



**Islington college**  
(इस्लिङ्टन कलेज)

**CS4001NI – PROGRAMMING**

**50% Individual Coursework**

**2019-20 Autumn**

**Student Name: Sunab Shrestha**

**London Met ID: 19031737**

**College ID: NP01CP4A190408**

**Assignment Due Date: 2020**

**Assignment Submission Date: , 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 mark of zero will be awarded.*

## Table of Contents

Introduction .....	4
Class Diagram .....	5
INGNepal: .....	5
PartTimeStaffHireGUI .....	6
FullTimeStaffHireGUI .....	7
Relation .....	8
Pseudocode.....	9
INGNepal .....	9
PartTimeStaffHireGUI .....	13
FullTimeStaffHireGUI .....	29
Method Descriptions .....	45
PartTimeStaffHireGUI and FullTimeStaffHireGUI: .....	45
Testing.....	47
Test 1: Output .....	47
Test 2: .....	48
Test 3: .....	57
Error Detection and Correction .....	59
Syntax Error: .....	59
Semantic Error: .....	60
Logical Error:.....	61
Conclusion .....	62
References .....	62
Appendix:.....	63
INGNepal .....	63
FullTimeStaffHire GUI .....	66
PartTimeStaffHireGUI .....	78

Figure 1: Class Diagram Of INGNepal .....	5
Figure 2: Class Diagram of PartTimeStaffHireGUI.....	6
Figure 3: Class Diagram of FullTimeStaffHireGUI.....	7
Figure 4: Relation of Class Diagrams .....	8
Figure 5: Showing the Directory.....	47
Figure 6: Output of command Prompt .....	47
Figure 7: Adding Information .....	48
Figure 8: Displaying Information.....	49
Figure 9: Output After Filling Up info .....	49
Figure 10: Adding Information In PartTimeStaff.....	50
Figure 12: Output after adding Information.....	51
Figure 11:Dispaly output.....	51
Figure 13: Adding Info in Full Time Staff .....	52
Figure 14: Dispaly Info of Full Time Staff Hire .....	53
Figure 15: Appointing PartTimeStaff.....	54
Figure 16: Display Output .....	55
Figure 17: Termanating Part Time Staff.....	56
Figure 18: Display Output .....	56
Figure 19: Showing error while putting string in vacancy number.....	57
Figure 20: Output of result.....	58
Figure 21: Syntax Error.....	59
Figure 22: Correction Of Syntax Error.....	59
Figure 23: Sementic Error.....	60
Figure 24: Correction Of Sementic Error.....	60
Figure 25: Logical Error .....	61
Figure 26: Correction Of Logical Error .....	61
Figure 1: Class Diagram Of StaffHire .....	Error! Bookmark not defined.
Figure 2: Class Diagram of PartTimeStaffHire.....	Error! Bookmark not defined.
Figure 3: Class Diagarm of FullTlmeStaffHire .....	Error! Bookmark not defined.
Figure 4: Combined Class Diagram .....	Error! Bookmark not defined.

## Introduction

This is our second 2nd semester programming coursework. We were told to do coding in this coursework and make GUI with the help of the one we need. Our main GUI in this work was INGNepal and PartTimeStaffHireGUI and FullTimeStaffHireGUI were other GUIs. This work has been given to us so that we can make GUI and take user input and do the specific function by calling the Parttimetaffhire, FulltimestaffHire and Staffhire functions. This was done with the aid of BlueJ which is a software. We've got coding and documentation part in that. My project involves six classes in which we've made Interfaces for my three classes and three others are classes of perpetual coursework. My coursework where INGNepal for ParttimeStaffHireGUI and FulltimePartTimeStaffGUI is the main class.

A class diagram is an illustration of the relationships and source code dependencies among classes in the Unified Modeling Language (UML). In this context, a class defines the methods and variables in an object, which is a specific entity in a program or the unit of code representing that entity. Class diagrams are useful in all forms of object-oriented programming (OOP). The concept is several years old but has been refined as OOP modeling paradigms have evolved. (TechTarget, 2019)

**INGNepal:**



*Figure 1: Class Diagram Of INGNepal*



Figure 2: Class Diagram of PartTimeStaffHireGUI



Figure 3: Class Diagram of FullTimeStaffHireGUI

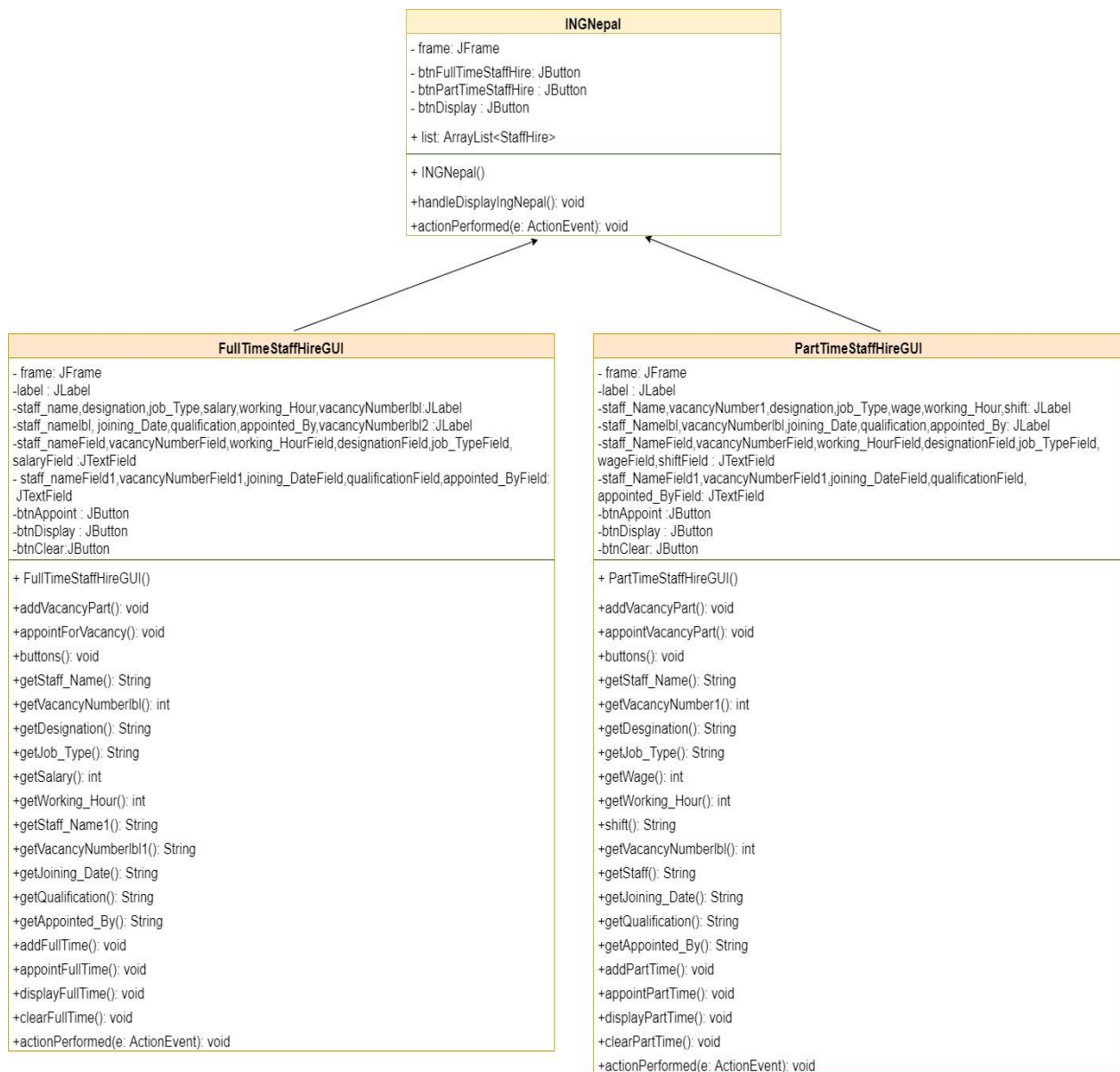


Figure 4: Relation of Class Diagrams



**Pseudocode**

Pseudocode is a detailed yet readable description of what a computer program or algorithm must do, expressed in a formally-styled natural language rather than in a programming language. Pseudocode is sometimes used as a detailed step in the process of developing a program. It allows designers or lead programmers to express the design in great detail and provides programmers a detailed template for the next step of writing code in a specific programming language. It is possible to write programs that will convert a given pseudocode language into a given programming language. (TechTarget, 2019)

**INGNepal**

**DEFINE** INGNepal

**DECLARE** instance variables

**INITIALIZE** frame as JFrame

**INITIALIZE** btnPartTimeStaffHire as

JButton **INITIALIZE** btnFullTimeStaffHire

as JButton **INITILIZE** btnDisplay as

JButton

**INITIALIZE** ArrayList<StaffHire > list as ArrayList<StaffHire>

**DEFINE** constructor INGNepal

**DO**

**CREATE** JFrame object as frame

**CALL** method diffPart()

**SET** frame default close operation

**SET** frame

size **SET**

frame layout

**SET** frame

visible

**DEFINE** method diffPart()

**DO**

**CREATE** JButton object as btnPartTimeStaffHire

**SET** bounds for btnPartTimeStaffHire

**ADD** ActionListener in btnPartTimeStaffHire

**ADD** btnFullTimeStaffHire in frame

**CREATE** JButton object as btnFullTimeStaffHire

**SET** bounds for btnFullTimeStaffHire

**ADD** ActionListener in btnFullTimeStaffHire

**ADD** btnFullTimeStaffHire in frame

**CREATE** JButton object as btnDisplay

**SET** bounds for btnDisplay

**ADD** ActionListener in btnDisplay

**ADD** btnDisplay in frame

**END DO**

**DEFINE** main method

**DO**

**CALL** INGNEpal();

**END DO**

**DEFINE** method displayINGNEpal

```
FOR StaffHire obj: list  
DO
```

```
    IF obj instanceof PartTimeStaffHire
```

```
DO
```

```
    Print "Part Time Information"
```

```
    ((PartTimeStaffHire)obj).displ
```

```
    ay()
```

```
END DO
```

```
    IF obj instanceof FullTimeStaffHire
```

```
DO
```

```
    Print "_____ " Print "Full Time  
    Information"
```

```
    ((FullTimeStaffHire)obj).display()
```

```
END DO
```

```
    END DO
```

```
DEFINE abstract method actionPerformed(ActionEvent e)
```

```
    IF btnFullTimeStaffHire
```

```
DO
```

```
    CALL class PartTimeStaffHireGUI
```

```
END DO
```

```
    ELSE IF btnPartTimeStaffHire
```

```
DO
```

```
    CALL class FullTimeStaffHireGUI
```

**END DO**

**ELSE IF** btnDisplay

**DO**

**CALL** method displayINGNepal

**END DO**

---

### **PartTimeStaffHireGUI**

**DEFINE** PartTimeStaffHireGUI implements ActionListener

**DECLARE** instance variables

**INITIALIZE** frame as

JFrame **INITIALIZE** label

as JLabel

**INITIALIZE**

name,vacancyNumber1,designation,job\_Type,wage, Working\_Hour,Shift

as JLabel

**INITIALIZE**

nameField,vacancyNumberField,wageField,shiftField,

working\_HourField,designationField,job\_TypeField as JTextField

**INITIALIZE**

nameLbl,vacancyNumberLbl,joining\_Date,qualification, Appointed\_By as

**INITIALIZE** nameField1,vacancyNumberField1, qualificationField,  
joining\_DateField, appointed\_ByField as JTextField

