Core | Deadline: Sunday of Week 1 | Difficulty Level: Basic | Est. Time: 2 - 3 hrs

Cafe Business Logic

Now that you've familiarized yourself with Java a bit, you have decided to pursue an internship working on Cafe Java's dev team. After applying, you were sent descriptions of requested behavior for each of the methods needed for creating the `CafeUtil` library for their application, as well as some sample test cases.

Learning Objectives:

- Write methods to solve basic logic problems in Java
- Practice `Array` and `ArrayList`
- Practice `loops`
- Understand the difference between printing and returning a value from a method
- Create your own methods file
- Write a test file and create an instance of another class to test the functionality

---

CafeUtil File and TestCafe File

For this assignment you will have two files, `CafeUtil.java` and `TestCafe.java` . You will be writing all of your methods inside the `CafeUtil` class. The second file, `TestCafe.java` , is a test file that includes the `main` method. We have provided the code in its entirety for testing. You can copy and paste from it from the bottom of this module but be sure to instantiate your class in the testing file!

Assignment:

Implement the four methods below. Important: Code all of the following methods in your `CafeUtil.java` file.

int getStreakGoal()

Cafe Java wants to implement a reward system for customers who always buy more drinks than they did the week before. To calculate how many drinks they need after 10 weeks, write a method that ***adds every consecutive integer from 1 to 10 and returns the sum***. In other words, add 1 + 2 + 3 and so on up to 10 and return the result.

Test your code before moving on! Don't forget to make an instance of your `CafeUtil` class to use in the test file. The number printed should be 55.

Ninja Bonus: Add a parameter, `numWeeks` so that an admin can change the number from 10 to whatever they want.

double getOrderTotal(double[] prices)

Given an array of item prices from an order, add all of the prices in the array and return the total.

Don't forget to test your code! Find the test code for this method in `TestCafe.java` and uncomment it before you compile and run.

void displayMenu(ArrayList<String> menuItems)

Given an `ArrayList` of menu items (strings), print out each index and menu item.

Sample output given an `ArrayList` with the items `"drip coffee"`, `"cappuccino"`, `"latte"` and `"mocha"`:

```
0 drip coffee
1 cappuccino
2 latte
3 mocha
```

Recap of ArrayList Syntax:

```
String name = myArray.get(0); // to access an element in an ArrayList using an index
```

addCustomer(ArrayList<String> customers)

Inside this method:

1. Print this string to the console: *"Please enter your name:"*.
2. Add this line of code to get user input from the terminal and store it in the variable *username*:

```
String username = System.console().readLine();
```

3. Print the username to the console, saying "Hello, [username here]!"
4. Print "There are ___ people in front of you" using the number for how many people are ahead of them (how many items already in the array.)
5. Add the customer's name to the given customers list and print the list.
6. There is no need to return anything.

Recap of ArrayList Syntax:

```
myArray.add("Heidi"); // to add an item to an ArrayList
```

How to Test: Test Early and Test Often

Write and test one method at a time. A good strategy for developing software is to code only a little and test. As they say, "Test early and test often." Go ahead and use the following code in your TestCafe.java file to test your methods by ***uncommenting*** the method you are testing. Test one at a time.

Note: `//` are for single line comments and `/* ...comments... */` are for multi-line comments.

TestCafe.java

```
import java.util.ArrayList;
import java.util.Arrays;
public class TestCafe {
    public static void main(String[] args) {

    /*
      You will need add 1 line to this file to create an instance
      of the CafeUtil class.
      Hint: it will need to correspond with the variable name used below..
    */

        /* ============ App Test Cases ============= */

        System.out.println("\n----- Streak Goal Test -----");
        System.out.printf("Purchases needed by week 10: %s \n\n",
appTest.getStreakGoal());

        // System.out.println("----- Order Total Test-----");
        // double[] lineItems = {3.5, 1.5, 4.0, 4.5};
        // System.out.printf("Order total: %s \n\n",appTest.getOrderTotal(lineItems));

        // System.out.println("----- Display Menu Test-----");

        // ArrayList<String> menu = new ArrayList<String>();
        // menu.add("drip coffee");
        // menu.add("cappuccino");
        // menu.add("latte");
        // menu.add("mocha");
        // appTest.displayMenu(menu);

        // System.out.println("\n----- Add Customer Test-----");
        // ArrayList<String> customers = new ArrayList<String>();
        // // --- Test 4 times ---
        // for (int i = 0; i < 4; i++) {
        //     appTest.addCustomer(customers);
        //     System.out.println("\n");
        // }
    }
}
```

Testing Example

Note: The output *does not* need to be formatted exactly the same as the given examples, as long as all the information is provided.

Ninja Bonuses!

void printPriceChart(String product, double price, int maxQuantity)

Given a `product`, `price` and a `maxQuantity`, create a method that prints the cost for buying 1, then the price for buying 2, then for 3 and so on, up to the max. For example, if the `product` is "Columbian Coffee Grounds" with a `price` of 15.0 and `maxQuantity` of 3, you should print:

```
Columbian Coffee Grounds
1 - $15.00
2 - $30.00
3 - $45.00
```

Tip: You can use the escape character `\n` within your string for line breaks.

Ninja bonus 2: Format the prices as currency. Hint: https://docs.oracle.com/javase/tutorial/java/data/numberformat.html

Ninja Bonus 3: Take $0.50 more off the original price every time the quantity increases.

Example: Given a $2.00 price and 4 max, where the prices would normally be $2, $4, $6 and $8, the discount would yield $2, $3.50, $5, and $6.50, progressively taking off $0.50, then $1, then $1.50 from the group tag price:

```
1 - $2.00
2 - $3.50
3 - $5.00
4 - $6.50
```

boolean displayMenu(ArrayList<String> menuItems, ArrayList<Double> prices)

Let's overload the display menu! Given 2 arrays, an `ArrayList` of menu items (strings), and an `ArrayList` of prices (doubles), print a menu!

Be sure to check if the arrays are not the same size. If they are, immediately return false.

To print the menu, iterate from 0 to the last index. Each time through, print on the following on the same line:

- a.) The index, b.) The menu item at that index, and c.) The price at that index.

Finally, return true.

Sample output:

```
0 drip coffee -- $1.50
1 cappucino -- $3.50
2 latte -- $4.50
3 mocha -- $3.50
```

Ninja Bonus 4!

Make a method `addCustomers` in which a barista can enter multiple customers.

Hint: You can use a while loop and ask the user to type "q" when they are finished entering names.

- [ ] Create a CafeUtil class and a testing file.
- [ ] Implement the getStreakGoal method and test.
- [ ] Implement the getOrderTotal method and test.

Submit

17050028471111-TestCafe.zip

Successfully submitted on January 11, 2024

Previous

Next

Privacy Policy