DESIGN DOCUMENTATION

Martin Rugani [Learnership - Curro Private College Cape Town - All Schools]

Contents

A) Desk Checking

Input

1. The Program prompts each worker to input their hours for the week

```
run:
Please enter you hours worked this week
45
```

2. Workers can select their shift

```
Plese select your shift number
 1) First shift
 2) Second Shift
 3) Third Shift
2
```

3. The program will ask the worker if they want to participate in the retirement plan if they work the second or third shift

```
2
Would you like to opt in for a retirement plan? (Yes/no
yes
```

Processing and Output

1. The workers' Hours worked and shift is displayed

```
yes
Hours Worked: 45 Hours
Shift: 2
```

2. The worker's Hourly rate is displayed

```
Hourly pay Rate: R70.0
```

4. Regular pay is calculated based on standard hourly rates and is displayed

```
Regular Pay: R2800.0
```

5. Overtime pay is calculated for hours exceeding 40 and their hourly rate is multiplied by 1.5 then their overtime will be displayed

```
Overtime pay: R525.0
```

6. The total of their regular and overtime pay is displayed

```
Total: R3325.0
```

7. If they opted for a retirement plan their monthly deduction will be calculated by multiplying their gross income by 5% then displayed

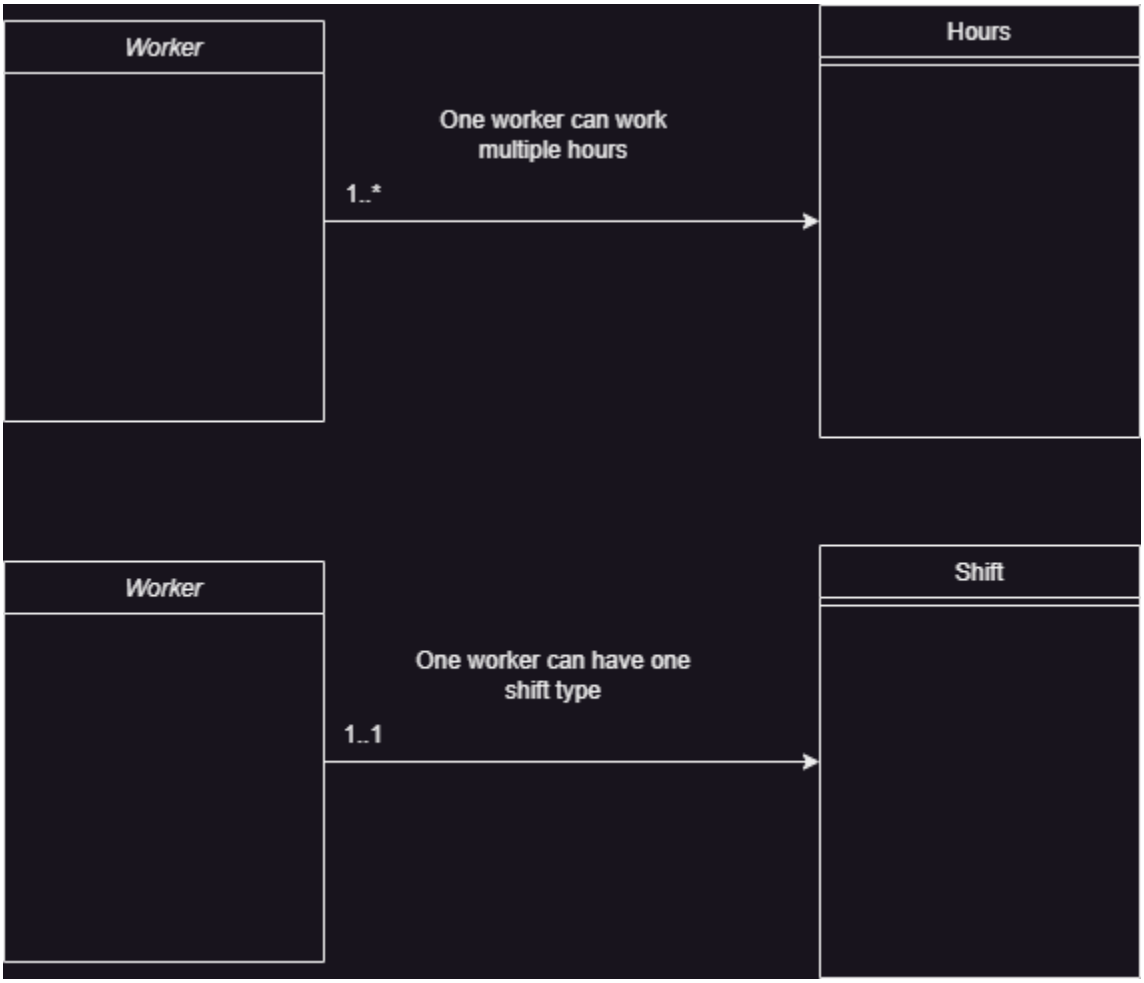
```
Retirement deduction: R166.25
```

8. Their total net pay is displayed

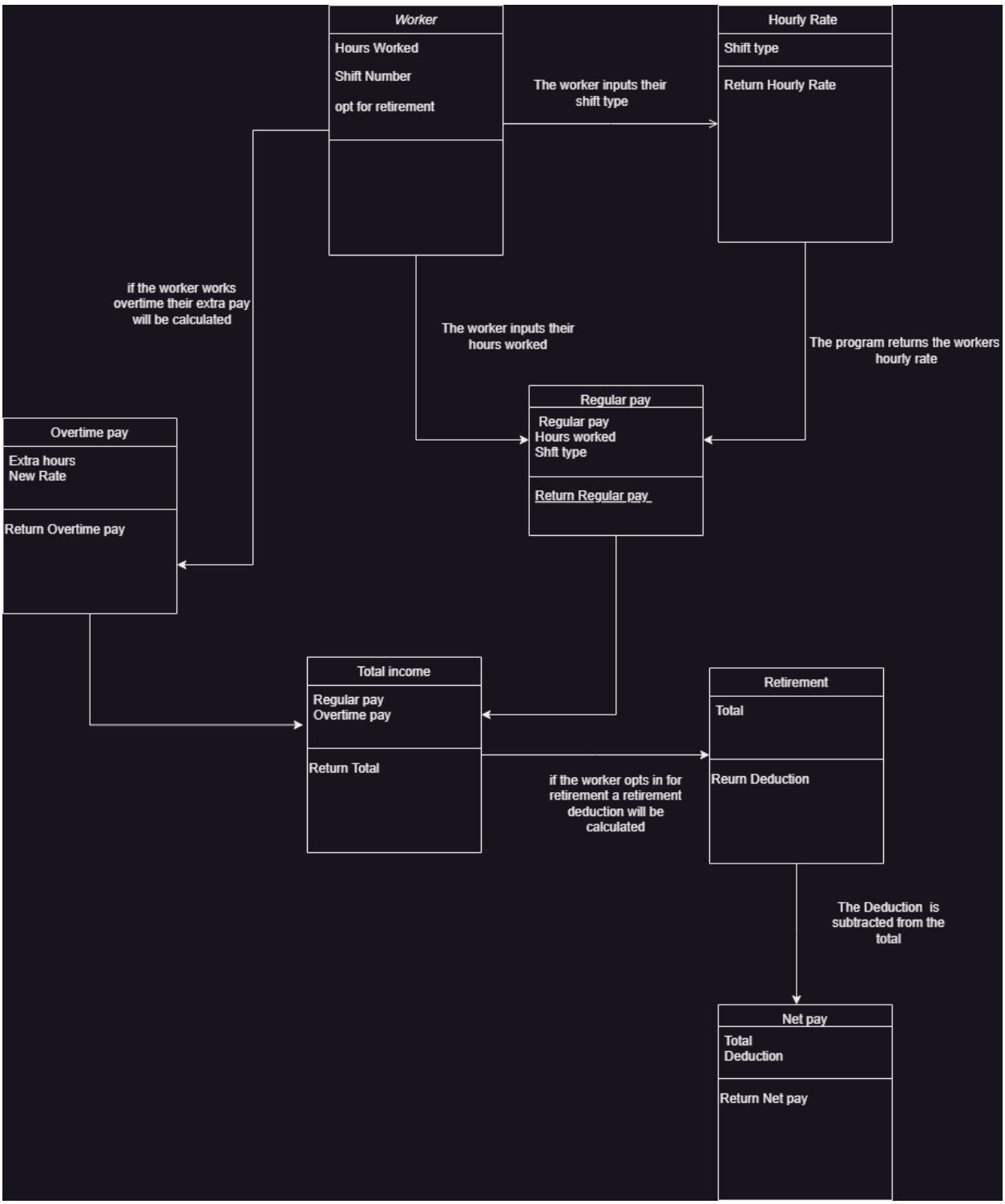
```
Netpay R3158.75
```

B) UML Diagrams

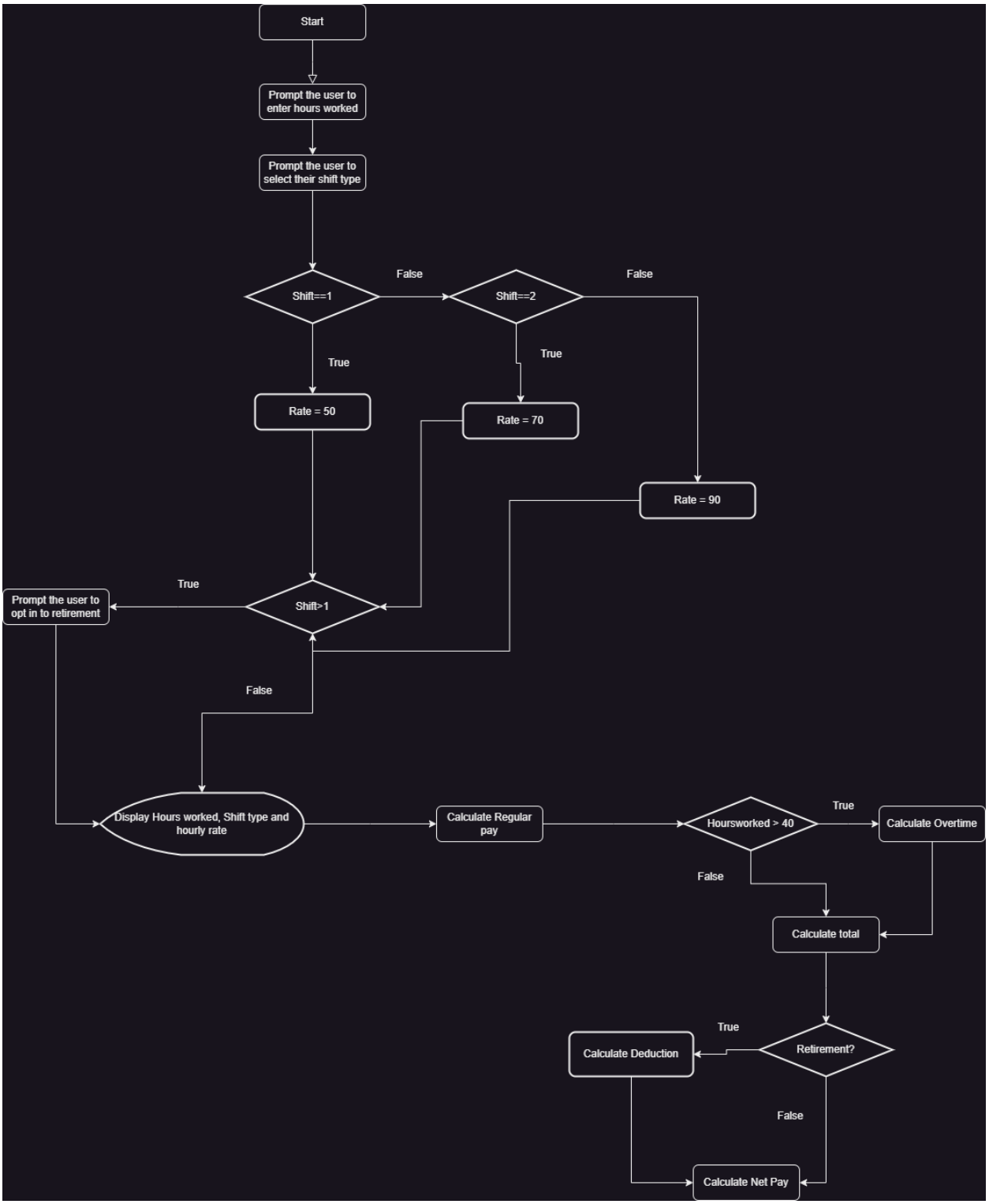
Input UML



Class Diagram



C) Flowchart


Start
Prompt the user to enter hours worked
Prompt the user to select their shift type
Shift==1
False
Shift==2
False
True
True
Rate = 50
Rate = 70
Rate = 90
True
Prompt the user to opt in to retirement
Shift>1
False
Display Hours worked, Shift type and hourly rate
Calculate Regular pay
Hoursworked > 40
True
Calculate Overtime
False
Calculate total
True
Calculate Deduction
Retirement?
False
Calculate Net Pay

Pseudocode

```
//Start
Input
Get Input type(1)New Data or 2)Stored data)
If (Input = 1){
Get Hours worked;
}else{
Read Hours from Textfile;
}
Get Shift type;

Process
If (Shift = first) {
Set rate as 50;
} else if(Shift = second {
Set rate as 70;
}else{
Set rate as 90;
};
If Shift>first {
Get Retirement Choice;
};

Set Regular pay as (Hoursworked * rate);
If (Hoursworked > 40) {
Set regular pay as (40 * rate);
Set overhours as (Hoursworked – 40);
Set overtimepayrate as (rate * 1.5);
```

```
Set Overtimepay as (over hours * overtimepayrate);
Set total as (Regularpay + overtimepay)
} else{
