



**Module Code & Module Title**

**CS4001NT Programming**

**Assessment Weightage & Type**

**50% Individual Coursework**

**Year and Semester**

**2019-20 Autumn / 2019-20 Spring**

**Student Name: SUJAN CHAUDHARY**

**London Met ID: 19031885**

**College ID:NP05CP4A190172**

**Assignment Due Date:**

**Assignment Submission Date:2020/06/05**

## Contents

1.0 java.....	1
1.1 Abstract class.....	1
1.2 Interfaces .....	2
1.3 Action listener .....	2
1.4Java swing .....	2
1.5 Number Format Exception.....	2
1.6 Exception handling .....	2
1.7 Event handling.....	3
2.0 Class Diagram- .....	3
3.0 Pseudo code – .....	5
3.1 Pseudocode: .....	5
4.0 A short description of what each method does .....	25
a. StaffHire Class.....	25
b. FullTimeStaffHire Class .....	25
c. PartTimeStaffHire class .....	25
d. INGNepal class.....	26
1. Public INGNepal().....	26
2.Public initializeFrame().....	26
3.Public initializeMenu() .....	26
4.Public initializeBody().....	26
5.public void actionPerformed (ActionEvent ae) .....	26
6.forFullTimeStaffHire() .....	26
7.Submit().....	26
8.Appoint() .....	26
9.forHaltTimeStaffHire() .....	27
10.Submit1().....	27
11.Appoint1() .....	27
12.Terminate().....	27
13.Display().....	27
14.public static void main(String [ ] args).....	27
5.0 Test .....	27
5.1 test-1 .....	27

5.2 Test-2 .....	28
5.3 Test-3 .....	29
5.4 Test-4 .....	30
5.5 Test-5 .....	31
5.6 Test-6 .....	32
6.0 Error .....	33
6.1 Syntax error .....	33
6.2 Semantics error .....	34
6.3 Run time error .....	34
7.0 Appendix .....	35
8.0 Conclusion .....	59
9.0 References.....	60
Bibliography .....	60

Figure 1:class diagram .....	4
Figure 2:test1 .....	28
Figure 3:test2 .....	29
Figure 4:test3 .....	30
Figure 5:test4 .....	31
Figure 6:test5 .....	32
Figure 7:test6 .....	33
Figure 8:syntax error .....	34
Figure 9:simantic error .....	34
Figure 10:run time error .....	35

## 1.0 java

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

The Java syntax is similar to C++, but is an object-oriented programming language. For example, most Java programs contain classes, which are used to define objects, and methods, which are assigned to individual classes. Java is also known for being more strict than C++, meaning variables and functions must be explicitly defined. This means Java source code may produce errors or "exceptions" more easily than other languages, but it also limits other types of errors that may be caused by undefined variables or unassigned types.

Java has many characteristics some of them are listed below:

- Platform independence - many languages are compatible with only one platform. Java was specially designed so that it would run on any computer, regardless if it was running windows, Linux, Mac or any other operating system.
- Simple and easy to use – java's creators tried to design it so code could be written efficiently and easily.
- Multi-Functional - java can produce many applications from command line programs to applets to swing windows. It has have some drawbacks. Since it has automated garbage collection, it can tend to use more memory then other similar language.

(techterm, n.d.)

## 1.1 Abstract class

A class which is declared with the abstract keyword is known as an abstract class in Java. It can have both abstract and non-abstract methods. It needs to be extended and its method implemented. It cannot be instantiated. Abstract class can have final, non-final and static, non-static variable. It can have class members like public, private protected.

(javatpoint, 2018)

## 1.2 Interfaces

An interface in Java is a blueprint of a class. It has static constants and abstract methods. It is a mechanism to achieve abstraction in java. There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple inheritance in Java.

## 1.3 Action listener

The Java ActionListener is notified whenever you click on the button or menu item. It is notified against ActionEvent. The ActionListener interface is found in java.awt.event package It has only one method: actionPerformed().

Example;

```
Public void actionPerformed(ActionEvent ae){
```

(javatpoint, n.d.)

## 1.4Java swing

Java swing is used to create window-based applications. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java.

The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

(javatpoint, n.d.)

## 1.5 Number Format Exception

A Java Number Format Exception usually occurs when you try to do something like convert a String to a numeric value, like an int, float, double, long, etc.

## 1.6 Exception handling

The Exception Handling plays vital role in programming which allows us to handle the runtime error caused by exceptions. It ensures that the flow of the program does not break when an exception occurs.

Two keywords used for java exception handling are:

1. try
2. catch

(beginnersbook, n.d.)

### 1.7 Event handling

Event Handling is the mechanism that controls the event and decides what should happen if an event occurs. This mechanism has the code which is known as event handler that is executed when an event occurs. Java Uses the Delegation Event Model to handle the event.

The java.awt.event package provides many event classes and Listener interfaces for event handling.

(tutorialpoint, n.d.)

### 2.0 Class Diagram-

A class diagram Class diagram are one of the most useful types of diagrams in UML (unfiled modeling language) as they clear map out the structure of a particular system by modeling its classes, attributes, operating and relationships between objects. It illustrate data models for information systems no matter how complex or simple.

INGNepal
-Frame: JFrame -panel1: JPanel -panel2: JPanel -panel3: JPanel -btn1: JButton -btn2: JButton -btnclear: JButton -btnclear2: JButton -btnappoint: JButton -btnappoint2: JButton -btnsubmit: JButton -btnsubmit2: JButton -btndisplay: JButton -btnterminate: JButton -btnback1: JButton -btnback2: JButton -tfvacancy1: JTextField -tfdesignation: JTextField -tfjobType: JTextField -tfworkingHour: JTextField -tfsalary: JTextField -tfvacancy2: JTextField -tffstaffName: JTextField -tfjoiningDate: JTextField -tfappointedBy: JTextField -tfqualification: JTextField -tfwagesPerHour: JTextField -tfshifts: JTextField -tfvacancy3: JTextField -lbl1: JLabel -lbl2: JLabel -lbvacancy1: JLabel -lbdesignation: JLabel -lbjobType: JLabel -lbworkingHour: JLabel -lbsalary: JLabel -lbvacancy2: JLabel -lbstaffName: JLabel -lbjoiningDate: JLabel -lbappointedBy: JLabel -lbqualification: JLabel -lbwagesPerHour: JLabel -lbshifts: JLabel -lbvacancy3: JLabel
+INGNepal () -initializeFrame (): void -initializeMenu (): void -initializeBody (): void +forFullTimeStaffHire():void +submit():void +appoint():void +display():void +forPartTimeStaffHire():void +clear1(): void +submit1():void +appoint1():void +display1():void +terminateStaff():void

Figure 1: class diagram



### 3.0 Pseudo code –

Pseudo code is an informal way of programming illustration that does not require any strict programming language syntax or underlying technology considerations. It is used for making an outline or a rough draft of a program. It summarizes a programs flow, but excludes underlying details. Programmers write pseudo code to understand the requirement and align code accordingly.

