**Overview**

We were asked to create a program which converted metric into imperial. The program would therefore have to take an input value, in meters, from the user. The input had to be a positive real number for the program to recognise it as valid. The program would then then calculate the number of yard, feet and inches, where applicable, from the number of meters input. Once the calculation was complete the program would output the conversion with the correct plurality and also omit feet and inches when their value is zero.

**Design**

The program first prompts the user to choose whether they wish to convert metric to imperial or imperial to metric. Once this choice is made the program will execute a new method. Multiple new strings are initialized, which are then manipulated throughout the program based on the plurality of each calculated measurement of distance.

Next the user is required to input a value in meters, which is stored as a float. This value will then be converted to yards, feet and inches. If this value is a positive real number the program will continue, otherwise an error message will be displayed and the user will be required to re-enter the value they wish to convert.

Once the program has received a valid input the conversion calculations code will be executed. The program will first convert the distance in metres to inches. The next calculation is the integer number of yards from the inches. Then, the remaining inches are calculated by subtracting the number of yards in inches from the inches. After this the integer number of feet from the remaining inches is calculated and finally the remaining inches are calculated by subtracting the number of feet in inches from the remaining inches.

I have used multiple ‘if’ and ‘else if’ statements which have enabled the program to detect whether the measurement of distance is equal to zero, one or more than one. Finally, I have used the Math.round function to round the value of the variable ‘inches’ to 2.d.p. Another method is then used to construct the correct string. The output string is then returned to the main method and printed.

If the user wishes to convert imperial distances to metric distances they will be able to do so by choosing when the program is initiated. If this is the case the user will be prompted to input three different values, these are: Yards, Feet and Inches. The program will then calculate the total amount of meters in the combined input distance. This value is then converted into centimetre’s. Following this a while loop is executed to calculate how many meters there are in the total amount of centimetre’s. Each loop increments the value of the variable ‘Meters’ and subtracts 1000 of the value of the variable ‘CM’. The loop continues until the variable ‘CM’ is less than 1000.

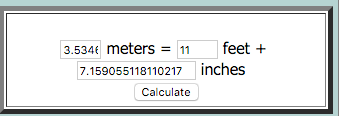
**Testing**

When it comes to testing my program, I will calculate the values the program should output by using a 3rd party unit converter, such as the one found at http://www.csgnetwork.com/. I will decide on test values and check the what the expected output should be. I will then compare these with the actual values the program output.

Another important feature that should be tested is the output formatting and I will make sure that the text output is in the correct format, with correct punctuation and plurality.

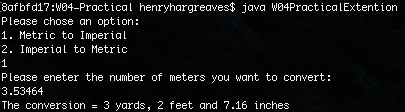
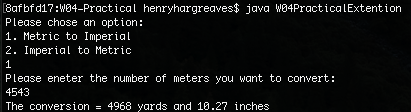
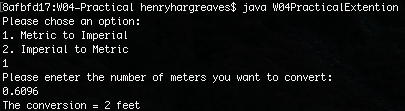
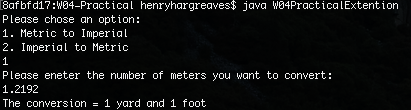
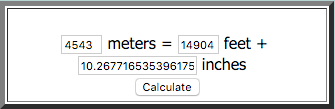
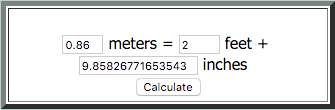
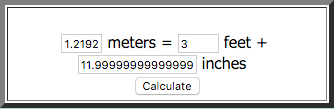
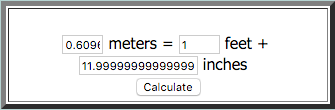
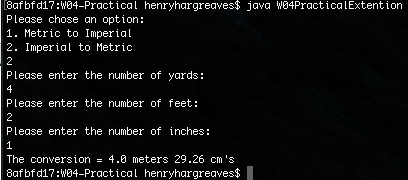
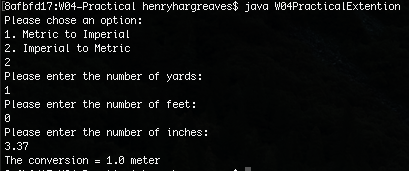
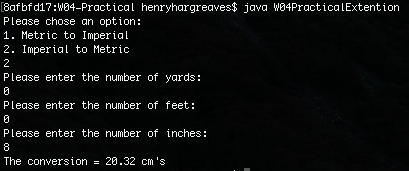
I will also make sure I have suitable test data when I am testing my extension; this will range from data that would expected by the program and may be commonly entered, as well a data that the program would not expect for example entering a letter, where an integer is expected.

