Hacettepe University

Department of Computer Engineering

BBM 104 Assignment 2 Report

Umut Bayındır - 2210356147

I.Table of Contents

1.Introduction

In this assignment, we were tasked with making a system to monitor payroll of personnel in a university. The system took two text files as an input, one giving information about the staff and the other giving information about their weekly working hours. After processing the data and calculate their salaries from gathered information, it returns a text file for each employee showing their information, including their salaries. We were expected to rely heavily on objected oriented programming and inheritance.

2.Main

Main.java is the heart of the assignment. It hosts the functions required to accomplish the task. I will briefly show them here.

```
4     //Useful methods to simplify other methods.
5     public static String read(String file) {
6         String output="";
7         try {
8             File text=new File(file);
9             Scanner reader=new Scanner(text);
10            while (reader.hasNextLine()) {
11                String data=reader.nextLine();
12                output+=data+"\n";}
13            reader.close();
14        } catch (FileNotFoundException error1) {
15            System.out.println("File" + file + "not found");
16            error1.printStackTrace();}
17        return output;}
18    public static void write(String file,String input) {
19        try {
20            FileWriter writer=new FileWriter(file);
21            writer.write(input);
22            writer.close();
23        } catch (IOException error2) {
24            System.out.println("An error occured.");
25            error2.printStackTrace();}}
```

Functions used to read the inputs and write the outputs

2.1.main()

I wanted to keep the main function simple. It calls the execute() method.

```
160     public static void main(String[] args) {execute(read(args[0]),read(args[1]));}}
```

2.2.execute()

This method takes the input files “personnel.txt” and “monitoring.text” written in terminal. Entire function relies on a for cycle which processes the info in personnel.txt one by one and makes each entry go through the other processes.

```
146     //Main method
147     public static void execute(String personnel,String monitoring) {
148         String[]personnelLine=personnel.split("\n");
149         for (int i=0;i<personnelLine.length;i++) {
150             String[]personnelWord=personnelLine[i].split("\t");
151             String name=personnelWord[0];
152             String surname=personnelWord[1];
153             String registerNumber=personnelWord[2];
154             String position=personnelWord[3];
155             int yearOfStart=Integer.parseInt(personnelWord[4]);
156             Personnel entry=personnelProcession(name,surname,registerNumber,position,yearOfStart);
157             entry=monitoringProcession(entry,monitoring);
158             output(entry);}}
```

2.3.Personnel and Monitoring Processions

Goal of these two methods are to create an object called entry of superclass Personnel, which holds info of one personnel. First method adds the info from personnel.txt and second method adds the info from monitoring.txt and keeps the data in an array called weeks, which holds all weeks’ entries.

```
103     public static Personnel personnelProcession(String name,String surname,String registerNumber,String position,int yearOfStart) {
104         //This method fills the personnel's information gained from personnel.txt.
105         Personnel entry=new Personnel();
106         entry.setName(name);
107         entry.setSurname(surname);
108         entry.setRegisterNumber(registerNumber);
109         entry.setPosition(position);
110         entry.setYearOfStart(yearOfStart);
111         return entry;}
112     public static Personnel monitoringProcession(Personnel entry,String monitoring) {
113         //This method fills the personnel's information gained from monitoring.txt.
114         int weekOfWork=0;
115         String monitoringLine="";
116         String[]monitoringLines=monitoring.split("\n");
117         for (int i=0;i<monitoringLines.length;i++) {
118             String[]monitoringWords=monitoringLines[i].split("\t");
119             if(entry.getRegisterNumber().equals(monitoringWords[0])) {
120                 weekOfWork+=monitoringWords.length-1;
121                 monitoringLine+=monitoringLines[i];
122                 }}
123         int[]weeks=new int[weekOfWork];
124         String[]monitoringWords=monitoringLine.split("\t");
125         for(int j=1,k=0;j<monitoringWords.length;j++) {
126             weeks[k++]=Integer.parseInt(monitoringWords[j]);
127         }
128         entry.setWeeks(weeks);
129         return entry;}
```

2.4.output()