#### 3.1 Pseudocode:

// constructor

FUNCTION INGNepal():

    CALL initializeFrame()

CALL initializeMenu()

    CALL initializeBody()

END

FUNCTION initializeFrame():

CREATE a JFrame title "INGNepal"

    SET bound of frame to 0,0,900, 700

    SET null layout

    SET EXIT\_ON\_CLOSE DefaultCloseOperation

    SET Resizable false

    SET default close operation to EXIT\_ON\_CLOSE

    END

FUNCTION initializeMenu():

CREATE a JmenuBar

SET Bounds 0,0,1095,26

GET ContentPane Add file to ContentPanel

CREATE a JMenu (file);

Add Menu to Menubar

```
CREATE a JMenuItem(new);
Add JMenuItem to MenuBar
ADD Seperator to MenuBar
ADD ActionListener to minew
FUNCTION new
IF(new is clicked)
    DO
    SetText textfieldvacancyNumber to empty string
    SetText textfielddesignation to empty string
    SetText textfieldjobType to empty string
    SetText textfieldworkingHour to empty string
    SetText textfieldsalary to empty string
    SetText textfieldvacancyNumber to empty string
    SetText textfieldstaffName to empty string
    SetText textfieldjoiningDate to empty string
    SetText textfieldappointedBy to empty string
    SetText textfieldqualification to empty string
    SetText textfieldwagesPerHour to empty string
END
SetText textfieldshifts to empty string
ADD ActionListener to minew
FUNCTION new
IF(new is clicked)
    DO
        Exit close the Frame
    END
ADD about to MenuBar
ADD ActionListener to miapp
```

FUNCTION

IF(App is clicked)

DO

Show information messageDialogbox("App:version 2.0")

END

ADD ActionListener to mideveloper

FUNCTION

IF(Developer is clicked)

DO

Show information messageDialogbox("App is desingned by sujan  
chaudhary")

END

FUNCTION initializeBody():

CREATE a Panel1 Add in frame

ADD Panel1 to Frame

SET size of frame to 900,700

SET null layout

// for FullTimeStaffHire button1

CREATE a JButton "FullTimeStaffHire"

SET bounds to 300,250,200,50

ADD btn1 to Panel1

ADD ActionListener to the button1

    FUNCTION actionPerformed(ActionEvent e):

        SET visibility of frame to False

        CALL forFullTimeStaffHire()

// for ParTimeStaffHirebutton2

CREATE a JButton "PartTimeStaffHire"

SET bounds to 300,350,200,50

ADD btn2 to Panel1

ADD ActionListener to the button2

    FUNCTION actionPerformed(ActionEvent e):

        SET visibility of frame to False

        CALL forPartTimeStaffHire ()

END

Function FullTimeStaffHire():

CREATE a Panel2 Add in Frame

    ADD Panel2 to Frame

    SET size of Frame to 600,500

    SET null layout

```
CREATE a JLabel "VacancyNumber:"  
    ADD label to Panel2
```

```
CREATE a JLabel "Designation:"  
    ADD label to Panel2
```

```
CREATE a JLabel " Salary:"  
    ADD label to Panel2
```

```
CREATE a JLabel "WorkingHour:"  
    ADD label to Panel2
```

```
CREATE a JLabel "JobType:"  
    ADD label to Panel2
```

```
CREATE a JLabel "VacancyNumber:"  
    ADD label to Panel2
```

```
CREATE a JLabel "staffName:"  
    ADD label to Panel2
```

```
CREATE a JLabel "joiningDate:"  
    ADD label to Panel2
```

```
CREATE a JLabel "Qualification:"  
    ADD label to Panel2
```

CREATE a JLabel "AppointedBy:"

ADD label to Panel2

CREATE a JTextField"VacancyNumber:"

ADD label to Panel2

CREATE a JTextField "Designation: "

ADD label to Panel2

CREATE a JTextField " Salary:"

ADD label to Panel2

CREATE a JTextField"WorkingHour:"

ADD label to Panel2

CREATE a JTextField"JobType:"

ADD label to Panel2

CREATE a JTextField"VacancyNumber:"

ADD label to Panel2

CREATE a JTextField "StaffName:"

ADD label to Panel2

CREATE a JTextField "JoiningDate:"

ADD label to Panel2

CREATE a JTextField"AppointedBy:"

ADD label to Panel2

CREATE a JTextField"Qualification:"

ADD label to Panel2

CREATE a JTextField"For Full Time Staff "

ADD label to Panel2

//bounding for text filed

VacancyNumber SET bounds 50,100,100,40

Designation SET Bounds 50,150,100,40

jobType SET Bounds 50,200,100,40

salary SET Bounds 50,250,100,40

workingHour SET Bounds 50,300,100,40

vacancynumber SET Bounds 450,100,100,40

staffName SET Bounds 450,150,100,40

joiningDate SET Bounds 450,200,100,40

appointedBy SET Bounds 450,250,100,40

qualification SET Bounds 450,300,100,40

//bounding for Text fields

VacancyNumber SET bounds 150,105,170,30

Designation SET Bounds 150,155,170,30

jobType SET Bounds 150,200,170,30

salary SET Bounds 150,250,100,30

workingHour SET Bounds 150,300,170,30

vacancynumber SET Bounds 550,105,100,30

staffName SET Bounds 550,155,170,30

joiningDate SET Bounds 550,205,170,30  
appointedBy SET Bounds 550,255,170,30  
qualification SET Bounds 550,305,170,30

CREATE a JButton "Clear"

ADD button to Panel2

SET bounds to 50,405,100,40

ADD ActionListener to the button

FUNCTION actionPerformed(ActionEvent e):

CALL clearForFullTime()

DO

setText of textVacancyNumberFullTimeStaffHire to empty String

setText of textDesignationFullTimeStaffHire to empty String

setText of textJobTypeFullTimeStaffHire to empty String

setText of textSalaryFullTimeStaffHire to empty String

setText of textworkingHourFullTimeStaffHire to empty String

setText of textStaffNameFullTimeStaffHire to empty String

setText of textJoiningDateFullTimeStaffHire to empty String

setText of textQualificationFullTimeStaffHire to empty String

setText of textAppointedByFullTimeStaffHire to empty String

END DO

END IF



CREATE a JButton "submit"

ADD button to Panel2

SET bounds to 350,140,100,30

ADD ActionListener to the button

FUNCTION actionPerformed(ActionEvent e):

CALL submit()

CREATE a JButton "Appoint"

ADD button to Frame Panel2

SET bounds to 600,405,100,40

ADD ActionListener to the button

FUNCTION actionPerformed(ActionEvent e):

CALL appoint()

CREATE a JButton "Back"

ADD button to Frame Panel2

ADD button to Frame Panel2

ADD ActionListener to the button

SET Bounds(350,430,100,40);

FUNCTION actionPerformed(ActionEvent e):

SET panel1 Visible true

SET panel2 Visible false

CREATE a JButton "Display"

SET bounds to 350,340,100,30

ADD button to frame Panel2

ADD ActionListener to the button

FUNCTION actionPerformed(ActionEvent e):

CALL display()

END

FUNCTION submit()

GET tfvacancyNumber as String

GET tfDesignation as String

GET tfJobType as String

GET tfvacancyNumber as String

GET tfWorkingHour as String

GET tfSalary as String

If(VacancyNumber=" "),(Designation=" "),(JobType=" "),(WorkingHour=" "),(Salary=" ")

SHOW dialog box "please fill all the fields"

TRY PARSEINT vc,wh,sy

CATCH NumberFormatException

SHOW dialog box "give correct input."

FOR (StaffHire s: abc)

If (s instanceof FullTimeStaffHire)

FullTimeStaffHire ft= (FullTimeStaffHire) s;

If (VacancyNumber = vc)

SHOW dialog box "give same vacancy no."

ADD FullTimStaffHire to arraylist named as abc

Show dialog box "Succuessfully added."

