Lab Report

Introduction

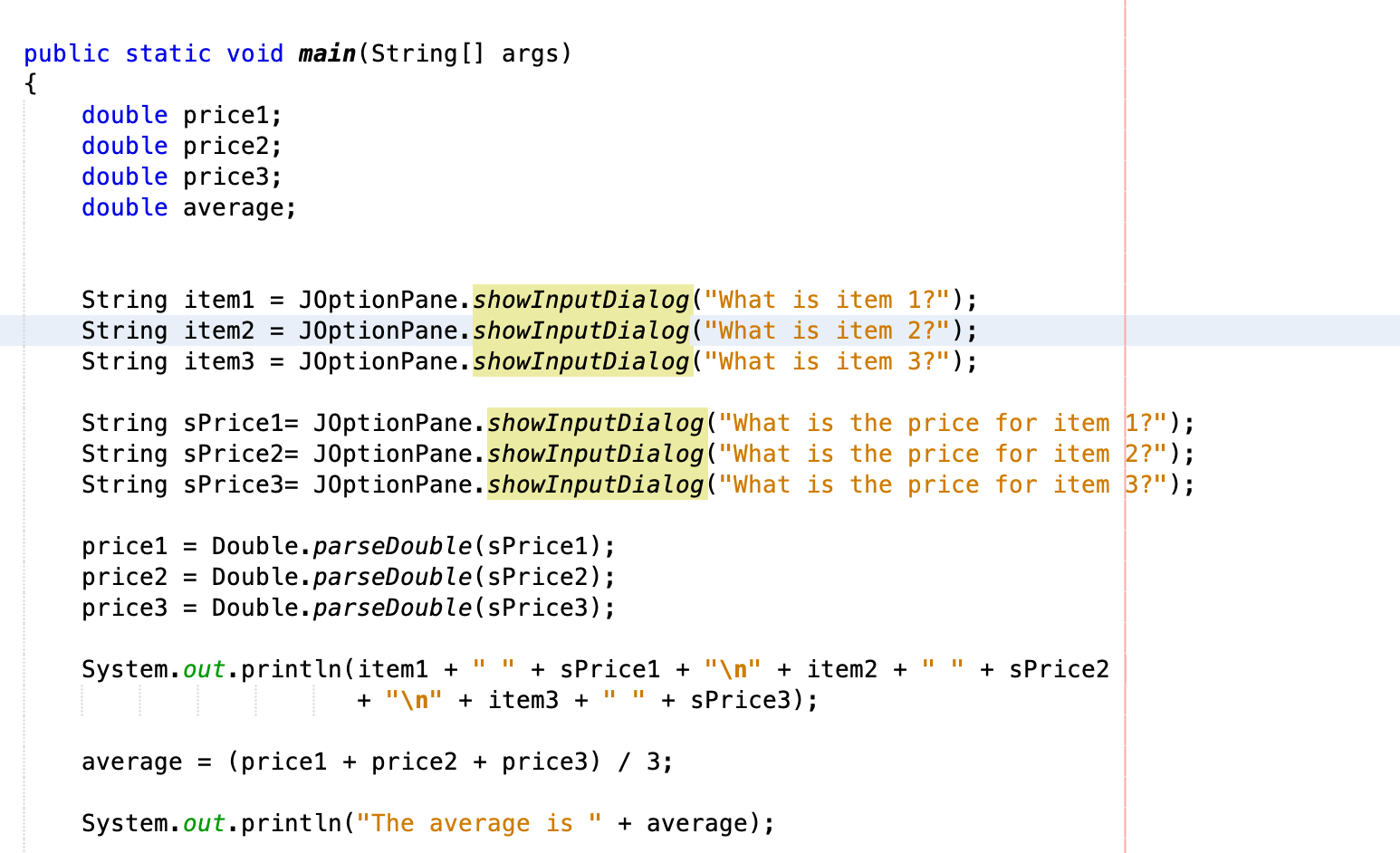
The purpose of this lab report is to allow the user to input something, allow the machine to accept the input, and display what was inputted as well as the average. This can be done multiple different ways. An algorithm can be used to allow for a quicker approach, while you can also use other methods, which may require some extra coding. This is beneficial to study because it shows some basic concepts of java and provides a gateway to more advanced java topics.

