Victoria Lyman

Professor Pablo Rivas

Software Development I

# Project 2: Writeup

8 May 2017

ID: 200-57-914

**Abstract**

This paper discusses the coffee ordering program created for Project 2 in Software Development I. It explains the methods contained in the java program, what they do, and how the user interacts with the program. It also discusses the inspiration behind the project, similar programs out there, and how to avoid user errors when running the program.

**Introduction**

As an avid coffee drinker, I hate when my coffee is incorrectly made. Even when I specify exactly what I want there is still times my drink has been wrong. This has influenced me to work on this project. A coffee ordering system, which allows users to specify their order, down to specific cream and sugar measurements. After all the specifications are in place users will have the option to verify and change if necessary before finalizing the order.

**Detail System Description**

I would like to briefly mention that the project changed drastically from the milestone writeup. Many of the methods mentioned in the milestone were renamed. Almost all of the methods are called within the main method now, not from other methods like the previous iteration. Also the way the order is saved and displayed changed as well. Basically I realized most of my program was very messy and thrown together and needed a change in order to be more efficient.

The entire program is in a class called “Project2” (very creative, right?) Within the class is a Scanner, named “input” to take user input. There is also eight arrays of strings in the class, that are used to display the various coffee options that are printed throughout the duration of the program. The arrays are named “fullorder”, “types”, “sizes”, “rOrD”, “creamer”, “milk”, “flavors”, and “noyes”.

In the main method another array of strings is created named “order”. Unlike the other arrays, this array does not have any declared strings. It is created to hold up to 6 strings based off the user’s input for their coffee order. From here the user is welcomed to the coffee shop and asked what they would like to order. Then seven methods are called to initiate the rest of the program. The methods are named “coffeeType”, “sizeSelection”, “regDecaf”, “creamSelection”, “sugarSelection”, “flavorSelection”, and “finalizeOrder”. Each method takes the “order” array of strings.

The “coffeeType” method prints out the “types” array. Users then type in the number corresponding with the drink they would like. For example, if a user wanted an Iced Latte they would input the number 4 because the program prints out “4 – Iced Latte” as one of the options. The user input is saved as an integer named “t”. Then “t” is put into a switch which designates what is saved to order[3] in the “order” array. In the example I just presented, “Iced Latte” would be the string saved to order[3] in this scenario.

The rest of the methods are all very similar. The method “sizeSelection” prints out the size options “1 – Small”, “2 – Medium”, or “3 – Large” from the “sizes” array. The user then inputs a number saved as an intger named “s”. “s” is put into the switch which declares order[0] in the “order” array as one of the sizes.

The method “regDecaf” asks the user if they would like regular or decaf coffee. The user inputs a number, saved as an integer “rd”, which is put into a switch to declare order[1] as “Regular” or “Decaf”.

The method “creamSelection” is slightly different from the previous ones. The “order” array is converted to a string named “orderString” to check the current selection. If the string contains “Coffee” the user is then asked if they would like to add a creamer with the options of “0 – No” or “1 – Yes” outputted. The user then inputs a number saved as “ac”. If “ac” is equal to 1 the creamer options will be printed from the array called “creamer”. The user then inputs a number corresponding to the creamer option they would like saved as an integer named “c”. The user is then asked to input another number based off how many tablespoons of the creamer they would like. This input is saved as a double named “cmeasurement”. The integer “c” is put into a switch and then order[4] is declared as a string beginning with the “cmeasurement” value and ending with a string “tablespoons of” and the name of the creamer type. If “ac” is equal to 0, order[4] is declared as null and the program continues onto the following method. If the user is going through the “creamSelection” for a second time, meaning order[4] is not null and “orderString” contains “Coffee”, the user is asked if they would like to remove the creamer or change it with the options “0 – No (Remove creamer)” or “1 – Yes (Change)”. The user input is saved as the integer “ac” and then the program continues the same as above based on if “ac” equals 0 or 1. When “orderString” does not contain the word “Coffee”, meaning it is a latte or macchiato, the user is asked what kind of milk they would like and the options are printed from the “milk” array. The user inputs a number saved as integer “c” which is put into a switch to declare order[4] as one of the milk options.

The next method called is “sugarSelection” which runs similarly to “creamSelection”. The user is asked if they would like any sugar with the options “0 –

No or 1 – Yes”. The user input is saved as an integer “su”. If “su” equals 1 then the user is asked how many teaspoons of sugar they would like. The user then inputs a number, which is saved as a double called “smeasurement”. order[5] is then declared with the value of “smeasurement” and ends with “teaspoons of sugar”. When “su” equals 0, order[5] is declared as null. If the user is going through the method for a second time, meaning order[5] is not null, the user is asked if they would like to change their measurement or remove the sugar. The options outputted to the user are “0 – No (Remove Sugar) or 1 – Yes (Change measurement)”. The user inputs a number saved as “su” and continues the program the same as above based on the value of “su”.