FUNCTION appoint()

    GET tfStaffName as String

    GET tfJoininigDate as String

    GET tfQualification as String

    GET tfAppointedBy as String

    GET tfVacancyNumber as String

    IF(StaffName=" "), (JoiningDate=" "), (Qualification=" "), (AppointedBy=" "), (VacancyNumber=" ")

        Show dialog box "please filll all the fields"

    TRY PARSEINT VC

    CATCH NumberFormatException

        Show dialog box "give correct input"

    BOOLEAN FOUND = false

    FOR (int i = 0; i < abc.size(); ++i)

        GET StaffHire in ArrayList named abc

        If(sh instanceof FullTimeStaffHire)

            FullTimeStaffHire full = (FullTimeStaffHire) sh

            IF(VacancyNumber=vc)

                Show dialog box "Successfully appointed"

                Found= true

            Break

        IF(not found)

            Show dialog box "vacancy number not found"

FUNCTION display()

```
FOR(StaffHire: abc)
  IF(hire instanceof FullTimeStaffHire)
    FullTimeStaffHire full = (FullTimeStaffHire) hire
    Display;
```

Function forPartTimeStaffHire():

CREATE a Panel3 Add in Frame

ADD Panel3 to Frame

SET size of Frame to 600,500

SET null layout

CREATE a JLabel "VacancyNumber:"

ADD label to Panel3

CREATE a JLabel "Designation:"

ADD label to Panel3

CREATE a JLabel "JobType:"

ADD label to Panel3

CREATE a JLabel " WagesPerHour:"

ADD label to Panel3

CREATE a JLabel "WorkingHour:"

ADD label to Panel3

CREATE a JLabel " Shifts:"

ADD label to Panel3

CREATE a JLabel "VacancyNumber:"

ADD label to Panel3

CREATE a JLabel "staffName:"

ADD label to Panel3

CREATE a JLabel "joiningDate:"

ADD label to Panel3

CREATE a JLabel "AppointedBy:"

ADD label to Panel3

CREATE a JLabel "Qualification:"

ADD label to Panel3

CREATE a JTextField "VacancyNumber:"

ADD label to Panel3

CREATE a JTextField "Designation: "

ADD label to Panel3

CREATE a JTextField"JobType:"

ADD label to Panel3

CREATE a JTextField"WagesPerHour:"

ADD label to Panel3

CREATE a JTextField"WorkingHour:"

ADD label to Panel3

CREATE a JTextField " Shifts:"

ADD label to Panel3

CREATE a JTextField"VacancyNumber:"

ADD label to Panel3

CREATE a JTextField "StaffName:"

ADD label to Panel3

CREATE a JTextField "JoiningDate:"

ADD label to Panel2

CREATE a JTextField"AppointedBy:"

ADD label to Panel2

CREATE a JTextField"Qualification:"

ADD label to Panel2

CREATE a JTextField"For Part Time Staff "

ADD label to Panel3

//bounding for labels

VacancyNumber SET bounds 50,100,100,40

Designation SET Bounds 50,150,100,40

jobType SET Bounds 50,200,100,40

wagesPerHour SET Bounds 50,250,100,40

workingHour SET Bounds 50,300,100,40

shifts SET Bound 50,350,100,40

vacancynumber SET Bounds 450,100,100,40

staffName SET Bounds 450,150,100,40

joiningDate SET Bounds 450,200,100,40

appointedBy SET Bounds 450,250,100,40

qualification SET Bounds 450,300,100,40

//bounding for Text fields

VacancyNumber SET bounds 150,105,170,30

Designation SET Bounds 150,155,170,30

jobType SET Bounds 150,200,170,30

wagesPerHour SET Bounds 150,250,100,30

workingHour SET Bounds 150,300,170,30

shifts SET Bounds 150,355,170,30

vacancynumber SET Bounds 550,105,100,30

staffName SET Bounds 550,155,170,30

joiningDate SET Bounds 550,205,170,30

appointedBy SET Bounds 550,255,170,30

qualification SET Bounds 550,305,170,30

CREATE a JButton "Clear"

ADD button to Panel3

SET bounds to 50,405,100,40

ADD ActionListener to the button

FUNCTION actionPerformed(ActionEvent e):

CALL clear1()

DO

setText of textVacancyNumberPartTimeStaffHire to empty String

setText of textDesignationPartTimeStaffHire to empty String

setText of textJobTypePartTimeStaffHire to empty String

setText of textShiftsPartTimeStaffHire to empty String

setText of textWagesPerHourPartTimeStaffHire to empty String

setText of textWorkingHourPartTimeStaffHire to empty String

setText of textStaffNamePartTimeStaffHire to empty String

setText of textJoiningDatePartTimeStaffHire to empty String

setText of textAppointedByPartTimeStaffHire to empty String

setText of textStaffNamePartTimeStaffHire to empty String

setText of textQualificationPartTimeStaffHire to empty String

END DO

END IF



CREATE a JButton "submit"

ADD button to Panel3

SET bounds to 50,450,100,40

ADD ActionListener to the button

FUNCTION actionPerformed(ActionEvent e):

CALL submit1()

CREATE a JButton "Back"

ADD button to Frame Panel2

ADD button to Frame Panel2

ADD ActionListener to the button

SET Bounds(350,430,100,40);

FUNCTION actionPerformed(ActionEvent e):

SET panel1 Visible true

SET panel3 Visible false

CREATE a JButton "Appoint"

ADD button to Frame Panel3

SET bounds to 600,405,100,40

ADD ActionListener to the button

FUNCTION actionPerformed(ActionEvent e):

CALL appoint1()

CREATE a JButton "terminate"

ADD button to frame panel3

SET Bounds 350,600,100,40

ADD ActionListener to the button

```
    FUNCTION actionPerformed(ActionEvent e):  
    CALL terminateStaff()
```

CREATE a JButton "Display"

SET bounds to 350,340,100,30

ADD button to frame Panel2

ADD ActionListener to the button

```
    FUNCTION actionPerformed(ActionEvent e):  
    CALL display1()
```

END

FUNCTION submit1()

GET tfDesignation as String

GET tfShifts as String

GET tfvacancyNumber as String

GET tfWorkingHour as String

GET tfWagesPerHour as String

GET tfJobType as String

```
    If(VacancyNumber=" "),(Designation=" "),(JobType=" "),(WorkingHour=" "  
    "),(WagesPerHour=" "),(Shifts=" ")
```

SHOW dialog box "please fill all the fields"

TRY PARSEINT vc,wh,wp

CATCH       NumberFormatException

SHOW dialog box "give correct input."

FOR (StaffHire s: abc)

If (s instanceof PartTimeStaffHire)

PartTimeStaffHire pt= (PartTimeStaffHire) s;

    If (VacancyNumber = vc)

SHOW dialog box "give same vacancy no."

ADD PartTimeStaffHire to arraylist named as abc

Show dialog box "Successfully added."

FUNCTION appoint1()

    GET tfStaffName as String

    GET tfJoiningDate as String

    GET tfQualification as String

    GET tfAppointedBy as String

    GET tfVacancyNumber as String

    IF (StaffName=" "),(JoiningDate=" "),(Qualification=" "),(AppointedBy=" "),(VacancyNumber=" ")

        Show dialog box "please fill all the fields"

TRY PARSEINT VC

CATCH NumberFormatException

    Show dialog box "give correct input"

BOOLEAN FOUND = false

FOR (int i = 0; i < abc.size(); ++i)

    GET StaffHire in ArrayList named abc

    If (sh instanceof PartTimeStaffHire)

```
PartTimeStaffHire full = (PartTimeStaffHire) sh
```

```
IF(VacancyNumber=vc)
```

```
Show dialog box"Successfully appointed"
```

```
Found= true
```

```
Break
```

```
IF(not found)
```

```
Show dialog box"vacancy number not found"
```

```
FUNCTION terminateStaff
```

```
GET tfVacancyNumber as String
```

```
IF(VacancyNumber)
```

```
Show dialog box"please fill all the field"
```

```
TRY PARSEINT vc
```

```
CATCH NumberFormatException
```

```
Show dialog box"please give data in correct format"
```

```
RETURN
```

```
BOOLEAN found= false
```

```
FOR(StaffHire hire sh : abc)
```

```
If(sh instanceof PartTimeStaffHire)
```

```
    PartTimeStaffHire pt= (PartTimeStaffHire)sh
```

```
    If(VacancyNumber=vc)
```

```
        pt.terminateStaff()
```

```
        abc.remove(pt)
```

```
Show dialog box"sucessfully terminated"
```

```
Found= true7
```

```
If(not found)
```

```
Show dialog box"vacancy number not found"
```

```
FUNCTION display1()
```

```

FOR(StaffHire: abc)
    IF(hire instanceof PartTimeStaffHire)
        PartTimeStaffHire full = (PartTimeStaffHire) hire
        Display;

```

## 4.0 A short description of what each method does

### a. StaffHire Class

- **StaffHire** It is the method which has accessor method getter and setter
- **Display method** A display method displays details of the vacancy number, designation, and jobtype.