This method creates a text file for each personnel with the help of write() command and calculates the salary via salaryCalculation() method.

```
130    public static void output(Personnel entry) {
131        //This method fills the output text files using the filled information.
132        String name=entry.getName();
133        String surname=entry.getSurname();
134        String registerNumber=entry.getRegisterNumber();
135        String position=entry.getPosition();
136        int yearOfStart=entry.getYearOfStart();
137        double salary=salaryCalculation(entry);
138        String file=registerNumber+".txt";
139        String input="Name : "+name+"\n";
140        input+="Surname : "+surname+"\n";
141        input+="Registiration Number : "+registerNumber+"\n";
142        input+="Position : "+position+"\n";
143        input+="Year of Start : "+yearOfStart+"\n";
144        input+="Total Salary : "+salary+"0 TL\n";
145        write(file,input);}
```

2.5.salaryCalculation()

I originally intended to use the subclasses for the salary calculation, but I quickly realized they did their calculations instantly and took a lot of values as 0, so I wrote this method instead. It relies a lot on polymorphism, especially up and down casting and gets the values based on the employee's profession. Each profession have their own income systems. Each one is stated as below in the assignment paper given to us:

Academicians have a base salary **(baseSalary)**, special service benefits **(ssBenefits)** and severance pay **(severancePay). A Faculty member** can have an additional course fee **(addCourseFee)** but they can get additional course fees up to 8 hours per week and for each working hour they are paid 20 TL. (if they teach more than 8 hours in a week, they will be paid only for 8 hours). A Faculty member might not teach a course in a term. Research assistants do not have an additional course fee **(addCourseFee)**.

Officers have a base salary **(baseSalary),** special service benefits **(ssBenefits)** and severance pay **(severancePay)**. Officers have also overwork salary **(overWorkSalary).** In order to earn an overwork salary, they can work up to 10 hours a week and for each working hour, they are paid 20 TL. (Even if they work more than 10 hours a week, they are paid up to 10 hours.) For Security personnel, salaries are calculated based on the number of working hours **(hourOfWork)** and severance pay **(severancePay)**. In addition to working hours, they are paid 5 TL for transportation **(transMoney)** and 10 TL food **(foodMoney)** per day. Security

can work a maximum of 9 hours and a minimum of 5 hours a day. They do not work one day of the week. For each working hour, they are paid 10 TL.

The salaries of Full-time Employees are calculated based on the number of working days **(dayOfWork)** and severance pay **(severancePay).** Full-time Employees do not work at weekends. Workers are paid 105 TL and Chiefs are paid 125 TL per day. Also, they have an overwork salary **(overWorkSalary).** Workers can work a maximum of 10 hours a week and are paid 11 TL per hour, while Chiefs can work a maximum of 8 hours a week and are paid 15 TL per hour to gain overwork salary. (If they work more than their maximum hours, they will not be paid additional money for these extra hours).
The salaries of Part-time Employees are calculated based on the number of working hours (**hourOfWork**) and severance pay **(severancePay)**. Part-time Employees can work a minimum of 10 hours and a maximum of 20 hours a week and they are paid 18 TL per hour.

* Academicians, Officers, Workers, and Chiefs work 40 hours per week, excluding the working hours to gain overwork salary and additional course fee.

* If a faculty member worked more than 40 hours a week, it means that he gives additional course that week. But, he/she will be paid up to 8 hours a week.

* If a research assistant works more than 40 hours in a week, they will not be paid additional money.

* One month is equal to four weeks (it means that 1 month is equal to 28 days)

* Base salary is 2600 TL and it is constant for Academician and Officer.

* Special service benefits are %135 of the base salary of a faculty member, %105 of the base salary of research assistant and %65 of the base salary of Officers.

* **Severance pay** changes according to the experience of Personnel, that is, the number of working years in the university. For each year, Personnel gains **20** points multiplied by **0,8.** It is calculated as follows: $(currentyear - yearofstart) * 20 * 0{,}8 = XTL$

```
39    public static double salaryCalculation(Personnel entry) {
40        //Categorizes entries using up and down casting and calculates the salaries using the two methods above.
41        double salary=0;
42        String position=entry.getPosition();
43        entry.setSeverancePay();
44        double severancePay=entry.getSeverancePay();
45        salary+=severancePay;
46        int[]weeks=entry.getWeeks();
47        int weekOfWork=weeks.length;
48        if(position.equals("FACULTY_MEMBER")) {
49            entry=new FacultyMember();
50            int workingHours=((FacultyMember) entry).getWorkingHours();
51            int addCourse=((FacultyMember) entry).getAddCourse();
52            int addCourseLimit=((FacultyMember) entry).getAddCourseLimit();
53            int addCourseFee=overWorkSalaryCalculation(workingHours,addCourse,addCourseLimit,weeks);
54            double ssBenefits=((FacultyMember) entry).getSSBenefits();
55            int baseSalary=((FacultyMember) entry).getBaseSalary();
56            salary+=addCourseFee+baseSalary+baseSalary*ssBenefits;}
```