**INITIALIZE** btnAppoint as JButton

**INITIALIZE** btnDisplay as JButton

**INITIALIZE** btnClear as JButton

**INITIALIZE** btnTerminate as

JButton **INITIALIZE** btnPartTime

as JButton **INITIALIZE** btnAdd as

JButton

**DEFINE** main method

**DO**

**CALL** constructor PartTimeStaffHireGUI();

**END DO**

**DEFINE** constructor PartTimeStaffHiringGUI()

**DO**

**CREATE** JFrame object as

frame **CREATE** JLabel object

as label **SET** bounds for label

**ADD** label to frame

**CALL** method addVacancyPart

**CALL** method

```
appointVacancyPart    CALL  
method buttons
```

```
SET size for frame
```

```
SET layout for
```

```
frame SET visible
```

```
for frame
```

```
END DO
```

```
DEFINE method addVacancyPart
```

```
DO
```

```
CREATE JLabel object as name
```

```
SET bounds for name
```

```
ADD name to frame
```

```
CREATE JTextField object as nameField
```

```
SET bounds for nameField
```

```
ADD nameField to frame
```

```
CREATE JLabel object as vacancyNumber1
```

```
SET bounds for vacancyNumber1
```

```
ADD vacancuNumber1 to frame
```

```
CREATE JTextField object as vacancyNumberField
```

**SET** bounds for vacancy numberField

**ADD** vacancyNumberField to frame

**CREATE** JLabel object as designation

**SET** bounds for designation

**ADD** designation to frame

**CREATE** JTextField object as designationField

**SET** bounds for designationField

**ADD** designationField to frame

**CREATE** JLabel object as job\_Type

**SET** bounds for job\_Type

**ADD** job\_Type to frame

**CREATE** JTextField object as job\_TypeField

**SET** bounds for job\_TypeField

**ADD** job\_TypeField to frame

**CREATE** JLabel object as wage

**SET** bounds for wage

**ADD** wage to frame



**CREATE** JTextField object as wageField

**SET** bounds for wageField

**ADD** wageField to frame

**CREATE** JLabel object as working\_Hour

**SET** bounds for working\_Hour

**ADD** working\_Hour to frame

**CREATE** JTextField object as working\_HourField

**SET** bounds for working\_HourField

**ADD** working\_HourField to frame

**CREATE** JLabel object as shift

**SET** bounds for shift

**ADD** shiftto frame

**CREATE** JTextField object as shiftField

**SET** bounds for shiftField

**ADD** shiftField to frame

**CREATE** JLabel object as line

**SET** bounds for line

**ADD** line to frame

**END DO**

**DEFINE** method  
appointVacancyPart

**DO**

**CREATE** JLabel object as nameLbl

**SET** bounds for nameLbl

**ADD** nameLbl to frame

**CREATE** JTextField object as nameField1

**SET** bounds for nameField1

**ADD** nameField1 to frame

**CREATE** JLabel object as vacancyNumberLbl

**SET** bounds for vacancyNumberLbl

**ADD** vacancyNumberLbl to frame

**CREATE** JTextField object as vacancyNumberField1

**SET** bounds for vacancyNumberField1

**ADD** vacancyNumberField1 to frame

**CREATE** JLabel object as joining\_Date

**SET** bounds for joining\_Date

**ADD** joining\_Date to frame

**CREATE** JTextField object as joining\_DateField

**SET** bounds for joining\_DateField

**ADD** joining\_DateField to frame

**CREATE** JLabel object as qualification

**SET** bounds for qualification

**ADD** qualification to frame

**CREATE** JTextField object as qualificationField

**SET** bounds for qualificationField

**ADD** qualificationField to frame

**CREATE** JLabel object as appointed\_By

**SET** bounds for appointed\_By

**ADD** appointed\_By to frame

**CREATE** JTextField object as appointed\_ByField

**SET** bounds for appointed\_ByField

**ADD** appointed\_ByField to frame

**END DO**

**DEFINE** method buttons

**DO**

**CREATE** JButton object as btnPartTime

**SET** bounds for btnPartTime

**ADD** ActionListener in btnPartTime

**ADD** btnPartTime in frame

**CREATE** JButton object as btnAppoint

**SET** bounds for btnAppoint

**ADD** ActionListener in btnAppoint

**ADD** btnAppoint in frame

**CREATE** JButton object as btnClear

**SET** bounds for btnClear

**ADD** ActionListener in btnClear

**ADD** btnClear in frame

**CREATE** JButton object as btnDisplay

**SET** bounds for btnDisplay

**ADD** ActionListener in btnDisplay

**ADD** btnDisplay in frame

**CREATE** JButton object as btnTerminate

**SET** bounds for btnTerminate

**ADD** ActionListener in btnTerminate

**ADD** btnTerminate in frame

**END DO**

**DEFINE** method getName() as return type String

**DO**

**RETURN** text extracted from "nameField"

**END DO**

**DEFINE** method getVacanxyNumber1() as return type int

**DECLARE** instance variable

**INITIALIZE** vacancyNumberInt

    = 0 **TRY**

        vacancyNumberInt = convert text obtained from vacancyNumberField to  
integer

**CATCH** NumberFormatException e

**DISPLAY** dialogbox "Please give a valid interger in Vacancy Number".

**RETURN** vacancyNumberInt

**DEFINE** method getDesignation() as return type String

**DO**

**RETURN** text extracted from "dFielesignationd"

**END DO**

**DEFINE** method getJob\_Type() as return type String

**DO**

**RETURN** text extracted from "job\_TypeField"

**END DO**

**DEFINE** method getWage() as return type

int **DECLARE** instance variable

**INITIALIZE** wageInt= 0

wageInt= convert text obtained from wageField to integer

**CATCH** NumberFormatException e

**DISPLAY** dialogbox "Please give a valid interger in Wage".

**RETURN** wageInt

**DEFINE** method getWorking\_Hour() as return type int

**DECLARE** instance variable

**INITIALIZE** workingHourInt= 0

**TRY**

workingHourInt= convert text obtained from workingHourField to integer

**CATCH** NumberFormatException e

**DISPLAY** dialogbox "Please give a valid interger in Working Hour"

**RETURN** workingHourInt

**DEFINE** method getShift() as return type String

**DO**

**RETURN** text extracted from "shiftField"

**END DO**

**DEFINE** method getStaff\_Name1() as return type String

**DO**

**RETURN** text extracted from "nameField1"

**END DO**

**DEFINE** method getVacancyNumberlbl1() as return type int

**DECLARE** instance variable

**INITIALIZE** vacancyNumberlblInt

= 0 **TRY**

vacancyNumberlblInt = convert text obtained from vacancyNumberField1 to  
integer

**CATCH** NumberFormatException e

**DISPLAY** dialogbox "Please give a valid interger in Vacancy Number".

**RETURN** vacancyNumberlblInt

**DEFINE** method getJoining\_Date() as return type String

**DO**

**RETURN** text extracted from "joining\_DateField"

**END DO**

**DEFINE** method getQualification() as return type String

**DO**

**RETURN** text extracted from "qualaificationField"

**END DO**

**DEFINE** method getAppointed\_By() as return type String

**DO**

**RETURN** text extracted from "appointed\_ByField"

**END DO**

**DEFINE** method addPartTime

**INITIALIZE** vno as int and assign getter method getVacancyNumber1

**INITIALIZE** wage as int and assign getter method getWage

**INITIALIZE** working\_Hour as int and assign getter method  
getWorking\_Hour

**INITIALIZE** job\_Type as String and assign getter method getJob\_Type

**INITIALIZE** designation as String and assign getter method  
getDesignation

**INITIALIZE** shift as String and assign getter method getShift

**INITIALIZE** VNum1 as Boolean and set it to false

**IF** vno & wage & working\_Hour is greater than 0 & jobt\_type &  
designation & shift is not equal to empty String

**FOR** StaffHire duplicate: INGNepal.list

**DO**

**IF** list's getVacancyNumber consists same value  
as vno

**SET** vNum1 as true

**DISPLAY** dialogbox "Fill the field which are required "

**END DO**

**IF** vNum1 is false

**ADD** in INGNepal.list(PartTimeStaffHire(pass  
instance argument designation , job\_Type, vno,  
salary, working\_Hour, shift) constructor as object)

**DISPLAY** dialogbox "Thank you for your information "



**DEFINE** method appointPartTime

**DO**

**DECLARE** instance variables

**INITIALIZE** vno as int and assign it to getter  
method getVacancyNumberField1

**INITIALIZE** name as String and assign it to getter method getName

**INITIALIZE** joining\_Date as String and assign it to getter  
method getJoining\_Date

**INITIALIZE** qualification as String and assign it to getter  
method getQualification

**DISPLAY** dialogbox "invalid vacancy"

**END DO**

**DEFINE** method ClearParTTime

**SET** nameField to text

"null" **SET** nameField1 to  
text "null"

**SET** vacancyNumberField to text

"null" **SET** vacancyNumberField1 to  
text "null" **SET** designationField to  
text "null"

**SET** job\_TypeField to text "null"

**SET** wageField to text "null"

**SET** working\_HourField to text "null"

**SET** shiftField to text "null"

```
        SET joining_DateField to text
        "null" SET qualificationField to
        text "null" SET
        appointed_ByField to text "null"

    END DO

    DEFINE method displayPartTime

        IF list of class INGNepal is empty

    DO

        DISPLAY dialogbox "Fill the field which are required "

    END DO
    ELSE

        DO

            FOR StaffHire      obj: list

    DO

        ((PartTimeStaffHire)obj).display()

    END DO

    END DO

    DEFINE          method
    terminatePartTime

        DECLARE instance variables

            INITIALIZE vNum as Boolean and set it to false

            INITIALIZE vacancy as int and assign it to getter
            method getVacancyNumber1
```

**IF** vacancy is greater than 0

**FOR** StaffHire obj: INGNepal.list

**DO**

**IF** list's getVacancyNumber consists same value as  
of vacancy

**SET** vNum= true

**IF** obj instanceof PartTimeStaffHire

**CREATE** PartTimeStaffHire  
object casting of  
PartTimeStaffHire as p

**IF** PartTimeStaffHire's  
getTerminated is false

**CALL** method terminateStaff

**DISPLAY** dialogbox "Staff has  
been terminated"

**ELSE**

**DISPLAY** dialogbox "Enter a valid

vacancy number.

Staff is already

Terminated".

