=new ArrayList<StaffHire>();

  public static void main(String []args){
      new INGNepal();
  }
   public INGNepal()
   {

       //To hire full time employee
frame = new JFrame("ING Nepal");
       Container container = frame.getContentPane();
       frame.setLayout(null);
panelStaffHire = new JPanel();
panelStaffHire.setBounds(0,0,900,900);
container.add(panelStaffHire);
       panelStaffHire.setLayout(null);
       frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

       labelFullTimeTitle = new JLabel("For Full Time Staff:");
labelFullTimeTitle.setFont (labelFullTimeTitle.getFont().deriveFont
(25.0f));
labelFullTimeTitle.setBounds(290,10,400,35);
       panelStaffHire.add(labelFullTimeTitle);

       labelVacancyNumber = new JLabel("Vacancy Number:");
labelVacancyNumber.setBounds(10,60,120,20);
       panelStaffHire.add(labelVacancyNumber);

       txtFullTimeVacancyNumber = new JTextField();
txtFullTimeVacancyNumber.setBounds(125,60,125,25);
panelStaffHire.add(txtFullTimeVacancyNumber);

       labelDesignation = new JLabel("Designation:");
labelDesignation.setBounds(290,60,80,20);
       panelStaffHire.add(labelDesignation);
```

```
        txtFullTimeDesignation = new JTextField();
    txtFullTimeDesignation.setBounds(380,60,125,25);
    panelStaffHire.add(txtFullTimeDesignation);

        labelSalary = new JLabel("Salary:");
    labelSalary.setBounds(10,95,60,20);
        panelStaffHire.add(labelSalary);

        txtFullTimeSalary = new JTextField();
        txtFullTimeSalary.setBounds(125,95,125,25);
        panelStaffHire.add(txtFullTimeSalary);

        labelJob = new JLabel("Job Type:");
    labelJob.setBounds(545,60,60,20);
        panelStaffHire.add(labelJob);

        txtFullTimeJob = new JTextField();
    txtFullTimeJob.setBounds(620,60,125,25);
        panelStaffHire.add(txtFullTimeJob);

        labelWorkingHour = new JLabel("Working Hour:");
    labelWorkingHour.setBounds(290,95,120,20);
        panelStaffHire.add(labelWorkingHour);

        txtFullTimeWorkingHour = new JTextField();
    txtFullTimeWorkingHour.setBounds(380,95,125,25);
        panelStaffHire.add(txtFullTimeWorkingHour);

        addFullTimeStaffButton = new JButton("Add");
    addFullTimeStaffButton.setBounds(125,130,125,35);
    addFullTimeStaffButton.addActionListener(this);
        panelStaffHire.add(addFullTimeStaffButton);

        //To appoint Full Time Staff GUI
        labelVacancyNumber = new JLabel("Vacancy Number:");
    labelVacancyNumber.setBounds(10,185,120,20);
    panelStaffHire.add(labelVacancyNumber);
```

```
        txtCheckFullTimeVacancyNumber = new JTextField();
txtCheckFullTimeVacancyNumber.setBounds(125,185,125,25);
panelStaffHire.add(txtCheckFullTimeVacancyNumber);

        labelStaffName = new JLabel("Staff Name:");
labelStaffName.setBounds(290,185,120,20);
        panelStaffHire.add(labelStaffName);

        txtFullTimeStaffName = new JTextField();
txtFullTimeStaffName.setBounds(380,185,125,25);
        panelStaffHire.add(txtFullTimeStaffName);

        labelJoined = new JLabel("Joined Date:");
        labelJoined.setBounds(285,220,120,20); panelStaffHire.add(labelJoined);
        txtFullTimeJoined = new JTextField();
        txtFullTimeJoined.setBounds(380,220,125,25);
        panelStaffHire.add(txtFullTimeJoined);

        labelQualification = new JLabel("Qualification:");
labelQualification.setBounds(545,185,100,20);
panelStaffHire.add(labelQualification);

