Programming AE1 Status Report

Kevin Wu 0808148w

The final state of my program consists of four classes (Wine class, CustomerAccount class, CustomerAccountException class and a class containing the main method). The program takes the user through various prompts to enter their name, opening balance and subsequent name, quantity and price of wine per transaction. If the filewriter doesn't work properly, they'll be prompted to input a directory and filename. One of my program's limitations is the reliance on String Format to display the balance to two decimal points. This way of displaying the currency doesn't properly account for rounding differences , the user may be shortchanged in some circumstances. Furthermore, I use an array to log all the user's transactions. Unfortunately, the array in the program must initialise with a fixed length, I a large enough number, 999, that will hopefully hold enough transactions. There are additional methods to grow and re-size the array in case the user either carries out more or less transactions respectively, than first imagine. These drawbacks lead to poorer efficiency regarding memory use. I could have chosen an arraylist instead which resizes in an ad-hoc manner and may be more efficient. Lastly, I also had to "re-run" the calculations through transactions again in the printing method to get the correct running balance. This duplication must also be a limitation.

The Wine class contains two constructors. One that takes no parameters and when initiated, includes an array as an instance variable. A separate constructor creates wine objects with instance variables – name, quantity and price per bottle (ppb) – resembling each transaction. I initially didn't consider using an array, thinking that using the name of the wine from the user input as a reference to a new wine object would be enough. However, I realised this could cause problems for example, the user could potentially enter the same wine name on more than one occasion but with different quantities and prices, therefore re-point pre-existing references to new objects. An array can store an object reference per transaction individually in a separate index.

The wine class has a method to add Wine objects. It checks that the array has space before adding another wine object – name, quantity and ppb – by considering the number of transactions carried so far and checking for equality with the size of the array. If there isn't enough space, it'll jump to the expandArray() method which grows the array size by one – (Figure 6 – this reflects one of the limitation of my program as previously mentioned.

Finally, the wine class has a print method, which prints the transaction history to a file on the desktop. This methods re-sizes the array so that it only contains the transaction/wine objects in each index leaving behind no 'null' ones (i.e. if the customer doesn't make 999 orders). If it tried to print without doing so, a null pointer exception would be thrown – (Figure 5). An object reference of the type CustomerAccount (i.e. the user details) is passed to it so it can access methods within that class. I initially struggled to get the correct running balance as the getBalance() method would return the final balance after all transactions had been completed. I overcame this by resetting the customer's balance back to the initial balance (i.e. number user came up with at the start) and going through the transactions with the wine objects stored in the array one by one again.

In the print method, the user is given the chance to input the directory and name of the file that they'd like to write the output file to if the String "filename" or ("outputFile" as it's called in my program) is incorrect. The method catches this as an IO exception or if the user specifies a directory rather than a file – (Figure 7). I was concerned that the user may accidentally specify a file path that already contains a file of the same name and had considered coming up with a method to check that the path is writable and to prompt the user whether they'd like to overwrite a file of the same name or if they preferred to input another different file path (by using 'y' or 'n') – this turned out to be far more complicated than I initially hoped so these checks were left out of the final program.

Various methods involving user input were initially in the main method but I moved them to this class for clarity. An object reference of the type CustomerAccount is initially set up in main to have access to methods within that class. A new object is returned containing the username and initial balance

The 'CustomerAccountException' has been included to throw a unique exception when the user completes simple mistakes such as entering £0 for price or quantity or entering a fractional number for quantity (Figure 1). I did have concerns that the user may not enter a valid string for a name (i.e. mistakenly include special characters, unnecessary spaces and/or number) and used a for loop to check a string made of special chars (Figure 3).