**ELSE**

**DISPLAY** dialogbox "Invalid  
vacancy number. Not valid for  
termination"

**END DO**

**IF** vNum is false

**DISPLAY** dialogbox "Staff is not hired. Enter  
another vacancy number."

**END DO**

**DEFINE** main  
method

**Call** PartTimeStaffHireGUI()

**DEFINE** abstract method actionPerformed(ActionEvent e)

**IF** btnPartTime

**DO**

**CALL** method addPartTime

**END DO**

**ELSE IF** btnappoint

**DO**

**CALL** method appointPartTime

**END DO**

**ELSE IF** btnDisplay

**DO**

**CALL** method displayPartTime

**END DO**

**ELSE IF** btnClear

**DO**

**CALL** method clearPartTime

**END DO**

**ELSE IF** btnTerminate

**DO**

**CALL** method terminatePartTime

**END DO**

---

### **FullTimeStaffHireGUI**

**DEFINE** FullTimeStaffHireGUI

**DECLARE** instance variables

**INITIALIZE** frame as

JFrame **INITIALIZE** label

as JLabel

**INITIALIZE**

staff\_name,vacancyNumberlbl,designation,job\_Type,salary, working\_Hour as

JLabel

**INITIALIZE**

staff\_nameField,vacancyNumberField,salaryField,

working\_HourField,designationField,job\_TypeField as JTextField

**INITIALIZE**

staff\_nameLbl,vacancyNumberLbl2,joining\_Date,qualification, Appointed\_By as  
JLabel

**INITIALIZE**

staff\_nameField1,vacancyNumberField1,  
qualificationField, joining\_DateField, appointed\_ByField as JTextField

**INITIALIZE** btnAppoint as

JButton **INITIALIZE** btnDisplay

as JButton **INITIALIZE** btnClear

as JButton **INITIALIZE**

btnFullTime as JButton

**DEFINE** main method

**END DO**

**DEFINE** constructor FullTimeStaffHireGUI()

**DO**

**CREATE** JFrame object as frame

**CREATE** JLabel object as label

**SET** bounds for label

**ADD** label to frame

**CALL** method addVacancyPart

**CALL** method appointVacancyPart

**CALL** method buttons

**SET** size for frame

**SET** layout for

frame **SET** visible

for frame

**END DO**

**DEFINE** method addVacancyPart

**DO**

**CREATE** JLabel object as staff\_name

**SET** bounds for staff\_name

**ADD** staff\_name to frame

**CREATE** JTextField object as staff\_nameField

**SET** bounds for staff\_nameField

**ADD** staff\_nameField to frame

**CREATE** JLabel object as vacancyNumberlbl

**SET** bounds for vacancyNumberlbl

**ADD** vacancuNumberlbl to frame

**CREATE** JTextField object as vacancyNumberField

**SET** bounds for vacancyNumberField

**ADD** vacancyNumberField to frame

**CREATE** JLabel object as designation

**SET** bounds for designation

**ADD** designation to frame

**CREATE** JTextField object as designationField

**SET** bounds for designationField

**ADD** designationField to frame

**CREATE** JLabel object as job\_Type

**SET** bounds for job\_Type

**ADD** job\_Type to frame

**CREATE** JTextField object as job\_TypeField

**SET** bounds for job\_TypeField

**ADD** job\_TypeField to frame

**CREATE** JLabel object as salary

**SET** bounds for salary

**ADD** salary to frame

**CREATE** JTextField object as salaryField

**SET** bounds for salaryField



**ADD** salaryField to frame

**CREATE** JLabel object as working\_Hour

**SET** bounds for working\_Hour

**ADD** working\_Hour to frame

**CREATE** JTextField object as working\_HourField

**SET** bounds for working\_HourField

**ADD** working\_HourField to frame

**END DO**

**DEFINE** method appointVacancyPart

**DO**

**CREATE** JLabel object as staff\_nameLbl

**SET** bounds for staff\_nameLbl

**ADD** staff\_nameLbl to frame

**CREATE** JTextField object as staff\_nameField1

**SET** bounds for staff\_nameField1

**ADD** staff\_nameField1 to frame

**CREATE** JLabel object as vacancyNumberLbl2

**SET** bounds for vacancyNumberLbl2

**ADD** vacancyNumberLbl2 to frame

**CREATE** JTextField object as vacancyNumberField1

**SET** bounds for vacancyNumberField1

**ADD** vacancyNumberField1 to frame

**CREATE** JLabel object as joining\_Date

**SET** bounds for joining\_Date

**ADD** joining\_Date to frame

**CREATE** JTextField object as joining\_DateField

**SET** bounds for joining\_DateField

**ADD** joining\_DateField to frame

**CREATE** JLabel object as qualification

**SET** bounds for qualification

**ADD** qualification to frame

**CREATE** JTextField object as qualificationField

**SET** bounds for qualificationField

**ADD** qualificationField to frame

**CREATE** JLabel object as appointed\_By

**SET** bounds for appointed\_By

**ADD** appointed\_By to frame

**CREATE** JTextField object as appointed\_ByField

**SET** bounds for appointed\_ByField

**ADD** appointed\_ByField to frame

**END DO**

**DEFINE** method buttons

**DO**

**CREATE** JButton object as btnFullTime

**SET** bounds for btnFullTime

**ADD** ActionListener in btnFullTime

**ADD** btnFullTime in frame

**CREATE** JButton object as btnAppoint

**SET** bounds for btnAppoint

**ADD** ActionListener in btnAppoint

**ADD** btnAppoint in frame

**CREATE** JButton object as btnClear

**SET** bounds for btnClear

**ADD** ActionListener in btnClear

**ADD** btnClear in frame

**CREATE** JButton object as btnDisplay

**SET** bounds for btnDisplay

**ADD** ActionListener in btnDisplay

**ADD** btnDisplay in frame

```
DEFINE method getstaff_name() as return type String
DO

    RETURN text extracted from "staff_nameField"

END DO
```

```
DEFINE method getVacancyNumberInt() as return type int
```

```
    DECLARE instance variable
```

```
    INITIALIZE vacancyNumberInt
```

```
    = 0 TRY
```

```
        vacancyNumberInt = convert text obtained from vacancyNumberField to
integer
```

```
    CATCH NumberFormatException e
```

```
        DISPLAY dialogbox "Please give a valid interger in Vacancy Number".
```

```
    RETURN vacancyNumberInt
```

```
DEFINE method getDesignation() as return type String
```

```
DO
```

```
    RETURN text extracted from "designationField"
```

```
END DO
```

```
DEFINE method getJob_Type() as return type String
```

```
DO
```

```
    RETURN text extracted from "job_TypeField"
```

**END DO**

**DEFINE** method getSalary() as return type

int **DECLARE** instance variable

**INITIALIZE** salaryInt= 0

**TRY**

wageInt= convert text obtained from salaryField to integer

**CATCH** NumberFormatException e

**DISPLAY** dialogbox "Please give a valid interger in salary".

**RETURN** salaryInt

**DEFINE** method getWorking\_Hour() as return type int

**DECLARE** instance

variable **INITIALIZE**

workingHourInt= 0 **TRY**

workingHourInt= convert text obtained from workingHourField to integer

**CATCH** NumberFormatException e

**DISPLAY** dialogbox "Please give a valid interger in Working Hour"

**RETURN** workingHourInt

**END DO**

**DEFINE** method getStaff\_Name1() as return type String

**DO**

**RETURN** text extracted from "staff\_nameField1"

**DEFINE** method getVacancyNumberlbl2() as return type int

**DECLARE** instance variable

**INITIALIZE** vacancyNumberlbl2Int

= 0 **TRY**

vacancyNumberlbl2Int = convert text obtained from vacancyNumberField1  
to integer

**CATCH** NumberFormatException e

**DISPLAY** dialogbox "Please give a valid interger in Vacancy Number".

**RETURN** vacancyNumberlbl2Int

**DEFINE** method getJoining\_Date() as return type String

**DO**

**RETURN** text extracted from "joining\_DateField"

**END DO**

**DEFINE** method getQualification() as return type String

**DO**

**RETURN** text extracted from "qualaificationField"

**END DO**

**DEFINE** method getAppointed\_By() as return type String

**DO**

**RETURN** text extracted from "appointed\_ByField"

**END DO**

**DO**

**INITIALIZE** vno as int and assign getter method getVacancyNumberlbl

**INITIALIZE** salary as int and assign getter method getSalary

**INITIALIZE** working\_Hour as int and assign getter method  
getWorking\_Hour

**INITIALIZE** job\_Type as String and assign getter method getJob\_Type

**INITIALIZE** designation as String and assign getter method  
getDesignation

**INITIALIZE** VNum1 as Boolean and set it to false

**IF** vno & wage & working\_Hour is greater than 0 & jobt\_type &  
designation & shift is not equal to empty String

**FOR** StaffHire duplicate: INGNepal.list

**DO**

**IF** list's getVacancyNumber consists same value  
as vno

**SET** vNum1 as true

**DISPLAY** dialogbox "Fill the field which are required "

**END DO**

**IF** vNum1 is false

**ADD** in INGNepal.list(FullTimeStaffHire(pass  
instance argument designation , job\_Type, vno,  
salary, working\_Hour) constructor as object)

working\_Hour) constructor as object)

**DISPLAY** dialogbox “Thank you for your information “

**END DO**

**DEFINE** method appointFullTime

**DO**

**DECLARE** instance variables

**INITIALIZE** vno as int and assign it to getter method  
getVacancyNumberField1

**INITIALIZE** staff\_name as String and assign it to getter method  
getStaff\_Name

**INITIALIZE** joining\_Date as String and assign it to getter method  
getJoining\_Date

**INITIALIZE** qualification as String and assign it to getter method  
getQualification

**INITIALIZE** appointed\_By as String and assign it to getter method  
getAppointed\_By

**INITIALIZE** vNum as Boolean and set it to false

**IF** vno is greater than 0

**FOR** StaffHire obj: INGNepal.list

**DO**

**IF** list's getVacancyNumber consists same value as vno

**SET** vNum as true

**IF** obj instanceof FullTimeStaffHire



**CREATE** FullTimeStaffHire object casting  
of FullimeStaffHire as h

**IF** FullTimeStaffHire getJoined is true

DISPLAY dialog "Sorry! Staff has been already hired".

**ELSE**

p.hireFullTimeStaff (staff\_name,Qualification,  
joining\_date,appointed\_by)

**DISPLAY** dialogbox "Staff has been hired!"

**END DO**

**ELSE**

**DISPLAY** dialogbox "It is not for fulltime staff Hire".

**END DO**

**IF** vNum is false

**DISPLAY** dialogbox "invalid vacancy"

**END DO**

**DEFINE** method clearFullTime

**SET** staff\_nameField to text "null"

**SET** staff\_nameField1 to text "null"

**SET** vacancyNumberField to text "null"

**SET** vacancyNumberField1 to text "null"

**SET** designationField to text "null"

**SET** job\_TypeField to text "null"

**SET** salaryField to text "null"

**SET** working\_HourField to text "null"

**SET** joining\_DateField to text "null"

**SET** qualificationField to text "null"

**SET** appointed\_ByField to text "null"

**END DO**

**DEFINE** method displayFullTime

**IF** list of class INGNepal is empty

**DISPLAY** dialogbox "Fill the field which are required "

**END DO ELSE**

**DO**

**FOR** StaffHire      obj: list

**DO**

((FullTimeStaffHire)obj).display()

**END DO**

**END DO**

**DEFINE** main method

**DO**

Call FullTimeStaffHireGUI

**END DO**

**DEFINE** abstract method actionPerformed(ActionEvent e)

**IF** btnFullTime

**DO**

**CALL** method addFullTime

**END DO**

**ELSE IF** btnAppoint

**DO**

**CALL** method appointFullTime

**END DO**

**ELSE IF** btnDisplay

**DO**

**CALL** method displayFullTime

**END DO**

**ELSE IF** btnClear is clicked

**DO**

**CALL** method clearFullTime

**END DO**

**END DO**

## Method Descriptions

### INGNepal:

- **diffPartMethod:**

It contains the GUI components in this method, and we have three different buttons listed as its size. Their object in this method too.

- **displayINGNepal:**

The main function in this method is to display button as shown in the list. If we clicked in that button, it will show us their button function that we did.

### PartTimeStaffHireGUI and FullTimeStaffHireGUI:

- **addVacancy:**

GUI component of add part time and full time section is done in this method. This method has components from JLabel, JTextField that help us find their place in GUI.

- **appointVacancy:**

GUI portion of adding part time and full time section is done in this process. This method has components from JLabel, JTextField that help us to locate their place in GUI.

- **getMethod:**

Get method is the accessor method, as each and every method is formed by return int type and string type. We also used methods that are in the type of Int to try and catch.

