Kent Murray

VendingMachineSimulator

1. General program design. How is the program organized? What major data structures were used? How did you divide the functionality among your classes? How are commands processed? Etc.

2. What alternative approaches were considered and why were they rejected?

3. What did you learn from doing this project and what would you do differently?

The whole key is figuring out how to break the problem into pieces.

I wanted to make sure I try and follow your instructions and not to make the vending machine specific to US currency, so I will model 'Coin' and 'Bill' objects. The vending machine code will only refer to Coins and Bills as Objects that have a getValue() method that returns their value in the most granular (smallest) value possible. For US currency, the smallest value is one cent, for Euro currency, it is one Euro-cent, for UK currency it is one penny, for Mexican currency it is one peseta, etc. That way our vending machine will work anywhere in the world; all we have to do is say what local currency is being used and what coins/bills are accepted by the vending machine. For example, even though we have a one-cent US coin (penny), vending machines don't accept it, because of the difficulty of holding so many coins. When it came to the Euro coins: <https://en.wikipedia.org/wiki/Euro_coins> I needed to know what coins they use. I decided that my vending machine should not accept 1-Euro-Cent or 2-Euro-Cent coins, for the same reason US machines don't take pennies.

I think of the vending machine as having these components:

ProductStore - this is where the Product inventory is stored. It might a method like dispense(ProductCode) that is called at the end of a successful sale, to tell the user "Here is your Twix Bar" for example, and to reduce the inventory of that product by one. It might have a load(Product[] products) method to add products to the inventory. The load method would be called when the attendant stocks the machine.

ChangeStore - this is where Coins are stored to make change. It might have a load(Coin[] coins) method to be called when the attendant stocks the machine, and a makeChange(int amount) method that is called when a sale is being made, and the user has put in more money than the requested product costs.  The makeChange will have to figure out what combination of coins to give out as change, and make sure there are enough Coins in the machine to make change. Suppose we owe the user 30 cents change. The 30 cents might be returned as:

1 quarter + 1 nickel

3 dimes

2 dimes + 2 nickels

1 dime + 4 nickels

6 nickels

All this will be done using the value of the Coins, no assumptions of what currency system is actually used.

Depository - this is where the Coins and Bills go after a successful sale. This is not the same as the ChangeStore. The attendant will periodically empty the depository when he services the machine, at the same time as he adds products and change. There will be no use of Coins inserted by previous users to make change. Even if the Depository has plenty of quarters, we won't use them to make change - only change put into the ChangeStore by the attendant on a periodic service visit will be used. I don't know if that is how actual vending machines works, but it make sense to me. What if in a US vending machine, a user was able to sneak in a Canadian quarter and pass it off as a US quarter? If the user got away with that, you don't want the machine passing on the Canadian quarter to the next user in change.

Controller - this is where the logic connecting the components is placed. It reacts to user commands, recognizes when the user has selected a product, when money has been put in, when change is needed, and when the sale is complete and the product should be dispensed.

This was a very challenging project. It tested your ability to build out an activity chart, breaking this down in pieces, a constant amount of rework. A lot of research and frustration! I admire you Professor Deal for your great deal of expertise and knowledge. Some people have a natural skill for this and I have to work for it. I will continue to grow my knowledge and skills in Java as it’s a must have.