Computer Program Solutions

QUESTION 2

Ethan Van Rensburg – System Development
Learnership Candidate

18 – 23 June 2024

Contents

a) Requirements:

Interface Design:

The program is described in the document as needing to prompt the workers for their shifts and work hours using Scanner.

```
Please select your assigned shift:
1 - Morning Shift
2 - Day Shift
3 - Night Shift
1
How many hours do you work in a week?
40
```

```
Please select your assigned shift:
1 - Morning Shift
2 - Day Shift
3 - Night Shift
2
How many hours do you work in a week?
40
```

```
Please select your assigned shift:
1 - Morning Shift
2 - Day Shift
3 - Night Shift
3
How many hours do you work in a week?
40
```

Interface Output:

The program will then take any inputs and values from the worker/user and display the suitable results in the output pane.

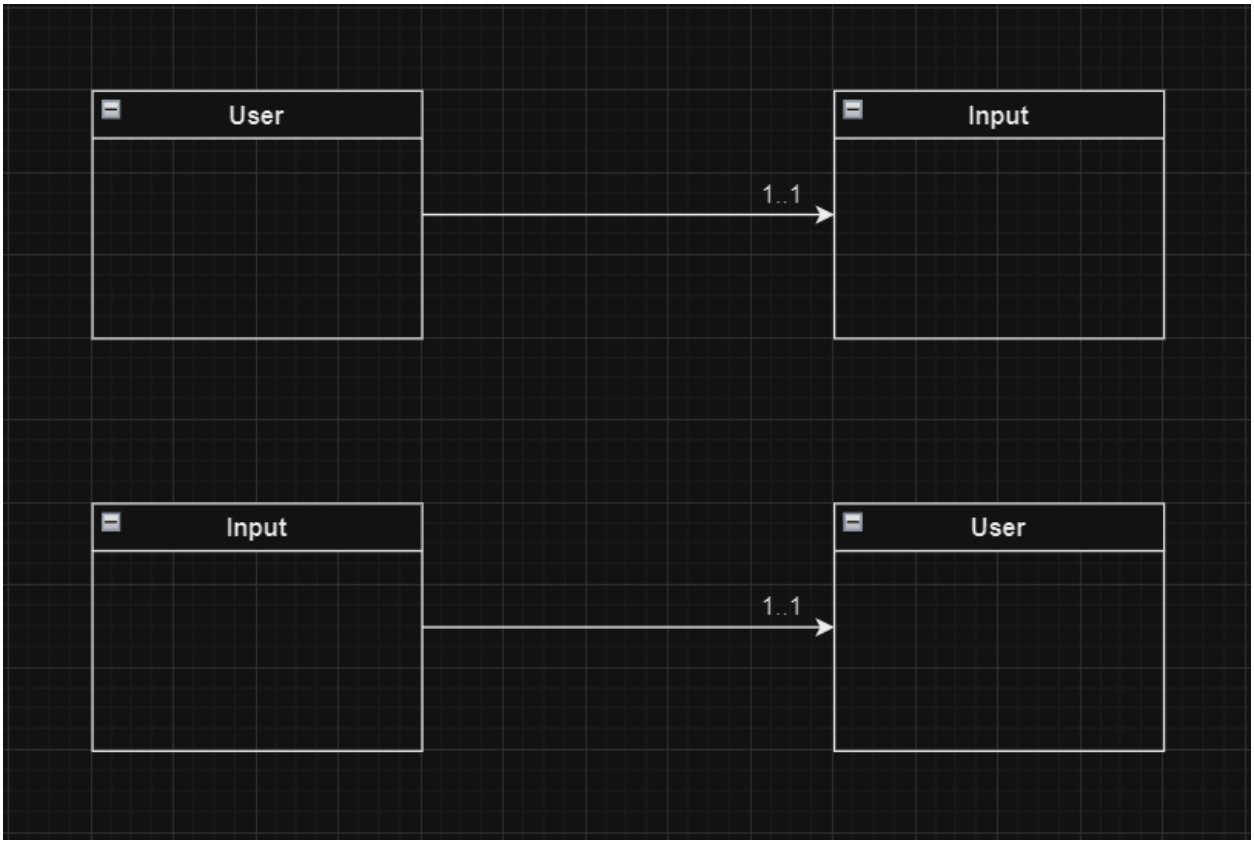
```
Payroll:
----------
Hours Worked: 40
Shift: Morning
Hourly Pay Rate: R50.0
Regular Pay: R2000
Overtime Pay: R0.00
Total of Regular and Overtime: R2000
Retirment Deduction: R0.00
Net Pay: R2000
BUILD SUCCESSFUL (total time: 2 seconds)
```

```
Payroll:
----------
Hours Worked: 40
Shift: Day
Hourly Pay Rate: R70.00
Regular Pay: R2800
Overtime Pay: R0.00
Total of Regular and Overtime: R2800
Retirment Deduction: R0.00
Net Pay: R2800
BUILD SUCCESSFUL (total time: 5 seconds)
```

```
Payroll:
----------
Hours Worked: 40
Shift: Night
Hourly Pay Rate: R90.00
Regular Pay: R3600
Overtime Pay: R0.00
Total of Regular and Overtime: R3600
Retirment Deduction: R0.00
Net Pay: R3600
BUILD SUCCESSFUL (total time: 4 seconds)
```

