LONDON METROPOLITAN UNIVERSITY

ITAHARI INTERNATIONAL COLLEGE

Module Code & Module Title

CS4001NT Programming

Assessment Weightage & Type

50% Individual Coursework

Year and Semester

2019-20 Autumn / 2019-20 Spring

Student Name: Ashesh Rai

London Met ID: 19031902

College ID: NP05CP4A190031

Extended Assignment Due Date: 5th June, 2020

Assignment Submission Date: 5th June, 2020

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

Table of Contents

Table of Tables

Table of Figures

Introduction

Java is a general-purpose object-oriented programming language that is easy to code and reliable. Java is very similar to C and C++; java is a write-once, run anywhere programming language. The platform independence and secure environment are one of the few reasons for the popularity of Java. Around three billion devices run Java making it a very suitable programming language to pick for the development of software.

There are three main components of Java which are Java Development Kit (JDK), Java Virtual Machine (JVM), and Java Runtime Environment (JRE). JDK enables the developers to write their code and run the code after compilation in JRE through the JVM. The compiled source codes are called bytecodes and are executed in JVM that converts it to machine code. The JRE consists of libraries (packages) that JVM can use to run the programs. Through the usage of JVM, Java has eliminated the need to develop cross-platform programs (Statler, 2019).

The four main object-oriented principles of Java are Encapsulation, Abstraction, Inheritance, and Polymorphism. Abstraction is the process of hiding the real working mechanism of an object and giving out the only required information of the object in a comprehensible fashion to reduce the programming complexity. Encapsulation is binding of data and methods together; it is performed to achieve the data security by creating private variables and provide the property to access the private data. Inheritance is the process by which one Java class acquires the properties of another class; the main goal behind the inheritance is to reuse, enhance, and customize the existing code. Polymorphism is another pillar in the object-oriented principles of Java and it defines the ability of an object to take many forms; polymorphism is the ability to redefine methods from inherited classes and can behave in different forms (Ahmed, 2018).

Java Interface

Interface is a reference type in Java, it is a collection of abstract methods that have empty bodies and can also contain constants, default methods, static methods, and nested types. The interface keyword is used to declare an interface while implements keyword is used to implement the interface; writing up an interface is very similar to writing a class, a class construes attributes and behaviors of an object while an interface consists of behaviors that a class implements or overrides.

There are a few similarities between an interface and a class. Some of them are listed below:

- An interface can hold any number of methods.
- An interface has the same .java extension as a class.
- The compiled bytecode appears as a .class file.
- Interfaces appear in packages, and their corresponding bytecode file must be in a directory structure that matches the package name.

Differences between an interface and a class are:

- An interface cannot be instantiated.
- Interface does not have constructors.
- All the methods in the interface are abstract.
- An interface is implemented by a class rather than extended, and an interface can extend multiple interfaces whereas a class can only extend one class.
- Classes represent the real object. Interfaces allow you to create a program that will manipulate the class in a predefined way.

(Tutorialspoint, 2016)

Swing

Swing is a GUI framework that was launched by the Sun Microsystems for easy front-end development. It was developed to fulfill the need for a more powerful and flexible GUI framework than Abstract Window Toolkit (AWT). Swing components are easy to use and have much more built-in functionality than AWT. Swing is built upon its predecessor AWT API and is entirely written in Java. Swing has many advanced features like JTable, JTabbed pane which is not available in AWT. Also. Swing components are called lightweight because they do not require a native OS object to implement their functionality. Many of the swing components were used in this coursework assignment to create the desired front-end design.

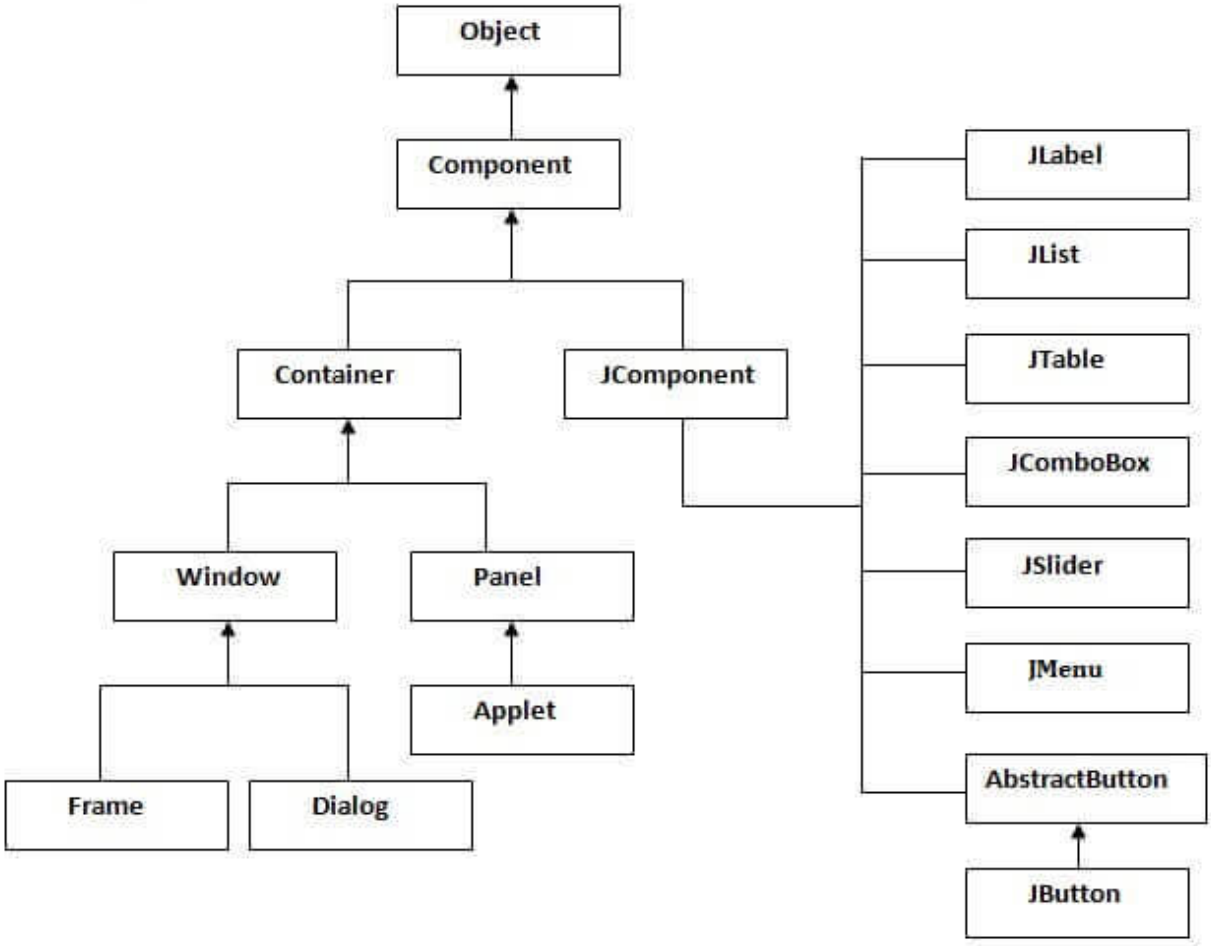

Figure 1: Hierarchy of Java Swing API

Abstract Window Toolkit (AWT)

AWT is the predecessor of Swing and is mostly phased out. It uses native GUI widgets so it is platform dependent meaning that the components are displayed according to the operating system. AWT is more labor-intensive than Swing and requires a lot of coding. The difference between AWT and Swing's component is the usage of J in the component, Swing's component use J before the component's name e.g. JLabel whereas AWT just uses Label. Although having lots of drawbacks than Swing it is still used mainly because of layout managers and its event and listener package. The layout manager is very essential in arranging the components in a particular manner whereas the event package is used to listen to the changing state of objects to perform the tasks accordingly.

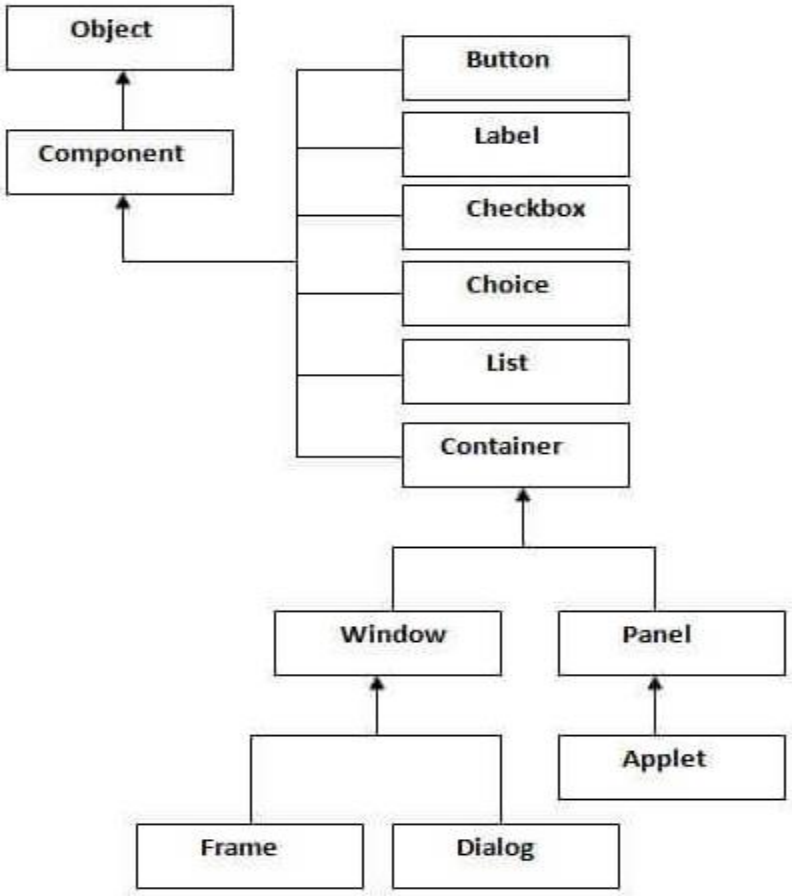

Figure 2: Hierarchy of Java AWT Classes

Class Diagram of INGNepal

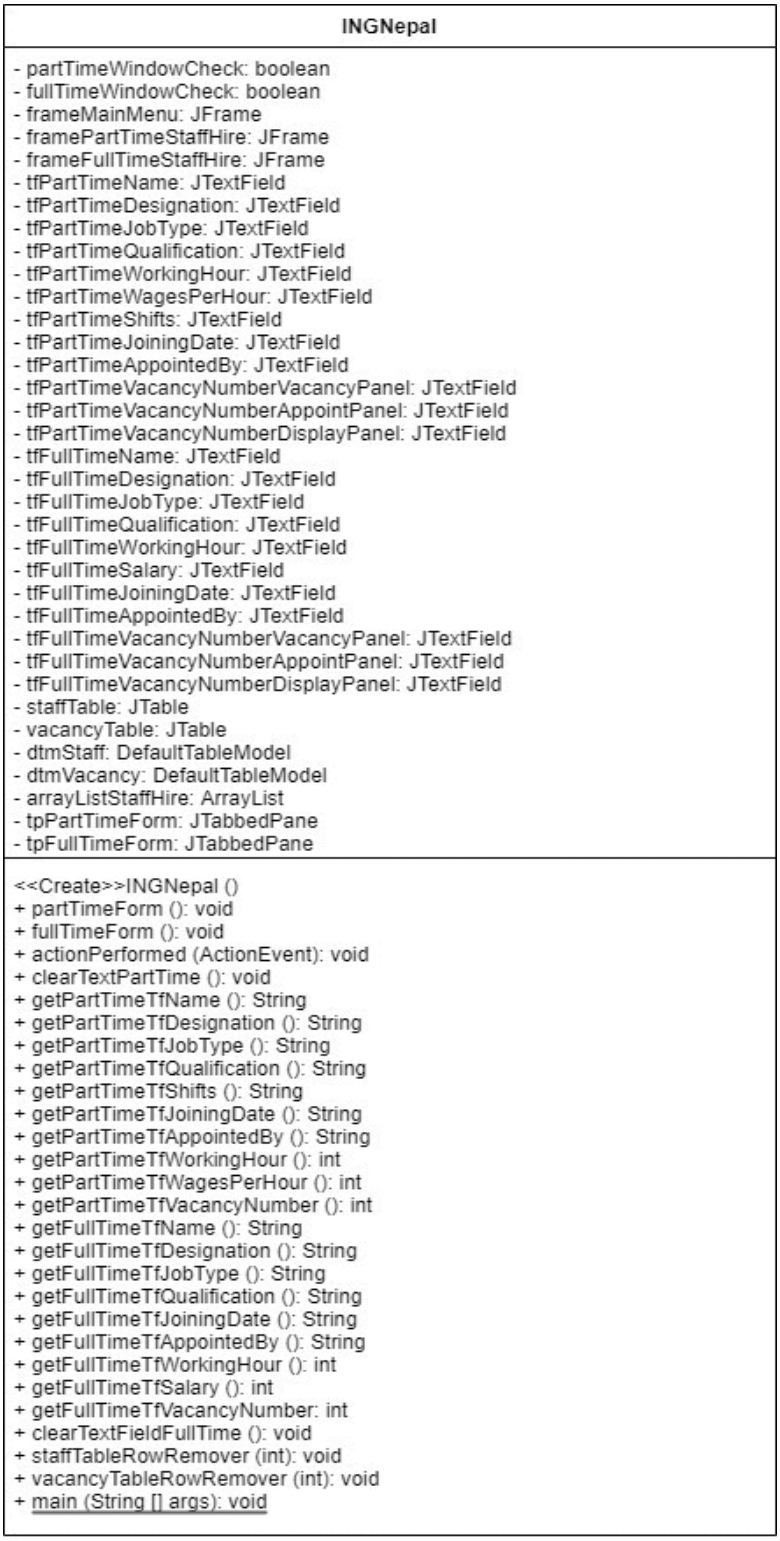

Figure 3: Class Diagram of INGNepal

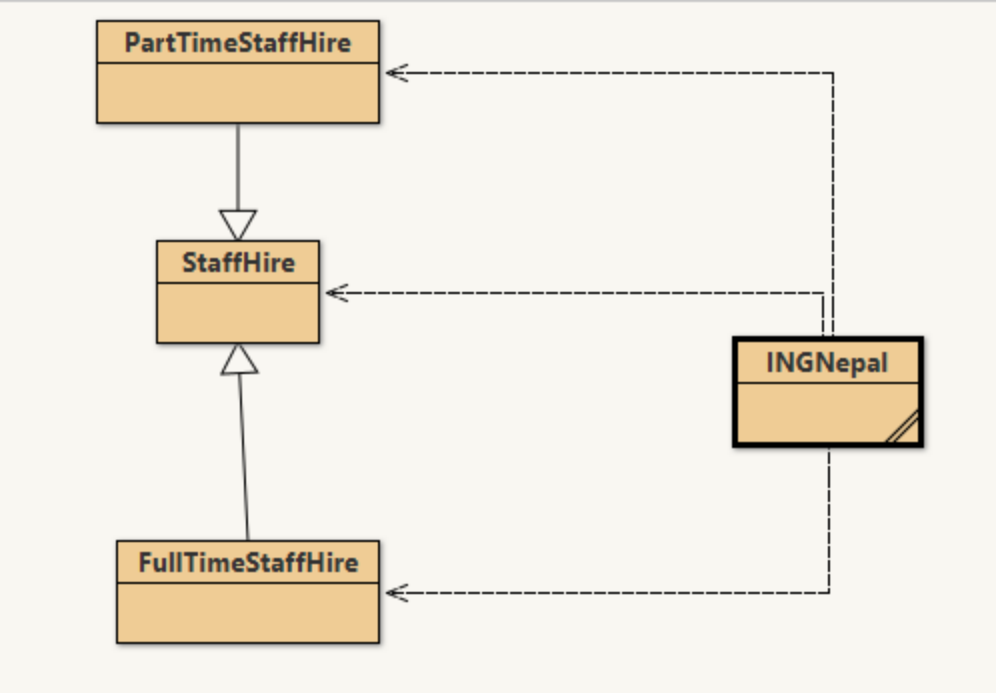

Figure 4: BlueJ Diagram

Pseudocode of INGNepal

initialize boolean partTimeWindowCheck and fullTimeWindowCheck to false

initialize JFrame frameMainMenu, framePartTimeStaffHire, and

frameFullTimeStaffHire

initialize JTextField tfPartTimeName, tfPartTimeDesignation, tfPartTimeJobType, tfPartTimeQualification, tfPartTimeWorkingHour,

tfPartTimeWagesPerHour, tfPartTimeShifts,

tfPartTimeJoiningDate,
tfPartTimeAppointedBy,tfPartTimeVacancyNumberVacancyPanel,
tfPartTimeVacancyNumberAppointPanel, tfPartTimeVacancyNumberDisplayPanel

initialize JTextField tfFullTimeName, tfFullTimeDesignation, tfFullTimeJobType, tfFullTimeQualification, tfFullTimeWorkingHour, tfFullTimeSalary, tfFullTimeJoiningDate, tfFullTimeAppointedBy,

tfFullTimeVacancyNumberVacancyPanel,

tfFullTimeVacancyNumberAppointPanel, tfFullTimeVacancyNumberDisplayPanel

initialize JTable staffTable, and vacancyTable

initialize DefaultTableModel dtmStaff, and dtmVacancy

initialize ArrayList arrayListStaffHire which takes instance of StaffHire as data type

initialize JTabbedPane tpPPartTimeForm, and tpFullTimeForm

function INGNepal ():

set the name of the JFrame frameMainMenu to ING Nepal

set the size of the JFrame frameMainMenu to 600x600

set the location of the JFrame frameMainMenu's location relative to null

set the default close operation of the JFrame frameMainMenu to nothing

set the resizable property of the JFrame frameMainMenu to true

add window listener to the frame for window closing event

if partTimeWindowCheck == true

request focus of framePartTimeStaffHire

set framePartTimeStaffHire's state to normal

else if fullTimeWindowCheck == true

request focus of frameFullTimeStaffHire

set frameFullTimeStaffHire's state to normal

else if partTimeWindowCheck == false and fullTimeWindowCheck == false

close frameMainMenu

end if

end listener

create an object of cursor class and set it to hand cursor

Initialize and **create** JMenuBar mb

Initialize and **create** JMenu menu with Menu as JMenu text

Initialize and **create** JMenuItem menuItemAbout with About as menuItemAbout's text

add action listener to menuItemAbout

    initialize and **create** JDialog jd

    set the properties of JDialog jd

    initialize and **create** JLabel l

    add JLabel l to the JDialog jd

    set JDialog jd to visible

    set JDialog jd's relative location null

end listener

initialize and **create** JMenuItem menuItemExit

add action listener to menuItemExit

    terminate the program

end listener

add JMenuItem menuItemAbout and menuItemExit to JMenu menu

add JMenu menu to JMenuBar mb

initialize JButton buttonPartTimeForm with Part Time Operations as button text

set JButton buttonPartTimeForm's cursor

add action listener to the JButton buttonPartTimeForm

    if partTimeWindowCheck == false and fullTimeWindowCheck == false

        call partTimeForm method

        set partTimeWindowCheck's value to true

    else if fullTimeWindowCheck == true

        request focus of JFrame frameFullTimeStaffHire

        set JFrame frameFullTimeStaffHire's state to normal

    else

        request focus of JFrame framePartTimeStaffHire

        set JFrame framePartTimeStaffHire's state to normal

    end if

end listener

Initialize and **create** JButton buttonFullTimeForm with Full Time Operations as button text

set JButton buttonFullTimeForm's cursor

add action listener to the JButton buttonFullTimeForm

    if fullTimeWindowCheck == false and partTimeWindowCheck == false

        call partTimeForm method

        set partTimeWindowCheck's value to true

    else if partTimeWindowCheck == true

        request focus of JFrame framePartTimeStaffHire

        set JFrame framePartTimeStaffHire's state to normal

    else

        request focus of JFrame frameFullTimeStaffHire

        set JFrame frameFullTimeStaffHire's state to normal

    end if

initialize and **create** JPanel mainMenuButtonPanel

set JPanel mainMenuButtonPanel's layout to flow layout

add JButton buttonPartTimeForm to JPanel mainMenuButtonPanel

add JButton buttonFullTimeForm to JPanel mainMenuButtonpanel

create JTable staffTable

create DefaultTableModel dtmStaff and **set** its column name

set the model of JTable staffTable to DefaulTableModel dtmStaff

create JScrollPane staffTableScrollPane

set corner of JScrollPane staffTableScrollPane to the upper right corner

set viewport view of JScrollPane staffTableScrollPane to the JTable staffTable

create JTable vacancyTable

create DefaultTableModel dtmVacancy and **set** its column name

set the model of JTable vacanccyTable to DefaulTableModel dtmVacancy

create JScrollPane vacancyTableScrollPane

set corner of JScrollPane vacancyTableScrollPane to the upper right corner

set viewport view of JScrollPane vacancyTableScrollPane to the JTable vacancyTable

create JPanel panelStaffTbl

set JPanel panelStaffTbl's layout to Border layout

add JScrollPane staffTableScrollPane to the JPanel panelStaffTbl and set the Border layout to the center

create JPanel panelVacancyTbl

set JPanel panelVacancyTbl's layout to Border layout

add JScrollPane vacancyTableScrollPane to the JPanel panelVacancyTbl and set the Border layout to the center

create JTabbedPane tpMain

add JPanel panelStaffTbl to the JTabbedPane tpMain and set the tab's name to Staffs

add JPanel panelVacancyTbl to the JTabbedPane tpMain and set the tab's name to Vacancy

add JTabbedPane tpMain to the JFrame frameMainMenu and set the position to the Border center

add JPanel mainMenuButtonpanel to the JFrame frameMainMenu and set the position to the Border south

add JMenuBar mb to the JFrame frameMainMenu and set the position to the Border north

end function

function partTimeForm ():

set the name of the JFrame framePartTimeStaffHire to Part Time Staff Hire

set the size of the JFrame framePartTimeStaffHire to 340x380

set the location of the JFrame framePartTimeStaffHire's location relative to null

set the default close operation of the JFrame framePartTimeStaffHire to dispose on close

set the resizable property of the JFrame framePartTimeStaffHire to false

set the layout of the JFrame framePartTimeStaffHire to Border layout

set JFrame framePartTimeStaffHire to be always on top

add window listener to the JFrame framePartTimeStaffHire

    set the partTimeWindowCheck to false

end listener

if partTimeWindowCheck false

    set the visibility of JFrame framePartTimeStaffHire to true

end if

create an object of cursor class and set it to hand cursor

initialize and **create** JLabel lblPartTimeName with Name as label text, lblPartTimeDesignation with Designation as label text,

lblPartTimeJobType with Job Type as label text,

lblPartTimeQualification with Qualification as label text,

lblPartTimeWorkingHour with Working Hour as label text,

lblPartTimeWagesPerHour with Wages Per Hour as label text,

lblPartTimeShifts with Shifts as label text,

lblPartTimeJoiningDate with Joining Date as label text,

lblPartTimeAppointedBy with Appointed By as label text,

lblPartTimeVacancyNumberVacancyPanel with Vacancy Number as label text, lblPartTimeVacancyNumberAppointPanel with Vacancy Number as label text, lblPartTimeVacancyNumberDisplayPanel with Vacancy Number as label text

create JTextField tfPartTimeName, tfPartTimeDesignation, tfPartTimeJobType, tfPartTimeQualification, tfPartTimeWorkingHour, tfPartTimeWagesPerHour, tfPartTimeShifts, tfPartTimeJoiningDate, tfPartTimeAppointedBy, tfPartTimeVacancyNumberVacancyPanel, tfPartTimeVacancyNumberAppointPanel, tfPartTimeVacancyNumberDisplayPanel

initialize and **create** JButton partTimeAddVacancyButton with Add Vacancy as button text

initialize and **create** JButton partTimeClearButtonVacancyPanel with Clear as button text

initialize and **create** JButton partTimeClearButtonAppointPanel with Clear as button text

initialize and **create** JButton partTimeClearButtonDisplayPanel with Clear as button text

initialize and **create** JButton partTimeDisplayButton with Display as button text

initialize and **create** JButton partTimeAppointButton with Appoint as button text

