TCSS143 Fundamentals of Object-Oriented Programming
Theory and Application
Programming Assignment #8
15 Points + 5 (Extra Credit)

This assignment will give you practice with classes.

Write a class called **Time** that can be used to store a time in hours, minutes and seconds. It should have the following public methods:

Time()	A constructor that takes no parameters and that constructs an object representing 0 time.
Time(int, int, int)	A constructor that takes three integers (hours, minutes, seconds) and that constructs a Time object storing that time. Throw an IlegalArgumentException if any of the parameters are negative.
void add(int, int, int)	A method which takes a time as three integers (hours, minutes, seconds) and that adds the time to the current time. Throw an IlegalArgumentException if any of the parameters are negative.
double getTime()	A method which returns the time as the real-valued number of hours with a decimal (as in 9.53).
String toString()	toString which returns a string representation of the time (hh:mm:ss) When converted to a string, the seconds and minutes should always be reported as being in the range of 0 to 59. That means that you may have to "carry" a full hour or minute to the next column.

For example, consider the following code:

```
Time t = new Time(3, 45, 15);
System.out.println(t + " equals " + t.getTime());
t.add(2, 30, 55);
System.out.println(t + " equals " + t.getTime());
t.add(0, 55, 45);
System.out.println(t + " equals " + t.getTime());
```

This code creates a Time object and adds three different times to it, each time reporting the time as a string and as a double. The output should be:

3:45:15 equals 3.754166666666667
6:16:10 equals 6.269444444444445
7:11:55 equals 7.198611111111111

Notice that the second time is not reported as 5 hours, 75 minutes and 70 seconds, even though that's what you'd get by a simple addition of values. Use military standard time which means time starts at 00:00:00 and ends at 23:59:59. ***You cannot create any additional methods other than those mentioned above***. Write your solution in **Time.java** and your main method in **TimeTest.java.** You must test every method from Time class in TimeTest.java.

(Extra Credit, 5 points) Write a class named `GroceryList` that represents a person's list of items to buy from the market, and another class named `GroceryItemOrder` that represents a request to purchase a particular item in a given quantity (example: 4 boxes of cookies).

The GroceryList class should use an array field to store the grocery items, as well as keeping track of its size (number of items in the list so far). Assume that a grocery list will have no more than 10 items. A GroceryList object should have the following methods:

public GroceryList()

Constructs a new empty grocery list.

public void add(GroceryItemOrder item)

Adds the given item order to this list, if the list is not full (has fewer than 10 items).

public double getTotalCost()

Returns the total sum cost of all grocery item orders in this list.

The GroceryItemOrder class should store an item quantity and price per unit. A

GroceryItemOrder object should have the following methods:

public GroceryItemOrder(String name, int quantity, double pricePerUnit)

Constructs an item order to purchase the item with the given name, in the given quantity, which costs the given price per unit. Throw an IllegalArgumentException if quantity is negative or pricePerUnit is negative or if name is empty.

public double getCost()

Returns the total cost of this item in its given quantity. For example, 4 boxes of cookies that are 2.30 per unit have a cost of 9.20.

public void setQuantity(int quantity)

Sets this grocery item's quantity to be the given value. Throw an IllegalArgumentException if quantity is negative

You cannot create any additional methods other than those mentioned above. Write your solution in **GroceryItemOrder.java** and **GroceryList.java** and your corresponding main method in

GroceryItemOrderTest.java and **GroceryListTest.java.** You must test every method from your classes in your Test classes.

Submission and Grading:

There will be points taken off for not following the conventions listed in this document regarding submissions, outputs and naming conventions.

You are required to properly indent your code and will lose points if you make significant indentation mistakes. See the coding conventions document on the course web page for an explanation and examples of proper indentation.

Give meaningful names to methods and variables in your code. Localize variables whenever possible -- that is, declare them in the smallest scope in which they are needed.

Include a comment at the beginning of your program with basic information and a description of the program **and include a comment at the start of each method**. Your comments should be written in your own words and not taken directly from this document.

You should include a comment at the beginning of your program (for each class) with some basic information and a description of the program, as in:

```
// Menaka Abraham
// 1/28/16
// TCSS 143
// Assignment #8
//
// This program will...
```

Your files **Time.java, TimeTest.java, (if attempting extra credit - GroceryItemOrder.java, GroceryItemOrderTest.java, GroceryList.java, GroceryListTest.java)** must be submitted on the course web page.