Program #1

Method and Implementation

The purpose of program #1 was to create an application that accepted three items and their prices as well as the average, and display it to the user. In our code, we first declared our variables. Then, we used JOptionpane windows to prompt the user to enter the three items and then prompt the user to enter the prices of the three items. Then we used system.out.println to output the three items along with their prices next to them. Lastly, we used a formula to take the average of all the prices, then use another system.out.println line of code to output the average.

Analysis and Time complexity:



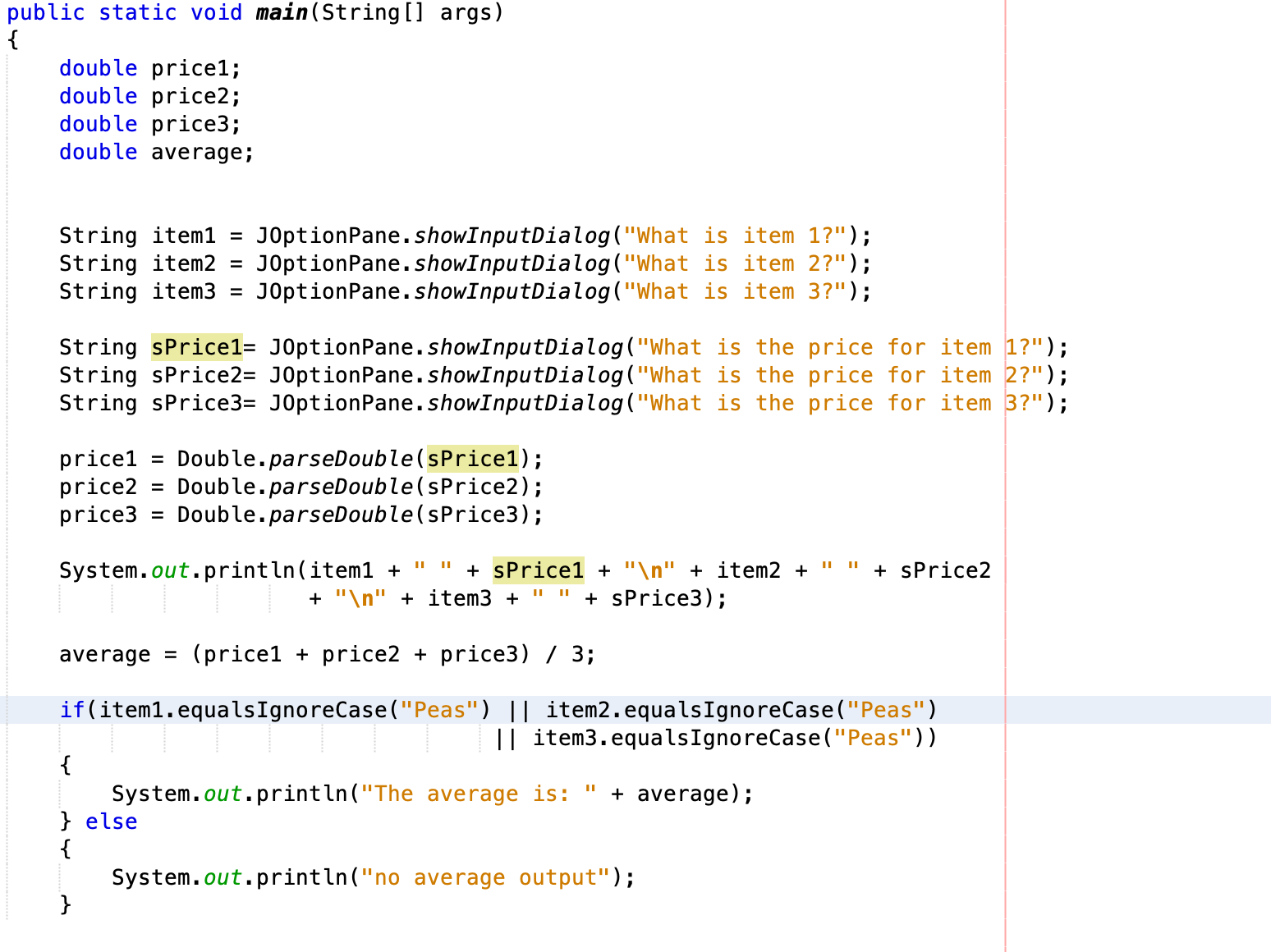
Every output is C2 and everything assigned is a C1. Every double declared is a C1 and every JOptionpane window declared is a C2. Every input made into those windows is a C1. All the parsed doubles are also C1’s. Both System.out.println messages are a C2 and the formula declaring an average is a C1. This makes the formula C1+ C1 +C1 +C1 + C2 + C1 + C2 + C1 +C2 +C1 + C2 + C1 + C2 + C1 + C2 + C1 + C1 + C1 + C1 + C2 + C1 + C2 = 14C1 + 8C2.