Programming AE1 Status Report

Kevin Wu 0808148w

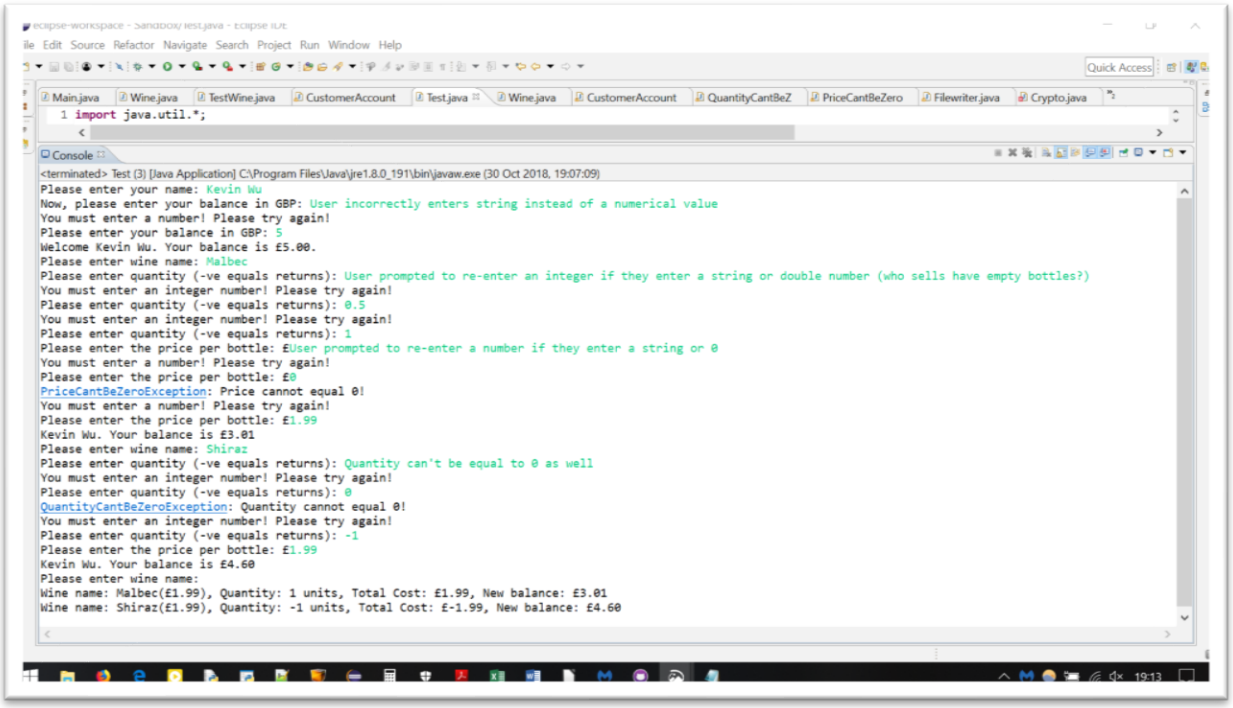

Figure 1: Eclipse Console showing user input and potential errors

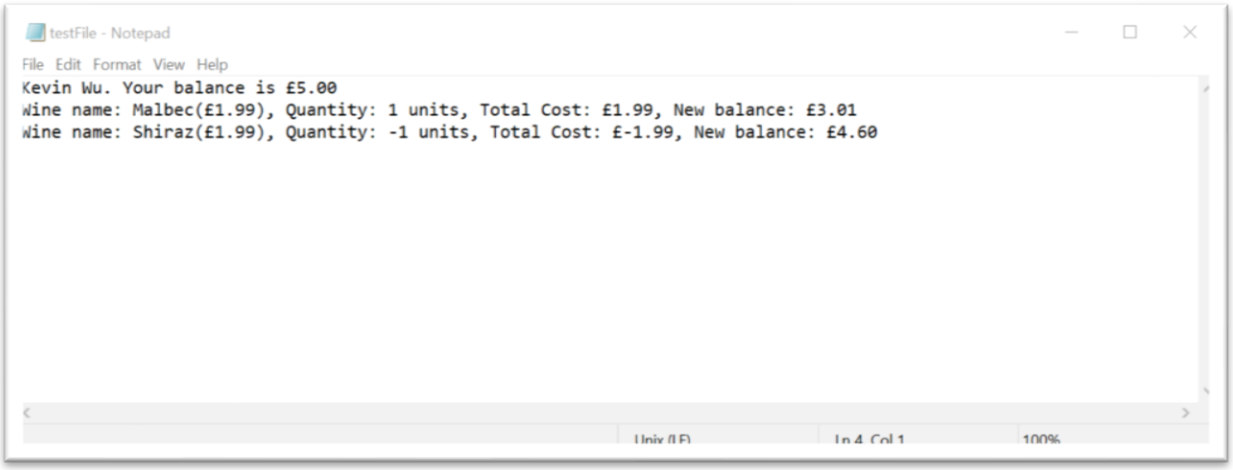

Figure 2: Output from inputs shown in figure 1 (user errors not displayed in output file)

