LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CS4001 Programming

Assessment Weightage & Type

30% Individual Coursework

Year and Semester

2019-20 Autumn

Student Name: Ramkrishna Yadav

Group: L1C1

London Met ID:

College ID: NP01CP4A190268

Assignment Due Date: 2020 March 10

Assignment Submission Date: 2020 April 17

Table of Contents

List of Figures

List of Tables

1.Introduction

As the module “Programming”, first course work carries 30% from total marks in the first semester. In the same way, the second course work also holds 30% which was given to all students individually in the second semester. And the main purpose of this course work is to score a good mark and to re-call the knowledge over the both semesters. The course work is about to add a class to the project which had been developed in the first part of the course work to make a graphical user interface(GUI). And the main work of this system is to hire staffs and to store the details of the vacancy and staff in the list.

The application which has been developed in this coursework is used by the organization to hire the staff as per their need. Even they can store their staff information easily in the list which can be seen as the most important part inside the firm and organization. The information of staff like name, designation, jobType, salary, working hour and many more information can be stored in this application.

Simply, java is general programming language that is object-oriented and concurrent in which java code are compiled and run on all platforms. Specially, it is used to develop the application (HowToDoInjava.com, 2016).

Similarly, Graphic user interface can be seen as the computer program in which the symbols like button, frame, text area, menu bar, navigation etc. are used instead of textual words. It enhance a people to communicate with the computer in the form of symbols, visual metaphors (Levy, 2020). Java swing is also used while making this application and without it, this application can’t be developed successfully.

And java Swing is used to develop the windows-based application(or software). And the package javax.swing allows different functions for the classes in java swing API. Java Swing is the part of java Foundation classes that provides various function like JTextField, JTextArea, JRadioButton, JCheckbox, JButton, JColorchooser etc (Java Point, 2018).

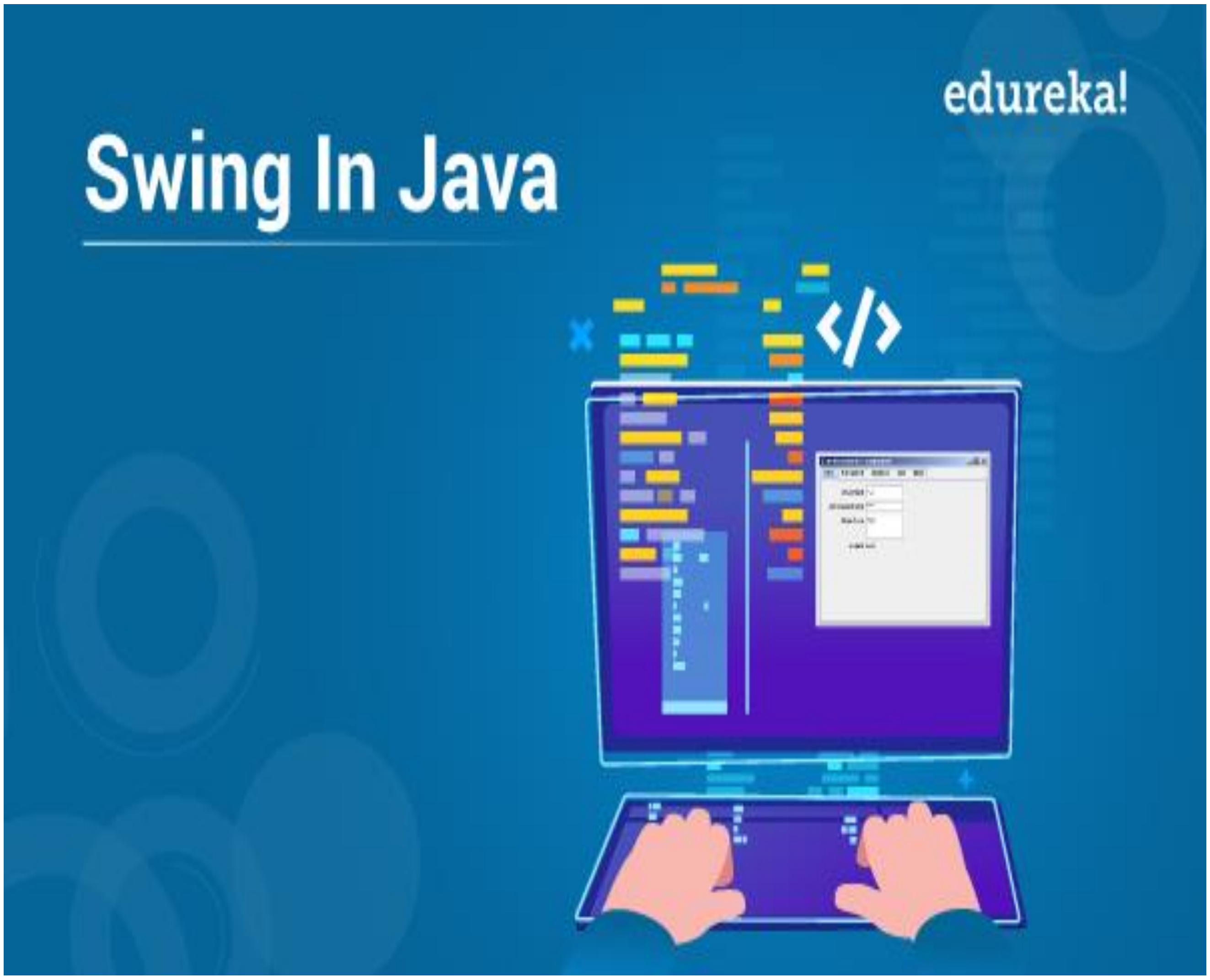


Figure 1: Java Swing

While coding to develop the given application, lots of research and study are done to handle the errors and bugs inside this program. Furthermore, the whole coding is written inside the software called “Blue J”. To complete the given task lots of tools are used such as language of program, compiler and source code.

After the completion of the program, this type of format form inside the “Blue J” windows. And the format is shown below in the form of screenshot;

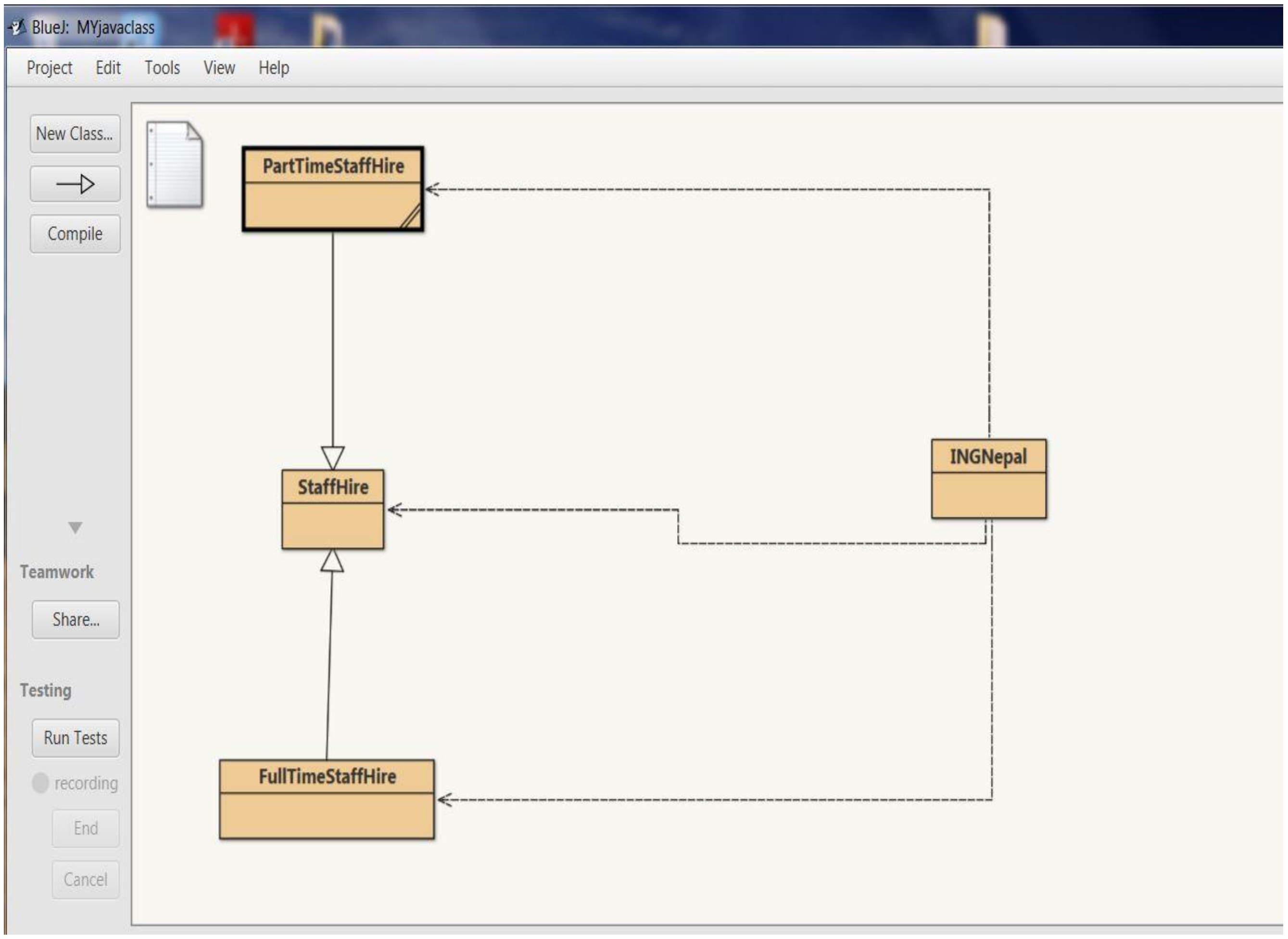


Figure 2:Three class Added to INGNepal

2.Class Diagram:

The class Diagram is the visual(or graphical or tabular) representation which describes the structure of a by showing the system of the classes. Even it also enchance to show the attributes, methods inside the classes and shows the relationships among objects. Simply, class diagrams formed with classes, attributes and methods. And, the main purpose of making class diagram to allocate the static architecture of the classes in a system. Even it makes more easier to the developer and member inside the team too (Visual Paradigm, 2020).

The below Class diagram contains three tables in which the first table(or box) have the class name "INGNepal" and the second table(or box) contains the attributes inside the classes. Similarly, And, third box contains the attributes which have a corresponding accessor method that is getter and setter method. As the "-" sign indicate the attributes which are private and "+" sign indicate the method which are public as shown in the below diagram.

INGNepal
-frame: JFrame
-frameFull: JFrame
-framePart: JFrame
-title: JLabel
-VacancyNumber: JLabel
-Designation: JLabel
-JobType: JLabel
-Salary: JLabel
-WorkingHour: JLabel
-ApplicantName: JLabel
-StaffName: JLabel
-Qualification: JLabel
-JoiningDate: JLabel
-AppointedBy: JLabel
-WorkingShift: JLabel
-WagesPerHour: JLabel
-vacancyNumber: JTextField
-designation: JTextField
-salary: JTextField
-applicantName: JTextField
-qualification: JTextField
-appointedBy: JTextField
-wagesPerHour: JTextField
-vacancyNumber1: JTextField

-vacancyNumberPT: JTextField
-staffNamePT: JTextField
-qualificationPT: JTextField
-appointedByPT: JTextField
-vacancyNumberFT: JTextField
-staffNameFT: JTextField
-qualificationFT: JTextField
-appointedByPT: JTextField
-workingHour: JComboBox
-joiningDateYear: JComboBox
-joiningDateDay: JComboBox
-joiningDateMonth: JComboBox
-joiningDateYear: JComboBox
-joiningDateDay: JComboBox
-joiningDateMonth: JComboBox
-joiningDateYear: JComboBox
-joiningDateDay: JComboBox
-joiningDateMonth: JComboBox
-FullTimeStaffHire: JRadioButton
-PartTimeStaffHire: JRadioButton
-morning: JRadioButton
-day: JRadioButton
-evening: JRadioButton
-AddPartTimeStaffHire: JButton
-AddFullTimeStaffHire: JButton
-AppointFullTimeStaffHire: JButton
-AppointPartTimeStaffHire: JButton
-Clear: JButton
-Terminate: JButton
-Display: JButton
-Exit: JButton

-ExitPart: JButton

-ExitFull: JButton

-Appoint: JButton

-OK: JButton

-p1: JPanel

-workingShift: ButtonGroup

-jobType: ButtonGroup

-flag: boolean

-vacNumberCheck: boolean

-list: ArrayList

+main(String[] args) : Void

+getCallVacancyNum(): int

+getCallDesignation(): String

+getCallSalary(): int

+getCallApplicantName(): String

+getCallStaffName(): String

+getCallQualification(): String

+getCallAppointedBy(): String

+getCallWagesPerHour(): int

+getCallWorkingHour(): int

+getCallJoiningDate(): String

+getVacNum(): int

+getCallVacancyNumber(): int

+getCallStaffName(): String

+getCallQualification(): String

+getCallAppointedBy(): String

+getCallJoiningDate(): String

+getCallVacancyNumber(): int

+getCallStaffName(): String

+getCallQualification(): String

+getCallAppointedBy(): String
+getCallJoiningDate(): String
+actionPerformed(ActionEvent e): Void
+checkAppoint(): void
+addPartTimeStaffHire(): void
+addFullTimeStaffHire(): void
+appointFullTimeStaff(): void
+appointPartTimeStaff(): void
+terminatePartTimeStaff(): void
+display(): void
+appointPartTimeStaffHire(): void
+appointFullTimeStaffHire(): void
+clear(): void

Table 1: Class Diagram

2.1Relational Diagram:

Simply, relational diagram is the graphical representation of the classes and their relationship to eachother. As in the below figure, the classes are related with each other. Firstly, PartTimeStaffHire class and FullTimeStaffHire classes are related with StaffHire class. And these three classes are related with the INGNepal class.

While coding, PartTimeStaffHire and FullTimeStaffHire are extends to StaffHire class. And, these all classes are added to INGNepal.

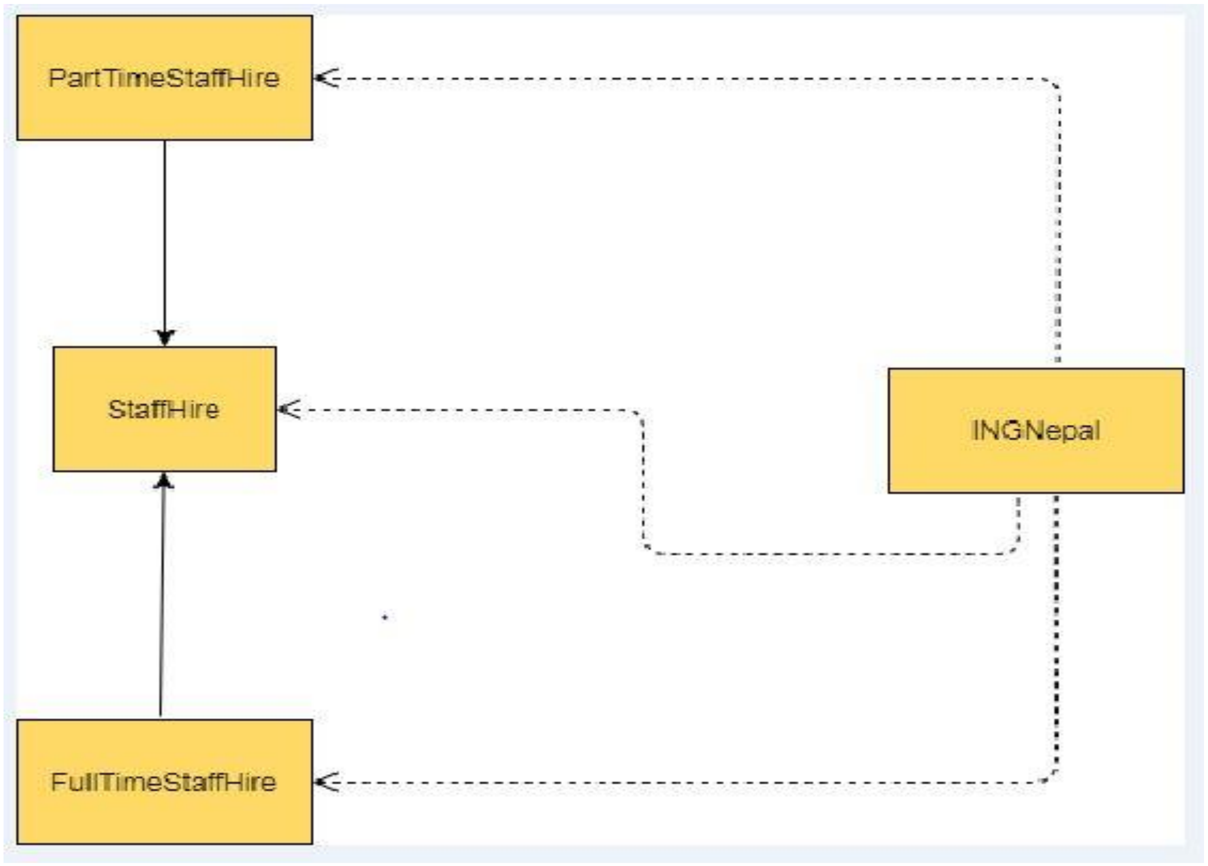


Figure 3: Relational Diagram

3.Pseudocode:

Pseudocode is simply known as false code. Pseudocode was widely used by the programmer to make code in readable format. And it is the simple version of programming language in which the codes are formally styled in simple(or natural) language, fairly than in a programming language. Pseudocode was mostly used by the program developers as a detail step in the case of designing the applications. Sometimes, pseudocode provides specific arrangement to the developers to write the codes in a next step while working on the specific program (Rouse, 2005).