The next method called in the main method is “flavorSelection”. The user is asked if they would like to add a flavor with the options “0 – No or 1 – Yes”. The number inputted by the user is saved to an integer called “af”. If “af” equals 1 the program prints out the flavor options from the array “flavors”. The user then inputs a number saved as an integer “f”. “f” is put into a switch which declares order[2] as one of the flavor options. If “af” equals 0, order[2] is declared null. If the user is going through “flavorSelection” for a second time, so order[2] is not null, the user is asked if they would like to change their flavor with the options “0 – No (Remove Flavor)” or “1 – Yes (Change Flavor)”. The user inputs a number saved as integer “af” and continues running like above based on if “af” equals 0 or 1.

The final method initiated in the main method is “finalizeOrder”. The user’s order is printed from the “order” array. The user is then asked if it is correct with the options “0 – No or 1 – Yes”. The number the user inputs is saved as an integer called “finalize”. If “finalize” equals 1 then the program prints out “Thank you for choosing the

Coffee Shop! Enjoy your drink!" and the program ends. If "finalize" equals 0 then the method "editOrder" is called.

The "editOrder" method asks the user what they would like to change and prints the array "fullorder" to display the options. The user then inputs a number saved as an integer "edit". If order[edit] is equal to null the program outputs "You currently do not have anything selected for this." If order[edit] is filled then the program displays their current option saved in that string. "edit" is then put into a switch and calls one of the methods that have already ran including "sizeSelection", "regDecaf", "flavorSelection", "coffeeType", "creamSelection", and "sugarSelection". After one of those methods is called the array is converted to a string named "orderString". The string is used to check if the coffee type has changed and the creamer type do not go together. For example, if the user switched from a "Hot Coffee" to an "Iced Macchiato", there is no measurement of cream for a macchiato so the creamer must be changed to a milk selection. In this case the string would contain both "Macchiato" and "tablespoons". The program would then output "You selected a new coffee type that has different cream specifications." The "creamSelection" method would then be initiated. After all of this the user is then asked if they would like to change anything else with the options "0 – No or 1 – Yes". The user input is saved in the integer "editing". This method will continue running all of the above in a while loop until "editing" is equal to 0. Once the while loop finished, "finalizeOrder" is implemented again.

Every time a user needs to input an integer, the program calls a method called "validInput". This method takes two integers "low" and "high" as well as an array of strings. The method is supposed to make sure users only input an integer but does not work that way. There it two while loops, one that runs while the input is not an integer and the other while the integer is larger than "high" or smaller than "low". Each while loop outputs the available options from the inputted array of strings and asks the user to enter one of those numbers. The method returns an integer called "good" that the user has inputted.

The method "validDouble" works exactly the same as "validInput" but is used when a double value is needed. The method takes two double values "low" and "high". While the input is not a double the user is asked to enter a number between 0 and 10. The second while loop checks the user inputted double against the "low" and "high" and continually asks the user for a number that is between the two. The method returns a double called "good" that the user has inputted.

The method "printArray" is also called several times throughout the program. It takes an array of strings. It is used almost every time numbered options need to be displayed for the user and when the user's final order is displayed. It is called in every method except the main.

**Requirements**

To run the program you need the Java file named "Project2.java" and a computer with a working Terminal program and the latest version of Java installed. The program must be compiled and run through the Terminal using "javac Project2.java" and "java Project2".

**Literature Survey**

The only program that I am aware of that is semi-similar to mine is the Dunkin Donuts app. Their application allows for On-The-Go ordering, where users can select their order and submit for pickup at various Dunkin Donut locations. Users can select any items on their menu for pickup. When you are ordering a coffee drink the application allows for you to tap buttons selecting flavors, cream amount, and sugar amount. The difference between their program and mine is that users can specify down to the exact decimal for cream and sugar measurements. The way the Dunkin Donuts application works is there is only whole numbers. An issue I have come across many of the times is that my preferred creamer amount is between two whole numbers. If I ask for 2 it is too much, but 1 is not enough. Dunkin Donuts does not allow for measurements of decimals or half sizes, which would solve my issue. That is why my application allows for decimal input, so everyone's coffee is exactly how they like it.

**User Manual**

While running the program the user will be asked several times to input numbers. The program never asks for strings or characters. If a string or character is inputted the user will be asked again to input a number. Also if the number is outside of the options that have been presented, the user will also be asked again to input a number within the options. Currently the program only catches characters and strings until a number is inputted. If a string or character is put in again the program prints out an exception that the input is mismatch to what is needed to run the program. For example, if the user is currently in the "sizeSelecion" method they are presented with "1 – Small", "2 – Medium", and "3 – Large". If the user inputs "small" the program will spit out the number options and ask for one of them again. Then if the user inputs "5" the same thing will happen. If after that the user still does not cooperate and inputs "y" then the exception is outputted and the program ends.  For best user results the user should only input numbers from the options provided.

**Conclusion**

This program went through several iterations to come to the final version it is now.  While doing this project I continually found better ways to make my program run faster and smoother.  And while I am no expert, I think this project definitely expanded my programming skills.  I still need a little practice when it comes to avoiding user errors but I have learned how to efficiently store and output various types of information.