LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

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:

College ID:NP01NT4A180002

Assignment Due Date:1/25/2019

Assignment Submission Date:1/25/2019

Word Count (Where Required):2034

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 Figures:

List of Tables:

Introduction:

This project is given to clear the concept of inheritance in java. Here, three different classes (Developer class, Senior Developer and Junior Developer) are created which are associated with each other. Here, we have applied the concept of inheritance where Developer class is the parent class and senior developer and junior developer are child classes. Different methods are created (accessor and setters) in all classes. Here, child having constructor and methods can access variables of the parent class using the term super.

Class Diagram:

a) Developer Class Diagram

Developer
#platform: String #interviewerName: String #developerName: String #workingHours:int
+getPlatform():String +getInterviewerName():String +getDeveloperName():String +getWorkingHours():int +setDeveloperName(developerName:String):String +display(): void

b) SeniorDeveloper Class Diagram:

SeniorDeveloper
#salary: int #joiningDate: String #staffRoomNumber: String #contractPeriod: int #advanceSalary: double #appointed: boolean #terminated: boolean
+getSalary(): int +getJoiningDate(): String +getStaffRoomNumber(): String +getContractPeriod(): int +getAdvanceSalary(): double +getAppointed(): appointed +getTerminated(): terminated +setSalary(salary: int): int +setContractPeriod(contractPeriod: int): int +hireDeveloper(developerName: String, joiningDate: String, staffRoomNumber: String): void +contractTermination(): void +print(): void +display(): void

c) JuniorDeveloper Class Diagram:

JuniorDeveloper
#salary: int #appointedDate: String #evaluationPeriod: String #terminationDate: String #specialization: String #appointedBy: String #joined: boolean
+getSalary(): int +getAppointedDate(): String +getEvaluationPeriod(): String +getTerminationDate(): String +getSpecialization(): String +getAppointedBy(): String +getJoined(): boolean +setSalary(salary: int): int +appointDeveloper(developerName: String, appointedDate: String, terminationDate: String,specialization: String): void +display(): void

Pseudocode:

Developer Class:

```
CREATE A METHOD NAME getPlatform: String

DO

        RETURN VARIABLE platform

END DO

CREATE A METHOD NAME getInterviewerName: String

DO

        RETURN VARIABLE interviewerName

END DO

CREATE A METHOD NAME getDeveloperName: String

DO

        RETURN VARIABLE developerName

END DO

CREATE A METHOD NAME getWorkingHours: int

DO

        RETURN VARIABLE workingHours

END DO

CREATE METHOD NAME setDeveloperName: void WITH String PARAMETER

DO

        SET developerName TO VALUE IN PARAMETER

END DO

CREATE METHOD NAME display: void
```

```
DO
    DISPLAY platform, workinghours, interviewerName
    IF developerName!=”” THEN
        DISPLAY developerName
    END IF
END DO
```

Senior Developer Class:

```
CREATE METHOD NAME getSalary: int
DO
    RETURN VARIABLE salary
END DO
CREATE A METHOD getJoiningDate: String
DO
    RETURN VARIABLE joiningDate
END DO
CREATE A METHOD getStaffRoomNumber: String
DO
    RETURN VARIABLE staffRoomNumber
END DO
CREATE A METHOD getContractPeriod: int
DO
    RETURN VARIABLE contractPeriod
```

END DO

CREATE A METHOD NAME getAdvanceSalary: double

DO

RETURN VARIABLE advanceSalary

END DO

CREATE A METHOD NAME getAppointed: boolean

DO

RETURN VARIABLE appointed

END DO

CREATE A METHOD NAME getTerminated: boolean

DO

RETURN terminated

END DO

CREATE A METHOD NAME setSalary WITH int PARAMETER

DO

SET salary AS IN PARAMETER

END DO

CREATE A METHOD NAME setContractPeriod WITH int PARAMETER

DO

SET contractPeriod AS IN PARAMETER

END DO

CREATE A METHOD NAME hireDeveloper WITH PARAMETER

(developerName:String, joiningDate:String, advanceSalary: double, satffRoonNumber:

```
String)
DO
    IF appointed THEN
        DISPLAY developerName
    ELSE
        DISPLAY suitable output
    END IF
END DO
CREATE A METHOD contractTermination()
DO
    IF terminated THEN
        DISPLAY developer has already been terminated
    ELSE
        Call setDeveloperName(“ ”)
        SET joining date as empty string
        Assign 0.0 to adavanceSalary
        SET the Boolean value of terminated to True and appointed to False
    END IF
END DO
CREATE A METHOD print()
DO
    DISPLAY platform, interviewerName and salary
END DO
```

CREATE A METHOD display()

DO

    DISPLAY terminated, joiningDate, advanceSalary and developerName

END DO

Junior Developer Class

CREATE A METHOD getSalary()

DO

    RETURN salary

END DO

CREATE A METHOD getAppointedDate()

DO

    RETURN appointedDate

END DO

CREATE A METHOD getEvaluationPeriod()

DO

    RETURN evaluationPeriod

END DO

CREATE A METHOD getTerminationDate()

