**CSC1021 – Project 1**

**Testing and Output Documentation**

**By**

**Joseph Whitten**

**140197916**

Testing the program

In order to make sure the project works to the specifications I have been given I have carried out extensive testing on the code I have written. I have tested the program by working my way up the call hierarchy of the program, starting with the methods in TaxCalculator.java that have no at the bottom of the hierarchy and working my way up to the main method in BogOff.java. This allowed me to systematically check my code by compiling the code and testing outputs and calculations and easily find any changes or errors that could occur.

In general I will be using the following format for testing my code:

I will test each method of my program by calling the method and inputting data as the parameter to the method.

I will then analyse the output and check to see if it is as expected, and then use that to determine whether or not there are any logic or compilation errors in my code. By doing this I will not have to go through testing the actual contents of the method and variables that control its behaviour, as I will be able to see whether or not it is working immediately

# TaxCalculator.java

This file contains the taxPayable and remainingIncome methods. The taxPayable method is responsible for calculating the tax on an inputted income, rounding it to the nearest integer and returning this value. The remainingIncome method takes the tax calculated in the taxPayable method and then subtracts this value to calculate the remaining income after tax and then returns this value.

taxPayable(double inputIncome)

The taxPayable method is very important to the program as it deals with the calculating the tax on an income, and so I will test it thoroughly to make sure that it performs as expected and returns the correct values.

I tested the method using this parameter to input various types of data into the method and then checked the outputs to see if they were as expected. The data I will input is as follows:

{0, 627, 343, 213, 171, 465, 526, 17}

|  |  |  |  |
| --- | --- | --- | --- |
| Input | Expected Output | Output | Conclusion |
| 0 | 0 | 0 | The program correctly calculated the tax on this input |
| 627 | 387 | 387 | The program correctly calculated and rounded the tax on this input |
| 343 | 81 | 81 | The program correctly calculated and rounded the tax on this input |
| 213 | 20 | 20 | The program correctly calculated and rounded the tax on this input |
| 171 | 9 | 9 | The program correctly calculated and rounded the tax on this input |
| 465 | 193 | 193 | The program correctly calculated and rounded the tax on this input |
| 526 | 266 | 266 | The program correctly calculated and rounded the tax on this input |
| 17 | 0 | **0** | The program correctly calculated and rounded the tax on this input |

The method has calculated the tax on all of the inputs and rounded them to the nearest whole number correctly. It has correctly applied the correct tax to the values in the tax band and recognized that values less than 101 should not be taxed.

|  |  |  |
| --- | --- | --- |
| Value | | Proof |
| 526 |  | Tax = (50\*0.1) + (50\*0.2) + (100\*0.4) + (100\*0.6) + ((526-400)\*1.2) = 266.2  266.2 🡪 Rounded 🡪 266 |
| 171 |  | Tax = (50\*0.1) + ((171 – 150)\*0.2) = 9.2  9.2 🡪 Rounded 🡪 9 |
| 17 |  | Tax = (17\*0) = 0 |
| 343 |  | Tax = (50\*0.1) + (50\*0.2) + (100\*0.4) + ((343-300)\*0.6) = 80.8  80.8 🡪 Rounded 🡪 81 |

Due to the structure of the method if the method returns the correct value of tax for the highest tax band (greater than 401) then it will correctly return all tax values for all tax bands as the method works its way from the highest tax band down to the lowest.

I have decided that in order to check the method properly I should input some values that make use of the double parameter (decimals, not just double representations of integers) and some values that are larger then average to test how efficiently the method handles them.

{175.50, 255.51, 355.49, 20000}

|  |  |  |  |
| --- | --- | --- | --- |
| Input | Expected output | Output | Conclusion |
| 175.50 | 10 | 10 | The program correctly calculated and rounded the tax on this input |
| 255.51 | 37 | 37 | The program correctly calculated and rounded the tax on this input |
| 355.49 | 88 | 88 | The program correctly calculated and rounded the tax on this input |
| 20000 | 20000 | 20000 | The program has correctly noticed that the tax cannot exceed the income |

The method has returned the correct tax values and has taxed the decimals appropriately and rounded the tax correctly to the nearest integer. The method has also dealt with the 20000-value input correctly as well, recognizing that the tax cannot exceed the income. I have made the method take in doubles as a parameter as incomes can have up to 2 decimal places (£0.00).

|  |  |
| --- | --- |
| Value | Proof |
| 255.51 | (50\*0.1) + (50\*0.2) + ((255.51-200)\*0.4) = 37.204  37.204 🡪 Rounded 🡪 37 |
| 20000 | (50\*0.1) + (50\*0.2) + (100\*0.4) + (100\*0.6) + ((20000-400)\*1.2) = 23635  23635 🡪 (Tax <= Income) logic 🡪 20000 |

The method has handled the values with decimals as it would any other value with no problems. It has also handled the large input value efficiently as it treats is as it would any other value then applies the (Tax <= Income) logic after it has done this calculation.

