Module Code & Module Title

CS4001NI Programming

Assessment Weightage & Type

30% Individual Coursework

Year and Semester

2018-19 Autumn

Student Name: Suraksha Shrestha

London Met ID: 18029473

College ID: NP01NT4A180002

Assignment Due Date: 19 April 19, 2019

Assignment Submission Date: 19 April 2019

Table of Contents

Table of Figures

Table of Tables

Introduction:

This is the second project of given module where we have to create a Graphical User Interface (GUI) for a Developer Appointment System. This project is linked with the first project i.e. the child classes we made (Junior and Senior Developer) are being added, appointed and terminated. This project was a bit difficult but with the help of our module leader and the lectures we attended, a frictionless project is created successfully.

This assignment consists of a class named RigoTechnology which is linked with different classes of our previous assignment i.e. Developer, JuniorDeveloper and SeniorDeveloper. While developing GUI JLabel, JButton, JRadioButton, JPanel, Jseparator and JTextField were used which were imported from the swing packet of javax. An arraylist, list is created to add the objects of the developers. So, when user wants to hire a developer, same object is extracted form the arraylist and performs object casting. For different buttons in the GUI, different methods are constructed. So, the required process or code is written in these methods, for the functionality of the buttons. The values entered by the users in the given fields are extracted using a method, getText(). These values are then used for displaying and storing. Try and catch blocks are implemented in order to avoid errors in the program. While dealing with these errors, there are commands to pop out the messages of errors, by using message dialog box.

PSEUDOCODE:

- **Pseudocode of RigoTechnology:**

```
DO
INITIALIZE frame


INITIALIZE p1,p2,p3
SETBOUNDS p1,p2,p3
ADD p1,p2,p3 TO frame


INITIALIZE Platform,platform
SETBOUNDS of the Platform,Platform
ADD Platform,platform TO frame
```

```
INITIALIZE InterviewerName,interviewerName
SETBOUNDS of the InterviewerName,interviewerName
ADD InterviewerName,interviewerName TO p1

INITIALIZE WorkingHours,workingHours
SETBOUNDS of the WorkingHours,workingHours
ADD WorkingHours,workingHours TO p1

INITIALIZE Salary,salary
SETBOUNDS of the Salary,salary
ADD Salary,salary TO p1

INITIALIZE AppointedBy,appointedBy
SETBOUNDS of the AppointedBy,appointedBy
ADD AppointedBy,appointedBy TO p1

INITIALIZE TerminationDate,terminationDate
SETBOUNDS of the TerminationDate,terminationDate
ADD TerminationDate,terminationDate TO p1

INITIALIZE ContractPeriod,contractPeriod
SETBOUNDS of the ContractPeriod,contractPeriod
ADD ContractPeriod,contractPeriod TO p1

INITIALIZE Name,name
SETBOUNDS of the Name,name
```

```
ADD Name,name TO p2

INITIALIZE DevNo,devNo
SETBOUNDS of the DevNo,devNo
ADD DevNo,devNo TO p2

INITIALIZE WorkingHours,workingHours
SETBOUNDS of the WorkingHours,workingHours
ADD WorkingHours,workingHours TO p2
appointedBy,appointedByTx

INITIALIZE JoiningDate,joiningDate
SETBOUNDS of the JoiningDate,joiningDate
ADD JoiningDate,joiningDate TO p2

INITIALIZE RoomNo,roomNo
SETBOUNDS of the RoomNo,roomNo
ADD RoomNo,roomNo TO p2

INITIALIZE AdvSalary,advSalary
SETBOUNDS of the AdvSalary,advSalary
ADD AdvSalary,advSalary TO p2

INITIALIZE NameJ,nameJ
SETBOUNDS of the NameJ,nameJ
ADD NameJ,nameJ TO p3
```

```
INITIALIZE Specialization,specialization
SETBOUNDS of the Specialization,specialization
ADD Specialization,specialization TO p3

INITIALIZE DevNoJ,devNoJ
SETBOUNDS of the DevNoJ,devNoJ
ADD DevNoJ,devNoJ TO p3

INITIALIZE AppointDate,appointDate
SETBOUNDS of the AppointDate,appointDate
ADD AppointDate,appointDate TO p3

INITIALIZE TerminateDate,terminateDate
SETBOUNDS of the TerminateDate,terminateDate
ADD TerminateDate,terminateDate TO p3

INITIALIZE AppointedBy,appointedBy
SETBOUNDS of the AppointedBy,appointedBy
ADD AppointedBy,appointedBy TO p3

INITIALIZE EvaluationPeriod,evaluationPeriod
SETBOUNDS of the EvaluationPeriod,evaluationPeriod
ADD EvaluationPeriod,evaluationPeriod TO p3

INITIALIZE AddJunior
SETBOUNDS of AddJunior
ADD AddJunior TO p1
```

```
INITIALIZE AddSenior
SETBOUNDS of AddSenior
ADD AddSenior TO p1

INITIALIZE Appoint
SETBOUNDS of Appoint
ADD Appoint TO p2

INITIALIZE Terminate
SETBOUNDS of Terminate
ADD AddJunior TO p2

INITIALIZE AppointJuniorDeveloper
SETBOUNDS of AppointJuniorDeveloper
ADD AppointJuniorDeveloperr TO p3

INITIALIZE Display
SETBOUNDS of Display
ADD Display TO frame

INITIALIZE Clear
SETBOUNDS of Clear
ADD Clear TO frame

INITIALIZE s1
SETBOUNDS of s1
```

```
ADD s1 TO p1

INITIALIZE Junior, Senior
SETBOUNDS of Junior, Senior
ADD Junior, Senior TO p1

SETBOUNDS FOR frame
SETDEFAULTCLOSEOPERATION  FOR frame to JFrame EXIT_ON_CLOSE
SET frame LAYOUT TO null
SET frame VISIBILTY to TRUE
END DO

BUILD A actionPerformed METHOD TAKING ActionEvent e AS A PARAMETER
  DO
    IF (e.getActionCommand is equal to ("ADD JUNIOR"))
      DO
        addJunior();
      END DO

    ELSE IF (e.getActionCommand is equal to("ADD SENIOR"))
      DO
        addSenior();
      END DO
    ELSE IF (e.getActionCommand is equal to ("SENIOR"))
      DO
        SET required fields to VISIBLE for senior developer
      END IF
```

```
ELSE IF (e.getActionCommand is equal to ("JUNIOR"))
    DO
        SET required fields to VISIBLE for junior developer
    END IF

ELSE IF (e.getActionCommand is equal to ("Appoint"))
    DO
         appoint();
    END IF
ELSE IF (e.getActionCommand is equal to ("Terminate"))
    DO
        terminate();
    END DO

ELSE IF (e.getActionCommand is equal to (""Appoint Junior Developer""))
    DO
        appointJuniorDeveloper();
    END DO

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

ELSE IF(e.getActionCommand is equal to ("Clear"))
    DO
        clear();
```

```
            END DO
   END DO

BUILD addSenior() METHOD
 DO
  START A TRY-CATCH
   STORE THE INPUT FROM platform,workingHours,salary,
   interviewerName,contractPeriod
   CHECKS IF  THE FIELDS ARE EMPTY
   IF THE fields ARE EMPTY THROWS ERROR

   ELSE

    CONVERT salary,contractPeriod TO INTEGER
    CREATE A NEW SeniorDeveloper OBJECT developer
    ADD developer
    SHOW A SUCCESS MESSAGE

  END IF
 CATCH exception:NumberFormatException E
  SHOW AN ERROR MESSAGE
 CATCH exception:NullPointerException E
  SHOW AN ERROR MESSAGE
 END TRY-CATCH
END Do

BUILD A addJunior() METHOD
```

```
 DO
  START A TRY-CATCH
   STORE THE INPUT FROM platform,workingHours,salary,
   interviewerName,terminationDate,appointedBy
   CHECKS IF  THE FIELDS ARE EMPTY
   IF THE fields ARE EMPTY THROW AND ERROR

   ELSE

    CONVERT salary TO INTEGER
    CREATE A NEW JuniorDeveloper OBJECT developer
    ADD developer
    SHOW A SUCCESS MESSAGE

  END IF
 CATCH exception:NumberFormatException E
  SHOW AN ERROR MESSAGE
 CATCH exception:NullPointerException E
  SHOW AN ERROR MESSAGE
 END TRY-CATCH
END Do

BUILD A appoint() METHOD
DO
  START A TRY-CATCH
   STORE THE INPUT FROM name,joiningDtae,devNo,roomNo,advSalary
   CHECKS IF  THE FIELDS ARE EMPTY
   IF THE fields ARE EMPTY THROW AN ERROR

   ELSE
    CONVERT AdvSalary,DevNo TO INTEGER
    IF DevNo1>=0 && DevNo1<list.size()
       IF list.get(DevNo1)instanceof SeniorDeveloper
          IF s1.getAppointed()==false
             CALL hireDeveloper(Name,JoiningDate,AdvSalary1,RoomNo)
             SHOW A SUITABLE MESSAGE
          ELSE
             SHOW A SUITABLE MESSAGE
       ELSE
        SHOW A SUITABLE MESSAGE
    ELSE
     SHOW A SUITABLE MESSAGE
    SHOW A SUCCESS MESSAGE

  END IF
 CATCH exception:NumberFormatException E
  SHOW AN ERROR MESSAGE
 CATCH exception:NullPointerException E
  SHOW AN ERROR MESSAGE
 END TRY-CATCH
END Do

BUILD A appoint() METHOD
DO
```

```
 START A TRY-CATCH
  STORE THE INPUT FROM devNo
  CHECKS IF  THE FIELD IS EMPTY
  IF THE field IS EMPTY THROW AN ERROR

  ELSE
   CONVERT DevNo TO INTEGER
   IF DevNo1>=0 && DevNo1<list.size()
      IF list.get(DevNo1)instanceof SeniorDeveloper
         IF s1.getTerminated()==false
            CALL contractTermination()
            SHOW A SUITABLE MESSAGE
         ELSE
            SHOW A SUITABLE MESSAGE
      ELSE
       SHOW A SUITABLE MESSAGE
   ELSE
    SHOW A SUITABLE MESSAGE
   SHOW A SUCCESS MESSAGE

 END IF
CATCH exception:NumberFormatException E
 SHOW AN ERROR MESSAGE
CATCH exception:NullPointerException E
 SHOW AN ERROR MESSAGE
END TRY-CATCH
END Do
```

```
BUILD A appointJuniorDeveloper() METHOD
DO
 START A TRY-CATCH
  STORE THE INPUT FROM nameJ,specialization,devNoJ,appointedBy,appointDtae,terminateDate,evaluationPeriod
  CHECKS IF  THE FIELD IS EMPTY
  IF THE field IS EMPTY THROW AN ERROR
  ELSE IF data type is invalid
    SHOW A SUITABLE MESSAGE
  ELSE
   CONVERT DevNoJ TO INTEGER
   IF DevNo1>=0 && DevNo1<list.size()
      IF list.get(DevNo1)instanceof JuniorDeveloper
         IF j1.getJoined()==false
            CALL appointDeveloper(Name,AppointDate,TerminateDate,Specialization
                   ,EvaluationPeriod,AppointedByJ);
            SHOW A SUITABLE MESSAGE
         ELSE
            SHOW A SUITABLE MESSAGE
      ELSE
       SHOW A SUITABLE MESSAGE
   ELSE
    SHOW A SUITABLE MESSAGE
   SHOW A SUCCESS MESSAGE

 END IF
CATCH exception:NumberFormatException E
 SHOW AN ERROR MESSAGE
CATCH exception:NullPointerException E
 SHOW AN ERROR MESSAGE
END TRY-CATCH

BUILD A display() METHOD
DO
IF list.size==0
  GIVE SUITABLE OUTPUT
ELSE
  FOR int i=0;i<list.size();i++
    IF list.get(i) instanceof SeniorDeveloper
       GET s1 FROM list
       GIVE SUITABLE MESSAGE
    ELSE
       GET j1 FROM list
       GIVE SUITABLE MESSAGE

END Do

BUILD A clear() METHOD
      SET ALL THE FIELD TO EMPTY
```

Class Diagram:

Here, Class Diagram is the tabular representation of all the instance variables and methods used in the given class.