        txtFullTimeQualification = new JTextField();
txtFullTimeQualification.setBounds(620,185,125,25);
panelStaffHire.add(txtFullTimeQualification);

        labelAppointedBy = new JLabel("Appointed By:");
labelAppointedBy.setBounds(10,220,120,20);
panelStaffHire.add(labelAppointedBy);

        txtFullTimeAppointedBy = new JTextField();
txtFullTimeAppointedBy.setBounds(125,220,125,25);
panelStaffHire.add(txtFullTimeAppointedBy);

        appointFullTimeStaffButton = new JButton("Appoint");
appointFullTimeStaffButton.setBounds(125,255,125,35);
appointFullTimeStaffButton.addActionListener(this);
panelStaffHire.add(appointFullTimeStaffButton);
```

```
        displayFullTimeButton = new JButton("Display");
displayFullTimeButton.setBounds(380,255,125,35);
displayFullTimeButton.addActionListener(this);
        panelStaffHire.add(displayFullTimeButton);

        //For Part Time Staff
        labelPartTimeTitle = new JLabel("For Part Time Staff:");
labelPartTimeTitle.setFont (labelFullTimeTitle.getFont().deriveFont
(25.0f));
        labelPartTimeTitle.setBounds(290,300,400,35);
panelStaffHire.add(labelPartTimeTitle);

        labelVacancyNumber = new JLabel("Vacancy Number:");
labelVacancyNumber.setBounds(10,345,120,20);
panelStaffHire.add(labelVacancyNumber);

        txtPartTimeVacancyNumber = new JTextField();
        txtPartTimeVacancyNumber.setBounds(125,345,125,25);
        panelStaffHire.add(txtPartTimeVacancyNumber);

        labelDesignation = new JLabel("Designation:");
labelDesignation.setBounds(290,345,80,20);
panelStaffHire.add(labelDesignation);

        txtPartTimeDesignation = new JTextField();
txtPartTimeDesignation.setBounds(380,345,125,25);
        panelStaffHire.add(txtPartTimeDesignation);

        labelJob = new JLabel("Job Type:");
labelJob.setBounds(545,345,80,20);
        panelStaffHire.add(labelJob);

        txtPartTimeJob = new JTextField();
txtPartTimeJob.setBounds(620,345,125,25);
        panelStaffHire.add(txtPartTimeJob);

        labelWages = new JLabel("Wages Per Hour:");
labelWages.setBounds(10,380,100,20);
        panelStaffHire.add(labelWages);
```

```
txtWages = new JTextField();
txtWages.setBounds(620,380,125,25);
panelStaffHire.add(txtWages);

labelWorkingHour = new JLabel("Working Hour:");
labelWorkingHour.setBounds(290,380,120,20);
panelStaffHire.add(labelWorkingHour);

txtPartTimeWorkingHour = new JTextField();
txtPartTimeWorkingHour.setBounds(380,380,125,25);
panelStaffHire.add(txtPartTimeWorkingHour);

labelShifts = new JLabel("Shifts:");
labelShifts.setBounds(545,380,40,20);
panelStaffHire.add(labelShifts);

txtShifts = new JTextField(); txtShifts.setBounds(125,380,125,25);
panelStaffHire.add(txtShifts);

addPartTimeStaffButton = new JButton("Add");
addPartTimeStaffButton.setBounds(125,415,125,35);
addPartTimeStaffButton.addActionListener(this);
panelStaffHire.add(addPartTimeStaffButton);

//To appoint Part Time Staff GUI
labelVacancyNumber = new JLabel("Vacancy Number:");
labelVacancyNumber.setBounds(10,470,120,20);
panelStaffHire.add(labelVacancyNumber);

txtCheckPartTimeVacancyNumber = new JTextField();
txtCheckPartTimeVacancyNumber.setBounds(125,470,125,25);
panelStaffHire.add(txtCheckPartTimeVacancyNumber);

labelStaffName = new JLabel("Staff Name:");
labelStaffName.setBounds(290,470,120,20);
panelStaffHire.add(labelStaffName);

txtPartTimeStaffName = new JTextField();
txtPartTimeStaffName.setBounds(380,470,125,25);
panelStaffHire.add(txtPartTimeStaffName);
```