- **addFullTime and addPartTime:**

The main goal in this method is to help add button functional role. With the help of those buttons it adds and checks whether or not the condition is fulfilled and displays the dialog box according to the condition.

- **appointFullTime and appointPartTime:**

The main goal in this method is to help with the appoint button's functional role. With the help of those buttons it adds and checks whether or not the condition is fulfilled and displays the dialog box according to the condition.

- **dispalayPartTime and displayFullTime:**

The main goal in this method is to help with the display button's functional role. When the button is clicked it displays the output we mentioned from those GUI's JTextField.

- **clearPartTime and clearFullTime:**

The main objective in this method is to assist in the functional role of a clear button. It helps clear the entire TextField when clicking in a simple circle.

- **terminatedPartTime:**

The main goal in this method is to help with the terminate button's functional role. It sets null and terminate staff values for all staff.

## Testing

### Test 1: Output

Result

```

2 Dir(s) 194,839,171,072 bytes free

C:\Users\14105\Documents\Coursework-Programing>javac INGNepal.java

C:\Users\14105\Documents\Coursework-Programing>dir
Volume in drive C has no label.
Volume Serial Number is A09C-5362

Directory of C:\Users\14105\Documents\Coursework-Programing

04/16/2020  04:36 PM    <DIR>          .
04/16/2020  04:36 PM    <DIR>          ..
04/16/2020  04:36 PM                2,720 FullTimeStaffHire.class
04/13/2020  04:33 PM                2,672 FullTimeStaffHire.java
04/16/2020  04:36 PM                7,948 FullTimeStaffHireGUI.class
04/16/2020  01:04 PM            11,311 FullTimeStaffHireGUI.java
04/16/2020  04:36 PM                2,475 INGNepal.class
04/15/2020  01:13 PM                2,348 INGNepal.java
04/16/2020  04:36 PM                3,176 PartTimeStaffHire.class
04/13/2020  04:35 PM                3,741 PartTimeStaffHire.java
04/16/2020  04:36 PM                8,855 PartTimeStaffHireGUI.class
04/16/2020  01:40 PM            14,133 PartTimeStaffHireGUI.java
04/16/2020  04:36 PM                1,288 StaffHire.class
04/11/2020  07:11 PM                1,100 StaffHire.java
               12 File(s)              61,767 bytes
               2 Dir(s) 194,838,769,664 bytes free

C:\Users\14105\Documents\Coursework-Programing>java INGNepal

```

Figure 5: Showing the Directory

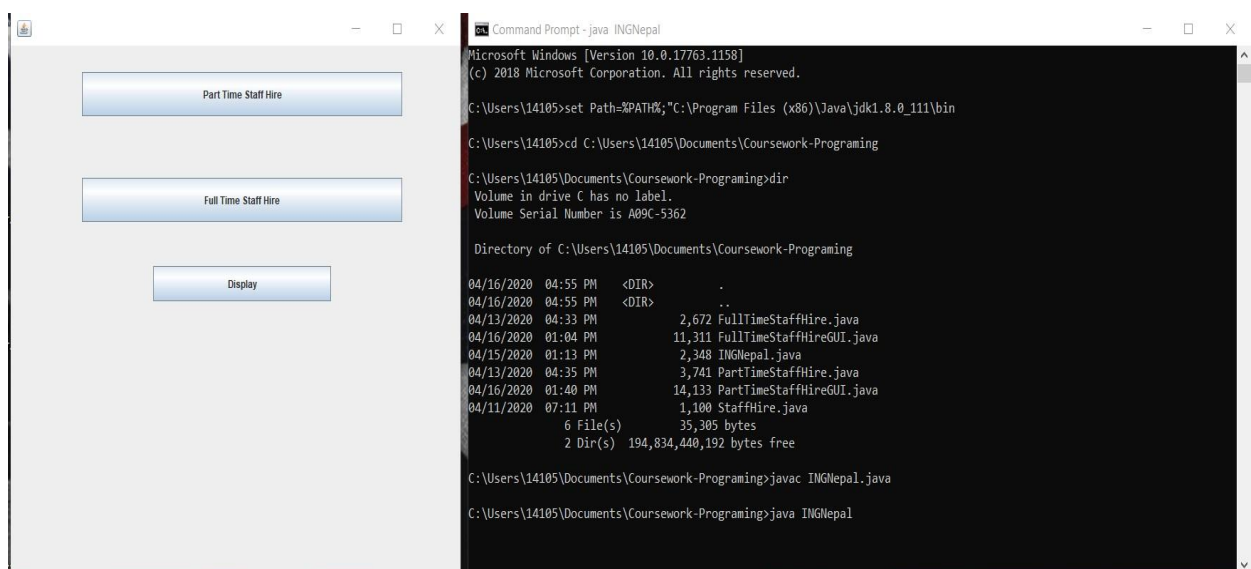


Figure 6: Output of command Prompt

**Test 2:**

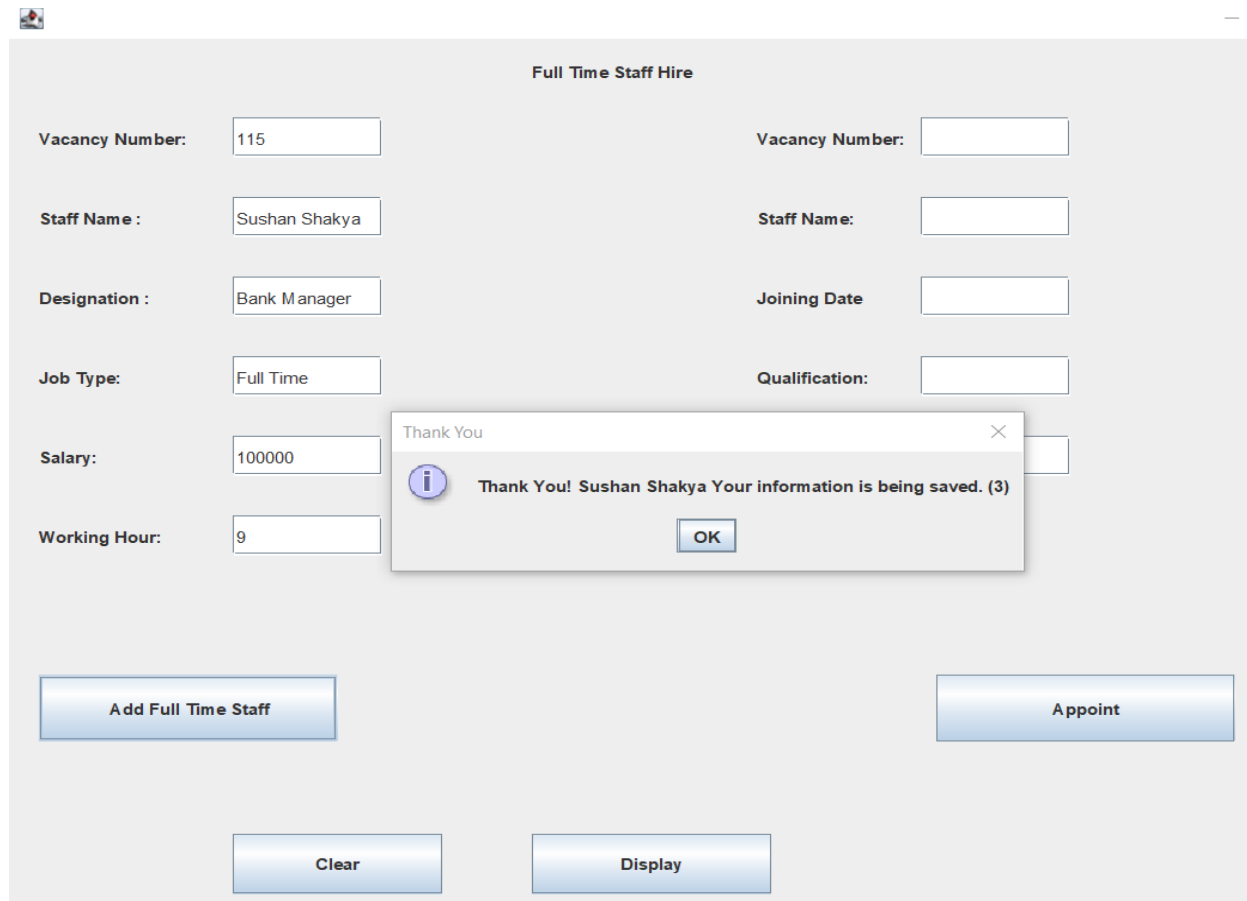
Add Vacancy of Full Time Staff.

Result:

The screenshot shows a web application window titled "Full Time Staff Hire". It contains two columns of input fields. The left column has fields for "Vacancy Number:" (value: 115), "Staff Name:" (value: Sushan Shakya), "Designation:" (value: Bank Manager), "Job Type:" (value: Full Time), "Salary:" (value: 100000), and "Working Hour:" (value: 9). The right column has empty fields for "Vacancy Number:", "Staff Name:", "Joining Date", "Qualification:", and "Appointed By:". At the bottom, there are four buttons: "Add Full Time Staff" (left), "Appoint" (right), "Clear" (bottom center-left), and "Display" (bottom center-right).

Figure 7: Adding Information





The screenshot shows a web application titled "Full Time Staff Hire". It contains two columns of input fields. The left column has fields for "Vacancy Number:" (value: 115), "Staff Name:" (value: Sushan Shakya), "Designation:" (value: Bank Manager), "Job Type:" (value: Full Time), "Salary:" (value: 100000), and "Working Hour:" (value: 9). The right column has fields for "Vacancy Number:", "Staff Name:", "Joining Date", and "Qualification:". Below the form are four buttons: "Add Full Time Staff", "Appoint", "Clear", and "Display". A modal dialog box titled "Thank You" is overlaid on the form, displaying the message "Thank You! Sushan Shakya Your information is being saved. (3)" and an "OK" button.