initialize and **create** JButton partTimeBackButtonVacancyPanel with Back as button text

initialize and **create** JButton partTimeBackButtonAppointPanel with Back as button text

initialize and **create** JButton partTimeBackButtonDisplayPanel with Back as button text

initialize and **create** JButton partTimeTerminateButton with Terminate as button text

initialize and **create** JPanel vacancyPanel

set the layout JPanel vacancyPanel to null

set Bounds for JLabel lblPartTimeVacancyNumberVacancyPanel and JTextField tfPartTimeVacancyNumberVacancyPanel

set Bounds for JLabel lblPartTimeDesignation and JTextField tfPartTimeDesignation

set Bounds for JLabel lblPartTimeJobType and JTextField tfPartTimeJobType

set Bounds for JLabel lblPartTimeWorkingHour and JTextField tfPartTimeWorkingHour

set Bounds for JLabel lblPartTimeWagesPerHour and JTextField tfPartTimeWagesPerHour

set Bounds for JLabel lblPartTimeShifts and JTextField tfPartTimeShifts

set Bounds for JButton partTimeBackButtonVacancyPanel

set cursor of JButton partTimeBackButtonVacancyPanel to the hand cursor using instance of cursor object

add action listener to the JButton partTimeBackButtonVacancyPanel

dispose framePartTimeStaffHire

set partTimeWindowCheck to false

end listener

set Bounds for JButton partTimeAddVacancyButton

set cursor of JButton partTimeAddVacancyButton to the hand cursor using instance of cursor object

add action listener to the JButton partTimeAddVacancyButton

initialize integer vacancyNumber and **set** its value by calling getPartTimeTfVacancyNumber method

initialize String designation and **set** its value by calling getPartTimeTfDesignation method

initialize String jobType and **set** its value by calling getPartTimeTfJobType method

initialize integer workingHour and **set** its value by calling getPartTimeTfWorkingHour method

initialize integer wagesPerHour and **set** its value by calling getPartTimeTfWagesPerHour method

initialize String shifts and set its value by calling getPartTimeTfShifts method

if vacancyNumber != 0 and workingHour != 0 and wagesPerHour != 0 and designation not empty and jobType not empty and shifts not empty

initialize integer size and **set** its value by calling the size method of ArrayList for arrayListStaffHire

if size > 0

for StaffHire sh : arrayListStaffHire

initialize integer num and get the vacancy number using getVacancyNumber method of StaffHire class

if num == vacancyNumber

show error message “Vacancy already exists”

return

end if

end for loop

call PartTimeStaffHire’s constructor and pass vacancyNumber, designation, jobType, workingHour, wagesPerHour, and shifts

add the PartTimeStaffHire’s instance to the ArrayList arrayListStaffHire

call clearTextPartTime method

else

call PartTimeStaffHire’s constructor and pass vacancyNumber, designation, jobType, workingHour, wagesPerHour, and shifts

add the PartTimeStaffHire's instance to the ArrayList arrayListStaffHire

call clearTextPartTime method

end if

downcast DefaultTableModel modelVacancyTable into DefaultTableModel vacancyTable and use getModel method

add row to the modelVacancyTable by creating a new object and adding the integer vacancyNumber in the vacancy number column and Part Time to the employment type column

else if designation is empty and jobType is empty and shifts is empty

show message "Designation, job type, and shifts are empty"

else if designation is empty and jobType is empty

show message "Designation, and job type are empty"

else if designation is empty and shifts is empty

show message "Designation, and shifts are empty"

else if shifts is empty and jobType is empty

show message "Shifts, and job type are empty"

else if designation is empty

show message "Designation is empty"

else if jobType is empty

show message "Job type is empty"

else

show message "Shifts is empty"

end if

end listener

set Bounds for JButton partTimeClearButtonVacancyPanel

set cursor of JButton partTimeClearButtonVacancypanel to the hand cursor using instance of cursor object

add action listener to the JButton partTimeClearButtonVacancyPanel

call clearTextPartTime method

add JLabel lblPartTimeVacancyNumberVacancyPanel, lblPartTimeDesignation, lblPartTimeJobType, lblPartTimeWorkingHour, lblPartTimeWagesPerHour, lblPartTimeShifts to the JPanel vacancyPanel

add JTextField tfPartTimeVacancyNumberVacancyPanel, tfPartTimeDesignation, tfPartTimeJobType, tfPartTimeWorkingHour, tfPartTimeWagesPerHour, tfPartTimeShifts to the JPanel vacancyPanel

add JButton partTimeBackButtonVacancyPanel, partTimeAddVacancyButton, and partTimeClearButtonVacancyPanel to the JPanel vacancyPanel

initialize and **create** JPanel appointPanel

set JPanel appointPanel's layout to null

set Bounds for JLabel lblPartTimeVacancyNumberAppointPanel and JTextField tfPartTimeVacancyNumberAppointPanel

set Bounds for JLabel lblPartTimeName and JTextField tfPartTimeName

set Bounds for JLabel lblPartTimeJoiningDate and JTextField tfPartTimeJoiningDate

set Bounds for JLabel lblPartTimeQualification and JTextField tfPartTimeQualification

set Bounds for JLabel lblPartTimeAppointedBy and JTextField tfPartTimeAppointedBy

set Bounds for JButton partTimeBackButtonAppointPanel

set cursor of JButton partTimeBackButtonAppointPanel to the hand cursor using instance of cursor object

add action listener to the JButton partTimeBackButtonAppointPanel

    dispose framePartTimeStaffHire

    set partTimeWindowCheck to false

end listener

set Bounds for JButton partTimeAppointButton

set cursor of JButton partTimeAppointButton to the hand cursor using instance of cursor object

add action listener to the JButton partTimeAppointButton

initialize String name and **set** it using getPartTimeTfName method

initialize String joiningDate and **set** it using getPartTimeTfJoiningDate method

initialize String qualification and **set** it using getPartTimeTfQualification method

initialize String appointedBy and **set** it using getPartTimeTfAppointedBy method

initialize integer vacancyNumber and **set** it using getPartTimeTfVacancyNumber method

initialize integer size **set** it using the size method for ArrayList on arrayListStaffHire

if vacancyNumber != 0 and joiningDate is not empty and qualification is not empty and name is not empty and appointedBy is not empty

if size > 0

for i = 0, i < size, ++i

get the arrayListStaffHire index using for loop and store it in StaffHire class's instance sh

if sh is instance of PartTimeStaffHire

downcast PartTimeStaffHire p to PartTimeStaffHire sh

initialize integer vacancyNo and set it using StaffHire class's getVacancyNumber method

initialize boolean joinCheck and set it using PartTimeStaffHire class's getJoined method

if vacancyNo == vacancyNumber and joinCheck == false

> **call** PartTimeStaffHire's hirePartTimeStaff method which takes name, joiningDate, qualification, appointedBy as parameter to hire the staff
>
> **downcast** DefaultTableModel modelStaffTable into DefaultTableModel staffTable and use getModel method
>
> **add** row to the modelStaffTable by creating a new object and adding the integer vacancyNumber in the vacancy number column, name in the name column, and Part Time to the employment type column
>
> **call** vacancyTableRowRemover and pass vacancyNumber as parameter
>
> **call** clearTextPartTime method
>
> **break**

else if vacancyNo == vacancyNumber and joinCheck == true

show error message “Vacancy number already filled”

break

else if (size - 1) == I and vacancyNo != vacancyNumber

show error message “Vacancy Number not found”

end if

end if

end for loop

end if

else if size == 0

show message “No vacancy available**”**

else if joiningDate is empty and qualification is empty and name is empty and appointedBy is empty

show message “Joining date, qualification, name, and appointed by are empty”

else if joiningDate is empty and qualification is empty and name is empty

show message “Joining date, qualification, and name are empty”

else if joiningDate and qualification and appointedBy is empty

show message “Joining date, qualification, and appointed by are empty”

else if qualification is empty and name is empty and appointedBy is empty

show message “Qualification, name, and appointed by are empty”

else if joiningDate is empty and qualificationis empty

show message “Joining date, and qualification are empty”

else if joiningDate is empty and name is empty

show message “Joining date, and name are empty”

else if joiningDate is empty and appointedBy is empty

show message “Joining date, and appointed by are empty”

else if qualification is empty and name is empty

show message “Qualification, and name are empty”

else if qualification is empty and appointedBy is empty

show message “Qualification, and appointed by are empty”

else if name is empty and appointed by is empty

show message “Name, and appointed by are empty”

else if joiningDate is empty

show message “Joining date is empty”

else if qualification is empty

show message “Qualification is empty”

else if name is empty

show message “Name is empty”

else

show message “Appointed by is empty”

end if

end listener

set Bounds of JButton partTimeClearButtonAppointPanel

set cursor of JButton partTimeClearButtonAppointPanel to the hand cursor using instance of cursor object

add action listener to JButton partTimeClearButtonAppointPanel

call clearTextPartTime method

end listener

add JLabel lblPartTimeVacancyNumberAppointPanel, lblPartTimeName, lblPartTimeJoiningDate, lblPartTimeQualification, lblPartTimeAppointedBy to the JPanel appointPanel

add JTextField tfPartTimeVacancyNumberAppointPanel, tfPartTimeName, tfPartTimeJoiningDate, tfPartTimeQualification, tfPartTimeAppointedBy to the JPanel appointPanel

add JButton partTimeBackButtonAppointPanel, partTimeAppointButton, and partTimeClearButtonAppointPanel to the JPanel appointPanel

initialize and **create** JPanel displayPanel

set JPanel displayPanel's layout to null

set Bounds for JLabel lblPartTimeVacancyNumberDisplayPanel and JTextField tfPartTimeVacancyNumberDisplayPanel

set Bounds for JButton partTimeBackButtonDisplayPanel

set cursor of JButton partTimeBackButtonDisplayPanel to the hand cursor using instance of cursor object

add action listener to JButton partTimeBackButtonDisplayPanel

dispose JFrame framePartTimeStaffHire

set value of partTimeWindowCheck to false

end listener

set Bounds for JButton partTimeClearButtonDisplayPanel

set cursor of JButton partTimeClearButtonDisplayPanel to the hand cursor using instance of cursor object

add action listener to JButton partTimeClearButtonDisplayPanel

call clearTextPartTime method

set Bounds for JButton partTimeDisplayButton

set cursor of JButton partTimeDisplayButton to the hand cursor using instance of cursor object

add action listener to JButton partTimeDisplayButton

initialize integer vacancyNo and **set** its value by calling getPartTimeTfVacancyNumber method

initialize boolean found and **set** its value to false

if vacancyNo != 0

for StaffHire s : arrayListStaffHire

if s is instance of PartTimeStaffHire

downcast PartTimeStaffHire p to PartTimeStaffHire s

if instance StaffHire returns vacancyNumber that is equal to vacancyNo

call displayDetails method of PartTimeStaffHire class

set found's value to true

show info message with staff's details

call clearTextPartTime method

break

end if

end if

end for

if found's value not true

show error message "Vacancy number not found"

end if

end if

end listener

set Bounds for partTimeTerminateButton

set cursor of JButton partTimeTerminateButton to the hand cursor using instance of cursor object

add action listener to the JButton partTimeTerminateButton

initialize integer vacancyNumber and **set** its value by calling getPartTimeTfVacancyNumber method

initialize boolean found and set it to false

if vacancyNumber != 0

for StaffHire s : arrayListStaffHire

if s is instance of PartTimeStaffHire

downcast PartTimeStaffHire p to PartTimeStaffHire s

if the vacancy number returned by the StaffHire class's getVacancyNumber method is exactly like vacancyNumber and the value returned by PartTimeStaffHire class's getTerminated method is false

call PartTimeStaffHire class's terminatePartTimeStaff method

remove the instance from ArrayList arrayListStaffHire

call staffTableRowRemover method and pass vacancyNumber as parameter

set found's value to true

show info message of termination

call clearTextPartTime method

break

end if

end if

end for

if found is false

show error message “Vacancy number not found”

end if

end if

end listener

add lblPartTimeVacancyNumberDisplayPanel to JPanel displayPanel

add tfPartTimeVacancyNumberDisplayPanel to JPanel displayPanel

add partTimeBackButtonDisplayPanel to JPanel displayPanel

add partTimeClearButtonDisplayPanel to JPanel displayPanel

add partTimeDisplayButton to JPanel displayPanel

add partTimeTerminateButton to JPanel displayPanel

create JTabbedPane tpPartTimeForm

add JPanel vacancyPanel to the JTabbedPane tpPartTimeForm and name the tab as Add Vacancy

add JPanel appointPanel to the JTabbedPane tpPartTimeForm and name the tab as Appoint

add JPanel displayPanel to the JTabbedPane tpPartTimeForm and name the tab as Display and terminate

add JTabbedPane tpPartTimeForm to the JFrame framePartTimeStaffHire with Border layout and position as center

end function

function fullTimeForm ():

set the name of the JFrame frameFullTimeStaffHire to Part Time Staff Hire

set the size of the JFrame frameFullTimeStaffHire to 340x360

set the location of the JFrame frameFullTimeStaffHire's location relative to null

set the default close operation of the JFrame frameFullTimeStaffHire to dispose on close

set the resizable property of the JFrame frameFullTimeStaffHire to false

set the layout of the JFrame frameFullTimeStaffHire to Border layout

set JFrame frameFullTimeStaffHire to be always on top

add window listener to the JFrame frameFullTimeStaffHire

    set the fullTimeWindowCheck to false

end listener

if fullTimeWindowCheck false

    set the visibility of JFrame framePartTimeStaffHire to true

end if

create an object of cursor class and set it to hand cursor

initialize and **create** JLabel lblFullTimeName with Name as label text, lblFullTimeDesignation with Designation as label text,

lblFullTimeJobType with Job Type as label text,

lblFullTimeQualification with Qualification as label text,

lblFullTimeWorkingHour with Working Hour as label text,

lblFullTimeSalary with Salary as label text,

lblFullTimeJoiningDate with Joining Date as label text,

lblFullTimeAppointedBy with Appointed By as label text,

lblFullTimeVacancyNumberVacancyPanel with Vacancy Number as label text, lblFullTimeVacancyNumberAppointPanel with Vacancy Number as label text, lblFullTimeVacancyNumberDisplayPanel with Vacancy Number as label text

create JTextField tfFullTimeName, tfFullTimeDesignation, tfFullTimeJobType, tfFullTimeQualification, tfFullTimeWorkingHour,

tfFullTimeSalary, tfFullTimeJoiningDate,

tfFullTimeAppointedBy,

tfFullTimeVacancyNumberVacancyPanel,
tfFullTimeVacancyNumberAppointPanel, t

fFullTimeVacancyNumberDisplayPanel

initialize and **create** JButton fullTimeAddVacancyButton with Add Vacancy as button text

initialize and **create** JButton fullTimeClearButtonVacancyPanel with Clear as button text

initialize and **create** JButton fullTimeClearButtonAppointPanel with Clear as button text

initialize and **create** JButton fullTimeClearButtonDisplayPanel with Clear as button text

initialize and **create** JButton fullTimeDisplayButton with Display as button text

initialize and **create** JButton fullTimeAppointButton with Appoint as button text

initialize and **create** JButton fullTimeBackButtonVacancyPanel with Back as button text

initialize and **create** JButton fullTimeBackButtonAppointPanel with Back as button text

initialize and **create** JButton fullTimeBackButtonDisplayPanel with Back as button text

initialize and **create** JPanel vacancyPanel

set the layout JPanel vacancyPanel to null

set Bounds for JLabel lblFullTimeVacancyNumberVacancyPanel and JTextField tfFullTimeVacancyNumberVacancyPanel

set Bounds for JLabel lblFullTimeDesignation and JTextField tfFullTimeDesignation

set Bounds for JLabel lblFullTimeJobType and JTextField tfFullTimeJobType

set Bounds for JLabel lblFullTimeWorkingHour and JTextField tfFullTimeWorkingHour

set Bounds for JLabel lblFullTimeSalary and JTextField tfFullTimeSalary

set Bounds for JButton fullTimeBackButtonVacancyPanel

set cursor of JButton fullTimeBackButtonVacancyPanel to the hand cursor using instance of cursor object

add action listener to the JButton fullTimeBackButtonVacancyPanel

dispose frameFullTimeStaffHire

set fullTimeWindowCheck to false

end listener

set Bounds for JButton fullTimeAddVacancyButton

set cursor of JButton fullTimeAddVacancyButton to the hand cursor using instance of cursor object

add action listener to the JButton fullTimeAddVacancyButton

initialize integer vacancyNumber and **set** its value by calling getFullTimeTfVacancyNumber method

initialize String designation and **set** its value by calling getFullTimeTfDesignation method

initialize String jobType and **set** its value by calling getFullTimeTfJobType method

initialize integer workingHour and **set** its value by calling getFullTimeTfWorkingHour method

initialize integer salary and **set** its value by calling getFullTimeTfSalary method

if vacancyNumber != 0 and workingHour != 0 and salary != 0 and designation is not empty and jobType is not empty

initialize integer size and **set** its value by calling the size method of ArrayList for arrayListStaffHire

if size > 0

for StaffHire sh : arrayListStaffHire

initialize integer num and get the vacancy number using getVacancyNumber method of StaffHire class

if num == vacancyNumber

show error message "Vacancy already exists"

return

end if

end for loop

call FullTimeStaffHire's constructor and pass vacancyNumber, designation, jobType, workingHour, and salary

add the FullTimeStaffHire's instance to the ArrayList arrayListStaffHire

call clearTextFullTime method

else

call FullTimeStaffHire's constructor and pass vacancyNumber, designation, jobType, workingHour, and salary

add the FullTimeStaffHire's instance to the ArrayList arrayListStaffHire

call clearTextFullTime method

end if

downcast DefaultTableModel modelVacancyTable into DefaultTableModel vacancyTable and use getModel method

add row to the modelVacancyTable by creating a new object and adding the integer vacancyNumber in the vacancy number column and Full Time to the employment type column

else if designation is empty and jobType is empty

show message "Designation and job type are empty"

else if designation is empty

show message "Designation is empty"

else

show message "Job type is empty"

end if

end listener

set Bounds for JButton fullTimeClearButtonVacancyPanel

set cursor of JButton fullTimeClearButtonVacancypanel to the hand cursor using instance of cursor object

add action listener to the JButton fullTimeClearButtonVacancyPanel

call clearTextFullTime method

end listener

add JLabel lblFullTimeVacancyNumberVacancyPanel, lblFullTimeDesignation, lblFullTimeJobType, lblFullTimeWorkingHour, lblFullTimeSalary to the JPanel vacancyPanel

add JTextField tfFullTimeVacancyNumberVacancyPanel, tfFullTimeDesignation, tfFullTimeJobType, tfFullTimeWorkingHour, tfFullTimeWagesPerHour to the JPanel vacancyPanel

add JButton fullTimeBackButtonVacancyPanel, fullTimeAddVacancyButton, and fullTimeClearButtonVacancyPanel to the JPanel vacancyPanel

initialize and **create** JPanel appointPanel

set JPanel appointPanel's layout to null

set Bounds for JLabel lblFullTimeVacancyNumberAppointPanel and JTextField tfFullTimeVacancyNumberAppointPanel

set Bounds for JLabel lblFullTimeName and JTextField tfFullTimeName

set Bounds for JLabel lblFullTimeJoiningDate and JTextField tfFullTimeJoiningDate

set Bounds for JLabel lblFullTimeQualification and JTextField tfFullTimeQualification

set Bounds for JLabel lblFullTimeAppointedBy and JTextField tfFullTimeAppointedBy

set Bounds for JButton fullTimeBackButtonAppointPanel

set cursor of JButton fullTimeBackButtonAppointPanel to the hand cursor using instance of cursor object

add action listener to the JButton fullTimeBackButtonAppointPanel

dispose frameFullTimeStaffHire

set fullTimeWindowCheck to false

end listener

set Bounds for JButton fullTimeAppointButton

set cursor of JButton fullTimeAppointButton to the hand cursor using instance of cursor object

add action listener to the JButton fullTimeAppointButton

initialize String name and **set** it using getFullTimeTfName method

initialize String joiningDate and **set** it using getFullTimeTfJoiningDate method

initialize String qualification and **set** it using getFullTimeTfQualification method

initialize String appointedBy and **set** it using getFullTimeTfAppointedBy method

initialize integer vacancyNumber and **set** it using getFullTimeTfVacancyNumber method

initialize integer size **set** it using the size method for ArrayList on arrayListStaffHire

if vacancyNumber != 0 and name is not empty and joiningDate is not empty and qualification is not empty and appointedBy is not empty

if size > 0

for i = 0, i < size, ++i

get the arrayListStaffHire index using for loop and store it in StaffHire class's instance sh

if sh is instance of FullTimeStaffHire

downcast FullTimeStaffHire f to FullTimeStaffHire sh

initialize integer vacancyNo and set it using StaffHire class's getVacancyNumber method

initialize boolean joinedCheck and set it using FullTimeStaffHire class's getJoined method

if vacancyNo == vacancyNumber and joinedCheck == false

call FullTimeStaffHire's hireFullTimeStaff method which takes name, joiningDate, qualification, appointedBy as parameter to hire the staff

downcast DefaultTableModel modelStaffTable into DefaultTableModel staffTable and use getModel method

add row to the modelStaffTable by creating a new object and adding the integer vacancyNumber in the vacancy number column, name in the name column, and Part Time to the employment type column

call vacancyTableRowRemover and pass vacancyNumber as parameter

call clearTextFullTime method

break

else if vacancyNo == vacancyNumber and joinCheck == true

show error message “Vacancy number already filled”

break

else if (size - 1) == I and vacancyNo != vacancyNumber

show error message “Vacancy Number not found”

end if

end if

end for loop

end if

else if size == 0

show message “No vacancy available”

else if joiningDate is empty and qualification is empty and name is empty and appointedBy is empty

show message “Joining date, qualification, name, and appointed by are empty”

else if joiningDate is empty and qualification is empty and name is empty

show message “Joining date, qualification, and name are empty”

else if joiningDate and qualification and appointedBy is empty

show message “Joining date, qualification, and appointed by are empty”

else if qualification is empty and name is empty and appointedBy is empty

show message “Qualification, name, and appointed by are empty”

else if joiningDate is empty and qualificationis empty

show message “Joining date, and qualification are empty”

else if joiningDate is empty and name is empty

show message “Joining date, and name are empty”

else if joiningDate is empty and appointedBy is empty

show message “Joining date, and appointed by are empty”

else if qualification is empty and name is empty

show message “Qualification, and name are empty”

else if qualification is empty and appointedBy is empty

show message “Qualification, and appointed by are empty”

else if name is empty and appointed by is empty

show message “Name, and appointed by are empty”

else if joiningDate is empty

show message “Joining date is empty”

else if qualification is empty

show message “Qualification is empty”

else if name is empty

show message “Name is empty”

else

show message “Appointed by is empty”

end if

end listener

set Bounds of JButton fullTimeClearButtonAppointPanel

set cursor of JButton fullTimeClearButtonAppointPanel to the hand cursor using instance of cursor object

add action listener to JButton fullTimeClearButtonAppointPanel

call clearTextFullTime method

end listener

add JLabel lblFullTimeVacancyNumberAppointPanel, lblFullTimeName, lblFullTimeJoiningDate, lblFullTimeQualification, lblFullTimeAppointedBy to the JPanel appointPanel

add JTextField tfFullTimeVacancyNumberAppointPanel, tfFullTimeName, tfFullTimeJoiningDate, tfFullTimeQualification, tfFullTimeAppointedBy to the JPanel appointPanel

add JButton fullTimeBackButtonAppointPanel, fullTimeAppointButton, and fullTimeClearButtonAppointPanel to the JPanel appointPanel

initialize and **create** JPanel displayPanel

set JPanel displayPanel's layout to null

set Bounds for JLabel lblFullTimeVacancyNumberDisplayPanel and JTextField tfFullTimeVacancyNumberDisplayPanel

set Bounds for JButton fullTimeBackButtonDisplayPanel

set cursor of JButton fullTimeBackButtonDisplayPanel to the hand cursor using instance of cursor object

add action listener to JButton fullTimeBackButtonDisplayPanel

    dispose JFrame frameFullTimeStaffHire

    set value of fullTimeWindowCheck to false

end listener

set Bounds for JButton fullTimeClearButtonDisplayPanel

set cursor of JButton fullTimeClearButtonDisplayPanel to the hand cursor using instance of cursor object