DO

    RETURN terminationDate

END DO

```
CREATE A METHOD getSpecialization()
DO
    RETURN specialization
END DO
CREATE A METHOD getAppointedBy()
DO
    RETURN appoitedBy
END DO
CREATE A METHOD getJoined()
DO
    RETURN joined
END DO
CREATE A METHOD setSalary()
DO
    IF joined THEN
        SET salary as in PARAMETER
    ELSE
        DISPLAY Salary cannot be changed
    END IF
END DO
CREATE A METHOD appointDeveleoper()
DO
    IF joined THEN
```

```
            CALL setDeveloperName using SUPER
            SET evaluationPeriod, specialization and appointedDate
            SET Boolean value of joined to True
        ELSE
            DISPLAY The developer is already appointed.
        END IF
END DO
CREATE A METHOD display()
DO
        CALL display() using SUPER
        IF joined==True THEN
            DISPLAY appointedDate, salary, evaluationPeriod, appointedBy,
            Specialization and terminationDate
        END IF
END DO
```

Method Description:

I. Method Description of Developer Class:

- getPlatform()

  It is an accessor method which is used to access the platform of every object of the Developer class.

- getInterviewerName()

  It is an accessor method which is used to access the interviewer name of every object of the developer class and its return type is string.

- getWorkingHours()

  In this accessor method, method is called to access the working hours of the object in Developer class and its return type is integer.

- setDeveloperName()

  It is a setter method which allows us to change the developer name of the object in Developer class and its return type is string.

- display()

  It is a method with void return type, whose function is to display all the information of the Developer class i.e. platform, interviewer name, working hours and developer name, if its value is already set.

II. Method Description of SeniorDeveloper Class:

- getSalary()

  It is an accessor method to access the salary of every object in SeniorDeveloper class whose return type is float.

- getJoiningDate()

  An accessor method to access the joining date of the object with the return type string.

- getStaffRoomNumber()

  An accessor method to access the staff room number of the object with the return type string.

- getContractPeriod()

  An accessor method to access the contract period of an object having return type integer.

- getAdvanceSalary()

  An accessor method to access the advance salary of the object with return type double return type.

- getAppointed()

  An accessor method to access the appointed status returning boolean data type.

- getTerminated()

  An accessor method to access the termination status with the return type boolean.

- setSalary()

  A setter method to set the salary of the object with return type integer.

- setContractPeriod()

  A setter method to set the contract period of the object in SeniorDeveloper class having return type interger.

- hireDeveloper()

  It is a setter method where developer name, joining date, advanced salary and staff room number are passed as parameters having no return value. This method checks the value of appointed, if the developer is appointed the developer name and staff room number is printed, else, the parent class with developer name as the parameter is called to set the developer name and value of joining date, staff room number and advance salary are updated aft6er passing the value while calling the method. Lastly, the boolean values of appointed and terminated are changed accordingly.

- contractTerminated()

  It is the mutator method which checks whether the contract period of object is terminated or not. If the termination is True, then a suitable output for termination is displayed.

- print()

  The function of this method is only to print out platform, interviewer name and salary of the object.

- display()

  This method checks the appointed value. If it is true, interviewer name, platform and working hours are printed else, if the developer is already appointed, additional

information like joining date, termination status, developer name and status is printed.

III. Method Description of JuniorDeveloper Class:

- getsalary()

  An accessror method to access the salary of the object having return type double.
- getAppointedDate()

  An accessor method that allows the access of appointed date of every instance of this class, having return type as string,
- getEvaluationPeriod()

  An accessor method to access the evaluation period of every instance of this class, having return type string.
- getTerminationDate()

  An accessor method to access the termination date of every instance of this class, having return type as string.
- getSpecialization()

  An accessor method to access the specialization of the object in the class, having return type string.
- getAppointedBy()

  An accessor method to access the appointer of every instance in the class, having return type string.
- getJoined()

  Accessor having return type boolean, to access the value of joined variable.
- setSalary()

  It is a setting method which sets the salary if the person's appointed boolean value is True. The change in salary cannot be made if the person is not appointed.
- appointDeveloper()

  It is a mutator method which checks the boolean value of joined. If joined, a suitable message is given saying the developer is already appointed. Else, calls the setDeveloperName method from the super class and sets the boolean value to False.
- display()

It is a non-return type method which calls display method from the superclass. It checks the boolean value of joined. If its true, prints out appointed date, evaluation period, termination date, salary, specialization and appointer.

