# Coffee Shop Coding Challenge:

Tom Chesser

## A picture containing text Description automatically generatedStructure

Text

Description automatically generatedThe solution developed utilises the functionality of object-oriented programming. There is one primary class called drink, and 4 interface objects. The Drink class has each interface object as a class attribute. Pricing and final description is calculated at runtime using a Drink class method.

Text

Description automatically generated

## Testing

I ran some basic tests using Junit 4 to ensure data quality. They focused primarily on testing total calculation and the final description output. These simple tests can be found in the source code under the DrinkTest Class.

## Evaluation

As the front end is likely to be a GUI application, the structure chosen was designed to allow easy modification/removal of class elements when making an order. Its segmented nature grants easy modification and allows the various sections to be easily expanded overtime. Interfaces were chosen to be used to ensure consistent class structure and functionality on each attribute. It also allows pricing to easy be calculated and formatted when a particular method is called. This allows the total to be re-calculated if modifications are made, without repeating the whole order process.

Error handling is minimal as this solution is primarily focused on the backend. While more time could have been spent ensuring that the primary attributes are valid values, this is something I believe should be handled in conjunction with the front end, restricting what and how the data is parsed.

## Potential Improvements

For a challenge as simple as this I could have cut the number of interfaces down to one, so every element shares the same structure. But I did this primarily for readability and package segmentation. It also would allow for diversification of class contents between attributes.