Figure 9: Output After Filling Up info

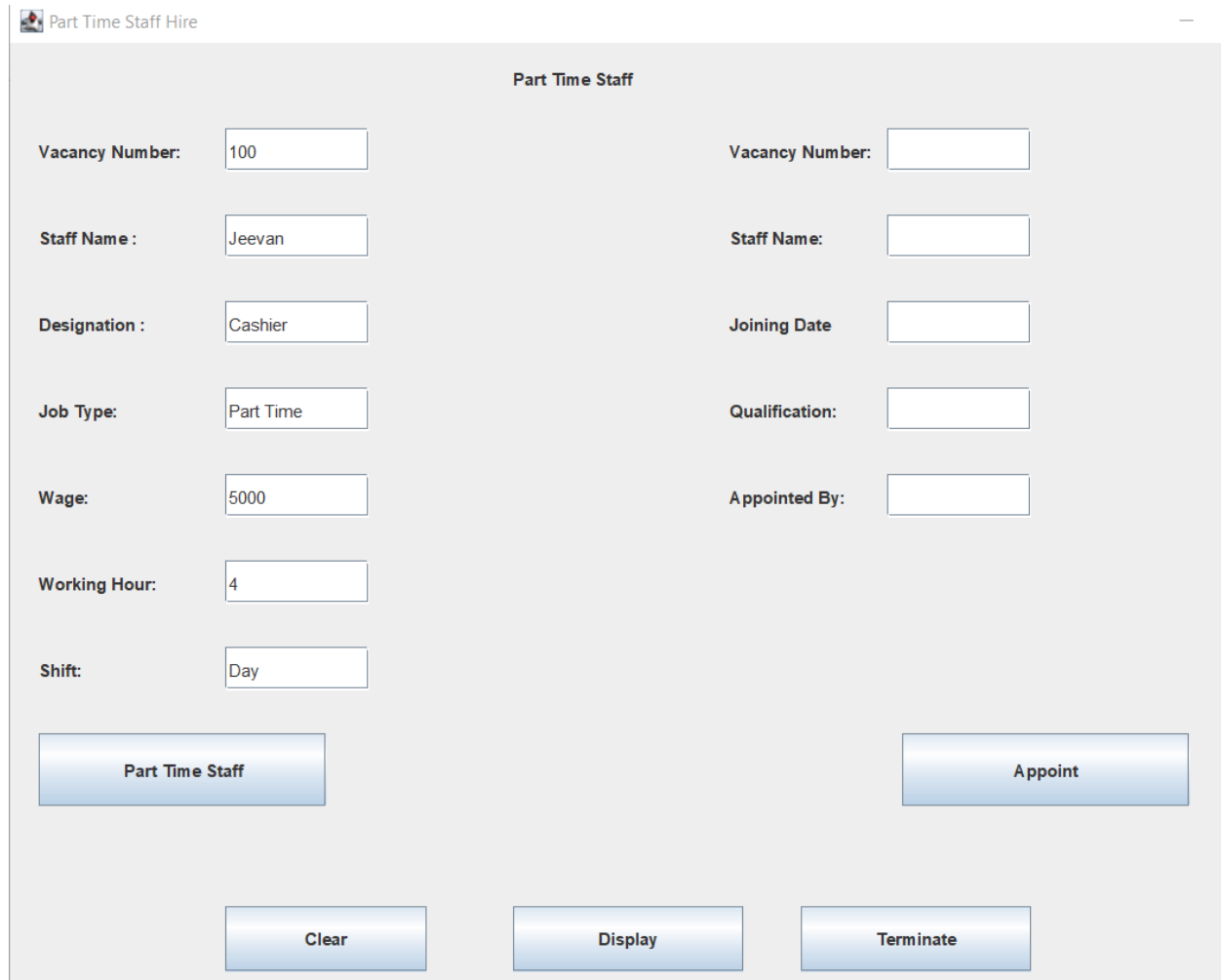
BlueJ: Terminal Window - Coursework-Programing

```
Options
Staff Detail Information
Designation: Bank Manager
JobType:Full time
Vacancy Number: 115
Staff Name:
Joining Date:
Qualification:
Appointed By:
Joined:false
```

Figure 8: Displaying Information

## Add Vacancy of PartTime Staff

Result:



The screenshot shows a window titled "Part Time Staff Hire" with a sub-header "Part Time Staff". The form is divided into two columns of input fields. The left column contains fields for "Vacancy Number:" (value: 100), "Staff Name :" (value: Jeevan), "Designation :" (value: Cashier), "Job Type:" (value: Part Time), "Wage:" (value: 5000), "Working Hour:" (value: 4), and "Shift:" (value: Day). The right column contains fields for "Vacancy Number:" (empty), "Staff Name:" (empty), "Joining Date" (empty), "Qualification:" (empty), and "Appointed By:" (empty). Below the input fields, there are three buttons: "Part Time Staff" on the left, "Appoint" on the right, and a row of three buttons at the bottom: "Clear", "Display", and "Terminate".

Field	Value
Vacancy Number:	100
Staff Name :	Jeevan
Designation :	Cashier
Job Type:	Part Time
Wage:	5000
Working Hour:	4
Shift:	Day
Vacancy Number:	
Staff Name:	
Joining Date	
Qualification:	
Appointed By:	

Buttons: Part Time Staff, Appoint, Clear, Display, Terminate

*Figure 10: Adding Information In PartTimeStaff*

Part Time Staff Hire

**Part Time Staff**

Vacancy Number:	<input type="text" value="100"/>	Vacancy Number:	<input type="text"/>
Staff Name :	<input type="text" value="Jeevan"/>	Staff Name:	<input type="text"/>
Designation :	<input type="text" value="Cashier"/>	Joining Date	<input type="text"/>
Job Type:	<input type="text" value="Part Time"/>	Qualification:	<input type="text"/>
Wage:	<input type="text" value="5000"/>		<input type="text"/>
Working Hour:	<input type="text" value="4"/>		
Shift:	<input type="text" value="Day"/>		

Thank You


 Thank You! Jeevan for the detail! (2)

Figure 12: Output after adding Information

BlueJ: Terminal Window - Coursework-Programing

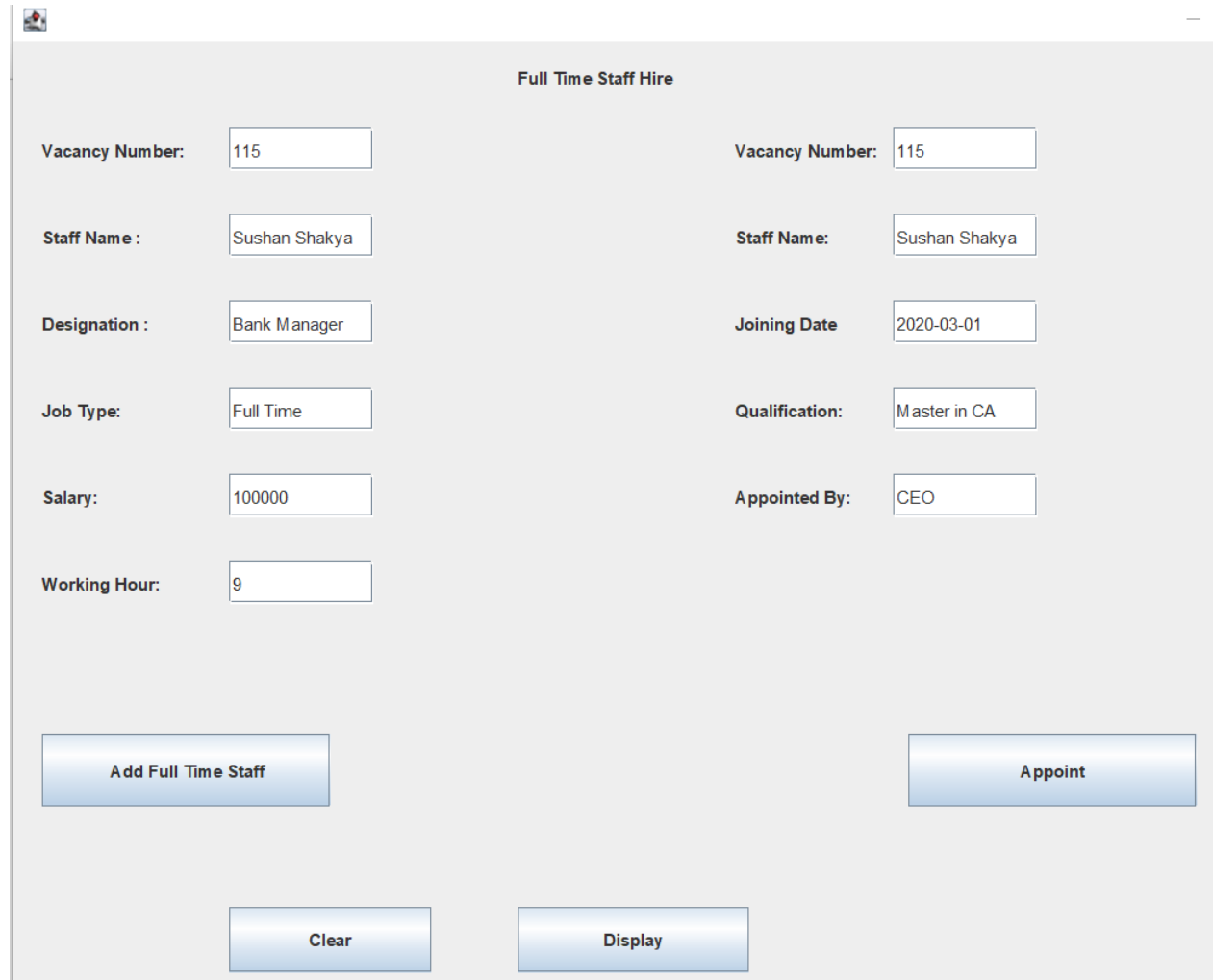
Options

```
Staff Detail Information
Designation: Cashier
JobType:Part Time
Vacancy Number: 100
```

Figure 11: Display output

## Appoint Full Time Staff

Result:



**Full Time Staff Hire**

Vacancy Number:	<input type="text" value="115"/>	Vacancy Number:	<input type="text" value="115"/>
Staff Name :	<input type="text" value="Sushan Shakya"/>	Staff Name:	<input type="text" value="Sushan Shakya"/>
Designation :	<input type="text" value="Bank Manager"/>	Joining Date	<input type="text" value="2020-03-01"/>
Job Type:	<input type="text" value="Full Time"/>	Qualification:	<input type="text" value="Master in CA"/>
Salary:	<input type="text" value="100000"/>	Appointed By:	<input type="text" value="CEO"/>
Working Hour:	<input type="text" value="9"/>		

Figure 13: Adding Info in Full Time Staff

 BlueJ: Terminal Window - Coursework-Programing

## Options

```
Staff Detail Information
Designation: Bank Manager
JobType:Full time
Vacancy Number: 115
Staff Name: Staff Name :
Joining Date: Joining Date
Qualification: Qualification:
Appointed By: Appointed By:
Joined:true
Staff Detail Information
Designation: Bank Manager
JobType:Full time
Vacancy Number: 115
Staff Name: Staff Name :
Joining Date: Joining Date
Qualification: Qualification:
Appointed By: Appointed By:
Joined:true
```

Figure 14: Display Info of Full Time Staff Hire

## Appoint Part Time Staff


Result:

 Part Time Staff Hire

**Part Time Staff**

Vacancy Number:	<input type="text" value="100"/>	Vacancy Number:	<input type="text" value="100"/>
Staff Name :	<input type="text" value="Jeevan"/>	Staff Name:	<input type="text" value="Jeevan"/>
Designation :	<input type="text" value="Cashier"/>	Joining Date	<input type="text" value="2020-06-01"/>
Job Type:	<input type="text" value="Part Time"/>	Qualification:	<input type="text" value="Bachelor"/>
Wage:	<input type="text" value="5000"/>	Appointed By:	<input type="text" value="Bank Manager"/>
Working Hour:	<input type="text" value="4"/>		
Shift:	<input type="text" value="Day"/>		
<input type="button" value="Part Time Staff"/>		<input type="button" value="Appoint"/>	
<input type="button" value="Clear"/>		<input type="button" value="Display"/>	<input type="button" value="Terminate"/>

*Figure 15: Appointing PartTimeStaff*


 BlueJ: Terminal Window - Coursework-Programing

## Options

```
Will be joining at 2020-01-01
Staff Detail Information
Designation: Cashier
JobType:PartTime
Vacancy Number: 100
Staff Name: Jeevan
Wages per Hour: 5000
Working Hour: 4
Date of Joining: 2020-01-01
Qualification: Bachelor
Appointed By: Bank Manager
Income Per Day: 20000
Joined:true
```

*Figure 16: Display Output*


## Terminate Part Time Staff: Result:

 Part Time Staff Hire

**Part Time Staff**

Vacancy Number:	<input type="text" value="100"/>	Vacancy Number:	<input type="text" value="100"/>
Staff Name :	<input type="text" value="Jeevan"/>	Staff Name:	<input type="text" value="Jeevan"/>
Designation :	<input type="text" value="Cashier"/>	Joining Date	<input type="text" value="2020-06-01"/>
Job Type:	<input type="text" value="Part Time"/>	Qualification:	<input type="text" value="Bachelor"/>
Wage:	<input type="text" value="5000"/>	Appointed By:	<input type="text" value="Bank Manager"/>
Working Hour:	<input type="text" value="4"/>		
Shift:	<input type="text" value="Day"/>		