```
        labelJoined = new JLabel("Joined Date:");
labelJoined.setBounds(10,505,100,20);
        panelStaffHire.add(labelJoined);

        txtPartTimeJoined = new JTextField();
txtPartTimeJoined.setBounds(125,505,125,25);
        panelStaffHire.add(txtPartTimeJoined);

        labelQualification = new JLabel("Qualification:");
labelQualification.setBounds(545,470,120,20);
        panelStaffHire.add(labelQualification);

        txtPartTimeQualification = new JTextField();
txtPartTimeQualification.setBounds(620,470,125,25);
panelStaffHire.add(txtPartTimeQualification);

        labelAppointedBy = new JLabel("Appointed By:");
labelAppointedBy.setBounds(290,505,120,20);
        panelStaffHire.add(labelAppointedBy);

        txtPartTimeAppointedBy = new JTextField();
txtPartTimeAppointedBy.setBounds(380,505,125,25);
        panelStaffHire.add(txtPartTimeAppointedBy);

        appointPartTimeStaffButton = new JButton("Appoint");
appointPartTimeStaffButton.setBounds(125,540,125,35);
appointPartTimeStaffButton.addActionListener(this);
        panelStaffHire.add(appointPartTimeStaffButton);

        displayPartTimeButton = new JButton("Display");
displayPartTimeButton.setBounds(380,540,125,35);
displayPartTimeButton.addActionListener(this);
        panelStaffHire.add(displayPartTimeButton);

        labelTerminateStaff = new JLabel("Vacancy Number:");
labelTerminateStaff.setBounds(10,595,120,20);
panelStaffHire.add(labelTerminateStaff);
```

```
        txtTerminateStaff = new JTextField();
txtTerminateStaff.setBounds(125,595,125,25);
        panelStaffHire.add(txtTerminateStaff);

        terminateStaffButton = new JButton("Terminate");
terminateStaffButton.setBounds(125,635,125,35);
terminateStaffButton.addActionListener(this);
        panelStaffHire.add(terminateStaffButton);

        clearButton = new JButton("Clear");
clearButton.setBounds(380,635,125,35);
clearButton.addActionListener(this);
        panelStaffHire.add(clearButton);

        frame.setSize(1000,700);
        frame.setVisible(true);

    }

    public void actionPerformed(ActionEvent event)
    {


        if(event.getSource()==addFullTimeStaffButton)
        {
try {
if(txtFullTi
meVacanc
yNumber.g
etText().is
Empty() ||
txtFullTime
Designatio
n.getText()
.isEmpty()
||
txtFullTimeJob.getText().isEmpty() || txtFullTimeSalary.getText().isEmpty()
|| txtFullTimeWorkingHour.getText().isEmpty()) {
                throw new NumberFormatException("TextField cannot be empty");
            }
```

```
                int fullTimeVacancyNumber =
Integer.parseInt(txtFullTimeVacancyNumber.getText());
                String fullTimedesignation = txtFullTimeDesignation.getText();
                String fullTimeJob = txtFullTimeJob.getText();
                int fullTimesalary =
Integer.parseInt(txtFullTimeSalary.getText());
int fullTimeworkingHour =
Integer.parseInt(txtFullTimeWorkingHour.getText());
                FullTimeStaffHire objectAddFullTime = new
FullTimeStaffHire(fullTimeVacancyNumber,fullTimedesignation,fullTimeJo
b,fullTimesalary,fullTimeworkingHour);
list.add(objectAddFullTime);
                JOptionPane.showMessageDialog(frame,"Full Time Staff has been
added");
            }
            catch(NumberFormatException exception)
            {