add action listener to JButton fullTimeClearButtonDisplayPanel

    call clearTextFullTime method

end listener

set Bounds for JButton fullTimeDisplayButton

set cursor of JButton fullTimeDisplayButton to the hand cursor using instance of cursor object

add action listener to JButton fullTimeDisplayButton

initialize integer vacancyNo and **set** its value by calling getFullTimeTfVacancyNumber method

initialize boolean found and **set** its value to false

if vacancyNo != 0

for StaffHire s : arrayListStaffHire

if s is instance of FullTimeStaffHire

downcast FullTimeStaffHire p to FullTimeStaffHire s

if instance StaffHire returns vacancyNumber that is equal to vacancyNo

call displayDetails method of FullTimeStaffHire class

set found's value to true

show info message with staff's details

call clearTextFullTime method

break

end if

end if

end for

if found's value not true

show error message "Vacancy number not found"

end if

end if

end listener

add lblFullTimeVacancyNumberDisplayPanel to JPanel displayPanel

add tfFullTimeVacancyNumberDisplayPanel to JPanel displayPanel

add fullTimeBackButtonDisplayPanel to JPanel displayPanel

add fullTimeClearButtonDisplayPanel to JPanel displayPanel

add fullTimeDisplayButton to JPanel displayPanel

create JTabbedPane tpFullTimeForm

add JPanel vacancyPanel to the JTabbedPane tpFullTimeForm and name the tab as Add Vacancy

add JPanel appointPanel to the JTabbedPane tpFullTimeForm and name the tab as Appoint

add JPanel displayPanel to the JTabbedPane tpFullTimeForm and name the tab as Display details

add JTabbedPane tpFullTimeForm to the JFrame frameFullTimeStaffHire with Border layout and position as center

end function

function clearTextPartTime ():

if JTabbedPane tpPartTimeForm returns selected tab index 0

set JTextField tfPartTimeDesignation text to null

set JTextField tfPartTimeJobType text to null

set JTextField tfPartTimeWorkingHour text to null

set JTextField tfPartTimeWagesPerHour text to null

set JTextField tfPartTimeShifts text to null

set JTextField tfPartTimeVacancyNumberVacancyPanel text to null

else if JTabbedPane tpPartTimeForm returns selected tab index 1

set JTextField tfPartTimeName text to null

set JTextField tfPartTimeQualification text to null

set JTextField tfPartTimeJoiningDate text to null

set JTextField tfPartTimeAppointedBy text to null

set JTextField tfPartTimeVacancyNumberAppointPanel text to null

else if JTabbedPane tpPartTimeForm returns selected tab index 2

set JTextField tfPartTimeVacancyNumberDisplayPanel text to null

end function

function getPartTimeTfName ():

initialize String tfName and set its value by calling getText and trim method for JTextField tfPartTimeName

return tfName

end function

function getPartTimeTfDesignation ():

initialize String tfDesignation and set its value by calling getText and trim method for JTextField tfPartTimeDesignation

return tfDesignation

end function

function getPartTimeTfJobType ():

initialize String tfJobType and set its value by calling getText and trim method for JTextField tfPartTimeJobType

return tfJobType

end function

function getPartTimeTfQualification ():

initialize String tfQualification and set its value by calling getText and trim method for JTextField tfPartTimeQualification

return tfQualification

end function

function getPartTimeTfShifts ():

initialize String tfShifts and set its value by calling getText and trim method for JTextField tfPartTimeShifts

return tfShifts

end function

function getPartTimeTfJoiningDate ():

initialize String tfJoiningDate and set its value by calling getText and trim method for JTextField tfPartTimeJoiningDate

return tfJoiningDate

end function

function getPartTimeTfAppointedBy ():

initialize String tfAppointedBy and set its value by calling getText and trim method for JTextField tfPartTimeAppointedBy

return tfAppointedBy

end function

function getPartTimeTfWorkingHour ():

    initialize integer tfWorkingHour to 0

    try

        set tfWorkingHour's value by using getText and trim method on JTextField tfPartTimeWorkingHour, then parsing the String to integer

    catch NumberFormatException

        show error message "enter the appropriate value"

    catch NullPointerException

        show error message "enter a value"

    return tfWorkingHour

end function

function getPartTimeTfWagesPerHour ():

initialize integer tfWagesPerHour to 0

try

set tfWagesPerHour's value by using getText and trim method on JTextField tfPartTimeWorkingHour, then parsing the String to integer

catch NumberFormatException

show error message "enter the appropriate value"

catch NullPointerException

show error message "enter a value"

return tfWagesPerHour

end function

function getPartTimeTfVacancyNumber ():

initialize integer tfVacancyNumber to 0

try

if JTabbedPane tpPartTimeForm returns selected tab index 0

set tfVacancyNumber's value by using getText and trim method on JTextField tfPartTimeVacancyNumberVacancyPanel, then parsing the String to integer

else if JTabbedPane tpPartTimeForm returns selected tab index 1

set tfVacancyNumber's value by using getText and trim method on JTextField tfPartTimeVacancyNumberAppointPanel, then parsing the String to integer

else if JTabbedPane tpPartTimeForm returns selected tab index 2

set tfVacancyNumber's value by using getText and trim method on JTextField tfPartTimeVacancyNumberDisplayPanel, then parsing the String to integer

catch NumberFormatException

show error message "enter the appropriate value"

catch NullPointerException

show error message "enter a value"

return tfVacancyNumber

end function

function getFullTimeTfName ():

initialize String tfName and set its value by calling getText and trim method for JTextField tfFullTimeName

return tfName

end function

function getFullTimeTfDesignation ():

initialize String tfDesignation and set its value by calling getText and trim method for JTextField tfFullTimeDesignation

return tfDesignation

end function

function getFullTimeTfJobType ():

initialize String tfJobType and set its value by calling getText and trim method for JTextField tfFullTimeJobType

return tfJobType

end function

function getFullTimeTfQualification ():

initialize String tfQualification and set its value by calling getText and trim method for JTextField tfFullTimeQualification

return tfQualification

end function

function getPartTimeTfJoiningDate ():

initialize String tfJoiningDate and set its value by calling getText and trim method for JTextField tfPartTimeJoiningDate

return tfJoiningDate

end function

function getFullTimeTfAppointedBy ():

initialize String tfAppointedBy and set its value by calling getText and trim method for JTextField tfFullTimeAppointedBy

return tfAppointedBy

end function

function getFullTimeTfWorkingHour ():

initialize integer tfWorkingHour to 0

try

set tfWorkingHour's value by using getText and trim method on JTextField tfFullTimeWorkingHour, then parsing the String to integer

catch NumberFormatException

show error message "enter the appropriate value"

catch NullPointerException

show error message "enter a value"

return tfWorkingHour

end function

function getFullTimeSalary ():

initialize integer tfSalary to 0

try

set tfSalary's value by using getText and trim method on JTextField tfFullTimeSalary, then parsing the String to integer

catch NumberFormatException

show error message "enter the appropriate value"

catch NullPointerException

show error message "enter a value"

return tfFullTimeSalary

end function

function getFullTimeTfVacancyNumber ():

initialize integer tfVacancyNumber to 0

try

if JTabbedPane tpFullTimeForm returns selected tab index 0

set tfVacancyNumber's value by using getText and trim method on JTextField tfFullTimeVacancyNumberVacancyPanel, then parsing the String to integer

else if JTabbedPane tpFullTimeForm returns selected tab index 1

set tfVacancyNumber's value by using getText and trim method on JTextField tfFullTimeVacancyNumberAppointPanel, then parsing the String to integer

else if JTabbedPane tpFullTimeForm returns selected tab index 2

set tfVacancyNumber's value by using getText and trim method on JTextField tfFullTimeVacancyNumberDisplayPanel, then parsing the String to integer

catch NumberFormatException

show error message "enter the appropriate value"

catch NullPointerException

show error message "enter a value"

return tfVacancyNumber

end function

function clearTextFullTime ():

if JTabbedPane tpFullTimeForm returns selected tab index 0

set JTextField tfFullTimeDesignation text to null

set JTextField tfFullTimeJobType text to null

set JTextField tfFullTimeWorkingHour text to null

set JTextField tfFullTimeSalary text to null

set JTextField tfFullTimeVacancyNumberVacancyPanel text to null

else if JTabbedPane tpFullTimeForm returns selected tab index 1

set JTextField tfFullTimeName text to null

set JTextField tfFullTimeQualification text to null

set JTextField tfFullTimeJoiningDate text to null

set JTextField tfFullTimeAppointedBy text to null

set JTextField tfFullTimeVacancyNumberAppointPanel text to null

else if JTabbedPane tpFullTimeForm returns selected tab index 2

set JTextField tfFullTimeVacancyNumberDisplayPanel text to null

end function

function staffTableRowRemover (int x):

initialize integer rowCount and **set** its value by calling getRowCount on JTable staffTable

initialize integer rowToDelete to 0

for i = 0, i < rowCount, i++

initialize String rowEntry and **set** its value by calling getValueAt on JTable staffTable to get the value of the current iteration of the row

initialize integer rowEntryNum and **parse** the rowEntry and **set** rowEntryNum's value

if x == rowEntryNum

set rowToDelete to current iteration of i

break

end if

end for

downcast DefaultTableModel modelStaffTable into DefaultTableModel staffTable and use getModel method

remove row of staffTable by calling the removeRow method and passing the rowToDelete as the row number

end function

function vacancyTableRowRemover (int x):

initialize integer rowCount and **set** its value by calling getRowCount on JTable vacancyTable

initialize integer rowToDelete to 0

for i = 0, i < rowCount, i++

initialize String rowEntry and **set** its value by calling getValueAt on JTable vacancyTable to get the value of the current iteration of the row

initialize integer rowEntryNum and **parse** the rowEntry and **set** rowEntryNum's value

if x == rowEntryNum

set rowToDelete to current iteration of i

break

end if

end for

downcast DefaultTableModel modelVacancyTable into DefaultTableModel vacancyTable and use getModel method

remove row of vacancyTable by calling the removeRow method and passing the rowToDelete as the row number

end function

function main (String [] args):

> **create** a new object of INGNepal and **call** JFrame frameMainMenu and set its visibility to true

end function

Method description

INGNepal ()

This is the constructor of INGNepal class and it opens and sets up the main menu frame of the program. It consists of two tabbed pane and each one of them holds a table for easy access and reference of the data that has already been inputted in the program. There are two buttons called Part Time Operations and Full Time Operations in the main menu which opens up new frames to operate on part time and full time respectively.

public void partTimeForm ()

This method consists of another JFrame called framePartTimeStaffHire which opens up when this method is called and this form or other form is not open. It consists of tabbed pane each holding add vacancy, appoint staff, and display and terminate operation. Each one of the tabbed panes holds a panel with a form in it with required fields to take the input and output suitably.

public void fullTimeForm ()

This method consists of a JFrame called frameFullTimeStaffHire which opens up when this method is called; it only opens up if this form is not already opened and another form is also not opened. This method creates a JFrame which consists a tabbed pane each holding operation related to add vacancy, appoint staff, and display staff details, the operations are performed by inputting in the required fields which resides inside a panel in tabbed panes.

public void actionPerformed (ActionEvent e)

This method has no body as it is only kept to override the actionPerformed method of ActionListener interface.

public void clearTextPartTime ()

ClearTextPartTime method clears the input fields of JFrame framePartTimeStaffHire; it requests the index of selected pane in the form and clears the text fields of current tab accordingly by setting the text fields' text to null.

public String getPartTimeTfName ()

This is a getter method takes the value from tfPartTimeName text field in JFrame framePartTimeStaffHire and returns the String.

public String getPartTimeTfDesignation ()

This is a getter method that takes the value from text field tfPartTimeDesignation in JFrame framePartTimeStaffHire and returns the String.

public String getPartTimeTfJobType ()

This is a getter method that takes the value from text field tfPartTimeJobType in JFrame framePartTimeStaffHire and returns the String

public String getPartTimeTfQualification ()

This is a getter method that takes the value from text field tfPartTimeQualification in JFrame framePartTimeStaffHire and returns the String

public String getPartTimeTfShifts ()

This is a getter method that takes the value from text field tfPartTimeShifts in JFrame framePartTimeStaffHire and returns the String.

public String getPartTimeTfJoiningDate ()

This is a getter method that takes the value from text field tfPartTimeJoiningDate in JFrame framePartTimeStaffHire and returns the String

public String getPartTimeTfAppointedBy ()

This is a getter method that takes the value from text field tfPartTimeAppointedBy in JFrame framePartTimeStaffHire and returns the String.

public int getPartTimeTfWorkingHour ()

This method is a getter method that takes the value from the text field tfPartTimeWorkingHour in JFrame framePartTimeStaffHire and checks whether the input is an integer or not if it is not an integer it throws a number format exception; it also checks whether there is a value in the text field or not, if there is not input inside the text field then it returns a null pointer exception. And if the input in the text field is an integer then it returns the integer.

public int getPartTimeTfWagesPerHour ()

This method is a getter method that takes the value from the text field tfPartTimeWagesPerHour in JFrame framePartTimeStaffHire and checks whether the input is an integer or not if it is not an integer it throws a number format exception; it also checks whether there is a value in the text field or not, if there is not input inside the text field then it returns a null pointer exception. And if the input in the text field is an integer then it returns the integer.

public int getPartTimeTfVacancyNumber ()

This method is a getter method that takes the value from the multiple text fields according to the selected tab in JFrame framePartTimeStaffHire. This method takes input from tfPartTimeVacancyNumberVacancyPanel, tfPartTimeVacancyNumberAppointPanel, and tfPartTimeVacancyNumberDisplayPanel when the index value returned by tabbed pane is 0, 1, and 2 respectively. Like other getter methods it checks whether the input is an integer or not and throws a number format exception if the inputted value is not an integer; this method also checks whether the input is null or not and throws null pointer exception if the text field is empty. And if the input in the text field in the selected tab is an integer it returns the integer.

public String getFullTimeTfName ()

This is a getter method takes the value from tfFullTimeName text field in JFrame frameFullTimeStaffHire and returns the String.

public String getFullTimeTfDesignation ()

This is a getter method that takes the value from text field tfFullTimeDesignation in JFrame frameFullTimeStaffHire and returns the String.

public String getFullTimeTfJobType ()

This is a getter method that takes the value from text field tfFullTimeJobType in JFrame frameFullTimeStaffHire and returns the String.

public String getFullTimeTfQualification ()

This is a getter method that takes the value from text field tfFullTimeQualification in JFrame frameFullTimeStaffHire returns the String.

public String getFullTimeTfJoiningDate ()

This is a getter method that takes the value from text field tfFullTimeJoiningDate in JFrame frameFullTimeStaffHire and returns the String.

public String getFullTimeTfAppointedBy ()

This is a getter method that takes the value from text field tfFullTimeAppointedBy in JFrame frameFullTimeStaffHire and returns the String

public int getFullTimeTfWorkingHour ()

This method is a getter method that takes the value from the text field tfFullTimeWorkingHour in JFrame frameFullTimeStaffHire and checks whether the input is an integer or not if it is not an integer it throws a number format exception; it also checks whether there is a value in the text field or not, if there is not input inside the text field then it returns a null pointer exception. And if the input in the text field is an integer then it returns the integer.

public int getFullTimeTfSalary ()

This method is a getter method that takes the value from the text field tfFullTimeSalary in JFrame frameFullTimeStaffHire and checks whether the input is an integer or not if it is not an integer it throws a number format exception; it also checks whether there is a value in the text field or not, if there is not input inside the text field then it returns a null pointer exception. And if the input in the text field is an integer then it returns the integer.

public int getFullTimeTfVacancyNumber ()

This method is a getter method that takes the value from the multiple text fields according to the selected tab in JFrame frameFullTimeStaffHire. This method takes input from tfFullTimeVacancyNumberVacancyPanel, tfFullTimeVacancyNumberAppointPanel, and tfFullTimeVacancyNumberDisplayPanel when the index value returned by tabbed pane is 0, 1, and 2 respectively. Like other getter methods it checks whether the input is an integer or not and throws a number format exception if the inputted value is not an integer; this method also checks whether the input is null or not and throws null pointer exception if the text field is empty. And if the input in the text field in the selected tab is an integer it returns the integer.

public void clearTextFullTime ()

ClearTextFullTime method clears the input fields of JFrame frameFullTimeStaffHire; it requests the index of selected pane in the form and clears the text fields of current tab accordingly by setting the text fields' text to null.

public void staffTableRowRemover (int x)

This method's primary function is to remove the rows from the staff table present in the tabbed pane of JFrame frameMainMenu. It has integer as a formal parameter and it takes that parameter and runs through a loop to check the parameter to the values inside the vacancy table column. When the value in the vacancy table column matches to the parameter it removes the row from the staff table in the main menu.

public void vacancyTableRowRemover (int x)

This method's primary function is to remove the rows from the vacancy table present in the tabbed pane of JFrame frameMainMenu. It has integer as a formal parameter and it takes that parameter and runs through a loop to check the parameter to the values inside the vacancy table column. When the value in the vacancy table column matches to the parameter it removes the row from the vacancy table in the main menu.

public static void main (String [] args)

This is the main method of this class and it does not have much implementation, as its primary function is to call the constructor of the INGNepal class and set the JFrame frameMainMenu's visibility to true.

Testing

Test 1

Table 1: Test 1

Objective	Test that the program can be compiled and run using the command prompt.
Action	The INGNepal.java file is compiled from command prompt using the javac command and after that it is ran using java command.
Expected result	The file will be compiled and the program will run without any error.
Actual result	The file will be compiled and the program will run without any error.
Conclusion	Test Successful

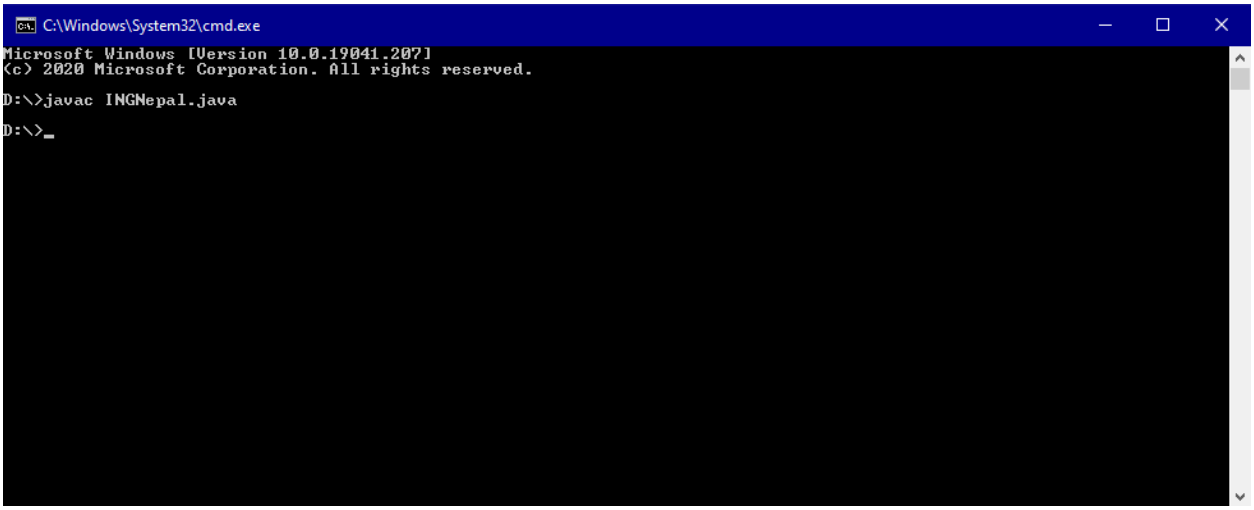

Figure 5: Test 1 Compiling the program

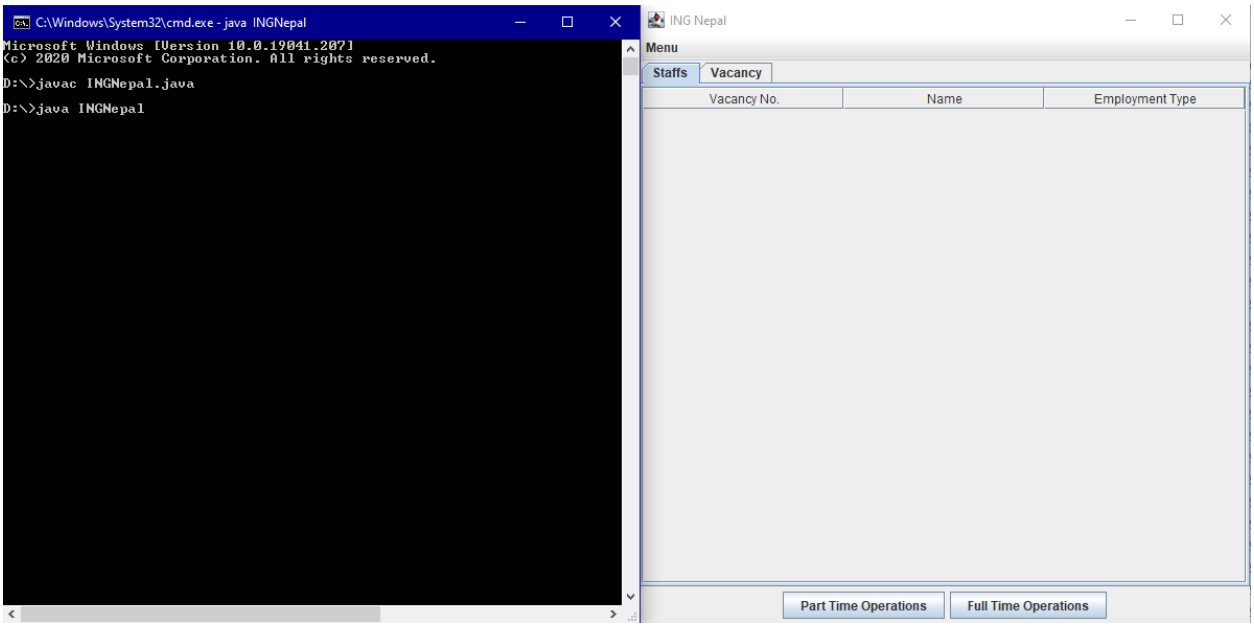

Figure 6: Test 1 Running the program

Test 2

Test 2-Part 1

Table 2: Test 2(1)

Objective	Test that the program can add vacancy, appoint and terminate for part time staff.
Action	Vacancy for part time staff is created with following attributes Vacancy number = 13 Working Hour = 8 Designation = Receptionist Wages Per Hour = 100 Job type = Administrative Shifts = Morning A part time staff is hired with following details Vacancy number = 13 Joining Date = 10/10/2019 Name = Sandip Tamang Qualification = BBA Appointed by = Samarpan Rai Display the details of part time staff Terminate the part time staff with vacancy number 13
Expected result	The operations will run without any error.
Actual result	The operations ran smoothly.
Conclusion	Test Successful.