### b. FullTimeStaffHire Class

- **FullTimeStaffHire** This method accepts five parameters staffname, joiningdate, qualification, and appointed by as an empty string, joined status is initialize to false. This method changes the salary of staff. The method changes working hour to new working hour as a parameter by assigning anew value.
- **Display method**

This method display a details only if staff is joined then staffname,salary , workinghour,joiningdate, qualification and appointed by also display.

### c. PartTimeStaffHire class

- **partTimeStaffhire** This method accepts six parameters which are the vacancy Number, designation, jobType, working Hour, Wages per hour, and Shifts. Additionally in the constructor assigning the value of staffname, joining date, qualification, appointed by as an empty string joined and terminate status is initialized to false. This method changes the working shift of the staff. If the staff is already hire, then method doesn't allow to set new shift.

This method also hire part time staff which accepts staff name joining date, qualification and appointed by. The joining status of the staff is changed to true and termination status of staff is changed to false.

This method accepts the staff name, joiningdate, qualification, and appointed By and update by the parameter values input to the method and joining status of the staff is changed to true and termination status of the staff is changed to true.

- Terminate This method terminate staff if the staff is already joined.
- Display method This method displays all the detail of the part time staff.

#### d. INGNepal class

##### 1. Public INGNepal()

- The method which is given described instruction which are called for execution using method name. It is also used for creating GUI(Graphical User Interface) of the program. In this method I had described the different panel and size of frame and bounds for label, textfield, button etc.

##### 2.Public initializeFrame()

- In this method I had described about name of frame. In this method described about layout of frame, size of frame, bounding of frame.

##### 3.Public initializeMenu()

- In this method add menu bars and menu items by giving proper bounding to look give GUI attractive.

##### 4.Public initializeBody()

- In this method all the create panels and set the size of panels. The layout, size, color, titled and bounding of all 3 panel are describd here. Then add two buttons in the frame by giving bounding for each buttons and named FullTimeStaffHire and HalfTimeStaffHire.

##### 5.public void actionPerformed (ActionEvent ae)

in this method all buttons are made responsive like for submit button it has only task to submit and there are other button like terminate, appoint, back, clear etc which it own task to perform which don within this method.

##### 6.forFullTimeStaffHire()

- in this method i add all the text fields and labels by giving proper bounding.

##### 7.Submit()

- In this method submit button is responsive made responsive.

##### 8.Appoint()

- In this method appoint button is made responsive for appointing full time staffs

## 9.forHaltTimeStaffHire()

- in this method i add all the text fields and labels by giving proper bounding.

## 10.Submit1()

- In this method submit button is responsive made responsive.

## 11.Appoint1()

- In this method appoint button is made responsive for appointing halt time staffs.

## 12.Terminate()

In this method the details of half time staff is get terminated.

## 13.Display()

- this method is used to display all the details we fill in the textfiled of both FullTimeStaffHire and HalfTimeStaffHire in the GUI by clicking button display.

## 14.public static void main(String [ ] args)

- This is the main method where the object of INGNepal class is created and initialized.

## 5.0 Test

## 5.1 test-1

.

Objective	Opening a java file from command prompt
Action	Running a program from command prompt

Expected Result	GUI from command prompt
Actual Result	GUI has displayed
Conclusion	Test was successfully tested

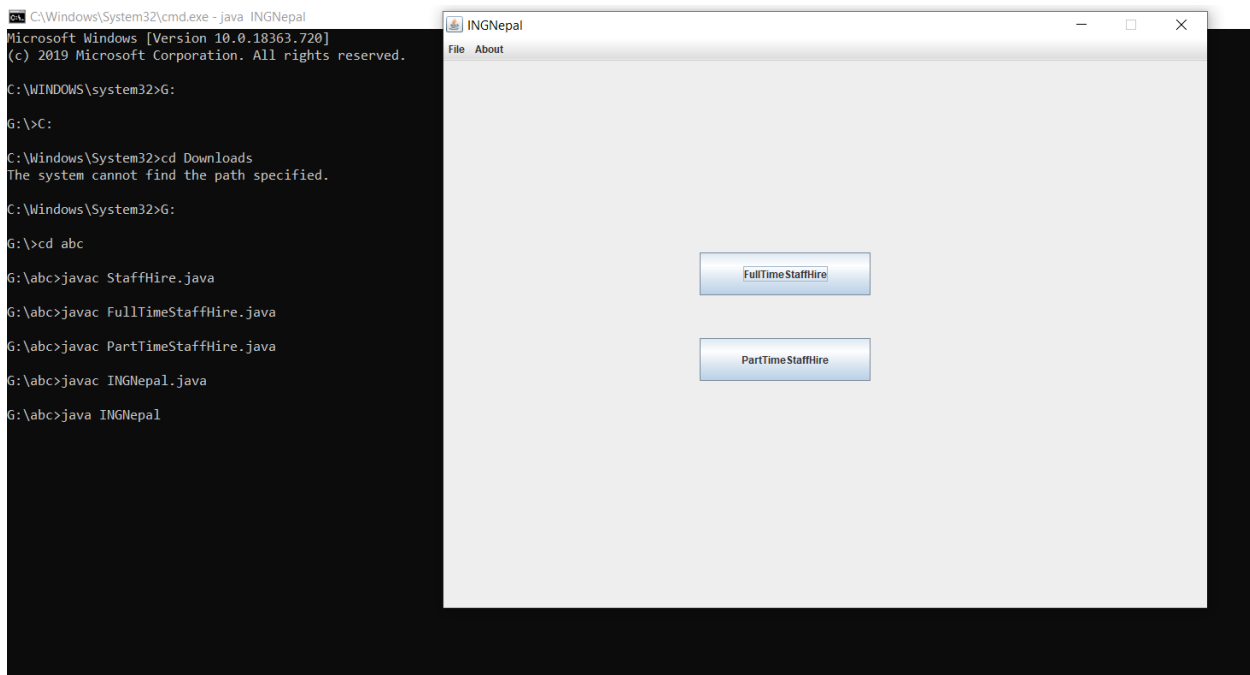


Figure 2:test1

## 5.2 Test-2

Objective	Adding vacancy number for Full Time Staff
Action	Vacancy number is add in textfiled
Expected Result	Dialog box of information message will appear
Actual Result	Dialog box was appeared



Conclusion	Test was successfully tested
------------	------------------------------

**For Full Time Staff**

Vacancy number:  Vacancy Number:

Designation:  Staff Name:

Job type:  Joining Date:

Salary:

WorkingHours:

Message ×


 **Succuessfully added.**

Figure 3:test2

### 5.3 Test-3

Objective	Adding vacancy number for Part Time Staff
Action	Vacancy number is add in textfiled
Expected Result	Dialog box of information message will appear
Actual Result	Dialog box was appeared
Conclusion	Test was successfully tested

**For Part Time Staff**

Vacancy number: <input type="text" value="1"/>	Vacancy Number: <input type="text"/>
Designation: <input type="text" value="doctor"/>	Staffname: <input type="text"/>
Job type: <input type="text" value="part"/>	Joining Date: <input type="text"/>
WagesPerHour: <input type="text" value="150"/>	<input type="text"/>
WorkingHours: <input type="text" value="6"/>	<input type="text"/>
shifts: <input type="text" value="day"/>	<input type="text"/>

×

**Message**

Succuessfully added.

