

Summary:

As my comments suggest, I quickly established that the program would be best split into three stages: Collecting input from the user to create their account, collecting input to create a wine purchase that affects the account balance, and exporting a summary of the transactions.

I am satisfied that part one meets the requirements of the assessment, and that it collects the appropriate data from the user. I was able to catch Input Mismatch Exceptions on the methods relying on scanners for numeric inputs, however I decided it would be safer to assume business names might have unusual or numeric characters that could make defining the input for the customer name difficult. I decided to allow the user to input anything into the initial variable, relying upon them to catch any misspellings or wrong characters. This can be seen in the figure below:

```
AssEx1 [Java Application] C:\Program Files\Java\jre1.8.0_181\bin\javaw.exe (31 Oct 2018, 10:12:38)
Please enter your name
2345trdsdbngfbxc
Please enter your balance:
345ytredsazvcxf
Please enter a valid number:
2345.976
Hello, 2345trdsdbngfbxc. Your balance is £2345.97.
Please enter the title of the wine you'd like to purchase or press return for the list of transactions.
```

This also demonstrates how I have approached the numeric inputs. Each balance and price per unit of wine is converted to pence upon input by multiplying the value by 100 and casting it as an integer. This helps catch typos on input like the one shown above and converts them to a correct value where rounding may have added or deducted a small amount from the user's initial balance.

While I tried to keep as much out of the main method as possible in the initial stage, in the second I have decided to split variables and the main functionality of the program between the main and other classes, in order to prevent either becoming too crowded or sparse and hard to follow. This is also why I created a whole class just to gather the user inputs, as well as it making the program easier to change or test. The main class is where the loop required for inputting wine is found, the maths for calculating the wine prices is in the wine class. Several Strings were required at different stages in the program, and I decided to keep ones required for the final output with the associated classes and ones immediately displayed to the user in the main method, near the stage in the program they were shown in order to make reading easier.

I applied the same logic to the title of each wine that I applied to the user names, to avoid any mishaps with unusual names. A side effect of converting the prices to integers was that occasionally the division required for calculating returns resulted in fractions being removed, so what when they were converted back to pounds the results were (infrequently) 'off' by a couple of 'pence'. Again, I decided this was preferable to any rounding which might unfairly favour the business or customer. The figure below demonstrates both these points:

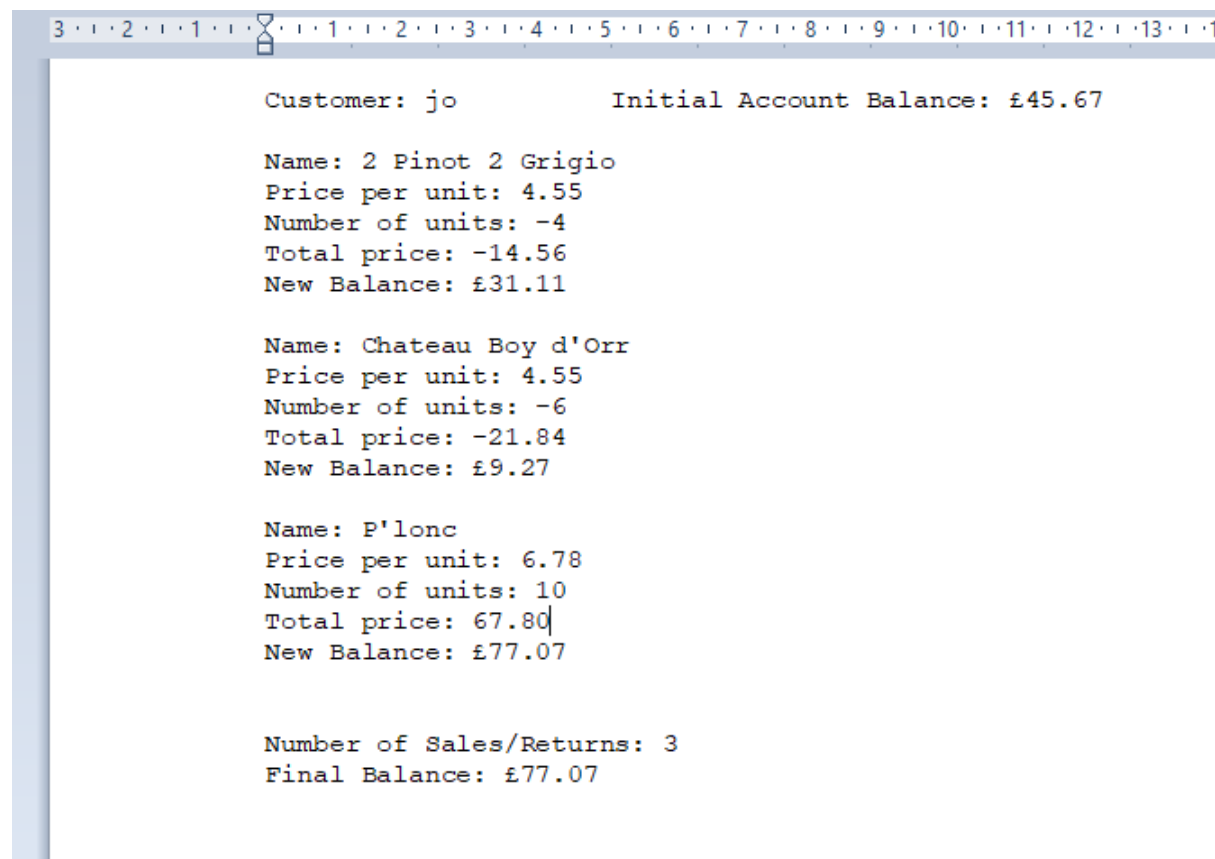
```
Please enter the title of the wine you'd like to purchase or press return for the list of transactions.
2 Pinot 2 Grigio
Please enter the price per unit of the wine:
7.89
Please enter the number of bottles you would like to purchase (Please enter the negative number for returns, eg: R:
-7
Thank you. After purchasing -7 bottles of 2 Pinot 2 Grigio, your new balance is: £-39.61
Price per Unit in pence: 789
Price in pence: -4416
```

If you use perform the calculations yourself, the result would be -44.184, and we might expect the final price to be 44.18. Removing any remainders during the calculations has resulted in a different price (I have also printed the wine's total and price per unit in pence for the demonstration, these are not included in displays to the user in the final program).

What I have been unable correct is when my code automatically rounds numbers down for reasons I cannot establish. Below we can see that when a user enters an amount as 4.56 it is multiplied by 100 and stored as an integer, but somehow as 455. I have asked a lab assistant about this and he was unable to explain why it was happening. It only happens occasionally and not consistently, and I've been unable to find problems in the code.

```
Price in pence: -1456
Please enter the title of the wine you'd like to purchase or pr
Chateau Boy d'Orr
Please enter the price per unit of the wine:
4.56
Please enter the number of bottles you would like to purchase (
-6
Thank you. After purchasing -6 bottles of Chateau Boy d'Orr, yc
Price per Unit in pence: 455
Price in pence: -2184
Please enter the title of the wine you'd like to purchase or pr
P'lonc
```

Beyond this I am happy that all requirements are met, as the third stage was without problems, as the final print out show below demonstrates. This also makes clear that the calculations are only affected by problems during any 'returns', and each affects the customer balance in the correct way.



```
Customer: jo          Initial Account Balance: £45.67

Name: 2 Pinot 2 Grigio
Price per unit: 4.55
Number of units: -4
Total price: -14.56
New Balance: £31.11

Name: Chateau Boy d'Orr
Price per unit: 4.55
Number of units: -6
Total price: -21.84
New Balance: £9.27

Name: P'lonc
Price per unit: 6.78
Number of units: 10
Total price: 67.80
New Balance: £77.07

Number of Sales/Returns: 3
Final Balance: £77.07
```