I decided to show only the intro part of this method. It is very long and basically showcases how each profession is calculated one by one. Method takes an entry as subject and returns the salary. I originally intended to set salary for the entry and returning, but downcasting seemed to ruin the values in the superclass.

3. Personnel and Sub Classes

The superclass. This is where all the details of all personnel is located. It also calculates the severance pay via setSeverancePay() constructor.

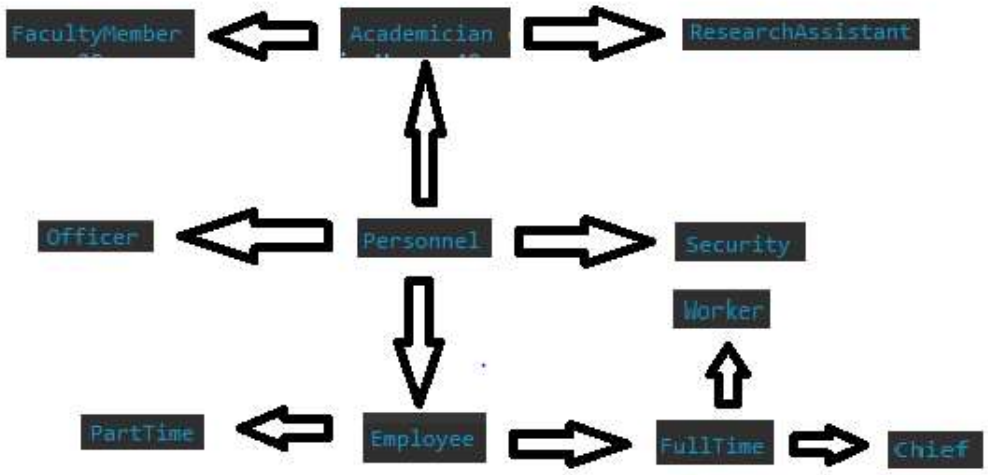
```java
public class Personnel {
    String name;
    String surname;
    String registerNumber;
    String position;
    int yearOfStart;
    int currentYear=2023;
    double severancePay;
    int[]weeks;
    public String getName() {return name;}
    public String getSurname() {return surname;}
    public String getRegisterNumber() {return registerNumber;}
    public String getPosition() {return position;}
    public int getYearOfStart() {return yearOfStart;}
    public int[] getWeeks() {return weeks;}
    public double getSeverancePay() {return severancePay;}
    public void setName(String newName) {this.name=newName;}
    public void setSurname(String newSurname) {this.surname=newSurname;}
    public void setRegisterNumber(String newRegisterNumber) {this.registerNumber=newRegisterNumber;}
    public void setPosition(String newPosition) {this.position=newPosition;}
    public void setYearOfStart(int newYearOfStart) {this.yearOfStart=newYearOfStart;}
    public void setWeeks(int[] newWeeks) {this.weeks=newWeeks;}
    public void setSeverancePay() {this.severancePay=(currentYear-yearOfStart)*20*0.8d;}}
```

3.1.Subclasses

There are four types of subclasses and some of them have their own subclasses and the list goes on.

```java
public class Academician extends Personnel{
    int workingHours=40;
    int baseSalary=2600;
    double ssBenefits;
    public int getWorkingHours() {return workingHours;}
    public int getBaseSalary() {return baseSalary;}}
```

Academician.java as an example of inheritance.



Relation of All Classes with Rach Other(Except Main)

4.Conclusion

This assignment was very useful and taught me a lot about object oriented programming, classes, inheritance and polymorphism, as well as how useful Java programming could be in real life situations.