Set total as Regularpay;
};
If (Retirement.equals(“yes”){
Set deduction as (netpay * 0.05) ;
Set netpay as (total – deduction) ;
Else{
Set netpay as total;
}
```

Output

```
Display Hoursworked;
Display shift;
Display payrate;
Display Regularpay;
Display Overtimepay;
Display Total;
Display deduction;
Display net pay;
Write hours worked to output.txt;
Display Hours recorded message;
//Stop
```

D) Data Structures

API

I import the java util and IO API's to use their appropriate objects where necessary

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

Class and Method

The program is implemented in the main class and main method without the use of different classes or methods

```
public class UrbanFurn {

    public static void main(String[] args) {

        //Declaring variables to process employee transactions
        int Hoursworked = 0;
        int Shift;

        double Rate;
        double Regularpay;
        double Overtimepay = 0;
        double Overtimepayrate;
        double total = 0;
        double retirementdeduction = 0;
        double netpay = 0;
        String Retirement = "";
```

- Each variable that is used in the program is declared in the beginning of the main method with appropriate data types

Objects

Objects used

- Scanner
- RandomAccessfile

Implementation

I create a new instance of the Scanner Api and declare it as an object called “scanner”

```
//Creating an object of scanner to collect user input
Scanner scanner = new Scanner(System.in);
```

- The object will use its associated methods and behaviors to collect user input for the following:
    - Hours worked
    - Shift Type
    - Retirement choice

Then I create an instance of RandomAccessfile and declare it as an object called “file”

```
//I create a new instance of random access file called file in read-w
//which means this method will read data then write it
//"rw" means readwrite
RandomAccessFile file = new RandomAccessFile("Output.txt","rw");
```

- The object is declared in read-write mode which means the program will read data from the program and write it into the “Output.txt” document

The file object is used to write and display the hours worked that a user enters each time the program is run

```
//This moves the cursor in the text file to the end of the file
//meaning we add new data after each other
file.seek(file.length());

//Using write bytes to add "hours worked" as single bytes in the
//the label and data are both their own byte hence I use bytes in

file.writeBytes("Hours worked: " + Hoursworked + "\n" );
```