Programming AE1 Status Report

Kevin Wu 0808148w

```
     */
    public String returnName(Scanner s) {
        String name = ""; //I've found that I Need to initialise name otherwise would return null
        System.out.print("Please enter your name: ");
        do {
            try {
                 name = s.nextLine();
                if (name.equals(" ") || name.isEmpty()) { //if user input is not satisfactory, exception is thrown, oth
                    throw new CustomerAccountException ("Name cannot be empty!"); //user is asked to input again theref
                }
                String test = "`¬!£$%^&*()_-+={}[]:;@\'<,>.?/|\'\"\\\t\r\n\f"; //String containing special chars which
                boolean containsSpecialChar = false;                            //user input name contains the char @ i
                for (int i = 0; i<name.length(); i++) {
                    for (int j = 0; j<test.length(); j++) {
                        if (name.substring(i,i+1).equals(test.substring(j,j+1))) {
                            containsSpecialChar = true;
                            throw new CustomerAccountException ("Name cannot contain special characters!");
                        }
                    }
                }
                break; // once user enters a correct name, no thrown exceptions, we can safely break out of do-while lo
        } catch (CustomerAccountException e) {
            System.out.println("You have entered an incorrect name!");
            System.out.println(e);
```

Console

```
<terminated> Test (3) [Java Application] C:\Program Files\Java\jre1.8.0_191\bin\javaw.exe (4 Nov 2018, 23:10:38)
Please enter your name: KÉvin
You have entered an incorrect name!
CustomerAccountException: Name cannot contain special characters!
Please enter your name again:
You have entered an incorrect name!
CustomerAccountException: Name cannot be empty!
Please enter your name again:
You have entered an incorrect name!
CustomerAccountException: Name cannot be empty!
Please enter your name again: Kevin
Now, please enter your balance in GBP:
```

Figure 3: Checking valid name input

```
//      } while (s.hasNextDouble());
//      return balance;
//  }

    public String returnName(Scanner s) {
        String name = ""; //I've found that I Need to initi
        System.out.print("Please enter your name: ");
        do {
            try {
                 name = s.nextLine();
                if (name.equals(" ") || name.isEmpty()) {
                    throw new StringException ("Name cannot
                }
                break;
        } catch (StringException e) {
            System.out.println("You have entered an incorre
            System.out.println(e);
            System.out.print("Please enter your name again:
        }
        } while (s.hasNextLine()); // not sure if I'm going
                                   //with this condition.
```

Console

```
Test (3) [Java Application] C:\Program Files\Java\jre1.8.0_191\bin\javaw.exe (30 Oct 2018,
Please enter your name: Kevin
Now, please enter your balance in GBP: 1200
Welcome Kevin. Your balance is £1200.00.
Please enter wine name:
```

```
//      } while (s.hasNextDouble());
//      return balance;
//  }

    public String returnName(Scanner s) {
//      String name = ""; //I've found that I Need to initialise name o
        System.out.print("Please enter your name: ");
        do {
            try {
                String name = s.nextLine();
                if (name.equals(" ") || name.isEmpty()) {
                    throw new StringException ("Name cannot be empty!")
                }
                break;
        } catch (StringException e) {
            System.out.println("You have entered an incorrect name!");
            System.out.println(e);
            System.out.print("Please enter your name again: ");
        }
        } while (s.hasNextLine()); // not sure if I'm going the right w
                                   //with this condition. Have similar
```

