Mini Project #2: Vending Machine Simulation – Reflection

**General Program design**

This program is designed around the two main elements: the currency information and the products that are loaded. The currency information is loaded through the LoadMoney class and the loadMoney method. It is read in through a BufferedReader which wraps a FileReader by splitting the text document with a Scanner nextLine. When the nextLine is not null or empty, the processing continues. The line is parsed with the .split() command to separate the document’s elements into a temporary array based on a delimiter (in this program, the currency is delimited by a “:” and the products are delimited with a “,”). The elements are then loaded through a LoadMoney constructor and finally assigned to the static array, machineMoney.

The currency file is organized into three elements: the denomination of currency, the quantity loaded into the vending machine (to be available as change), and the value of each denomination. The first 7 lines of the document can hold any currency denomination information. Line 8 holds the name of the currency loaded, while the last line holds the currency symbol and the value modifier (in the value element). This is used to divide the base value (which products are loaded with), so that it can be used by any currency.

The products are similarly loaded, through the Inventory class into the productList static array.

User’s inserted money is stored in the denomsInserted static array in the CurrentCredit class. denomsInserted holds the quantity of each denomination in a different array index element. For example, currently nickels are held in denomsInserted[0], and increasing denominations are held in increasing index elements of the array. The totalCredit is calculated by multiplying the quantity of the denomsInserted elements by the value of their counterpart elements in the machineMoney array. The getValue method of the MachineMoney class is called on the machineMoney array to extract the value modifier for each index of the array, and is multiplied by the quantity of the denomsInserted of the same index element.

The main method controls the looping of the program from the VendingMachineSimulator class. After declaring some instance variables and calling the loadInventory and loadDenoms methods, it collects user input on which actions to take in a while loop. Based on the switch cases of user’s input, main menu options include redisplaying the main menu, making a purchase, viewing available credit, adding money, viewing the available items in the machine, and cancelling the transaction.

Various try/catch statements catch user input errors when credit is too low to purchase items (IO Exception) or selection is out of range (NullPointerException).

**Alternative Approaches**

An alternative approach considered in the final week of the program is one where the money is stored externally, in a text document. This approach would simply read in the products and currency, complete the transaction at hand, and then write the details to a text document until needed. This would be more practical in the real world because it would hold on to the information after one user completes a transaction.

Also, in the beginning of my program, I outlined some interfaces that I would use. In the end, I did not use them because each function provided by the machine is fairly unique and the overlap didn’t seem to warrant the use of distinctly creating the interfaces. However, an interface might be useful for the “cancel” function, which is slightly different depending on the point at which the user cancels a transation.

Additionally, I first designed my currency document to hold the machine’s money in the first 8 lines and the user’s inserted money in the next 8. After working with the program, I realized I didn’t need to have two sections, I could just read in the machine’s money information and create a single dimension array for the user’s input, cross-referencing when needed for denomination names and value.

Finally, if I were to do this project again, I would probably create a holding container where inserted money would be stored until the transaction completes. This would make aborting the purchase a lot easier.

**Lessons Learned**

I learned a lot from this assignment. First, the importance of keeping your code organized and flexible. I don’t think I did a very good job of that in this program. Throughout the assignment I had to restructure multiple elements and in the end, I lost some of the functionality I once had while trying to correct for other issue. I think this is due to some overlapping methods and redundant variables.

I did get a lot of practice referencing class objects and working with different visibility levels for objects. I also decided to use some static objects for some of the most important and commonly used objects.

This was definitely a learning exercise and I still have many areas where I can improve my coding.