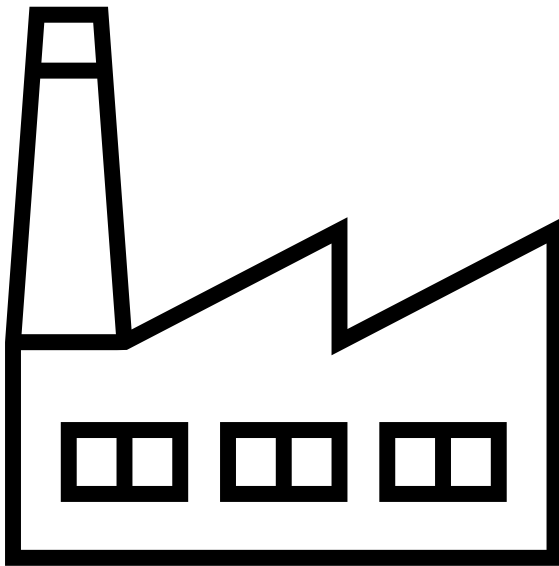


# Summative Assessment 4

18-23 July 2024

Jamie Myburgh



Documentation

Question 6

# Contents

a) Purpose Description .....	3
Data Flow .....	3
b) Code Documentation .....	4
c) READ-ME file .....	11
d) Change Log .....	13

## a) Purpose Description

The program that was created is an employee payroll calculator for the company UrbanFurn, following a resurfacing error in the overtime pay department. The calculator takes user input from the employee, which is the hours that they have worked and the shift rate they were working for, and uses this to compile their net income or payroll for the week.

There are three shifts in which a user can work. The first shift pays R50 an hour, the second shift pays R70 an hour, and the third shift pays R90 an hour. The user can input any number of hours.

The standard hours worked in a week is 40 hours. Any hour that extends this is considered overtime. The overtime pay is calculated by taking the hours extending 40 and multiplying that by the rate, produced by one and one-half. This only occurs for the hours exceeding 40, not the total hours.

Should a user be working for the second or third shift, they will be offered an optional retirement plan. Should they accept the offer, 5% of their total pay will be deducted from their gross income, including their overtime pay. Should they reject the offer, the deduction will not go through.

Once the necessary information has been processed, a breakdown of the user's payroll will be displayed. This includes their hours, their shift number, the shift rate, the regular pay, the overtime payroll if applicable, the total payroll, the retirement fund if applicable, and the net payment.

### Data Flow

The user inputs their hours. The user will then enter their shift. If the user's hours exceed 40, the regular payroll is still calculated. The overtime payroll is also calculated, by multiplying the hours exceeding 40 by the shift rate produced by one and one-half.

Their regular income, overtime income, and net income will then be displayed/

If the second or third shift is selected, the user will be asked if they'd like to apply for the retirement fund. If the user accepts the offer, 5% of their total payroll, which includes the overtime value, is calculated. This value is then subtracted from their total payroll to acquire their net income.

If the user does not accept the retirement plan, their total payroll and net income will be displayed.

## b) Code Documentation

Below is the first part of the main class. Here, the necessary utilities and extensions were imported. The variables that were to be used throughout all three classes were also initialized inside the class, outside of the main method.

```
package jamie_payroll_solution2;

import java.util.Scanner;
import java.io.RandomAccessFile;
import java.io.IOException;

public class main {

    //Declaring the variables that will be used throughout the classes
    public static String input; //Variable for user input
    public static int hours; //Variable for the hours
    public static int shift; //Variable for the shift rate.
```

Instances of the relevant classes were instantiated within this main method, to call upon their methods and functions.

```
public static void main(String[] args) {

    /*
    Creating an instance of scanner and the other methods to call
    on their methods and functions.
    */

    Scanner sc = new Scanner(System.in);
    calculations pay = new calculations(); //'pay' is an instance of the calculations class
    retirement ret = new retirement(); //'ret' is an instance of the retirement class
```

A welcome screen first displayed when the program is run. The program first prompts the user for their worked hours, and captures their response using the Scanner function, and places their value in the “hours” variable. It then prompts the user to enter their shift number, can stores their response in the “shiftInt” variable.

```
//Text for the welcome screen and the printed out prompts
System.out.println("Welcome to the UrbanFurn Payroll Calculator!"); //Welcoming screen
System.out.println("-----");
System.out.println("");
System.out.println("Please enter the hours you've worked for the week:"); //Prompts user to enter hours
hours = sc.nextInt(); //Hours is collected from user input
System.out.println("");
System.out.println("Please enter your shift number: (1, 2, or 3)"); //Prompts user to enter shift category
System.out.println("Shift 1: R50 per hour"); //First shift is at R50 an hour
System.out.println("Shift 2: R70 per hour"); //Second shift is at R70 an hour
System.out.println("Shift 3: R90 per hour"); //Third shift is at R90 an hour
int shiftInt = sc.nextInt(); //Shift category collected from user input
System.out.println("");
```

If “shiftInt” is ‘1’, then the “shift” variable is stored with the value ‘50’, if “shiftInt” is ‘2’, then the “shift” variable is stored with the value ‘70’, if “shiftInt” is ‘3’, then the “shift” variable is stored with the value ‘90’.