Console

```
Test (3) [Java Application] C:\Program Files\Java\jre1.8.0_191\bin\javaw.exe (30 Oct 2018, 19:43:48)
Please enter your name: kevin
Now, please enter your balance in GBP: 1200
Welcome null. Your balance is £1200.00.
Please enter wine name:
```

Figure 4: String name requires initialisation otherwise defers to a null value

Programming AE1 Status Report

Kevin Wu 0808148w

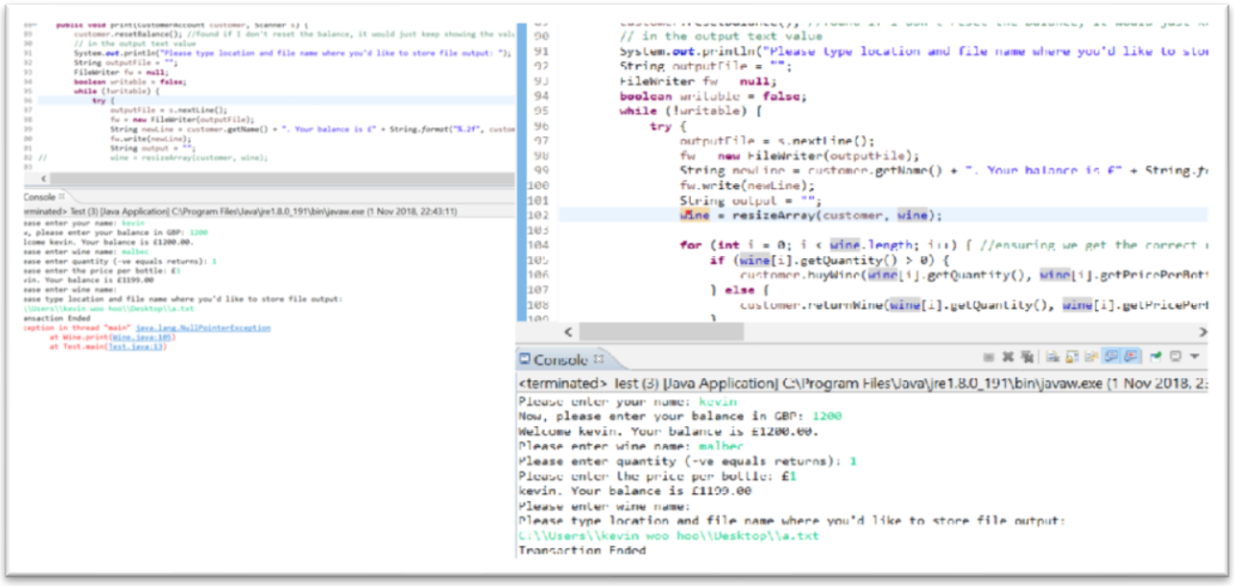

Figure 5: Null Pointer Exception when trying to print an array containing null values - requires the resize method