b) System Architecture:

UML Relationship Diagram



c) Operational Flow:

Flow Chart:

The pdf file for the flowchart will be provided, this link is here just in case.

Click here

Pseudocode:

The following is a sample of pseudocode to showcase the formatting of all the Shift classes and their executable code:

```
Start

1. Declare and initialize variables:
   - hours (integer)
   - hourly1 (double) = 50
   - hourly2 (double) = 70
   - hourly3 (double) = 90


2. Display a menu for shift selection:
   - "Please select your assigned shift:"
     - 1 - Morning Shift
     - 2 - Day Shift
     - 3 - Night Shift
   - Read the user's choice into shifChoice (integer).


3. Switch on shifChoice:
   a. Case 1 (Morning Shift):
      - Prompt the user: "How many hours do you work in a week?"
      - Read the input into hours.
      - Try the following:
        - Create a file named "Hours.txt" (path:
C:\Users\ETHAN.V\Documents\NetBeansProjects\Ethan_UrbanFurn).
        - Append the line "Hours Worked: <hours>" to the file.
        - Calculate weekPay = hours * hourly1.
        - Display payroll details:
          - "Payroll:"
          - "----------"
```

- "Hours Worked: <hours>"
- "Shift: Morning"
- "Hourly Pay Rate: R<hourly1>"
- "Regular Pay: R<weekPay>"
- "Overtime Pay: R0.00"
- "Total of Regular and Overtime: R<weekPay>"
- "Retirement Deduction: R0.00"
- "Net Pay: R<weekPay>"

- End of Case 1.

Case 2 (Day Shift):

1. Declare and initialize variables:
   - hours (integer)
   - hourly2 (double) = 70

2. Prompt the user: "How many hours do you work in a week?"
   Read the input and store it in the 'hours' variable.

3. Try the following:
   a. Create a file named "Hours.txt" (path: C:\Users\ETHAN.V\Documents\NetBeansProjects\Ethan_UrbanFurn).
   b. Append the line "Hours Worked: <hours>" to the file.
   c. Calculate weekPay = hours * hourly2.

4. If 'hours' is equal to 40 and not equal to 0:
   a. Call the method 'retire2' from the 'plan' object with the following arguments:
      - weekPay
      - hours
      - hourly2

b. End of Case 2.

5. Else if 'hours' is greater than 40 and not equal to 0:

a. Calculate overtime hours: overTime = hours - 40.

b. Calculate overtime pay: calcOver = overTime * (70 * 1.5).

c. Calculate total pay (including overtime): weekOver = calcOver + (40 * 70).

d. Call the method 'retireOver2' from the 'plan' object with the following arguments:

- weekOver
- hours
- calcOver
- hourly2

e. End of Case 2.

6. Else:

a. Display "Invalid input."

b. End of Case 2.