The calculations class was created next, extending the main class, and the necessary variables were instantiated.

```
package jamie_payroll_solution2;

public class calculations extends main {

    protected double total; //Variable for a regular payroll
    protected double payroll; //Variable for a payroll depending on the condition
    protected double overtime; //Variable for the overtime payment
```

The method to perform the calculation is opened and an if-statement is opened. If the hours exceed 40, the regular payment and overtime payment are calculated and added together and stored in the “payroll” variable.

```
public void perfCalc() {

    //IF HOURS ARE GREATER THAN 40
    if (hours > 40) { //If hours are greater than 40

        total = 40 * shift; //Calculate the regular payment and store it in 'total'
        overtime = 1.5 * (hours - 40) * shift; //Calculate the overtime value
        payroll = total + overtime; //Calculate the sum and store it in the payroll
```

If the hours are equal to or less than 40, the shift rate and hours are multiplied together, and the result is stored in the payroll value. The method closes.

```
//IF HOURS ARE LESS THAN 40
} else {

    payroll = hours * shift;

}

}
```

A method to display the information is created, with an if-else-statement. If the hours are over 40, it first performs the calculations and then the regular payment is displayed alongside the overtime hours. The overtime pay is also displayed with the payroll.

```
public void dispCalc() {

    //IF HOURS ARE GREATER THAN 40
    if (hours > 40) { //If hours are greater than 40

        perfCalc(); //Perform the calculations method

        System.out.println("Regular pay: R" + total); //Display the regular pay
        System.out.println("Overtime: " + (hours - 40) + " hours."); //Display the overtime hours
        System.out.println("Overtime pay: R" + overtime); //Display the overtime pay
        System.out.println("Total: R" + payroll); //Display the total payroll

    }

}
```

If the hours are equal to or less than 40, it first performs the calculation and displays the payroll.

```
//IF HOURS ARE LESS THAN 40
} else {

    perfCalc(); //Performs the calculations method

    System.out.println("Regular Pay: R" + payroll); //Prints the payroll
}

}

}
```

The retirement class was then constructed, extending the calculations class. The necessary variables were also instantiated.

```
package jamie_payroll_solution2;

public class retirement extends calculations {

    protected double retirement; //Variable for retirement value
    protected double net; //Variable for net income
```

A method to perform the retirement calculations is then constructed, alongside an if-else statement. If the user's input from the main class is equal to "yes", it performs the calculations in the "calculations" class. It then calculates 5% of the payroll and stores it in the "retirement" variable. The method then subtracts the retirement from the payroll and stores it in the "net" variables. The deducted retirement amount and net payroll are displayed.

```
public void perfRetire() {

    if (input.equalsIgnoreCase("yes")) { //If the value is equal to "yes"
        perfCalc(); //Perform the calculations method from the previous class
        retirement = payroll * 0.05; //The retirement is set to 5% of the payroll
        net = payroll - retirement; //The net payment is the retirement subtracted from the payroll
        System.out.println("Retirement deduction: R" + retirement); //Display the retirement amount
        System.out.println("Net: R" + net); //Display the net income
```

If the user's input is not equal to "yes", it still performs the calculations, except it stores the "payroll" variable's value in the "net" variable. It then prints the net-value. The method closes

```
    } else {
        perfCalc();
        net = payroll;
        System.out.println("Net: R" + net);
    }

}
```

This is a method to calculate the net-payment for a user who selected the first shift/is not eligible for a retirement fund. It calls the method from the “calculations” class and sets the value in the payroll-variable into the net-variable. It displays the net-value and the method closes.

```
public void regularNet() {  
    perfCalc();  
    net = payroll;  
    System.out.println("Net: R" + net);  
}  
  
}
```