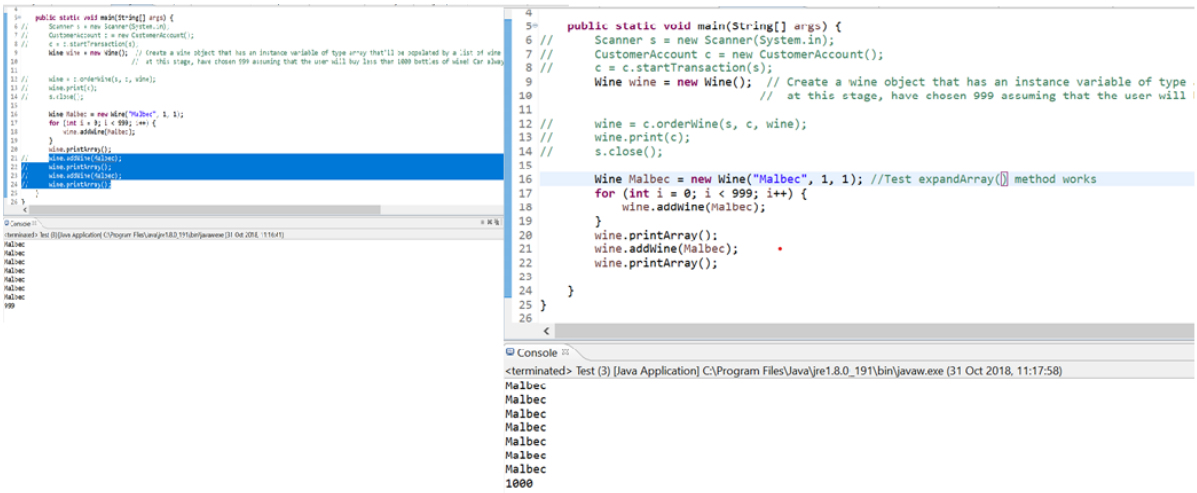

Figure 6: Checking expandArray() by printing it out

Programming AE1 Status Report

Kevin Wu 0808148w

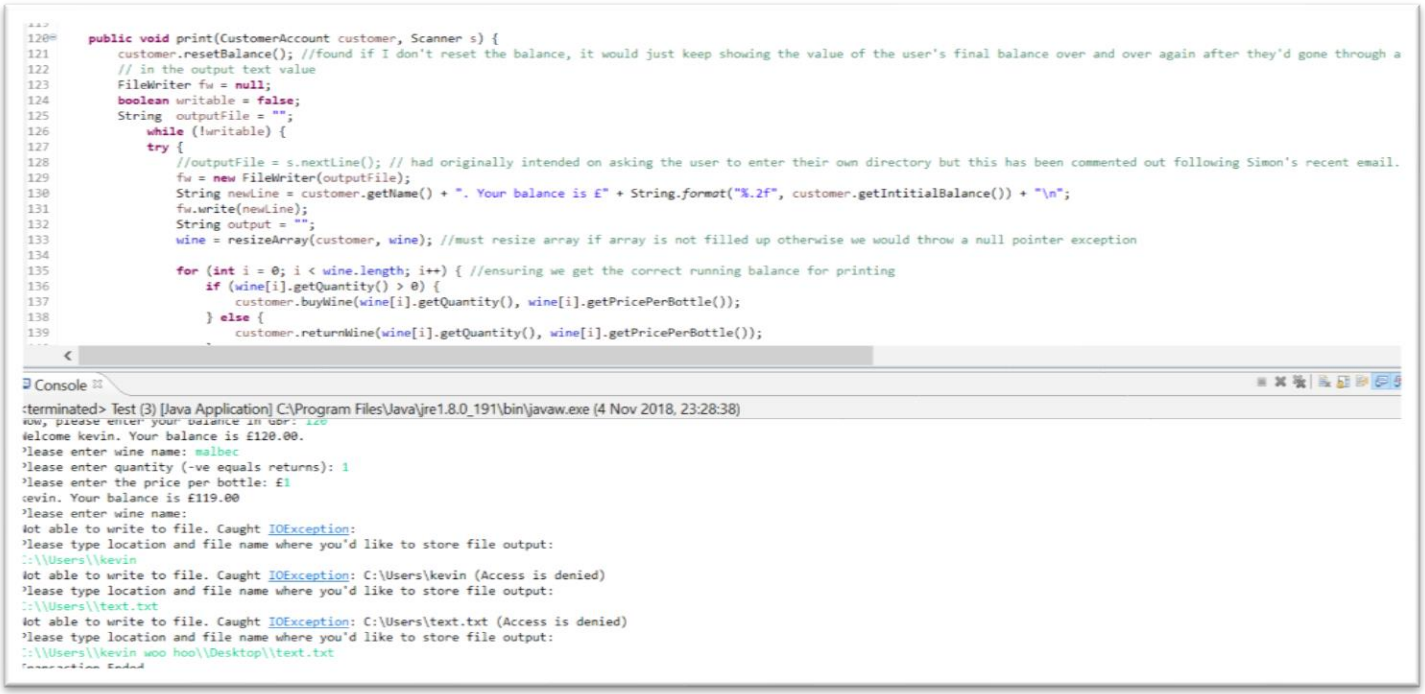

Figure 7: Catching the IOException if filename is incorrect and then letting user decide where to store the file.