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

Design Documentation

Contents

[A) Desk Checking 2](#_Toc172617039)

[B) UML Diagrams 4](#_Toc172617040)

[Input UML 4](#_Toc172617041)

[Class Diagram 5](#_Toc172617042)

[C) Flowchart 6](#_Toc172617043)

[Pseudocode 7](#_Toc172617044)

[D) Data Structures 9](#_Toc172617045)

[API 9](#_Toc172617046)

[Class and Method 9](#_Toc172617047)

[Objects 10](#_Toc172617048)

# Desk Checking

**Input**

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

A black and white text

Description automatically generated

1. Workers can select their shift

A white background with black text

Description automatically generated

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

A black and white text

Description automatically generated

**Processing and Output**

1. The workers' Hours worked and shift is displayed

A black and white text

Description automatically generated

1. The worker’s Hourly rate is displayed



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



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



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



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

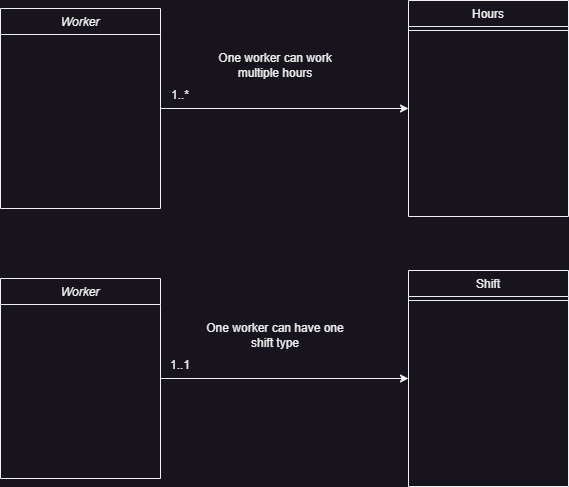


1. Their total net pay is displayed



# UML Diagrams

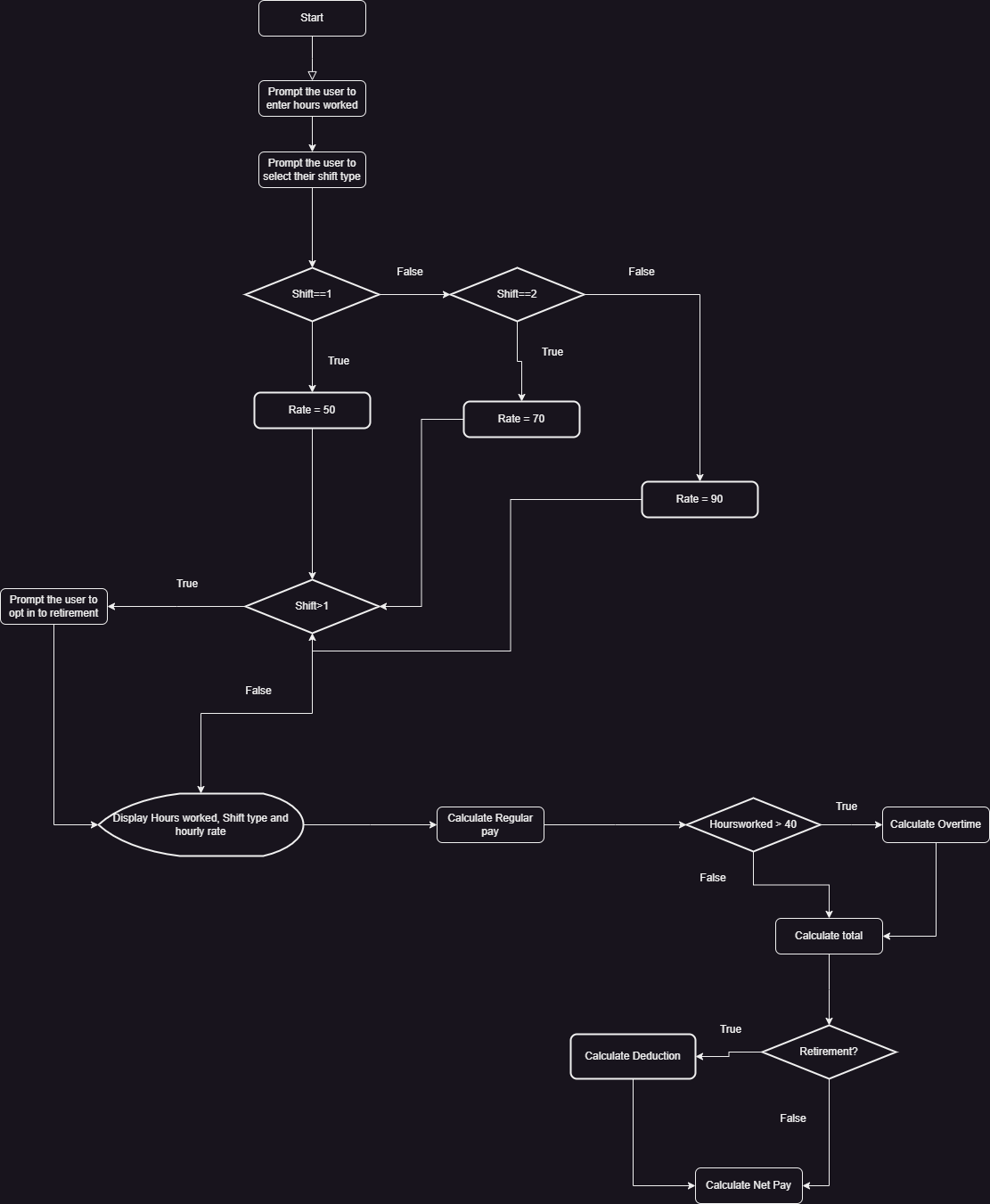
## Input UML



## Class Diagram



# Flowchart



## 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

# Data Structures

## API

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

A close up of words

Description automatically generated

## Class and Method

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

A screenshot of a computer code

Description automatically generated

* 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”

A close up of words

Description automatically generated

* 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”

A close up of text

Description automatically generated

* 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

A white text box with black text

Description automatically generated with medium confidence