Program #2

Method and Implementation:

The purpose of program #2 is very similar to program #1, except this time we want it to display an output when the word “peas” is inputted (not case sensitive). In order to do this, the same code for the first program can be used, but we add an if-else statement to the end. The purpose of this if-else statement is to tell the computer to output an average when the word “peas” is entered. If the item entered does not equal “peas”, then it will display “no average output” instead. Inside the if statement we used double bars, || , to indicate that if any of the items equaled “peas”, print out the average. The double bars are the “or’s”.

Analysis and Time Complexity:



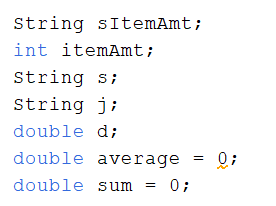
Just like in the previous program C2 is every output and C1 is everything assigned. Every double declared is a C1 and every JOptionpane window declared is a C2. Every input made into those windows is a C1. All the parsed doubles are also C1’s. The first system.out.println message is a C2 and the average formula is a C1. In the if statement, both system print messages are a C2. C1+ C1 +C1 +C1 + C2 + C1 + C2 + C1 +C2 +C1 + C2 + C1 + C2 + C1 + C2 + C1 + C1 + C1 + C1 + C2 + C1 + C2 + C2 = 13C1 + 9C2

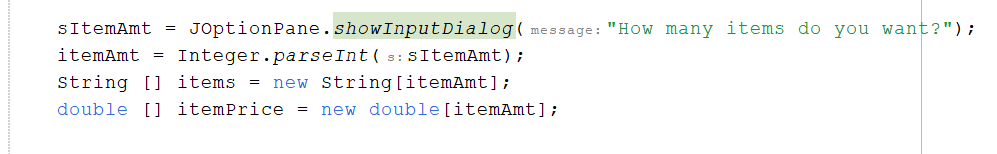
Program #3

Method and Implementation:

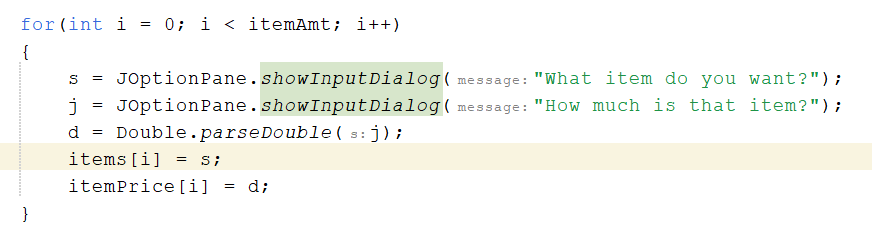
The purpose of program 3 was to accept a given number of items along with their prices and then output them in reverse order. Similar to program 2, we then have to output an average of the prices if the item “peas” is entered, and if “peas” is not detected then we output “no average output.”

First we started this problem off with declaring our necessary variables at the very top of our lines of code.

