Overview:

This coursework asked me to write a program that reads input from a user and converts between several different baking measures. The coursework placed particular emphasis on defining classes, calling their constructors and methods in other classes and using conditional tests.

My solution successfully implemented all parts of the specification.

Code Design:

As per specification, my code revolved around three classes, these being UserInerfaceMain, BakingMeasure and MetricVolume. The UserInterfaceMain class contained the main method and called the other two classes along with methods within those classes.

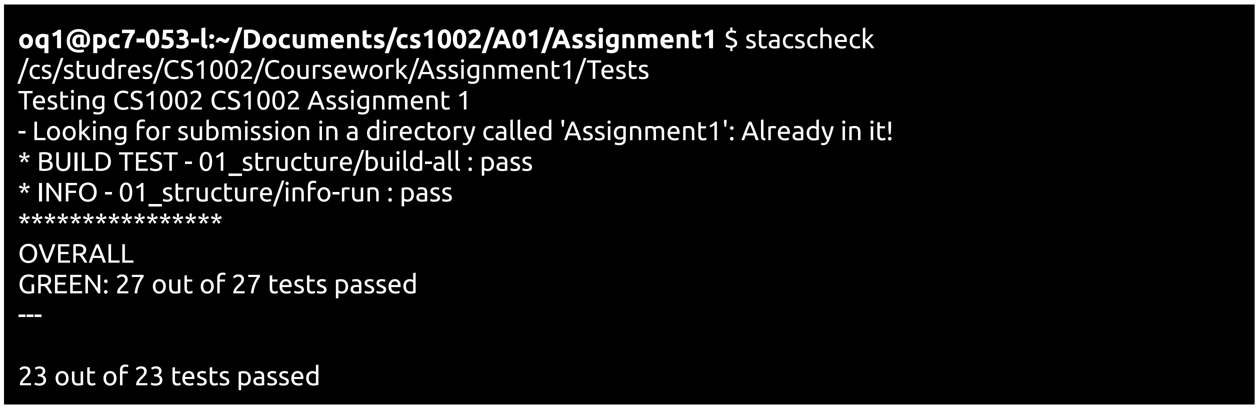
The BakingMeasure class' main purpose was to print out a number of cups, tablespoons and teaspoons in an aesthetic manner. To accomplish this I used a series of nested if statements in the prettyPrint() method. These if statements checked whether or not to print out certain units, pluralize certain units and add in commas or "and"s based on whether or not a certain measurement was present and whether or not there was more than one of a certain measurement. I felt this was fairly efficient as when certain conditions aren't met, much of the code within the method simply doesn't run as they are contained within if statements.

The MetricVolume class was intended to convert milliitres into cups, tablespoons and teaspoons. I accomplished this by using the division operator to convert millilitres into teaspoons and then used Math.round() to convert the quotient into an int data type. Then I used a series of division and modulo operators to turn this into cups, tablespoons and teaspoons. Finally, I created a BakingMeasure object and initialized its fields with the cups, tablespoons and teaspoons which resulted from the prior code. I felt that this was the most straightforward method for converting millitres into cups, tablespoons and teaspoons.

My UserInterfaceMain class was responsible for handling user input. It used a Scanner object to get user input and a switch-case along with two helper methods within individual cases to handle the input. I decided to use a switch-case because I felt it was clearer than using a series of if statements. With the switch-case, the code would only compare the user input to the cases one time instead of going through multiple if statements. I decided to use helper methods instead of coding directly inside of the cases in order to make the main method cleaner and more readable as well as making the program more reusable. The two helper methods were responsible for handling each case, the first ran if the user wanted to "pretty print" a baking measure and the second ran if the user wanted to convert a metric volume to a baking measure. The first helper method, called "bakingMeasureInput", queries the user for cups, tablespoons and teaspoons. It then creates a BakingMeasure object with its fields initialized to the user input. Finally, it calls the prettyPrint() method from the newly created BakingMeasure object. The second helper method, called "metricVolumeInput ", asks the user for an amount of millitires. It then calls a new MetricVolume object with its field initialized to the user input. The code then uses the convert() method from the newly called MetricVolume object to create a new BakingMeasure object and subsequently runs the prettyPrint() method on this new BakingMeasure object.

Testing:

To begin with, I made sure that my code passed all of the automatic stacschecks tests. The figure below shows the summary statements and the overall stacksckeck summary that resulted from the tests. For a more detailed view of the stacscheck tests, refer to the output.html file included in the submission.



```
oq1@pc7-053-l:~/Documents/cs1002/A01/Assignment1 $ stacscheck
/cs/studres/CS1002/Coursework/Assignment1/Tests
Testing CS1002 CS1002 Assignment 1
- Looking for submission in a directory called 'Assignment1': Already in it!
* BUILD TEST - 01_structure/build-all : pass
* INFO - 01_structure/info-run : pass
****************
OVERALL
GREEN: 27 out of 27 tests passed
---

23 out of 23 tests passed
```

Figure 1: All stacscheck tests passed successfully.

Next, I tested if my code caught any errors it was supposed to. These included, but were not limited to if the user input a number corresponding to a case which didn't exist in the main method and if the user input invalid numbers for any of the measurements when prompted after choosing either case. In addition, I tested the BakingMeasure class' prettyPrint() method for all combinations of plurality and omission between cups, tablespoons and teaspoons. I found that the method held up to this testing.

I also tested for very large numbers in my measurements and found that my code didn't work in these cases. Upon further research, I found out that this was a result of my inputs exceeding the maximum value of the int data type. This means my code will fail to function if the user inputs any number greater than 2^{31}. [1] In order to catch such inputs, the program would need to handle the long data type along with the int data type.

Evaluation:

My solution to the assignment was able to fulfill all parts of the specification. After completing my testing, I found my program to reliably behave as intended other than when the user inputs numbers that fall out of the range of the int data type.

Conclusion:

Overall, I found this coursework to be relatively manageable. I didn't find any parts of it particularly difficult. However, if I were to select the component of the coursework which gave me the most trouble, I would say that creating the prettyPrint() method and handling all the grammatical nuances of plurality and omission was complex and took time to implement.

With additional time, I would have liked to find a more efficient method to run the prettyPrint() method. While I think my solution was relatively efficient and accomplished what it needed to, I have the feeling that there remains a more elegant and faster solution to the problem. Furthermore, I would have liked to give my code the ability to handle user inputs larger than 2^{31}.

References:

[1] https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html