Vending Machine MiniProject Write-up

Alex Paris

Johns Hopkins School of Engineering for Professionals

Vending Machine MiniProject Write-up

This Vending Machine Writeup is the text that accompanies a JAVA programming assignment for JHU Engineering Professionals Intro to JAVA course, my approach is detailed below.

General Program Design

The Vending Machine is emulated using an inheritance tree and some polymorphism. A class called inventory is extended to various subclasses representing inventory items found inside the Vending Machine (e.g. Coke, Juice, Water, Pretzle). Each of these classes contains a static variable with the price in USD corresponding to the item. Getter and Setter methods from the Inventory super class can be extended here (such is the case in the constructors). The Inventory Class starts out with instance variables displaying prices, maxQuantities for the machines, and currentQuanities (set to maxQuantities) at instantiation. Then the Class contains methods that set inventory prices, add and remove quantities and return quantities of inventories, see the JAVAdocs documentation for detailed names of these members. The main Method is located in the VendingMachine Class. Vending Machine makes use of the java.util.Scanner class to take user input surrounding menu navigation; purchase options are displayed via calling the menu class methods (showMenu, showSelectionMenu, selectionItem) as well as Transactions methods (show Inventory, Show Prices) to display. In tandem with the call to the menu class, user input is then fed into methods of the Transactions class, particularly the “purchaseItem” method which processes the user’s selection (calculates change, vends the item, changes inventory counts). The high level algorithm for calculating change of a transaction basically takes the total value provided by the user (sum of inputed 5s, 1s, quarters) and separates it into two parts: the bill part and the quarters part. This is accomplished by using the Math.floor method from the Java core library. Floor

truncates the total value into the bill portion and the decimal part is found by taking the change – bill portion. A series of if -else statements then tests if the truncated bill portion of the change is 0.25, 0.5, 0.75 or 0. Depending on the outcome Quarters an appropriate number of Quarters are given back to the user of the Machine (e.g. 1 for 0.25, 2 for 0.5 etc). The remaining bill change is then given out in 1 dollar bills. All currency counts are defined and updated with static members of the Currency class. There are more nuanced nested conditionals that handle error cases of no change available , too little currency inserted for a given selection, or no inventory available (this last one is handled in the Main method). Finally all currency counts are handled in the Currency Class through static members.

Alternative Approaches

My implementation reformats much of the menus and user displays in the instructional video that are tough to read (due to long lines) into smaller compacts menus that use many line breaks. My change algorithm is a bit naïve in that it will display "out of change" in the rare cases where it's out of dollars and could use quarters to give change for an integer change amount. This drastically reduces the complexity of all the nested conditional statements though, so it's a bit of a trade-off.

What did you learn and what would you do differently ?

I learned how to create various JAVA classes that follow an inheritance tree, call instances of them from other classes, call static members and methods from other classes, use iterative logic including Loops and Switches, use the Math.floor() method. approximate solution as it could undershoot slightly). If I had more time I'd tie the Currency Class into the Inventory inheritance

tree since it's a type of inventory conceptually speaking. I'd also make the change algorithm more sophisticated to handle the aforementioned case from the Alternative Approaches section.

References

[1]Java – floor () Method [Web log post]. Retrieved November 8, 2017, from https://www.tutorialspoint.com/java/number_floor.htm