To complete testing of this method I am going to test types that do not match the parameter type of the method and a value that would not happen in using this program and therefore should not be calculated (a negative income).

The program encountered a compilation error when the parameter was presented with a value of an incorrect type showing that it will only accept values of type double. For the negative income value the method calculated that the tax was 0, but since 0 is greater the negative value the (Tax <= Income) logic was applied to return the negative value that was inputted. This is incorrect since the value of the income cannot be negative but the method still calculated it anyway since I have placed the check to see if the income is negative and the logic that dictates what happens when the income is negative higher up the call hierarchy (Negative incomes print an error message and do not get calculated).

remainingIncome(double inputIncome)

The remainingIncome method simply takes the inputted income and returns what will remain of that income after it ha been taxed. The method does this by taking the income as a parameter then calling the taxPayable method to work out and subtract the tax from the income. So I expect that it will respond similarly to testing as the taxPayable method the main difference being that it will return a double and not an integer.

{627, 343, 213, 17}

|  |  |  |  |
| --- | --- | --- | --- |
| Input | Expected Output | Output | Conclusion |
| 627 | 240 | 240 | The method has correctly subtracted the tax from the income |
| 343 | 262 | 262 | The method has correctly subtracted the tax from the income |
| 213 | 193 | 193 | The method has correctly subtracted the tax from the income |
| 17 | 17 | 17 | The method has correctly subtracted the tax from the income |

The method has correctly subtracted the tax and returned the correct income remaining for all values that have been entered. I have determined that this method will also show similar results for the values shown in the taxPayable. This is due to the fact that it uses the taxPayable method for the calculation, and so the method will correctly work for all valid data it is presented with. I can claim this as both method use the double type as their parameter and any value that will work for the taxPayable method will also work for the incomeRemaining method and vice versa, the same goes for values that will not work. This is also easy to show as the method only does a simple subtraction and cannot return a negative income as the parameter used for the taxPayable method call is the same value that is used as the parameter for the income in the method, therefore the remaining income method can not return a negative value as required for the specification (no negative net incomes).

I now know that the TaxCalculator.java file is working correctly as both methods are outputting the expected values respective to the inputs they have been given, I have also been through the code that dictates the behaviour of the methods and have checked to see that the mathematics of my code are correct and that this isn’t just a coincidence and have determined that the methods will work with all given double values. I have also been through the code to make sure it is efficient and easy to understand, and believe I have made it reliable and efficient, it would also be easily changes if the specification were to suddenly change the values of the tax bands with minimal code needing to be rewritten.

# TaxChart.java

This file is responsible for the creation of the graph and table that will be outputted from the program. This includes the drawing of the bars that will make up the axis and data in the graph and the formatting of the table, this is also where the check occurs to see if the income entered is negative or not occurs.

Draw()

This method draws the graph using a series of bars to represent the incomes and their relative taxes. As required by the specification the height of the bars had to be proportional to their magnitude be it income or tax, the x-axis of the graph also had to be proportional to the magnitude of the income and the top of the tax bars had to be in the same position as the top of the income bars to give a visual representation of how much of each total income was being taxed and how much would remain.

The draw method takes no parameters and does not return a value, it instead works with variables and uses the taxPayable method to draw and scale the graph. So I cannot just test the parameters to see if the method is working correctly.

To test this method I will be looking at the outputted graph that it produces and making sure that the graph has the expected characteristics. The characteristics I will be looking for will be the quantity of bars drawn and the placement of both the income and tax bars, these are both determined by the inputSet array, which is required by the object as the parameter for the constructor.