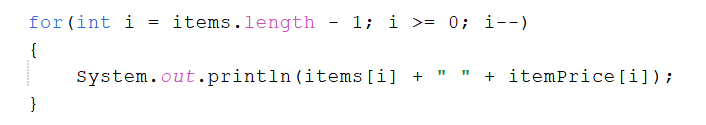




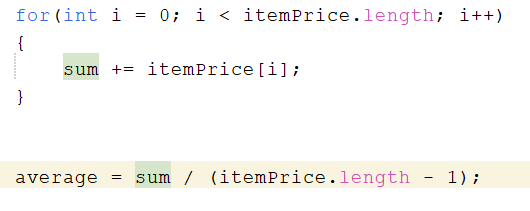
Along with declaring our variables, we set sItemAmt to a JOptionPane to grab how many items the user wanted. Then underneath, we parsed that string into a double so that when we use it later in our for loop, Java doesn’t hate us.

Our first of four total for loops was used to ask the user which item they wanted, and how much money that item cost. We would iterate that for loop x amount of times, x being the parsed string they entered in the first dialog input box. Once we asked which item they wanted, and how much that item cost using more JOptionPane, we would then grab the users data they enter and store in our array for items and prices at each “i” index in the array; after of course parsing the price into a double so when we calculate the sum later on it makes life easier. After each iteration, the for loop would increment up until it reaches the users desired “itemAmt” number (see code below).

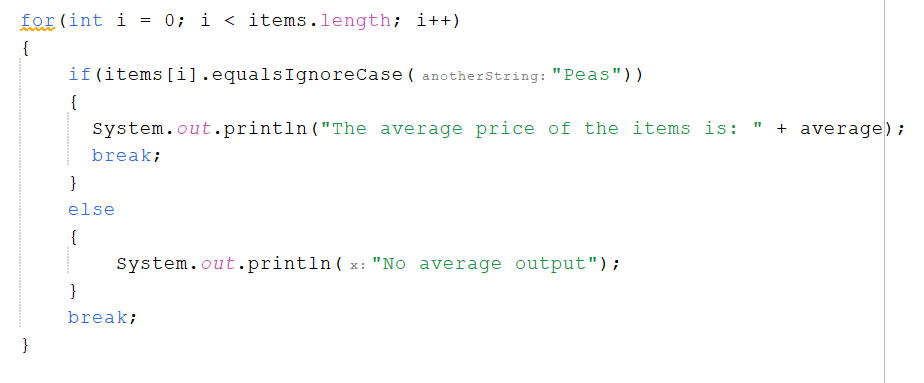
Our next for loop was used to print out each item and their corresponding price out to the console is reverse order. Here we started at the item arrays length but subtracted 1 to avoid an index out of bounds error. We would stop iterating when the length gets to 0, 0 being our first element, and then decrement by 1. Inside the for loop, a simple print statement would print out each item and price at each index of each array, starting at the highest index (the last item and price the user enters). This is how we print out the items and their prices in reverse order.