Testing:

Test 1:

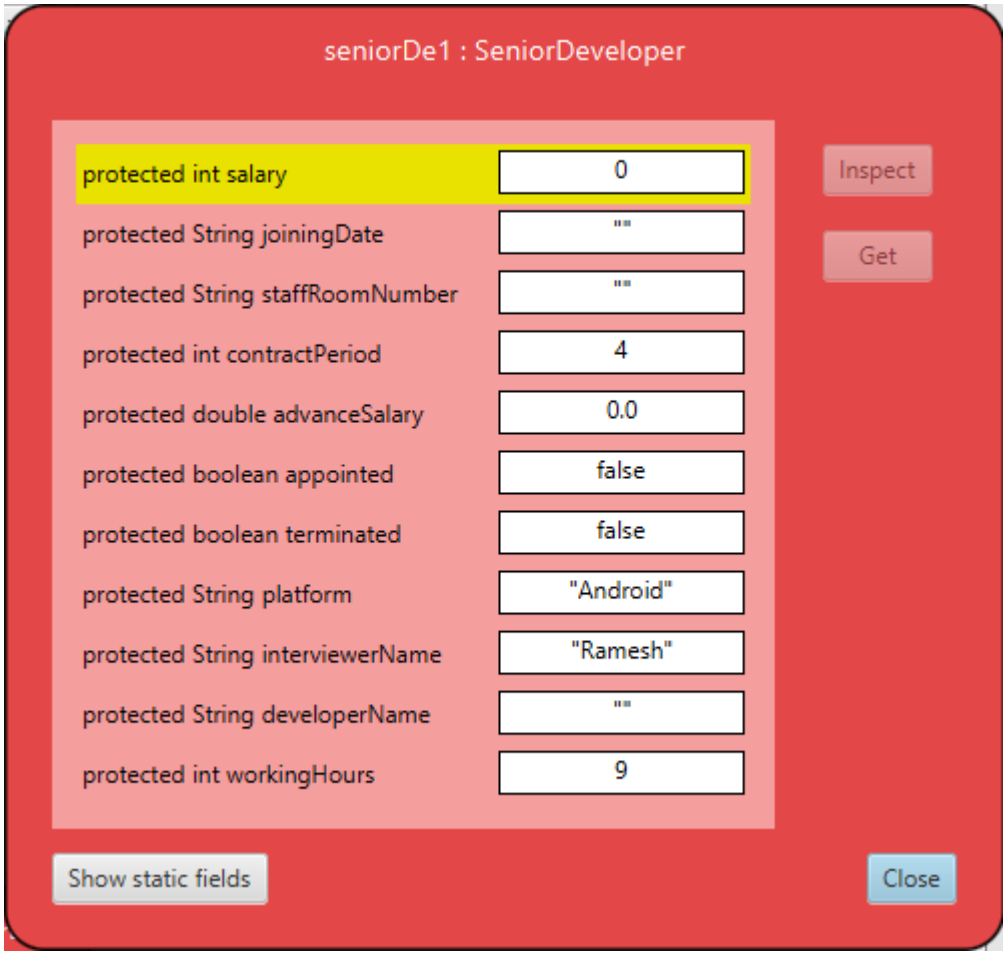

Figure 1: Inspecting Senior Developer Class

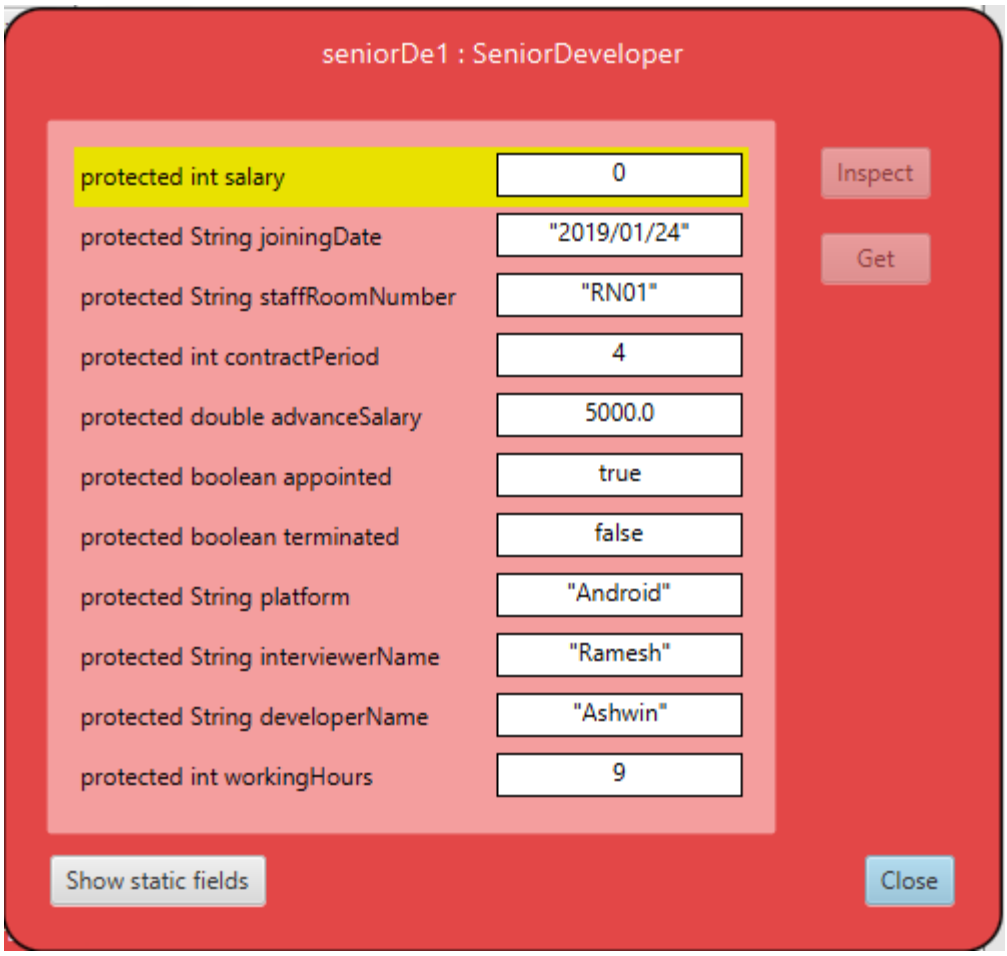

Figure 2: Output After Object Creation

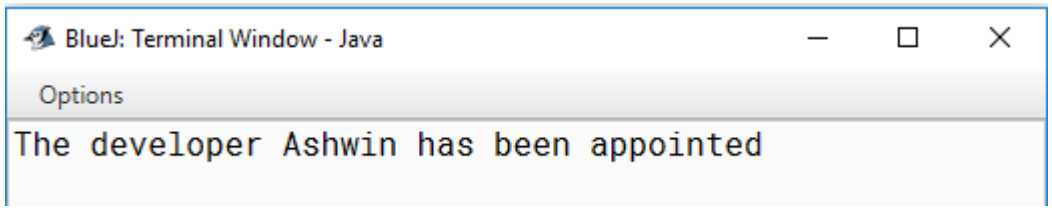

Figure 3: Inspecting After Hiring Developer

Test	1
Task Performed	Inspect, appoint developer, re-inspect.
Expected Result	After developer is appointed, changes should be made.
Actual Result	Developer was appointed and changes were made.
Test Result	Test Successful.

Table 1: Test 1 Table

Test 2:

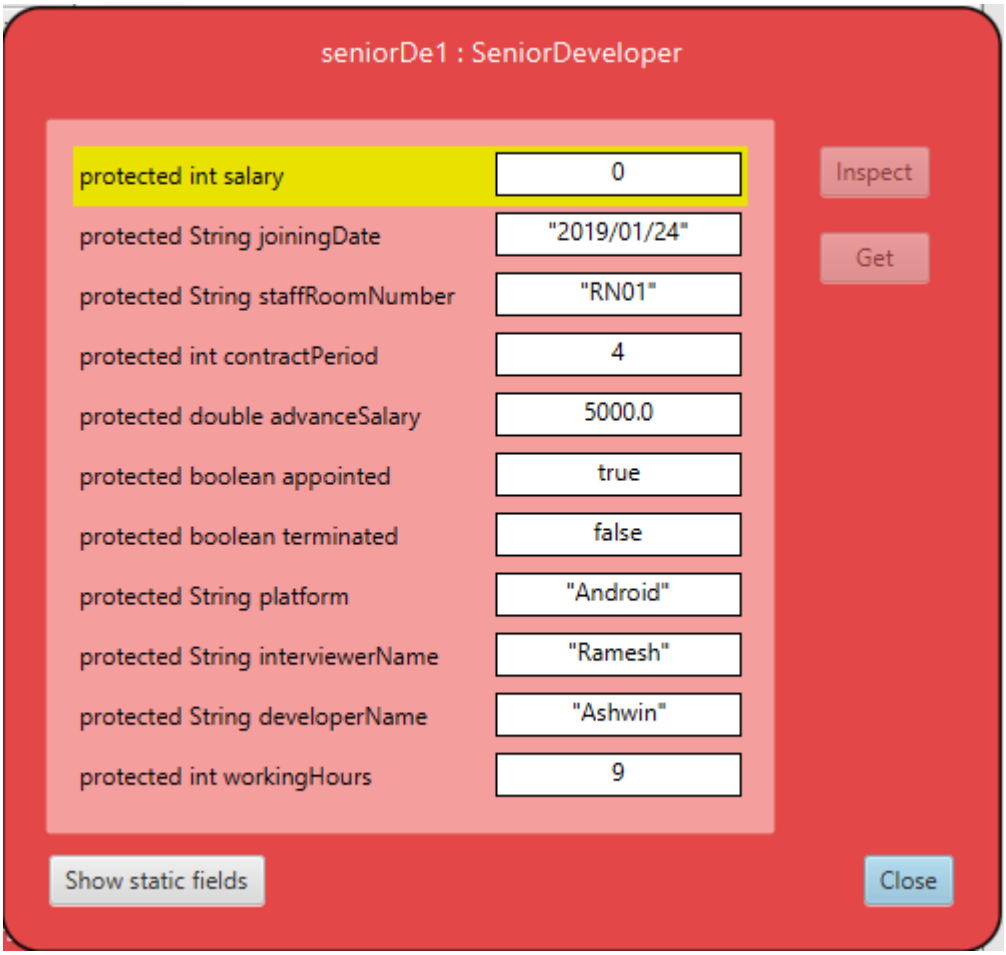

Figure 4: Inspecting Senior Developer

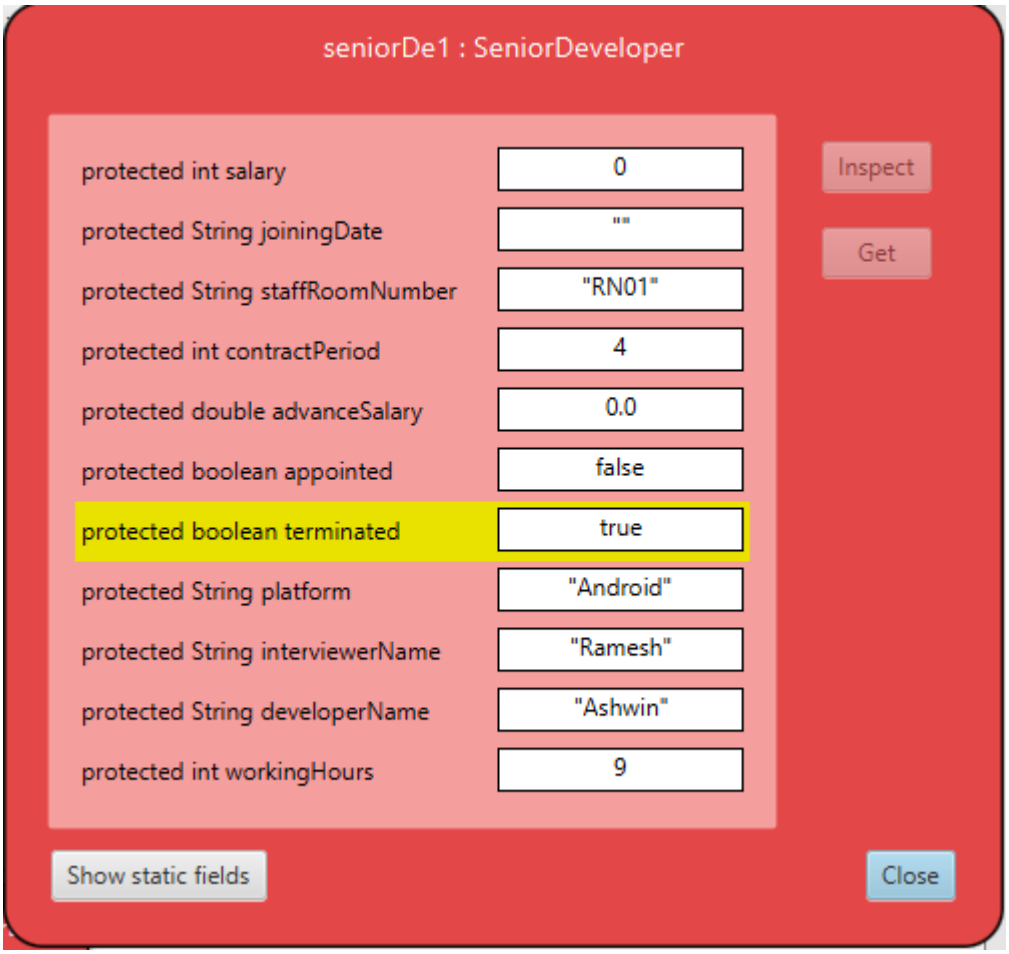

Figure 5: Inspecting After Termination

Test	2
Task Performed	Changing the appointed status.
Expected Result	Terminated status must be changed to True and appointed to False.
Actual Result	Terminated status was changed to False and appointed to True.
Test Result	Test Successful.

Table 2: Test 2 Table

Test 3:

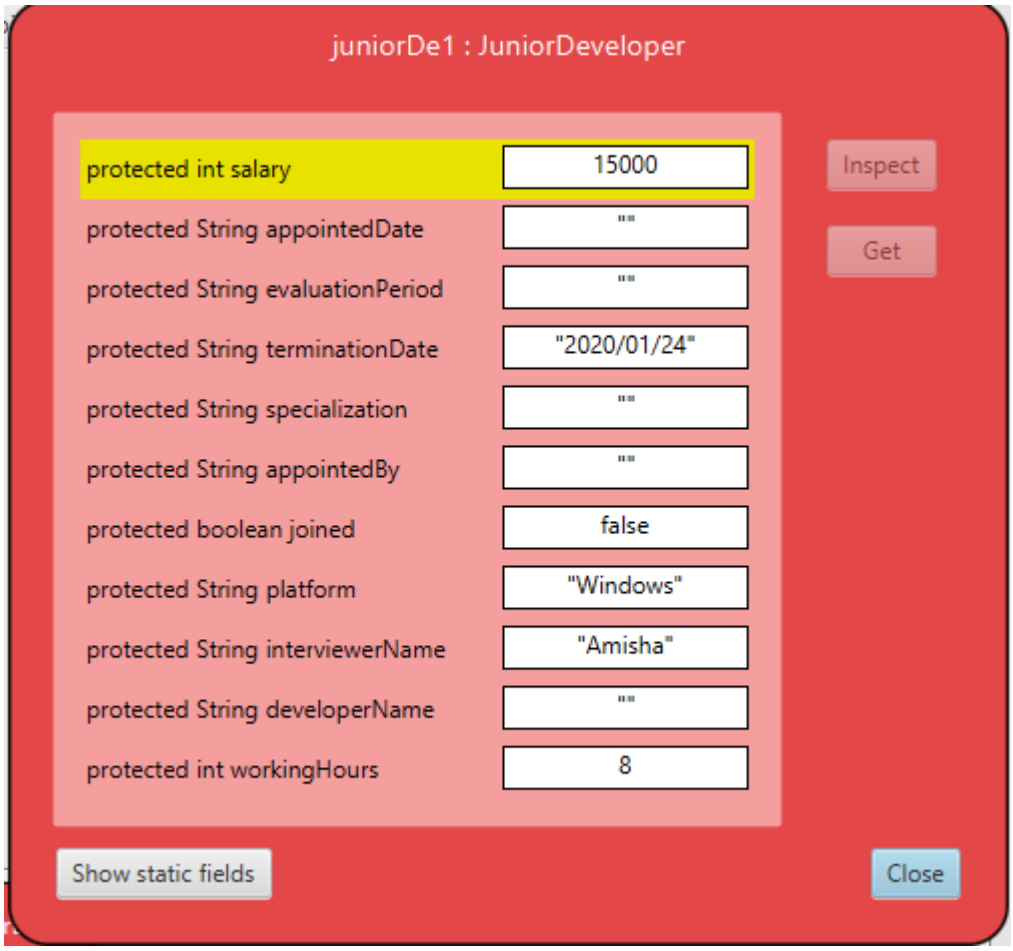

Figure 6: Inspecting Junior Developer

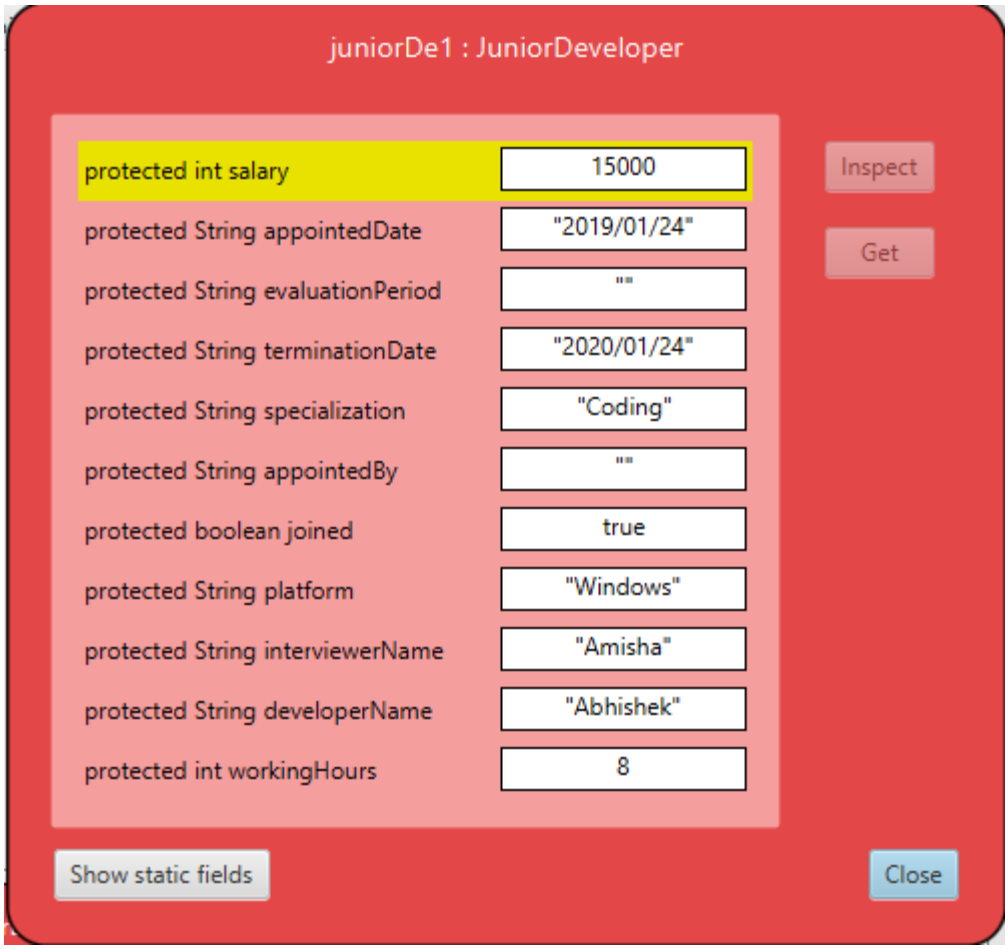

Figure 7: Inspection After Appointing Developer

Test	3
Task Performed	Inspect, appoint, re-inspect
Expected Result	After developer is appointed, changes must be made.
Actual Result	Changes were made after developer was appointed.
Test Result	Test Successful.