And mainly pseudocode helps developer to make the codes in readable format so that others can understand the code easily. And pseudocode was widely used in algorithm and programming fields. Furthermore, pseudocode helps to represent the implementation of the program.

The pseudocode for the program is given below:

3.1INGNepal

```
DEFINE class INGNepal which implements ActionListener

DEFINE instance variables of JFrame as frame, frameFull, framePart, frameAppoint.

DEFINE instance variables of JLabel as title, VacancyNumber, Designation, JobType, Salary, WorkingHour, ApplicantName, StaffName, Qualification, JoiningDate, AppointedBy, WorkingShift, WagesPerHour.

DEFINE instance variables of JTextField as vacancyNumber, designation, salary,applicantName, staffName, qualification, appointedBy, wagesPerHour, vacancyNumber1

    DEFINE instance variables as JComboBox as workingHour, joiningDateYear, joiningDateMonth, joiningDateDay

    DEFINE instance variables of JRadioButton as morning, day, fullTimeStaffHire, partTimeStaffHire, evening

    DEFINE instance variables of JButton as AddPartTimeStaffHire, AddFullTimeStaffHire, AppointFullTimeStaffHire, AppointPartTimeStaffHire, Clear, Terminate, Display, Exit, ExitPart, ExitFull, Appoint, OK

    DEFINE instance variables of JPanel as p1
    DEFINE instance variables of ButtonGroup as workingShift, jobType
    DEFINE instance variables of boolean as flag, vNoCheck
    DEFINE instance variables of ArrayList as list
    Create a constructor and initialize the variables
```

```
DEFINE INGNepal

    DO
        INITIALIZE frame
        SET positions of frame in width, height as 700,350
        INITIALIZE panel p1
        SET layout null to p1
        SET border to p1 as color black
        SET border of frame in x,y, width, height as 10,10,656,300
        SET set visible to true in p1

    ADD p1 to frame

        INITIALIZE color coding for button

        INITIALIZE title, VacancyNumber, Designation, JobType, WorkingHour,
        ApplicantName, Qualification, Salary, JoiningDate, AppointedBy, WorkingShift,
        WagesPerHour as JLabel

        INITIALIZE vacancyNumber, designation salary, applicantName, qualification,
        appointedBy, wagesPerHour as JTextField

        INITIALIZE workingHour, joiningDateYear, joiningDateDay, joiningDateMonth
        as JComboBox

        INITIALIZE fullTimeStaffHire, partTimeStaffHire, morning, day, evening as
        JRadioButton

        INITIALIZE AddPartTimeStaffHire, AddFullTimeStaffHire, Display, Exit,
        Appoint as JButton
```

SET title name of frame to “Job Application Form”

SET the Boundary of all the components of the GUI form

SET positions of title in x, y, width, height as 250,5,300,30 respectively as JLabel

SET positions of VacancyNumber in x, y, width, height as 20,40,150,20 respectively as JLabel

SET positions of Designation in x, y, width, height as 350,40,130,20 respectively as JLabel

SET positions of JobType in x, y, width, height as 20,70,80,20 respectively as JLabel

SET positions of WorkingHour in x, y, width, height as 20,100,130,20 respectively as JLabel

SET positions of Salary in x,y, width, height as 350,100,70,20 respectively as JLabel

SET positions of Qualification in x, y, width, height as 20,130,100,20 respectively as JLabel

SET positions of JoiningDate in x, y, width, height as 350,130,130,20 respectively as JLabel

SET positions of WagesPerHour in x,y, width, height as 20,160,140,20 respectively as JLabel

SET positions of WorkingShift in x, y, width, height as 350,160,160,20 respectively as JLabel

SET positions of AppointedBy in x,y, width, height as 20,190,130,20 respectively as JLabel

SET positions of vacancyNumber in x, y, width, height as 150,40,170,20 respectively as JTextField

SET positions of designation in x, y, width, height as 450,40,195,20 respectively as JTextField

SET positions of partTimeStaffHire in x, y, width, height as 90,70,115,20 respectively as JRadioButton

SET positions of fullTimeStaffHire in x,y, width, height as 205,70,115,20 respectively as JRadioButton

SET positions of workingHour in x, y, width, height as 130,100,80,20 respectively as JComboBox

SET positions of joiningDateYear in x, y, width, height as 450,130,65,20 respectively as JComboBox

SET positions of joiningDateMonth in x,y, width, height as 520,130,70,20 respectively as JComboBox

SET positions of joiningDateDay in x, y, width, height as 595,130,50,20 respectively as JComboBox

SET positions of wagesPerHour in x,y, width, height as 145,160,175,20 respectively as JTextField

SET positions of morning in x,y, width, height as 451,160,75,20 respectively as JRadioButton

SET positions of day in x,y, width, height as 528,160,50,20 respectively as JRadioButton

SET positions of evening in x,y, width, height as 578,160,75,20 respectively as JRadioButton

SET positions of appointedBy in x,y, width, height as 130,190,190,20 respectively as JTextField

SET positions of AddPartTimeStaffHire in x,y, width, height as 90,230,150,50 respectively as JButton

SET positions of AddFullTimeStaffHire in x,y, width, height as 260,230,150,50 respectively as JButton

SET positions of Display in x,y, width, height as 430,230,150,50 respectively as JButton

SET positions of Exit in x,y, width, height as 20,260,60,20 respectively as JButton

ADD all the components of JLabel, JTextFeild, JComboBox, JRadioButton, JButton to panel p1

ADD ActionListener to all button

```
            SET font as Arial and Bold
            SET defaultCloseOperation to exit on close
            SET LocationRelative to null
            SET setvisible to true

        END DO

 DEFINE method checkAppoint

        DO
          INITIALIZE frameAppoint
          SET positions of frameAppoint in width, height as 700,330
          INITIALIZE panel p1
          SET layout null to p1
          SET border to p1 as color black
          SET border of frameAppoint in x,y, width, height as 10,10,656,275
          SET set visible to true in p1
          ADD p1 to frameAppoint

          INITIALIZE title, VacancyNumber as JLabel
          INITIALIZE VacancyNumber1 as JTextField
          INITIALIZE OK as JButton

              SET title name of frameAppoint to "Vacancy Number Check"

              SET the Boundary of all the components of the GUI form

              SET positions of title in x, y, width, height as 10,10,656,275 respectively as
         JLabel
              SET positions of VacancyNumber in x, y, width, height as 20,40,150,20
        respectively as JLabel
```

```
            ADD all the components of JLabel, JTextFeild, JButton to panel p1
            ADD ActionListener to button OK
            SET font as Arial and Bold
            SET defaultCloseOperation to exit on close
            SET LocationRelative to null
            SET Resizable to false
            SET setvisible to true

    END DO

DEFINE method appointPartTimeStaffHire

    DO

        INITIALIZE framePart
        SET positions of framePart in width, height as 700,330
        INITIALIZE panel p1
        SET layout null to p1
        SET border to p1 as color black
        SET border of framePart in x,y, width, height as 10,10,656,275
        SET set visible to true in p1
        ADD p1 to framePart

        INITIALIZE color coding for button

        INITIALIZE title, VacancyNumber, StaffName, JoiningDate, Qualification,
AppointedBy as JLabel
```

INITIALIZE vacancyNumberPT, staffName, qualification, appointedBy as JTextField

INITIALIZE joiningDateYear, joiningDateDay, joiningDateMonth as JComboBox

INITIALIZE Terminate, AppointPartTimeStaffHire, Display, Clear, ExitPart as JButton

SET title name of framePart to “Appoint/Terminate Part Time Staff”

SET the Boundary of all the components of the GUI form

SET positions of title in x, y, width, height as 250,5,300,30 respectively as JLabel

SET positions of VacancyNumber in x, y, width, height as 20,40,150,20 respectively as JLabel

SET positions of StaffName in x, y, width, height as 350,40,130,20 respectively as JLabel

SET positions of Qualification in x,y, width, height as respectively 20,70,100,20 as JLabel

SET positions of JoiningDate in x, y, width, height as 350,70,130,20 respectively as JLabel

SET positions of AppointedBy in x, y, width, height as 20,100,130,20 respectively as JLabel

SET positions of vacancyNumber in x, y, width, height as 150,40,130,20 respectively as JTextField

SET positions of qualification in x,y, width, height as 130,70,150,20 respectively as JTextField

SET positions of staffName in x, y, width, height as 450,40,195,20 respectively as JTextField

SET positions of joiningDateYear in x, y, width, height as 450,70,65,20 respectively as JComboBox

SET positions of joiningDateDay in x, y, width, height as 595,70,50,20 respectively as JComboBox

SET positions of joiningDateMonthPT in x, y, width, height as 520,70,70,20 respectively as JComboBox

SET positions of appointedByPT in x, y, width, height as 130,100,150,20 respectively as JTextField

SET positions of Terminate in x,y, width, height as 445,130,200,50 respectively as JButton

SET positions of AppointPartTimeStaffHire in x,y, width, height as 230,130,200,50 respectively as JButton

SET positions of Display in x,y, width, height as 20,190,210,50 respectively as JButton

SET positions of Clear in x,y, width, height as 445,190,200,50 respectively as JButton

```
            SET positions of ExitPart in x,y, width, height as 585,247,60,20 respectively
        as JButton

            ADD all the components of JLabel, JTextFeild, JComboBox, JButton to
        panel p1

            ADD ActionListener to all button
            SET font as Arial and Bold
            SET defaultCloseOperation to exit on close
            SET LocationRelative to null
            SET setvisible to true

        END DO

  DEFINE method appointFullTimeStaffHire

     DO
          INITIALIZE frameFull
          SET positions of frame in width, height as 700,330
          INITIALIZE panel p1
          SET layout null to p1
          SET border to p1 as color black
          SET border of frameFull in x,y, width, height as 10,10,656,275
          SET set visible to true in p1
          ADD p1 to frameFull

          INITIALIZE color coding for button

          INITIALIZE title, VacancyNumber, StaffName, JoiningDate, Qualification,
  AppointedBy as JLabel
```

```
INITIALIZE vacancyNumber, staffName, qualification, appointedBy as JTextField

INITIALIZE joiningDateYear, joiningDateDay, joiningDateMonth as JComboBox

INITIALIZE AppointFullTimeStaffHire, Display, Clear, ExitFull as JButton

SET title name of frameFull to “Appoint Full Time Staff”
SET the Boundary of all the components of the GUI form
SET positions of title in x, y, width, height as 250,5,300,30 respectively as JLabel
    SET positions of VacancyNumber in x, y, width, height as 20,40,150,20 respectively as JLabel

    SET positions of Qualification in x,y, width, height as 20,70,100,20 respectively as JLabel

    SET positions of StaffName in x, y, width, height as 350,40,130,20 respectively as JLabel

    SET positions of JoiningDate in x, y, width, height as 350,70,130,20 respectively as JLabel

    SET positions of AppointedBy in x, y, width, height as 20,100,130,20 respectively as JLabel

    SET positions of vacancyNumber in x, y, width, height as 150,40,130,20 respectively as JTextField

    SET positions of qualification in x,y, width, height as 130,70,150,20 respectively as JTextField
```

SET positions of staffName in x, y, width, height as 450,40,195,20 respectively as JTextField

SET positions of joiningDateYear in x, y, width, height as 450,70,65,20 respectively as JComboBox

SET positions of joiningDateDay in x, y, width, height as 595,70,50,20 respectively as JComboBox

SET positions of joiningDateMonth in x, y, width, height as 520,70,70,20 respectively as JComboBox

SET positions of appointedBy in x, y, width, height as 130,100,150,20 respectively as JTextField

SET positions of AppointFullTimeStaffHire in x,y, width, height as 445,130,200,50 respectively as JButton

SET positions of Display in x,y, width, height as 20,130,220,50 respectively as JButton

SET positions of Clear in x,y, width, height as 20,200,220,50 respectively as JButton

SET positions of Exit in x,y, width, height as 585,250,60,20 respectively as JButton