Entering invalid data to try and produce errors will help find bugs and glitches in my program. Again I can then compare the expected outcome, and the actual outcome. If the program executes and uses the data in the correct manner, returning what I expected, I will know it works to a degree of accuracy.

**Examples**

The input was valid and this time the conversion left no feet and so the string printed only included yards and inches.

The first test, all inputs where valid and the output was as expected. As there 3 feet to a yard (11/3 = 3 reminder 2). The correct plurality and punctuation where also used.



I also tested the program to see how it delt with conversions with outcomes less than 1 meter. As you can see the output did not display any meters and just the correct quantity of centimetres.

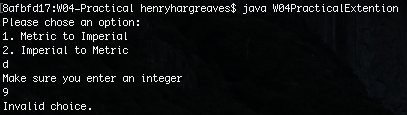
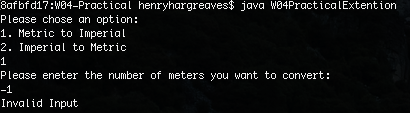
To test if the output would use the correct plurality I input value which would convert to equal 1 meter. As you can see the output from the program is therefore correct.

After carrying out the basic tests on the core functions in my program I then decided to test the extension. As you can see here I have entered 4 meters, 2 feet and 1 inch to convert to meters and centimetres. The output is displaying the correct conversion with the correct plurality.

The input was valid and this time the conversion left no inches and there was only one yard and one foot, so the string printed was correctly punctuated and used the correct plurality.

The input was valid and this time the conversion left no yards and so the string printed only included feet and inches.

The input was valid and this time the conversion left no yards or inches and so the string printed only included feet.



These two screen shots show the basic error checking I implemented into the program to catch invalid inputs.

**Evaluation**

Both my initial code and my extension worked as expected. The initial requirement presented quite a hard challenge and an efficient solution to the problem required different the use of different methods.

Without the extension the user was only able to convert from metric to imperial, the program was however able to calculate the correct number of yards, feet and inches in the number of meters input.

The extension I made allowed the user to choose which way they convert and thus made the program more flexible and interactive, offering users a greater range of functionality and generally extending the practicality of the program. The extension successfully allowed the users input a distance in yards, feet and inches and the program would convert this to meters and centimeters and display the result.

**Conclusion**

One of the first problems I had was: how efficiently calculate the number of yards’ feet and inches in the number of meters input. Because I also wanted the the user to be able to input decimal numbers this meant making the ‘Inches’ variables a float type but making the ‘Yards’ and ‘Feet’ variables integer types, thus ensuring they where whole numbers. However, some of the calculation needed make use of the dividing operator and so I had to implement code which would allow the program to divide two numbers and remove, where applicable, any reminder.

Once I has accomplished this, the next issue was to make sure that the correct plurality for each variable was displayed, e.g. inches if the value of the variable ‘Inches’ was greater than 1. I also had to ensure that the program would omit yards, feet and inches, when their value is zero, from the output. Finally, the program also had to recognise where to put what punctuation, so for example if the program were to print ‘yards, feet and inches’. The ‘,’ and the ‘and’ would be correctly used in a suitable position. To do this I used a separate method with multiple different if statements. The parameters handed to the method would then for fill one of the conditions and the correct string would be returned from the method.

Overall, I am happy with my program, as it meets the requirements and also has more features which makes the program more helpful. However, I If I had more time I would have liked to create a GUI (graphical user interface) which would have made the program more user friendly and may have also resulted in, making the process of inputting and outputting data a quicker and more intuitive process.