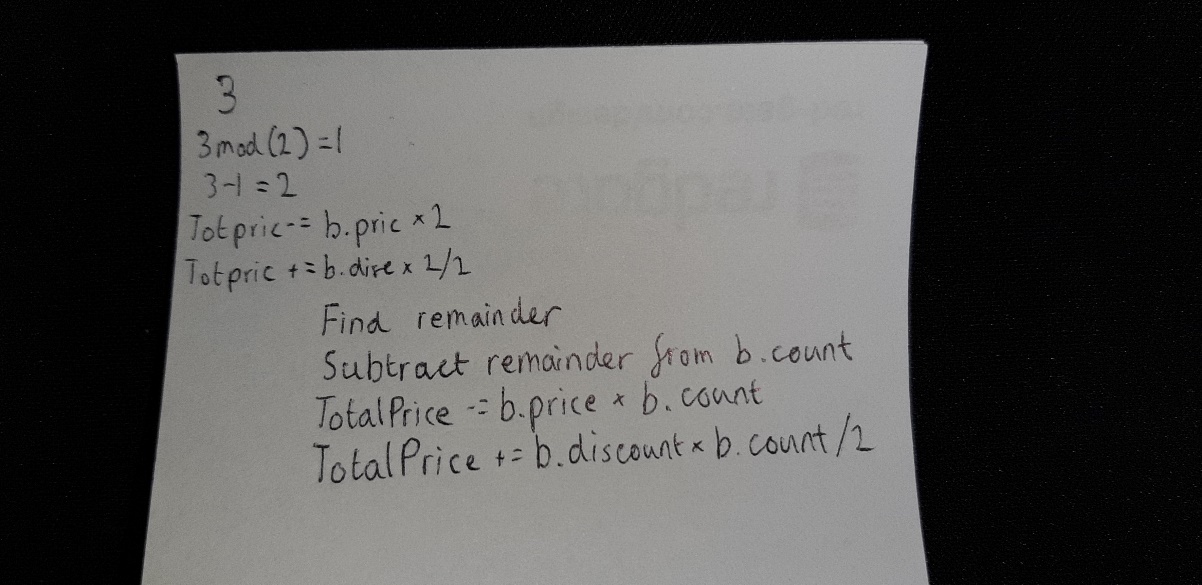
The first thing I did was set up a class diagram which would outline how the different classes interact with each other, and what each one does. At first, I was going to have the shop data exist in the window’s code area, but I decided to change that and separate the two which gives it a bit more versatility (one no longer depends on the other).

Once I had set up a rather basic looking window (it didn’t have to look good to start writing code for it), I wrote the lines of code that allowed the user to click on different items and add them to the basket. After this worked, I wrote the instructions at the top of the window for adding items. The basket was just a list of “Item” objects. I created an “Inventory” class which stored all the different item objects that could be bought. Clicking on buttons would add items from the inventory into the basket.

Next, I worked on removing items from the basket. Which just involves clicking on one of the items in either of the list boxes and then clicking remove. I had decided to use two list boxes to display the basket at this time, because it made formatting much easier (otherwise longer names would offset the price for that said item). I made it so that clicking on one of the boxes would automatically update the other box to select the respective item. This way, an item can be selected from either the name box or the price box. With this done, all I had to do was to get the selected index from one of the boxes, and then remove that index from the basket. I also had to introduce a failsafe which would output an error window letting the user know if they tried to remove something before selecting one of the items.

After this, I worked on setting up the discounts, this consisted of a set of methods in the Rules class which would go through the basket and count how many of each item there was, and how many discounts would have to be applied. I did some working out and came up with a simple algorithm for this on paper (seen to the right). This algorithm worked for all the discounts except for ItemD, which I simplified further because of the nature of the discount (buy one get one free)

After all the previous code worked, I decided to format the window to look more organised, so that it is easier for a user to navigate. This simply consisted of aligning the buttons, list boxes, and labels. I also changed the fonts to be larger. To set up the delivery section, I just checked if there was a cost of less than £50 in the basket, and not £0, if it satisfied this, it would show a deliver charge, and not if not.

I originally wanted to have the delivery added onto the total price and written one label, however, in the Uniday’s GitHub page the two were separated, so I kept it that way in my program too.