ADD all the components of JLabel, JTextFeild, JComboBox, JButton to panel p1

ADD ActionListener to all button

```
            SET font as Arial and Bold
            SET defaultCloseOperation to exit on close
            SET LocationRelative to null
            SET setvisible to true

    END DO

DEFINE main (String[] args) as void static type

    DO
        Call INGNepal

    END DO

DEFINE method getCallVacancyNum () as int type

     DO
          CONVERT conVac TO integer
          Return conVac

       END DO

DEFINE method getCallDesignation() as String type

    DO
        STORE input FROM des
        Return des

    END DO

DEFINE method getCallSalary () as int type
```

```
    DO
        CONVERT conSalary TO integer
        Return conSalary
    END DO

DEFINE method getCallApplicantName () as String type

    DO
        STORE input FROM appName
        Return appName
    END DO

DEFINE method getCallQualification () as String type

    DO
        STORE input FROM qualified
        Return qualified

    END DO

DEFINE method getCallAppointedBy () as String type

    DO
        STORE input FROM appBy
        Return appBy
    END DO

DEFINE method getCallWagesPerHour () as int type

    DO
```

```
            CONVERT wPHour TO integer
            Return wPHour
        END DO

DEFINE method getCallWorkingHour () as int type

       DO
            CONVERT wHour TO integer
            Return wHour
        END DO

DEFINE method getCallJoiningDate () as String type

      DO
           STORE input FROM JoiningDate
           Return JoiningDate

      END DO

DEFINE method getVacNum () as int type

       DO
            CONVERT vacNum TO integer
            Return vacNum
        END DO

DEFINE method getCallVacancyNumber () as int type

       DO
            CONVERT getVacancyNumber TO integer
            Return getVacancyNumber
        END DO

DEFINE method getCallStaffName () as String type

        DO
           STORE input FROM Name
           Return Name

         END DO
DEFINE method getCallQualification () as String type

       DO
           STORE input FROM qualified
           Return qualified
        END DO

DEFINE method getCallAppointedBy () as String type

      DO
           STORE input FROM appBy
           Return appBy
      END DO

DEFINE method getCallJoiningDate () as String type

        DO
           STORE input FROM JoiningDate
           Return JoiningDate
        END DO

DEFINE method getCallVacancyNumber () as int type
```

```
    DO
        CONVERT getVacancyNumber TO integer
        Return getVacancyNumber
    END DO

DEFINE method getCallStaffName () as String type

    DO
        STORE input FROM sName
        Return sName
    END DO

DEFINE method getCallQualification () as String type

    DO
        STORE input FROM qualified
        Return qualified
    END DO

DEFINE method getCallAppointedBy () as String type

    DO
        STORE input FROM appBy
        Return appBy

    END DO

DEFINE method getCallJoiningDate () as String type

    DO
```

```
            STORE input FROM JoiningDate
            Return JoiningDate
          END DO

DEFINE method actionPerformed(accepts ActionEvent e as String type) as void type

          DO
   IF (e gets Exit)
          DO
Call frame.dispose() method

        END DO

        END IF

               ELSE IF (e gets ExitFull)

       DO

               Call frameFull.dispose() method

        END DO

        END ELSE IF

               ELSE IF (e gets ExitPart)
        DO

               Call framePart.dispose() method

        END DO
        END ELSE IF

               ELSE IF (e.getActionCommand is equal to ("Full Time")

          DO

               SET required text fields and labels only to VISIBLE for full time

        END DO

               END ELSE IF

               ELSE IF (e.getActionCommand is equal to ("Part Time")
          DO

               SET required text fields and labels only to VISIBLE for part time

        END DO

END ELSE IF

ELSE IF (e.getActionCommand is equal to ("Add Part Time Staff")

        DO

                Call addPartTimeStaffHire () method

          IF (flag is true and vNoCheck is false)

            DO
```

```
                Show confirmation dialog box whether to appoint staff or not
            END DO
            IF (yes is click)
                DO
                    Call appointPartTimeStaffHire () method
                  END DO
 END ELSE IF
 ELSE IF (e.getActionCommand is equal to ("Add Full Time Staff")
        DO
            Call addFullTimeStaffHire () method
            IF (flag is true and vNoCheck is false)
         END DO
            DO
                Show confirmation dialog box whether to appoint staff or not
            END DO
 IF (yes is click)
```

```
                DO
               Call appointFullTimeStaffHire () method
               END DO

ELSE IF (e.getActionCommand is equal to ("Appoint Full Time Staff")
      DO
            Call appointFullTimeStaff () method
       END DO
END ELSE IF
ELSE IF (e.getActionCommand is equal to ("Appoint Part Time Staff")
       DO
Call appointPartTimeStaff () method
       END DO
END ELSE IF
ELSE IF (e.getActionCommand is equal to ("Terminate Part Time Staff")
      DO
Call terminatePartTimeStaff () method
```

```
        END DO

    END ELSE IF

    ELSE IF (e.getActionCommand is equal to ("Display Part Time Staff Record ")

        DO
    Call display () method
        END DO
    END ELSE IF

    ELSE IF (e.getActionCommand is equal to ("Display Full Time Staff Record ")

        DO
    Call display () method
          END DO

    END ELSE IF
    ELSE IF (e.getActionCommand is equal to ("Display ")

        DO
    Call display () method
          END DO

    END ELSE IF
    ELSE IF (e.getActionCommand is equal to ("Clear Staff Information ")

        DO
    Call clear () method
          END DO
```

```
END ELSE IF
ELSE IF (e.getActionCommand is equal to ("Appoint ")

      DO
Call checkAppoint () method
       END DO

END ELSE IF
ELSE IF (e.getActionCommand is equal to ("OK ")
      DO

      START A TRY
         creating addedVacancy = false
FOR (staffHire staff:list)
            IF (staff.getVacancyNumber() is equals to getVacNum)
               addedVacancy=true
               IF (staff instanceof fullTimeStaffHire class)
Call appointFullTimeStaffHire() method
Break

END IF
ELSE IF (staff instanceof partTimeStaffHire class)
Call appointPartTimeStaffHire() method
Break
END ELSE IF
END IF

END FOR
IF(addedVacancy is equals to false)
Error message about vacancy number
END IF
END TRY

CATCH (NumberFormatException e3)
Shows a suitable message
END CATCH

END ELSE IF
      END DO

DEFINE method addPartTimeStaffHire() as void type

       DO

START OF TRY

   vNoCheck is false
   FOR (staffHire staff:list)
      IF (getCallVacancyNum() is equals to staff.getVacancyNumber())
         vNoCheck is true
      END IF

      IF (vNoCheck is equals to false)
         IF (check all the fields are empty or not)
            Shows appropriate message if fields are empty
         END IF

         ELSE
Type casting all the required variables
            IF (working shift is not selected)
Message pop up as "Please select the working shift!!"
```

```
            END IF

            ELSE IF (working hour and wages per hour is less than 0)
Message pop up as wages and hour cannot be negative
            END ELSE IF

            ELSE IF (radio button of job type is not valid for part time)
Appropriate message that job type you selected is invalid
            END ELSE IF
            ELSE
Creating object addPartTimeStaff
Adding the object to the arraylist list
Shows pop up as “Part time staff hired”
flag is true
            END ELSE
         END ELSE

      END IF
      ELSE
Message dialog box appears with “Vacancy Number already exists!!”
      END ELSE
END TRY

CATCH (NumberFormatException e1)
   Shows suitable error
   flag is false
END CATCH

CATCH (ArithmeticException e2)
   Shows arithmetic error
   flag is false
```

```
END CATCH

        END DO

DEFINE method addFullTimeStaffHire() as void type

        DO
START OF TRY
   vNoCheck is false
   FOR (staffHire staff:list)
      IF (getCallVacancyNum() is equals to staff.getVacancyNumber())
         vNoCheck is true
      END IF

      IF (vNoCheck is equals to false)
         IF (check all the fields are empty or not)
            Shows appropriate message if fields are empty
         END IF

         ELSE
Type casting all the required variables
            IF (radio button of job type is not valid for part time)
Appropriate message that job type you selected is invalid
            END IF

            ELSE
Creating object addFullTimeStaff
Adding the object to the arraylist list
Shows pop up as “Full time staff hired”
flag is true
            END ELSE
         END ELSE

      END IF
      ELSE
Message dialog box appears with "Vacancy Number already exists!!"
      END ELSE
END TRY

CATCH (Exception e1)
   Shows suitable error
   flag is false
END CATCH
END DO



DEFINE method appointFullTimeStaff() as void type

   DO
      ELSE IF (staff name isnot matched with string values)
         Message display as staff name requires only text
      END ELSE IF

ELSE IF (qualification isnot matched with string values)
         Display message as qualification requires only text
      END ELSE IF

      ELSE
         FOR (staffHire vNo:list)
IF (vNo.getVacancyNumber() is equals to getCallVacancyNum())
   f1=true
END IF
```

```
          END FOR
          IF (f1 is equals to true)
             FOR (staffHire vNo:list)

                IF (vNo instanceof fullTimeStaffHire)

 IF (vNo.getVacancyNumber() is euals to getCallVacancyNumberFT())
    Checking vacancy number whether it is added to full time or not

 IF (addFullTimeStaff.getJoined() is equals to false)
 Call hireFullTimeStaff (sNameFT , JoiningDateFT, qualifiedFT,  appByFT)
 Message display as "Congratulations!!  full time staff hired"
                      Dispose frameFull

                   END IF
                   ELSE

                      Dispose frameFull
 Displays message as "full time staff    already hired"
                   END ELSE
                   END FOR

                END IF
                ELSE
 Shows suitable message as "vacancy number does not belong to full time staff hire"
                END ELSE
             END FOR

          END IF
          ELSE
```

```
            Shows dialog box as "vacancy does not exist in database"
         END ELSE
      END ELSE
      END TRY

      CATCH (NumberFormatException e)
         Display message as "data type for vacancy number is numeric or integer"
      END CATCH

      CATCH (NullPointerException e)
         Shows specific error message information
      END CATCH

END DO

DEFINE method appointPartTimeStaff() as void type

        DO
               START A TRY
      Creating f1 as false
      Type casting all the required variables
      IF (the fields are empty)
         Shows message as "Please enter all the required fields"
        END IF

      ELSE IF (staff name isnot matched with string values)
         Message display as staff name requires only text
      END ELSE IF
      ELSE

         FOR (staffHire vNo:list)
```

```
IF (vNo.getVacancyNumber() is equals to getCallVacancyNum())
   f1=true
END IF
         END FOR
         IF (f1 is equals to true)
            FOR (staffHire vNo:list)
               IF (vNo instanceof partTimeStaffHire)

IF (vNo.getVacancyNumber() is euals to getCallVacancyNumberPT())
   Checking vacancy number whether it is added to part time or not

IF (addPartTimeStaff.getJoined() is equals to false)
Call hirePartTimeStaff (sNamePT, JoiningDatePT, qualifiedPT,appByPT)
Message display as "Congratulations!!  part time staff hired"
Terminated is set to be false
                     Dispose framePart

                  END IF
                  ELSE

                     Dispose framePart
Displays message as "part time staff    already hired"
                  END ELSE
                  END IF

               END IF
               ELSE
Shows suitable message as "vacancy number does not belong to part time staff hire"
               END ELSE
            END FOR
         END IF
         ELSE
            Shows dialog box as “vacancy does not exist in database”
         END ELSE
      END ELSE
      END TRY

      CATCH (NumberFormatException e)
         Display message as “data type for vacancy number is numeric or integer”
      END CATCH

      CATCH (NullPointerException e)
         Shows specific error message information
      END CATCH
END DO



DEFINE method terminatePartTimeStaff() as void type

   DO
START A TRY
      IF (the fields are empty)
         Shows message as “Please enter all the required fields”
      END IF
      ELSE

         FOR (staffHire vNo:list)
IF (vNo.getVacancyNumber() is equals to getCallVacancyNum())
               IF (vNo instanceof partTimeStaffHire)

Checking vacancy number whether it is added to part time or not
```

```
IF (addPartTimeStaff.getTerminated() is equals to true)
Message display as “part time staff has been already terminated”

                  END IF
                  ELSE

IF (addPartTimeStaff.getJoined() is equals to false)
Message display as “part time staff has hot hired yet”

                     END IF

ELSE IF (getterminated is equals to false)
Clear the detail information of terminated staff
Message display as “part time staff has been terminated”

                     END ELSE IF
                     ELSE

Message display as “part time staff has been already terminated”

                     END ELSE
                  END ELSE
               END IF
               ELSE

Shows dialog box as “vacancy does not match to part time staff”

               END ELSE
            END IF
            ELSE
```

```
Shows dialog box as “vacancy does not match to part time staff”

        END ELSE
      END FOR
    END ELSE
    END TRY

    CATCH (NumberFormatException e)
      Display message as “data type for vacancy number is numeric or integer”
    END CATCH

    CATCH (NullPointerException e)
      Shows specific error message information
    END CATCH
END DO

DEFINE method display () as void type

  DO
    IF (list size is equals to 0)
    Displays information box as “no staff are added nothing to display”
  END IF
  ELSE

    FOR (initializing i=0, i is less than list size, increment of i by 1)
        IF (list.get(i) instanceof fullTimeStaffHire))

          CHECKS IF the index is index of Full Time Staff
          Display information of full-time staff
        END IF
```

```
            ELSE (list.get(i) instanceof partTimeStaffHire))

                CHECKS IF the index is index of Part Time Staff
                Display information of part-time staff

            END ELSE
        END FOR

    END ELSE
END DO


DEFINE method clear () as void type

    DO

        SET all the text fields of GUI empty

    END DO
```

4.Method Decription:

Simply, method description is the short information over the method used inside the program. And, while developing the program different program are used for the different classes which are listed below with short description over it.

The method description for INGNepal is given below:

- **getCallVacancyNum():**

This is a getter method method which helps to return the vacancy number.

- **getCallDesignation():**

This a type of getter method. This getter method helps to return the Designation.

- **getCallSalary():**

A type of getter method which return the salary of Salary of the staff.

- **getCallApplicantName():**

  It is a type of getter method in which the applicant name is returned and stored.

- **getCallStaffName():**

  The staff name was returned by using the **getcallStaffName** method.

- **getCallQualification():**

  It is the type of getter method which return the Qualification.

- **getCallAppointedBy():**

  It is a type of getter method. This getter method helps to return the AppointedBy.

- **getCallWagesPerHour():**

  It is type of getter method and it return the WagesPerHour of staff.

- **getCallWorkingHour():**

  A type of getter mrthod which returns WorkingHour of staff.

- **getCallJoiningDate():**

  It is a type of getter method. This method helps to return the joining date of staff.

- **getCallJobType():**

  It is a getter type method and JobType of staff was returned by it.

- **actionPerformed(ActionEvent e):**

This is a method which belongs to abstract class named Actionlistener. It is void method so that it doesn't have any return type. It is a method which is used in processing an action event which occurs when a user clicks a certain button. This method is connected with the function of the buttons.

- **addPartTimeStaffHire() method:**
  This method is void type method so this method doesn't return any type. This is a method which helps to add the PartTimeStaffHire in the Graphic User interface. Even it helps to check the vacancy number and other fields inside the GUI. If the fields are empty(or not filled) with different values then pop of occurs "Empty Fields" when click on button "Add Part".

- **addFullTimeStaffHire() method:**
  It is a void and private type method. It doesn't return any type. It is method in which FullTimeStaffHire Class is added. This method is associated with functionality of the buttons. Whenever the user clicks on the certain button, the task is performed whatever he/she want.

- **appointFullTimeStaff() method:**
  This is also a void type method in which the method doesn't return any type. In this method try and catch statement is used for testing the error while coding and to execute the block of codes if the error occurs in try block. And, in this method, the staffs are appointed after clicking over the specific button.
- **appointPartTimeStaff() method:**
  This method is void type method. This method doesn't return any type. In this method try and catch statement is used for testing and executing the codes in the the program. Here, PartTime staff is appointed by clicking over the certain button. And for appointing the staffs certain criteria must be filled.

- **terminatePartTimeStaff() method:**

A void type method which doesn’t return any type. In this method, the staff are terminated by clicking over the terminate button. And terminating staff is only applicable in PartTimeStaff only in this software. Try and catch statement is for handling and executing the program during coding.

- **void display():**
  It is a void type method and used to display the information of the staff while adding and appointing.

- **appointPartTimeStaffHire() method:**
  This is a void type method. This method doesn’t return any type. In this method, frame, panel, vacancy number, staff name, Qualification, joiningdate is used to make the GUI. Different buttons are used in this method for appoint Part Time staff.

- **appointFullTimeStaffHire() method:**
  This is a void type method. This method doesn’t return any type. Frame, label, textfield, combobox and bounds are set in this method. Different buttons like clear, appoint Full, display is used also.

- **clear() method:**
  This method helps to clear the values inside the Textfield after clicking over the clear button.

5.Testing:

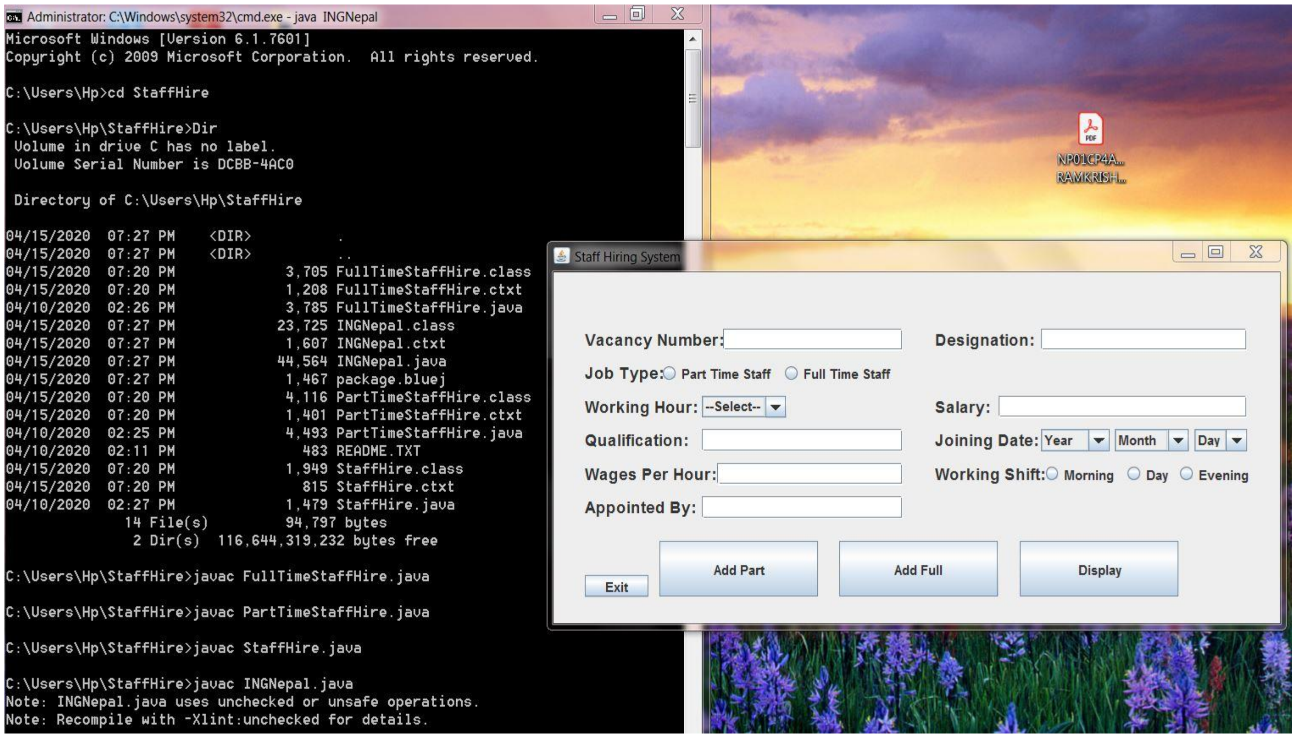
Testing is important after the completion of the program. It helps to know the nature of the program. And testing is done to clearify the doubt that the program is running smoothly or not. After the testing all the doubts are clear that the program is running smoothly without any obstacle. And the tests are done in three ways, first the program is opened in command prompt by giving certain command. Secondly, the program was opened and tested by adding and appointing the staff. Thirdly, by introducing the unvalid values inside the program.

And the screenshot after testing the program is given below:

5.1Test 1: Command prompt

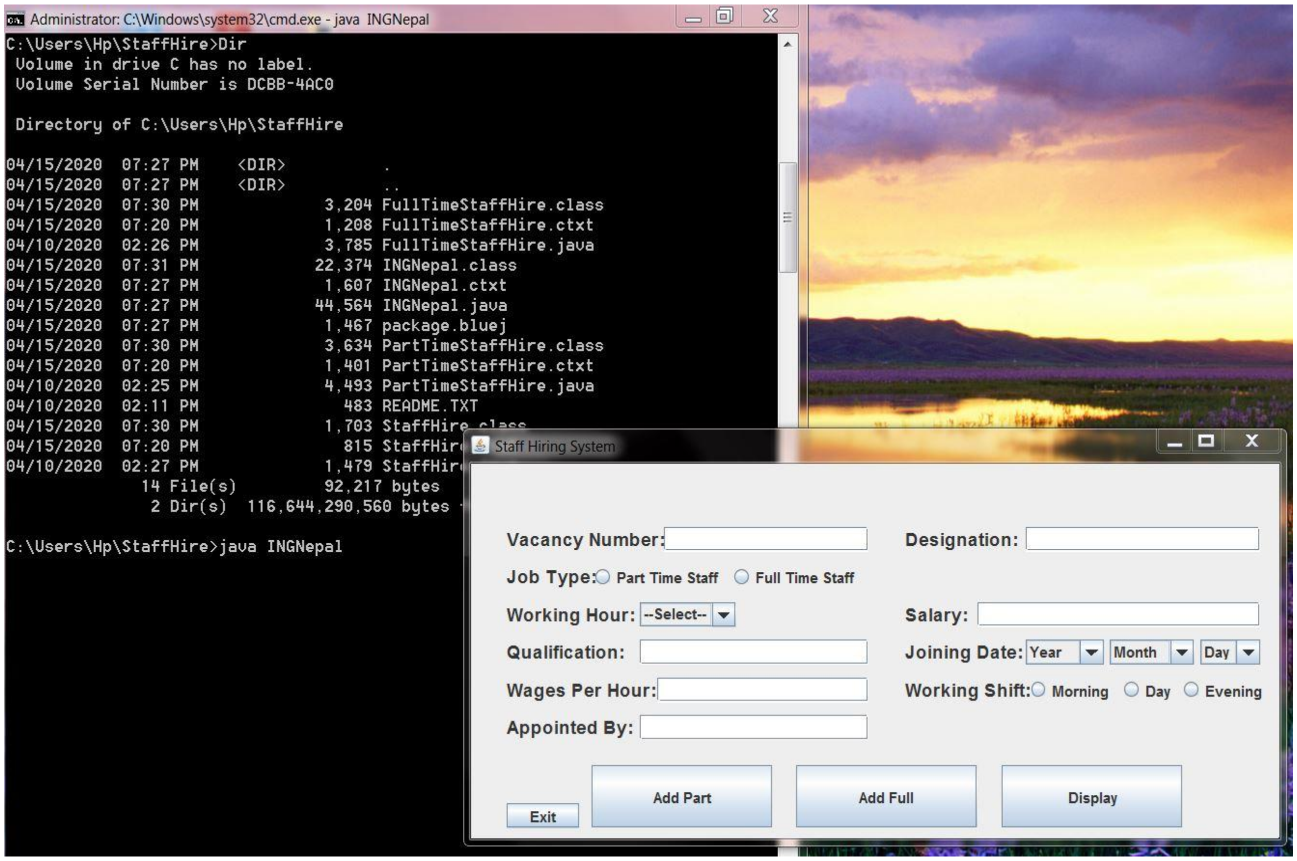
The program was compiled and runned by using command prompt. Firstly the program was compiled by providing command “javac classname” in cmd and after that it was runned by using command java “INGNepal”.