Test 2-Part 2

Table 3: Test 2(2)

Objective	Test that the program can add vacancy and appoint for full time staff.
Action	Vacancy for full time staff is created with following attributes Vacancy number = 19 Working Hour = 10 Designation = Manager Salary = 30000 Job Type = Management A full-time staff is hired with following details Vacancy number = 19 Qualification = MBA Name = Sonu Nakarmi Joining Date = 11/12/2019 Appointed by = Samriddhi Sai Display the details of full time staff
Expected result	The operations will run without any error.
Actual result	The operations ran smoothly.
Conclusion	Test Successful

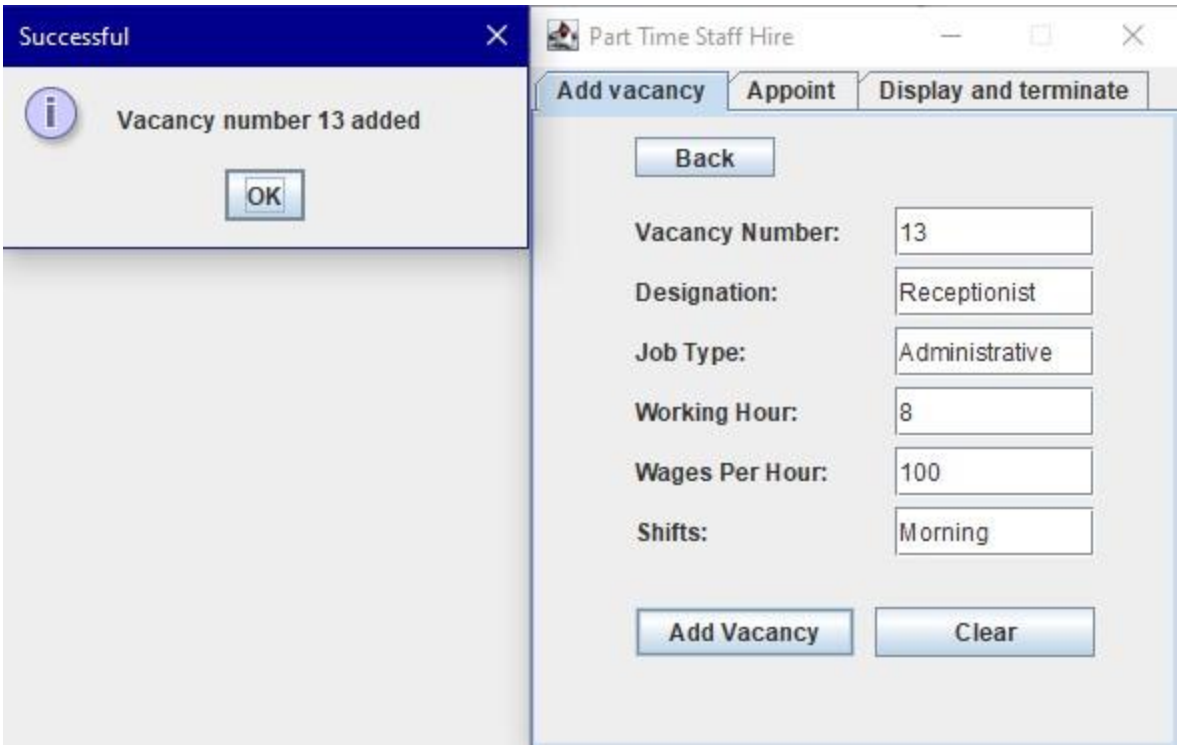

Figure 7: Test 2 Adding part time vacancy

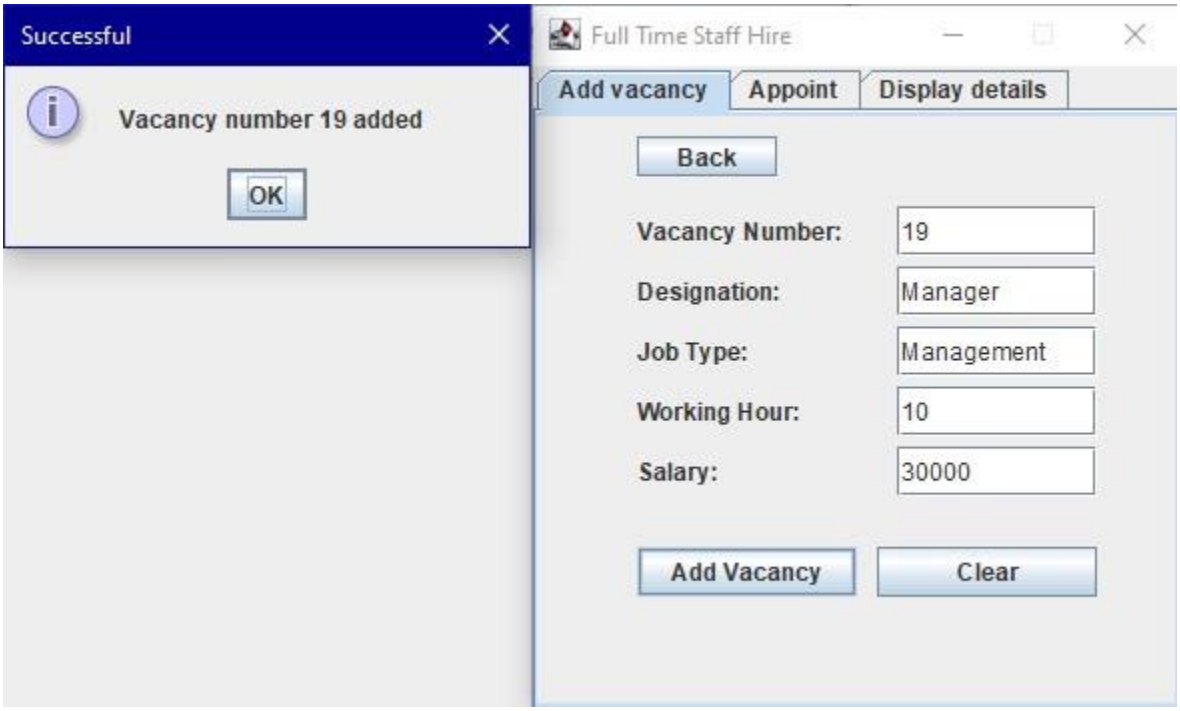

Figure 8: Test 2 Adding full time vacancy

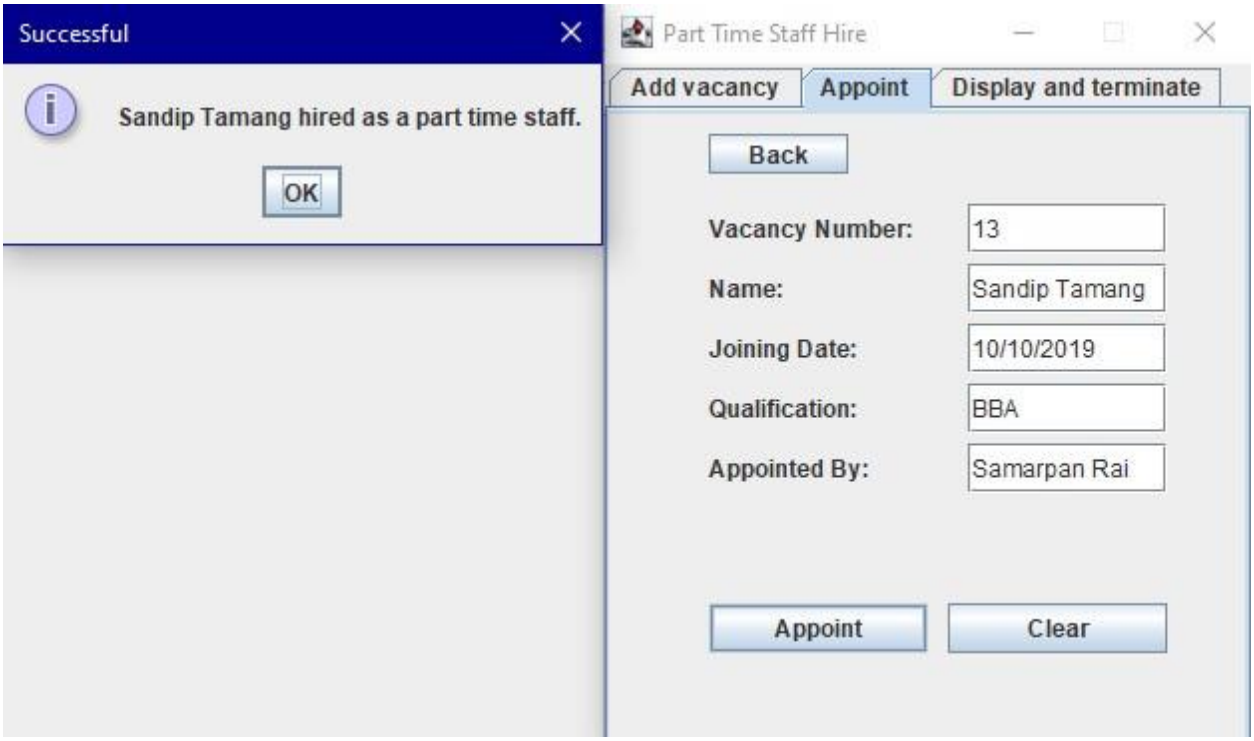

Figure 9: Test 2 Appointing a part time staff

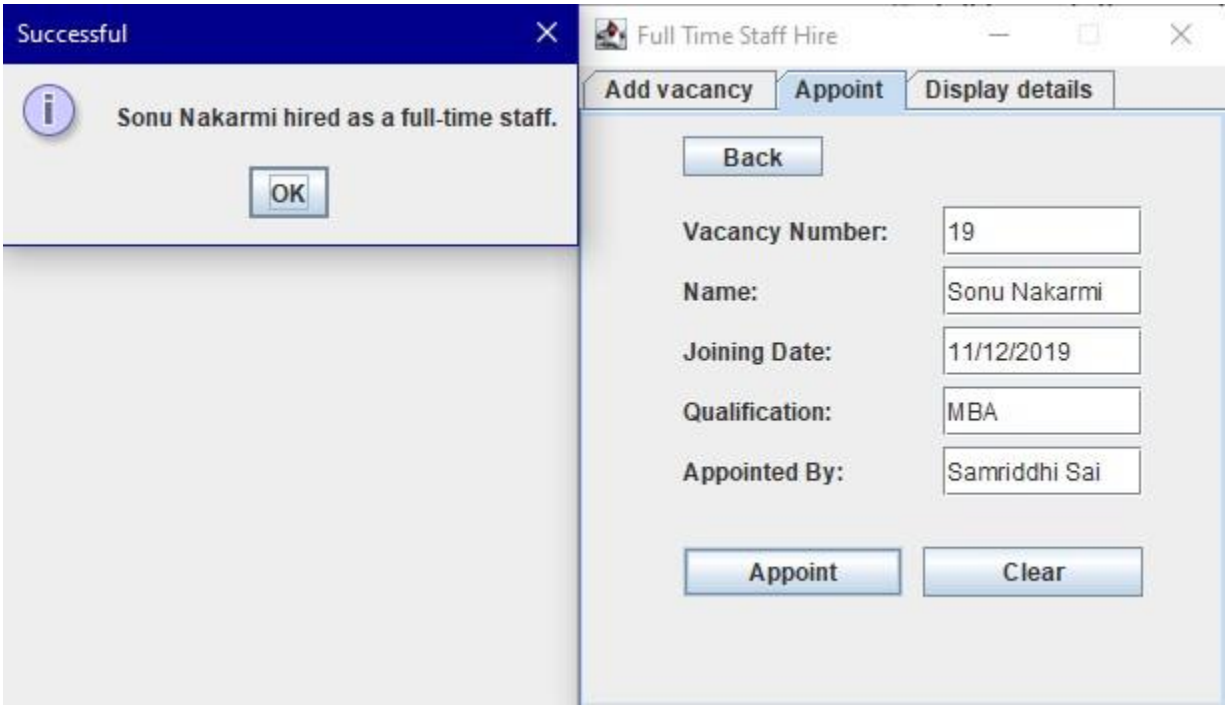

Figure 10: Test 2 Appointing a full time staff

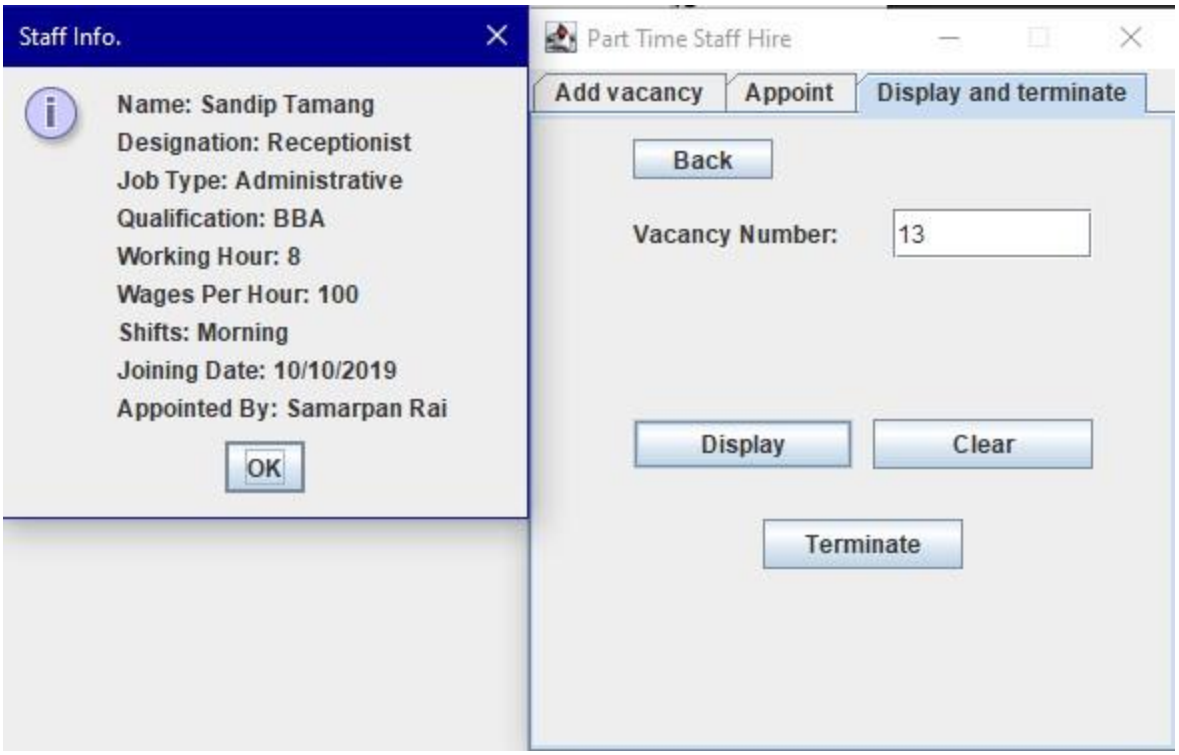

Figure 11: Test 2 Displaying part time staff details(1)

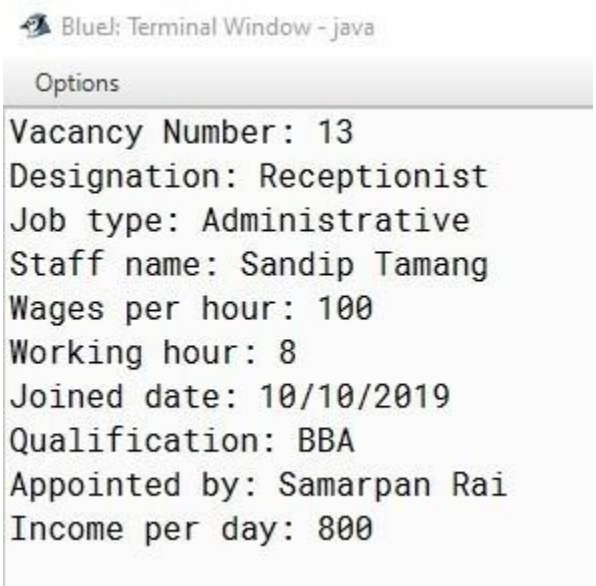

Figure 12: Test 2 Displaying part time staff details(2)

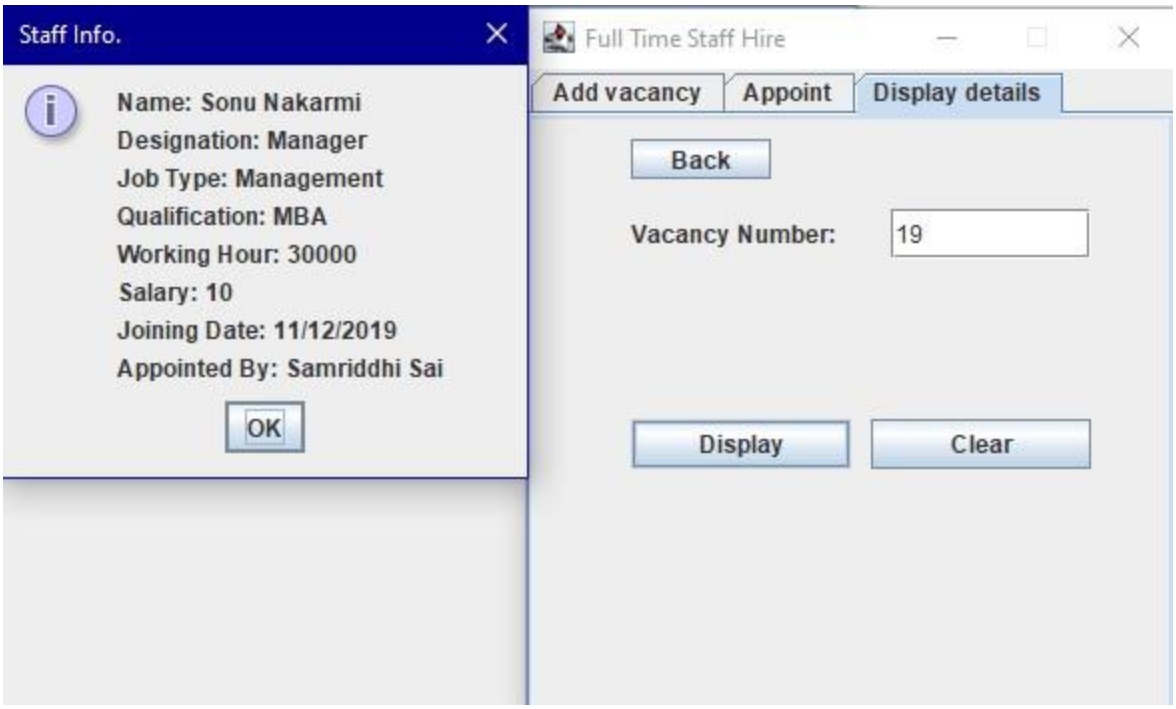

Figure 13: Test 2 Displaying full time staff details(1)

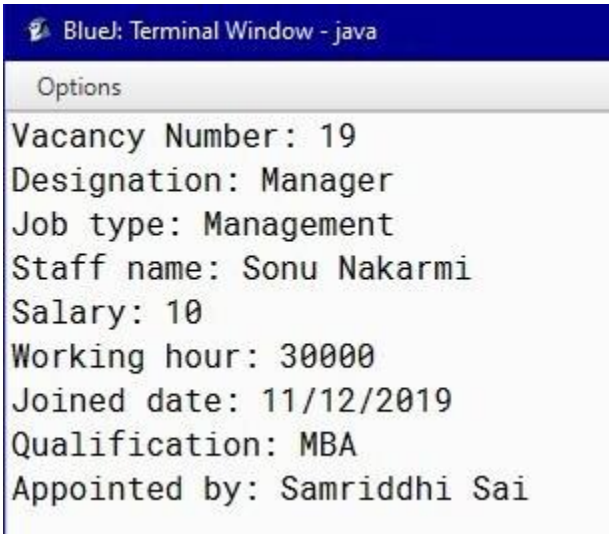

Figure 14: Test 2 Displaying full time staff details(2)

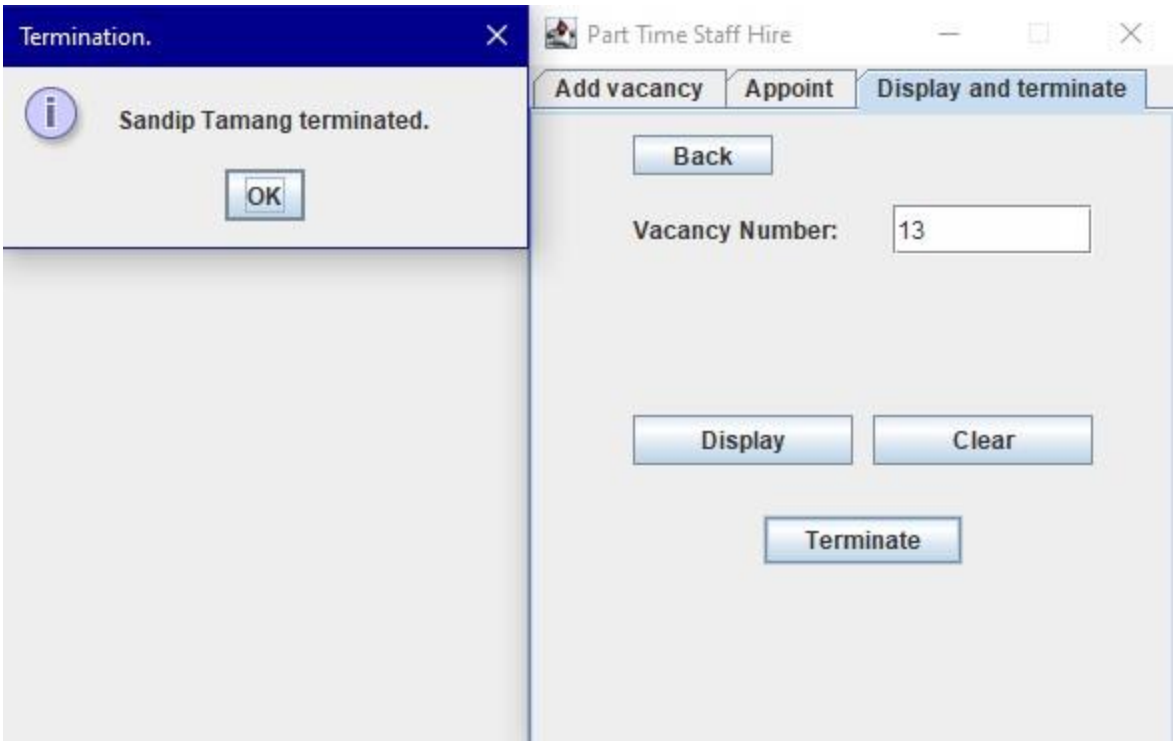

Figure 15: Test 2 Terminating a part time staff

Test 3

Table 4: Test 3

Objective	Test that appropriate dialog boxes appear when unsuitable values are entered for the vacancy number.
Action	Characters are inputted into the vacancy number text field and empty vacancy number is also passed into the text field.
Expected result	Error messages should pop up and operation is not performed.
Actual result	Error messages appeared and operation was not performed.
Conclusion	Test Successful