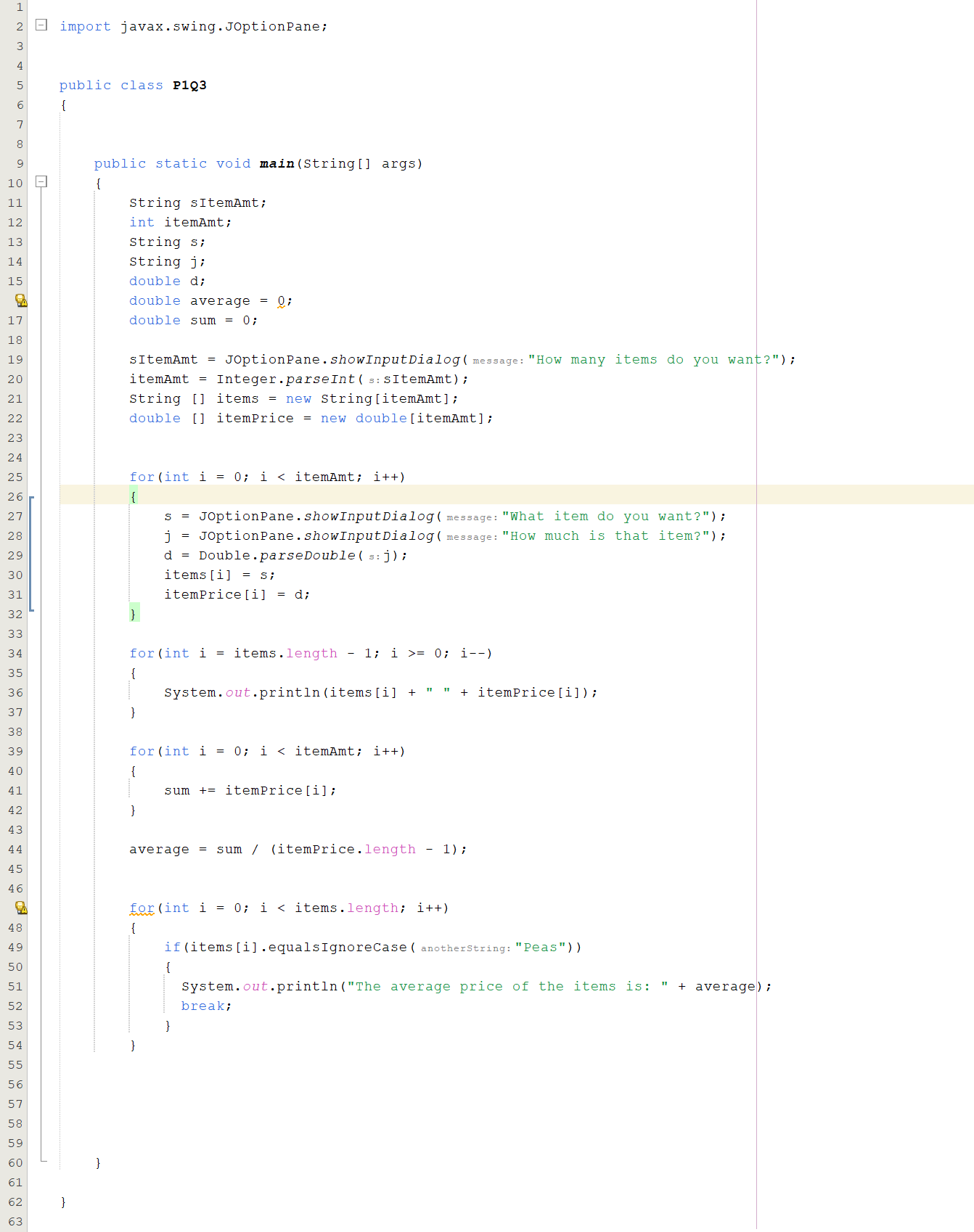
Our third for loop simply iterates the same amount of times as the length of our price array is. Through each iteration, it adds what's inside of that index up and sets that equal to sum, grabbing our total price. We then take our declared variable “average” and set that equal to the sum divided by the length of our itemPrice array.



Our very last for loop has a nested if statement that seemed tricky until we thought to add a break, to break out of the for loop if one index in our items array was equal to “peas”. This helped avoid the possible instance of the average being printed out over and over again if the user entered “peas” multiple times. We used this same concept after our else statement.



Analysis and Time Complexity:



In this program, all the variables declared in the beginning are a C1 including the strings, integer, and doubles. The following JOptionpane line of code is a C2. The item amount, string array, and double array are a C1. In the first for statement, both JOptionpane lines are a C2. The parse on the double is a C1 and both item arrays are a C1. In the second for statement, the system.out.println is a C2. In the third for statement, the two equations are both a C1. In the final for statement the system.out.println is a C2. The equation is C1 + C1 + C1 + C1 + C1 + C1 + C1 + C2 + C1 + C1 + C1 + C2 + C2 + C1 + C1 + C1 + C2 + C1 + C1 + C2 = 15C1 + 4C2

Conclusion

Time complexity is very important because it allows us to create and execute efficient programs. You would want a lower time complexity because a lower time complexity means that the algorithm will execute faster. Good algorithms in programs allow for fast time complexity, allowing for a faster and more efficient program. These three programs helped us get a better understanding of time complexity and what it actually is. These three programs each had different time complexities since each program had a different amount of code to run. Time complexity is very important to take into consideration when creating a program.