**CS 342** **Homework #5** **Fall 2020**

# Coffee Shop FXML and CSS

## Due Date: Thursday 11/19/2020, by 11:59PM

## Description:

In this homework, you will replicate the GUI you made for homework #4 utilizing FXML and CSS style sheets. You can use the fxml sample maven project on Blackboard for your template or your actual submission for homework #4. You must have at least one of each: a controller class, a .fxml file and a .css file. The user interface must render only using this approach. The appropriate GUI elements must be clickable and have methods that are called when the user interacts with it. These methods are defined in the controller class and attached to the widgets in the .fxml files.
While you can submit a working version of homework #4 that creates and manages the GUI with fxml instead of programmatically, **we will not be grading the functionality of homework #4 again**. When widgets are clicked, you can simply have the methods attached print something to the terminal. **We will only concentrate on the successful creation of a user interface using fxml and css.**
I have included the write up for homework #4 below for your reference.

## Homework #4: This was the write up for homework #4

## Description:

In class, we will see how to use the Decorator design pattern to build an order for coffee; decorating the basic black coffee with cream, sugar and extra shots. In this homework, you will take the code shown in class and add two more items that can be additions to a coffee order. You will then create a JavaFX program, utilizing the decorator design pattern included, that provides the user interface to make coffee orders and display them. As a reminder, **ALL HOMEWORKS ARE INDIVIDUAL ASSIGNMENTS.**

## Implementation Details:

You will create a maven project, including unit tests, using the template provided for this assignment.

# CS 342 Homework #5 Fall 2020

The GUI:

You must include a way to start a new order, delete an order, order each additional item and display the order and total cost when the order is complete. Once the order is complete, you must display the entire order including the cost of each item, the add ons and total cost of the order. For example:

Black Coffee: $3.99
+ extra shot: $1.20
+ cream: $.50
+ sugar: $.50
Total: 6.19

The user must be able to build another order after each order is completed. You must also create some kind of color/design scheme for you app, it can not just be the defaults. Otherwise, you are free to be creative with your user interface.

The Code:

Your orders must be built utilizing the design pattern code included. For example, if I wanted to order a coffee with an extra shot, cream and sugar, it would be built like this:

**Coffee order = new Sugar(new Cream( new ExtraShot(new BasicCoffee())));**

For this HW, it is assumed that every coffee order will start with **BasicCoffee.** You do not need to include functionality to remove certain items once they are added. The user can just delete the order and start again. You must add two more “add ons” for a basic coffee. This will require two new classes that follow the same construction as the Cream, Sugar and ExtraShot classes.

Hint 1:

You will want to utilize a separate class to control the building of the orders. This class could have a data member (Coffee order) and methods that add items to the order (order = new Cream(order);). You could initialize the data member order to a BasicCoffee in the constructor since each order starts with that.

Hint 2:

Remember nested classes share data members with the enclosing class. You do not need to keep all the classes in separate files. You may also add code to the existing files if need be but not remove any code that already exists.

# CS 342 Homework #5 Fall 2020

## Electronic Submission:

Put the Maven template folder with your files in a .zip and name it with your netid + Homework5: for example, I would have a submission called mhalle5Homework5.zip, and submit it to the link on Blackboard course website.

## Assignment Details:

Late work on a homework is **NOT ACCEPTED.** Anything past the deadline will result in a zero.

**We will test all homework on the command line using Maven 3.6.3. You may develop in any IDE you chose but make sure your homework can be run on the command line using Maven commands. Any homework that does not run will result in a zero. If you are unsure about using Maven, come see your TA or Professor.**

Unless stated otherwise, all work submitted for grading *must* be done individually. While we encourage you to talk to your peers and learn from them, this interaction must be superficial with regards to all work submitted for grading. This means you *cannot* work in teams, you cannot work side-by-side, you cannot submit someone else's work (partial or complete) as your own. The University's policy is available here:

https://dos.uic.edu/conductforstudents.shtml.

In particular, note that you are guilty of academic dishonesty if you extend or receive any kind of unauthorized assistance. Absolutely no transfer of program code between students is permitted (paper or electronic), and you may not solicit code from family, friends, or online forums. Other examples of academic dishonesty include emailing

your program to another student, copying-pasting code from the internet, working in a group on a homework assignment, and allowing a tutor, TA, or another individual to write an answer for you. It is also considered academic dishonesty if you click someone else's iClicker with the intent of answering for that student, whether for a quiz, exam, or class participation. Academic dishonesty is unacceptable, and penalties range from a letter grade drop to expulsion from the university; cases are handled via the official student conduct process described at https://dos.uic.edu/conductforstudents.shtml.