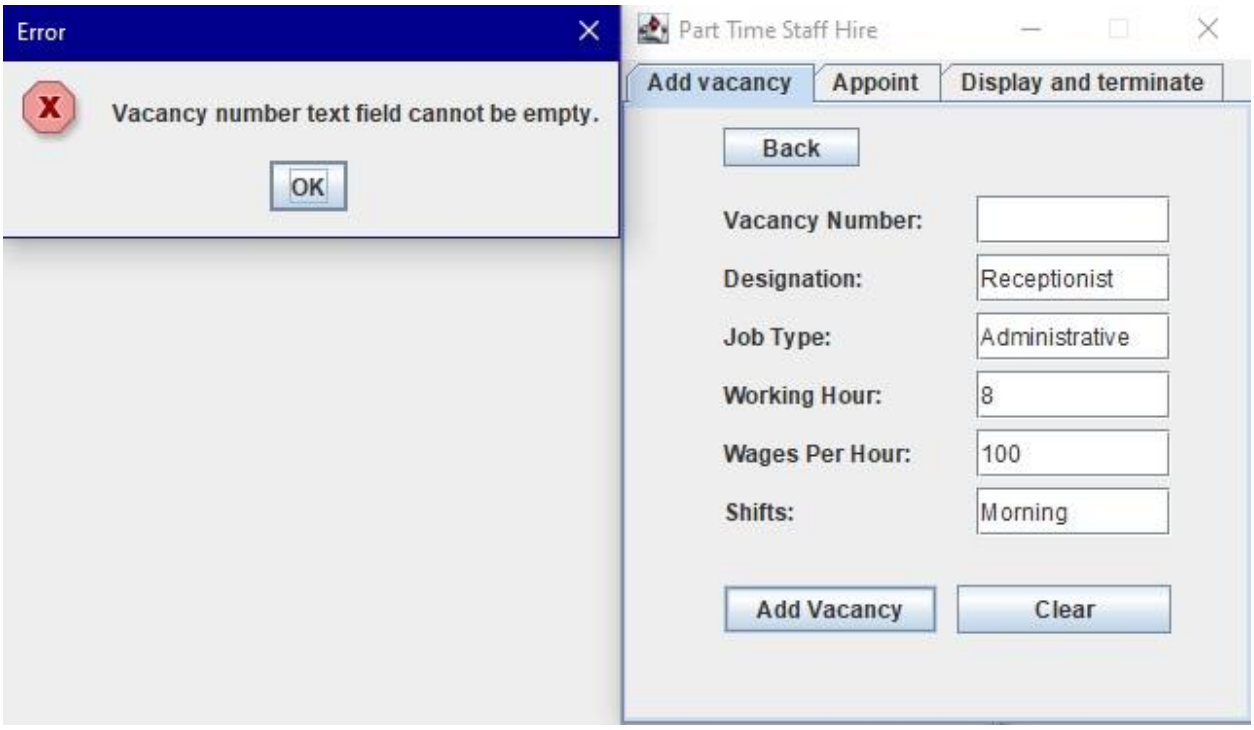

Figure 16: Test 3 Null value in add vacancy of part time staff hire

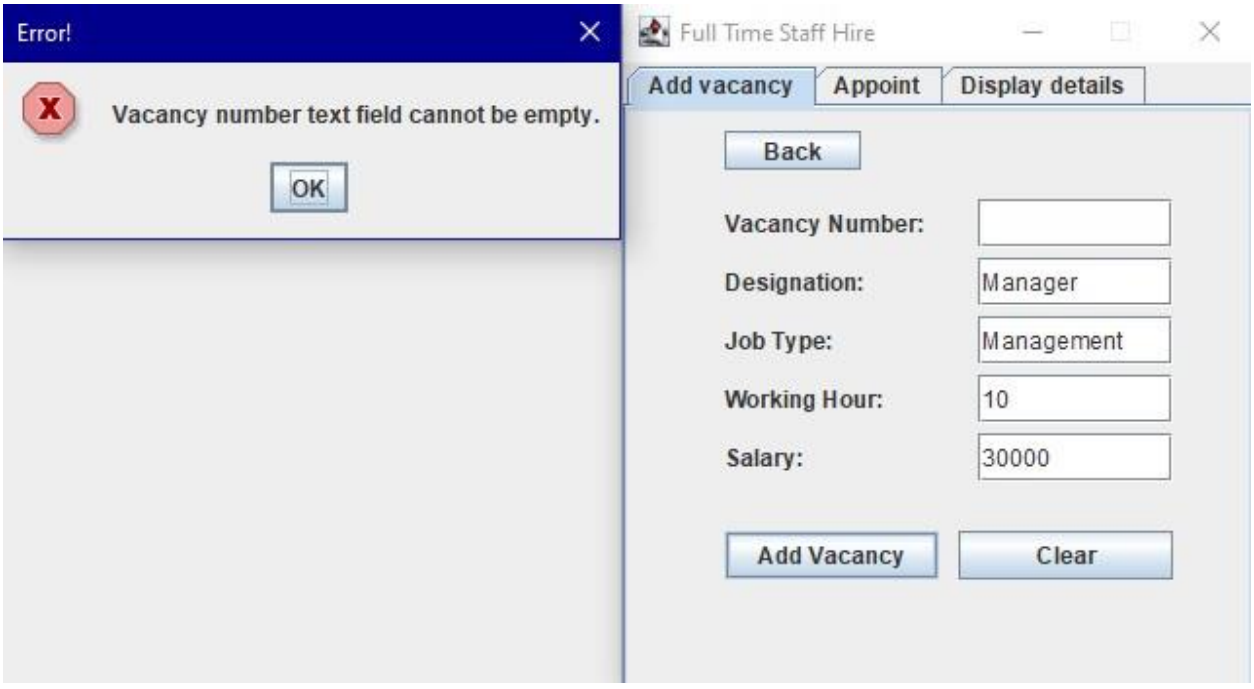

Figure 17: Test 3 Null value in add vacancy of full time staff hire

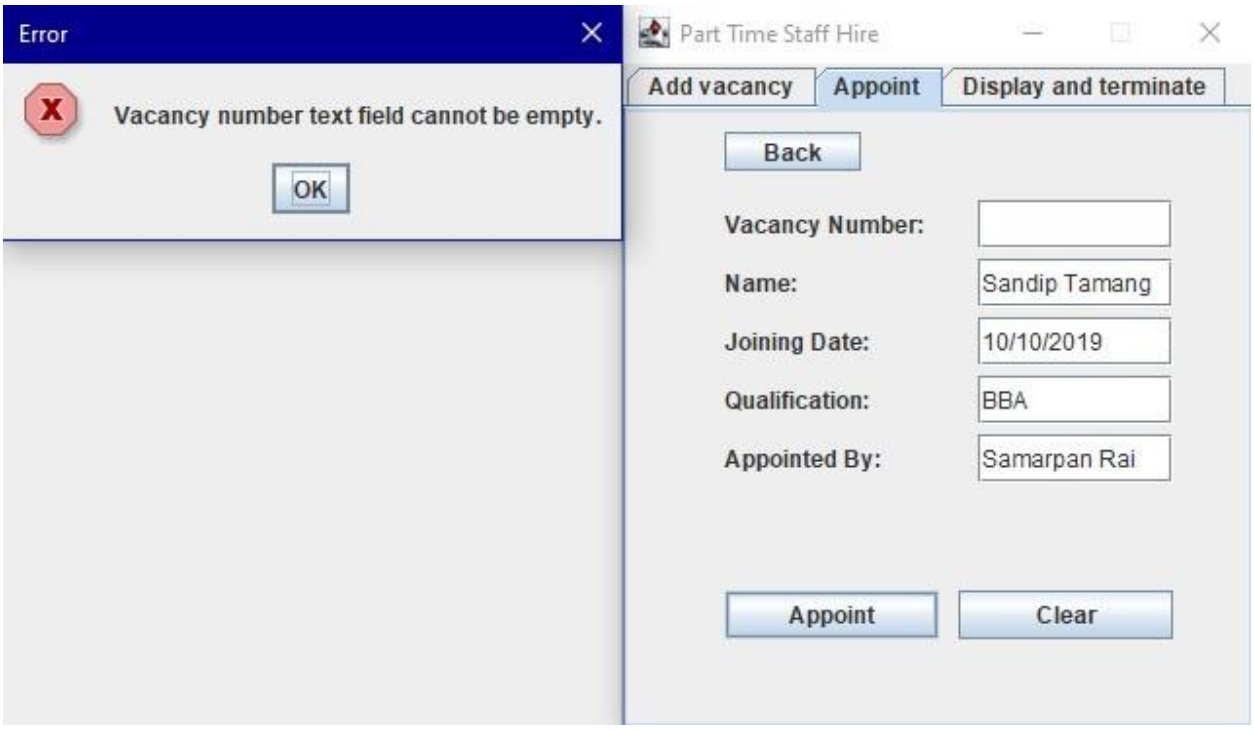

Figure 18: Test 3 Null value in appoint of part time staff hire

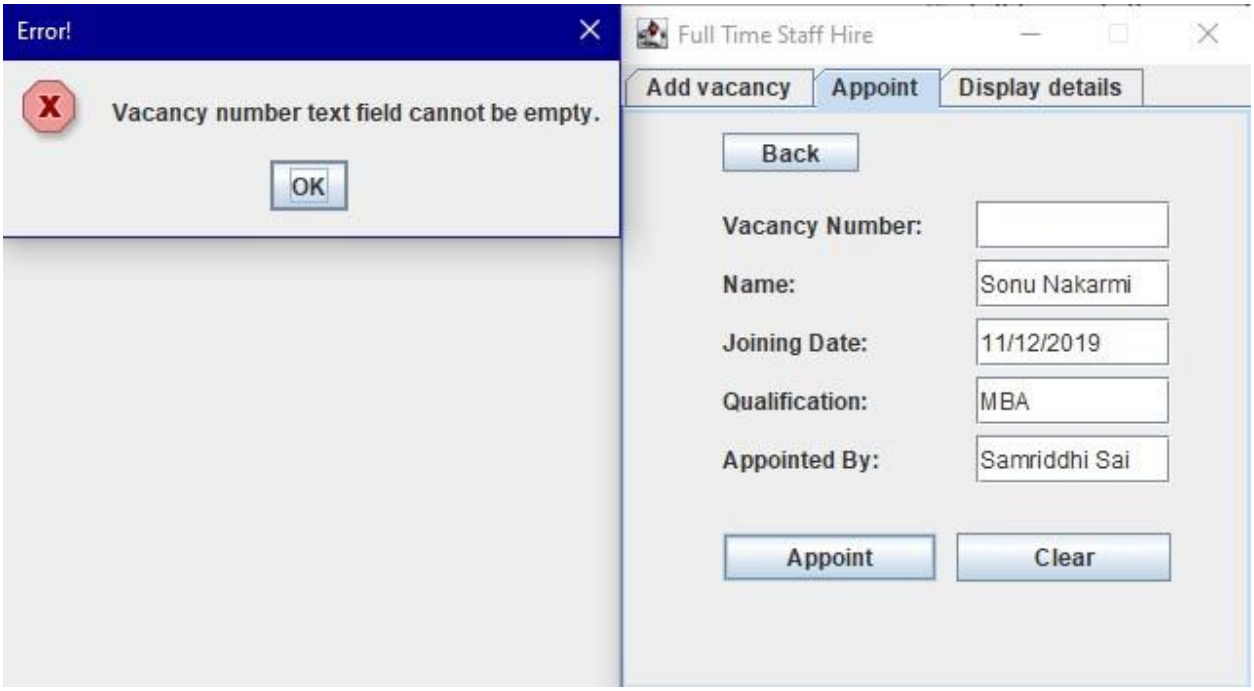

Figure 19: Test 3 Null value in appoint of full time staff hire

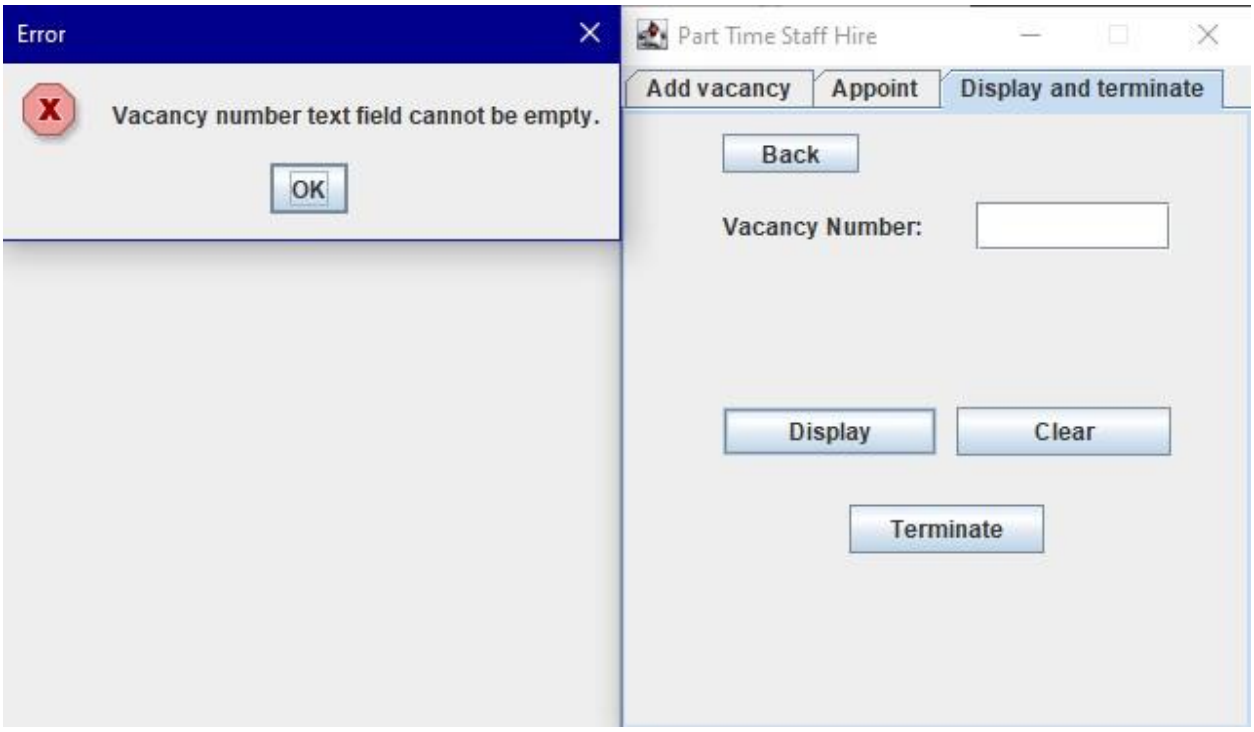

Figure 20: Test 3 Null value in display and terminate of part time staff hire

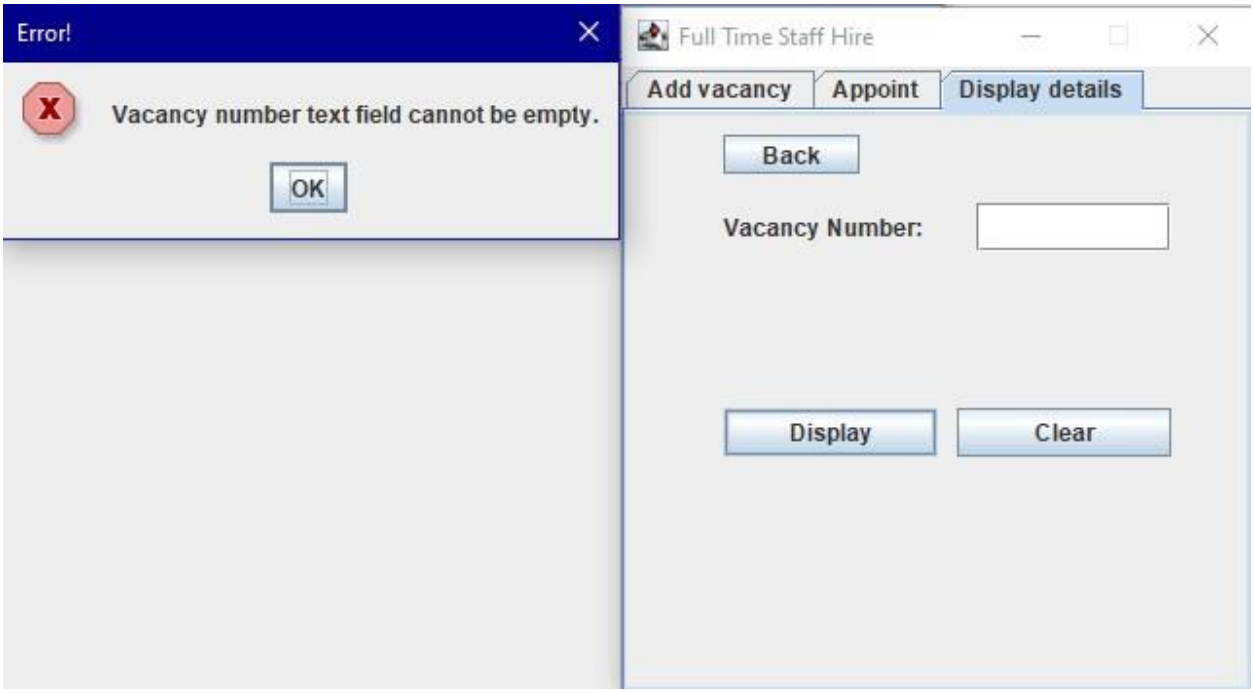

Figure 21: Test 3 Null value in display of full time staff hire

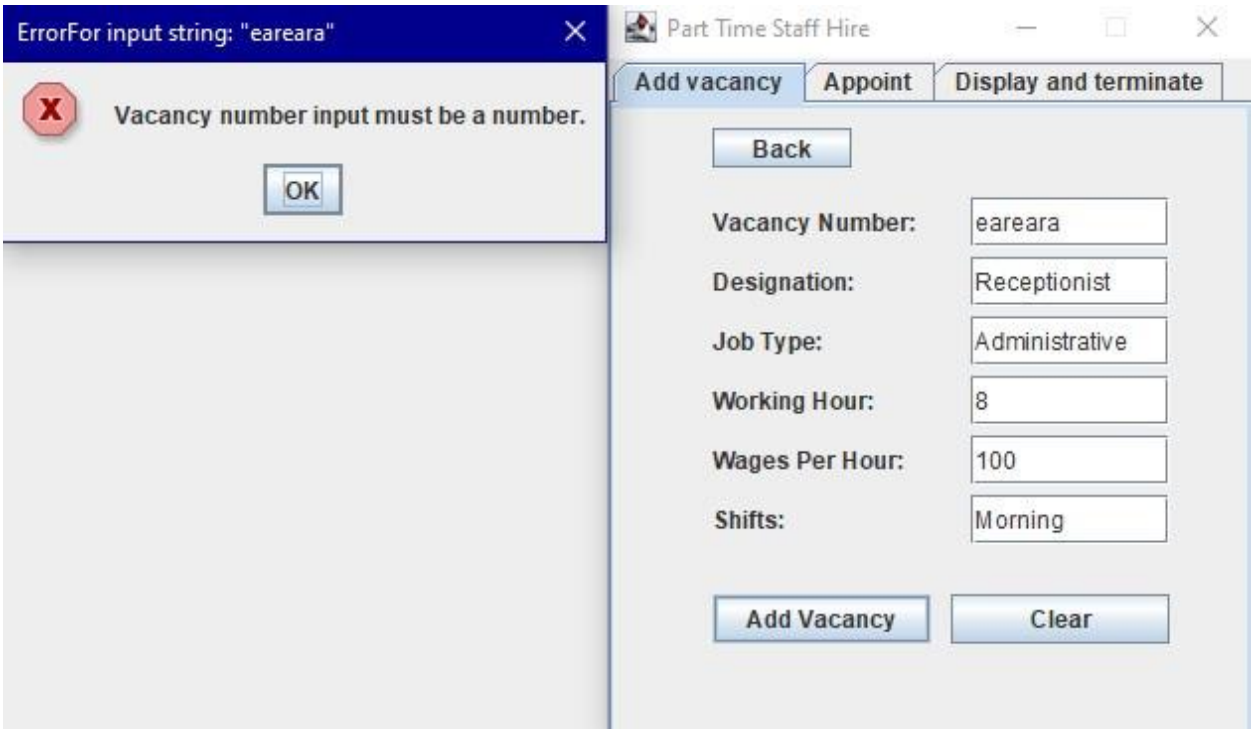

Figure 22: Test 3 String input in vacancy text field of add vacancy in part time staff hire

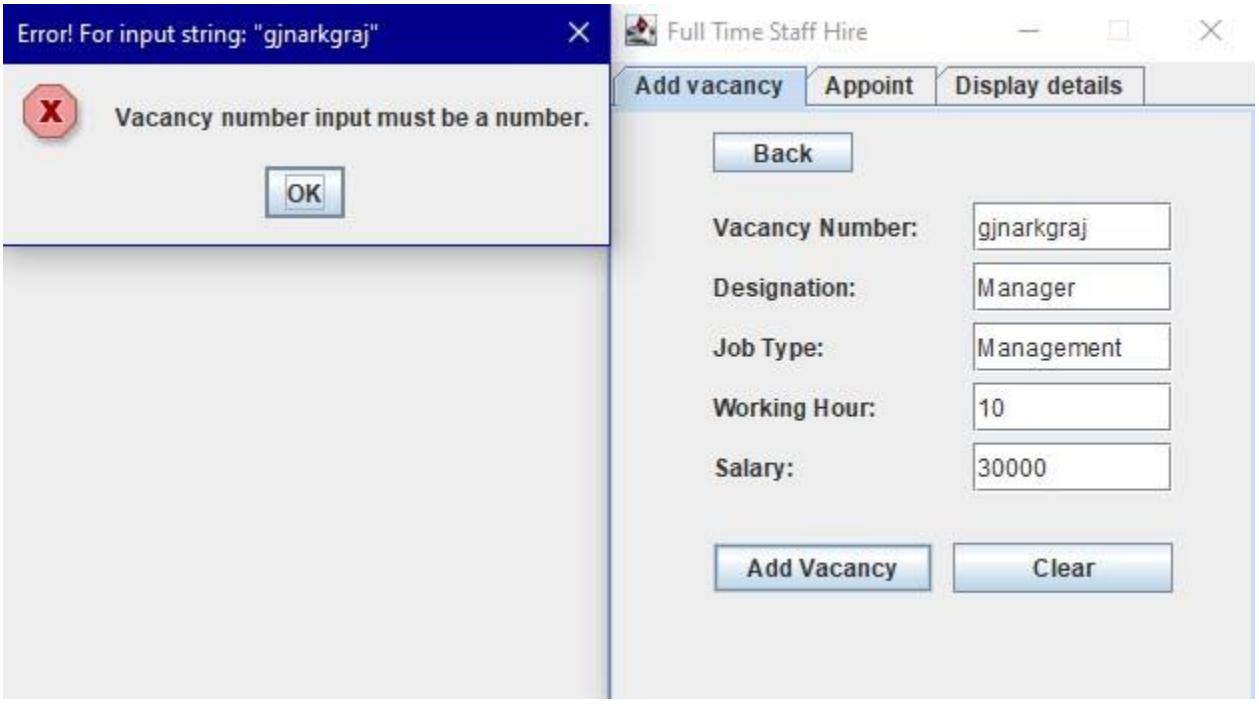

Figure 23: Test 3 String input in vacancy text field of add vacancy in part time staff hire

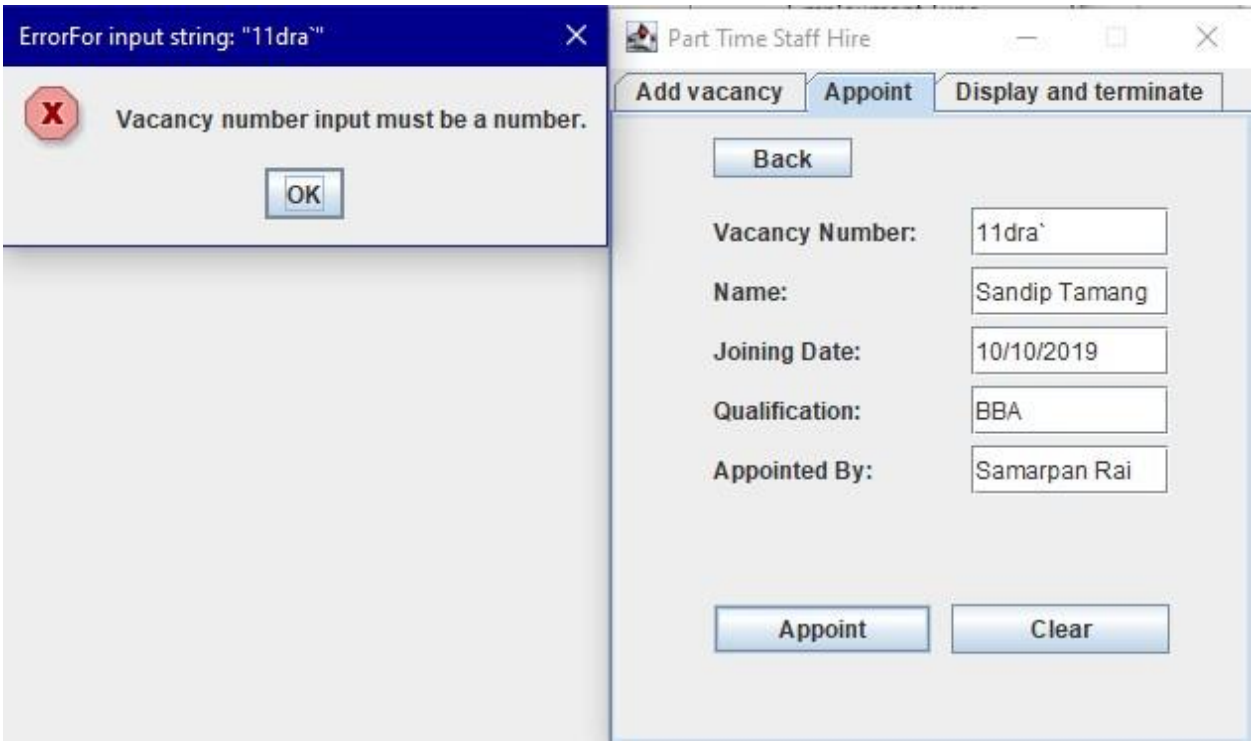

Figure 24: Test 3 String input in vacancy text field of appoint in part time staff hire