JOptionPane.showMessageDialog(frame,exception.getMessage());
            }
        }

        if(event.getSource()==addPartTimeStaffButton)
        {
try
            {
              if(txtPartTimeVacancyNumber.getText().isEmpty() ||
txtPartTimeDesignation.getText().isEmpty() ||
txtPartTimeJob.getText().isEmpty()  || txtShifts.getText().isEmpty() ||
txtWages.getText().isEmpty() || txtPartTimeWorkingHour.getText().isEmpty())
{
                throw new NumberFormatException("TextField cannot be empty");
              }
              int partTimevacancyNumber =
Integer.parseInt(txtPartTimeVacancyNumber.getText());
              String partTimedesignation = txtPartTimeDesignation.getText();
String partTimeJob = txtPartTimeJob.getText();              int
wagesPerHour = Integer.parseInt(txtWages.getText());
              int partTimeworkingHour =
Integer.parseInt(txtPartTimeWorkingHour.getText());
```

```
            String shifts = txtShifts.getText();
            PartTimeStaffHire objectAddPartTime = new
PartTimeStaffHire(partTimevacancyNumber,partTimedesignation,partTim
eJob,partTimeworkingHour,wagesPerHour,shifts);
list.add(objectAddPartTime);
            JOptionPane.showMessageDialog(frame,"Part Time Staff has been
added");
          }
          catch(NumberFormatException exception)
          {

JOptionPane.showMessageDialog(frame,exception.getMessage());
          }
        }

        if (event.getSource() == appointFullTimeStaffButton)
        {
try {
              if (txtCheckFullTimeVacancyNumber.getText().isEmpty() ||
txtFullTimeStaffName.getText().isEmpty() ||
txtFullTimeJoined.getText().isEmpty() ||
txtFullTimeQualification.getText().isEmpty() ||
txtFullTimeAppointedBy.getText().isEmpty()) {
                throw new NumberFormatException("TextField cannot be empty");
              }
              int checkFullTimeVacancyNumber =
Integer.parseInt(txtCheckFullTimeVacancyNumber.getText());
              String fullTimeStaffName = txtFullTimeStaffName.getText();
              String fullTimeJoined = txtFullTimeJoined.getText();
              String fullTimeQualification = txtFullTimeQualification.getText();
              String fullTimeAppointedBy = txtFullTimeAppointedBy.getText();

              for (StaffHire x:list)
              {
                if (x instanceof FullTimeStaffHire)
                {
                  FullTimeStaffHire f = (FullTimeStaffHire) x;
```

```
f.HirefulltimeStaff(fullTimeStaffName,fullTimeJoined,fullTimeQualification,f
ullTimeAppointedBy);
                JOptionPane.showMessageDialog(frame,"Full Time Staff has been
appointed");
              }
            }
          }
          catch(NumberFormatException exception)
          {

JOptionPane.showMessageDialog(frame,exception.getMessage());
          }
        }

        if (event.getSource() == appointPartTimeStaffButton)
        {
try {
            if (txtCheckPartTimeVacancyNumber.getText().isEmpty() ||
txtPartTimeStaffName.getText().isEmpty() ||
txtPartTimeJoined.getText().isEmpty() ||
txtPartTimeQualification.getText().isEmpty() ||
txtPartTimeAppointedBy.getText().isEmpty()) {
                throw new NumberFormatException("TextField cannot be empty");
            }
            int checkPartTimeVacancyNumber =
Integer.parseInt(txtCheckPartTimeVacancyNumber.getText());
            String partTimeStaffName = txtPartTimeStaffName.getText();
            String partTimeJoined = txtPartTimeJoined.getText();
            String partTimeQualification =
txtPartTimeQualification.getText();
String partTimeAppointedBy =
txtPartTimeAppointedBy.getText();
            for (StaffHire y:list)
            {
              if (y instanceof PartTimeStaffHire)
              {
                PartTimeStaffHire objPartTime = (PartTimeStaffHire) y;
```