Class Diagram for RigoTechnology class is given below:

RigoTechnology
-frame:JFrame -l1:JLabel -Platform:JLabel -InterviewerName: JLabel -WorkingHours: JLabel -Salary: JLabel -AppointedBy: JLabel -TerminationDate: JLabel -ContractPeriod: JLabel -Name: JLabel -DevNo: JLabel -JoiningDate: JLabel -RoomNo:J JLabel -AdvSalary: JLabel -NameJ: JLabel -Specialization: JLabel -DevNoJ: JLabel -AppointDate: JLabel -TerminateDate: JLabel -AppointedBy: JLabel -EvaluationPeriod: JLabel -platform:JTextField -interviewerName :JTextField -salary:JTextField -appointedBy:JTextField -terminationDate:JTextField -contractPeriod:JTextField -name:JTextField -devNo:JTextField -joiningDate:JTextField -roomNo:JTextField -advSalary:JTextField -nameJ:JTextField -specialization:JTextField -devnoJ:JTextField -appointDate:JTextField

-TerminateDate:JTextField -appointedBy:JTextField -evaluationPeriod:JTextField -workingHours:JComboBox -AddJunior:JButton - AddSenior:JButton - Appoint:JButton - Terminate:JButton - AppointJuniorDeveloper:JButton - Display:JButton - Clear:JButton -p1:Jpanel -p2:JPanel -p3:JPanel -s1:JSeparator -Junior:JRadioButton -Senior :JRadioButton -group:ButtonGroup
+actionPerformed(ActionEvent e):void +addJunior():void +addSenior():void +appoint():void +terminate():void +appointJuniorDeveloper():void +display():void +clear():void

Method Description:

❖ Method Description for RigoTechnology Class:

Method Name: actionPerformed()

This method is an abstract class of the ActionListener class. Here, the class RigoTechnology implements ActionListener class and overrides the method, actionPerformed(). It is used for an event such as a method runs when a button in the GUi is clicked.

Method Name: Junior()

This method run when the user clicks on junior radio button. This method hides all those fields which is not necessary while adding the junior developer, in GUI.

Method Name: Senior()

This method run when the user clicks on senior radio button. This method hides all those fields which is not necessary while adding the senior developer, in GUI.

Method Name: addJunior()

Whenever the user clicks on the add junior button, this method gets run. This method firstly extract all the entered values of the textfield and checks whether the fields are left blank. If yes, suitable message is thrown out. It also changes the datatype of those fields. After that, those values are passed as parameters in the constructor of the junior class, creating the object of junior developer and adding it to an arraylist.

Method Name: addSenior()

Whenever the user clicks on the add senior button, this method gets run. This method firstly extract all the entered values of the textfield and checks whether the fields are left blank. If yes, suitable message is thrown out. It also changes the datatype of those fields. After that, those values are passed as parameters in the constructor of the senior class, creating the object of senior developer and adding it to an arraylist.

Method Name: appoint()

This method gets run when user clicks on appoint button. It extracts the values of the text field such as Name, JoiningDate, DevNo, RoomNo and AdvSalary and changes them to respective datatypes. After checking the validity of the developer number, it accesses the appropriate developer from the arraylist or sends a suitable message that the developer number is invalid. Here, method of appointing is called from senior developer class and the developer is hired.

Method Name: terminate()

This method gets run when the user clicks on the terminate button. It only extracts the value of developer number and checks its validity. If the developer number is valid, it accesses the appropriate developer and terminates it else, gives an suitable error message to the user.

Method Name: appointJuniorDeveloper()

Whenever the user clicks on appoint junior developer button, this method is called. In this method, the values of textfields are accessed and changed into respective datatype. It checks the validity of the developer number. If it is valid, hireDeveloper() method is called and developer is appointed else a suitable error message is given out.

Method Name: display()

This method is called when the user clicks on the button display. This method is called to display all the information of certain developer.

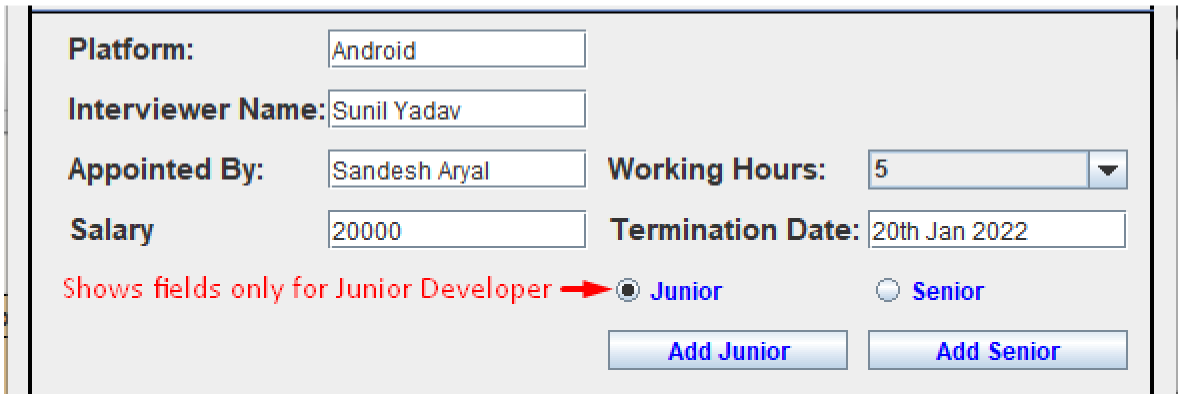
Method Name: clear ()

This method is called when the user clicks on the clear button in the GUI. This method clears all the textfield so that new values can be entered by the user.

\

Testing:

❖ **Test No. 1: Add platform to Junior Developer List:**



Platform:	Android		
Interviewer Name:	Sunil Yadav		
Appointed By:	Sandesh Aryal	Working Hours:	5
Salary	20000	Termination Date:	20th Jan 2022

Shows fields only for Junior Developer → [✓ Junior] Senior

Figure 1: Filling the fields of junior developer

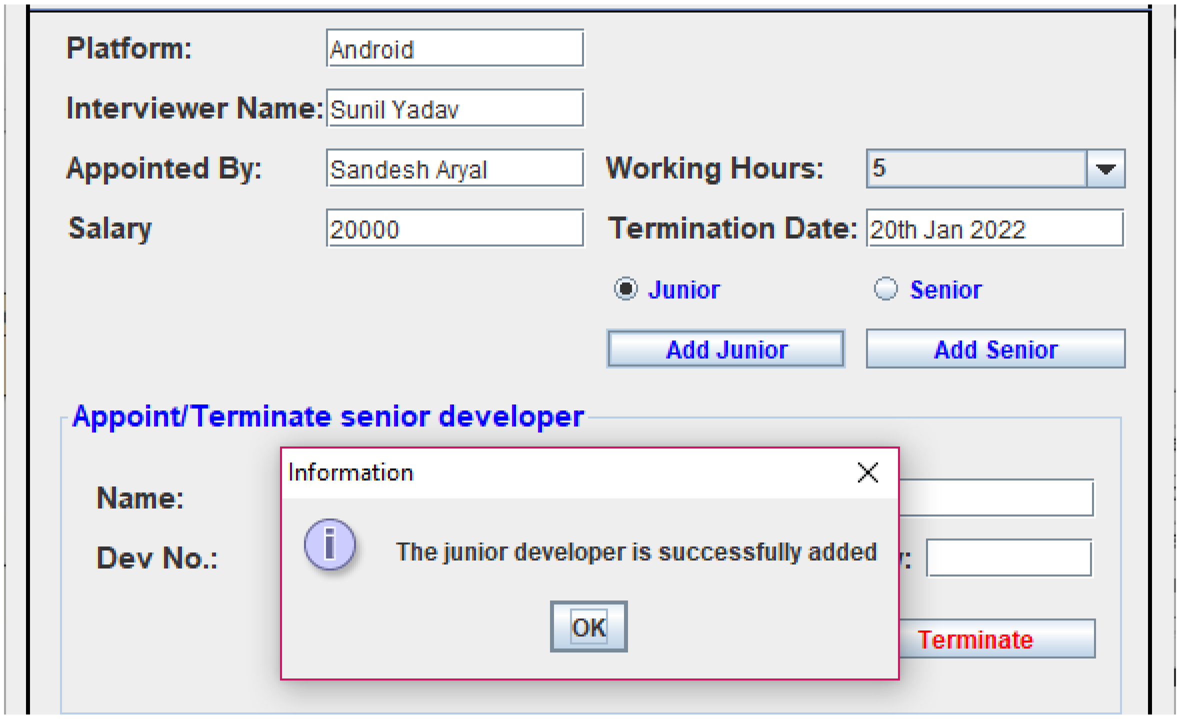

Platform: Android
Interviewer Name: Sunil Yadav
Appointed By: Sandesh Aryal
Working Hours: 5
Salary 20000
Termination Date: 20th Jan 2022
Junior [✓] Senior
Information
The junior developer is successfully added

Figure 2:Junior developer successfully added

Test	1
Task Performed	Fill up the fields in GUI and click on "Add Junior" button.
Expected Result	A message will be thrown out with suitable message.
Actual Result	A message is prompt out saying that the developer has been successfully added.
Test Result	Test Successful