Figure 17: Termanating Part Time Staff

 BlueJ: Terminal Window - Coursework-Programing

```
Options
Will be joining at 2020-01-01
Staff has been terminated
Staff Detail Information
Designation: Cashier
JobType:PartTime
Vacancy Number: 100
```

Figure 18: Display Output



**Test 3:**


Result:

Part Time Staff Hire

**Part Time Staff**

Vacancy Number:	15A	Vacancy Number:	
Staff Name :	Jeevan	Staff Name:	
Designation :	Cashier	Joining Date	
Job Type:	Part Time	Qualification:	
Wage:	5000		
Working Hour:	4		
Shift:	day		

**Error**

 Please give a valid interger in Vacancy Number

OK

Part Time Staff

Appoint

Clear

Display

Terminate

*Figure 19: Showing error while putting string in vacancy number*

Part Time Staff Hire

**Part Time Staff**

Vacancy Number:	14	Vacancy Number:	15
Staff Name :	Jeevan	Staff Name:	Jeevan
Designation :	Cashier	Joining Date	2020-06-01
Job Type:	Full Time	Qualification:	Bachelor
Wage:	5000		Bank Manager
Working Hour:	4		
Shift:	Day		

Part Time Staff

Appoint

Clear Display Terminate

Error  
invalid vacancy  
OK

Figure 20: Output of result

## Error Detection and Correction

### Syntax Error:

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

@Override
public void actionPerformed(ActionEvent e){
    if (e.getSource() = btnPartTime){
        addPartTime();
    }else if(e.getSource() == btnAppoint){
        appointPartTime();
    }else if(e.getSource() == btnDisplay){
        displayPartTime();
    }else if(e.getSource() == btnClear){
        clearPartTime();
    }else if(e.getSource() == btnTerminate){
        terminatePartTime();
    }
}
```

Figure 21: Syntax Error

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

@Override
public void actionPerformed(ActionEvent e){
    if (e.getSource() == btnPartTime){
        addPartTime();
    }else if(e.getSource() == btnAppoint){
        appointPartTime();
    }else if(e.getSource() == btnDisplay){
        displayPartTime();
    }else if(e.getSource() == btnClear){
        clearPartTime();
    }else if(e.getSource() == btnTerminate){
        terminatePartTime();
    }
}
```

Figure 22: Correction Of Syntax Error

**Semantic Error:**

```
public String getWorking_Hour(){  
    int working_HourInt=0;  
    try{  
        working_HourInt = Integer.parseInt(working_HourField.getText());  
    }  
    catch(NumberFormatException e){  
        JOptionPane.showMessageDialog(frame,"Please give a valid interger in Working Hour","Error",  
            JOptionPane.ERROR_MESSAGE);  
    }  
    return working_HourInt;  
}
```

*Figure 23: Sementic Error*

```
public int getWorking_Hour(){  
    int working_HourInt=0;  
    try{  
        working_HourInt = Integer.parseInt(working_HourField.getText());  
    }  
    catch(NumberFormatException e){  
        JOptionPane.showMessageDialog(frame,"Please give a valid interger in Working Hour","Error",  
            JOptionPane.ERROR_MESSAGE);  
    }  
    return working_HourInt;  
}
```

*Figure 24: Correction Of Sementic Error*

**Logical Error:**

```

public void addPartTime(){
    boolean Vacancy = true;
    if(getVacancyNumber1() == 0 || getWage() == 0 || getWorking_Hour() == 0 ||
       getName() == "" || getDesignation() == "" || getJob_Type() == "" ||
       getShift() == ""){
        JOptionPane.showMessageDialog(frame,"Fill all the Field","Attention",
        JOptionPane.WARNING_MESSAGE);
    }else{
        for(StaffHire obj : INGNepal.list){
            if(obj.getVacancyNumber() == getVacancyNumber1()){
                Vacancy = true;
            }
        }
        if(!Vacancy){
            INGNepal.list.add(new PartTimeStaffHire(getDesignation(),getJob_Type(),getWage(),getWorking_Hour(),getVacancyNumber1(),
            getShift()));

            JOptionPane.showMessageDialog(frame,"Thank you " +getName()+" for your information ","Thank you",
            JOptionPane.INFORMATION_MESSAGE);
        }else{
            JOptionPane.showMessageDialog(frame, "Staff has been added use another Vacancy Number. ","Error",
            JOptionPane.ERROR_MESSAGE);
        }
    }
}

```

Figure 25: Logical Error

```

public void addPartTime(){
    boolean Vacancy = false;
    if(getVacancyNumber1() == 0 || getWage() == 0 || getWorking_Hour() == 0 ||
       getName() == "" || getDesignation() == "" || getJob_Type() == "" ||
       getShift() == ""){
        JOptionPane.showMessageDialog(frame,"Fill all the Field","Attention",
        JOptionPane.WARNING_MESSAGE);
    }else{
        for(StaffHire obj : INGNepal.list){
            if(obj.getVacancyNumber() == getVacancyNumber1()){
                Vacancy = true;
            }
        }
        if(!Vacancy){
            INGNepal.list.add(new PartTimeStaffHire(getDesignation(),getJob_Type(),getWage(),getWorking_Hour(),getVacancyNumber1(),
            getShift()));

            JOptionPane.showMessageDialog(frame,"Thank you " +getName()+" for your information ","Thank you",
            JOptionPane.INFORMATION_MESSAGE);
        }else{
            JOptionPane.showMessageDialog(frame, "Staff has been added use another Vacancy Number. ","Error",
            JOptionPane.ERROR_MESSAGE);
        }
    }
}

```

Figure 26: Correction Of Logical Error

## Conclusion

Our main goal in this coursework was to build a class and render Interface for those classes INGNepal, PartTimeStaffHireGUI, and FullTimeStaffHireGUI. I have learned all the concepts after working on this coursework to make a GUI and add different Button, TextField and other components, and their call process and Interface feature. We're learning how to create Label, button, textfield and others. We had tested for each method we told us to do while doing this process. We also need to make class diagrams, and write pseudocodes that are easy to understand. We have learned a lot about the principles of programming. We also heard about error and its correction. We have encountered some difficulties when doing the coursework. We are able to complete all of the coursework with the teacher.

## References

*Study.com*. (2020). Retrieved from Study: <https://study.com/academy/lesson/pseudocode-definition-examples-quiz.html>

*UML*. (2020). Retrieved from Tutorials Point: [https://www.tutorialspoint.com/uml/uml\\_class\\_diagram.htm](https://www.tutorialspoint.com/uml/uml_class_diagram.htm)

**Appendix:****INGNepal**

```
/* Making GUI of INGNepal */
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JFrame;
import javax.swing.JButton;
import java.util.ArrayList;

public class INGNepal implements ActionListener{
    private JFrame frame;
    private JButton btnPartTimeStaffHire;
    private JButton btnFullTimeStaffHire;
    private JButton btnDisplay;
    public static ArrayList<StaffHire> list = new ArrayList<StaffHire>();

    /* GUI of INGNepal */
    public INGNepal(){
        frame=new JFrame();

        diffPart();

        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(650, 450);
        frame.setLayout(null);
        frame.setVisible(true);
    }
}
```

```
}

/* GUI components */

public void diffPart(){

    btnPartTimeStaffHire = new JButton("Part Time Staff Hire");
    btnPartTimeStaffHire.setBounds(100, 30, 450, 50);
    btnPartTimeStaffHire.addActionListener(this);
    frame.add(btnPartTimeStaffHire);

    btnFullTimeStaffHire = new JButton("Full Time Staff Hire");
    btnFullTimeStaffHire.setBounds(100, 150, 450, 50);
    btnFullTimeStaffHire.addActionListener(this);
    frame.add(btnFullTimeStaffHire);

    btnDisplay = new JButton("Display");
    btnDisplay.setBounds(200, 250, 250, 40);
    btnDisplay.addActionListener(this);
    frame.add(btnDisplay);
}

/* Main Method */

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

}

/* Method to show function od display buttom*/

public void displayINGNepal() {
    for (StaffHire obj : list) {
        if (obj instanceof PartTimeStaffHire) {
```



```
        System.out.println("Part Time Detail:");
        ((PartTimeStaffHire) obj).display();
    } else if (obj instanceof FullTimeStaffHire) {

System.out.println("_____");
        System.out.println("Full Time Detail:");
        ((FullTimeStaffHire) obj).display();
    }
}
}
/* Abstract Method */
@Override
public void actionPerformed(ActionEvent e){
    if(e.getSource() == btnPartTimeStaffHire){
        new PartTimeStaffHireGUI();
    }
    else if (e.getSource() == btnFullTimeStaffHire){
        new FullTimeStaffHireGUI();
    }
    else if (e.getSource() == btnDisplay){
        displayINGNepal();
    }
}
}
```