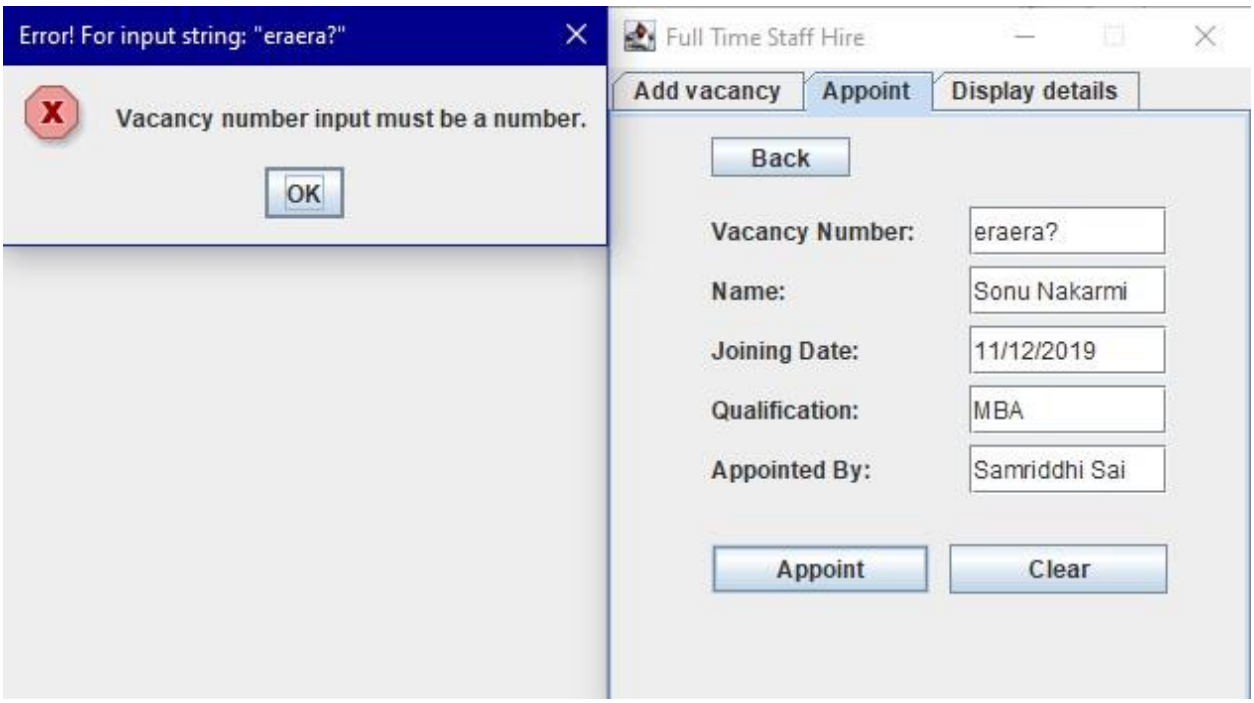

Figure 25: Test 3 String input in vacancy text field of appoint in full time staff hire

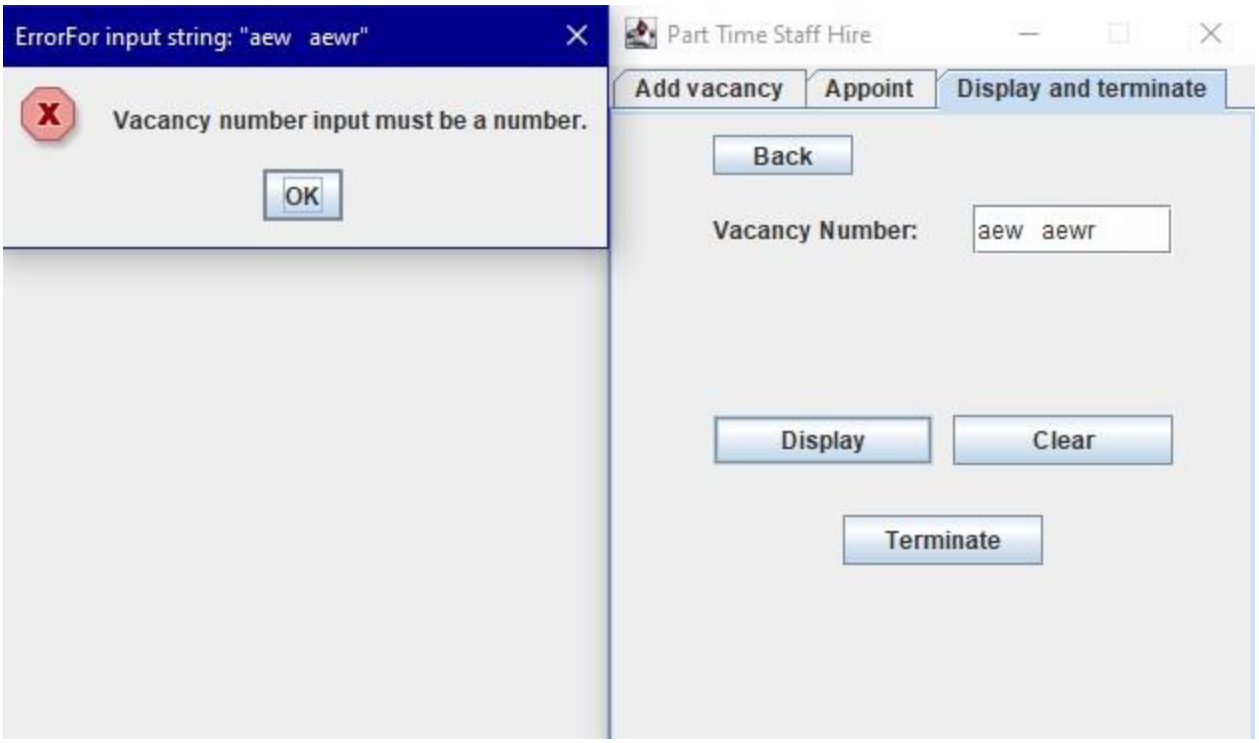

Figure 26: Test 3 String input in vacancy text field of display in part time staff hire

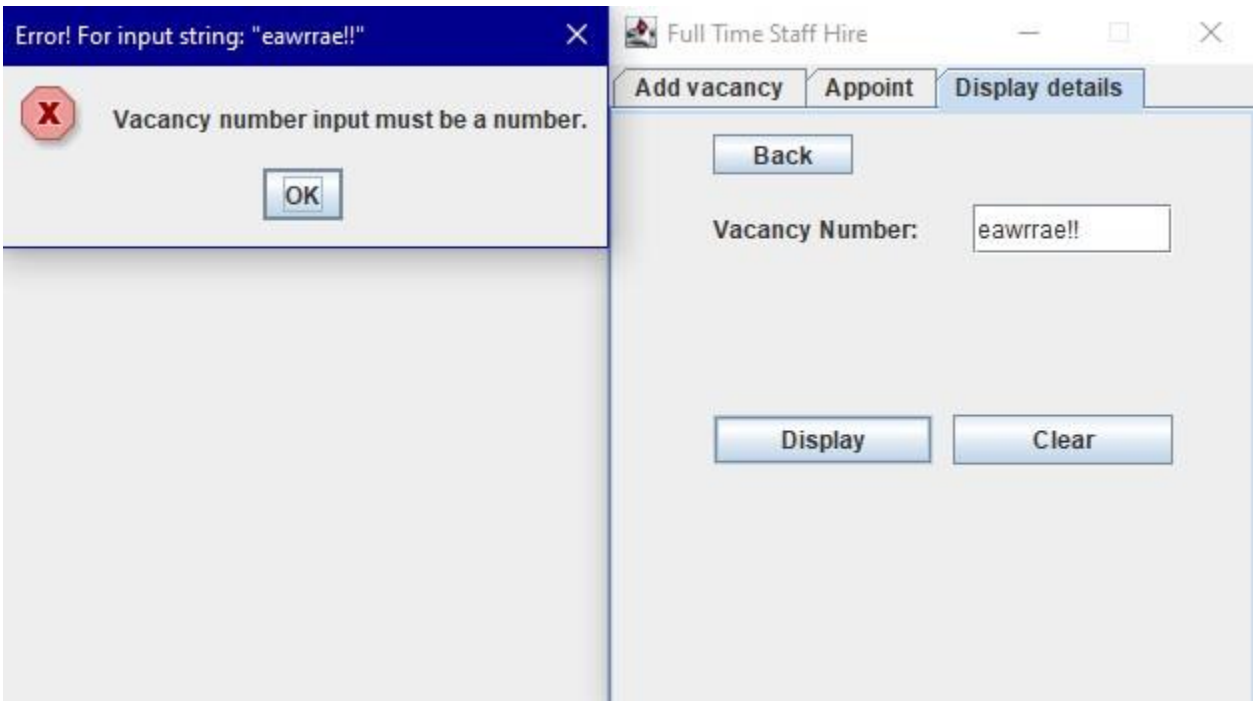

Figure 27: Test 3 String input in vacancy text field of display vacancy in full time staff hire

Error detection and solution

During developing a program various errors can occur and those errors can seriously lengthen the time needed for the development of the program. Some of the errors that a programmer can encounter are given below:

1. Syntax error
2. Logical / Semantics error
3. Runtime error

Syntax error

A punctuation mistake is an error in the grammatical/syntax in the utilization of the programming language. Since compiler requires the ideal usage of syntax to compile the program, any parts of the code that don't cling to the grammar dictated by the programming language will deliver a syntax error. Few examples that can cause syntax error are given below:

- Misspelled variable and function names
- Incorrect formatting of selection and loop statements
- Missing curly braces, parenthesis, and semicolons
- Splitting String in two lines

Syntax error

```
if(!found) {
    JOptionPane.showMessageDialog(frameFullTimeStaffHire, "Vacancy number
    not found.", "Error!", JOptionPane.ERROR_MESSAGE);
    }
```

Figure 28: Syntax error(1)

Syntax error correction

```
if(!found) {
    JOptionPane.showMessageDialog(frameFullTimeStaffHire, "Vacancy number not found.", "Error!", JOptionPane.ERROR_MESSAGE);
    }
```

Figure 29: Syntax error(2)

Logical / Semantics error

A logical error can be portrayed as a blunder that makes the program run erroneously; it is generally brought about by the defect in your program's structure. It is a kind of runtime mistake that may deliver a wrong output or may make a program crash while running. Few of the examples of logical error are given below:

- Wrong use of operators
- Opening and using data from the wrong file

Logical error

```
if(vacancyNumber != 0 || workingHour != 0 || wagesPerHour != 0) {
    int size = arrayListStaffHire.size();
```

Figure 30: Logical error(1)

Logical error correction

```
if(vacancyNumber != 0 && workingHour != 0 && wagesPerHour != 0) {
    int size = arrayListStaffHire.size();
```

Figure 31: Logical error(2)

In the above example, AND operator should have been used but due to the logical error OR operator was used which was later corrected.

A semantic error occurs when a programmer uses program statements improperly causing the error. It may be detected by compile or during run-time.

Semantics error

```
if(vacancyNumber != 0 && workingHour != 0 && wagesPerHour != 0) {
    int vacancyNumber = getPartTimeTfVacancyNumber();
```

Figure 32: Semantics error(1)

Semantics error correction

```
int vacancyNumber = getPartTimeTfVacancyNumber();
if(vacancyNumber != 0 && workingHour != 0 && wagesPerHour != 0) {
```

Figure 33: Semantics error(2)

In the above example, the vacancyNumber variable is out of if condition's scope causing the out of scope error during compilation.

Runtime error

An error that occurs during the execution of a program rather than the compilation of a program is a runtime error. Unforeseen conditions usually cause runtime errors. E.g. wrong input of data from the user. Also, running out of memory can cause a runtime error. Some examples of runtime error are: There are many runtime errors some of them are given below:

- Logical runtime error causing the output to vary from the intended output
- Wrong datatype input by the user
- Out of memory

Runtime error

```
if (s.getVacancyNumber() == vacancyNumber && p.getTerminated() == false) {
    p.terminatePartTimeStaff();
    arrayListStaffHire.remove(s);
    staffTableRowRemover(vacancyNumber);
    found = true;
    JOptionPane.showMessageDialog(framePartTimeStaffHire, p.getStaffName() + " terminated.", "Termination.",JOptionPane.INFORMATION_MESSAGE);
    clearTextPartTime();
    break;
```

Figure 34: Runtime error(1)

Runtime error correction

```
if (s.getVacancyNumber() == vacancyNumber && p.getTerminated() == false) {
    String name = p.getStaffName();
    p.terminatePartTimeStaff();
    arrayListStaffHire.remove(s);
    staffTableRowRemover(vacancyNumber);
    found = true;
    JOptionPane.showMessageDialog(framePartTimeStaffHire, name + " terminated.", "Termination.",JOptionPane.INFORMATION_MESSAGE);
    clearTextPartTime();
    break;
}
```

Figure 35: Runtime error(2)

In the above example, the program compiles but the method to call staff name was called after terminating the staff which causes the staff name to be set as null or empty resulting in different outputs than what was intended. Creating a variable before terminating the staff and storing the staff's name in it fixes the problem or showing the information message before terminating the staff also works fine.

Conclusion

The Swing and Abstract Windows Toolkit was used to complete this task. Swing is the primary widget toolkit in Java. Swing has been primarily used because it is easy to bundle the Java program and Swing application. Both Swing and AWT are platform-independent and can run the same UI on multiple platforms without any hitch. The components shown by the Swing depend upon the view of the operating system but the usage and other factors are not affected by it.

During this coursework completion process, the getter method for the String variable posed a challenge. The null pointer exception was not caught by the try catch method and an empty string was returned by the getter method which was then inputted while adding a vacancy or appointing a staff. A fix to this problem was throwing a custom exception or throwing a null pointer exception manually if String was found to be empty, but most of the articles explicitly stated not to stop the method using such procedures. It seems the usage of throw must not be done other than for its intended purpose of throwing exceptions. Also, creating many buttons of the same type had to be done to accommodate the same type of button in different panels, as Swing does not provide the flexibility of adding the same button in multiple places. Windows event listener was used to listen to the window close event and was very useful to operate when a JFrame was closed.

Swing and AWT are very versatile and easy to use, they belong to the Oracle Corporation, being of the same company means that they are easy to use together and as expected it ran smoothly as hoped. Although Swing and AWT have numerous benefits it also has its flaws. Some of the important and needed features are missing and changing the background of JFrame or JPanel when nested with other components is a hassle. The JTable was very confusing and lacking. Also, Java methods with return type were hard to stop from executing when certain conditions were not met, the usage of void return type

was then done to achieve the desired result. Java as a whole is a very wholesome programming language but still, it can be lacking at times.

Bibliography

Ahmed, H., 2018. *Four Pillars of Object Oriented Programming (OOP) - Hamza Ahmed (Bidchol) - Medium.* [Online] Available at: https://medium.com/@hamzzza.ahmed95/four-pillars-of-object-oriented-programming-oop-e8d7822aa219 [Accessed 7 4 2020].

Statler, T., 2019. *How Does Java Work? A Concise Guide - Comp Sci Central.* [Online] Available at: https://compscicentral.com/how-does-java-work/ [Accessed 7 4 2020].

Tutorialspoint, 2016. *Java - Interfaces - Tutorialspoint.* [Online] Available at: https://www.tutorialspoint.com/java/java_interfaces.htm [Accessed 7 4 2020].