Table 3: Test 3 Table

Test 4:

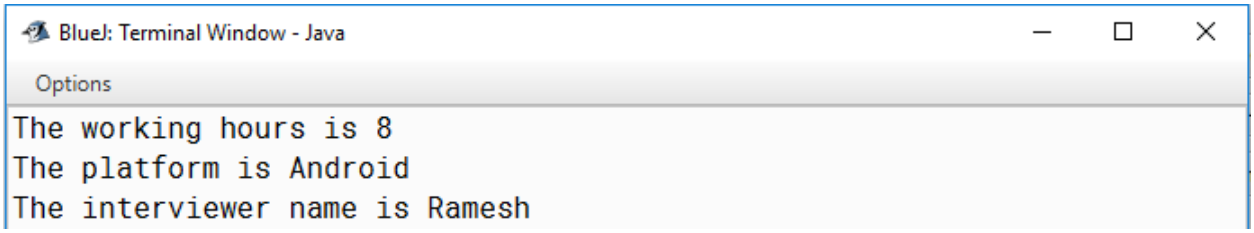

Figure 8: Display Method Output of Senior Developer

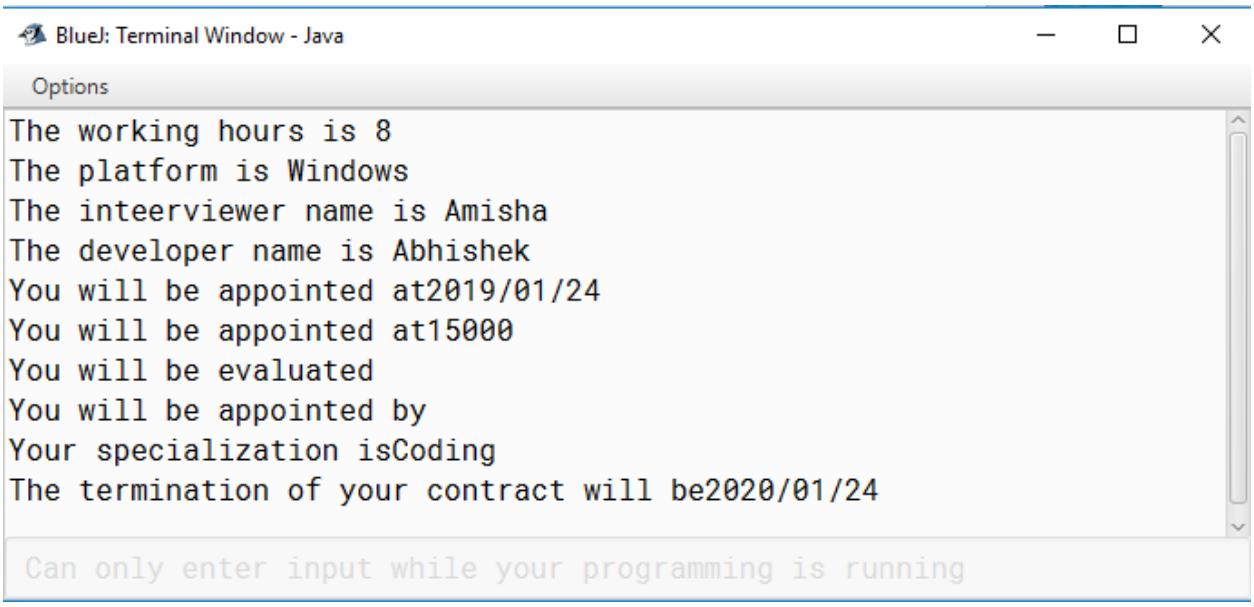

Figure 9: Display Method Output of Junior Developer

Test	4
Task Performed	Display method was called.
Expected Result	All the details must be displayed.
Actual Result	All the details were displayed.
Test Result	Test Successful.

Table 4: Test 4 Table

Error:

While performing the given program, a lot of error was faced which were tackled with the help of research and our module leader's guide. Error can occur if the data type of input does not match the data type the program demands for. For instance, if the user inputs an integer while the program is asking for string then an output is given "The integer cannot be changed into string." The users give those incorrect or irrelevant data either to check the performance of the program or the user is not paying attention to the data type asked by the program. Some of the errors are given as below:

Error 1:

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

}
```

Figure 10:Error While Passing Parameters

```
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;

}
```

Figure 11: Debugging error 1

Error 2:

```
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);
    }
}
```

Figure 12: Syntax Error