Table 1: Test No.1

❖ **Test No. 2: Add Platform to Senior Developer List**

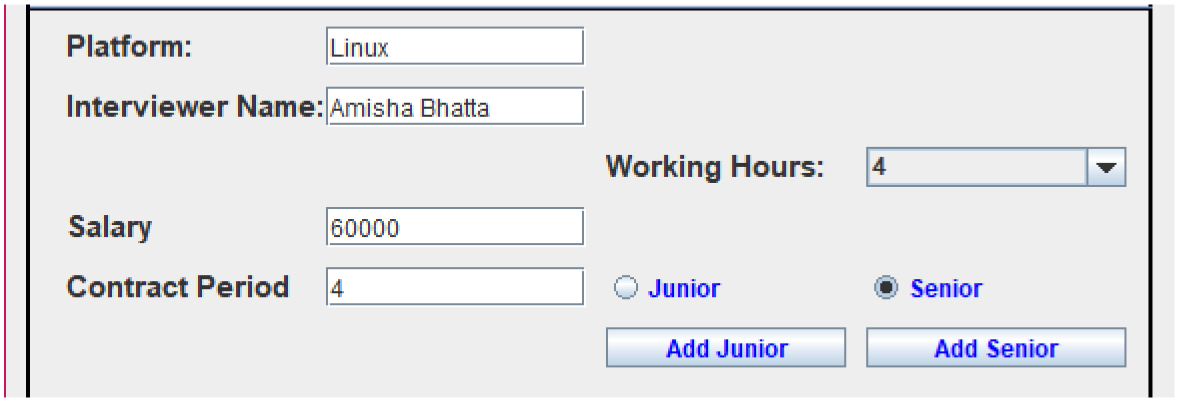
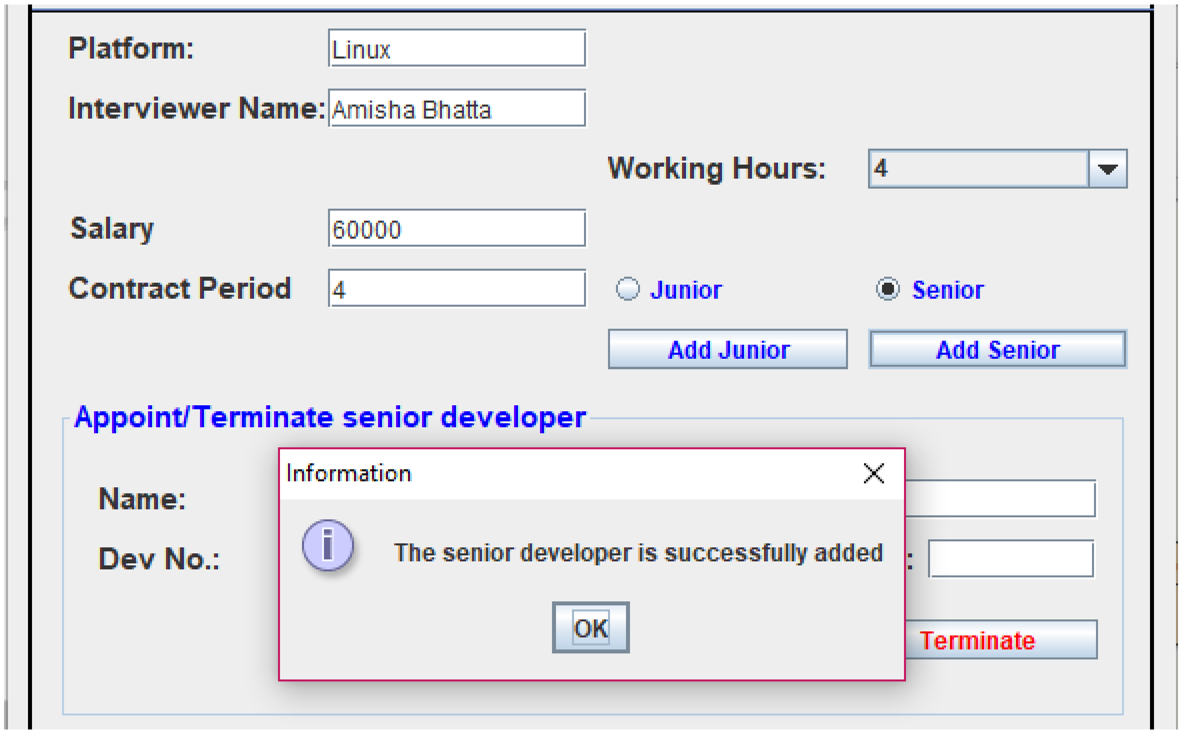

Platform: Linux
Interviewer Name: Amisha Bhatta
Working Hours: 4
Salary 60000
Contract Period 4

Figure 3:Filling fields for senior developer


Platform: Linux
Interviewer Name: Amisha Bhatta
Working Hours: 4
Salary 60000
Contract Period 4

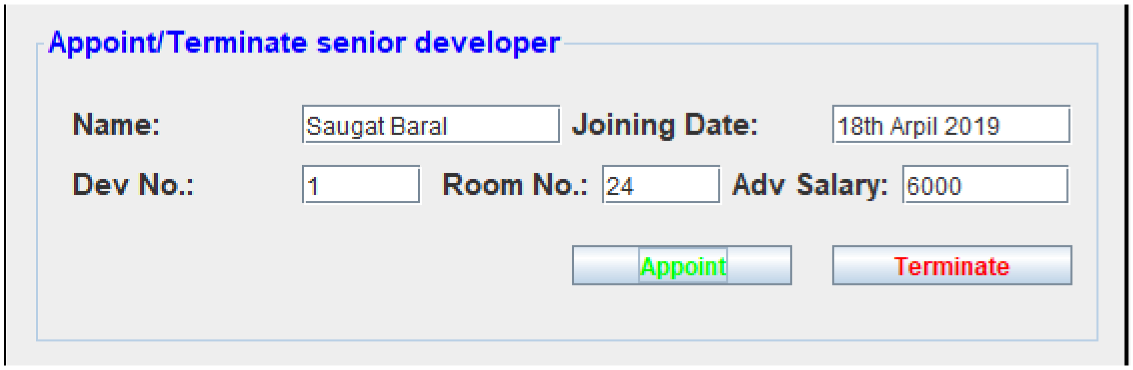
The senior developer is successfully added

Figure 4: Senior Developer successfully added

Test	2
Task Performed	Fill up the fields in GUI and click on “Add Senior” button.
Expected Result	A message will be thrown out with suitable message.
Actual Result	A message is prompt out saying that the developer has been successfully added.
Test Result	Test Successful

Table 2: Test No.2

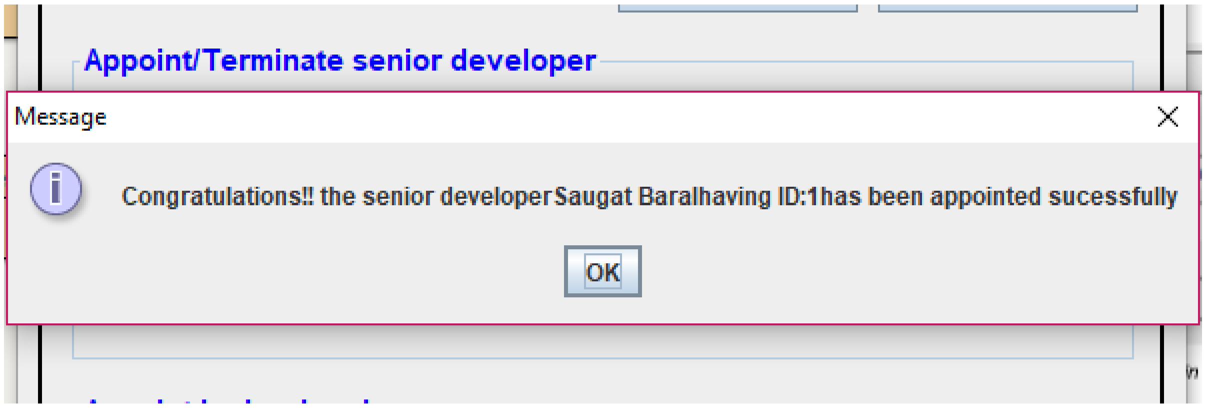
❖ Test No. 3: Appointing Senior Developer:



Appoint/Terminate senior developer

Name: Saugat Baral
Joining Date: 18th Arpil 2019
Dev No.: 1
Room No.: 24
Adv Salary: 6000

Figure 5: Appointing Senior Developer



Appoint/Terminate senior developer

Message

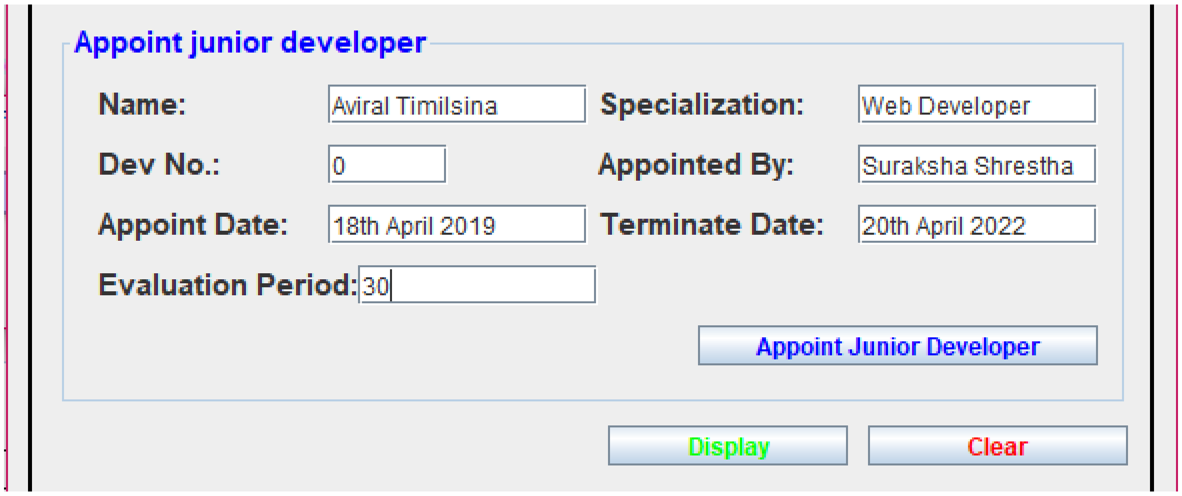
Congratulations!! the senior developerSaugat Baralhaving ID:1has been appointed sucessfully

Figure 6: Senior Developer Successfully Appointed

Test	3
Task Performed	Fill up the fields in GUI and click on “Add Senior” button.
Expected Result	A message will be thrown out with suitable message.
Actual Result	A message is prompt out saying that the developer has been successfully appointed.
Test Result	Test Successful