Appendix

```
import javax.swing.JFrame;
import javax.swing.JButton;
import java.util.ArrayList;
import javax.swing.JTextField;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;
import javax.swing.JTabbedPane;
import javax.swing.JMenuBar;
import javax.swing.JMenu;
import javax.swing.JMenuItem;
import javax.swing.JDialog;
import javax.swing.SwingConstants;

import java.awt.Cursor;
import java.awt.FlowLayout;
import java.awt.BorderLayout;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

public class INGNepal implements ActionListener {
```

```
    private boolean partTimeWindowCheck, fullTimeWindowCheck = false;

    private JFrame frameMainMenu, framePartTimeStaffHire, frameFullTimeStaffHire;

    private JTextField tfPartTimeName, tfPartTimeDesignation, tfPartTimeJobType,
tfPartTimeQualification, tfPartTimeWorkingHour, tfPartTimeWagesPerHour,
tfPartTimeShifts, tfPartTimeJoiningDate,
tfPartTimeAppointedBy,tfPartTimeVacancyNumberVacancyPanel,
tfPartTimeVacancyNumberAppointPanel, tfPartTimeVacancyNumberDisplayPanel;

    private JTextField tfFullTimeName, tfFullTimeDesignation, tfFullTimeJobType,
tfFullTimeQualification, tfFullTimeWorkingHour, tfFullTimeSalary, tfFullTimeJoiningDate,
tfFullTimeAppointedBy, tfFullTimeVacancyNumberVacancyPanel,
tfFullTimeVacancyNumberAppointPanel, tfFullTimeVacancyNumberDisplayPanel;


    private JTable staffTable, vacancyTable;

    private DefaultTableModel dtmStaff, dtmVacancy;

    private ArrayList <StaffHire> arrayListStaffHire = new ArrayList <StaffHire>();

    private JTabbedPane tpPartTimeForm, tpFullTimeForm;


    INGNepal () {
        frameMainMenu = new JFrame("ING Nepal");
        frameMainMenu.setSize(600, 600);
        frameMainMenu.setLocationRelativeTo(null);
        frameMainMenu.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
        frameMainMenu.setResizable(true);
        frameMainMenu.addWindowListener(new WindowAdapter () {
            public void windowClosing(WindowEvent we) {
                if (partTimeWindowCheck == true) {
                    framePartTimeStaffHire.requestFocus();
                    framePartTimeStaffHire.setState(JFrame.NORMAL);
                }
                else if (fullTimeWindowCheck == true) {
                    frameFullTimeStaffHire.requestFocus();
```

```
                frameFullTimeStaffHire.setState(JFrame.NORMAL);
            }
            else if (partTimeWindowCheck == false && fullTimeWindowCheck == false) {
                System.exit(0);
            }
        }
    });

    Cursor cursor = new Cursor(Cursor.HAND_CURSOR);

    JMenuBar mb = new JMenuBar();
    JMenu menu = new JMenu("Menu");
    JMenuItem menuItemAbout = new JMenuItem("About");
    menuItemAbout.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent ae) {
            JDialog jd = new JDialog(frameMainMenu, "About");
            jd.setSize(250, 150);

            JLabel l = new JLabel("<html>ING Nepal v1.0.0<br>Developed by : Ashesh Rai</html>", SwingConstants.CENTER);

            jd.add(l);

            jd.setVisible(true);
            jd.setLocationRelativeTo(null);
        }
    });
    JMenuItem menuItemExit = new JMenuItem("Exit");
    menuItemExit.addActionListener(new ActionListener() {
```

```
    public void actionPerformed(ActionEvent ae) {
        System.exit(0);
    }
});
menu.add(menuItemAbout);
menu.add(menuItemExit);
mb.add(menu);

JButton buttonPartTimeForm = new JButton("Part Time Operations");
buttonPartTimeForm.setCursor(cursor);
buttonPartTimeForm.addActionListener (new ActionListener() {
    public void actionPerformed (ActionEvent ae) {
        if (partTimeWindowCheck == false && fullTimeWindowCheck == false) {
            partTimeForm();
            partTimeWindowCheck = true;
        }
        else if (fullTimeWindowCheck == true) {
            frameFullTimeStaffHire.requestFocus();
            frameFullTimeStaffHire.setState(JFrame.NORMAL);
        }
        else {
            framePartTimeStaffHire.requestFocus();
            framePartTimeStaffHire.setState(JFrame.NORMAL);
        }
    }
}
);
JButton buttonFullTimeForm = new JButton("Full Time Operations");
```

```
        buttonFullTimeForm.setCursor(cursor);
        buttonFullTimeForm.addActionListener (new ActionListener() {
            public void actionPerformed (ActionEvent ae) {
                if (fullTimeWindowCheck == false && partTimeWindowCheck == false) {
                    fullTimeForm();
                    fullTimeWindowCheck = true;
                }
                else if (partTimeWindowCheck == true) {
                    framePartTimeStaffHire.requestFocus();
                    framePartTimeStaffHire.setState(JFrame.NORMAL);
                }
                else {
                    frameFullTimeStaffHire.requestFocus();
                    frameFullTimeStaffHire.setState(JFrame.NORMAL);
                }
            }
        });

        JPanel mainMenuButtonPanel = new JPanel();
        mainMenuButtonPanel.setLayout(new FlowLayout());
        mainMenuButtonPanel.add(buttonPartTimeForm);
        mainMenuButtonPanel.add(buttonFullTimeForm);

        staffTable = new JTable();
        dtmStaff = new DefaultTableModel(new Object[]{"Vacancy No.", "Name",
"Employment Type"}, 0);
        staffTable.setModel(dtmStaff);
        JScrollPane staffTableScrollPane = new JScrollPane();
```

```
        staffTableScrollPane.setCorner(JScrollPane.UPPER_RIGHT_CORNER,
mainMenuButtonPanel);
        staffTableScrollPane.setViewportView(staffTable);


        vacancyTable = new JTable();
        dtmVacancy = new DefaultTableModel(new Object[]{"Vacancy No.", "Employment
Type"}, 0);
        vacancyTable.setModel(dtmVacancy);
        JScrollPane vacancyTableScrollPane = new JScrollPane();
        vacancyTableScrollPane.setCorner(JScrollPane.UPPER_RIGHT_CORNER,
mainMenuButtonPanel);
        vacancyTableScrollPane.setViewportView(vacancyTable);


        JPanel panelStaffTbl = new JPanel();
        panelStaffTbl.setLayout(new BorderLayout());
        panelStaffTbl.add(staffTableScrollPane, BorderLayout.CENTER);
        JPanel panelVacancyTbl = new JPanel();
        panelVacancyTbl.setLayout(new BorderLayout());
        panelVacancyTbl.add(vacancyTableScrollPane, BorderLayout.CENTER);


        JTabbedPane tpMain = new JTabbedPane();
        tpMain.add("Staffs", panelStaffTbl);
        tpMain.add("Vacancy", panelVacancyTbl);


        frameMainMenu.add(tpMain, BorderLayout.CENTER);
        frameMainMenu.add(mainMenuButtonPanel, BorderLayout.SOUTH);
        frameMainMenu.add(mb, BorderLayout.NORTH);


}
```

```
    public void partTimeForm () {
        framePartTimeStaffHire = new JFrame("Part Time Staff Hire");
        framePartTimeStaffHire.setSize(340, 380);

framePartTimeStaffHire.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        framePartTimeStaffHire.setLayout(new BorderLayout());
        framePartTimeStaffHire.setResizable(false);
        framePartTimeStaffHire.setAlwaysOnTop(true);
        framePartTimeStaffHire.setLocationRelativeTo(null);
        framePartTimeStaffHire.addWindowListener(new WindowAdapter() {
        @Override
            public void windowClosing(WindowEvent we) {
                partTimeWindowCheck = false;
            }
        });

        if(!partTimeWindowCheck){
            framePartTimeStaffHire.setVisible(true);
        }

        Cursor cursor = new Cursor(Cursor.HAND_CURSOR);

        JLabel lblPartTimeName = new JLabel("Name:");
        JLabel lblPartTimeDesignation = new JLabel("Designation:");
        JLabel lblPartTimeJobType = new JLabel("Job Type:");
        JLabel lblPartTimeQualification = new JLabel("Qualification:");
        JLabel lblPartTimeWorkingHour = new JLabel("Working Hour:");
        JLabel lblPartTimeWagesPerHour = new JLabel("Wages Per Hour:");
```

```
JLabel lblPartTimeShifts = new JLabel("Shifts:");
JLabel lblPartTimeJoiningDate = new JLabel("Joining Date:");
JLabel lblPartTimeAppointedBy = new JLabel("Appointed By:");
JLabel lblPartTimeVacancyNumberVacancyPanel = new JLabel("Vacancy Number:");
JLabel lblPartTimeVacancyNumberAppointPanel = new JLabel("Vacancy Number:");
JLabel lblPartTimeVacancyNumberDisplayPanel = new JLabel("Vacancy Number:");

tfPartTimeName = new JTextField();
tfPartTimeDesignation = new JTextField();
tfPartTimeJobType = new JTextField();
tfPartTimeQualification = new JTextField();
tfPartTimeWorkingHour = new JTextField();
tfPartTimeWagesPerHour = new JTextField();
tfPartTimeShifts = new JTextField();
tfPartTimeJoiningDate = new JTextField();
tfPartTimeAppointedBy = new JTextField();
tfPartTimeVacancyNumberVacancyPanel = new JTextField();
tfPartTimeVacancyNumberAppointPanel = new JTextField();
tfPartTimeVacancyNumberDisplayPanel = new JTextField();

JButton partTimeAddVacancyButton = new JButton("Add Vacancy");
JButton partTimeClearButtonVacancyPanel = new JButton("Clear");
JButton partTimeClearButtonAppointPanel = new JButton("Clear");
JButton partTimeClearButtonDisplayPanel = new JButton("Clear");
JButton partTimeDisplayButton = new JButton("Display");
JButton partTimeAppointButton = new JButton("Appoint");
```

```
JButton partTimeBackButtonVacancyPanel = new JButton("Back");
JButton partTimeBackButtonAppointPanel = new JButton("Back");
JButton partTimeBackButtonDisplayPanel = new JButton("Back");
JButton partTimeTerminateButton = new JButton("Terminate");

JPanel vacancyPanel = new JPanel ();
vacancyPanel.setLayout(null);

lblPartTimeVacancyNumberVacancyPanel.setBounds(50, 45, 120, 25);
tfPartTimeVacancyNumberVacancyPanel.setBounds(180, 45, 100, 25);

lblPartTimeDesignation.setBounds(50, 75, 100, 25);
tfPartTimeDesignation.setBounds(180, 75, 100, 25);

lblPartTimeJobType.setBounds(50, 105, 100, 25);
tfPartTimeJobType.setBounds(180, 105, 100, 25);

lblPartTimeWorkingHour.setBounds(50, 135, 100, 25);
tfPartTimeWorkingHour.setBounds(180, 135, 100, 25);

lblPartTimeWagesPerHour.setBounds(50, 165, 100, 25);
tfPartTimeWagesPerHour.setBounds(180, 165, 100, 25);

lblPartTimeShifts.setBounds(50, 195, 100, 25);
tfPartTimeShifts.setBounds(180, 195, 100, 25);

partTimeBackButtonVacancyPanel.setBounds(50, 10, 70, 20);
partTimeBackButtonVacancyPanel.setCursor(cursor);
```

```
        partTimeBackButtonVacancyPanel.addActionListener (new ActionListener() {
            public void actionPerformed (ActionEvent ae) {
                framePartTimeStaffHire.dispose();
                partTimeWindowCheck = false;
            }
        }
        );

        partTimeAddVacancyButton.setBounds(50, 245, 110, 25);
        partTimeAddVacancyButton.setCursor(cursor);
        partTimeAddVacancyButton.addActionListener (new ActionListener() {
            public void actionPerformed (ActionEvent ae) {
                int vacancyNumber = getPartTimeTfVacancyNumber();
                String designation = getPartTimeTfDesignation();
                String jobType = getPartTimeTfJobType();
                int workingHour = getPartTimeTfWorkingHour();
                int wagesPerHour = getPartTimeTfWagesPerHour();
                String shifts = getPartTimeTfShifts();

                if(vacancyNumber != 0 && workingHour != 0 && wagesPerHour != 0 &&
!designation.isEmpty() && !jobType.isEmpty() && !shifts.isEmpty()) {
                    int size = arrayListStaffHire.size();
                    if(size > 0) {
                        for(StaffHire sh : arrayListStaffHire) {
                            int num = sh.getVacancyNumber();

                            if(num == vacancyNumber) {
```

```
                    JOptionPane.showMessageDialog(framePartTimeStaffHire,
"Vacancy number " + vacancyNumber + " already exists.", "Error!",
JOptionPane.ERROR_MESSAGE);
                    return;
                }
            }
            PartTimeStaffHire partTime = new PartTimeStaffHire (vacancyNumber,
designation, jobType, workingHour, wagesPerHour, shifts);
            arrayListStaffHire.add(partTime);
            JOptionPane.showMessageDialog(framePartTimeStaffHire, "Vacancy
number " + vacancyNumber + " added", "Successful",
JOptionPane.INFORMATION_MESSAGE);

            clearTextPartTime();
        }

        else {
            PartTimeStaffHire partTime = new PartTimeStaffHire (vacancyNumber,
designation, jobType, workingHour, wagesPerHour, shifts);
            arrayListStaffHire.add(partTime);
            JOptionPane.showMessageDialog(framePartTimeStaffHire, "Vacancy
number " + vacancyNumber + " added", "Successful",
JOptionPane.INFORMATION_MESSAGE);

            clearTextPartTime();
        }

        DefaultTableModel modelVacancyTable = (DefaultTableModel)
vacancyTable.getModel();
        modelVacancyTable.addRow(new Object[]{vacancyNumber, "Part Time"});
```

```
            }
            else if (           ) {
                JOptionPane.showMessageDialog(framePartTimeStaffHire, "Designation, job type, and shifts text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);
            }
            else if (designation.isEmpty() && jobType.isEmpty()) {
                JOptionPane.showMessageDialog(framePartTimeStaffHire, "Designation and job type text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);
            }
            else if (designation.isEmpty() && shifts.isEmpty()) {
                JOptionPane.showMessageDialog(framePartTimeStaffHire, "Designation and shifts text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);
            }
            else if (shifts.isEmpty() && jobType.isEmpty()) {
                JOptionPane.showMessageDialog(framePartTimeStaffHire, "Shifts and job type text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);
            }
            else if (designation.isEmpty()) {
                JOptionPane.showMessageDialog(framePartTimeStaffHire, "Designation text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);
            }
            else if (jobType.isEmpty()) {
                JOptionPane.showMessageDialog(framePartTimeStaffHire, "Job type text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);
            }
            else {
                JOptionPane.showMessageDialog(framePartTimeStaffHire, "Shifts text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);
            }
        }
```

```
}
);

partTimeClearButtonVacancyPanel.setBounds(170, 245, 110, 25);
partTimeClearButtonVacancyPanel.setCursor(cursor);
partTimeClearButtonVacancyPanel.addActionListener (new ActionListener() {
    public void actionPerformed (ActionEvent ae) {
        clearTextPartTime();
    }
}
);

vacancyPanel.add(lblPartTimeVacancyNumberVacancyPanel);
vacancyPanel.add(tfPartTimeVacancyNumberVacancyPanel);
vacancyPanel.add(lblPartTimeDesignation);
vacancyPanel.add(tfPartTimeDesignation);
vacancyPanel.add(lblPartTimeJobType);
vacancyPanel.add(tfPartTimeJobType);
vacancyPanel.add(lblPartTimeWorkingHour);
vacancyPanel.add(tfPartTimeWorkingHour);
vacancyPanel.add(lblPartTimeWagesPerHour);
vacancyPanel.add(tfPartTimeWagesPerHour);
vacancyPanel.add(lblPartTimeShifts);
vacancyPanel.add(tfPartTimeShifts);
vacancyPanel.add(partTimeBackButtonVacancyPanel);
vacancyPanel.add(partTimeAddVacancyButton);
vacancyPanel.add(partTimeClearButtonVacancyPanel);
```

```
JPanel appointPanel = new JPanel ();
appointPanel.setLayout(null);

lblPartTimeVacancyNumberAppointPanel.setBounds(50, 45, 120, 25);
tfPartTimeVacancyNumberAppointPanel.setBounds(180, 45, 100, 25);

lblPartTimeName.setBounds(50, 75, 100, 25);
tfPartTimeName.setBounds(180, 75, 100, 25);

lblPartTimeJoiningDate.setBounds(50, 105, 100, 25);
tfPartTimeJoiningDate.setBounds(180, 105, 100, 25);

lblPartTimeQualification.setBounds(50, 135, 100, 25);
tfPartTimeQualification.setBounds(180, 135, 100, 25);

lblPartTimeAppointedBy.setBounds(50, 165, 100, 25);
tfPartTimeAppointedBy.setBounds(180, 165, 100, 25);

partTimeBackButtonAppointPanel.setBounds(50, 10, 70, 20);
partTimeBackButtonAppointPanel.setCursor(cursor);
partTimeBackButtonAppointPanel.addActionListener (new ActionListener() {
    public void actionPerformed (ActionEvent ae) {
        framePartTimeStaffHire.dispose();
        partTimeWindowCheck = false;
    }
}
);
```

```
        partTimeAppointButton.setBounds(50, 245, 110, 25);
        partTimeAppointButton.setCursor(cursor);
        partTimeAppointButton.addActionListener (new ActionListener() {
            public void actionPerformed (ActionEvent ae) {
                String name = getPartTimeTfName();
                String joiningDate = getPartTimeTfJoiningDate();
                String qualification = getPartTimeTfQualification();
                String appointedBy = getPartTimeTfAppointedBy();
                int vacancyNumber = getPartTimeTfVacancyNumber();
                int size = arrayListStaffHire.size();
                if (vacancyNumber != 0 && !joiningDate.isEmpty() && !qualification.isEmpty()
&& !name.isEmpty() && !appointedBy.isEmpty()) {
                    if(size > 0) {
                        for(int i = 0; i < arrayListStaffHire.size(); ++i) {
                            StaffHire sh = arrayListStaffHire.get(i);
                            if (sh instanceof PartTimeStaffHire) {
                                PartTimeStaffHire p = (PartTimeStaffHire) sh;
                                int vacancyNo = sh.getVacancyNumber();
                                boolean joinCheck = p.getJoined();
                                if (vacancyNo == vacancyNumber && joinCheck == false) {
                                    p.hirePartTimeStaff(name, joiningDate, qualification,
appointedBy);
                                    arrayListStaffHire.set(i, p);
                                    JOptionPane.showMessageDialog(framePartTimeStaffHire,
name + " hired as a part time staff.", "Successful",
JOptionPane.INFORMATION_MESSAGE);

                                    DefaultTableModel modelStaffTable = (DefaultTableModel)
staffTable.getModel();
```

```
                    modelStaffTable.addRow(new Object[]{vacancyNo, name, "Part Time"});


                    vacancyTableRowRemover(vacancyNumber);


                    clearTextPartTime();
                    break;
                }
                else if (vacancyNo == vacancyNumber && joinCheck == true) {
                    JOptionPane.showMessageDialog(framePartTimeStaffHire, "Vacancy number " + vacancyNumber + " already filled.", "Error!", JOptionPane.ERROR_MESSAGE);
                    break;
                }
                else if ((size - 1) == i && vacancyNo != vacancyNumber) {
                    JOptionPane.showMessageDialog(framePartTimeStaffHire, "Vacancy number " + vacancyNumber + " not found.", "Error!", JOptionPane.ERROR_MESSAGE);
                }
            }


        }
    }
    else if (size == 0) {
        JOptionPane.showMessageDialog(framePartTimeStaffHire, "No vacancy available.", "Error!", JOptionPane.ERROR_MESSAGE);
    }
}
else if (joiningDate.isEmpty() && qualification.isEmpty() && name.isEmpty() && appointedBy.isEmpty()) {
```

```
            JOptionPane.showMessageDialog(framePartTimeStaffHire, "Joining date, qualification, name, and appointed by text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);

        }

        else if (joiningDate.isEmpty() && qualification.isEmpty() && name.isEmpty()) {

            JOptionPane.showMessageDialog(framePartTimeStaffHire, "Joining date, qualification, and name text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);

        }

        else if (joiningDate.isEmpty() && qualification.isEmpty() && appointedBy.isEmpty()) {

            JOptionPane.showMessageDialog(framePartTimeStaffHire, "Joining date, qualification, and appointed by text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);

        }

        else if (qualification.isEmpty() && name.isEmpty() && appointedBy.isEmpty()) {

            JOptionPane.showMessageDialog(framePartTimeStaffHire, "Qualification, name, and appointed by text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);

        }

        else if (joiningDate.isEmpty() && qualification.isEmpty()) {

            JOptionPane.showMessageDialog(framePartTimeStaffHire, "Joining date and qualification text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);

        }

        else if (joiningDate.isEmpty() && name.isEmpty()) {

            JOptionPane.showMessageDialog(framePartTimeStaffHire, "Joining date and name text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);

        }

        else if (joiningDate.isEmpty() && appointedBy.isEmpty()) {

            JOptionPane.showMessageDialog(framePartTimeStaffHire, "Joining date and appointed by text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);
```

```
            }
            else if (qualification.isEmpty() && name.isEmpty()) {
                JOptionPane.showMessageDialog(framePartTimeStaffHire, "Qualification and name text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);
            }
            else if (qualification.isEmpty() && appointedBy.isEmpty()) {
                JOptionPane.showMessageDialog(framePartTimeStaffHire, "Qualification and appointedBy text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);
            }
            else if (name.isEmpty() && appointedBy.isEmpty()) {
                JOptionPane.showMessageDialog(framePartTimeStaffHire, "Name and appointed by text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);
            }
            else if (joiningDate.isEmpty()) {
                JOptionPane.showMessageDialog(framePartTimeStaffHire, "Joining date text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);
            }
            else if (qualification.isEmpty()) {
                JOptionPane.showMessageDialog(framePartTimeStaffHire, "Qualification text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);
            }
            else if (name.isEmpty()) {
                JOptionPane.showMessageDialog(framePartTimeStaffHire, "Name text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);
            }
            else {
                JOptionPane.showMessageDialog(framePartTimeStaffHire, "Appointed by text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);
            }
        }
```

```
}
);

partTimeClearButtonAppointPanel.setBounds(170, 245, 110, 25);
partTimeClearButtonAppointPanel.setCursor(cursor);
partTimeClearButtonAppointPanel.addActionListener (new ActionListener() {
    public void actionPerformed (ActionEvent ae) {
        clearTextPartTime();
    }
}
);

appointPanel.add(lblPartTimeVacancyNumberAppointPanel);
appointPanel.add(tfPartTimeVacancyNumberAppointPanel);
appointPanel.add(lblPartTimeName);
appointPanel.add(tfPartTimeName);
appointPanel.add(lblPartTimeJoiningDate);
appointPanel.add(tfPartTimeJoiningDate);
appointPanel.add(lblPartTimeQualification);
appointPanel.add(tfPartTimeQualification);
appointPanel.add(lblPartTimeAppointedBy);
appointPanel.add(tfPartTimeAppointedBy);
appointPanel.add(partTimeBackButtonAppointPanel);
appointPanel.add(partTimeAppointButton);
appointPanel.add(partTimeClearButtonAppointPanel);

JPanel displayPanel = new JPanel ();
displayPanel.setLayout(null);
```

```
lblPartTimeVacancyNumberDisplayPanel.setBounds(50, 45, 120, 25);
tfPartTimeVacancyNumberDisplayPanel.setBounds(180, 45, 100, 25);

partTimeBackButtonDisplayPanel.setBounds(50, 10, 70, 20);
partTimeBackButtonDisplayPanel.setCursor(cursor);
partTimeBackButtonDisplayPanel.addActionListener (new ActionListener() {
    public void actionPerformed (ActionEvent ae) {
        framePartTimeStaffHire.dispose();
        partTimeWindowCheck = false;
    }
}
);

partTimeClearButtonDisplayPanel.setBounds(170, 150, 110, 25);
partTimeClearButtonDisplayPanel.setCursor(cursor);
partTimeClearButtonDisplayPanel.addActionListener (new ActionListener() {
    public void actionPerformed (ActionEvent ae) {
        clearTextPartTime();
    }
}
);

partTimeDisplayButton.setBounds(50, 150, 110, 25);
partTimeDisplayButton.setCursor(cursor);
partTimeDisplayButton.addActionListener (new ActionListener() {
    public void actionPerformed (ActionEvent ae) {
        int vacancyNo = getPartTimeTfVacancyNumber();
```

```
            boolean found = false;


            if (vacancyNo != 0) {
                for(StaffHire s : arrayListStaffHire) {
                    if(s instanceof PartTimeStaffHire) {
                        PartTimeStaffHire p = (PartTimeStaffHire) s;


                        if (s.getVacancyNumber() == vacancyNo) {
                            p.displayDetails();
                            found = true;
                            JOptionPane.showMessageDialog(framePartTimeStaffHire, "Name: " + p.getStaffName() + "\nDesignation: " + s.getDesignation() + "\nJob Type: " + s.getJobType() + "\nQualification: " + p.getQualification() + "\nWorking Hour: " + p.getWorkingHour() + "\nWages Per Hour: " + p.getWagesPerHour() + "\nShifts: " + p.getShifts() + "\nJoining Date: " + p.getJoiningDate() + "\nAppointed By: " + p.getAppointedBy(), "Staff Info.",JOptionPane.INFORMATION_MESSAGE);
                            clearTextPartTime();
                            break;
                        }
                    }
                }
                if(!found) {
                JOptionPane.showMessageDialog(framePartTimeStaffHire, "Vacancy number not found.", "Error!", JOptionPane.ERROR_MESSAGE);
                }
            }


        }
    }
    );
```

```
        partTimeTerminateButton.setBounds(115, 200, 100, 25);
        partTimeTerminateButton.setCursor(cursor);
        partTimeTerminateButton.addActionListener (new ActionListener() {
            public void actionPerformed (ActionEvent ae) {
                int vacancyNumber = getPartTimeTfVacancyNumber();
                boolean found = false;
                boolean terminateCheck = false;
                if (vacancyNumber != 0) {
                    for(StaffHire s : arrayListStaffHire) {
                        if(s instanceof PartTimeStaffHire) {
                            PartTimeStaffHire p = (PartTimeStaffHire) s;
                            if (s.getVacancyNumber() == vacancyNumber && p.getTerminated() ==
false) {
                                String name = p.getStaffName();
                                p.terminatePartTimeStaff();
                                arrayListStaffHire.remove(s);
                                staffTableRowRemover(vacancyNumber);
                                found = true;
                                JOptionPane.showMessageDialog(framePartTimeStaffHire, name +
" terminated.", "Termination.",JOptionPane.INFORMATION_MESSAGE);
                                clearTextPartTime();
                                break;
                            }
                        }
                    }
                        if(!found) {
                        JOptionPane.showMessageDialog(framePartTimeStaffHire,      "Vacancy
number " + vacancyNumber + " not found.", "Error!", JOptionPane.ERROR_MESSAGE);
```

```
            }
        }

    }
}

);


displayPanel.add(lblPartTimeVacancyNumberDisplayPanel);
displayPanel.add(tfPartTimeVacancyNumberDisplayPanel);
displayPanel.add(partTimeBackButtonDisplayPanel);
displayPanel.add(partTimeClearButtonDisplayPanel);
displayPanel.add(partTimeDisplayButton);
displayPanel.add(partTimeTerminateButton);

tpPartTimeForm = new JTabbedPane();
tpPartTimeForm.add("Add vacancy", vacancyPanel);
tpPartTimeForm.add("Appoint", appointPanel);
tpPartTimeForm.add("Display and terminate", displayPanel);

framePartTimeStaffHire.add(tpPartTimeForm, BorderLayout.CENTER);
}

public void fullTimeForm () {
frameFullTimeStaffHire = new JFrame("Full Time Staff Hire");
frameFullTimeStaffHire.setSize(340, 360);

frameFullTimeStaffHire.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
```

```
        frameFullTimeStaffHire.setLayout(new BorderLayout());
        frameFullTimeStaffHire.setResizable(false);
        frameFullTimeStaffHire.setAlwaysOnTop(true);
        frameFullTimeStaffHire.setLocationRelativeTo(null);
        frameFullTimeStaffHire.addWindowListener(new WindowAdapter() {
        @Override
            public void windowClosing(WindowEvent we) {
                fullTimeWindowCheck = false;
            }
        });

        if(!fullTimeWindowCheck){
            frameFullTimeStaffHire.setVisible(true);
        }

        Cursor cursor = new Cursor(Cursor.HAND_CURSOR);

        JLabel lblFullTimeName = new JLabel("Name:");
        JLabel lblFullTimeDesignation = new JLabel("Designation:");
        JLabel lblFullTimeJobType = new JLabel("Job Type:");
        JLabel lblFullTimeQualification = new JLabel("Qualification:");
        JLabel lblFullTimeWorkingHour = new JLabel("Working Hour:");
        JLabel lblFullTimeSalary = new JLabel("Salary:");
        JLabel lblFullTimeJoiningDate = new JLabel("Joining Date:");
        JLabel lblFullTimeAppointedBy = new JLabel("Appointed By:");
        JLabel lblFullTimeVacancyNumberVacancyPanel = new JLabel("Vacancy
Number:");
        JLabel lblFullTimeVacancyNumberAppointPanel = new JLabel("Vacancy
Number:");
```

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

tfFullTimeName = new JTextField();
tfFullTimeDesignation = new JTextField();
tfFullTimeJobType = new JTextField();
tfFullTimeQualification = new JTextField();
tfFullTimeWorkingHour = new JTextField();
tfFullTimeSalary = new JTextField();
tfFullTimeJoiningDate = new JTextField();
tfFullTimeAppointedBy = new JTextField();
tfFullTimeVacancyNumberVacancyPanel = new JTextField();
tfFullTimeVacancyNumberAppointPanel = new JTextField();
tfFullTimeVacancyNumberDisplayPanel = new JTextField();

JButton fullTimeAddVacancyButton = new JButton("Add Vacancy");
JButton fullTimeClearButtonVacancyPanel = new JButton("Clear");
JButton fullTimeClearButtonAppointPanel = new JButton("Clear");
JButton fullTimeClearButtonDisplayPanel = new JButton("Clear");
JButton fullTimeDisplayButton = new JButton("Display");
JButton fullTimeAppointButton = new JButton("Appoint");
JButton fullTimeBackButtonVacancyPanel = new JButton("Back");
JButton fullTimeBackButtonAppointPanel = new JButton("Back");
JButton fullTimeBackButtonDisplayPanel = new JButton("Back");

JPanel vacancyPanel = new JPanel ();
vacancyPanel.setLayout(null);

lblFullTimeVacancyNumberVacancyPanel.setBounds(50, 45, 120, 25);
```

```
tfFullTimeVacancyNumberVacancyPanel.setBounds(180, 45, 100, 25);

lblFullTimeDesignation.setBounds(50, 75, 100, 25);
tfFullTimeDesignation.setBounds(180, 75, 100, 25);

lblFullTimeJobType.setBounds(50, 105, 100, 25);
tfFullTimeJobType.setBounds(180, 105, 100, 25);

lblFullTimeWorkingHour.setBounds(50, 135, 100, 25);
tfFullTimeWorkingHour.setBounds(180, 135, 100, 25);

lblFullTimeSalary.setBounds(50, 165, 100, 25);
tfFullTimeSalary.setBounds(180, 165, 100, 25);

fullTimeBackButtonVacancyPanel.setBounds(50, 10, 70, 20);
fullTimeBackButtonVacancyPanel.setCursor(cursor);
fullTimeBackButtonVacancyPanel.addActionListener (new ActionListener() {
    public void actionPerformed (ActionEvent ae) {
        frameFullTimeStaffHire.dispose();
        fullTimeWindowCheck = false;
    }
}
);

fullTimeAddVacancyButton.setBounds(50, 215, 110, 25);
fullTimeAddVacancyButton.setCursor(cursor);
fullTimeAddVacancyButton.addActionListener (new ActionListener (){
    public void actionPerformed (ActionEvent ae) {
```

```
            int vacancyNumber = getFullTimeTfVacancyNumber();
            String designation = getFullTimeTfDesignation();
            String jobType = getFullTimeTfJobType();
            int workingHour = getFullTimeTfWorkingHour();
            int salary = getFullTimeTfSalary();


            if (vacancyNumber != 0 && workingHour != 0 && salary != 0 &&
!designation.isEmpty() && !jobType.isEmpty()) {
                int size = arrayListStaffHire.size();
                if (size > 0) {
                    for(StaffHire sh : arrayListStaffHire) {
                        int num = sh.getVacancyNumber();
                        if(num == vacancyNumber) {
                            JOptionPane.showMessageDialog(frameFullTimeStaffHire,
"Vacancy number " + vacancyNumber + " already exists.", "Error!",
JOptionPane.ERROR_MESSAGE);
                            return;
                        }


