Honor Code

Like all work for this class, the Emory Honor Code applies. You should do your own work on all problems, unless you are explicitly instructed otherwise. If you get stuck or have questions, ask your instructor or a TA for help. For EACH of the files you submit, be sure to put the appropriate honor code statement (as specified on the course syllabus) at the top of the file in comments. Otherwise, you will be assessed a 10 point deduction.

Preparation

1. Create a directory called hw9 inside your cs170 project directory to save your hw9 files.
   Copy all the files from folder hw9 in the shared directory to your work directory:
   ```
cp /home/cs170002/share/hw9/* ~/cs170/hw9/
```

Problem

1. CoffeeShop, TestCoffeeShop
   The purpose of this assignment is to practice objects by managing a Coffee Shop. This Coffee Shop contains a set of all the coffees they can make and a set of all customers.
   When you copy the files from the shared directory, you will notice that some classes have already been programmed for you:
   - **Class Coffee**: Defines the object Coffee. The properties of a coffee are name and price. It has an empty constructor, a constructor with 2 parameters, and a copy constructor. It also has all the get and set methods needed. Additionally, two other methods are provided, one to determine if two coffees are equal, and another one display information about the object coffee.
   - **Class Customer**: Defines the object Customer. The properties are the customer’s name, how many times she has visited, and her favorite coffee. Three different constructors are provided. This class also has all the get and set methods needed. Additionally, the following methods are provided: 1) a method to determine if two customers are the same, 2) a method to print information about the customer, 3) a method to determine if the customer is a member of a loyalty program (i.e. if the customer has come 3 or more times), and 4) a method to increase the customer’s visits.
The two classes that need programming are \texttt{CoffeeShop} and \texttt{TestBookStore}.

• \texttt{CoffeeShop}. This is the class that defines the object \texttt{CoffeeShop}. The properties of it are an array of objects of type \texttt{Coffee} and an array of objects of type \texttt{Customer}. Additionally, this class will need a counter for how many coffees we actually have, as well as a counter for how many customers we actually have. In other words, the array of all our coffees could be of size 100, but we might actually just have 3 coffees. This class will additionally require:

  – A constructor which, given two int parameters: \texttt{maxCoffeeCount} and \texttt{maxCustomerCount}, initializes the array of objects of type \texttt{Coffee} and the array of objects of type \texttt{Customer} of that given size, respectively. The constructor also sets both counters to zero. You can add an empty constructor or any other constructor if you want.

  – A void method \texttt{addCoffee}, which given a name (String) and a price (double), it adds a Coffee to the coffee array. The counter variable will be helpful for this. If the coffee cannot be added to the array (the array is full), then display an appropriate message.

  – A method, called \texttt{checkCustomers}, that returns an object of type \texttt{Customer}. Given a name (String), it looks in the array of Customers for a matching name (use the method \texttt{getName()}). If it finds it (meaning that the customer had been there before), return that customer from the array. Otherwise, create a new customer (use the constructor that only needs a name) and place her in the Customer’s array. The counter of customers will be helpful to know where to place her in the array. If the customer cannot be added, display an appropriate message. Once you place the customer in the array, increment the counter. Return this new customer.

  – A void method \texttt{sale}, which given the name of a customer (String), and the name of a coffee, checks if the customer has been there before (use \texttt{checkCustomers}). If this person is a loyal member, greet her differently. Now, check if the coffee that the user is asking for is in our Coffee array. If it is, display an appropriate message for the sale, increase the customer’s visits count, and set her favoriteCoffee as this coffee. If we don’t have that coffee, display an appropriate message.
– A String method `toString` which returns a string with all the appropriate information: All the coffees that the CoffeeShop has, and all the Customers that the CoffeeShop has. I would suggest this to be one of the first methods that you program, so that you can know if you are properly adding coffees and customers to the appropriate arrays.

- **TestCoffeeShop.** This class will test how your CoffeeShop was programmed. A template has been made for you, which will be useful, but it doesn’t test all the cases. Modify this class as you need to test CoffeeShop.java. This class will not be turned in, but class CoffeeShop.java will be tested with several cases (i.e. we’ll have our own TestCoffeeShop.java).

A sample output for the existing `TestCoffeeShop.java` is shown below:

```
Hi, Larry. Welcome to your visit number: 1
Thanks for shopping. The price for Cappuccino is 5.0

Hi, Larry. Welcome to your visit number: 2
Thanks for shopping. The price for Cappuccino is 5.0

Hi, Larry. Welcome to your visit number: 3
Thanks for shopping. The price for Cappuccino is 5.0

Hi, Adam. Welcome to your visit number: 1
Thanks for shopping. The price for Cappuccino is 5.0

Hi, Adam. Welcome to your visit number: 2
Thanks for shopping. The price for Espresso is 3.2

Hi, Paul. Welcome to your visit number: 1
Sorry, we don’t have that coffee

Hi, Paul. Welcome to your visit number: 1
Thanks for shopping. The price for Mocha is 5.5

Hi, David. Welcome to your visit number: 1
Thanks for shopping. The price for Mocha is 5.5

Hi, David. Welcome to your visit number: 2
```
Thanks for shopping. The price for Cappuccino is 5.0

Hi, David. Welcome to your visit number: 3
Thanks for shopping. The price for Espresso Macchiato is 4.2

These are all the available coffees:
Coffee: Cappuccino Price: 5.0
Coffee: Mocha Price: 5.5
Coffee: Espresso Price: 3.2
Coffee: Espresso Macchiato Price: 4.2

These are all our customers:
Customer: Larry Has Come: 3 times Likes: Cappuccino
Customer: Adam Has Come: 2 times Likes: Espresso
Customer: Paul Has Come: 1 times Likes: Mocha
Customer: David Has Come: 3 times Likes: Espresso Macchiato

Grading

Your grade will be determined based on the correctness of your programs as well as program style. Program style includes such things as comments, variable/method names, and readability. Late penalties apply as per the class syllabus.

Submission

- When you finish, you should have the file CoffeeShop.java in your /cs170/hw9:

- Be sure to include the Honor Code statement at the top of the file you submit.

- Only your last submission will be graded.

- Submit your work using the following commands. You need to be in your /cs170/hw9 directory when you issue it:

  $/home/cs170002/turnin-hw CoffeeShop.java hw9

- Your homework is not turned in unless the above commands are successful (you will get a ”success” message when turn in was successful).