Once all the classes were finished, the “main” class was adjusted accordingly. If statements were created with the necessary content. If the first shift is entered, the shift rate is set to 50. The hours, shift, and rate are displayed, and the calculations are performed by calling the method. The method to calculate the net pay is also called.

```
//IN SHIFTS INPUT IS EQUAL TO 1  
if (shiftInt == 1) { //If 1st category  
    shift = 50; //Shift rate is at R50 per hour  
  
    System.out.println("Hours: " + hours); //Prints the hours  
    System.out.println("Shift: 1st"); //Prints '1st' shift  
    System.out.println("Rate: R50 per hour"); //Prints the rate  
    System.out.println("");  
  
    pay.dispCalc(); //Performs calculations applicable  
    ret.regularNet(); //Prints the regular net pay
```



If the second shift is entered, the rate is set to 70. A prompt is displayed, asking if the user would like to apply for a retirement fund. The input is collected and stored in the “input” variable. The hours, shift, and rate are printed out, and the methods to perform the calculations and retirement are called.

```
//IF SHIFTS INPUT IS EQUAL TO 2
} else if (shiftInt == 2) { //If 2nd category
    shift = 70; //Shift rate is at R70 an hour

    System.out.println("Would you like to apply for a retirement fund? (yes/no)"); //Prompt for retirement
    input = sc.next(); //Collects user input and stores it in variable 'input'

    System.out.println("");
    System.out.println("Hours: " + hours); //Prints the hours
    System.out.println("Shift: 2nd"); //Prints '2nd' shift
    System.out.println("Rate: R70 per hour"); //Prints the rate
    System.out.println("");

    pay.dispCalc(); //Performs calculations applicable
    ret.perfRetire(); //Performs retirement calculations
}
```

If the third shift is entered, the rate is set to 90. A prompt is displayed, asking if the user would like to apply for a retirement fund. The input is collected and stored in the “input” variable. The hours, shift, and rate are printed out, and the methods to perform the calculations and retirement are called.

```
//IF SHIFTS INPUT IS EQUAL TO 3
} else if (shiftInt == 3) { //If 3rd category
    shift = 90;

    System.out.println("Would you like to apply for a retirement fund? (yes/no)"); //Prompt for retirement
    input = sc.next(); //Collects user input and store it in variable 'input'

    System.out.println("");
    System.out.println("Hours: " + hours); //Prints the hours
    System.out.println("Shift: 3rd"); //Prints '3rd' shift
    System.out.println("Rate: R90 per hour"); //Prints the rate
    System.out.println("");

    pay.dispCalc(); //Performs calculations applicable
    ret.perfRetire(); //Performs retirement calculations
}
```

This is a function that stores the hours and shifts entered at each runtime. The program searches through the files for a text-file titled “Hours and Shifts” and inputs the data here. If the file does not exist, it is created. The main method is closed, and the class is closed accordingly.

```
try {
    RandomAccessFile file = new RandomAccessFile("Hours and Shifts.txt", "rw"); //Creates a text file called "Hours and Shifts.txt"
    file.seek(file.length());
    file.writeBytes("Hours: " + String.valueOf(hours) + "\n"); //Prints inputted hours
    file.writeBytes("Shift: R" + String.valueOf(shift) + " per hour" + "\n"); //Prints inputted shift
    file.writeBytes(" "); //Prints a space
    file.close();
} catch (IOException exception) {
    exception.printStackTrace();
}

}

}
```

## c) READ-ME file

A READ-ME.txt file has been created and is stored in the files of the program. The following displays the content within said READ-ME.txt file.

The program that was created is an employee payroll calculator for the company UrbanFurn, following a resurfacing error in the overtime pay department. The calculator takes user input from the employee, which is the hours that they have worked and the shift rate they were working for, and uses this to compile their net income or payroll for the week.

There are three shifts in which a user can work. The first shift pays R50 an hour, the second shift pays R70 an hour, and the third shift pays R90 an hour. The user can input any number of hours.

The standard hours worked in a week is 40 hours. Any hour that extends this is considered overtime. The overtime pay is calculated by taking the hours extending 40 and multiplying that by the rate, produced by one and one-half. This only occurs for the hours exceeding 40, not the total hours.

Should a user be working for the second or third shift, they will be offered an optional retirement plan. Should they accept the offer, 5% of their total pay will be deducted from their gross income, including their overtime pay. Should they reject the offer, the deduction will not go through.

Once the necessary information has been processed, a breakdown of the user's payroll will be displayed. This includes their hours, their shift number, the shift rate, the regular pay, the overtime payroll if applicable, the total payroll, the retirement fund if applicable, and the net payment.