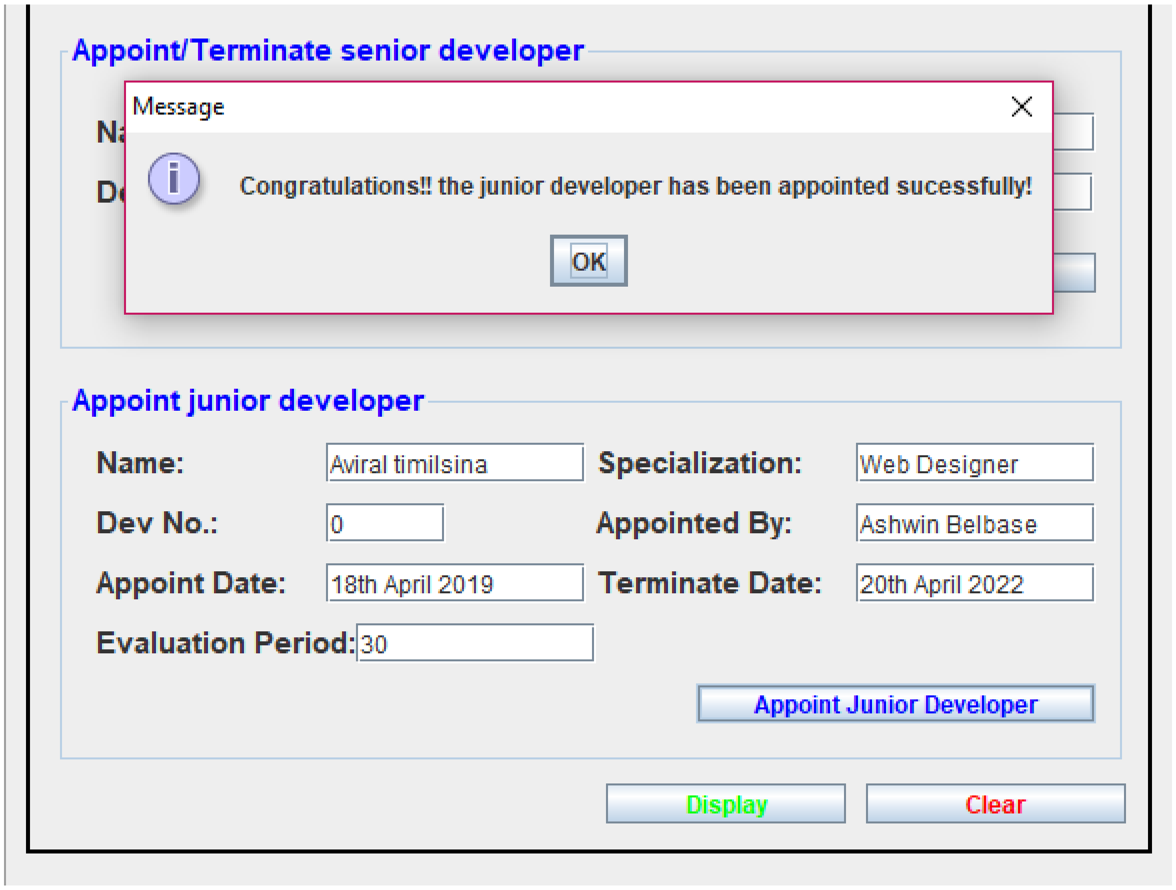
Table 3: Test No.3

❖ Test No. 4: Appointing Junior Developer:



Name: Aviral Timilsina
Specialization: Web Developer
Dev No.: 0
Appointed By: Suraksha Shrestha
Appoint Date: 18th April 2019
Terminate Date: 20th April 2022
Evaluation Period: 30

Figure 7: Appointing Junior Developer



Congratulations!! the junior developer has been appointed sucessfully!

Name: Aviral timilsina
Specialization: Web Designer
Dev No.: 0
Appointed By: Ashwin Belbase
Appoint Date: 18th April 2019
Terminate Date: 20th April 2022
Evaluation Period: 30

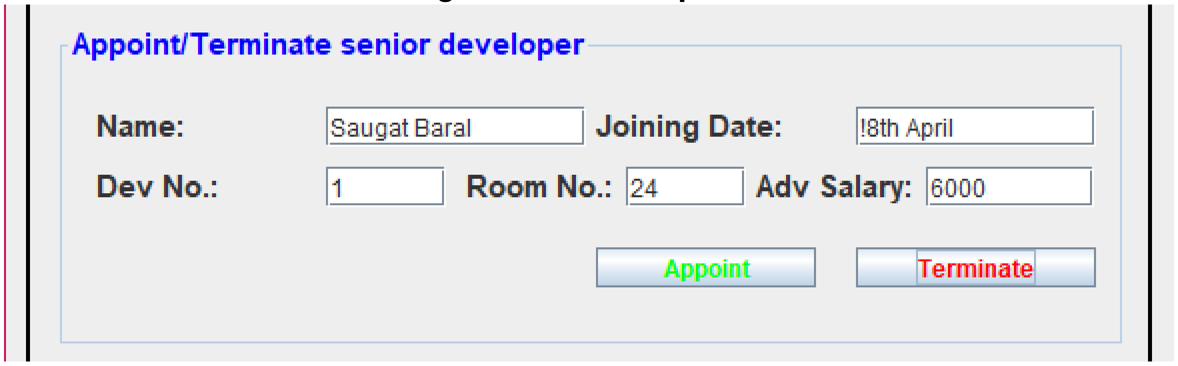
Figure 8:Junior Developer Successfully Appointed

Test	4
Task Performed	Fill up the fields in GUI and click on “Add Junior Developer” button.

Expected Result	A message will be thrown out with suitable message.
Actual Result	A message is prompt out saying that the developer has been successfully appointed.
Test Result	Test Successful

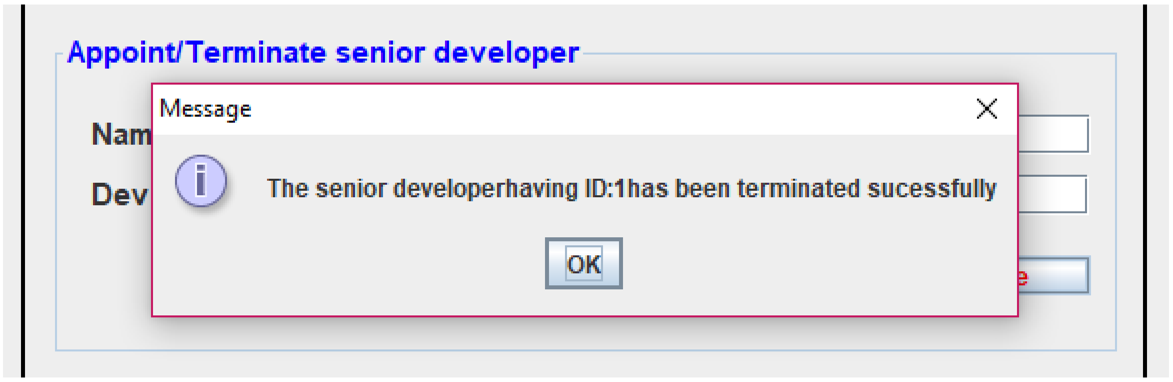
Table 4: Test No.4

❖ **Test No. 5: Terminating Senior Developer:**



Name: Saugat Baral
Joining Date: !8th April
Dev No.: 1
Room No.: 24
Adv Salary: 6000

Figure 9: Terminating Senior Developer



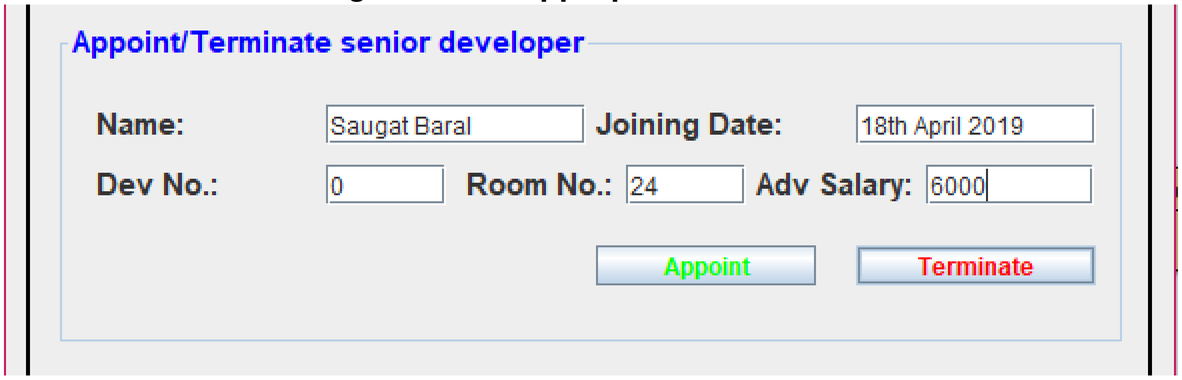
The senior developerhaving ID:1has been terminated sucessfully

Figure 10: Terminating Senior Developer Successfully

Test	5
Task Performed	Fill up the fields in GUI and click on “Terminate” button.
Expected Result	A message will be thrown out with suitable message.
Actual Result	A message is prompt out saying that the developer has been successfully terminated.
Test Result	Test Successful

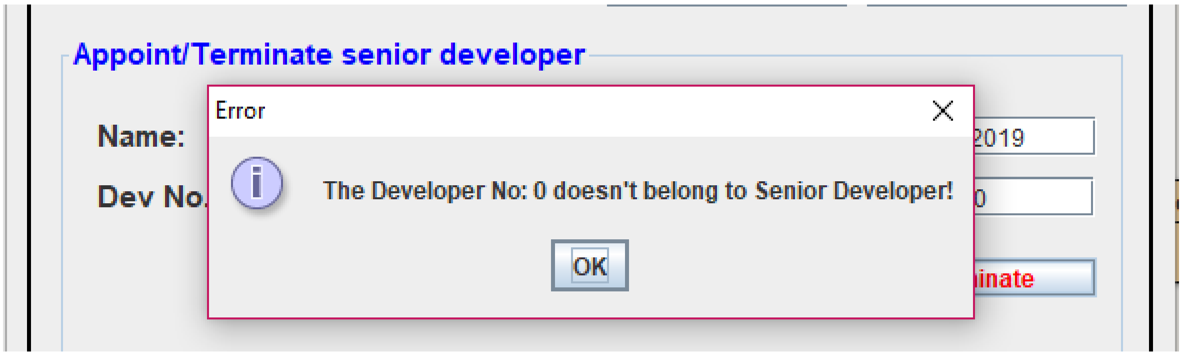
Table 5: Test No.5

- **Test no. 6: Dialog box for inappropriate value entered in DevNo:**


Appoint/Terminate senior developer

Name:	Saugat Baral	Joining Date:	18th April 2019
Dev No.:	0	Room No.: 24	Adv Salary: 6000

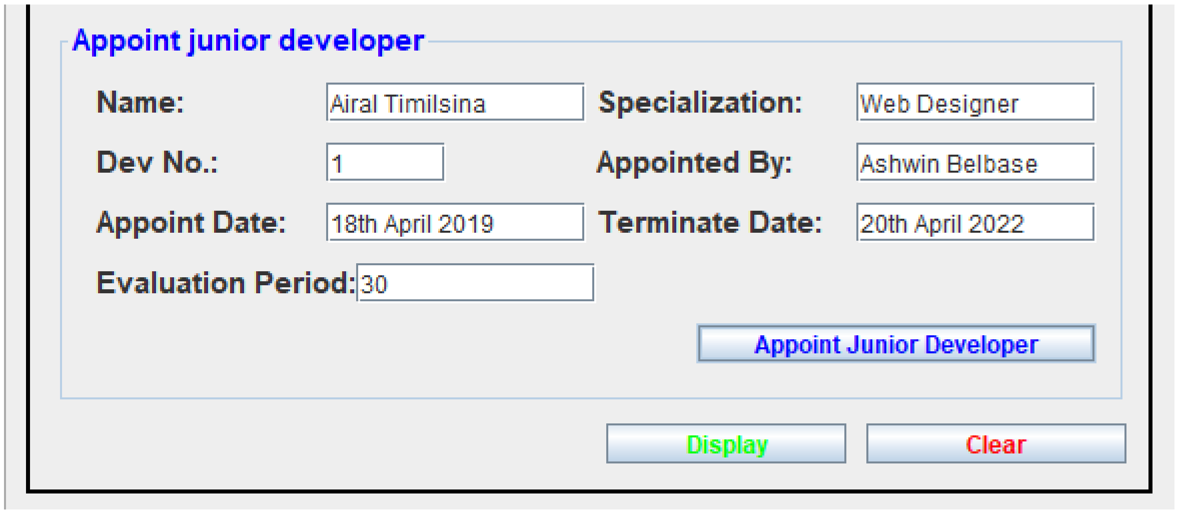
Figure 11: Entering invalid devNo in Senior Developer