The screenshot of command with GUI is given below:



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Hp>cd StaffHire

C:\Users\Hp\StaffHire>Dir
 Volume in drive C has no label.
 Volume Serial Number is DCBB-4AC0

 Directory of C:\Users\Hp\StaffHire

04/15/2020  07:27 PM    <DIR>          .
04/15/2020  07:27 PM    <DIR>          ..
04/15/2020  07:20 PM             3,705 FullTimeStaffHire.class
04/15/2020  07:20 PM             1,208 FullTimeStaffHire.ctxt
04/10/2020  02:26 PM             3,785 FullTimeStaffHire.java
04/15/2020  07:27 PM            23,725 INGNepal.class
04/15/2020  07:27 PM             1,607 INGNepal.ctxt
04/15/2020  07:27 PM            44,564 INGNepal.java
04/15/2020  07:27 PM             1,467 package.bluej
04/15/2020  07:20 PM             4,116 PartTimeStaffHire.class
04/15/2020  07:20 PM             1,401 PartTimeStaffHire.ctxt
04/10/2020  02:25 PM             4,493 PartTimeStaffHire.java
04/10/2020  02:11 PM               483 README.TXT
04/15/2020  07:20 PM             1,949 StaffHire.class
04/15/2020  07:20 PM               815 StaffHire.ctxt
04/10/2020  02:27 PM             1,479 StaffHire.java
              14 File(s)         94,797 bytes
               2 Dir(s)  116,644,319,232 bytes free

C:\Users\Hp\StaffHire>javac FullTimeStaffHire.java

C:\Users\Hp\StaffHire>javac PartTimeStaffHire.java

C:\Users\Hp\StaffHire>javac StaffHire.java

C:\Users\Hp\StaffHire>javac INGNepal.java
Note: INGNepal.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
```

Figure 4: Command prompt



```
Administrator: C:\Windows\system32\cmd.exe - java INGNepal
C:\Users\Hp\StaffHire>Dir
 Volume in drive C has no label.
 Volume Serial Number is DCBB-4AC0

 Directory of C:\Users\Hp\StaffHire

04/15/2020  07:27 PM    <DIR>          .
04/15/2020  07:27 PM    <DIR>          ..
04/15/2020  07:30 PM             3,204 FullTimeStaffHire.class
04/15/2020  07:20 PM             1,208 FullTimeStaffHire.ctxt
04/10/2020  02:26 PM             3,785 FullTimeStaffHire.java
04/15/2020  07:31 PM            22,374 INGNepal.class
04/15/2020  07:27 PM             1,607 INGNepal.ctxt
04/15/2020  07:27 PM            44,564 INGNepal.java
04/15/2020  07:27 PM             1,467 package.bluej
04/15/2020  07:30 PM             3,634 PartTimeStaffHire.class
04/15/2020  07:20 PM             1,401 PartTimeStaffHire.ctxt
04/10/2020  02:25 PM             4,493 PartTimeStaffHire.java
04/10/2020  02:11 PM               483 README.TXT
04/15/2020  07:30 PM             1,703 StaffHire.class
04/15/2020  07:20 PM               815 StaffHir
04/10/2020  02:27 PM             1,479 StaffHir
              14 File(s)         92,217 bytes
               2 Dir(s)  116,644,290,560 bytes

C:\Users\Hp\StaffHire>java INGNepal
```

Figure 5:GUI run from command prompt

The information of the test in the form of table is listed below. The left side of the table shows the Objectives, action and results whereas the right side of the table shows the corresponding answer of them.

Objectives	To compile and run the program in command prompt.
Action	Different commands are written in the command prompt to compile and run the program from command prompt.
Expected Result	Program must be compiled and run.
Actual Results	Program is compiled and run successfully.
Results	Test is successfully done.

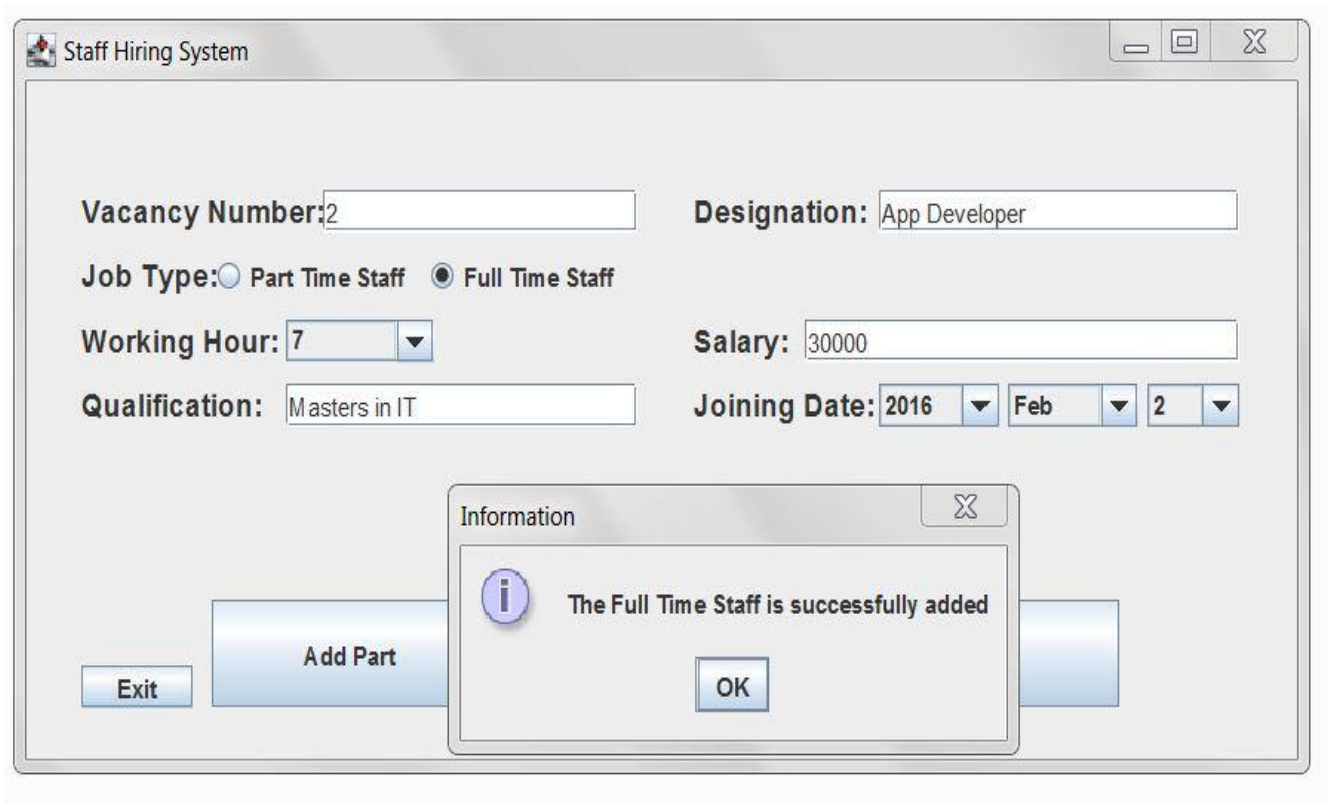
Table 2: Details for test of command prompt

5.2Test 2: Adding and Appointing the staffs

The staffs are added and appointed by introducing the values in the TextField and clicking over the certain button. Here, Part time and Full time staffs, both are added and appointed by pressing over the certain buttons. Whenever the user clicks over the “Add Part” button, the part time staff was added and after that clicking over the “Appoint Part” button, the was appoint. In the same way Full time staff was added and appointed by pressing over the “Add Part” and “Appoint Part” respectively.
The evidence of the test is shown in the form screenshot while adding and appointing the staff.

5.2.1FullTimeStaffHire

The screenshot is given below as the evidence while adding and appointing for Full time staff:

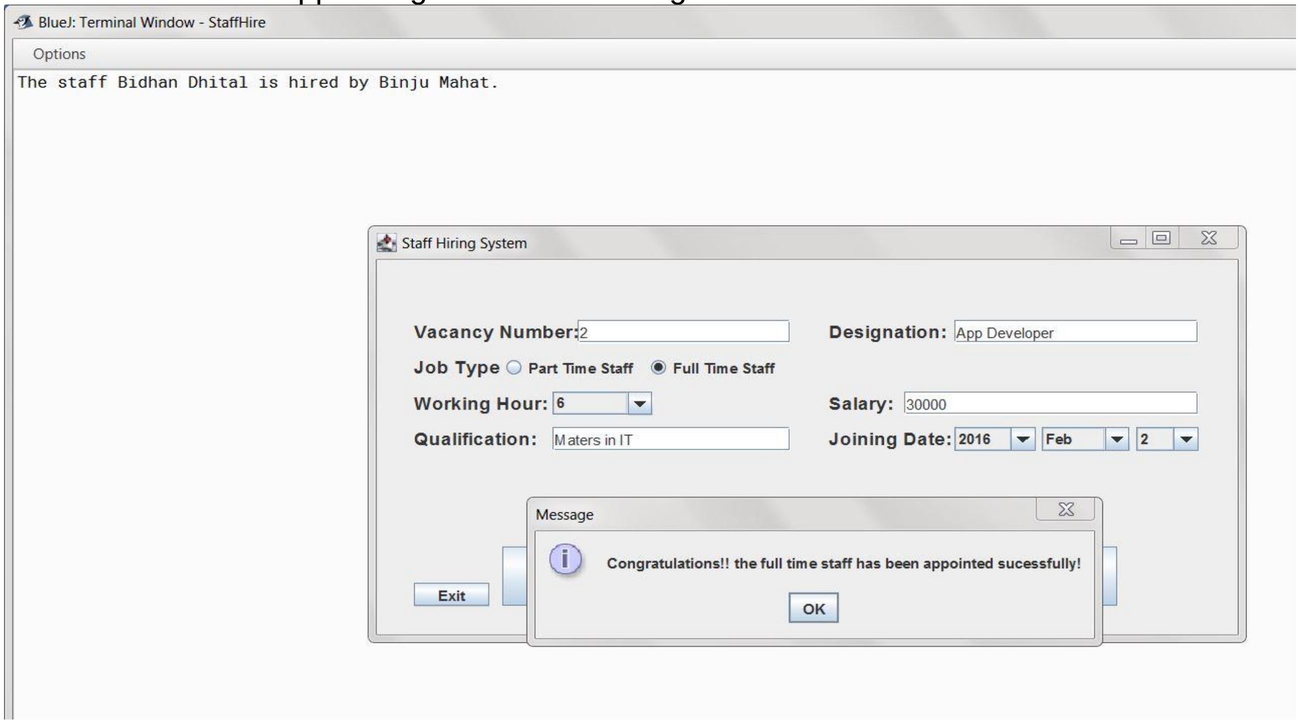


Vacancy Number: 2
Designation: App Developer
Job Type: ○ Part Time Staff ● Full Time Staff
Working Hour: 7
Salary: 30000
Qualification: Masters in IT
Joining Date: 2016 Feb 2

The Full Time Staff is successfully added

Figure 6: Adding FullTime Staff

The screenshot for appointing FullTime staff is given below:



```
The staff Bidhan Dhital is hired by Binju Mahat.
```

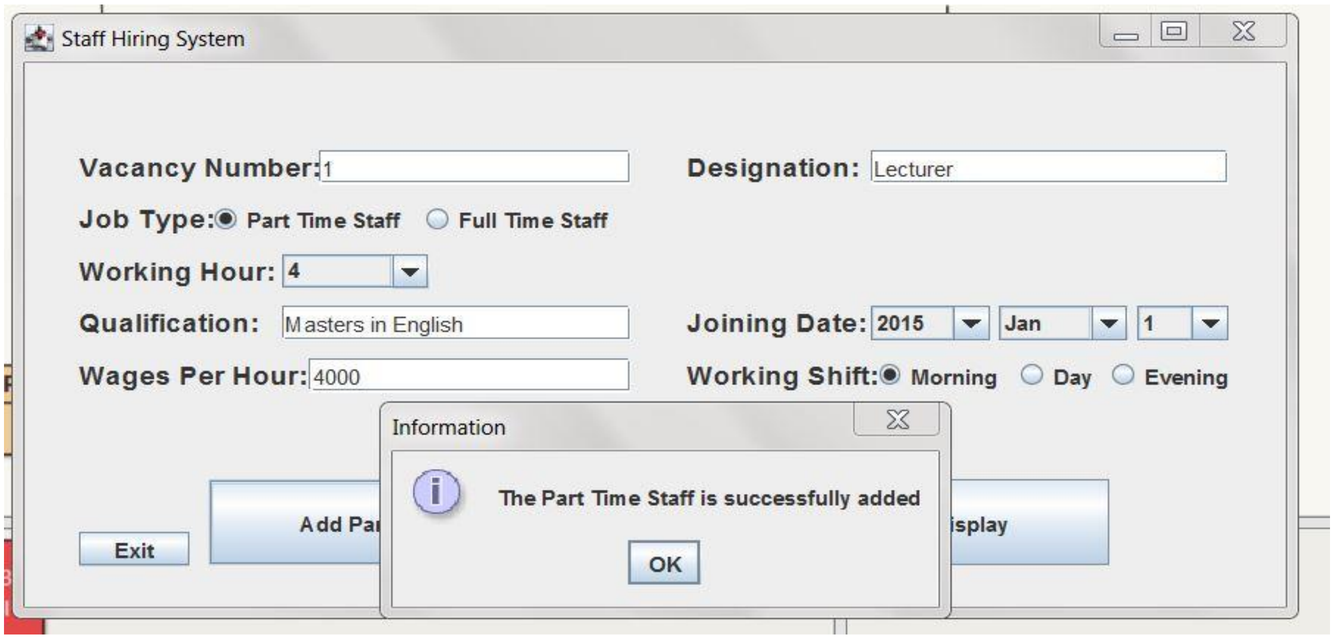
Vacancy Number: 2
Designation: App Developer
Job Type: ○ Part Time Staff ● Full Time Staff
Working Hour: 6
Salary: 30000
Qualification: Maters in IT
Joining Date: 2016 Feb 2

Congratulations!! the full time staff has been appointed sucessfully!

Figure 7: Appointing Full Time staff

5.2.2PartTimeStaffHire

The screenshot is given below as evidence while adding and appointing the staff as part time staff.

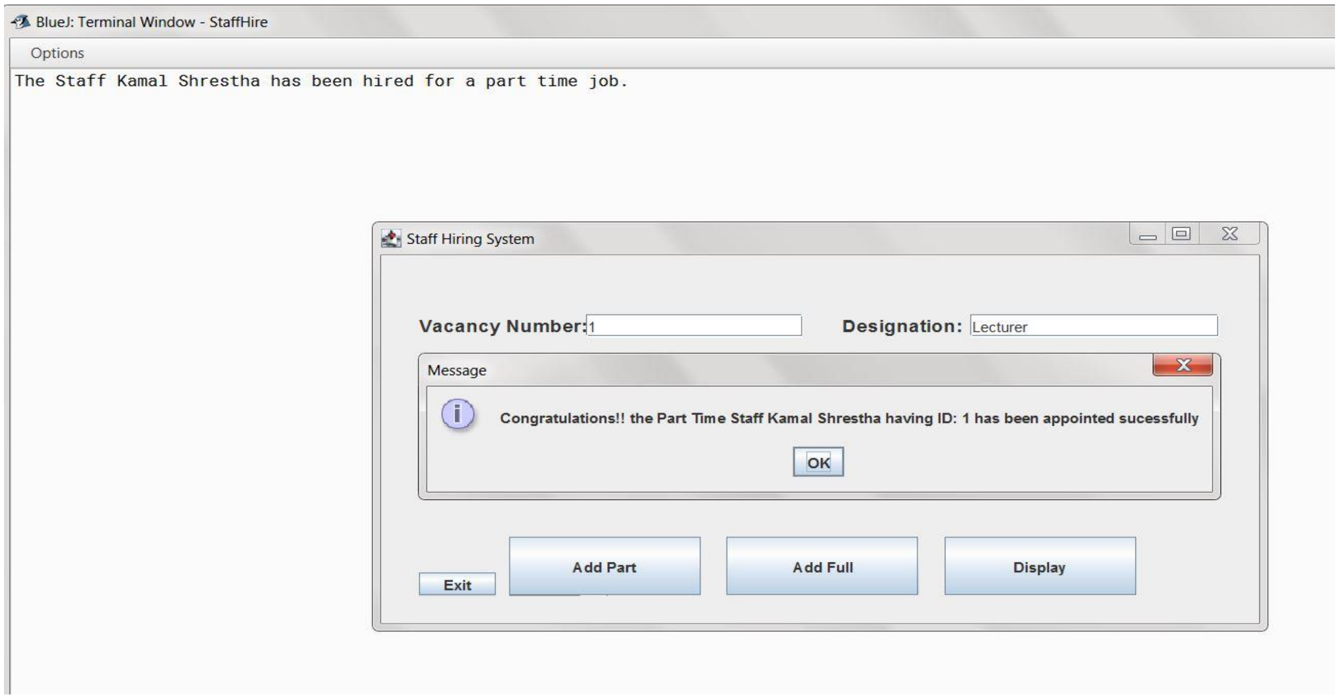


Vacancy Number: 1
Designation: Lecturer
Job Type: ● Part Time Staff ○ Full Time Staff
Working Hour: 4
Qualification: Masters in English
Joining Date: 2015 Jan 1
Wages Per Hour: 4000
Working Shift: ● Morning ○ Day ○ Evening

The Part Time Staff is successfully added

Figure 8: Adding Part Time Staff

The screenshot for appointing part time staff is given below:



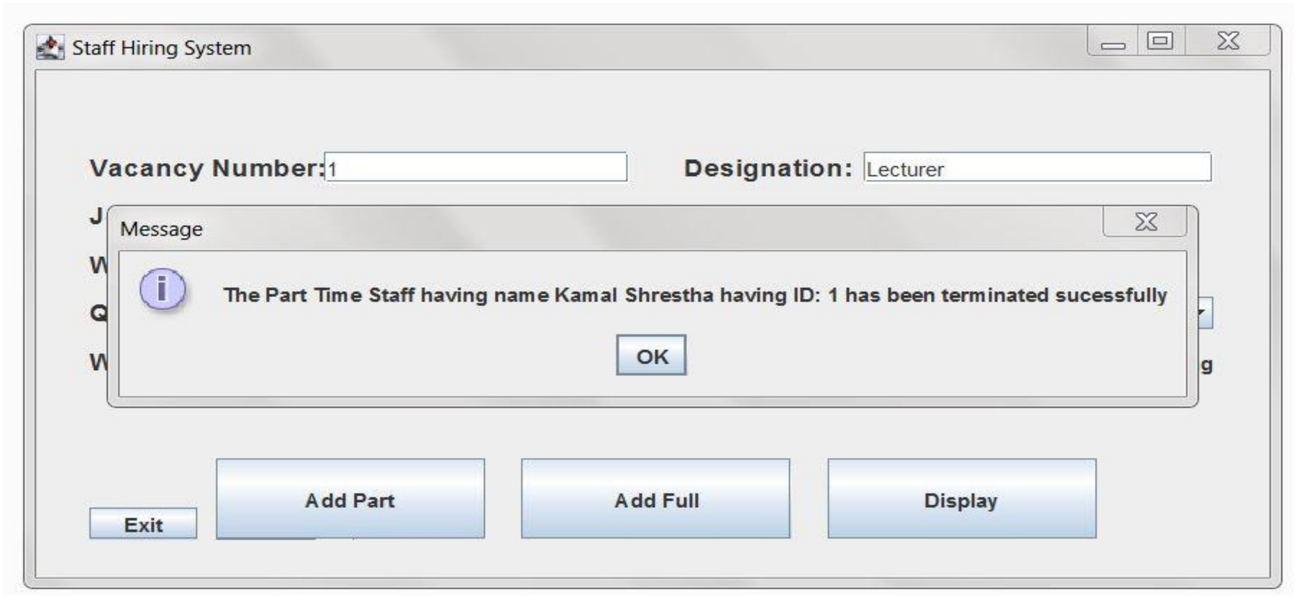
```
The Staff Kamal Shrestha has been hired for a part time job.
```

Congratulations!! the Part Time Staff Kamal Shrestha having ID: 1 has been appointed sucessfully

Figure 9: Appointing Part Time staff

5.2.3Terminating PartTimeStaffHire

In this Staffs are terminated by pressing over the terminate button as shown in the screenshot:



The Part Time Staff having name Kamal Shrestha having ID: 1 has been terminated sucessfully

Figure 10: Terminating Part Time Staff

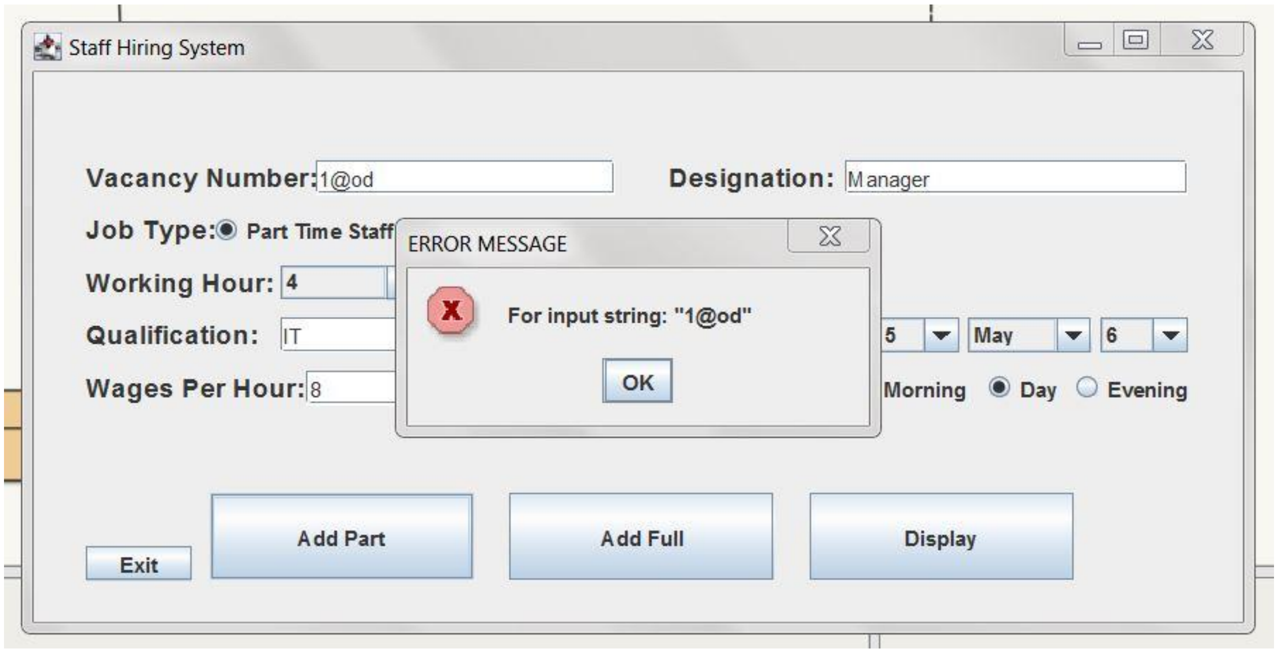
The information of the test in the form of table is listed below. The left side of the table shows the Objectives, action and results whereas the right side of the table shows the corresponding answer of them.

Objectives	To add and appoint the Part time and full time staff respectively. And to terminate the part time staff.
Action	Certain values are introduced to the Textfield area and the certain button was pressed for the adding, appointing and terminating the staffs. For appointing and adding part time staff, “Appoint Part” and “Add Part” button was pressed. Similarly While adding and appoint full time staff, “Add Part” and “Appoint Full” was pressed.
Expected Result	Staff must be added and appointed.
Actual Results	Staffs are added and appointed successfully.
Results	Test is successfully done.

Table 3: Details for test of adding and appointing staffs

5.3Test 3: Introducing the wrong values

In this test wrong data type was introduced in the Text Field. The test was to know whether the program is accepting the wrong data type or not. As the string data type is used in the Textfield instead of using integer data type. When it was introduced and pressed over the “Add Part” button the following information is shown in the below screenshot:


Staff Hiring System

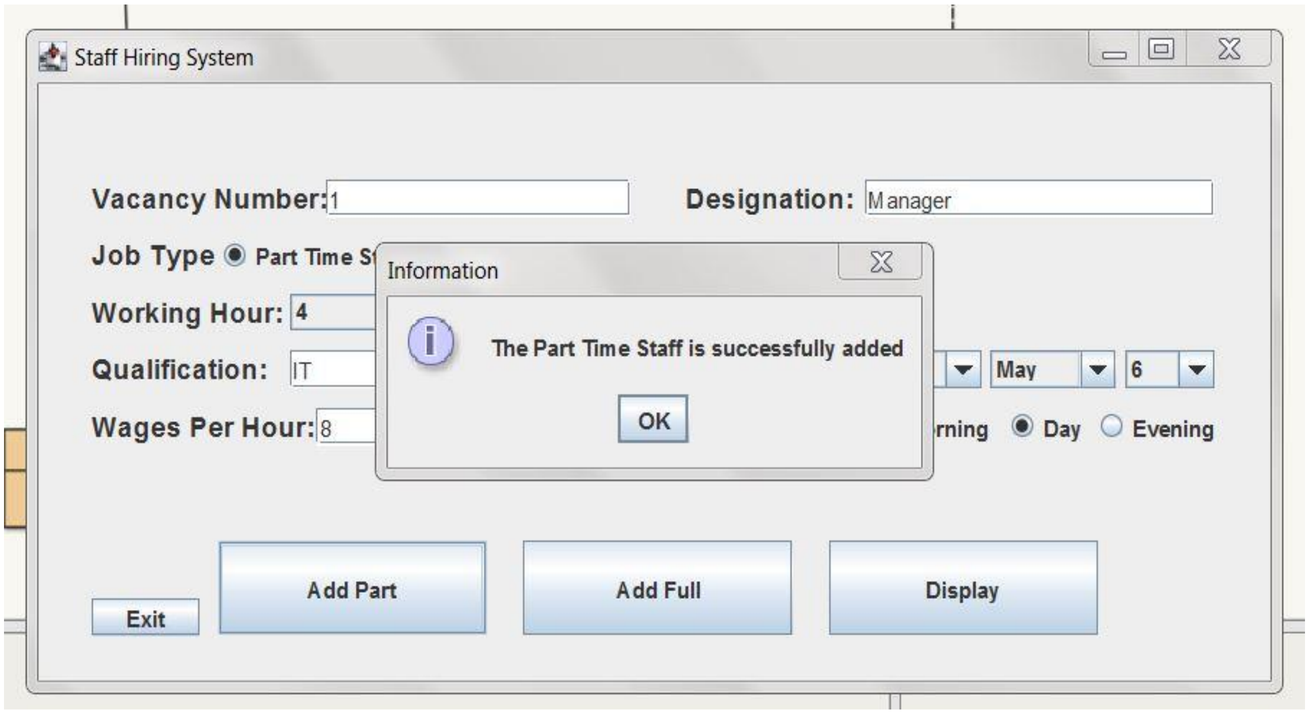
Vacancy Number: 1@od
Designation: Manager
Job Type: ● Part Time Staff
Working Hour: 4
Qualification: IT
Wages Per Hour: 8

ERROR MESSAGE

For input string: "1@od"

Figure 11: Introducing wrong Data Type

When the integer data type was introduced, it won't show the error message. The screenshot for it, is shown as evidence.


Staff Hiring System

Vacancy Number: 1
Designation: Manager
Job Type ● Part Time S
Working Hour: 4
Qualification: IT
Wages Per Hour: 8

Information

The Part Time Staff is successfully added

Figure 12: Intoducing the right Data Type

The information of the test in the form of table is listed below. The left side of the table shows the Objectives, action and results whereas the right side of the table shows the corresponding answer of them.

Objectives	To test the program whether the it is accepting the wrong data type or not.
Action	String data type was introduced in the TextField instead of introducing the integer data type.
Expected Result	Program must not accept the String Data type in integer data type place.
Actual Results	Program didn’t accept wrong data type.
Results	Test is successfully done.

Figure 13: Details For Test 3

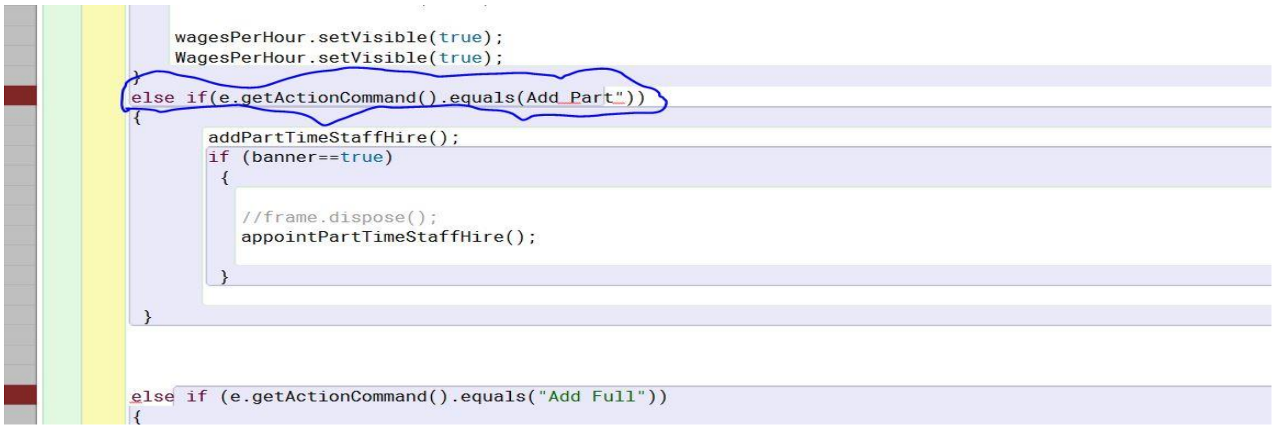
6.Errors detection and Error correction:

As developing the program, lots of error are detected while compiling the program. And hence it was corrected as well. Some of the errors are shown in form screenshot below:

6.1Syntax Error:

Syntax error is mostly occurred due to missing of the sign and symbols. The error is shown below in form of screenshot with its solution.

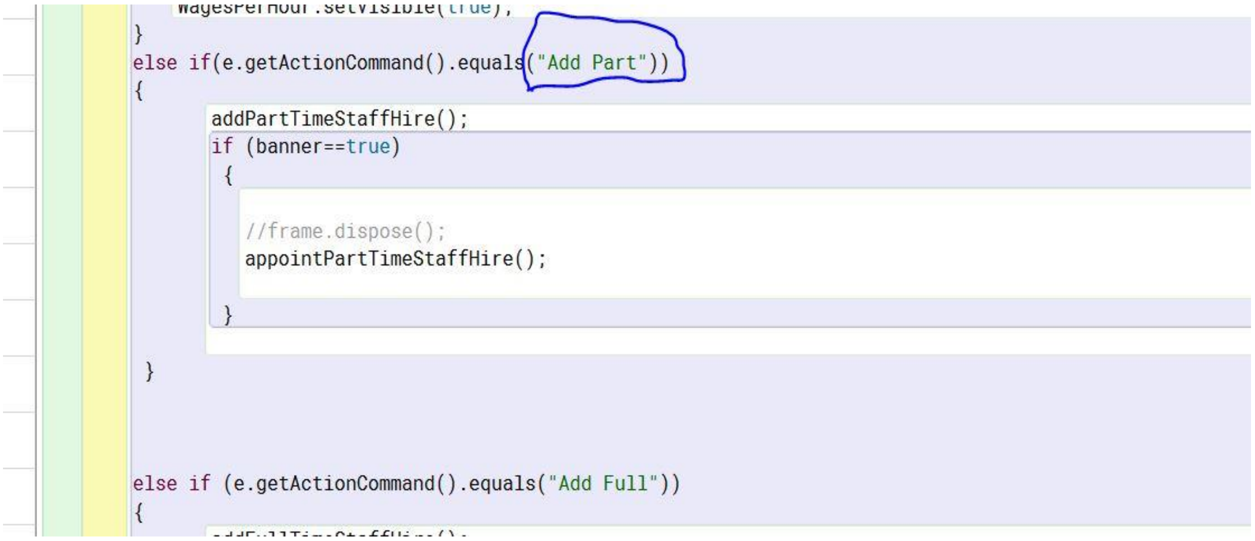
In the below screenshot, the double quotation mark is left, and after compiling the error is detected.



```
    wagesPerHour.setVisible(true);
    WagesPerHour.setVisible(true);
}
else if(e.getActionCommand().equals(Add Part"))
{
        addPartTimeStaffHire();
        if (banner==true)
         {
            //frame.dispose();
            appointPartTimeStaffHire();
         }
 }


else if (e.getActionCommand().equals("Add Full"))
{
```

Figure 14: Syntax Error

The error was corrected by introducing double quotation sign and was shown in the form of screenshot. it was underlined by blue colour line.



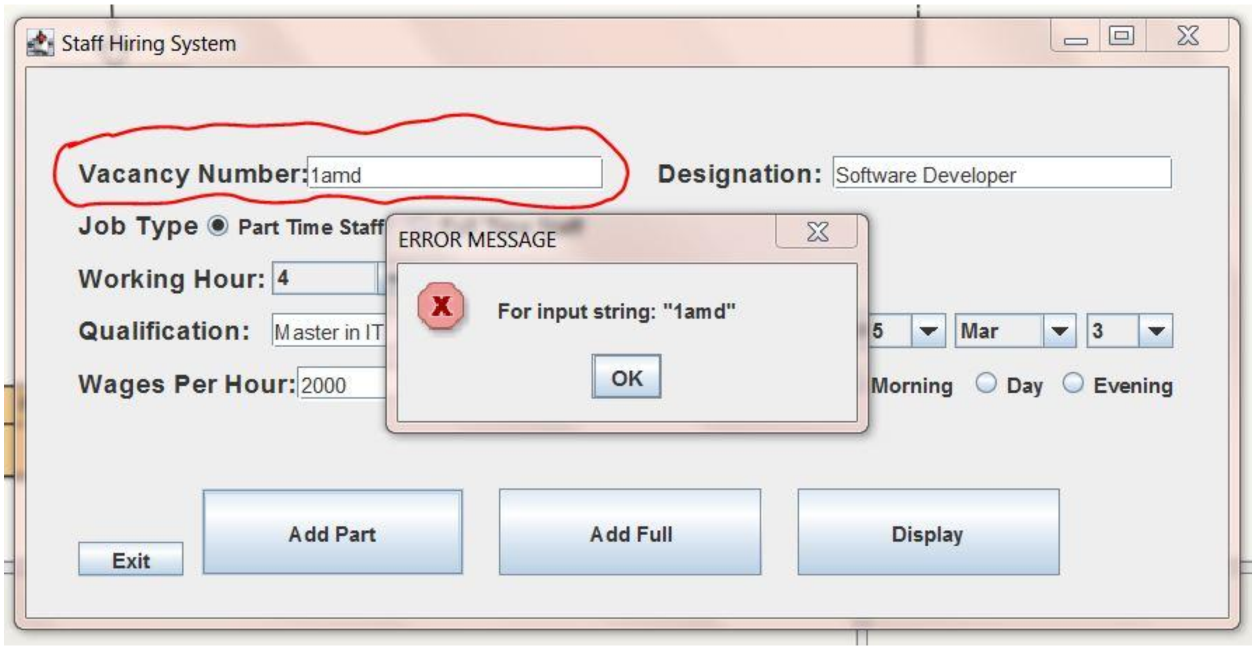
```
}
else if(e.getActionCommand().equals("Add Part"))
{
        addPartTimeStaffHire();
        if (banner==true)
         {