**FullTimeStaffHire GUI**

```
/*Making GUI of FullTimeStaffHire and its function */
```

```
import java.awt.event.ActionEvent;
```

```
import java.awt.event.ActionListener;
```

```
import javax.swing.JFrame;
```

```
import javax.swing.JButton;
```

```
import javax.swing.JLabel;
```

```
import javax.swing.JTextField;
```

```
import javax.swing.JOptionPane;
```

```
public class FullTimeStaffHireGUI implements ActionListener{
```

```
    private JFrame frame;
```

```
    private JLabel label;
```

```
    private JLabel staff_name,designation,job_Type,salary,working_Hour,vacancyNumberlbl;
```

```
    private JTextField  
staff_nameField,vacancyNumberField,working_HourField,designationField,job_TypeField,salaryFi  
eld;
```

```
    private JLabel staff_namelbl, joining_Date,qualification,appointed_By,vacancyNumberlbl2;
```

```
    private JTextField  
staff_nameField1,vacancyNumberField1,joining_DateField,qualificationField,appointed_ByField;
```

```
    private JButton btnAppoint;
```

```
    private JButton btnDisplay;
```

```
    private JButton btnClear;
```

```
    private JButton btnFullTime;
```

```
/* Method containing GUI */
```

```
public FullTimeStaffHireGUI(){
```

```
    frame = new JFrame();
```

```
label = new JLabel("Full Time Staff Hire");
label.setBounds(350, 10, 200, 30);
frame.add(label);

addVacancyFull();
addAppointForVacancy();
buttons();

frame.setSize(950, 750);
frame.setLayout(null);
frame.setVisible(true);
}
/* GUI of addVacancyFull*/
public void addVacancyFull() {
    staff_name = new JLabel("Staff Name : ");
    staff_name.setBounds(20, 120, 90, 30);
    frame.add(staff_name);

    staff_nameField = new JTextField();
    staff_nameField.setBounds(150, 120, 100, 30);
    frame.add(staff_nameField);

    vacancyNumberlbl = new JLabel("Vacancy Number:");
    vacancyNumberlbl.setBounds(20, 60, 100, 30);
    frame.add(vacancyNumberlbl);

    vacancyNumberField = new JTextField();
    vacancyNumberField.setBounds(150, 60, 100, 30);
```

```
frame.add(vacancyNumberField);

designation = new JLabel("Designation : ");
designation.setBounds(20, 180, 100, 30);
frame.add(designation);

designationField = new JTextField();
designationField.setBounds(150, 180, 100, 30);
frame.add(designationField);

job_Type = new JLabel("Job Type: ");
job_Type.setBounds(20, 240, 100, 30);
frame.add(job_Type);

job_TypeField = new JTextField();
job_TypeField.setBounds(150, 240, 100, 30);
frame.add(job_TypeField);

salary = new JLabel("Salary:");
salary.setBounds(20, 300, 100, 30);
frame.add(salary);

salaryField = new JTextField();
salaryField.setBounds(150, 300, 100, 30);
frame.add(salaryField);

working_Hour = new JLabel("Working Hour:");
working_Hour.setBounds(20, 360, 100, 30);
frame.add(working_Hour);
```

```
        working_HourField = new JTextField();
        working_HourField.setBounds(150, 360, 100, 30);
        frame.add(working_HourField);

    }

    /*GUI of addAppointForVacancy*/
    public void addAppointForVacancy(){
        staff_nameLbl = new JLabel("Staff Name:");
        staff_nameLbl.setBounds(500, 120, 100, 30);
        frame.add(staff_nameLbl);

        staff_nameField1 = new JTextField();
        staff_nameField1.setBounds(610, 120, 100, 30);
        frame.add(staff_nameField1);

        vacancyNumberLbl2 = new JLabel("Vacancy Number:");
        vacancyNumberLbl2.setBounds(500, 60, 100, 30);
        frame.add(vacancyNumberLbl2);

        vacancyNumberField1 = new JTextField();
        vacancyNumberField1.setBounds(610, 60, 100, 30);
        frame.add(vacancyNumberField1);

        joining_Date = new JLabel("Joining Date");
        joining_Date.setBounds(500, 180, 100, 30);
        frame.add(joining_Date);

        joining_DateField = new JTextField();
```

```
    joining_DateField.setBounds(610,180, 100, 30);  
    frame.add(joining_DateField);  
  
    qualification = new JLabel("Qualification: ");  
    qualification.setBounds(500, 240, 100, 30);  
    frame.add(qualification);  
  
    qualificationField = new JTextField();  
    qualificationField.setBounds(610, 240, 100, 30);  
    frame.add(qualificationField);  
  
    appointed_By = new JLabel("Appointed By: ");  
    appointed_By.setBounds(500, 300, 100, 30);  
    frame.add(appointed_By);  
  
    appointed_ByField = new JTextField();  
    appointed_ByField.setBounds(610, 300, 100, 30);  
    frame.add(appointed_ByField);  
  
}  
/*GUI of Buttons*/  
public void buttons() {  
    btnFullTime = new JButton("Add Full Time Staff");  
    btnFullTime.setBounds(20, 480, 200, 50);  
    btnFullTime.addActionListener(this);  
    frame.add(btnFullTime);  
  
    btnAppoint = new JButton("Appoint");  
    btnAppoint.setBounds(620, 480, 200, 50);
```

```
        btnAppoint.addActionListener(this);

        frame.add(btnAppoint);


        btnClear = new JButton("Clear");
        btnClear.setBounds(150, 600, 140, 45);
        btnClear.addActionListener(this);
        frame.add(btnClear);


        btnDisplay = new JButton("Display");
        btnDisplay.setBounds(350, 600, 160, 45);
        btnDisplay.addActionListener(this);
        frame.add(btnDisplay);
    }

    /* get method of Staff Name */
    public String getstaff_name(){
        return staff_nameField.getText();
    }

    /* get method of Vacancy Number */
    public int getVacancyNumberIbl() {
        int vacancyNumberInt = 0;
        try {
            vacancyNumberInt = Integer.parseInt(vacancyNumberField.getText());

        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(frame, "Please give a valid interger in Vacancy
Number", "Error",
                JOptionPane.ERROR_MESSAGE);
        }
        return vacancyNumberInt;
    }
}
```

```
/* get method of designation */  
public String getDesignation(){  
    return designationField.getText();  
}  
  
/* get method of job type */  
public String getJob_Type(){  
    return job_TypeField.getText();  
}  
  
/* get method of Salary*/  
public int getSalary() {  
    int salaryInt = 0;  
    try {  
        salaryInt = Integer.parseInt(salaryField.getText());  
    } catch (NumberFormatException e) {  
        JOptionPane.showMessageDialog(frame, "Please give a valid interger in Salary","Error",  
            JOptionPane.ERROR_MESSAGE);  
    }  
    return salaryInt;  
}  
  
/* get method of working hour */  
public int getWorking_Hour(){  
    int working_HourInt=0;  
    try{  
        working_HourInt = Integer.parseInt(working_HourField.getText());  
    }  
    catch(NumberFormatException e){  
        JOptionPane.showMessageDialog(frame,"Please give a valid interger in Working  
Hour","Error",  
            JOptionPane.ERROR_MESSAGE);  
    }  
}
```



```
    }  
    return working_HourInt;  
}  
/* get method of Staff Name1 */  
public String getStaff_Name1() {  
    return staff_nameField1.getText();  
}  
/* get method of Vacancy Number 2 */  
public int getVacancyNumberIbl2() {  
    int vacancyNumberIbl2Int = 0;  
    try {  
        vacancyNumberIbl2Int = Integer.parseInt(vacancyNumberField1.getText());  
    } catch (NumberFormatException e) {  
        JOptionPane.showMessageDialog(frame, "Enter a valid int", "Error",  
JOptionPane.ERROR_MESSAGE);  
    }  
    return vacancyNumberIbl2Int;  
  
}  
/* get method of Joining date */  
public String getJoining_Date() {  
    return joining_DateField.getText();  
}  
/* get method of qualification */  
public String getQualification() {  
    return qualificationField.getText();  
}  
/* get method of appointed by */  
public String getAppointed_By() {  
    return appointed_ByField.getText();
```

```

    }

    /* Function of addFullTime Button*/
    public void addFullTime(){
        int vno = getVacancyNumberlbl();
        int salary = getSalary();
        int working_Hour = getWorking_Hour();
        String job_Type = getJob_Type();
        String designation = getDesignation();
        boolean vNum1 = false;

        if (vno > 0 && salary > 0 && working_Hour > 0 && !job_Type.equals("") &&
!designation.equals("")) {
            for (StaffHire duplicate : INGNepal.list) {
                if (duplicate.getVacancyNumber() == vno) {
                    vNum1 = true;
                    JOptionPane.showMessageDialog(frame,
                        "Staff has been already added. Please try a different vacancy number.", "Invalid
Input",
                        JOptionPane.ERROR_MESSAGE);
                    break;
                }
            }
            if (!vNum1) {
                INGNepal.list.add(new FullTimeStaffHire(designation, job_Type, vno, salary,
working_Hour));
                JOptionPane.showMessageDialog(frame, "Thank You! " + getstaff_name()
                    + " Your information is being saved. (" + INGNepal.list.size() + ")", "Thank You",
                    JOptionPane.INFORMATION_MESSAGE);
            }
        }
    }
}

```

```
/* Fucntion of appointFullTime Buttion*/
```

```
public void appointFullTime(){
    int vno=getVacancyNumberlbl2();
    String staffName=getStaff_Name1();
    String joinDate=getJoining_Date();
    String Qualification=getQualification();
    String appointedBy=getAppointed_By();
    boolean vNum=false;
    for(StaffHire obj:INGNepal.list){
        if(obj.getVacancyNumber()==vno){
            vNum=true;
            if(obj instanceof FullTimeStaffHire){
                FullTimeStaffHire p=(FullTimeStaffHire)obj;
                if(p.getJoined()==true){
                    JOptionPane.showMessageDialog(frame,"Sorry! Staff has been already hired");
                }else{
                    p.hireFullTimeStaff(staffName,joinDate,Qualification,appointedBy);
                    JOptionPane.showMessageDialog(frame,"Staff has been hired!");
                    break;
                }
            }else{
                JOptionPane.showMessageDialog(frame,"not for fulltime staff Hire");
                break;
            }
        }
    }
    if(!vNum){
        JOptionPane.showMessageDialog(frame,"invalid vacancy");
    }
}
```

```
    }  
}  
  
/* Fucntion of clearPartTime Button*/  
public void clearPartTime(){  
    staff_nameField.setText(null);  
    staff_nameField1.setText(null);  
    vacancyNumberField.setText(null);  
    vacancyNumberField1.setText(null);  
    designationField.setText(null);  
    job_TypeField.setText(null);  
    salaryField.setText(null);  
    working_HourField.setText(null);  
    joining_DateField.setText(null);  
    qualificationField.setText(null);  
    appointed_ByField.setText(null);  
}  
  
/* Fucntion of displayFullTime Button*/  
public void displayFullTime(){  
    if(INGNepal.list.isEmpty()){  
        JOptionPane.showMessageDialog(frame, "Fill the field which are required ", "Error",  
        JOptionPane.ERROR_MESSAGE);  
  
    }else{  
        for (StaffHire obj: INGNepal.list){  
            ((FullTimeStaffHire) obj).display();  
        }  
    }  
}
```

```
public static void main (String[] args){  
    new FullTimeStaffHireGUI();  
}  
/*Abstract Method*/  
@Override  
public void actionPerformed(ActionEvent e){  
    if (e.getSource() == btnFullTime){  
        addFullTime();  
    }else if(e.getSource() == btnAppoint){  
        appointFullTime();  
    }else if(e.getSource() == btnDisplay){  
        displayFullTime();  
    }else if(e.getSource() == btnClear){  
        clearPartTime();  
    }  
}  
}
```

