Homework 6: Writing Classes
Due Thursday, October 21st, 9:45 AM

This assignment asks you to write an object class to support a driver class maintaining the desired interface between it and the driver class. Be sure to read this entire assignment through carefully and complete all of the steps in order. Your submission should adhere to all of the course style guidelines and should be submitted electronically.

Given the driver class GrilledCheese, write an object class Order to support the functionality implied. Your object class should assume that it takes seven minutes to make a single grilled cheese sandwich, and that grilled cheese sandwiches are made in serial, not in parallel. All times in GrilledCheese and Order should be given in 24-hour format.

The output of running the driver class should follow the behaviors specified in the comments and be exactly:

Order 76: 2 grilled cheeses to be done at 10:56
Order 92: 6 grilled cheeses to be done at 00:27

Adding sandwich to order 76...
Order 76: 3 grilled cheeses to be done at 11:03

Removing sandwich from order 92...
Order 92: 5 grilled cheeses to be done at 00:20

The ratio of duration of Alice's order to duration of Bob's order is 0.6.

1. Begin by declaring instance data for the object and by writing a constructor for the Order class. First, consider what information should be managed by the object. Look through the driver class to see what information gets passed through to the object in the constructor upon instantiation. Consider how this information is used by the object class, then declare private instance data to keep track of the information used by other methods. Declare a constructor that matches the format of the constructor assumed by the driver class and assign the argument values to instance data appropriately.

2. Now write the beginning of a toString() method for your object. At this point, don’t worry about calculating the end time of the order. You should just be concerned with overriding the toString() method successfully. At this point, outputting the orders should look like this:

   Order 92: 6 grilled cheeses

3. Next, write the addSandwich() and deleteSandwich() methods. Declare the methods in the object class. Adding a sandwich should increase the size of the order by one; likewise, deleting a sandwich should decrease the size of the order by one. Changing the size of the order does not affect the start time of the order. Also, write a method that allows the user to retrieve the customer’s order number.

4. Now, write a simple method that returns the calculated duration in minutes of the order. Remember that every sandwich in an order takes no more nor less than seven minutes to cook, and that it may be helpful to declare some static value in your object class to handle this time constraint.
5. Finally modify the `toString()` method you wrote to calculate the finish time for the order based off of a twenty-four hour clock. Your finished method should handle wrap around from minutes to hours and hours to days. It may be helpful to break this step into a few parts:

a. Calculate the minutes portion of the end time, neglecting wrap around after sixty minutes. For example one line of the output might read:

   **Order 92: 6 grilled cheeses to be done at 23:87**

b. Consider next how you might go about wrapping around after sixty minutes. Do not be concerned about incrementing the hour value quite yet.

   **Order 92: 6 grilled cheeses to be done at 23:27**

c. Next decide how to represent whether or not the finish time occurs in a different hour than the starting time. Change the hour portion of the end time accordingly. Don’t worry quite yet about wrap around from one day to the next.

   **Order 92: 6 grilled cheeses to be done at 24:27**

d. Now update the hour value of the time so that it wraps around after 24 hours.

   **Order 92: 6 grilled cheeses to be done at 0:27**

e. Finally, format your output so that both the hours and the minutes are represented as two digit numbers.

   **Order 92: 6 grilled cheeses to be done at 00:27**

Keep in mind that it may be helpful to comment out several lines of `GrilledCheese.java` in order to test and debug the methods you’ve written. Feel free to test these methods, especially the `toString()` method, in separate driver classes. Remember, that your final object class must suit the functionality implied by `GrilledCheese.java`, and that the `GrilledCheese` class should not be modified in any way.