CS 342 **Homework #2** **Fall 2019**

Build A Burger

Due Date: Sunday October 6th 2019, by 11:59PM

Description:

I hate it when you go to order a cheeseburger and all the good toppings cost extra! However, it can be a good example of the Decorator Design Pattern. Using the Decorator Design pattern, create a java program that will allow the user to create a cheeseburger with extra toppings and print out the order and total cost. The burger in question could be made of ground beef, fish or plant based like the very delicious Beyond Burger. In the next homework, you will be able to chose the type of burger using the Factory Design Pattern. But for now, you may assume it is of any type you choose.

Implementation Details:

You will create a maven project, including unit tests, using the template provided for this assignment. Each class and interface will be in a separate file. You will create the interface, base class, abstract decorator class and five decorator classes. The interface will be named Burger and should contain a single abstract method: public double makeBurger().

Your base class should be called BasicBurger.

The names of the five decorator classes are: GrilledOnions, FriedEgg, Bacon, Avocado and RoastedPeppers.

The basic burger will cost $6.50 and the cost of the five extra toppings a customer can pay for will be $2.00 each.

If you ran your program in main by building different burgers and printing the results; for example:

```
Burger order = new Avocado(new FriedEgg(new GrilledOnions(new BasicBurger())));

double cost = order.makeBurger();
System.out.println("Total: $" + cost);
```

Will result in the following output:

```
Basic Cheeseburger $6.50
+ Grilled Onions $2.00
+ Fried egg $2.00
+ Avocado $2.00
Total: $12.50
```

Test Cases:

You must include a minimum of 10 unit tests in the DecoratorTest.java file provided in the Maven template project. These must run with the maven command “test”.

Electronic Submission:

Put the Maven template folder with your files in a .zip and name it with your netid + Decorator: for example, I would have a submission called mhalle5Decorator.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 3.6.1. 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.