            //frame.dispose();
            appointPartTimeStaffHire();

         }
 }



else if (e.getActionCommand().equals("Add Full"))
{
```

Figure 15: Solved Syntax Error

6.2Runtime Error:

Runtime error is simply known as a program error which occurs while running the program. And the error is shown below in the form screenshot:

In this screenshot runtime error is shown. Here, the user had input the string data type instead of integer data type so that the error message is occurred.



Staff Hiring System

Vacancy Number: 1amd

Designation: Software Developer

Job Type ● Part Time Staff

Working Hour: 4

Qualification: Master in IT

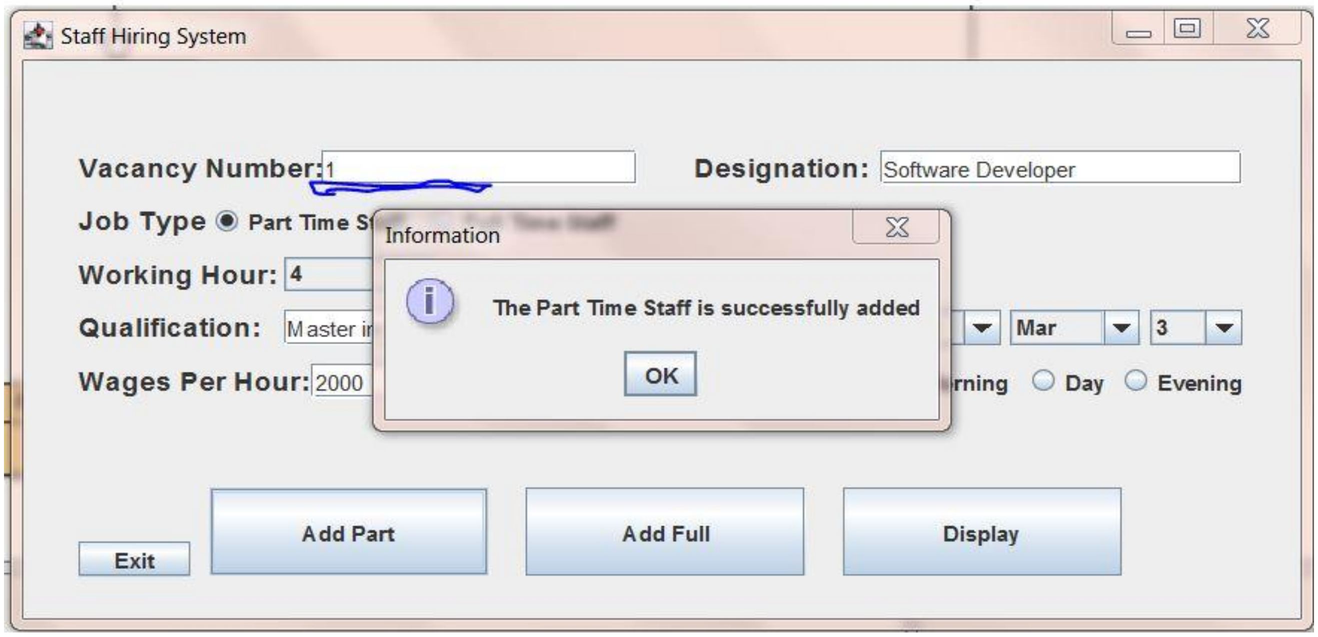
Wages Per Hour: 2000

ERROR MESSAGE

For input string: "1amd"

Figure 16: RunTime Error

And the runtime error is solved by introducing integer variable instead of string variable. And solved screenshot is shown below:


Staff Hiring System

Vacancy Number: 1
Designation: Software Developer
Job Type ● Part Time S
Working Hour: 4
Qualification: Master i
Wages Per Hour: 2000
Mar 3
rning ○ Day ○ Evening

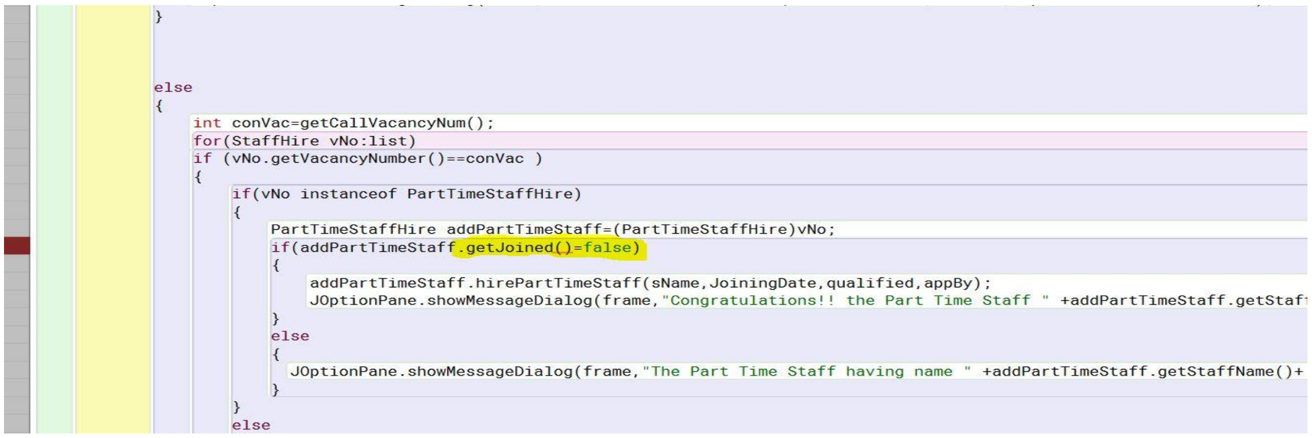
Information
The Part Time Staff is successfully added
OK

Exit
Add Part
Add Full
Display

Figure 17: RunTime Error Solved

6.3Logical Error:

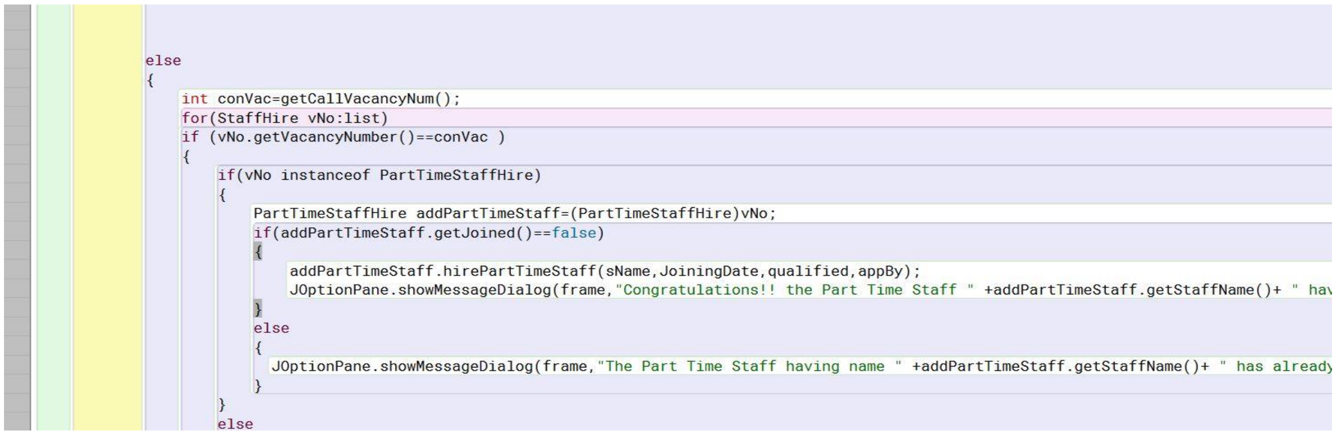
The unwanted or unexpected behaviour in the program is logical error. It is due to the mistake in source code of program. And one of the screenshots of the logical error is shown below:


```
}



else
{
    int conVac=getCallVacancyNum();
    for(StaffHire vNo:list)
    if (vNo.getVacancyNumber()==conVac )
    {
        if(vNo instanceof PartTimeStaffHire)
        {
            PartTimeStaffHire addPartTimeStaff=(PartTimeStaffHire)vNo;
            if(addPartTimeStaff.getJoined()=false)
            {
                addPartTimeStaff.hirePartTimeStaff(sName,JoiningDate,qualified,appBy);
                JOptionPane.showMessageDialog(frame,"Congratulations!! the Part Time Staff " +addPartTimeStaff.getStaf
            }
            else
            {
              JOptionPane.showMessageDialog(frame,"The Part Time Staff having name " +addPartTimeStaff.getStaffName()+
            }
        }
        else
```

.Figure 18: Logical Error

In the above screenshot, error is highlighted with the green colour. There two equals are needed to complete the code but it was forgotten.

And hence it was solved by introducing the equals to and screenshot was shown below:



```
else
{
    int conVac=getCallVacancyNum();
    for(StaffHire vNo:list)
    if (vNo.getVacancyNumber()==conVac )
    {
        if(vNo instanceof PartTimeStaffHire)
        {
            PartTimeStaffHire addPartTimeStaff=(PartTimeStaffHire)vNo;
            if(addPartTimeStaff.getJoined()==false)
            {
                addPartTimeStaff.hirePartTimeStaff(sName,JoiningDate,qualified,appBy);
                JOptionPane.showMessageDialog(frame,"Congratulations!! the Part Time Staff " +addPartTimeStaff.getStaffName()+ " hav
            }
            else
            {
             JOptionPane.showMessageDialog(frame,"The Part Time Staff having name " +addPartTimeStaff.getStaffName()+ " has already
            }
        }
        else
```

Figure 19 : Solved Logical Error

7.Conclusion:

As we are instructed to develop the program which can hire the staff for the organization. The work is given to all students individually. while developing the application lots of errors and bugs are detected. Even, at first I felt so difficulties while coding(or developing) the program. Sometimes, programs are compiled but it shows the usual behaviour after running it. Unwanted actions are shown by the programs.

So, to tackle this type of issue inside the program, lots of research are carried out. Knowledge over this coursework was gained from the internet, books and from teachers. After the research over the related topic of this coursework, it became easy for me to develop the program as per the instruction of the coursework. And hence, the course was finally developed by detecting the errors and correcting it.

After the completion of the program, tests are done to clearify the doubt whether the program is working smoothly or not. And finally, the program found that it is running smoothly. Lots of knowledge over the related topic was learnt by me. And with the help of deep research, I became able accomplish this coursework.

8.Appendix:

The full code of the program basically known as Appendix. The appendix for the program is given below:

8.1INGNepal

```
public class INGNepal implements ActionListener
{
   //creating main frame named frame
   private JFrame frame,frameFull,framePart;
   //creating necessary labels
   private JLabel title, VacancyNumber, Designation, JobType, Salary, WorkingHour,ApplicantName, StaffName, Qualification, JoiningDate, AppointedBy, WorkingShift, WagesPerHour;
   private JTextField vacancyNumber, designation, salary,applicantName, staffName, qualification, appointedBy, wagesPerHour;
   private JComboBox workingHour, joiningDateYear,joiningDateDay,joiningDateMonth;
   private JRadioButton fullTimeStaffHire,partTimeStaffHire,morning,day,evening;
   private JButton AddPartTimeStaffHire, AddFullTimeStaffHire,AppointFullTimeStaffHire,AppointPartTimeStaffHire,Clear,Terminate,display,Display,Exit,ExitPart,ExitFull;
   private JPanel p1;
   private boolean banner=false;
   private boolean vacNumberCheck=false;
   private ButtonGroup workingShift,jobType;
   ArrayList<StaffHire>list=new ArrayList<StaffHire>();

   public INGNepal()
   {
      frame = new JFrame("Staff Hiring System");
      frame.setSize(700,350);
```

```
p1=new JPanel();
p1.setLayout(null);
p1.setBounds(10,10,656,300);
p1.setVisible(true);
frame.add(p1);

title=new JLabel("Form of Staff Hiring");
title.setBounds(250,5,300,30);
title.setFont(new Font("Arial",Font.BOLD,15));
//p1.add(title);


VacancyNumber = new JLabel("Vacancy Number: ");
VacancyNumber.setBounds(20,40,150,20);
VacancyNumber.setFont(new Font("Arial",Font.BOLD,15));
p1.add(VacancyNumber);

Designation=new JLabel("Designation:");
Designation.setBounds(350,40,130,20);
Designation.setFont(new Font("Arial",Font.BOLD,15));
p1.add(Designation);

JobType=new JLabel("Job Type:");
JobType.setBounds(20,70,80,20);
JobType.setFont(new Font("Arial",Font.BOLD,15));
p1.add(JobType);

WorkingHour=new JLabel("Working Hour:");
WorkingHour.setBounds(20,100,130,20);
WorkingHour.setFont(new Font("Arial",Font.BOLD,15));
p1.add(WorkingHour);
```

```
Salary=new JLabel("Salary:");
Salary.setBounds(350,100,70,20);
Salary.setFont(new Font("Arial",Font.BOLD,15));
p1.add(Salary);

Qualification=new JLabel("Qualification:");
Qualification.setBounds(20,130,100,20);
Qualification.setFont(new Font("Arial",Font.BOLD,15));
p1.add(Qualification);

JoiningDate=new JLabel("Joining Date:");
JoiningDate.setBounds(350,130,130,20);
JoiningDate.setFont(new Font("Arial",Font.BOLD,15));
p1.add(JoiningDate);

WagesPerHour=new JLabel("Wages Per Hour:");
WagesPerHour.setBounds(20,160,140,20);
WagesPerHour.setFont(new Font("Arial",Font.BOLD,15));
p1.add(WagesPerHour);

WorkingShift=new JLabel("Working Shift:");
WorkingShift.setBounds(350,160,160,20);
WorkingShift.setFont(new Font("Arial",Font.BOLD,15));
p1.add(WorkingShift);

AppointedBy=new JLabel("Appointed By:");
AppointedBy.setBounds(20,190,130,20);
AppointedBy.setFont(new Font("Arial",Font.BOLD,15));
p1.add(AppointedBy);
//txt
```

```
vacancyNumber= new JTextField();
vacancyNumber .setBounds(150,40,170,20);
p1.add(vacancyNumber );

designation=new JTextField();
designation.setBounds(450,40,195,20);
p1.add(designation);

partTimeStaffHire=new JRadioButton("Part Time Staff");
partTimeStaffHire.setBounds(90,70,115,20);
partTimeStaffHire.setActionCommand("Part Time Staff");
partTimeStaffHire.addActionListener(this);
ButtonGroup job_Type=new ButtonGroup();
job_Type.add(partTimeStaffHire);
p1.add(partTimeStaffHire,true);



fullTimeStaffHire=new JRadioButton("Full Time Staff");
fullTimeStaffHire.setBounds(205,70,115,20);
fullTimeStaffHire.setActionCommand("Full Time Staff");
fullTimeStaffHire.addActionListener(this);
job_Type.add(fullTimeStaffHire);
p1.add(fullTimeStaffHire);



salary=new JTextField();
salary.setBounds(410,100,235,20);
p1.add(salary);

workingHour=new JComboBox();
```

```
workingHour.addItem("--Select--");
workingHour.addItem(1);
workingHour.addItem(2);
workingHour.addItem(3);
workingHour.addItem(4);
workingHour.addItem(5);
workingHour.addItem(6);
workingHour.addItem(7);
workingHour.addItem(8);
workingHour.addItem(9);
workingHour.addItem(10);
workingHour.addItem(11);
workingHour.addItem(12);
workingHour.setBounds(130,100,80,20);
p1.add(workingHour);


qualification=new JTextField();
qualification.setBounds(130,130,190,20);
p1.add(qualification);

joiningDateYear=new JComboBox();
joiningDateYear.addItem("Year");
joiningDateYear.addItem(2015);
joiningDateYear.addItem(2016);
joiningDateYear.addItem(2017);
joiningDateYear.addItem(2018);
joiningDateYear.addItem(2019);
joiningDateYear.addItem(2020);
joiningDateYear.addItem(2021);
joiningDateYear.addItem(2022);
joiningDateYear.addItem(2023);
joiningDateYear.addItem(2024);
joiningDateYear.addItem(2025);
joiningDateYear.addItem(2026);
joiningDateYear.addItem(2027);
joiningDateYear.addItem(2028);
joiningDateYear.addItem(2029);
joiningDateYear.addItem(2030);
joiningDateYear.addItem(2031);
joiningDateYear.addItem(2032);
joiningDateYear.addItem(2033);
joiningDateYear.addItem(2034);
joiningDateYear.addItem(2035);
joiningDateYear.addItem(2036);
joiningDateYear.addItem(2037);
joiningDateYear.addItem(2038);
joiningDateYear.addItem(2039);
joiningDateYear.addItem(2040);
joiningDateYear.addItem(2041);
joiningDateYear.addItem(2042);
joiningDateYear.addItem(2043);
joiningDateYear.addItem(2044);
joiningDateYear.addItem(2045);
joiningDateYear.addItem(2046);
joiningDateYear.addItem(2047);
joiningDateYear.addItem(2048);
joiningDateYear.addItem(2049);
joiningDateYear.addItem(2050);
joiningDateYear.setBounds(450,130,65,20);
p1.add(joiningDateYear);
```