Figure 4:test3

## 5.4 Test-4

Objective	Appointing a staff of full time staff
Action	Inserting all the value in textfield
Expected Result	Dialog box of information message will appear
Actual Result	Dialog box was appeared of successfully appointed
Conclusion	Test was successfully tested

**For Full Time Staff**

Vacancy number: <input type="text" value="1"/>	Vacancy Number: <input type="text" value="1"/>
Designation: <input type="text" value="doctor"/>	Staff Name <input type="text" value="sujan"/>
Job type: <input type="text" value="full time"/>	Joining Date: <input type="text" value="2075-05-15"/>
Salary: <input type="text" value="4500"/>	<input type="text" value=""/>
WorkingHours: <input type="text" value="12"/>	<input type="text" value=""/>

clear

submit

Back

Appoint

Display

×

**Message**

**Succuessfully appointed.**

Figure 5:test4

## 5.5 Test-5

Objective	Appointing a staff of Part time staff
Action	inserting all the value in textfield
Expected Result	Dialog box of information message will appear
Actual Result	Dialog box was appeared of successfully appointed
Conclusion	Test was successfully tested

**For Part Time Staff**

Vacancy number: <input type="text" value="1"/>	Vacancy Number: <input type="text" value="1"/>
Designation: <input type="text" value="doctor"/>	Staffname: <input type="text" value="sujan"/>
Job type: <input type="text" value="part"/>	Joining Date: <input type="text" value="2075-05-12"/>
WagesPerHour: <input type="text" value="150"/>	<input type="text" value="ish"/>
WorkingHours: <input type="text" value="6"/>	<input type="text" value="s"/>
shifts: <input type="text" value="day"/>	

vacancy:

Message ×

**Succuessfully appointed.**

Figure 6:test5

## 5.6 Test-6

Objective	terminating a staff of Part time staff
Action	Adding vacancy number of staff which is to be terminated
Expected Result	Dialog box of information message will appear
Actual Result	Dialog box was appeared of successfully terminated
Conclusion	Test was successfully tested

**For Part Time Staff**

Vacancy number: 1	Vacancy Number: 1
Designation: doctor	Staffname: sujan
Job type: part	Joining Date: 2075-05-12
WagesPerHour: 150	ish
WorkingHours: 6	
shifts: day	

clear

Back

Appoint

submit

Display

vacancy: 1

terminate

Figure 7:test6

## 6.0 Error

Errors are the mistakes or faults in the program that cause our programs and it is no doubt that the well versed and experienced programmers also makes mistakes. Programming error are also called as bugs to remove bugs the process is called debugging.

### 6.1 Syntax error

A syntax error in computer science is an error in the syntax of a coding or programming language, entered by programmer. It is detected by software while compiling the code.

```

    tfwagesPerHour.setText(null);
    tfshifts.setText(null);
}
JMenuItem miexit= new JMenuItem("Exit");

```

Missing of closing of brackets.

```

panel12.setLayout(null);

```

Missing of coma.

Figure 8:syntax error

## 6.2 Semantics error

This error occurs when the syntax of your code is correct but code usage is not correct. The most common semantic error is one in which the code uses a variable that is not initialized properly. Example: if you try to assign a float to an int variable, the compiler displays an error message.

```

public void submit(){
    string vc= 0,wh= 0,sy= 0;
    String designation = tfdesignation.getText();
}

public void submit(){
    int vc= 0,wh= 0,sy= 0;
    String designation = tfdesignation.getText();
}

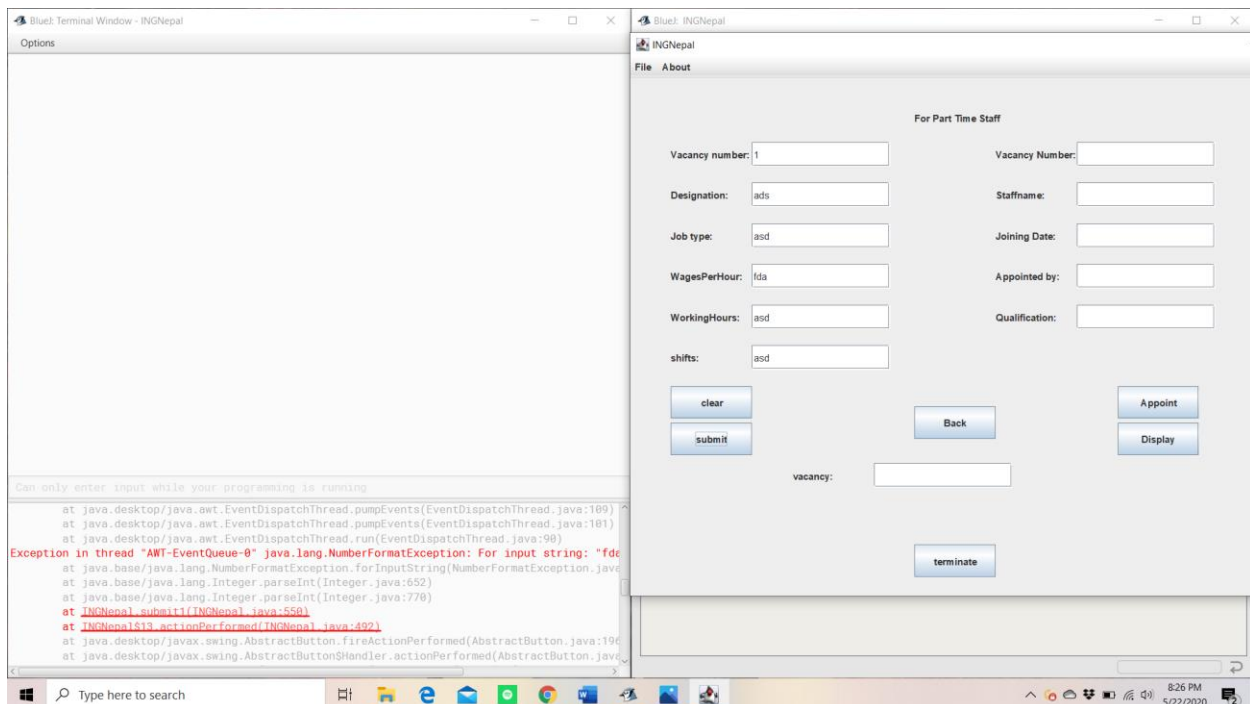
```

Figure 9:semantic error

The error occurs due to wrong data types written during compiling the program

## 6.3 Run time error

A run time error is a program error that occurs while the program is running. The term is often used in contrast to other types of program errors such as syntax errors and compile time errors. It occurs by not using try catch block.



```

public void submit1(){
    int vc=0,wh=0,wp=0;
    String designation = tfdesignation.getText();
    String shifts = tfshifts.getText();
    String vacancy= tfvacancy1.getText();
    String workingHour= tfworkingHour.getText();
    String wagesPerHour= tfwagesPerHour.getText();
    String jobType= tfjobType.getText();
    if(vacancy.trim().equals("")||designation.trim().equals("")||jobType.trim().equals("")||workingHour.trim()
        JOptionPane.showMessageDialog(Frame, "please fill all the fields", "error",0);
    }else{
        vc = Integer.parseInt(vacancy);
        wh = Integer.parseInt(workingHour);
        wp = Integer.parseInt(wagesPerHour);
    }
    for(StaffHire s: abc){

```

Figure 10:run time error

## 7.0 Appendix

```

import javax.swing.JFrame;

import java.awt.EventQueue;

import javax.swing.JPanel;

import javax.swing.JButton;

import javax.swing.JLabel;

