Running Head: DESIGN AND ANALYSIS OF VENDING MACHINE SIMULATOR

Design and Analysis of Vending Machine Simulator Title Page

Code Summary and Reflection

Brian Loughran

Johns Hopkins University

General Design

This program is a text based command line program which is designed to mimic core functionality of a vending machine. Two major objects are used to make the handling of this logic a bit easier. The first is MoneyInMachine. This handles some simple accounting for each of the denominations of money that can be placed in the machine, such as adding or subtracting coins, calculating the total value of coins, or generating change from inputted money and price. The second object is VendingItem. VendingItem is an object which represents something which can be loaded into machines, including its name, packaging, cost and the current quantity.

One item to note for MoneyInMachine (but also throughout the whole code) is that money is calculated using integer values (where 1 is $0.01, and $1.00 is represented as 100). This arose from the difficulty of using doubles for calculations. E.g. System.out.println(0.1d + 0.2d) -> 3.00000000000000004. This is painful. However, by multiplying values by 100, I was able to have the value of money be int, therefore making addition and subtraction easier. Stripe (an online payment platform) does a similar thing in their code.

This program is designed to be easily swapped for different currencies. This is done by keeping the names for the money denominations vague. Instead of “pennies” I used “hundredth\_coins” (because a penny is 1/100 of a dollar). This is done for all denominations, and allows for easily changing currency, whether that be euro, ruble, yen, etc.

Commands are processed in a switch statement which takes up the bulk of the program’s main() method. The switch method automatically prints the menu on the first iteration (program open), and gives the user 5 options of actions that they can do (display menu, display vending inventory, display money in machine, buy an item, exit).

Alternative Approaches

One question in implementing the buy() method was how to handle what happens when a user does not put in enough money to purchase the selected item. One method would be if the user did not specify enough money to decline the transaction, and not have their money enter the machine. This, I thought, was not fully accurate for the behavior of a real-life vending machine. A real machine would take the money, and if it is not enough as specified by the price, to return the money as change. The money, for a time, is still in the machine. So I implemented returning change in this fashion. I found it to be slightly more accurate in imitating the behavior of real-world vending machines.

Learning and Looking Back

Through working on this project I learned quite a bit. The scope of the project is quite nice in that it is a good summary of the programming constructs we have worked on recently in class. A vending machine gives good practice in creating classes, dealing with exceptions, and dealing with different data types.

After finishing the code, and thinking about other currencies it may be better to allow higher bill denominations. For example, the Zimbabwe dollar has an exchange rate of 30,000 to 1. The highest denomination available in my implementation is 10. It would be quite time consuming for some currencies to only be able to put in up to ten bills. An improvement could be to add more available denominations.

References

https://stackoverflow.com/questions/10977638/loss-of-precision-after-subtracting-double-from-double