```
joiningDateMonth=new JComboBox();
joiningDateMonth.addItem("Month");
joiningDateMonth.addItem("Jan");
joiningDateMonth.addItem("Feb");
joiningDateMonth.addItem("Mar");
joiningDateMonth.addItem("April");
joiningDateMonth.addItem("May");
joiningDateMonth.addItem("June");
joiningDateMonth.addItem("July");
joiningDateMonth.addItem("Aug");
joiningDateMonth.addItem("Sep");
joiningDateMonth.addItem("Oct");
joiningDateMonth.addItem("Nov");
joiningDateMonth.addItem("Dec");
joiningDateMonth.setBounds(520,130,70,20);
p1.add(joiningDateMonth);

joiningDateDay=new JComboBox();
joiningDateDay.addItem("Day");
joiningDateDay.addItem(1);
joiningDateDay.addItem(2);
joiningDateDay.addItem(3);
joiningDateDay.addItem(4);
joiningDateDay.addItem(5);
joiningDateDay.addItem(6);
joiningDateDay.addItem(7);
joiningDateDay.addItem(8);
joiningDateDay.addItem(9);
joiningDateDay.addItem(10);
joiningDateDay.addItem(11);
joiningDateDay.addItem(12);
joiningDateDay.addItem(13);
joiningDateDay.addItem(14);
joiningDateDay.addItem(15);
joiningDateDay.addItem(16);
joiningDateDay.addItem(17);
joiningDateDay.addItem(18);
joiningDateDay.addItem(19);
joiningDateDay.addItem(20);
joiningDateDay.addItem(21);
joiningDateDay.addItem(22);
joiningDateDay.addItem(23);
joiningDateDay.addItem(24);
joiningDateDay.addItem(25);
joiningDateDay.addItem(26);
joiningDateDay.addItem(27);
joiningDateDay.addItem(28);
joiningDateDay.addItem(29);
joiningDateDay.addItem(30);
joiningDateDay.addItem(31);
joiningDateDay.addItem(32);
joiningDateDay.setBounds(595,130,50,20);
p1.add(joiningDateDay);

wagesPerHour=new JTextField();
wagesPerHour.setBounds(145,160,175,20);
p1.add(wagesPerHour);

morning=new JRadioButton("Morning");
morning.setBounds(451,160,75,20);
morning.setActionCommand("Morning");
morning.addActionListener(this);
```

```
p1.add(morning);
ButtonGroup workingShift = new ButtonGroup();
workingShift.add(morning);

day=new JRadioButton("Day");
day.setBounds(528,160,50,20);
day.setActionCommand("Day");
day.addActionListener(this);
p1.add(day);
workingShift.add(day);

evening=new JRadioButton("Evening");
evening.setBounds(578,160,75,20);
evening.setActionCommand("Evening");
evening.addActionListener(this);
p1.add(evening);
workingShift.add(evening);

appointedBy=new JTextField();
appointedBy.setBounds(130,190,190,20);
p1.add(appointedBy);


AddPartTimeStaffHire=new JButton("Add Part");
AddPartTimeStaffHire.setBounds(90,230,150,50);
AddPartTimeStaffHire.setActionCommand("Add Part");
AddPartTimeStaffHire.addActionListener(this);
p1.add(AddPartTimeStaffHire);

AddFullTimeStaffHire=new JButton("Add Full");
AddFullTimeStaffHire.setBounds(260,230,150,50);
```

```
        AddFullTimeStaffHire.setActionCommand("Add Full");
        AddFullTimeStaffHire.addActionListener(this);
        p1.add(AddFullTimeStaffHire);


        display=new JButton("Display");
        display.setBounds(430,230,150,50);
        display.setActionCommand("Display");
        display.addActionListener(this);
        p1.add(display);

        Exit=new JButton("Exit");
        Exit.setBounds(20,260,60,20);
        Exit.setActionCommand("Exit");
        Exit.addActionListener(this);
        p1.add(Exit);



        frame.setLocationRelativeTo(null);
        frame.setLayout(null);
        frame.setVisible(true);
    }
    public static void main(String args[]){
        new INGNepal();
    }

    public int getCallVacancyNum()
    {
        int conVac=Integer.parseInt(vacancyNumber.getText());
        return conVac;
    }
    public String getCallDesignation()
    {
        String des;
        des= designation.getText();
        return des;
    }
    public int getCallSalary()
    {
        int conSalary=Integer.parseInt(salary.getText());
        return conSalary;
    }
    public String getCallApplicantName()
    {
        String appName;
        appName= applicantName.getText();
        return appName;
    }
    public String getCallStaffName()
    {
        String sName;
        sName=staffName.getText();
        return sName;
    }
    public String getCallQualification()
    {
        String qualified;
        qualified=qualification.getText();
        return qualified;
    }
    public String getCallAppointedBy()
    {
        String appBy;
        appBy=appointedBy.getText();
        return appBy;
    }
    public int getCallWagesPerHour()
    {
        int wPHour=Integer.parseInt(wagesPerHour.getText());
        return wPHour;
    }
    public int getCallWorkingHour()
    {
      int wHour=(int) workingHour.getSelectedItem();
      return wHour;

    }
    public String getCallJoiningDate()
    {
        String JoiningDate;
        JoiningDate=            joiningDateYear.getSelectedItem().toString()            +"/"+
 joiningDateMonth.getSelectedItem().toString()            +            "/"            +
 joiningDateDay.getSelectedItem().toString();
        return JoiningDate;

    }

    public String getCallJobType()
    {
       String jType= jobType.getSelection().getActionCommand();
       return jType;
```

```
}
public void actionPerformed(ActionEvent e)
{
    if(e.getSource()==Exit)
    {
        frame.dispose();
    }
    else if(e.getSource()==ExitFull)
    {
        frameFull.dispose();
    }
    else if(e.getSource()==ExitPart)
    {
        framePart.dispose();
    }
    else if (e.getActionCommand().equals("Full Time Staff"))
    {  //Making required textfields and labels only visible
        WorkingShift.setVisible(false);
        morning.setVisible(false);
        day.setVisible(false);
        evening.setVisible(false);

        Salary.setVisible(true);
        salary.setVisible(true);

        appointedBy.setVisible(false);
        AppointedBy.setVisible(false);

        wagesPerHour.setVisible(false);
        WagesPerHour.setVisible(false);
    }
```

```
else if (e.getActionCommand().equals("Part Time Staff"))
{  //Making required textfields and labels only visible
    WorkingShift.setVisible(true);
    morning.setVisible(true);
    day.setVisible(true);
    evening.setVisible(true);

    Salary.setVisible(false);
    salary.setVisible(false);

    appointedBy.setVisible(false);
    AppointedBy.setVisible(false);

    wagesPerHour.setVisible(true);
    WagesPerHour.setVisible(true);
}
else if(e.getActionCommand().equals("Add Part"))
{
       addPartTimeStaffHire();
       if (banner==true)
        {

          //frame.dispose();
          appointPartTimeStaffHire();


        }

 }
```

```
else if (e.getActionCommand().equals("Add Full"))
{
      addFullTimeStaffHire();
      if (banner==true)
      {



            //frame.dispose();
            appointFullTimeStaffHire();


      }
}






else if (e.getActionCommand().equals("Appoint Full"))
   {
      appointFullTimeStaff();
   }

else if (e.getActionCommand().equals("Appoint Part"))
   {
      appointPartTimeStaff();
   }

else if (e.getActionCommand().equals("Terminate"))
   {
      terminatePartTimeStaff();
   }
```

```
        else if (e.getActionCommand().equals("Display Part Time Staff"))
          {
            display();
          }
        else if (e.getActionCommand().equals("Display Full Time Staff"))
          {
            display();
          }
        else if (e.getActionCommand().equals("Display"))
          {
            display();
          }

        else if (e.getActionCommand().equals("Clear"))
          {
            clear();
          }
       }
```

8.2Adding PartTimeStaffHire

```
    public void addPartTimeStaffHire()
    {
        try
        {

          vacNumberCheck=false;
          for(StaffHire staff:list)
          {
           if (getCallVacancyNum()==staff.getVacancyNumber())
           {
```

```
        vacNumberCheck=true;
      }

      }

     if(vacNumberCheck==false)
      {
   if   (vacancyNumber.getText().equals("")||   workingHour.getSelectedItem()=="--Select--"||  wagesPerHour.getText().equals("")  ||  designation.getText().equals("")  || joiningDateYear.getSelectedItem()=="Year" || joiningDateMonth.getSelectedItem()=="Month" || joiningDateDay.getSelectedItem()=="Day")
   {
       // System.out.println("error");//throw new Exception("Cannot leave the field blank!");
       JOptionPane.showMessageDialog(frame,"Empty Fields","Checking Input Value",JOptionPane.INFORMATION_MESSAGE);

   }
   else{
        int conVac=getCallVacancyNum();
        String des=getCallDesignation();
        String jobType= (partTimeStaffHire.isSelected()) ? "Part Time" :  "Full Time";
        int wHour= getCallWorkingHour();
        int wPHour=getCallWagesPerHour();
        String wShift = (morning.isSelected()) ? "Morning" : (day.isSelected()) ? "Day" : "Evening";

       if(!partTimeStaffHire.isSelected())
       {
            JOptionPane.showMessageDialog(null,"The job type you selected isnot
valid!","Checking radio button Value",JOptionPane.INFORMATION_MESSAGE);

            }
        else
            {

                //creating object
                PartTimeStaffHire addPartTimeStaff=new PartTimeStaffHire( conVac,
des,jobType,  wHour,  wPHour,  wShift);
                //Adding the object to the arraylist
                list.add(addPartTimeStaff);
                JOptionPane.showMessageDialog(frame,"The Part Time Staff is
successfully added","Information",JOptionPane.INFORMATION_MESSAGE);
                this.banner=true;
            }
        }
        }

      else
    {
     JOptionPane.showMessageDialog(null,"Tvacancy Number already exist!","checking
vacancy number",JOptionPane.INFORMATION_MESSAGE);

    }
}
        catch(NumberFormatException e1)
        {
           //Fetching the message
            JOptionPane.showMessageDialog(frame,e1.getMessage(),"ERROR
MESSAGE",JOptionPane.ERROR_MESSAGE);
```

```
            this.banner=false;
        }
        catch(ArithmeticException e2)
        {
            JOptionPane.showMessageDialog(frame,e2.getMessage(),"ERROR
Arithmetic",JOptionPane.ERROR_MESSAGE);
            this.banner=false;

        }

    }
```

8.3Adding FullTimeStaffHire

```
    public void addFullTimeStaffHire()
    {
        try
        {

        //Checking if the field is empty
        if (vacancyNumber.getText().equals("")|| workingHour.getSelectedItem()=="--
Select--"|| designation.getText().equals("") ||
joiningDateYear.getSelectedItem()=="Year" ||
joiningDateMonth.getSelectedItem()=="Month" ||
joiningDateDay.getSelectedItem()=="Day")
        {
            // System.out.println("error");//throw new Exception("Cannot leave the field
blank!");
            JOptionPane.showMessageDialog(frame,"Empty Fields","Checking Input
Value",JOptionPane.INFORMATION_MESSAGE);
```

```
        }
        else{
              int conVac=getCallVacancyNum();
              String des=getCallDesignation();
              String jobType= (partTimeStaffHire.isSelected()) ? "Part Time" :  "Full Time";
              int conSalary=getCallSalary();
              int wHour= getCallWorkingHour();

              if(!fullTimeStaffHire.isSelected())
              {
                   JOptionPane.showMessageDialog(null,"The job type you selected isnot
 valid!","Checking radio button Value",JOptionPane.INFORMATION_MESSAGE);

              }
              else
              {

              //creating object
              FullTimeStaffHire addFullTimeStaff=new  FullTimeStaffHire( conVac,  des,
 jobType, conSalary, wHour);
              //Adding the object to the arraylist
              list.add(addFullTimeStaff);
              JOptionPane.showMessageDialog(frame,"The Full Time Staff is successfully
 added","Information",JOptionPane.INFORMATION_MESSAGE);
              this.banner=true;
              }
           }
        }



        catch(Exception e1)
        {
            //Fetching the message
            JOptionPane.showMessageDialog(frame,e1.getMessage(),"ERROR
MESSAGE",JOptionPane.ERROR_MESSAGE);
            this.banner=false;
        }
    }
```

8.4Appointing FullTimeStaff

```
    public void appointFullTimeStaff(){
        try
        {
            String sName=getCallStaffName();
            String qualified =getCallQualification();
            String appBy=getCallAppointedBy();
            String JoiningDate=getCallJoiningDate();

            if (vacancyNumber.getText().equals("")|| staffName.getText().equals("") ||
qualification.getText().equals("") || joiningDateYear.getSelectedItem()=="Year" ||
joiningDateMonth.getSelectedItem()=="Month" ||
joiningDateDay.getSelectedItem()=="Day" || appointedBy.getText().equals("") )
            {
                JOptionPane.showMessageDialog(frame,"Please enter all the required
fields.","Error",JOptionPane.ERROR_MESSAGE);
            }

            else
            {
                int conVac=getCallVacancyNum();
                for(StaffHire vNo:list){
```

```
        if (vNo.getVacancyNumber()==conVac )
        {
          if(vNo instanceof FullTimeStaffHire)
          {
            FullTimeStaffHire addFullTimeStaff=(FullTimeStaffHire)vNo;
            if(addFullTimeStaff.joined()==false)
            {
              addFullTimeStaff.hireFullTimeStaff(sName , JoiningDate, qualified,
appBy);
              JOptionPane.showMessageDialog(frame,"Congratulations!! the full
time staff has been appointed
sucessfully!","Message",JOptionPane.INFORMATION_MESSAGE);

            }
            else
            {
             JOptionPane.showMessageDialog(frame,"The full time staff having
name "+getCallStaffName()+" has already been
appointed!","Message",JOptionPane.INFORMATION_MESSAGE);
            }
          }
          else
          {
            JOptionPane.showMessageDialog(frame,"The vacancy No: "
+getCallVacancyNum()+" doesn't belong to Full time staff
!","Error",JOptionPane.INFORMATION_MESSAGE);
          }
        }
        else
        {
                JOptionPane.showMessageDialog(frame,"The vacancy No: " +getCallVacancyNum()+" doesn't belong to Full time staff!","Error out",JOptionPane.INFORMATION_MESSAGE);
            }
        }
    }

    }
    catch(NumberFormatException e)
    {
        JOptionPane.showMessageDialog(frame,"Data type for VacancyNumber is numeric or integer!","Error",JOptionPane.INFORMATION_MESSAGE);
    }
    catch(NullPointerException e)
    {
        JOptionPane.showMessageDialog(frame,"Do not leave the fields blank!","Error",JOptionPane.INFORMATION_MESSAGE);
    }

}
```

8.5Appointing PartTimestaff

```
  public void appointPartTimeStaff()
  {
    try
    {
        String sName=getCallStaffName();
        String JoiningDate=getCallJoiningDate();
        String qualified =getCallQualification();
        String appBy=getCallAppointedBy();
        if((vacancyNumber.getText().equals("")||        staffName.getText().equals("")    ||
qualification.getText().equals("")    ||    joiningDateYear.getSelectedItem()=="Year"    ||
joiningDateMonth.getSelectedItem()=="Month"                                                ||
joiningDateDay.getSelectedItem()=="Day" || appointedBy.getText().equals("") ))
        {
           JOptionPane.showMessageDialog(frame,"Please   enter   all   the   required
fields.","Error",JOptionPane.ERROR_MESSAGE);
        }



        else
        {
           int conVac=getCallVacancyNum();
           for(StaffHire vNo:list)
           if (vNo.getVacancyNumber()==conVac )
           {
              if(vNo instanceof PartTimeStaffHire)
              {
                 PartTimeStaffHire addPartTimeStaff=(PartTimeStaffHire)vNo;
                 if(addPartTimeStaff.getJoined()==false)
                 {

addPartTimeStaff.hirePartTimeStaff(sName,JoiningDate,qualified,appBy);
                    JOptionPane.showMessageDialog(frame,"Congratulations!!  the  Part
Time Staff " +addPartTimeStaff.getStaffName()+ " having ID: " +getCallVacancyNum()+
"                has                been                appointed
sucessfully","Message",JOptionPane.INFORMATION_MESSAGE);
                 }
                 else
```

```
                    {
                     JOptionPane.showMessageDialog(frame,"The Part Time Staff having name " +addPartTimeStaff.getStaffName()+ " has already been appointed!","Message",JOptionPane.INFORMATION_MESSAGE);
                    }
                 }
                 else
                 {
                    JOptionPane.showMessageDialog(frame,"The vacancy No: " +getCallVacancyNum()+" doesn't belong to Part Time Staff!","Error",JOptionPane.INFORMATION_MESSAGE);
                 }
              }
              else
              {
                 JOptionPane.showMessageDialog(frame,"The vacancy No: " +getCallVacancyNum()+" doesn't belong to Part Time staff!","Error",JOptionPane.INFORMATION_MESSAGE);
              }
           }

        }
        catch(NumberFormatException e)
        {
           JOptionPane.showMessageDialog(frame,"Only numeric values can be used to add ID and Advance Salary of senior developer!","Error",JOptionPane.INFORMATION_MESSAGE);
        }
        catch(NullPointerException e)
        {
```

```
        JOptionPane.showMessageDialog(frame,"Please enter the value!","Error",JOptionPane.INFORMATION_MESSAGE);
      }

   }
```

8.6 TerminatePartTimeStaff

```
   public void terminatePartTimeStaff()
   {
      try
      {
         if((vacancyNumber.getText().equals("")|| staffName.getText().equals("") || qualification.getText().equals("") || joiningDateYear.getSelectedItem()=="Year" || joiningDateMonth.getSelectedItem()=="Month" || joiningDateDay.getSelectedItem()=="Day" || appointedBy.getText().equals("") ))
         {
            JOptionPane.showMessageDialog(frame,"Please enter all the required fields.","Error",JOptionPane.ERROR_MESSAGE);
         }

         else
         {
            int conVac=getCallVacancyNum();
            for(StaffHire vNo:list)
            if (vNo.getVacancyNumber()==conVac)
            {
               if(vNo instanceof PartTimeStaffHire)
               {
                  PartTimeStaffHire addPartTimeStaff=(PartTimeStaffHire)vNo;
                  if (addPartTimeStaff.getTerminated()==true)
                  {
```