```

```
import javax.swing.JTextField;
import javax.swing.JMenuBar;
import javax.swing.JMenu;
import javax.swing.JMenuItem;
import javax.swing.JOptionPane;
import java.awt.Event;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.util.ArrayList;
public class INGNepal{
    ArrayList<StaffHire> abc = new ArrayList<StaffHire>();
    private JFrame Frame;

    private JPanel panel1,panel2,panel3;
    private JButton btn1;
    private JButton btn2;
    private JButton btnclear;
    private JButton btnclear2;
    private JButton btnappoint;
    private JButton btnappoint2;

    private JButton btnsubmit;
    private JButton btnsubmit2;

    private JButton btndisplay;
    private JButton btndisplay2;
    private JButton btnterminate;
```



```
private JButton btnBack1,btnBack2;

private JTextField
tfvacancy1,tfdesignation,tfjobType,tfworkingHour,tfsalary,tfvacancy2,tfstaffName,tfjoiningDate,tfappoi
ntedBy,tfqualification,tfwagesPerHour,tfshifts,tfvacancy3;

private JLabel
lbl1,lbl2,lbvacancy1,lbdesignation,lbjobType,lbworkingHour,lbsalary,lbvacancy2,lbstaffName,lbjoiningDa
te,lbappointedBy,lbqualification,lbwagesPerHour,lbshifts,lbvacancy3;

public INGNepal(){
    initializeFrame();
    initializeMenu();
    initializeBody();
}

private void initializeFrame(){
    Frame = new JFrame("INGNepal");
    Frame.setBounds(0,0,900,700);
    Frame.setLayout(null);
    Frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    Frame.setResizable(false);
}

private void initializeMenu(){
    JMenuBar file = new JMenuBar();
    file.setBounds(0,0,1095,26);
    Frame.getContentPane().add(file);
    JMenu mfile = new JMenu("File");
    file.add(mfile);
    JMenuItem minew = new JMenuItem("New");
    mfile.add(minew);
```

```
mfile.addSeparator();

minew.addActionListener(new ActionListener(){

    public void actionPerformed(ActionEvent ae){

        tfvacancy1.setText(null);

        tfdesignation.setText(null);

        tfjobType.setText(null);

        tfworkingHour.setText(null);

        tfsalary.setText(null);

        tfvacancy2.setText(null);

        tfstaffName.setText(null);

        tfjoiningDate.setText(null);

        tfappointedBy.setText(null);

        tfqualification.setText(null);

        tfwagesPerHour.setText(null);

        tfshifts.setText(null);

    }

});

JMenuItem miexit= new JMenuItem("Exit");

mfile.add(miexit);

miexit.addActionListener(new ActionListener(){

    public void actionPerformed(ActionEvent ae) {

        System.exit(0);

    };

});

JMenu mabout= new JMenu("About");

file.add(mabout);
```

```
JMenuItem miapp = new JMenuItem("App");
mabout.add(miapp);
miapp.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae){
        JOptionPane.showMessageDialog(Frame, "App:version
2.0","App",JOptionPane.INFORMATION_MESSAGE);
    };
});

JMenuItem mideveloper = new JMenuItem("Developer");
mabout.add(mideveloper);
mabout.addSeparator();
mideveloper.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae){
        JOptionPane.showMessageDialog(Frame, "App is desingned by sujan
chaudhary","Developer",JOptionPane.INFORMATION_MESSAGE);
    };
});
}

private void initializeBody(){
    panel1=new JPanel();
    Frame.add(panel1);
    panel1.setSize(900,700);
    panel1.setLayout(null);

    btn1=new JButton("FullTimeStaffHire");
    btn1.setBounds(300,250,200,50);
```

```
panel1.add(btn1);
btn1.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae){
        panel1.setVisible(false);
        forFullTimeStaffHire();
    }
});

btn2=new JButton("PartTimeStaffHire");
btn2.setBounds(300,350,200,50);
panel1.add(btn2);
btn2.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae){
        panel1.setVisible(false);
        forPartTimeStaffHire();
    }
});
}
```

```
public void forFullTimeStaffHire(){
    panel2=new JPanel();
    Frame.add(panel2);
    panel2.setSize(900,700);
    panel2.setLayout(null);

    lbvacancy1 = new JLabel("Vacancy number:");
```

```
panel2.add(lbvacancy1);  
lbdesignation = new JLabel("Designation:");  
panel2.add(lbdesignation);  
lbjobType = new JLabel("Job type:");  
panel2.add(lbjobType);  
lbsalary = new JLabel("Salary:");  
panel2.add(lbsalary);  
lbworkingHour = new JLabel("WorkingHours:");  
panel2.add(lbworkingHour);  
lbvacancy2 = new JLabel("Vacancy Number:");  
panel2.add(lbvacancy2);  
lbstaffName = new JLabel("Staff Name");  
panel2.add(lbstaffName);  
lbjoiningDate = new JLabel("Joining Date:");  
panel2.add(lbjoiningDate);  
lbappointedBy = new JLabel("Appointed by:");  
panel2.add(lbappointedBy);  
lbqualification = new JLabel("Qualification:");  
panel2.add(lbqualification);  
lbl1 = new JLabel("For Full Time Staff");  
panel2.add(lbl1);  
  
tfvacancy1 = new JTextField();  
panel2.add(tfvacancy1);  
tfdesignation = new JTextField();  
panel2.add(tfdesignation);  
tfjobType= new JTextField();  
panel2.add(tfjobType);
```

```
tfsalary = new JTextField();  
panel2.add(tfsalary);  
tfworkingHour = new JTextField();  
panel2.add(tfworkingHour);  
tfvacancy2 = new JTextField();  
panel2.add(tfvacancy2);  
tfstaffName= new JTextField();  
panel2.add(tfstaffName);  
tfjoiningDate = new JTextField();  
panel2.add(tfjoiningDate);  
tfappointedBy = new JTextField();  
panel2.add(tfappointedBy);  
tfqualification = new JTextField();  
panel2.add(tfqualification);
```

```
lbl1.setBounds(350,50,150,50);  
lbvacancy1.setBounds(50,100,100,40);  
lbdesignation.setBounds(50,150,100,40);  
lbjobType.setBounds(50,200,100,40);  
lbsalary.setBounds(50,250,100,40);  
lbworkingHour.setBounds(50,300,100,40);  
lbvacancy2.setBounds(450,100,100,40);  
lbstaffName.setBounds(450,150,100,40);  
lbjoiningDate.setBounds(450,200,100,40);  
lbappointedBy.setBounds(450,250,100,40);  
lbqualification.setBounds(450,300,100,40);
```

```
tfvacancy1.setBounds(150,105,170,30);
tfdesignation.setBounds(150,155,170,30);
tfjobType.setBounds(150,205,170,30);
tfsalary.setBounds(150,255,170,30);
tfworkingHour.setBounds(150,305,170,30);
tfvacancy2.setBounds(550,105,170,30);
tfstaffName.setBounds(550,155,170,30);
tfjoiningDate.setBounds(550,205,170,30);
tfappointedBy.setBounds(550,255,170,30);
tfqualification.setBounds(550,305,170,30);

btnclear = new JButton("clear");
panel2.add(btnclear);
btnclear.setBounds(50,405,100,40);
btnclear.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae){
        btnclear.setText("cleared");
        tfvacancy1.setText(null);
        tfdesignation.setText(null);
        tfjobType.setText(null);
        tfworkingHour.setText(null);
        tfsalary.setText(null);
        tfvacancy2.setText(null);
        tfstaffName.setText(null);
        tfjoiningDate.setText(null);
        tfappointedBy.setText(null);
        tfqualification.setText(null);
    }
});
```

```
    };  
});  
  
btnsubmit = new JButton("submit");  
panel2.add(btnsubmit);  
btnsubmit.setBounds(50,450,100,40);  
btnsubmit.addActionListener(new ActionListener(){  
    public void actionPerformed(ActionEvent ae){  
        submit();  
    };  
});  
  
btnappoint = new JButton("Appoint");  
panel2.add(btnappoint);  
btnappoint.setBounds(600,405,100,40);  
btnappoint.addActionListener(new ActionListener(){  
    public void actionPerformed(ActionEvent ae){  
        appoint();  
    };  
});  
  
btnBack1=new JButton("Back");  
btnBack1.setBounds(350,430,100,40);  
panel2.add(btnBack1);  
btnBack1.addActionListener(new ActionListener(){  
    public void actionPerformed(ActionEvent ae){  
        panel1.setVisible(true);  
        panel2.setVisible(false);  
    }  
});
```



```
        };  
    });  
  
    btndisplay=new JButton("Display");  
    panel2.add(btndisplay);  
    btndisplay.setBounds(600,450,100,40);  
    btndisplay.addActionListener(new ActionListener(){  
        public void actionPerformed(ActionEvent ae){  
            display();  
        }  
    });  
}  
  
public void submit(){  
    int vc= 0,wh= 0,sy= 0;  
    String designation = tfdesignation.getText();  
    String jobType = tfjobType.getText();  
    String vacancyNumber= tfvacancy1.getText();  
    String workingHour= tfworkingHour.getText();  
    String salary= tfsalary.getText();  
  
    if(vacancyNumber.trim().equals("") || designation.trim().equals("") || jobType.trim().equals("") || working  
    Hour.trim().equals("") || salary.trim().equals("")){  
        JOptionPane.showMessageDialog(Frame, "please fill all the fields", "error",0);  
    }else{  
        try{  
            vc = Integer.parseInt(vacancyNumber);  
            wh = Integer.parseInt(workingHour);
```

```
        sy = Integer.parseInt(salary);
    }catch(NumberFormatException e){
        JOptionPane.showMessageDialog(Frame, "give correct input.", "error", 0);
        return;
    }
    for(StaffHire s: abc){
        if(s instanceof FullTimeStaffHire){
            FullTimeStaffHire ft= (FullTimeStaffHire) s;
            if(ft.getVacancyNumber()== vc){
                JOptionPane.showMessageDialog(Frame, "give same vacancy no.", "error", 0);
                return;
            }else{
                }
            }
        }
    }

    FullTimeStaffHire full=new FullTimeStaffHire(vc,sy,wh,designation,jobType);
    abc.add(full);
    JOptionPane.showMessageDialog(Frame, "Succuessfully added.");
}

}

public void appoint(){
    int vc=0;
    String staffName = tfstaffName.getText();
    String joiningDate = tfjoiningDate.getText();
```

```

String qualification = tfqualification.getText();

String appointedBy = tfappointedBy.getText();

String vacancyNumber = tfvacancy2.getText();

if(staffName.trim().equals("") || joiningDate.trim().equals("") || qualification.trim().equals("") || appointed
By.trim().equals("") || vacancyNumber.trim().equals("")){

    JOptionPane.showMessageDialog(Frame, "please fill all the fields", "error",0);

}else{

    try{

        vc = Integer.parseInt(vacancyNumber);

    }

    catch(NumberFormatException e){

        JOptionPane.showMessageDialog(Frame, "give correct input.", "error",0);

        return;

    }

    boolean found= false;

    for(int i = 0; i < abc.size(); ++i){

        StaffHire sh= abc.get(i);

        if(sh instanceof FullTimeStaffHire) {

            FullTimeStaffHire full = (FullTimeStaffHire) sh;

            if(full.getVacancyNumber()==vc){

                full.hireFullTimeStaff (staffName,joiningDate,qualification,appointedBy);

                JOptionPane.showMessageDialog(Frame, "Succuessfully appointed.");

                found= true;

                break;

            }

        }

    }

}

