How to build the program:

My program is written in Python (version 3.8) and will run on any machine with Python installed.

How to use the program:

The program will prompt you to add an item to the basket by entering one of the letters ‘A’, ‘B’, ‘C’, ‘D’, or ‘E’. You can enter as many items as you like but they have to be added one at a time. To finish shopping and calculate the total of the basket, just enter the letter ‘X’ instead of one of the item letters. It will then output the total before and after the discounts are applied, and then the total including any delivery costs. When the total has been given, you are presented with the option to start again with an empty basket by entering the single letter ‘y’ or enter ‘n’ to stop.

My approach to programming the solution:

My program consists of 2 classes.

The UnidaysDiscountChallenge class deals with the user input and the output. I decided to output the total price before and after the discounts, before adding the delivery fee so the user can tell that the discount has been successfully applied, and also how much money they have saved.

There is a different class to deal with the prices of each item and the discounts that can be applied, called PricingRules. This allows the pricing and discounts to be changed easily without affecting the running of the whole program. I made the price variables defined within this class private variables, with a getter function for each, so they won’t be accidentally changed. For the scope of this challenge I think this is unnecessary, however it is good practice in object-oriented programming. It is also good to put them in as if the program was going to be expanded at a later date, where they would be needed.

I have added some simple validation to all user input, so that the program is not case sensitive. If the user enters an invalid input, the program will keep prompting for a new input until a valid one is given.

When deciding on my approach to the program, I considered making each item its own class, with its price as an attribute and its discount application as a method. I decided against this, as this approach would mean that a new instance of the item class would be created each time it was added to the basket even though there may be several identical instances of it already in the basket. Alternatively, there would be need to create an instance of each item before the any instance of the main UnidaysDiscountChallenge class was created in order to initialise this class, which wouldn’t be efficient as there is a possibility that not all or even none of these items will be added to the basket. I the end I think the two classes I used are enough and they give a much simpler structure than I would have created following my original plan.

If there is any problem with my program e.g. you can’t read the code or the program won’t run, contact me at brianna5556@outlook.com