---

To run the program effectively, the following specifications are recommended, but not necessarily required:

**PROCESSOR (CPU):**

Intel: Core i5

**MEMORY (RAM):**

Recommended: 8GB

**STORAGE:**

Type: Solid State Drive for faster performance

Recommended Capacity: 256GB

**GRAPHICS CARD:**

Integrated CPU

**DISPLAY:**

Size: 13-14 inches (sufficient for easy usage)

Resolution: Full HD (1920 x 1080)

**BATTERY LIFE:**

Span: approximately 8 hours of battery life for all-day usage

**OPERATING SYSTEM:**

Windows

macOS

Linux

**CONNECTIVITY:**

Sufficient ports

Wi-Fi 6

Bluetooth 5.0

---

This program was created by Jamie Myburgh.

## d) Change Log

The if-statements for the shift-input had an additional “else” statement for invalid input. This was done so that if any other value was inputted into the system, there wouldn’t be a type-mismatch error or “build success” situation.

```
} else {  
    System.out.println("Invalid input.");  
}
```

The else-if statements referring to hours less than or equal to 40 were changed to specifically mention that the hours lie between 0 and 40. This was done so that negative values were not accepted.








```
//IF HOURS ARE LESS THAN 40  
} else if (hours <= 40 && hours > 0) {  
  
    payroll = hours * shift;  
  
}  
  
}  
  
//IF HOURS ARE LESS THAN 40  
} else if (hours <= 40 && hours > 0) {  
  
    perfCalc(); //Performs the calculations method  
  
    System.out.println("Regular Pay: R" + payroll); //Prints the payroll  
}  
  
}  
  
}
```

The conditions within the retirement class regarding the input were adjusted to accept only “yes” or “no”, and to otherwise display text to alert the user that the input is invalid.

```
} else if (input.equalsIgnoreCase("no")) {  
    perfCalc();  
    net = payroll;  
    System.out.println("Net: R" + net);  
  
} else {  
    System.out.println("Invalid input.");  
}  
}
```

## GitHub Changes

I originally uploaded the last document as a Word document; however, it ran through a series of changes and will be re-uploaded as a pdf file.

 MYB01CPT Add files via upload	89ff1af · now	 5 Commits
 JamieQ1.pdf	Add files via upload	14 minutes ago
 JamieQ2.pdf	Add files via upload	1 minute ago
 JamieQ4.pdf	Add files via upload	1 minute ago
 JamieQ5.pdf	Add files via upload	now
 JamieQ6.docx	Add files via upload	14 minutes ago

The program's file also underwent some changes, seeing as the title of the "total" variable in the "calculations" class was changed to "regular" instead, to avoid confusion or misinterpretation of the variable's purpose. In every place this variable was used, the variable name changed. This includes the classes within the test package.

```
package jamie_payroll_solution2;
```

```
public class calculations extends main {
```

```
    protected double regular; //Variable for a regular payroll
```

```
    protected double payroll; //Variable for a payroll depending on the condition
```

```
    protected double overtime; //Variable for the overtime payment
```

```
@Test
```

```
public void testNotPayroll() {
```

```
    calculations calc = new calculations();
```

```
    main.hours = 50;
```

```
    main.shift = 70;
```

```
    calc.perfCalc();
```

```
    assertEquals(3500, calc.regular, 0.01);
```

```
}
```

```
@Test
```

```
public void testTotal() {
```

```
    calculations calc = new calculations();
```

```
    main.hours = 50;
```

```
    main.shift = 70;
```

```
    calc.perfCalc();
```

```
    assertEquals(2800, calc.regular, 0.01);
```

```
}
```



**MYB01CPT** Add files via upload

18a2dda · now

**6 Commits**



JamieQ1.pdf

Add files via upload

19 minutes ago



JamieQ2.pdf

Add files via upload

6 minutes ago



JamieQ4.pdf

Add files via upload

6 minutes ago



JamieQ5.pdf

Add files via upload

5 minutes ago



JamieQ6.docx

Add files via upload

19 minutes ago



Jamie\_Payroll.zip

Add files via upload

now



**MYB01CPT** Add files via upload

55dfa39 · now

**7 Commits**



JamieQ1.pdf

Add files via upload

23 minutes ago



JamieQ2.pdf

Add files via upload

10 minutes ago



JamieQ4.pdf

Add files via upload

9 minutes ago



JamieQ5.pdf

Add files via upload

9 minutes ago



JamieQ6.docx

Add files via upload

23 minutes ago



Jamie\_Payroll.zip

Add files via upload

4 minutes ago



Jamie\_Payroll\_Version2.zip

Add files via upload

now



READ-ME.txt

Add files via upload

now