```

```
    }  
    if(!found){  
        JOptionPane.showMessageDialog(Frame, "vacancy number not found.", "error", 0);  
    }  
}  
}
```

```
public void display(){  
    for(StaffHire hire: abc){  
        if(hire instanceof FullTimeStaffHire){  
            FullTimeStaffHire full = (FullTimeStaffHire) hire;  
            full.display();  
        }  
    }  
}
```

```
public void forPartTimeStaffHire(){  
    panel3=new JPanel();  
    Frame.add(panel3);  
    panel3.setSize(900,700);  
    panel3.setLayout(null);  
  
    lbvacancy1 = new JLabel("Vacancy number:");  
    panel3.add(lbvacancy1);  
  
    lbdesignation = new JLabel("Designation:");  
    panel3.add(lbdesignation);
```

```
lbjobType = new JLabel("Job type:");  
panel3.add(lbjobType);
```

```
lbwagesPerHour = new JLabel("WagesPerHour:");  
panel3.add(lbwagesPerHour);
```

```
lbworkingHour = new JLabel("WorkingHours:");  
panel3.add(lbworkingHour);
```

```
lbshifts = new JLabel("shifts:");  
panel3.add(lbshifts);
```

```
lbvacancy2 = new JLabel("Vacancy Number:");  
panel3.add(lbvacancy2);
```

```
lbstaffName = new JLabel("Staffname:");  
panel3.add(lbstaffName);
```

```
lbjoiningDate = new JLabel("Joining Date:");  
panel3.add(lbjoiningDate);
```

```
lbappointedBy = new JLabel("Appointed by:");  
panel3.add(lbappointedBy);
```

```
lbqualification = new JLabel("Qualification:");  
panel3.add(lbqualification);
```

```
lbl2 = new JLabel("For Part Time Staff");  
panel3.add(lbl2);
```

```
tfvacancy1 = new JTextField();  
panel3.add(tfvacancy1);
```

```
tfdesignation = new JTextField();  
panel3.add(tfdesignation);
```

```
tfjobType= new JTextField();  
panel3.add(tfjobType);
```

```
tfwagesPerHour= new JTextField();  
panel3.add(tfwagesPerHour);
```

```
tfworkingHour = new JTextField();  
panel3.add(tfworkingHour);
```

```
tfshifts = new JTextField();  
panel3.add(tfshifts);
```

```
tfvacancy2 = new JTextField();  
panel3.add(tfvacancy2);
```

```
tfstaffName= new JTextField();  
panel3.add(tfstaffName);
```

```
tfjoiningDate = new JTextField();
panel3.add(tfjoiningDate);

tfappointedBy = new JTextField();
panel3.add(tfappointedBy);

tfqualification = new JTextField();
panel3.add(tfqualification);

lbvacancy3= new JLabel("vacancy:");
panel3.add(lbvacancy3);
lbvacancy3.setBounds(200,500,170,30);
tfvacancy3 = new JTextField();
panel3.add(tfvacancy3);
tfvacancy3.setBounds(300,500,170,30);
// bounding for labels
lbl2.setBounds(350,50,150,50);
lbvacancy1.setBounds(50,100,100,40);
lbdesignation.setBounds(50,150,100,40);
lbjobType.setBounds(50,200,100,40);
lbwagesPerHour.setBounds(50,250,100,40);
lbworkingHour.setBounds(50,300,100,40);
lbshifts.setBounds(50,350,100,40);
lbvacancy2.setBounds(450,100,100,40);
lbstaffName.setBounds(450,150,100,40);
lbjoiningDate.setBounds(450,200,100,40);
lbappointedBy.setBounds(450,250,100,40);
```

```
lbqualification.setBounds(450,300,100,40);
//bounding for text field
tfvacancy1.setBounds(150,105,170,30);
tfdesignation.setBounds(150,155,170,30);
tfjobType.setBounds(150,205,170,30);
tfwagesPerHour.setBounds(150,305,170,30);
tfworkingHour.setBounds(150,255,170,30);
tfshifts.setBounds(150,355,170,30);
tfvacancy2.setBounds(550,105,170,30);
tfstaffName.setBounds(550,155,170,30);
tfjoiningDate.setBounds(550,205,170,30);
tfappointedBy.setBounds(550,255,170,30);
tfqualification.setBounds(550,305,170,30);

btnclear2 = new JButton("clear");
panel3.add(btnclear2);
btnclear2.setBounds(50,405,100,40);
btnclear2.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae){
        clear1();
    }
});

public void clear1(){
    btnclear.setText("cleared");
    tfvacancy1.setText(null);
    tfdesignation.setText(null);
    tfshifts.setText(null);
    tfwagesPerHour.setText(null);
}
```



```
        tfworkingHour.setText(null);
        tfjobType.setText(null);
        tfvacancy2.setText(null);
        tfjoiningDate.setText(null);
        tfappointedBy.setText(null);
        tfstaffName.setText(null);
        tfqualification.setText(null);

    }
    });

    btnsubmit2 = new JButton("submit");
    panel3.add(btnsubmit2);
    btnsubmit2.setBounds(50,450,100,40);
    btnsubmit2.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent ae){
            submit1();
        }
    });

    btnBack2=new JButton("Back");
    btnBack2.setBounds(350,430,100,40);
    panel3.add(btnBack2);
    btnBack2.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent ae){
            panel1.setVisible(true);
            panel3.setVisible(false);

        }
    });
```

```
});

btnappoint2 = new JButton("Appoint");
panel3.add(btnappoint2);
btnappoint2.setBounds(600,405,100,40);
btnappoint2.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae){
        appoint1();
    }
});

btnterminate= new JButton("terminate");
panel3.add(btnterminate);
btnterminate.setBounds(350,600,100,40);
btnterminate.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae){
        terminateStaff();
    }
});

btndisplay2=new JButton("Display");
panel3.add(btndisplay2);
btndisplay2.setBounds(600,450,100,40);
btndisplay2.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae){
        display1();
    }
});
```

```
    });

}

public void submit1(){
    int vc=0,wh=0,wp=0;

    String designation = tfdesignation.getText();
    String shifts = tfshifts.getText();
    String vacancy= tfvacancy1.getText();
    String workingHour= tfworkingHour.getText();
    String wagesPerHour= tfwagesPerHour.getText();
    String jobType= tfjobType.getText();

    if(vacancy.trim().equals("") || designation.trim().equals("") || jobType.trim().equals("") || workingHour.trim().equals("") || wagesPerHour.trim().equals("") || shifts.trim().equals("")){
        JOptionPane.showMessageDialog(Frame, "please fill all the fields", "error",0);
    }else{
        try{
            vc = Integer.parseInt(vacancy);
            wh = Integer.parseInt(workingHour);
            wp = Integer.parseInt(wagesPerHour);

        }
        catch(NumberFormatException e){
            JOptionPane.showMessageDialog(Frame, "give correct input.", "error",0);
            return;
        }
        for(StaffHire s: abc){
```

```

        if(s instanceof PartTimeStaffHire){
            PartTimeStaffHire pt= (PartTimeStaffHire) s;
            if(pt.getVacancyNumber()== vc){
                JOptionPane.showMessageDialog(Frame, "give same vacancy no. ","error",0);
                return;
            }else{
            }
        }
    }

    PartTimeStaffHire part= new PartTimeStaffHire(vc,designation,shifts,wp,wh,jobType);
    abc.add(part);
    JOptionPane.showMessageDialog(Frame, "Succuessfully added.");
}

}

public void appoint1(){
    int vc=0;
    String staffName = tfstaffName.getText();
    String joiningDate = tfjoiningDate.getText();
    String qualification = tfqualification.getText();
    String appointedBy = tfappointedBy.getText();
    String vacancyNumber = tfvacancy2.getText();

    if(staffName.trim().equals("") || joiningDate.trim().equals("") || qualification.trim().equals("") || appointed
    By.trim().equals("") || vacancyNumber.trim().equals("")){
        JOptionPane.showMessageDialog(Frame, "please fill all the fileds","error",0);
    }else{
        try{