Appoint/Terminate senior developer

Error

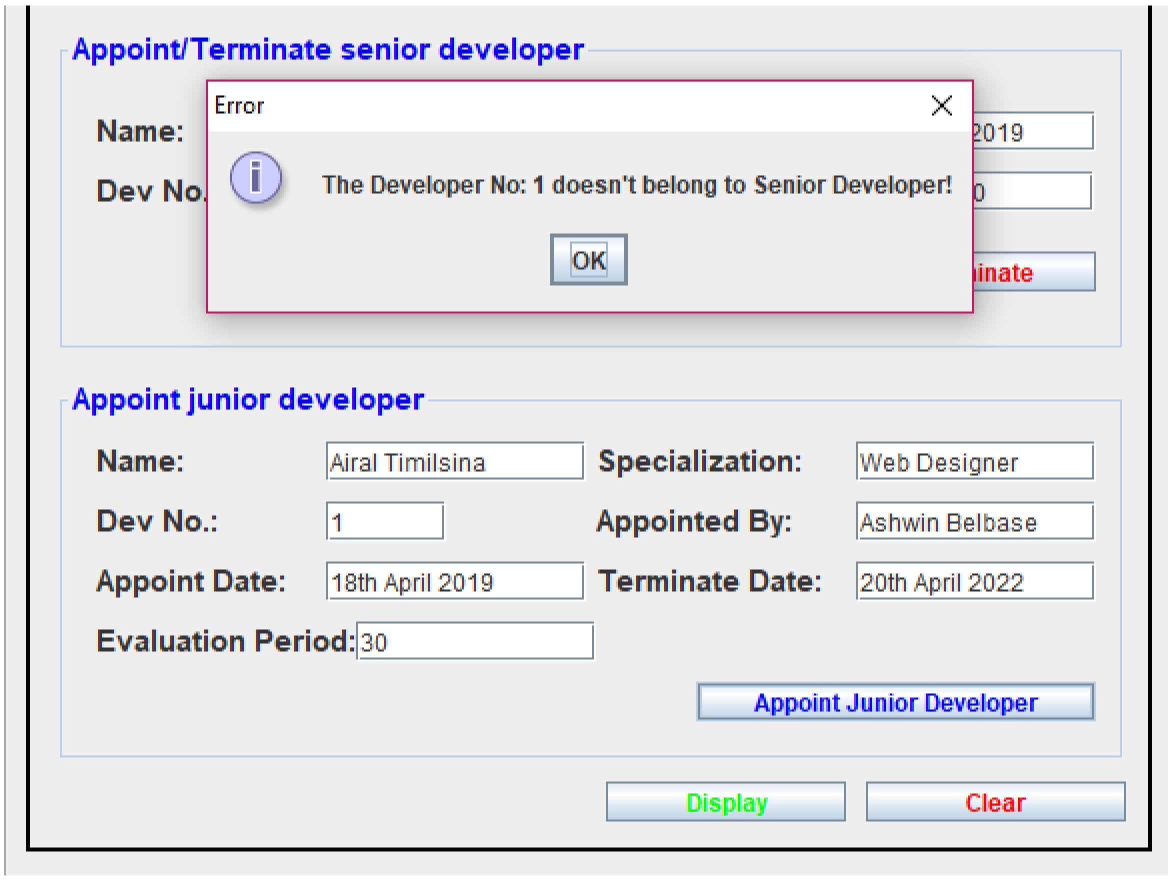
The Developer No: 0 doesn't belong to Senior Developer!

Figure 12: Error Message for Senior Developer


Appoint junior developer

Name:	Airal Timilsina	Specialization:	Web Designer
Dev No.:	1	Appointed By:	Ashwin Belbase
Appoint Date:	18th April 2019	Terminate Date:	20th April 2022
Evaluation Period:	30		

Figure 13: Entering invalid devNo for Junior Developer



Error

The Developer No: 1 doesn't belong to Senior Developer!

OK

Name:	Airal Timilsina	Specialization:	Web Designer
Dev No.:	1	Appointed By:	Ashwin Belbase
Appoint Date:	18th April 2019	Terminate Date:	20th April 2022
Evaluation Period:	30		

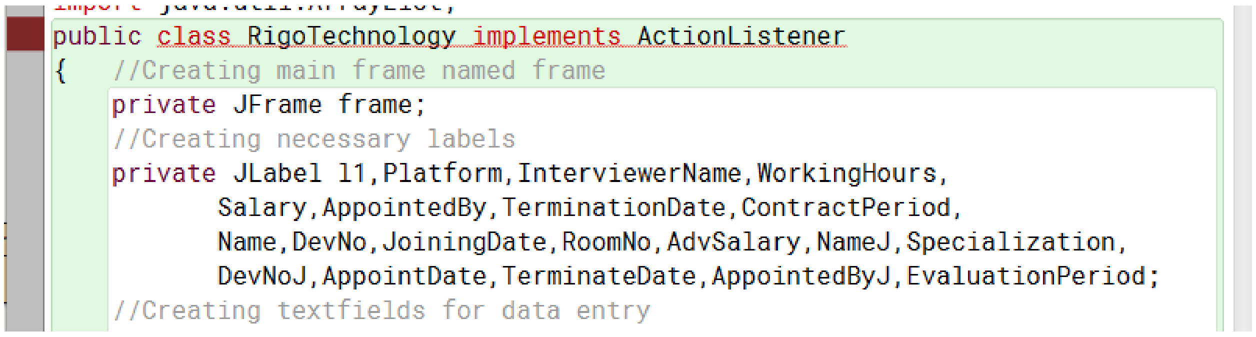
Figure 14: Error Message for Junior Developer

Test	6
Task Performed	Fill up the field devNo with invalid developer number and click on “Appoint” or “Appoint Junior Developer” button.
Expected Result	An error message will be thrown out with suitable message.
Actual Result	An error message is prompt out saying that the developer does not exist in the arraylist.
Test Result	Test Successful

Table 6: Test No.6

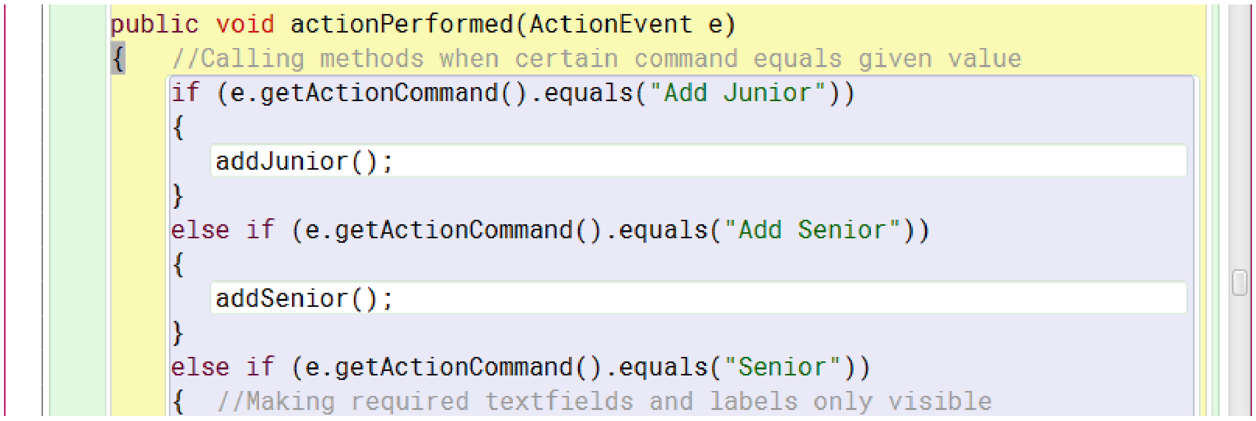
Error Detection and its Correction:

❖ **Error 1:**



```
public class RigoTechnology implements ActionListener
{   //Creating main frame named frame
    private JFrame frame;
    //Creating necessary labels
    private JLabel l1,Platform,InterviewerName,WorkingHours,
           Salary,AppointedBy,TerminationDate,ContractPeriod,
           Name,DevNo,JoiningDate,RoomNo,AdvSalary,NameJ,Specialization,
           DevNoJ,AppointDate,TerminateDate,AppointedByJ,EvaluationPeriod;
    //Creating textfields for data entry
```

Figure 15: Error 1

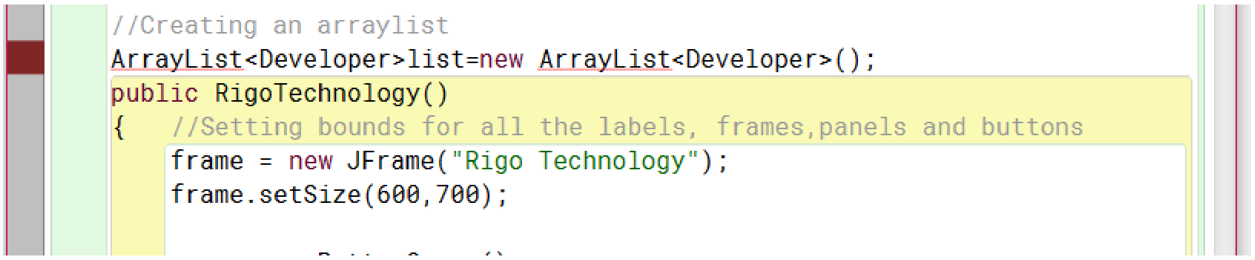


```
public void actionPerformed(ActionEvent e)
{   //Calling methods when certain command equals given value
    if (e.getActionCommand().equals("Add Junior"))
    {
        addJunior();
    }
    else if (e.getActionCommand().equals("Add Senior"))
    {
        addSenior();
    }
    else if (e.getActionCommand().equals("Senior"))
    {   //Making required textfields and labels only visible
```

Figure 16: Error 1 Solution

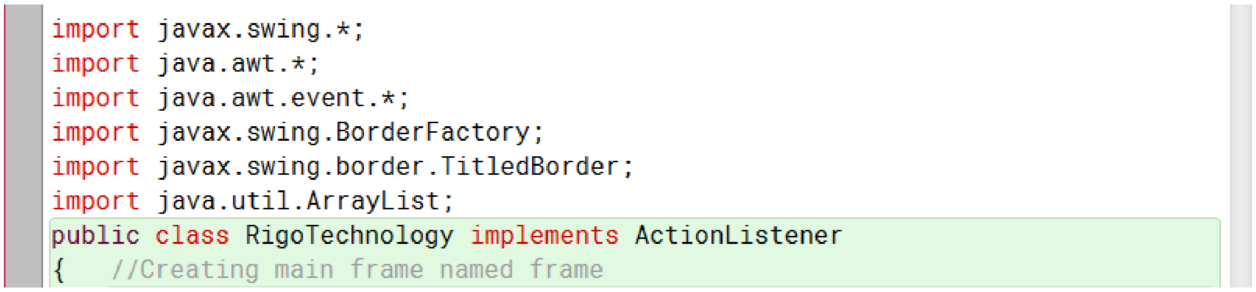
While implementing the ActionListener to the class RigoTechnology, an error occurs and it is solved by overriding an abstract method actionPerformed.