Case 3 (Night Shift):

1. Declare and initialize variables:

- hours (integer)
- hourly3 (double) = 90

2. Prompt the user: "How many hours do you work in a week?"

Read the input and store it in the 'hours' variable.

3. Try the following:

a. Create a file named "Hours.txt" (path: C:\Users\ETHAN.V\Documents\NetBeansProjects\Ethan_UrbanFurn).

b. Append the line "Hours Worked: <hours>" to the file.

c. Calculate weekPay = hours * hourly3.

4. If 'hours' is equal to 40 and not equal to 0:

a. Call the method 'retire3' from the 'plan' object with the following arguments:

- weekPay

- hours

- hourly3

b. End of Case 3.

5. Else if 'hours' is greater than 40 and not equal to 0:

a. Calculate overtime hours: overTime = hours - 40.

b. Calculate overtime pay: calcOver = overTime * (90 * 1.5).

c. Calculate total pay (including overtime): weekOver = calcOver + (40 * 90).

d. Call the method 'retireOver3' from the 'plan' object with the following arguments:

- weekOver

- hours

- calcOver

- hourly3

e. End of Case 3.

6. Else:

a. Display "Invalid input."

b. End of Case 3.

End

d) Data Structures:

Application Programming Interfaces:

The program utilized 2 primary APIs in Java for their imports and objects/functions:

```
import java.util.Scanner;
import java.io.*;
```

The Scanner API was useful in calling on objects for input purposes. The IO API helped in allowing the read/write functions for input documentation.

Classes:

The program and the team initially began using multiple classes in Multi-Level Inheritance but then transitioned to a single Main class:

```
public class Ethan_UrbanFurn {

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        Ethan_UrbanFurn plan = new Ethan_UrbanFurn();
        int hours;
        double hourly1 = 50;
        double hourly2 = 70;
        double hourly3 = 90;
```

Objects:

Within the program, many objects and their methods were used alongside each other to produce the needed results of calculations and displays.

Firstly, a Scanner created object was the primary asset for input and data:

```
System.out.println("""
                   Please select your assigned shift:
                   1 - Morning Shift
                   2 - Day Shift
                   3 - Night Shift""");
int shifChoice = scan.nextInt();
```

```
System.out.println("How many hours do you work in a week?");
hours = scan.nextInt();
```

```
if (hours <= 40 && hours != 0) {
    int weekPay = hours * 50;
    System.out.println();
    System.out.println("Payroll:");
    System.out.println("----------");
    System.out.println("Hours Worked: " + hours);
    System.out.println("Shift: Morning");
    System.out.println("Hourly Pay Rate: R" + hourly1);
    System.out.println("Regular Pay: R" + weekPay);
    System.out.println("Overtime Pay: R0.00");
    System.out.println("Total of Regular and Overtime: R" + weekPay);
    System.out.println("Retirment Deduction: R0.00");
    System.out.println("Net Pay: R" + weekPay);
```

Secondly, the different objects were created for the read/write methods and functions, each had a dedicated purpose, such as creating the file; writing the text/inputs into the file, and finding the file for reading and writing.

```
String content = ("Hours Worked: " + hours + "\n");
File fil = new File("C:\\Users\\ETHAN.V\\Documents\\NetBeansProjects\\Ethan_payroll\\Hours.txt");
FileWriter fw = new FileWriter((fil.getAbsoluteFile()), true);
BufferedWriter bw = new BufferedWriter(fw);
bw.write(content);
bw.close();
```

Finally, the last object of completion was using an instance of the main class itself to call its own methods outside its main method:

```
Ethan_UrbanFurn plan = new Ethan_UrbanFurn();
```

```
if (hours <= 40 && hours != 0) {
    int weekPay = hours * 70;
    plan.retire2(weekPay, hours, hourly2);
} else if (hours > 40 && hours != 0) {
    int overTime = hours - 40;
    int calcOver = overTime * (int) (70 * 1.5);
    int weekOver = calcOver + (40 * 70);
    plan.retireOver2(weekOver, hours, calcOver, hourly2);
```