```

```
        vc = Integer.parseInt(vacancyNumber);

    }
    catch(NumberFormatException e){
        JOptionPane.showMessageDialog(Frame, "give correct input.");
        return;
    }
    boolean found= true;
    for(int i = 0; i < abc.size(); i++){
        StaffHire sh= abc.get(i);
        if(sh instanceof PartTimeStaffHire) {
            PartTimeStaffHire part = (PartTimeStaffHire) sh;
            if(part.getVacancyNumber()==vc){
                part.hirePartTimeStaff (staffName,joiningDate,qualification,appointedBy);
                JOptionPane.showMessageDialog(Frame, "Succuessfully appointed.");
                found= true;
                break;
            }
        }
    }
    if(!found){
        JOptionPane.showMessageDialog(Frame, "vacancy number not found.", "error", 0);
    }
}
}
```

```
public void terminateStaff(){
    int vc=0;
    String vacancyNumber= tfvacancy3.getText();
    if(vacancyNumber.trim().equals("")){
        JOptionPane.showMessageDialog(Frame,"please fill all the field", "error",0);
    }else{
        try{
            vc = Integer.parseInt(vacancyNumber);
        }catch(NumberFormatException e){
            JOptionPane.showMessageDialog(Frame,"please give data in correct format", "error",0);
            return;
        }
        boolean found= false;
        for(StaffHire sh: abc){
            if(sh instanceof PartTimeStaffHire){
                PartTimeStaffHire pt= (PartTimeStaffHire)sh;
                if(sh.getVacancyNumber()==vc){
                    pt.terminateStaff();
                    abc.remove(pt);
                    JOptionPane.showMessageDialog(Frame,"successfully terminated.");
                    found = true;
                    return;
                }
            }
        }
    }
}
```

```

        if(!found){
            JOptionPane.showMessageDialog(Frame,"vacancy number not found", "error",0);
        }
    }
}

```

```

public void display1(){
    for(StaffHire hire: abc){
        if(hire instanceof PartTimeStaffHire){
            PartTimeStaffHire part = (PartTimeStaffHire) hire;
            part.display();
        }
    }
}

```

```

public static void main(String []args){
    new INGNepal().Frame.setVisible(true);
}
}

```

## 8.0 Conclusion

In this coursework many error arises during doing this coursework due to this lockdown period as you also know that our college was shut down due to the shutdown many problems comes because we cannot directly interact with teachers but we were thank full that our college manage to give online classes this was new thing to us in the first class the lecture was taken by unknown teacher which hard to understand but after 2 or 3 classes I get used to it which allows me to understand the lectures. On this online classes there is one advantages that we can also watch it later if we did not understand during class. Coursework is to create the GUI to hire the staff to make this GUI I had used null layout which was easier according to me so I used it and the main problem

was exception handling in this course. By watching a video of coursework discussion which allow me to solve the problem which was while coding exception handling was much more difficult before watching of discussion video. That video helps me a lot during coding. I have used null layout to make GUI nice and well managed because using set bounds, we can put textfields, label as our desire. Using java swing can be quite a daunting experience which requires understanding of all libraries of java swing. All the exception was handled by using try and catch. There exception of number format exception which was handled by using try catch.

## 9.0 References

### Bibliography

(2018). Retrieved from javatpoint: <https://www.javatpoint.com/abstract-class-in-java>

*beginnersbook*. (n.d.). Retrieved from <https://beginnersbook.com/2013/04/java-exception-handling/>

*javatpoint*. (n.d.). Retrieved from <https://www.javatpoint.com/abstract-class-in-java>

*javatpoint*. (n.d.). Retrieved from <https://www.javatpoint.com/abstract-class-in-java>

*techterm*. (n.d.). Retrieved from <https://techterms.com/definition/java>

*tutorialpoint*. (n.d.).