❖ **Error 2:**



```
//Creating an arraylist
ArrayList<Developer>list=new ArrayList<Developer>();
public RigoTechnology()
{   //Setting bounds for all the labels, frames,panels and buttons
    frame = new JFrame("Rigo Technology");
    frame.setSize(600,700);
```

Figure 17: Error 2

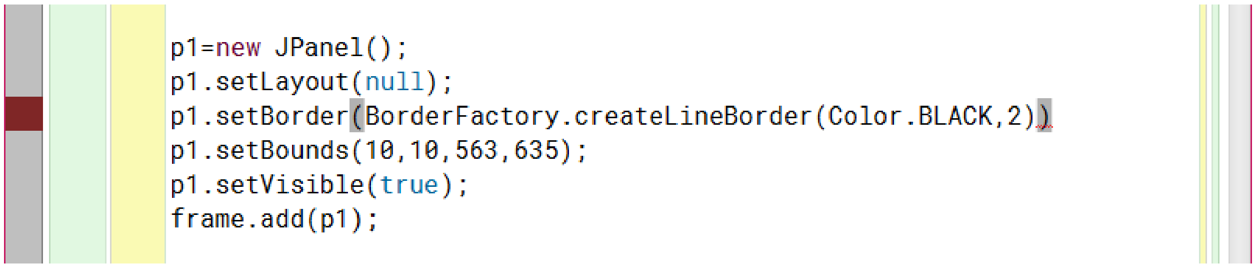


```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.BorderFactory;
import javax.swing.border.TitledBorder;
import java.util.ArrayList;
public class RigoTechnology implements ActionListener
{   //Creating main frame named frame
```

Figure 18: Error 2 Solution

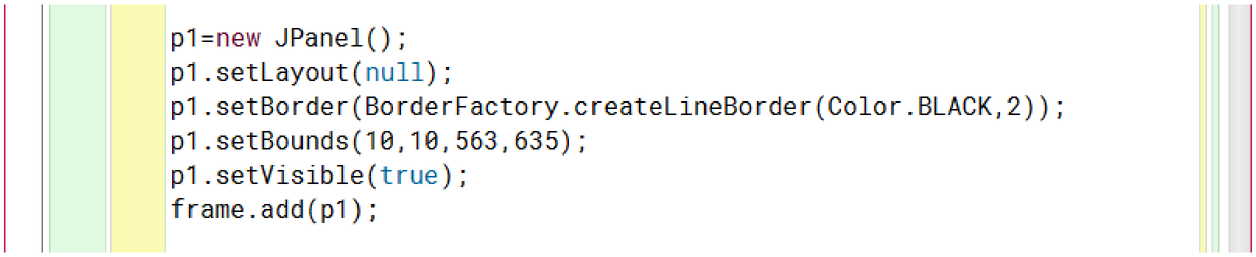
An error occurs while creating an arraylist, which is solved by importing arraylist from java util package.

❖ **Error 3:**



```
p1=new JPanel();
p1.setLayout(null);
p1.setBorder(BorderFactory.createLineBorder(Color.BLACK,2))
p1.setBounds(10,10,563,635);
p1.setVisible(true);
frame.add(p1);
```

Figure 19: Error 3



```
p1=new JPanel();
p1.setLayout(null);
p1.setBorder(BorderFactory.createLineBorder(Color.BLACK,2));
p1.setBounds(10,10,563,635);
p1.setVisible(true);
frame.add(p1);
```

Figure 20: Error 3 Solution

It is a simple error we face while coding and that is missing of the ";" at the end of the code which is solved by adding ";" at the end of every command.

Conclusion:

After the success of the project, the concept of creating GUI and exceptional handling became clear. Many problems were faced while constructing the GUI but with the help of intense research and hardwork, a frictionless program was created. The aim of the project was to create a GUI for developer appointment system. The main motto of the program is to clear the concept of event handling in java.

While starting the research, basic concepts were only implemented but as soon as researches were made, many more concepts were introduced and programming became easier. I started researches from books, journals and websites, in the relevant topic and built some concrete knowledge on making GUI. I encountered many issues while writing the code. To overcome those issues and problems, I consulted with our module teacher Mr, Saroj Kumar Yadav. Thus, after many trials and encountering many errors, this program was completed successfully.

Many journals and websites were accessed to debug the errors which were encountered, Discussion and lecture slides handy as well. This project was interesting and fruitful to learn more about this particular programming language. This project can be helpful for the students and beginners of java programming language who want to create a GUI for any system.

Finally, working with this programming language for building a developer appointment system was fruitful because it cleared more concepts of java programming language.

Appendix:

1 Developer:

```
public class Developer
{
    protected String platform;
    protected String interviewerName;
    protected String developerName;
    protected int workingHours;
    public Developer(String platform,String interviewerName,int workingHours)
    {
        this.platform=platform;
        this.interviewerName=interviewerName;
        this.workingHours=workingHours;
        this.developerName="";
    }
    public String getPlatform()
    {
        return this.platform;
    }
    public String getInterviewerName()
    {
        return this.interviewerName;
    }
    public String getDeveloperName()
    {
        return this.developerName;
    }
    public int getWorkingHours()
    {
        return this.workingHours;
    }
    public void setDeveloperName(String developerName)
    {
        this.developerName=developerName;
    }
    public void display()
    {
        System.out.println("The working hours is "+this.workingHours);
        System.out.println("The platform is "+this.platform);
        System.out.println("The interviewer name is "+this.interviewerName);
        if(this.developerName!="")
        {
            System.out.println("The developer name is "+this.developerName);
        }
    }
}
```

2 Senior Developer:

```
public class SeniorDeveloper extends Developer
{
    protected int salary;
    protected String joiningDate;
    protected String staffRoomNumber;
    protected int contractPeriod;
    protected double advanceSalary;
    protected boolean appointed;
    protected boolean terminated;
    public SeniorDeveloper(String platform, String interviewerName,int workingHours,int
salary,int contractPeriod)
    {
        super(platform,interviewerName,workingHours);
        this.salary=salary;
        this.contractPeriod=contractPeriod;
        this.joiningDate="";
        this.staffRoomNumber="";
        this.advanceSalary=0.0;
        this.appointed=false;
        this.terminated=false;
```

```
}
public int getSalary()
{
    return this.salary;
}
public String getJoiningDate()
{
    return this.joiningDate;
}
public String getStaffRoomNumber()
{
    return this.staffRoomNumber;
}
public int getContractPeriod()
{
    return this.contractPeriod;
}
public double getAdvanceSalary()
{
    return this.advanceSalary;
}
public boolean getAppointed()
{
    return this.appointed;
}
public boolean getTerminated()
{
    return this.terminated;
```

```
}
public void setSalary(int salary)
{
    this.salary=salary;
}
public void setContractPeriod(int contractPeriod)
{
    this.contractPeriod=contractPeriod;
}
public void hiredeveloper(String developerName,String joiningDate,double
advanceSalary,String staffRoomNumber)
{
    if(this.appointed)
    {
        System.out.println(this.developerName+"has already been selected!");


    }
    else{
    this.setDeveloperName(developerName);
    this.joiningDate=joiningDate;
    this.staffRoomNumber=staffRoomNumber;
    this.advanceSalary=advanceSalary;
    this.appointed=true;
    this.terminated=false;
    System.out.println("The developer "+this.developerName+" has been appointed");
}
}
public void contractTermination()
{
    if (this.terminated)
    {
        System.out.println("Contract has already been terminated");
    }
    else{
        this.setDeveloperName("");
        this.joiningDate="";
        this.advanceSalary=0.0;
        this.appointed=false;
        this.terminated=true;
    }
}
public void print()
{
    System.out.println("The platform is "+getPlatform());
    System.out.println("The interviewer name is "+getInterviewerName());
    System.out.println("The developer salary is "+getSalary());

}
public void display()
{
    super.display();
    if(appointed);
    {
      System.out.println("The terminated status is "+this.terminated);
      System.out.println("The joining date is "+this.joiningDate);
      System.out.println("The advance salary is "+this.advanceSalary);
      System.out.println("The developer name is "+this.developerName);
    }



}




}
```

3 Junior Developer

```
public class JuniorDeveloper extends Developer
{
    protected int salary;
    protected String appointedDate;
    protected String evaluationPeriod;
    protected String terminationDate;
    protected String specialization;
    protected String appointedBy;
    protected boolean joined;
    public JuniorDeveloper(String platform,String interviewerName,int workingHours,int
salary,String appointedBy,String terminationDate)
    {
        super(platform,interviewerName,workingHours);
        this.salary=salary;
        this.appointedDate="";
        this.appointedBy="";
        this.terminationDate=terminationDate;
        this.evaluationPeriod="";
        this.specialization="";
        this.joined=false;


    }
        public int getSalary()
    {
        return this.salary;
    }
    public String getAppointedDate()
    {
        return this.appointedDate;
    }
    public String getEvaluationPeriod()
    {
        return this.evaluationPeriod;
    }
    public String getTerminationDate()
    {
        return this.terminationDate;
    }
    public String getSpecialization()
    {
        return this.specialization;
    }
    public String getAppointedBy()
```

```
    {
        return this.appointedBy;
    }
    public Boolean getJoined()
    {
        return this.joined;
    }
    public void setSalary(int salary)
    {

        if (!(joined))
        {
            this.salary=salary;
        }
        else{
            System.out.println("Salary cannot be changed");
        }



    }
    public void appointDeveloper(String developerName,String appointedDate,String
terminationDate,String specialization,String evaluationPeriod, String appointedBy)
    {
        if(joined==false)
        {
            super.setDeveloperName(developerName);
            this.joined=true;
        }
        else{
            System.out.println("The developer is already appointed in"+this.appointedDate);
        }
        this.evaluationPeriod=evaluationPeriod;
        this.specialization=specialization;
        this.appointedDate=appointedDate;
        this.appointedBy=appointedBy;



    }
    public void diplay()
    {
        super.display();
        if(joined==true)
        {
            System.out.println("You will be appointed at"+this.appointedDate);
            System.out.println("You will be appointed at"+this.salary);
            System.out.println("You will be evaluated"+this.evaluationPeriod);
            System.out.println("You will be appointed by"+this.appointedBy);
            System.out.println("Your specialization is"+this.specialization);
            System.out.println("The termination of your contract will
be"+this.terminationDate);
        }


    }
```

```
}
```

