CS1002 W05 Practical Report

Samuel Wykes

October 8, 2018

Overview

For this task I was asked to take a certain integer number of kilograms as input and then convert this into imperial stones, pounds and ounces. This information would then be outputted in a readable way with the correct plurality for the units. The program also needed to return an erorr if the user entered a value that wasn't a positive integer.

Design

The first decision I had to make when designing this program was how to decompose the problem into easier to solve subproblems. Using a single method to do the whole program would be code inefficient, confusing for me, and time consuming. I decided it would be sensible to separate the conversion process into its own class which would be called Converter. I could further break down the conversion process by calculating the number of each unit individually and then recombining to give a final answer at the end. I would use conversion constants defined within the Converter class to tell the program how much of each unit corresponded to other untis for example I would have a constant to represent how many pounds are in a stone, ounces in a pound, and kilograms in an ounce. I knew that the Converter objects would also need to store the number of kilograms inputted and the number of stones, pounds and ounces in that number of kilograms so that a suitable output could be provided by the program. This meant I needed integer variables to store all these values. Central to the object would be the convert method which would take an integer kilogram paramater as input set the kilogram variable equal to that parameter and do calculations to set the imperial variables to the appropriate values. Because I needed to output the values in the form "x" stones, "y" pounds and "z" ounces I would need to work out the number of stones first as then I could work out how many pounds remained once the stones had been removed from a running total. Both the stones and pounds variables had to be floored so that you don't say more stones or pounds than were actually inputted. However the number of ounces were to be rounded as there was no smaller subunit being used by the program. I decided to break down the problem even further it would be wise to have three separate methods to convert between units to reduce code reuse. The convert method didn't need to return anything as instead it would store the calculated values within variables in the Converter object and then a separate method called printValues could provide a print out of the stored values.

The printValues method would be where most of the conditionals of the program would go as I would have to make sure that the correct plurality is used for each unit. This would be done by an if statement checking if the value of each of the respective unit variables were equal to one. If this is true then the singular version of the unit name would need to be printed, for example 1 pound instead of 1 pounds, however an if else statement would be called to check if the variable value was greater than one and then output the plural version. An additional check would need to be performed to check that the user had inputted an integer less than or equal to 0. If the stored kilogram value was less than or equal to zero the error message "Invalid input!" would be printed and the program wouldn't print anything else.

The rest of the problem would be solved in the main method in the W05Practical class. Here instances of EasyIn2 and Converter would be created to take input and do conversion respectively. The main class would also have to print a message to prompt the user to input the number of kilograms they wished to convert.

Testing

Because I decomposed the problem in the design stage of this solution testing the program was relatively straightforward as I could test each individual component separately. This made it easier to isolate and solve bugs. The easiest way to do this style of testing is to test each method individually comparing the expected values to the actual return values. One of the first thing I had to test was whether the individual unit conversions were working. Here I shall provide the testing results for the kilogramsToOunces method. Here I shall be testing a range of inputs and comparing the outputs to what I'd expect by using an online converter and comparing the output of calling the method from the main method with the result given by the online converter for the same arguments. From this I deduced that the constant I was using for conversion was incrorect.

Input	**Expected Value**	**Returned Value**
1	35.274	32.151239430280036
7	246.918	225.05867601196027
72	2539.73	2314.8892389801626

I changed this value to what it should be (it turns out my google search for conversion of kilogram to ounce gave me an incorrect value.) Once I reran the program the converted values were the same as those given by the conversion website so I deemed the method to be working.

Next I needed to test the actual conversion, the way I did this was temporarily making my getters for the different unit variables public and then doing the conversion and printing the variables' values to console. This could then be compared against values obtained using an online converter. In the below table the values in brackets represents (stones, pounds, ounces) outputted.

Input	Expected Values	Returned Values
1	(0, 2, 3)	(0, 2, 3)
7	(1, 1, 6)	(1, 1, 6)
72	(11, 4, 12)	(11, 4, 11)