Computer Program Solutions

Question 2

Ethan Van Rensburg – System Development Learnership Candidate

*18 – 23 June 2024*

Contents

[a) Requirements: 1](#_Toc172639900)

[**Interface Design:** 1](#_Toc172639901)

[**Interface Output:** 2](#_Toc172639902)

[b) System Architecture: 3](#_Toc172639903)

[**UML Relationship Diagram** 3](#_Toc172639904)

[c) Operational Flow: 4](#_Toc172639905)

[**Flow Chart:** 4](#_Toc172639906)

[**Pseudocode:** 5](#_Toc172639907)

[d) Data Structures: 9](#_Toc172639908)

[**Application Programming Interfaces:** 9](#_Toc172639909)

[**Classes:** 9](#_Toc172639910)

[**Objects:** 9](#_Toc172639911)

# Requirements:

## **Interface Design:**

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

A white background with black text

Description automatically generated

A white background with black text

Description automatically generated

A white background with black text

Description automatically generated

## **Interface Output:**

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

A screenshot of a computer screen

Description automatically generated

A screen shot of a computer

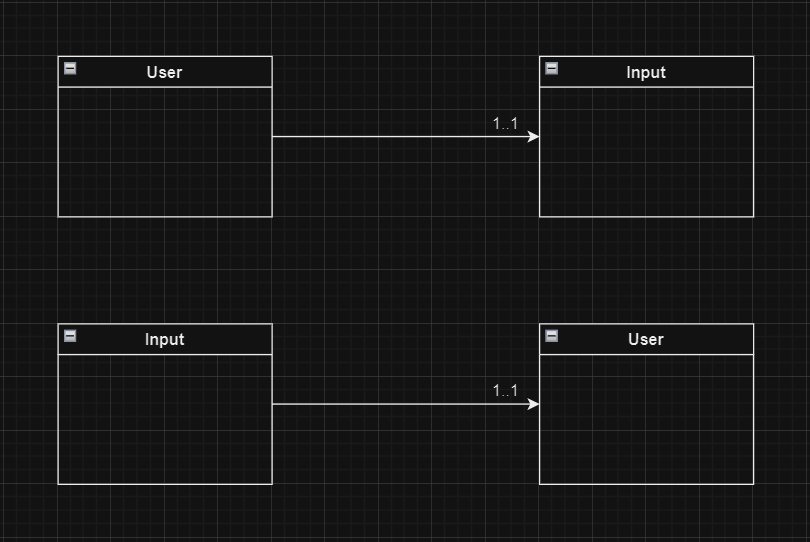
Description automatically generated

A screen shot of a computer

Description automatically generated

# System Architecture:

## **UML Relationship Diagram**



# Operational Flow:

## **Flow Chart:**

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

[Click here](../../Downloads/Flowchart.pdf)

## **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

# Data Structures:

## **Application Programming Interfaces:**

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

A close up of text

Description automatically generated

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:

A screenshot of a computer program

Description automatically generated

## **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:

A screenshot of a computer program

Description automatically generated



A screenshot of a computer program

Description automatically generated

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.

A close-up of a computer code

Description automatically generated

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