4 RigoTechnology:

```
/**
 * Write a description of class REE here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.BorderFactory;
import javax.swing.border.TitledBorder;
import java.util.ArrayList;
public class RigoTechnology implements ActionListener
{   //Creating main frame named frame
    private JFrame frame;
    //Creating necessary labels
    private JLabel l1,Platform,InterviewerName,WorkingHours,
          Salary,AppointedBy,TerminationDate,ContractPeriod,
          Name,DevNo,JoiningDate,RoomNo,AdvSalary,NameJ,Specialization,
          DevNoJ,AppointDate,TerminateDate,AppointedByJ,EvaluationPeriod;
    //Creating textfields for data entry
    private JTextField platform,interviewerName,salary,appointedBy,
              terminationDate,contractPeriod,name,devNo,joiningDate,
              roomNo,advSalary,nameJ,specialization,devNoJ,
              appointDate,terminateDate,appointedByJ,evaluationPeriod;
    //Creating drop down box
    private JComboBox workingHours;
    //Creating buttons
    private JButton AddJunior ,AddSenior, Appoint , Terminate, AppointJuniorDeveloper,
Display, Clear;
//Creating three panels
    private JPanel p1,p2,p3;
    //Creating lines
    private JSeparator s1;
    //Creati9ng radiobuttons
    private JRadioButton Junior,Senior;
    ButtonGroup group;
    //Creating an arraylist
    ArrayList<Developer>list=new ArrayList<Developer>();
    public RigoTechnology()
    {   //Setting bounds for all the labels, frames,panels and buttons
        frame = new JFrame("Rigo Technology");
        frame.setSize(600,700);

        group=new ButtonGroup();

        p1=new JPanel();
        p1.setLayout(null);
        p1.setBorder(BorderFactory.createLineBorder(Color.BLACK,2));
        p1.setBounds(10,10,563,635);
        p1.setVisible(true);
        frame.add(p1);

        p2=new JPanel();
        p2.setLayout(null);
        p2.setBorder(BorderFactory.createTitledBorder(null,"Appoint/Terminate senior developer",TitledBorder.LEFT, TitledBorder.TOP,new Font("Arial", Font.BOLD,15), Color.BLUE));

        p2.setBounds(15,225,535,160);

        p2.setVisible(true);

        p1.add(p2);


        p3=new JPanel();

        p3.setLayout(null);

        p3.setBorder(BorderFactory.createTitledBorder(null,"Appoint junior developer",TitledBorder.LEFT, TitledBorder.TOP,new Font("Arial", Font.BOLD,15), Color.BLUE));

        p3.setBounds(15,400,535,190);

        p3.setVisible(true);

        p1.add(p3);

    l1=new JLabel("Add platform for senior/junior developer");

        l1.setBounds(150,5,300,30);

        l1.setFont(new Font("Arial",Font.BOLD,15));

        l1.setForeground(Color.BLUE);

        p1.add(l1);


        s1=new JSeparator();

        s1.setBackground(Color.black);

        s1.setBounds(2,30,563,5);

        p1.add(s1);


        Platform = new JLabel("Platform: ");

        Platform.setBounds(20,40,130,20);

        Platform.setFont(new Font("Arial",Font.BOLD,15));

        p1.add(Platform);
```

```
InterviewerName=new JLabel("Interviewer Name:");
InterviewerName.setBounds(20,70,130,20);
InterviewerName.setFont(new Font("Arial",Font.BOLD,15));
p1.add(InterviewerName);

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

WorkingHours=new JLabel("Working Hours:");
WorkingHours.setBounds(290,100,130,20);
WorkingHours.setFont(new Font("Arial",Font.BOLD,15));
p1.add(WorkingHours);

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

TerminationDate=new JLabel("Termination Date:");
TerminationDate.setBounds(290,130,130,20);
TerminationDate.setFont(new Font("Arial",Font.BOLD,15));
p1.add(TerminationDate);

ContractPeriod=new JLabel("Contract Period");
ContractPeriod.setBounds(20,160,130,20);
ContractPeriod.setFont(new Font("Arial",Font.BOLD,15));
p1.add(ContractPeriod);

Name=new JLabel("Name:");
Name.setBounds(20,40,100,20);
Name.setFont(new Font("Arial",Font.BOLD,15));
p2.add(Name);

JoiningDate=new JLabel("Joining Date:");
JoiningDate.setBounds(270,40,130,20);
JoiningDate.setFont(new Font("Arial",Font.BOLD,15));
p2.add(JoiningDate);

DevNo=new JLabel("Dev No.:");
DevNo.setBounds(20,70,130,20);
DevNo.setFont(new Font("Arial",Font.BOLD,15));
p2.add(DevNo);

RoomNo=new JLabel("Room No.:");
RoomNo.setBounds(205,70,80,20);
RoomNo.setFont(new Font("Arial",Font.BOLD,15));
p2.add(RoomNo);

AdvSalary=new JLabel("Adv Salary:");
AdvSalary.setBounds(350,70,80,20);
AdvSalary.setFont(new Font("Arial",Font.BOLD,15));
p2.add(AdvSalary);
```

```
NameJ=new JLabel("Name:");
NameJ.setBounds(20,30,100,20);
NameJ.setFont(new Font("Arial",Font.BOLD,15));
p3.add(NameJ);

Specialization=new JLabel("Specialization:");
Specialization.setBounds(270,30,130,20);
Specialization.setFont(new Font("Arial",Font.BOLD,15));
p3.add(Specialization);

DevNoJ=new JLabel("Dev No.:");
DevNoJ.setBounds(20,60,130,20);
DevNoJ.setFont(new Font("Arial",Font.BOLD,15));
p3.add(DevNoJ);

AppointDate=new JLabel("Appoint Date:");
AppointDate.setBounds(20,90,100,20);
AppointDate.setFont(new Font("Arial",Font.BOLD,15));
p3.add(AppointDate);

TerminateDate=new JLabel("Terminate Date:");
TerminateDate.setBounds(270,90,130,20);
TerminateDate.setFont(new Font("Arial",Font.BOLD,15));
p3.add(TerminateDate);

AppointedByJ=new JLabel("Appointed By:");
AppointedByJ.setBounds(270,60,130,20);
AppointedByJ.setFont(new Font("Arial",Font.BOLD,15));
```

```
p3.add(AppointedByJ);

EvaluationPeriod=new JLabel("Evaluation Period:");
EvaluationPeriod.setBounds(20,120,130,20);
EvaluationPeriod.setFont(new Font("Arial",Font.BOLD,15));
p3.add(EvaluationPeriod);

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

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

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

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

terminationDate=new JTextField();
terminationDate.setBounds(420,130,130,20);
p1.add(terminationDate);

contractPeriod=new JTextField();
```

```
contractPeriod.setBounds(150,160,130,20);
p1.add(contractPeriod);

name=new JTextField();
name.setBounds(135,40,130,20);
p2.add(name);

joiningDate=new JTextField();
joiningDate.setBounds(400,40,120,20);
p2.add(joiningDate);

devNo=new JTextField();
devNo.setBounds(135,70,60,20);
p2.add(devNo);

roomNo=new JTextField();
roomNo.setBounds(285,70,60,20);
p2.add(roomNo);

advSalary=new JTextField();
advSalary.setBounds(435,70,84,20);
p2.add(advSalary);

nameJ=new JTextField();
nameJ.setBounds(135,30,130,20);
p3.add(nameJ);

specialization=new JTextField();
```

```
specialization.setBounds(400,30,120,20);
p3.add(specialization);

devNoJ=new JTextField();
devNoJ.setBounds(135,60,60,20);
p3.add(devNoJ);

appointDate=new JTextField();
appointDate.setBounds(135,90,130,20);
p3.add(appointDate);

terminateDate=new JTextField();
terminateDate.setBounds(400,90,120,20);
p3.add(terminateDate);

appointedByJ=new JTextField();
appointedByJ.setBounds(400,60,120,20);
p3.add(appointedByJ);

evaluationPeriod=new JTextField();
evaluationPeriod.setBounds(150,120,120,20);
p3.add(evaluationPeriod);

workingHours=new JComboBox();
workingHours.addItem(4);
workingHours.addItem(5);
workingHours.addItem(6);
workingHours.addItem(7);
workingHours.addItem(8);
workingHours.addItem(9);
workingHours.addItem(10);
workingHours.addItem(11);
workingHours.addItem(12);
workingHours.setBounds(420,100,130,20);
p1.add(workingHours);

Junior=new JRadioButton("Junior");
Junior.setBounds(290,160,120,20);
Junior.setForeground(Color.BLUE);
Junior.setActionCommand("Junior");
Junior.addActionListener(this);
group.add(Junior);
p1.add(Junior,true);

Senior=new JRadioButton("Senior");
Senior.setBounds(420,160,130,20);
Senior.setForeground(Color.BLUE);
Senior.setActionCommand("Senior");
Senior.addActionListener(this);
group.add(Senior);
p1.add(Senior);

group.clearSelection();

AddJunior=new JButton("Add Junior");
AddJunior.setBounds(290,190,120,20);
AddJunior.setForeground(Color.BLUE);
AddJunior.setActionCommand("Add Junior");
AddJunior.addActionListener(this);
p1.add(AddJunior);

AddSenior=new JButton("Add Senior");
AddSenior.setBounds(420,190,130,20);
AddSenior.setForeground(Color.BLUE);
AddSenior.setActionCommand("Add Senior");
AddSenior.addActionListener(this);
p1.add(AddSenior);
Appoint=new JButton("Appoint");
Appoint.setBounds(270,110,110,20);
Appoint.setForeground(Color.GREEN);
Appoint.setActionCommand("Appoint");
Appoint.addActionListener(this);
p2.add(Appoint);

Terminate=new JButton("Terminate");
Terminate.setBounds(400,110,120,20);
Terminate.setForeground(Color.RED);
Terminate.setActionCommand("Terminate");
Terminate.addActionListener(this);
p2.add(Terminate);

AppointJuniorDeveloper=new JButton("Appoint Junior Developer");
AppointJuniorDeveloper.setBounds(320,150,200,20);
AppointJuniorDeveloper.setForeground(Color.BLUE);
```

```
        AppointJuniorDeveloper.setActionCommand("Appoint Junior Developer");
        AppointJuniorDeveloper.addActionListener(this);
        p3.add(AppointJuniorDeveloper);

        Display=new JButton("Display");
        Display.setBounds(290,600,120,20);
        Display.setForeground(Color.GREEN);
        Display.setActionCommand("Display");
        Display.addActionListener(this);
        p1.add(Display);

        Clear=new JButton("Clear");
        Clear.setBounds(420,600,130,20);
        Clear.setForeground(Color.RED);
        Clear.setActionCommand("Clear");
        Clear.addActionListener(this);
        p1.add(Clear);

        frame.setLayout(null);
        frame.setVisible(true);
    }
    @Override
    public void actionPerformed(ActionEvent e)
    {   //Calling methods when certain command equals given value
        if (e.getActionCommand().equals("Add Junior"))
        {
            addJunior();
        }
```

```
else if (e.getActionCommand().equals("Add Senior"))
{
  addSenior();
}
else if (e.getActionCommand().equals("Senior"))
{  //Making required textfields and labels only visible
  appointedBy.setVisible(false);
  terminationDate.setVisible(false);
  AppointedBy.setVisible(false);
  TerminationDate.setVisible(false);
  contractPeriod.setVisible(true);
  ContractPeriod.setVisible(true);
}

else if (e.getActionCommand().equals("Junior"))
{  //Making required are visible
  contractPeriod.setVisible(false);
  ContractPeriod.setVisible(false);
  appointedBy.setVisible(true);
  terminationDate.setVisible(true);
  AppointedBy.setVisible(true);
  TerminationDate.setVisible(true);
}
else if (e.getActionCommand().equals("Appoint"))
{
  appoint();
}
else if (e.getActionCommand().equals("Terminate"))
```

```
        {
          terminate();
        }
        else if (e.getActionCommand().equals("Appoint Junior Developer"))
        {
          appointJuniorDeveloper();
        }
        else if (e.getActionCommand().equals("Display"))
        {
          display();
        }
        else if (e.getActionCommand().equals("Clear"))
        {
          clear();
        }
    }
    public void addJunior()
    {
        try
        {
           String Platform=platform.getText();
           String InterviewerName=interviewerName.getText();
           int WorkingHours=(int) workingHours.getSelectedItem();
           String Salary=salary.getText();
           String AppointedBy=appointedBy.getText();
           String TerminationDate=terminationDate.getText();
           //Checking if the field is empty
           if(Platform.isEmpty() || InterviewerName.isEmpty() || Salary.isEmpty()
||AppointedBy.isEmpty()|| TerminationDate.isEmpty())
```

```
            {
                throw new Exception("Cannot leave the field blank!");


            }
                //converting salary to int
                int salary1=Integer.parseInt(Salary);
                //creating object
                JuniorDeveloper                                              junior1=new
JuniorDeveloper(Platform,InterviewerName,WorkingHours,salary1,AppointedBy,Termin
ationDate);
                //Adding the object to the arraylist
                list.add(junior1);
                JOptionPane.showMessageDialog(frame,"The junior developer is successfully
added","Information",JOptionPane.INFORMATION_MESSAGE);
            }
        catch(Exception e1)
        {
            //Fetching the message
            JOptionPane.showMessageDialog(frame,e1.getMessage(),"ERROR
MESSAGE",JOptionPane.ERROR_MESSAGE);
        }



    }


    public void addSenior()
    {
        try
        {   //storing the data of textfield in a variable
            String Platform=platform.getText();
            String InterviewerName=interviewerName.getText();
            int WorkingHours=(int) workingHours.getSelectedItem();
            String Salary=salary.getText();
            String ContractPeriod=contractPeriod.getText();
            if(Platform.isEmpty() || InterviewerName.isEmpty() || Salary.isEmpty()
||ContractPeriod.isEmpty())
            {
                throw new Exception("Cannot leave the field blank!");


            }
                int salary1=Integer.parseInt(Salary);
                int ContractPeriod1=Integer.parseInt(ContractPeriod);
                SeniorDeveloper senior1=new
SeniorDeveloper(Platform,InterviewerName,WorkingHours,salary1,ContractPeriod1);
                list.add(senior1);
                JOptionPane.showMessageDialog(frame,"The senior developer is
successfully added","Information",JOptionPane.INFORMATION_MESSAGE);
            }
        catch(Exception e2)
        {
            JOptionPane.showMessageDialog(frame,e2.getMessage(),"ERROR
MESSAGE",JOptionPane.ERROR_MESSAGE);
        }
    }



   public void appoint()
    {
        try
        {
```

```
        String Name=name.getText();
        String JoiningDate=joiningDate.getText();
        String DevNo=devNo.getText();
        String RoomNo=roomNo.getText();
        String AdvSalary=advSalary.getText();
        if(Name.isEmpty() || JoiningDate.isEmpty() || DevNo.isEmpty()
||RoomNo.isEmpty() || AdvSalary.isEmpty())
        {
           JOptionPane.showMessageDialog(frame,"Please enter all the required
fields.","Error",JOptionPane.ERROR_MESSAGE);
        }
        //checking the data type
        else if(!(Name.matches("^[A-Za-z]+$")))
        {
           JOptionPane.showMessageDialog(frame,"This field prefers only
text","Invalid",JOptionPane.ERROR_MESSAGE);
        }
        else if(Integer.parseInt(AdvSalary)<4000 ||Integer.parseInt(AdvSalary)>10000)
        {
           JOptionPane.showMessageDialog(frame,"Given Advance Salary is
invalid!","Invalid",JOptionPane.ERROR_MESSAGE);
        }
        else
        {
           int AdvSalary1=Integer.parseInt(AdvSalary);
           int DevNo1=Integer.parseInt(DevNo);
           if (DevNo1>=0 && DevNo1<list.size())
           {
              if(list.get(DevNo1)instanceof SeniorDeveloper)
              {
                    SeniorDeveloper s1=(SeniorDeveloper) list.get(DevNo1);
                    if(s1.getAppointed()==false)
                    {
                        s1.hiredeveloper(Name,JoiningDate,AdvSalary1,RoomNo);
                        JOptionPane.showMessageDialog(frame,"Congratulations!! the senior developer"+s1.getDeveloperName()+"having ID:"+DevNo1+ "has been appointed sucessfully","Message",JOptionPane.INFORMATION_MESSAGE);
                    }
                    else
                    {
                      JOptionPane.showMessageDialog(frame,"The Senior Developer having name "+s1.getDeveloperName()+" has already been appointed!","Message",JOptionPane.INFORMATION_MESSAGE);
                    }
                }
                else
                {
                    JOptionPane.showMessageDialog(frame,"The Developer No: " +DevNo1+" doesn't belong to Senior Developer!","Error",JOptionPane.INFORMATION_MESSAGE);
                }
            }
            else
            {
                JOptionPane.showMessageDialog(frame,"The Developer No: " +DevNo1+" doesn't belong to Senior Developer!","Error",JOptionPane.INFORMATION_MESSAGE);
            }
        }


    }
    catch(NumberFormatException e)
```