```
objPartTime.PartTimeStaffHire(partTimeStaffName,partTimeJoined,partTi
meQualification,partTimeAppointedBy);
JOptionPane.showMessageDialog(frame,"Part Time Staff has been appointed");
            }
          }
        }
        catch(NumberFormatException exception)
        {

JOptionPane.showMessageDialog(frame,exception.getMessage());
        }
      }

      if (event.getSource() == displayFullTimeButton)
      {
        for (StaffHire a:list)
        {
          if (a instanceof FullTimeStaffHire)
          {
            FullTimeStaffHire objDisplayFullTime = (FullTimeStaffHire) a;
objDisplayFullTime.display();
          }
        }
      }

      if (event.getSource() == displayPartTimeButton)
      {
        for (StaffHire b:list)
        {
          if (b instanceof PartTimeStaffHire)
          {
            PartTimeStaffHire objDisplayPartTime = (PartTimeStaffHire) b;
objDisplayPartTime.display();
          }
        }
      }

      if (event.getSource()==clearButton)
      {
        if (txtCheckFullTimeVacancyNumber.getText().isEmpty() == false)
```

```
{
  txtCheckFullTimeVacancyNumber.setText("");
}
if (txtFullTimeVacancyNumber.getText().isEmpty() == false)
{
  txtFullTimeVacancyNumber.setText("");
}
if (txtFullTimeDesignation.getText().isEmpty() == false)
{
  txtFullTimeDesignation.setText("");
}
if (txtFullTimeJob.getText().isEmpty() == false)
{
  txtFullTimeJob.setText("");
}
if (txtFullTimeSalary.getText().isEmpty() == false)
{
  txtFullTimeSalary.setText("");
}
if (txtFullTimeWorkingHour.getText().isEmpty() == false)
{
  txtFullTimeWorkingHour.setText("");
}
if (txtFullTimeStaffName.getText().isEmpty() == false)
{
  txtFullTimeStaffName.setText("");
}
if (txtFullTimeJoined.getText().isEmpty() == false)
{
  txtFullTimeJoined.setText("");
}
if (txtFullTimeQualification.getText().isEmpty() == false)
{
  txtFullTimeQualification.setText("");
}
if (txtFullTimeAppointedBy.getText().isEmpty() == false)
{
  txtFullTimeAppointedBy.setText("");
}
if (txtCheckPartTimeVacancyNumber.getText().isEmpty() == false)
```

```
{
  txtCheckPartTimeVacancyNumber.setText("");
}
if (txtPartTimeVacancyNumber.getText().isEmpty() == false)
{
  txtPartTimeVacancyNumber.setText("");
}
if (txtPartTimeDesignation.getText().isEmpty() == false)
{
  txtPartTimeDesignation.setText("");
}
if (txtPartTimeJob.getText().isEmpty() == false)
{
  txtPartTimeJob.setText("");
}
if (txtPartTimeWorkingHour.getText().isEmpty() == false)
{
  txtPartTimeWorkingHour.setText("");
}
if (txtPartTimeStaffName.getText().isEmpty() == false)
{
  txtPartTimeStaffName.setText("");
}
if (txtPartTimeJoined.getText().isEmpty() == false)
{
  txtPartTimeJoined.setText("");
}
if (txtPartTimeQualification.getText().isEmpty() == false)
{
  txtPartTimeQualification.setText("");
}
if (txtPartTimeAppointedBy.getText().isEmpty() == false)
{
  txtPartTimeAppointedBy.setText("");
}
if (txtWages.getText().isEmpty() == false)
{
  txtWages.setText("");
}
if (txtShifts.getText().isEmpty() == false)
```

```
                {
                  txtShifts.setText("");
                }
                if (txtTerminateStaff.getText().isEmpty() == false)
                {
                  txtTerminateStaff.setText("");
                }
              }
            }
        }
```