**PartTimeStaffHireGUI**

```
/* Making GUI of PartTimeStaffHire and its Function */
```

```
import java.awt.event.ActionEvent;
```

```
import java.awt.event.ActionListener;
```

```
import javax.swing.JFrame;
```

```
import javax.swing.JButton;
```

```
import javax.swing.JLabel;
```

```
import javax.swing.JTextField;
```

```
import javax.swing.JOptionPane;
```

```
public class PartTimeStaffHireGUI implements ActionListener{
```

```
    private JFrame frame;
```

```
    private JLabel label;
```

```
    private JLabel name,vacancyNumber1,designation,job_Type,wage,working_Hour,shift;
```

```
    private                                                                                      JTextField  
nameField,vacancyNumberField,working_HourField,designationField,job_TypeField,wageField,shi  
ftField;
```

```
    private JLabel nameLbl,vacancyNumberLbl,joining_Date,qualification,appointed_By;
```

```
    private                                                                                      JTextField  
nameField1,vacancyNumberField1,joining_DateField,qualificationField,appointed_ByField;
```

```
    private JButton btnAppoint;
```

```
    private JButton btnDisplay;
```

```
    private JButton btnClear;
```

```
    private JButton btnTerminate;
```

```
    private JButton btnPartTime;
```

```
    private JButton btnAdd;
```

```
/* Method containing GUI */
```

```
public PartTimeStaffHireGUI(){  
    frame = new JFrame("Part Time Staff Hire");  
  
    label = new JLabel("Part Time Staff ");  
    label.setBounds(350, 10, 200, 30);  
    frame.add(label);  
  
    addVacancyPart();  
    addAppointForVacancyPart();  
    buttons();  
  
    frame.setSize(950, 700);  
    frame.setLayout(null);  
    frame.setVisible(true);  
}  
  
/* GUI of addVacancyPart*/  
public void addVacancyPart() {  
    name = new JLabel("Staff Name : ");  
    name.setBounds(20, 120, 90, 30);  
    frame.add(name);  
  
    nameField = new JTextField();  
    nameField.setBounds(150, 120, 100, 30);  
    frame.add(nameField);  
  
    vacancyNumber1 = new JLabel("Vacancy Number:");  
    vacancyNumber1.setBounds(20, 60, 100, 30);  
    frame.add(vacancyNumber1);
```

```
vacancyNumberField = new JTextField();  
vacancyNumberField.setBounds(150, 60, 100, 30);  
frame.add(vacancyNumberField);
```

```
designation = new JLabel("Designation : ");  
designation.setBounds(20, 180, 100, 30);  
frame.add(designation);
```

```
designationField = new JTextField();  
designationField.setBounds(150, 180, 100, 30);  
frame.add(designationField);
```

```
job_Type = new JLabel("Job Type: ");  
job_Type.setBounds(20, 240, 100, 30);  
frame.add(job_Type);
```

```
job_TypeField = new JTextField();  
job_TypeField.setBounds(150, 240, 100, 30);  
frame.add(job_TypeField);
```

```
wage = new JLabel("Wage:");  
wage.setBounds(20, 300, 100, 30);  
frame.add(wage);
```

```
wageField = new JTextField();  
wageField.setBounds(150, 300, 100, 30);  
frame.add(wageField);
```



```
    working_Hour = new JLabel("Working Hour:");
    working_Hour.setBounds(20, 360, 100, 30);
    frame.add(working_Hour);

    working_HourField = new JTextField();
    working_HourField.setBounds(150, 360, 100, 30);
    frame.add(working_HourField);

    shift = new JLabel("Shift:");
    shift.setBounds(20, 420, 100, 30);
    frame.add(shift);

    shiftField = new JTextField();
    shiftField.setBounds(150, 420, 100, 30);
    frame.add(shiftField);

}

/*GUI of addAppointVacancyPart*/
public void addAppointForVacancyPart(){
    nameLbl = new JLabel("Staff Name:");
    nameLbl.setBounds(500, 120, 100, 30);
    frame.add(nameLbl);

    nameField1 = new JTextField();
    nameField1.setBounds(610, 120, 100, 30);
    frame.add(nameField1);

    vacancyNumberLbl = new JLabel("Vacancy Number:");
```

```
vacancyNumberLbl.setBounds(500, 60, 100, 30);  
frame.add(vacancyNumberLbl);
```

```
vacancyNumberField1 = new JTextField();  
vacancyNumberField1.setBounds(610, 60, 100, 30);  
frame.add(vacancyNumberField1);
```

```
joining_Date = new JLabel("Joining Date");  
joining_Date.setBounds(500, 180, 100, 30);  
frame.add(joining_Date);
```

```
joining_DateField = new JTextField();  
joining_DateField.setBounds(610, 180, 100, 30);  
frame.add(joining_DateField);
```

```
qualification = new JLabel("Qualification: ");  
qualification.setBounds(500, 240, 100, 30);  
frame.add(qualification);
```

```
qualificationField = new JTextField();  
qualificationField.setBounds(610, 240, 100, 30);  
frame.add(qualificationField);
```

```
appointed_By = new JLabel("Appointed By: ");  
appointed_By.setBounds(500, 300, 100, 30);  
frame.add(appointed_By);
```

```
appointed_ByField = new JTextField();  
appointed_ByField.setBounds(610, 300, 100, 30);
```

```
    frame.add(appointed_ByField);

}

/*GUI of Buttons*/

public void buttons() {

    btnPartTime = new JButton("Part Time Staff");
    btnPartTime.setBounds(20, 480, 200, 50);
    btnPartTime.addActionListener(this);
    frame.add(btnPartTime);

    btnAppoint = new JButton("Appoint");
    btnAppoint.setBounds(620, 480, 200, 50);
    btnAppoint.addActionListener(this);
    frame.add(btnAppoint);

    btnClear = new JButton("Clear");
    btnClear.setBounds(150, 600, 140, 45);
    btnClear.addActionListener(this);
    frame.add(btnClear);

    btnDisplay = new JButton("Display");
    btnDisplay.setBounds(350, 600, 160, 45);
    btnDisplay.addActionListener(this);
    frame.add(btnDisplay);

    btnTerminate = new JButton("Terminate");
    btnTerminate.setBounds(550, 600, 160, 45);
    btnTerminate.addActionListener(this);
    frame.add(btnTerminate);
```

```
}

/* get method of Name */
public String getName(){
    return nameField.getText();
}

/* get method of Vacancy Number*/
public int getVacancyNumber1(){
    int vacancyNumberInt=0;
    try{
        vacancyNumberInt = Integer.parseInt(vacancyNumberField.getText());

    }
    catch(NumberFormatException e){
        JOptionPane.showMessageDialog(frame,"Please give a valid interger in Vacancy
Number","Error",
        JOptionPane.ERROR_MESSAGE);

    }
    return vacancyNumberInt;
}

/*get method of designation*/
public String getDesignation(){
    return designationField.getText();
}

/*get method of Job Type*/
public String getJob_Type(){
    return job_TypeField.getText();
}

/*get method of wage*/
public int getWage(){
```

```
    int wageInt=0;

    try{
        wageInt = Integer.parseInt(wageField.getText());
    }
    catch(NumberFormatException e){
        JOptionPane.showMessageDialog(frame,"Please give a valid interger in Wage","Error",
            JOptionPane.ERROR_MESSAGE);
    }
    return wageInt;
}

/*get method of working hour*/
public int getWorking_Hour(){
    int working_HourInt=0;
    try{
        working_HourInt = Integer.parseInt(working_HourField.getText());

    }
    catch(NumberFormatException e){
        JOptionPane.showMessageDialog(frame,"Please give a valid interger in Working
Hour","Error",
            JOptionPane.ERROR_MESSAGE);
    }
    return working_HourInt;
}

/*get method of shift*/
public String getShift(){
    return shiftField.getText();
}

/*get method of Staff Name*/
public String getStaff_Name1(){
```

```
        return nameField1.getText();
    }

    /*get method of Vacancy Number 1*/
    public int getVacancyNumberlbl1(){
        int vacancyNumberlblInt=0;
        try{
            vacancyNumberlblInt = Integer.parseInt(vacancyNumberField1.getText());

        }
        catch(NumberFormatException e){
            JOptionPane.showMessageDialog(frame,"Please give a valid interger in Vacancy
Number","Error",
            JOptionPane.ERROR_MESSAGE);

        }
        return vacancyNumberlblInt;
    }

    /*get method of Joining Date*/
    public String getJoining_Date(){
        return joining_DateField.getText();
    }

    /*get method of Qualification*/
    public String getQualification(){
        return qualificationField.getText();
    }

    /*get method of Appointed By*/
    public String getAppointed_By(){
        return appointed_ByField.getText();
    }
```

```
/* Fucntion of addPartTime Button*/

public void addPartTime(){
    int vno = getVacancyNumber1();
    int wage = getWage();
    int working_Hour = getWorking_Hour();
    String designation = getDesignation();
    String job_Type = getJob_Type();
    String shift = getShift();
    boolean vNum1 = false;

    if (vno > 0 && working_Hour > 0 && wage > 0 && !designation.equals("") &&
!job_Type.equals("")
        && !shift.equals("")) {
        for (StaffHire duplicate : INGNepal.list) {
            if (duplicate.getVacancyNumber() == vno) {
                vNum1 = true;
                JOptionPane.showMessageDialog(frame,
                    "Enter anither Vacancy number.It is already added.", "Error",
                    JOptionPane.ERROR_MESSAGE);
                break;
            }
        }
        if (!vNum1) {
            INGNepal.list.add(new PartTimeStaffHire(designation, job_Type, vno, wage,
working_Hour, shift));
            JOptionPane.showMessageDialog(frame,
                "Thank You " + getName() + " for the detail (" + INGNepal.list.size() + ")",
                "Thank You", JOptionPane.INFORMATION_MESSAGE);
        }
    }
}
```

```
/* Fucntion of appointPartTime Button*/
```

```
public void appointPartTime(){  
    int vno=Integer.parseInt(vacancyNumberField1.getText());  
    String staff_name=getStaff_Name1();  
    String joining_date=getJoining_Date();  
    String Qualification=getQualification();  
    String appointed_by=getAppointed_By();  
    boolean vNum=false;  
    if(vno>0){  
        for(StaffHire obj:INGNepal.list){  
            if(obj.getVacancyNumber() == vno){  
                vNum=true;  
                if(obj instanceof PartTimeStaffHire){  
                    PartTimeStaffHire p=(PartTimeStaffHire)obj;  
                    if(!p.getJoined()){  
                        p.hirePartTimeStaff(staff_name,joining_date,Qualification,appointed_by);  
                        JOptionPane.showMessageDialog(frame," Staff has been hired");  
                    }else{  
  
                        JOptionPane.showMessageDialog(frame,"Staff has been already hired!");  
                        break;  
                    }  
                }else{  
                    JOptionPane.showMessageDialog(frame,"It is not for fulltime staff Hire");  
                    break;  
                }  
            }  
        }  
    }  
}
```



```
        if(!vNum){
            JOptionPane.showMessageDialog(frame,"invalid vacancy","Error",
            JOptionPane.ERROR_MESSAGE);
        }
    }
}

/* Fucntion of clearPartTime Button*/
public void clearPartTime(){
    nameField.setText(null);
    nameField1.setText(null);
    vacancyNumberField.setText(null);
    vacancyNumberField1.setText(null);
    designationField.setText(null);
    job_TypeField.setText(null);
    wageField.setText(null);
    working_HourField.setText(null);
    shiftField.setText(null);
    joining_DateField.setText(null);
    qualificationField.setText(null);
    appointed_ByField.setText(null);
}

/* Fucntion of displayPartTime Button*/
public void displayPartTime(){
    if(INGNepal.list.isEmpty()){
        JOptionPane.showMessageDialog(frame, "Fill the field which are required ", "Error",
        JOptionPane.ERROR_MESSAGE);
    }
}
```

```
    }else{
        for (StaffHire dis: INGNepal.list){
            ((PartTimeStaffHire) dis).display();
        }
    }
}

/* Fucntion of terminatePartTime Buttion*/
public void terminatePartTime() {
    boolean vNo = false;
    int vacancy = getVacancyNumber1();
    if (vacancy > 0) {
        for (StaffHire obj : INGNepal.list) {
            if (obj.getVacancyNumber() == vacancy) {
                vNo = true;
                if (obj instanceof PartTimeStaffHire) {
                    PartTimeStaffHire p = (PartTimeStaffHire) obj;
                    if (!p.getTerminated()) {
                        p.staffTerminated();
                        JOptionPane.showMessageDialog(frame, "Staff has been terminated", "Done!",
                            JOptionPane.INFORMATION_MESSAGE);
                    } else {
                        JOptionPane.showMessageDialog(frame,
                            "Enter a valid vacancy number. Staff is already terminated",
                            "Error", JOptionPane.ERROR_MESSAGE);
                    }
                } else {
                    JOptionPane.showMessageDialog(frame, "Invalid vacancy number. Not valid for
termination",
                        "Invalid", JOptionPane.ERROR_MESSAGE);
                }
            }
        }
    }
}
```

```
        }
    }
}
if (!vNo) {
    JOptionPane.showMessageDialog(frame, "Staff is not hired. Enter another vacancy
number.",
        "Invalid", JOptionPane.ERROR_MESSAGE);
}
}
}
/* main method*/
public static void main (String[] args){
    new PartTimeStaffHireGUI();
}
/* abstract Method */
@Override
public void actionPerformed(ActionEvent e){
    if (e.getSource() == btnPartTime){
        addPartTime();
    }else if(e.getSource() == btnAppoint){
        appointPartTime();
    }else if(e.getSource() == btnDisplay){
        displayPartTime();
    }else if(e.getSource() == btnClear){
        clearPartTime();
    }else if(e.getSource() == btnTerminate){
        terminatePartTime();
    }
}
}
```