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


    }
    public void terminate()
    {
        try
        {
            String DevNo=devNo.getText();
            if( DevNo.isEmpty())
            {
                JOptionPane.showMessageDialog(frame,"Please enter DevNo.","Error",JOptionPane.ERROR_MESSAGE);
            }
            else
            {
                int DevNo1=Integer.parseInt(DevNo);
                if (DevNo1>=0 && DevNo1<list.size())
                {
                    if(list.get(DevNo1)instanceof SeniorDeveloper)
                    {
                        SeniorDeveloper s1=(SeniorDeveloper) list.get(DevNo1);
                        if(s1.getTerminated()==false)
                        {
                           s1.contractTermination();
                           JOptionPane.showMessageDialog(frame,"The senior developer"+s1.getDeveloperName()+"having ID:"+DevNo1+ "has been terminated sucessfully","Message",JOptionPane.INFORMATION_MESSAGE);
                        }
                        else
                        {
                          JOptionPane.showMessageDialog(frame,"The Senior Developer having name "+s1.getDeveloperName()+" has already been terminated!","Error",JOptionPane.INFORMATION_MESSAGE);
                        }
                    }
                    else
                    {
                        JOptionPane.showMessageDialog(frame,"The Developer No: " +DevNo1+" doesn't belong to Senior Developer!","Error",JOptionPane.INFORMATION_MESSAGE);
                    }
                }
                else
                {
                    JOptionPane.showMessageDialog(frame,"The Developer No: " +DevNo1+" doesn't belong to Senior Developer!","Error",JOptionPane.INFORMATION_MESSAGE);
                }
            }


        }
        catch(NumberFormatException e)
        {
            JOptionPane.showMessageDialog(frame,"Data type for DevNo is integer!!","Error",JOptionPane.INFORMATION_MESSAGE);
        }
        catch(NullPointerException e)
        {
            JOptionPane.showMessageDialog(frame,"Please enter the value!","Error",JOptionPane.INFORMATION_MESSAGE);
        }
    }
    public void appointJuniorDeveloper()
    {
      try
        {
            String Name=nameJ.getText();
            String Specialization=specialization.getText();
            String DevNo=devNoJ.getText();
            String AppointedByJ=appointedByJ.getText();
            String AppointDate=appointDate.getText();
            String TerminateDate=terminateDate.getText();
            String EvaluationPeriod=evaluationPeriod.getText();
            if(Name.isEmpty() ||AppointedByJ.isEmpty() || EvaluationPeriod.isEmpty() || Specialization.isEmpty() || DevNo.isEmpty() ||AppointDate.isEmpty() || TerminateDate.isEmpty())
            {
                JOptionPane.showMessageDialog(frame,"Please enter all the required fields.","Error",JOptionPane.ERROR_MESSAGE);
            }
            else if(!(Name.matches("^[A-Za-z]+$")))
            {
```

```
            JOptionPane.showMessageDialog(frame,"This field requires only text!","Invalid",JOptionPane.ERROR_MESSAGE);
         }
         else if(!(Specialization.matches("^[A-Za-z]+$")))
         {
            JOptionPane.showMessageDialog(frame,"This field requires only text!","Invalid",JOptionPane.ERROR_MESSAGE);
         }
         else
         {
            int DevNoJ1=Integer.parseInt(DevNo);
            if (DevNoJ1>=0 && DevNoJ1<list.size())
            {
               if(list.get(DevNoJ1)instanceof JuniorDeveloper)
               {
                  JuniorDeveloper j1=(JuniorDeveloper) list.get(DevNoJ1);
                  if(j1.getJoined()==false)
                  {
j1.appointDeveloper(Name,AppointDate,TerminateDate,Specialization,EvaluationPeriod,AppointedByJ);
                      JOptionPane.showMessageDialog(frame,"Congratulations!! the junior developer has been appointed sucessfully!","Message",JOptionPane.INFORMATION_MESSAGE);

                  }
                  else
                  {
                     JOptionPane.showMessageDialog(frame,"The Junior Developer having name "+j1.getDeveloperName()+" has already been appointed!","Message",JOptionPane.INFORMATION_MESSAGE);
```

```
                    }
                }
                else
                {
                    JOptionPane.showMessageDialog(frame,"The Developer No: " +DevNoJ1+" doesn't belong to Senior Developer!","Error",JOptionPane.INFORMATION_MESSAGE);
                }
            }
            else
            {
                JOptionPane.showMessageDialog(frame,"The Developer No: " +DevNoJ1+" doesn't belong to Senior Developer!","Error",JOptionPane.INFORMATION_MESSAGE);
            }
        }


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


}
```

```
  public void display()
 {  //checking if the arraylist is empty
    if(list.size()==0)
    {
       JOptionPane.showMessageDialog(frame,"No developers are added so nothing to display","Information!!!",JOptionPane.INFORMATION_MESSAGE);
    }

    else
    {//using for loop
    for(int i=0;i<list.size();i++)
    {
      if(list.get(i) instanceof SeniorDeveloper)
      {   //checking if the index is index of senior developer
          SeniorDeveloper s1=(SeniorDeveloper) list.get(i);
          System.out.println("Information of Senior Developer");
          s1.display();
          System.out.println();
      }

      else
      {   //checking if the index is index of senior developer
          JuniorDeveloper j1=(JuniorDeveloper) list.get(i);
          System.out.println("Information of Junior Developer");
          j1.display();
          System.out.println();
      }
    }
  }
 }
   public void clear()
   {  //clearing all the text fields
      platform.setText("");
      interviewerName.setText("");
      int WorkingHours=(int) workingHours.getSelectedItem();
      salary.setText("");
      appointedBy.setText("");
      terminationDate.setText("");
      contractPeriod.setText("");
      name.setText("");
      devNo.setText("");
      joiningDate.setText("");
      roomNo.setText("");
      advSalary.setText("");
      nameJ.setText("");
      specialization.setText("");
      appointDate.setText("");
      terminateDate.setText("");
      appointedByJ.setText("");
      evaluationPeriod.setText("");
      devNoJ.setText("");



   }
}
```