{0, 627, 343, 213, 171, 465, 526, 17}, {295, 173, 481}

|  |  |  |
| --- | --- | --- |
| Input | Output | Conclusion |
| {0, 627, 343, 213, 171, 465, 526, 17} |  | The method has correctly produced the 7 bars (the 0 bar will not be drawn as it will have 0 height for both income and tax). It has also correctly positioned the incomes on the x-axis and has drawn the respective taxes in the correct positions |
| {295, 173, 481} |  | The method has correctly produced 3 bars with correctly proportioned and positioned income and tax bars |

I also tested the constructor parameter for types and negative numbers. The graph for this data cannot be shown as it caused a compilation error, the constructor of this object will accept only array inputs of type double, so values such as 1.0 will be invalid as it is not part of an array. The negative value will not be drawn, as there is a check to see whether the input is negative, if it is the method skips it.

Overall the method is working as expected and to the requirements of the specification.

printTable()

The printTable method prints out a table of incomes and their respective taxes to the console in a certain format. The method also checks to see if the income is negative or not, if it is the method will print an error to the console telling the user the position of the member of the array that is negative. Otherwise the method will use the taxPayable method to get the tax on the income and print out a row of the table.

I will be testing this method the same way I tested the draw() method, by providing the constructor of the object with different sets of data for its parameter. I will not be looking at the values that are returned by the taxPayable method as it has already been tested and I know its working.

The data I inputted:

{0, 627, 343, 213, 171, 465, 526, 17}, {295, 173, 481}

|  |  |  |
| --- | --- | --- |
| Input | Output | Conclusion |
| {0, 627, 343, 213, 171, 465, 526, 17} |  | The method has correctly printed the table in he specified format |
| {295, 173, 481} |  | The method has correctly printed the table in he specified format |

The tables that have been printed show the two values asked for by the specification (tax and remaining Income) along side their respective incomes. The method has successfully formatted the table making sure that each value is underneath its respective header and that they are all in line.

Again I have tested the parameter type and for negative numbers to see what would happen. The incorrect type produced a compilation error as expected, the negative number was noticed by the method and an error message was printed.



After testing the TaxChart.java file I am confident that it will meet the needs of the specification and function perfectly. I am sure of this as I have tested both the methods and have received the correct and expected output for these values, also values that may be incorrect or invalid will be recognised by the method and skipped provided they don’t cause a compilation error.

We have now tested all the parts of the program responsible for any kind of output or calculations and I have determined they are all working as intended and expected by the specification with no errors.

# BogOff.java

This file simply initializes the inputSet and then makes a new instance of the TaxChart object using the inputSet as a parameter, it then calls the two methods in TaxChart in order to get the appropriate output. Calling the single method inside of the main method and waiting to see if any output is produced from the methods that have been called can simply test this file.

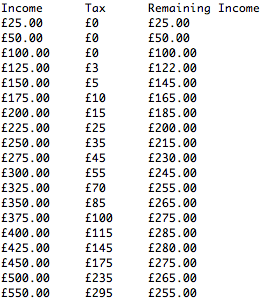
After running the program from this file I have seen that it correctly goes through the call hierarchy and produces the correct output from all the methods in the program.

Required output

In the specification I was given the following set of incomes to use as the input set in my program and retrieve output for:

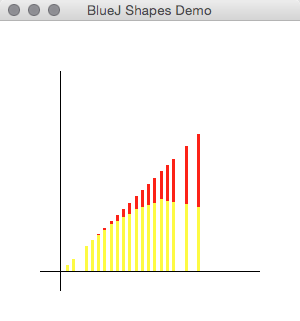
{25, 50, 100, 125, 150, 175, 200, 225, 250, 275, 300, 325, 350, 375, 400, 425, 450, 500, 550}

The program worked as expected and encountered no errors; the following is the table that the program produced:



The table shows that the program has calculated all the tax correctly; this is shown by the fact that all incomes under £101 have not been taxed (as the tax rate for values 0 – 100 is 0%) and that if you were to calculate the tax yourself you would get the same value as the program, the program has also rounded the tax correctly as shown by the rounding of the £2.5 tax on the £125 income (it has been rounded up to £3). The program has also formatted the output in the table correctly.

The program also produced a graph of the input set:



The program successfully produced an accurate and proportional graph that represented the input set incomes and tax on those incomes in a tidy and simple format. The positions of the income bars on the x-axis are proportional to the size of the income and the tax bars on the incomes have been placed on their respective incomes in the correct position, and the height of the bars themselves is proportional to their respective incomes and taxes.

Summary

After comprehensive testing of the program I have decided that it is functioning correctly and accurately. There are no logical errors in the program as all the methods are giving the expected output and no errors that would cause a crash or compilation error. I think the program has met all the targets presented by the specification and has done so in an efficient and structured manner.