                    }
                    FullTimeStaffHire fullTime = new FullTimeStaffHire (vacancyNumber,
designation, jobType, workingHour, salary);
                    arrayListStaffHire.add(fullTime);
                    JOptionPane.showMessageDialog(frameFullTimeStaffHire, "Vacancy
number " + vacancyNumber + " added", "Successful",
JOptionPane.INFORMATION_MESSAGE);


                    clearTextFullTime();
                }
```

```
                else {
                    FullTimeStaffHire fullTime = new FullTimeStaffHire (vacancyNumber, designation, jobType, workingHour, salary);
                    arrayListStaffHire.add(fullTime);
                    JOptionPane.showMessageDialog(frameFullTimeStaffHire, "Vacancy number " + vacancyNumber + " added", "Successful", JOptionPane.INFORMATION_MESSAGE);

                    clearTextFullTime();
                }

                DefaultTableModel modelVacancyTable = (DefaultTableModel) vacancyTable.getModel();
                modelVacancyTable.addRow(new Object[]{vacancyNumber, "Full Time"});
            }
            else if (designation.isEmpty() && jobType.isEmpty()) {
                JOptionPane.showMessageDialog(frameFullTimeStaffHire, "Designation and job type text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);
            }
            else if (designation.isEmpty()) {
                JOptionPane.showMessageDialog(frameFullTimeStaffHire, "Designation text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);
            }
            else {
                JOptionPane.showMessageDialog(frameFullTimeStaffHire, "Job type text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);
            }
        }
    }
    );
```

```
fullTimeClearButtonVacancyPanel.setBounds(170, 215, 110, 25);
fullTimeClearButtonVacancyPanel.setCursor(cursor);
fullTimeClearButtonVacancyPanel.addActionListener (new ActionListener (){
    public void actionPerformed (ActionEvent ae) {
        clearTextFullTime();
    }
}
);

vacancyPanel.add(lblFullTimeVacancyNumberVacancyPanel);
vacancyPanel.add(tfFullTimeVacancyNumberVacancyPanel);
vacancyPanel.add(lblFullTimeDesignation);
vacancyPanel.add(tfFullTimeDesignation);
vacancyPanel.add(lblFullTimeJobType);
vacancyPanel.add(tfFullTimeJobType);
vacancyPanel.add(lblFullTimeWorkingHour);
vacancyPanel.add(tfFullTimeWorkingHour);
vacancyPanel.add(lblFullTimeSalary);
vacancyPanel.add(tfFullTimeSalary);
vacancyPanel.add(fullTimeBackButtonVacancyPanel);
vacancyPanel.add(fullTimeAddVacancyButton);
vacancyPanel.add(fullTimeClearButtonVacancyPanel);

JPanel appointPanel = new JPanel();
appointPanel.setLayout(null);

lblFullTimeVacancyNumberAppointPanel.setBounds(50, 45, 120, 25);
tfFullTimeVacancyNumberAppointPanel.setBounds(180, 45, 100, 25);
```

```
lblFullTimeName.setBounds(50, 75, 100, 25);
tfFullTimeName.setBounds(180, 75, 100, 25);

lblFullTimeJoiningDate.setBounds(50, 105, 100, 25);
tfFullTimeJoiningDate.setBounds(180, 105, 100, 25);

lblFullTimeQualification.setBounds(50, 135, 100, 25);
tfFullTimeQualification.setBounds(180, 135, 100, 25);

lblFullTimeAppointedBy.setBounds(50, 165, 100, 25);
tfFullTimeAppointedBy.setBounds(180, 165, 100, 25);

fullTimeBackButtonAppointPanel.setBounds(50, 10, 70, 20);
fullTimeBackButtonAppointPanel.setCursor(cursor);
fullTimeBackButtonAppointPanel.addActionListener (new ActionListener() {
    public void actionPerformed (ActionEvent ae) {
        frameFullTimeStaffHire.dispose();
        fullTimeWindowCheck = false;
    }
}
);

    fullTimeAppointButton.setBounds(50, 215, 110, 25);
    fullTimeAppointButton.setCursor(cursor);
    fullTimeAppointButton.addActionListener (new ActionListener (){
        public void actionPerformed(ActionEvent ae){
            String name = getFullTimeTfName();
```

```
                String joiningDate = getFullTimeTfJoiningDate();
                String qualification = getFullTimeTfQualification();
                String appointedBy = getFullTimeTfAppointedBy();
                int vacancyNumber = getFullTimeTfVacancyNumber();
                int size = arrayListStaffHire.size();
                if (vacancyNumber != 0 && !name.isEmpty() && !joiningDate.isEmpty() &&
!qualification.isEmpty() && !appointedBy.isEmpty()) {
                    if(size > 0) {
                        for(int i = 0; i < arrayListStaffHire.size(); ++i) {
                            StaffHire sh = arrayListStaffHire.get(i);
                            if (sh instanceof FullTimeStaffHire) {
                                FullTimeStaffHire f = (FullTimeStaffHire) sh;
                                int vacancyNo = sh.getVacancyNumber();
                                boolean joinedCheck = f.getJoined();
                                if(vacancyNo == vacancyNumber && joinedCheck == false) {

                                    f.hireFullTimeStaff(name, joiningDate, qualification,
appointedBy);
                                    arrayListStaffHire.set(i, f);
                                    JOptionPane.showMessageDialog(frameFullTimeStaffHire,
name + " hired as a full-time staff.", "Successful",
JOptionPane.INFORMATION_MESSAGE);

                                    DefaultTableModel modelStaffTable = (DefaultTableModel)
staffTable.getModel();
                                    modelStaffTable.addRow(new Object[]{vacancyNo, name, "Full
Time"});

                                    vacancyTableRowRemover(vacancyNumber);
```

```
                    clearTextFullTime();
                    break;
                }
                else if (vacancyNo == vacancyNumber && joinedCheck == true) {
                    JOptionPane.showMessageDialog(frameFullTimeStaffHire,
"Vacancy number " + vacancyNumber + " already filled.", "Error!",
JOptionPane.ERROR_MESSAGE);
                    break;
                }
                else if ((size - 1) == i && vacancyNo != vacancyNumber) {
                    JOptionPane.showMessageDialog(frameFullTimeStaffHire,
"Vacancy number " + vacancyNumber + " not found.", "Error!",
JOptionPane.ERROR_MESSAGE);
                }
            }

        }
    }
    else if (size == 0) {
        JOptionPane.showMessageDialog(framePartTimeStaffHire, "No
vacancy available.", "Error!", JOptionPane.ERROR_MESSAGE);
    }
}
else if (joiningDate.isEmpty() && qualification.isEmpty() && name.isEmpty()
&& appointedBy.isEmpty()) {
    JOptionPane.showMessageDialog(frameFullTimeStaffHire, "Joining
date, qualification, name, and appointed by text field cannot be empty.", "Error",
JOptionPane.ERROR_MESSAGE);
}
else if (joiningDate.isEmpty() && qualification.isEmpty() && name.isEmpty())
{
```

```
                JOptionPane.showMessageDialog(frameFullTimeStaffHire, "Joining date, qualification, and name text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);

            }

            else if (joiningDate.isEmpty() && qualification.isEmpty() && appointedBy.isEmpty()) {

                JOptionPane.showMessageDialog(frameFullTimeStaffHire, "Joining date, qualification, and appointed by text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);

            }

            else if (qualification.isEmpty() && name.isEmpty() && appointedBy.isEmpty()) {

                JOptionPane.showMessageDialog(frameFullTimeStaffHire, "Qualification, name, and appointed by text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);

            }

            else if (joiningDate.isEmpty() && qualification.isEmpty()) {

                JOptionPane.showMessageDialog(frameFullTimeStaffHire, "Joining date and qualification text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);

            }

            else if (joiningDate.isEmpty() && name.isEmpty()) {

                JOptionPane.showMessageDialog(frameFullTimeStaffHire, "Joining date and name text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);

            }

            else if (joiningDate.isEmpty() && appointedBy.isEmpty()) {

                JOptionPane.showMessageDialog(frameFullTimeStaffHire, "Joining date and appointed by text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);

            }

            else if (qualification.isEmpty() && name.isEmpty()) {

                JOptionPane.showMessageDialog(frameFullTimeStaffHire, "Qualification and name text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);

            }
```

```
                else if (qualification.isEmpty() && appointedBy.isEmpty()) {
                    JOptionPane.showMessageDialog(frameFullTimeStaffHire, "Qualification and appointedBy text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);
                }
                else if (name.isEmpty() && appointedBy.isEmpty()) {
                    JOptionPane.showMessageDialog(frameFullTimeStaffHire, "Name and appointed by text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);
                }
                else if (joiningDate.isEmpty()) {
                    JOptionPane.showMessageDialog(frameFullTimeStaffHire, "Joining date text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);
                }
                else if (qualification.isEmpty()) {
                    JOptionPane.showMessageDialog(frameFullTimeStaffHire, "Qualification text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);
                }
                else if (name.isEmpty()) {
                    JOptionPane.showMessageDialog(frameFullTimeStaffHire, "Name text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);
                }
                else {
                    JOptionPane.showMessageDialog(frameFullTimeStaffHire, "Appointed by text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);
                }
            }
        }
        );

        fullTimeClearButtonAppointPanel.setBounds(170, 215, 110, 25);
        fullTimeClearButtonAppointPanel.setCursor(cursor);
```

```
fullTimeClearButtonAppointPanel.addActionListener (new ActionListener (){
    public void actionPerformed (ActionEvent ae) {
        clearTextFullTime();
    }
}
);

appointPanel.add(lblFullTimeVacancyNumberAppointPanel);
appointPanel.add(tfFullTimeVacancyNumberAppointPanel);
appointPanel.add(lblFullTimeName);
appointPanel.add(tfFullTimeName);
appointPanel.add(lblFullTimeJoiningDate);
appointPanel.add(tfFullTimeJoiningDate);
appointPanel.add(lblFullTimeQualification);
appointPanel.add(tfFullTimeQualification);
appointPanel.add(lblFullTimeAppointedBy);
appointPanel.add(tfFullTimeAppointedBy);
appointPanel.add(fullTimeBackButtonAppointPanel);
appointPanel.add(fullTimeAppointButton);
appointPanel.add(fullTimeClearButtonAppointPanel);

JPanel displayPanel = new JPanel();
displayPanel.setLayout(null);

lblFullTimeVacancyNumberDisplayPanel.setBounds(50, 45, 120, 25);
tfFullTimeVacancyNumberDisplayPanel.setBounds(180, 45, 100, 25);

fullTimeBackButtonDisplayPanel.setBounds(50, 10, 70, 20);
```

```
fullTimeBackButtonDisplayPanel.setCursor(cursor);
fullTimeBackButtonDisplayPanel.addActionListener (new ActionListener() {
    public void actionPerformed (ActionEvent ae) {
        frameFullTimeStaffHire.dispose();
        fullTimeWindowCheck = false;
    }
}
);

fullTimeClearButtonDisplayPanel.setBounds(170, 150, 110, 25);
fullTimeClearButtonDisplayPanel.setCursor(cursor);
fullTimeClearButtonDisplayPanel.addActionListener (new ActionListener (){
    public void actionPerformed (ActionEvent ae) {
        clearTextFullTime();
    }
}
);

fullTimeDisplayButton.setBounds(50, 150, 110, 25);
fullTimeDisplayButton.setCursor(cursor);
fullTimeDisplayButton.addActionListener (new ActionListener (){
    public void actionPerformed (ActionEvent ae) {

        int vacancyNo = getFullTimeTfVacancyNumber();
        boolean found = false;

        if (vacancyNo != 0) {
            for (StaffHire s : arrayListStaffHire) {
```

```
                if(s instanceof FullTimeStaffHire) {
                    FullTimeStaffHire f = (FullTimeStaffHire) s;


                    if (s.getVacancyNumber() == vacancyNo) {
                        f.displayDetails();
                        found = true;
                        JOptionPane.showMessageDialog(frameFullTimeStaffHire, "Name: " + f.getStaffName() + "\nDesignation: " + s.getDesignation() + "\nJob Type: " + s.getJobType() + "\nQualification: " + f.getQualification() + "\nWorking Hour: " + f.getWorkingHour() + "\nSalary: " + f.getSalary() + "\nJoining Date: " + f.getJoiningDate() + "\nAppointed By: " + f.getAppointedBy(), "Staff Info.",JOptionPane.INFORMATION_MESSAGE);
                        clearTextFullTime();
                        break;
                    }
                }
            }
            if(!found) {
                JOptionPane.showMessageDialog(frameFullTimeStaffHire, "Vacancy number not found.", "Error!", JOptionPane.ERROR_MESSAGE);
                }
            }


        }
    }
    );


    displayPanel.add(lblFullTimeVacancyNumberDisplayPanel);
    displayPanel.add(tfFullTimeVacancyNumberDisplayPanel);
    displayPanel.add(fullTimeBackButtonDisplayPanel);
```

```
    displayPanel.add(fullTimeClearButtonDisplayPanel);
    displayPanel.add(fullTimeDisplayButton);

    tpFullTimeForm = new JTabbedPane();
    tpFullTimeForm.add("Add vacancy", vacancyPanel);
    tpFullTimeForm.add("Appoint", appointPanel);
    tpFullTimeForm.add("Display details", displayPanel);

    frameFullTimeStaffHire.add(tpFullTimeForm, BorderLayout.CENTER);
}

@Override
public void actionPerformed(ActionEvent e) {

}

public void clearTextPartTime () {
    if (tpPartTimeForm.getSelectedIndex() == 0) {
        tfPartTimeDesignation.setText(null);
        tfPartTimeJobType.setText(null);
        tfPartTimeWorkingHour.setText(null);
        tfPartTimeWagesPerHour.setText(null);
        tfPartTimeShifts.setText(null);
        tfPartTimeVacancyNumberVacancyPanel.setText(null);
    }
    else if (tpPartTimeForm.getSelectedIndex() == 1) {
        tfPartTimeName.setText(null);
        tfPartTimeQualification.setText(null);
```

```
        tfPartTimeJoiningDate.setText(null);
        tfPartTimeAppointedBy.setText(null);
        tfPartTimeVacancyNumberAppointPanel.setText(null);
    }
    else if (tpPartTimeForm.getSelectedIndex() == 2) {
        tfPartTimeVacancyNumberDisplayPanel.setText(null);
    }

}

public String getPartTimeTfName () {
    String tfName = tfPartTimeName.getText().trim();

    return tfName;
}

public String getPartTimeTfDesignation () {
    String tfDesignation = tfPartTimeDesignation.getText().trim();

    return tfDesignation;
}

public String getPartTimeTfJobType () {
    String tfJobType = tfPartTimeJobType.getText().trim();

    return tfJobType;
}
```

```
    public String getPartTimeTfQualification () {
        String tfQualification = tfPartTimeQualification.getText().trim();

        return tfQualification;
    }

    public String getPartTimeTfShifts () {
        String tfShifts = tfPartTimeShifts.getText().trim();

        return tfShifts;
    }

    public String getPartTimeTfJoiningDate () {
        String tfJoiningDate = tfPartTimeJoiningDate.getText().trim();

        return tfJoiningDate;
    }

    public String getPartTimeTfAppointedBy () {
        String tfAppointedBy = tfPartTimeAppointedBy.getText().trim();

        return tfAppointedBy;
    }

    public int getPartTimeTfWorkingHour () {
        int tfWorkingHour = 0;
        if((tfPartTimeWorkingHour.getText().trim()).equals("")){
            JOptionPane.showMessageDialog(framePartTimeStaffHire, "Working hour text
field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);
```

```
        }
        else{
            try {
                tfWorkingHour = Integer.parseInt((tfPartTimeWorkingHour.getText().trim()));
            }

            catch (NumberFormatException nfe) {
                JOptionPane.showMessageDialog(framePartTimeStaffHire, "Working hour input must be a number.", "Error", JOptionPane.ERROR_MESSAGE);
            }
        }
        return tfWorkingHour;
    }

    public int getPartTimeTfWagesPerHour () {
        int tfWagesPerHour = 0;
        if((tfPartTimeWagesPerHour.getText().trim()).equals("")) {
            JOptionPane.showMessageDialog(framePartTimeStaffHire, "Wages per hour text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);
        }
        else {
            try {
                tfWagesPerHour = Integer.parseInt((tfPartTimeWagesPerHour.getText().trim()));
            }

            catch (NumberFormatException nfe) {
                JOptionPane.showMessageDialog(framePartTimeStaffHire, "Wages per hour input must be a number.", "Error", JOptionPane.ERROR_MESSAGE);
            }
```

```
        }
        return tfWagesPerHour;
    }

    public int getPartTimeTfVacancyNumber () {
        int tfVacancyNumber = 0;
        if ((tfPartTimeVacancyNumberVacancyPanel.getText().trim()).equals("") && tpPartTimeForm.getSelectedIndex() == 0){
            JOptionPane.showMessageDialog(framePartTimeStaffHire, "Vacancy number text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);
        }
        else if ((tfPartTimeVacancyNumberAppointPanel.getText().trim()).equals("") && tpPartTimeForm.getSelectedIndex() == 1) {
            JOptionPane.showMessageDialog(framePartTimeStaffHire, "Vacancy number text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);
        }
        else if ((tfPartTimeVacancyNumberDisplayPanel.getText().trim()).equals("") && tpPartTimeForm.getSelectedIndex() == 2) {
            JOptionPane.showMessageDialog(framePartTimeStaffHire, "Vacancy number text field cannot be empty.", "Error", JOptionPane.ERROR_MESSAGE);
        }
        else {

            try {
                if (tpPartTimeForm.getSelectedIndex() == 0) {
                    tfVacancyNumber = Integer.parseInt((tfPartTimeVacancyNumberVacancyPanel.getText().trim()));
                }
                else if (tpPartTimeForm.getSelectedIndex() == 1) {
                    tfVacancyNumber = Integer.parseInt((tfPartTimeVacancyNumberAppointPanel.getText().trim()));
```

```
            }
            else {
                tfVacancyNumber = Integer.parseInt((tfPartTimeVacancyNumberDisplayPanel.getText().trim()));
            }
        }

        catch (NumberFormatException nfe) {
            JOptionPane.showMessageDialog(framePartTimeStaffHire, "Vacancy number input must be a number.", "Error" + nfe.getMessage(), JOptionPane.ERROR_MESSAGE);
        }
    }

    return tfVacancyNumber;
}

public String getFullTimeTfName() {
    String tfName = tfFullTimeName.getText().trim();

    return tfName;
}

public String getFullTimeTfDesignation() {
    String tfDesignation = tfFullTimeDesignation.getText().trim();

    return tfDesignation;
}
```

```
    public String getFullTimeTfJobType() {
        String tfJobType = tfFullTimeJobType.getText().trim();

        return tfJobType;
    }

    public String getFullTimeTfQualification() {
        String tfQualification = tfFullTimeQualification.getText().trim();

        return tfQualification;
    }

    public String getFullTimeTfJoiningDate() {
        String tfJoiningDate = tfFullTimeJoiningDate.getText().trim();

        return tfJoiningDate;
    }

    public String getFullTimeTfAppointedBy() {
        String tfAppointedBy = tfFullTimeAppointedBy.getText().trim();

        return tfAppointedBy;
    }

    public int getFullTimeTfWorkingHour() {
        int tfWorkingHour = 0;
        if((tfFullTimeWorkingHour.getText().trim().equals(""))) {
            JOptionPane.showMessageDialog(frameFullTimeStaffHire, "Working hour text
field cannot be empty.", "Error! ", JOptionPane.ERROR_MESSAGE);
```

```
        }
        else {
            try {
                tfWorkingHour = Integer.parseInt((tfFullTimeWorkingHour.getText().trim()));
            }

            catch (NumberFormatException nfe) {
                JOptionPane.showMessageDialog(frameFullTimeStaffHire, "Working hour input must be a number.", "Error! " + nfe.getMessage(), JOptionPane.ERROR_MESSAGE);
            }
        }
        return tfWorkingHour;
    }

    public int getFullTimeTfSalary() {
        int tfSalary = 0;
        if ((tfFullTimeSalary.getText().trim()).equals("")) {
            JOptionPane.showMessageDialog(frameFullTimeStaffHire, "Salary text field cannot be empty.", "Error! ", JOptionPane.ERROR_MESSAGE);
        }
        else{
            try {
                tfSalary = Integer.parseInt((tfFullTimeSalary.getText().trim()));
            }

            catch (NumberFormatException nfe) {
                JOptionPane.showMessageDialog(frameFullTimeStaffHire, "Salary input must be a number.", "Error! " + nfe.getMessage(), JOptionPane.ERROR_MESSAGE);
            }
```

```
        }
        return tfSalary;
    }

    public int getFullTimeTfVacancyNumber() {
        int tfVacancyNumber = 0;
        if ((tfFullTimeVacancyNumberVacancyPanel.getText().trim()).equals("") && tpFullTimeForm.getSelectedIndex() == 0) {
            JOptionPane.showMessageDialog(frameFullTimeStaffHire, "Vacancy number text field cannot be empty.", "Error! ", JOptionPane.ERROR_MESSAGE);
        }
        else if ((tfFullTimeVacancyNumberAppointPanel.getText().trim()).equals("") && tpFullTimeForm.getSelectedIndex() == 1) {
            JOptionPane.showMessageDialog(frameFullTimeStaffHire, "Vacancy number text field cannot be empty.", "Error! ", JOptionPane.ERROR_MESSAGE);
        }
        else if ((tfFullTimeVacancyNumberDisplayPanel.getText().trim()).equals("") && tpFullTimeForm.getSelectedIndex() == 2) {
            JOptionPane.showMessageDialog(frameFullTimeStaffHire, "Vacancy number text field cannot be empty.", "Error! ", JOptionPane.ERROR_MESSAGE);
        }
        else {
            try {
                if (tpFullTimeForm.getSelectedIndex() == 0) {
                    tfVacancyNumber = Integer.parseInt((tfFullTimeVacancyNumberVacancyPanel.getText().trim()));
                }
                else if (tpFullTimeForm.getSelectedIndex() == 1) {
                    tfVacancyNumber = Integer.parseInt((tfFullTimeVacancyNumberAppointPanel.getText().trim()));
                }
```

```
            else {
                tfVacancyNumber = Integer.parseInt((tfFullTimeVacancyNumberDisplayPanel.getText().trim()));
            }
        }

        catch (NumberFormatException nfe) {
            JOptionPane.showMessageDialog(frameFullTimeStaffHire, "Vacancy number input must be a number.", "Error! " + nfe.getMessage(), JOptionPane.ERROR_MESSAGE);
        }
    }
    return tfVacancyNumber;
  }

  public void clearTextFullTime() {
    if (tpFullTimeForm.getSelectedIndex() == 0) {
        tfFullTimeDesignation.setText(null);
        tfFullTimeJobType.setText(null);
        tfFullTimeWorkingHour.setText(null);
        tfFullTimeSalary.setText(null);
        tfFullTimeVacancyNumberVacancyPanel.setText(null);
    }
    else if (tpFullTimeForm.getSelectedIndex() == 1) {
        tfFullTimeName.setText(null);
        tfFullTimeQualification.setText(null);
        tfFullTimeJoiningDate.setText(null);
        tfFullTimeAppointedBy.setText(null);
        tfFullTimeVacancyNumberAppointPanel.setText(null);
```

```
    }
    else if (tpFullTimeForm.getSelectedIndex() == 2) {
        tfFullTimeVacancyNumberDisplayPanel.setText(null);
    }
}

public void staffTableRowRemover(int x) {
    int rowCount = staffTable.getRowCount();
    int rowToDelete = 0;

    for (int i = 0; i < rowCount; i++) {
        String rowEntry = staffTable.getValueAt(i, 0).toString().trim();
        int rowEntryNum = Integer.parseInt(rowEntry);

        if (x == rowEntryNum) {
            rowToDelete = i;
            break;
        }
    }
    DefaultTableModel modelStaffTable = (DefaultTableModel) staffTable.getModel();
    modelStaffTable.removeRow(rowToDelete);
}

public void vacancyTableRowRemover(int x) {
    int rowCount = vacancyTable.getRowCount();
    int rowToDelete = 0;

    for (int i = 0; i < rowCount; i++) {
```

```
            String rowEntry = vacancyTable.getValueAt(i, 0).toString().trim();
            int rowEntryNum = Integer.parseInt(rowEntry);

            if (x == rowEntryNum) {
                rowToDelete = i;
                break;
            }
        }
        DefaultTableModel modelVacancyTable = (DefaultTableModel) vacancyTable.getModel();
        modelVacancyTable.removeRow(rowToDelete);
    }

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

}
```