```
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);
    }
}
```

Figure 13: Debugging Syntax Error

Error 3:

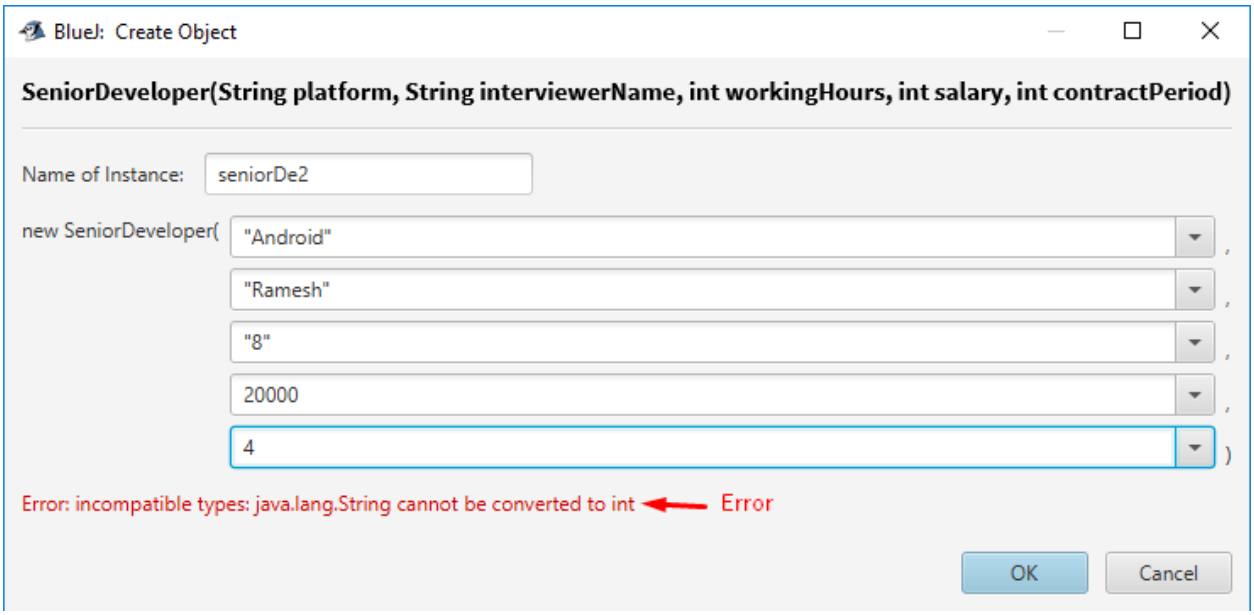

Figure 14: Data Type Error

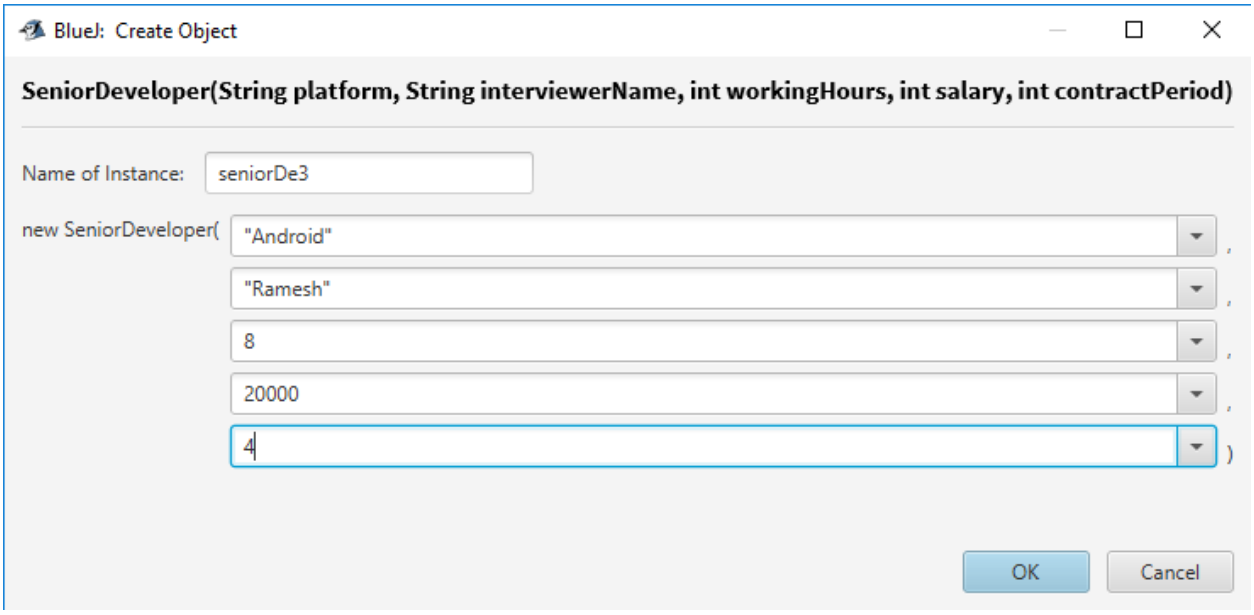

Figure 15: Debugging Data Type Error

Conclusion:

After the success of the project, the concept of inheritance became clear. Many problems were faced while making the program but with research and hard work, the program was complete. The aim of the project was not only to create a java program, but to acknowledge the association of the parent and child class. The main motto of this program is to develop a clear concept of inheritance in Java.

While starting the research, basic concept was only implemented but as soon as the research was done, many more concepts were introduced and programming became easier. Firstly, small programs were made, which was quite easy, but when the concept of inheritance came in, the program faced a lot of errors. The errors could be debugged with the help of researches and lecture slides. The main problem was to fix errors on program. It took almost 2 weeks to achieve the final program but writing the documentation clarified the working mechanism of the program. By understanding the program, documentation becomes easier.

Many websites and journals were accessed to overcome the errors of the program. Discussion with the lecturers became useful for the completion of the project. Also, vital role was played by the lecture slides provided by our teachers. Visiting the websites again and again, made the doubt clear. Hence, with many researches and hard-work, the program was completed along with the documentation.

Appendix:

```
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);
        }
```

```
    }



}
import java.util.Scanner;
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.salary=0;
        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.getDeveloperName()+"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(this.getDeveloperName()+"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);



        }
```

```
    }



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

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



    }
    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);
        }

    }

}
```