```
                JOptionPane.showMessageDialog(frame,"The Part Time Staff having
name " +getCallStaffName()+ " has already been
terminated!","Error",JOptionPane.INFORMATION_MESSAGE);
            }
            else
            {
                if( addPartTimeStaff.getJoined()==false)
                {
                    JOptionPane.showMessageDialog(frame,"The Part Time Staff
hasnot been appointed yet","Message",JOptionPane.INFORMATION_MESSAGE);

                }



                else if(addPartTimeStaff.getTerminated()==false)
                {
                    //addPartTimeStaff.terminateStaff();
                    JOptionPane.showMessageDialog(frame,"The Part Time Staff
having name "/*+addPartTimeStaff.getStaffName()+*/+getCallStaffName()+" having ID:
"+getCallVacancyNum()+ " has been terminated
sucessfully","Message",JOptionPane.INFORMATION_MESSAGE);
                }
                else
                {
                    JOptionPane.showMessageDialog(frame,"The Part Time Staff
having name " +getCallStaffName()+ " has already been
terminated!","Error",JOptionPane.INFORMATION_MESSAGE);
                }
            }
        }
        else
```

```
                {
                    JOptionPane.showMessageDialog(frame,"The Vacancy No: " +getCallVacancyNum()+ " doesn't belong to Part Time Staff!","Error",JOptionPane.INFORMATION_MESSAGE);
                }
            }
            else
            {
                JOptionPane.showMessageDialog(frame,"The Vacancy No: " +getCallVacancyNum()+ " doesn't belong to Part Time Staff!","Error",JOptionPane.INFORMATION_MESSAGE);
            }
        }


    }
    catch(NumberFormatException e)
    {
        JOptionPane.showMessageDialog(frame,"Data type for DevNo is integer!!","Error",JOptionPane.INFORMATION_MESSAGE);
    }
    catch(NullPointerException e)
    {
        JOptionPane.showMessageDialog(frame,"Please enter the value!","Error",JOptionPane.INFORMATION_MESSAGE);
    }
}
public void display()
 {  //checking if the arraylist is empty
  if(list.size()==0)
  {
        JOptionPane.showMessageDialog(frame,"No staff are added so nothing to
display","Information!!!",JOptionPane.INFORMATION_MESSAGE);
    }

    else
    {//using for loop
        for(int i=0;i<list.size();i++)
        {
            if(list.get(i) instanceof FullTimeStaffHire)
            {   //checking if the index is index of Full Time Staff
                FullTimeStaffHire addFullTimeStaff=(FullTimeStaffHire) list.get(i);
                System.out.println("Information of Full Time Staff");
                addFullTimeStaff.displayInfo();
                System.out.println();
              }

              else
              {   //checking if the index is index of Part Time Staff
                  PartTimeStaffHire addPartTimeStaff=(PartTimeStaffHire) list.get(i);
                  System.out.println("Information of Part Time Staff");
                  addPartTimeStaff.displayInfo();
                  System.out.println();
              }
          }
    }
  }
  public void appointPartTimeStaffHire()
  {

     framePart = new JFrame("Part Time Staff Hiring");
     framePart.setSize(700,330);
```

```
p1=new JPanel();
p1.setLayout(null);
p1.setBounds(10,10,656,275);
p1.setVisible(true);
framePart.add(p1);

title=new JLabel("Appoint Part Time Staff");
title.setBounds(250,5,300,30);
title.setFont(new Font("Arial",Font.BOLD,15));
p1.add(title);

VacancyNumber = new JLabel("Vacancy Number: ");
VacancyNumber.setBounds(20,40,150,20);
VacancyNumber.setFont(new Font("Arial",Font.BOLD,15));
p1.add(VacancyNumber);

StaffName=new JLabel("Staff Name:");
StaffName.setBounds(350,40,130,20);
StaffName.setFont(new Font("Arial",Font.BOLD,15));
p1.add(StaffName);

Qualification=new JLabel("Qualification:");
Qualification.setBounds(20,70,100,20);
Qualification.setFont(new Font("Arial",Font.BOLD,15));
p1.add(Qualification);

JoiningDate=new JLabel("Joining Date:");
JoiningDate.setBounds(350,70,130,20);
JoiningDate.setFont(new Font("Arial",Font.BOLD,15));
p1.add(JoiningDate);
```

```
    AppointedBy=new JLabel("Appointed By:");
    AppointedBy.setBounds(20,100,130,20);
    AppointedBy.setFont(new Font("Arial",Font.BOLD,15));
    p1.add(AppointedBy);

    vacancyNumber= new JTextField();
    vacancyNumber .setBounds(150,40,130,20);
    p1.add(vacancyNumber );

    staffName=new JTextField();
    staffName.setBounds(450,40,195,20);
    p1.add(staffName);

    qualification=new JTextField();
    qualification.setBounds(130,70,150,20);
    p1.add(qualification);

    joiningDateYear=new JComboBox();
    joiningDateYear.addItem("Year");
    joiningDateYear.addItem(2015);
    joiningDateYear.addItem(2016);
    joiningDateYear.addItem(2017);
    joiningDateYear.addItem(2018);
    joiningDateYear.addItem(2019);
    joiningDateYear.addItem(2020);
    joiningDateYear.addItem(2021);
    joiningDateYear.addItem(2022);
    joiningDateYear.addItem(2023);
    joiningDateYear.addItem(2024);
    joiningDateYear.addItem(2025);
    joiningDateYear.addItem(2026);
```

```
joiningDateYear.addItem(2027);
joiningDateYear.addItem(2028);
joiningDateYear.addItem(2029);
joiningDateYear.addItem(2030);
joiningDateYear.addItem(2031);
joiningDateYear.addItem(2032);
joiningDateYear.addItem(2033);
joiningDateYear.addItem(2034);
joiningDateYear.addItem(2035);
joiningDateYear.addItem(2036);
joiningDateYear.addItem(2037);
joiningDateYear.addItem(2038);
joiningDateYear.addItem(2039);
joiningDateYear.addItem(2040);
joiningDateYear.addItem(2041);
joiningDateYear.addItem(2042);
joiningDateYear.addItem(2043);
joiningDateYear.addItem(2044);
joiningDateYear.addItem(2045);
joiningDateYear.addItem(2046);
joiningDateYear.addItem(2047);
joiningDateYear.addItem(2048);
joiningDateYear.addItem(2049);
joiningDateYear.addItem(2050);
joiningDateYear.setBounds(450,70,65,20);
p1.add(joiningDateYear);

joiningDateMonth=new JComboBox();
joiningDateMonth.addItem("Month");
joiningDateMonth.addItem("Jan");
joiningDateMonth.addItem("Feb");
joiningDateMonth.addItem("Mar");
joiningDateMonth.addItem("April");
joiningDateMonth.addItem("May");
joiningDateMonth.addItem("June");
joiningDateMonth.addItem("July");
joiningDateMonth.addItem("Aug");
joiningDateMonth.addItem("Sep");
joiningDateMonth.addItem("Oct");
joiningDateMonth.addItem("Nov");
joiningDateMonth.addItem("Dec");
joiningDateMonth.setBounds(520,70,70,20);
p1.add(joiningDateMonth);

joiningDateDay=new JComboBox();
joiningDateDay.addItem("Day");
joiningDateDay.addItem(1);
joiningDateDay.addItem(2);
joiningDateDay.addItem(3);
joiningDateDay.addItem(4);
joiningDateDay.addItem(5);
joiningDateDay.addItem(6);
joiningDateDay.addItem(7);
joiningDateDay.addItem(8);
joiningDateDay.addItem(9);
joiningDateDay.addItem(10);
joiningDateDay.addItem(11);
joiningDateDay.addItem(12);
joiningDateDay.addItem(13);
joiningDateDay.addItem(14);
joiningDateDay.addItem(15);
joiningDateDay.addItem(16);
joiningDateDay.addItem(17);
joiningDateDay.addItem(18);
joiningDateDay.addItem(19);
joiningDateDay.addItem(20);
joiningDateDay.addItem(21);
joiningDateDay.addItem(22);
joiningDateDay.addItem(23);
joiningDateDay.addItem(24);
joiningDateDay.addItem(25);
joiningDateDay.addItem(26);
joiningDateDay.addItem(27);
joiningDateDay.addItem(28);
joiningDateDay.addItem(29);
joiningDateDay.addItem(30);
joiningDateDay.addItem(31);
joiningDateDay.addItem(32);
joiningDateDay.setBounds(595,70,50,20);
p1.add(joiningDateDay);

appointedBy=new JTextField();
appointedBy.setBounds(130,100,150,20);
p1.add(appointedBy);

Terminate=new JButton("Terminate");
Terminate.setBounds(445,130,200,50);
Terminate.setActionCommand("Terminate");
Terminate.addActionListener(this);
p1.add(Terminate);

AppointPartTimeStaffHire=new JButton("Appoint Part");
AppointPartTimeStaffHire.setBounds(230,130,200,50);
```

```
        AppointPartTimeStaffHire.setActionCommand("Appoint Part");
        AppointPartTimeStaffHire.addActionListener(this);
        p1.add(AppointPartTimeStaffHire);

        Display=new JButton("Display Part Time Staff");
        Display.setBounds(20,190,210,50);
        Display.setActionCommand("Display Part Time Staff");
        Display.addActionListener(this);
        p1.add(Display);

        Clear=new JButton("Clear");
        Clear.setBounds(445,190,200,50);
        Clear.setActionCommand("Clear");
        Clear.addActionListener(this);
        p1.add(Clear);

        ExitPart=new JButton("Exit");
        ExitPart.setBounds(585,247,60,20);
        ExitPart.setActionCommand("Exit");
        ExitPart.addActionListener(this);
        p1.add(ExitPart);

        framePart.setLocationRelativeTo(null);
        framePart.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        framePart.setLayout(null);
        framePart.setVisible(true);

    }

    public void appointFullTimeStaffHire()
    {
```

```
frameFull = new JFrame("Full Time Staff Hiring");
frameFull.setSize(700,330);
p1=new JPanel();
p1.setLayout(null);
p1.setBounds(10,10,656,275);
p1.setVisible(true);
frameFull.add(p1);

title=new JLabel("Appoint Full Time Staff");
title.setBounds(250,5,300,30);
title.setFont(new Font("Arial",Font.BOLD,15));
p1.add(title);

VacancyNumber = new JLabel("Vacancy Number: ");
VacancyNumber.setBounds(20,40,150,20);
VacancyNumber.setFont(new Font("Arial",Font.BOLD,15));
p1.add(VacancyNumber);

StaffName=new JLabel("Staff Name:");
StaffName.setBounds(350,40,130,20);
StaffName.setFont(new Font("Arial",Font.BOLD,15));
p1.add(StaffName);

Qualification=new JLabel("Qualification:");
Qualification.setBounds(20,70,100,20);
Qualification.setFont(new Font("Arial",Font.BOLD,15));
p1.add(Qualification);

JoiningDate=new JLabel("Joining Date:");
JoiningDate.setBounds(350,70,130,20);
JoiningDate.setFont(new Font("Arial",Font.BOLD,15));
```

```
p1.add(JoiningDate);

AppointedBy=new JLabel("Appointed By:");
AppointedBy.setBounds(20,100,130,20);
AppointedBy.setFont(new Font("Arial",Font.BOLD,15));
p1.add(AppointedBy);

vacancyNumber= new JTextField();
vacancyNumber .setBounds(150,40,130,20);
p1.add(vacancyNumber );

staffName=new JTextField();
staffName.setBounds(450,40,195,20);
p1.add(staffName);

qualification=new JTextField();
qualification.setBounds(130,70,150,20);
p1.add(qualification);

joiningDateYear=new JComboBox();
joiningDateYear.addItem("Year");
joiningDateYear.addItem(2015);
joiningDateYear.addItem(2016);
joiningDateYear.addItem(2017);
joiningDateYear.addItem(2018);
joiningDateYear.addItem(2019);
joiningDateYear.addItem(2020);
joiningDateYear.addItem(2021);
joiningDateYear.addItem(2022);
joiningDateYear.addItem(2023);
joiningDateYear.addItem(2024);
```

```
        joiningDateYear.addItem(2025);
        joiningDateYear.addItem(2026);
        joiningDateYear.addItem(2027);
        joiningDateYear.addItem(2028);
        joiningDateYear.addItem(2029);
        joiningDateYear.addItem(2030);
        joiningDateYear.addItem(2031);
        joiningDateYear.addItem(2032);
        joiningDateYear.addItem(2033);
        joiningDateYear.addItem(2034);
        joiningDateYear.addItem(2035);
        joiningDateYear.addItem(2036);
        joiningDateYear.addItem(2037);
        joiningDateYear.addItem(2038);
        joiningDateYear.addItem(2039);
        joiningDateYear.addItem(2040);
        joiningDateYear.addItem(2041);
        joiningDateYear.addItem(2042);
        joiningDateYear.addItem(2043);
        joiningDateYear.addItem(2044);
        joiningDateYear.addItem(2045);
        joiningDateYear.addItem(2046);
        joiningDateYear.addItem(2047);
        joiningDateYear.addItem(2048);
        joiningDateYear.addItem(2049);
        joiningDateYear.addItem(2050);
        joiningDateYear.setBounds(450,70,65,20);
        p1.add(joiningDateYear);

        joiningDateMonth=new JComboBox();
        joiningDateMonth.addItem("Month");
        joiningDateMonth.addItem("Jan");
        joiningDateMonth.addItem("Feb");
        joiningDateMonth.addItem("Mar");
        joiningDateMonth.addItem("April");
        joiningDateMonth.addItem("May");
        joiningDateMonth.addItem("June");
        joiningDateMonth.addItem("July");
        joiningDateMonth.addItem("Aug");
        joiningDateMonth.addItem("Sep");
        joiningDateMonth.addItem("Oct");
        joiningDateMonth.addItem("Nov");
        joiningDateMonth.addItem("Dec");
        joiningDateMonth.setBounds(520,70,70,20);
        p1.add(joiningDateMonth);

        joiningDateDay=new JComboBox();
        joiningDateDay.addItem("Day");
        joiningDateDay.addItem(1);
        joiningDateDay.addItem(2);
        joiningDateDay.addItem(3);
        joiningDateDay.addItem(4);
        joiningDateDay.addItem(5);
        joiningDateDay.addItem(6);
        joiningDateDay.addItem(7);
        joiningDateDay.addItem(8);
        joiningDateDay.addItem(9);
        joiningDateDay.addItem(10);
        joiningDateDay.addItem(11);
        joiningDateDay.addItem(12);
        joiningDateDay.addItem(13);
        joiningDateDay.addItem(14);
```

```
    joiningDateDay.addItem(15);
    joiningDateDay.addItem(16);
    joiningDateDay.addItem(17);
    joiningDateDay.addItem(18);
    joiningDateDay.addItem(19);
    joiningDateDay.addItem(20);
    joiningDateDay.addItem(21);
    joiningDateDay.addItem(22);
    joiningDateDay.addItem(23);
    joiningDateDay.addItem(24);
    joiningDateDay.addItem(25);
    joiningDateDay.addItem(26);
    joiningDateDay.addItem(27);
    joiningDateDay.addItem(28);
    joiningDateDay.addItem(29);
    joiningDateDay.addItem(30);
    joiningDateDay.addItem(31);
    joiningDateDay.addItem(32);
    joiningDateDay.setBounds(595,70,50,20);
    p1.add(joiningDateDay);

    appointedBy=new JTextField();
    appointedBy.setBounds(130,100,150,20);
    p1.add(appointedBy);

    AppointPartTimeStaffHire=new JButton("Appoint Full");
    AppointPartTimeStaffHire.setBounds(445,130,200,50);
    AppointPartTimeStaffHire.setActionCommand("Appoint Full");
    AppointPartTimeStaffHire.addActionListener(this);
    p1.add(AppointPartTimeStaffHire);
```

```
        Display=new JButton("Display Full Time Staff");
        Display.setBounds(20,130,220,50);
        Display.setActionCommand("DDisplay Full Time Staff");
        Display.addActionListener(this);
        p1.add(Display);

        Clear=new JButton("Clear");
        Clear.setBounds(20,200,220,50);
        Clear.setActionCommand("Clear");
        Clear.addActionListener(this);
        p1.add(Clear);

        ExitFull=new JButton("Exit");
        ExitFull.setBounds(585,250,60,20);
        ExitFull.setActionCommand("Exit");
        ExitFull.addActionListener(this);
        p1.add(ExitFull);

        frameFull.setLocationRelativeTo(null);
        frameFull.setLayout(null);
        frameFull.setVisible(true);




    }
    public void clear()
    {  //clearing all the text fields
        vacancyNumber.setText("");
        designation.setText("");
        salary.setText("");
        appointedBy.setText("");
        workingHour.setSelectedItem("--Select--");
        staffName.setText("");
        qualification.setText("");
        joiningDateYear.setSelectedItem("Year");
        joiningDateMonth.setSelectedItem("Month");
        joiningDateDay.setSelectedItem("Day");
    }


}
```

9.References

HowToDoInjava.com. (2016). Retrieved from https://howtodoinjava.com/java/basics/what-is-java-programming-language/.

Java Point. (2018). Retrieved from https://www.javatpoint.com/java-swing.

Levy, S. (2020). Retrieved from https://www.britannica.com/biography/Ivan-Edward-Sutherland.

Rouse, M. (2005, April). Retrieved from https://whatis.techtarget.com/definition/pseudocode.

Visual Paradigm. (2020). Retrieved from https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/.