CECS 475
Lab Assignment 3
Assigned date: 9/4
Due date: Monday 9/18
50 points

Problem 1 [10 points] - Due Wed 9/11

Read the following lecture note

Delegates and Events pdf word
Code example word

and implement the following:

a. Delegate event without passing data
b. Delegate event with passing data
c. Rewrite item a using .NET EventHandler
d. Rewrite item b using .NET EventHandler

Grading

- Demonstrate the result to the instructor in the class
- Return a printed copy of the answers to the following questions:
    1. Briefly describe how you convert part a to c by showing the code that is modified for using .NET event handler.
    2. Briefly describe how you convert part a to c by showing the code that is modified for using .NET event handler.

Problem 2 [40 points] - Due Wednesday 9/18

In this lab assignment , you create an application that models the ups and downs of a particular stock. The value of a stock is assumed to change by plus or minus a specified number of units after every time unit (such as one hour). A notification is generated each time a stock changes more than a specified number of units above or below its intial value. A collection of brokers who control the stock must receive this notification. The range within which the stock can change every time unit and the threshold above or below which collection of brokers who control the stock must be notified are specified when the stock is created (using its constructor).

You shall design and implement a C# application that satisfies the specification given above. This application involves delegates, events and threads. You begin by defining a delegate class StockNotification. This is shown below.

public delegate void StockNotification(String stockName, int currentValue, int numberChanges);

This delegate is designed so that when an event of this type is fired, the stock's name, value, and the number of changes in value can be sent to the listener.

Create the class *Stock* with the following attributes:

- Stock name
- Stock intial value
- Maximum change (the range within a stock can change every time unit)
- Notification threshold (the threshold above or below which the collection of brokers who control the stock must be notified)

You are required to implement other members in the class *Stock* that are needed. When a stock object is created, a thread is started. This thread causes the stock's value to be modified every 500 milliseconds. If its value changes from its initial value by more than the specified notification threshold, an event method is invoked. This invokes the *stockEvent* (of event-type StockNotification) and multicasts a notification to all listeners who have registered with *stockEvent*.

Create another event to notify saving the following information to a file when the stock's threshold is reached: date and time, stock name, inital value and current value.

Create the class *StockBroker* which has fields broker name and stocks, a List of Stock. This latter field is not used in this application but could be used to obtain the stocks currently controlled by a given broker. The *addStock* method registers the *Notify* listerner with the stock (in addition to adding it to the list of stocks held by the broker). This *Notify* method outputs to the console the name, value, and the number of changes of the stock whose value is out of the range given the stock's notification threshold.

The listing below presents a main driver class, StockApplication.

```
static void Main(string[] args)
{
Stock stock1 = new Stock("Technology",160, 5, 15);
Stock stock2 = new Stock("Retail",30, 2, 6);
```

```
Stock stock3 = new Stock("Banking", 90, 4, 10);
Stock stock4 = new Stock("Commodity", 500, 20, 50);

StockBroker b1 = new StockBroker("Broker 1");
b1.AddStock(stock1);
b1.AddStock(stock2);

StockBroker b2 = new StockBroker("Broker 2");
b2.AddStock(stock1);
b2.AddStock(stock3);
b2.AddStock(stock4);

StockBroker b3 = new StockBroker("Broker 3");
b3.AddStock(stock1);
b3.AddStock(stock3);

StockBroker b4 = new StockBroker("Broker 4");
b4.AddStock(stock1);
b4.AddStock(stock2);
b4.AddStock(stock3);
b4.AddStock(stock4);
}
```

The output sample:

```
Broker          Stock           Value           Changes

Broker 2        Commodity       553             4
Broker 4        Commodity       553             4
Broker 1        Technology      177             5
Broker 2        Technology      177             5
Broker 3        Technology      177             5
Broker 4        Technology      177             5
Broker 2        Banking         103             5
Broker 2        Commodity       571             5
Broker 4        Commodity       571             5
Broker 3        Banking         103             5
Broker 4        Banking         103             5
Broker 1        Technology      179             6
Broker 2        Technology      179             6
Broker 3        Technology      179             6
Broker 4        Technology      179             6
Broker 2        Banking         105             6
Broker 3        Banking         105             6
Broker 4        Banking         105             6
Broker 2        Commodity       581             6
Broker 4        Commodity       581             6
```

```
Broker 1        Technology      181             7
Broker 2        Technology      181             7
Broker 1        Retail          37              7
Broker 4        Retail          37              7
Broker 2        Banking         107             7
Broker 3        Banking         107             7
Broker 4        Banking         107             7
Broker 2        Commodity       588             7
Broker 4        Commodity       588             7
Broker 3        Technology      181             7
Broker 4        Technology      181             7
Broker 2        Banking         108             8
Broker 1        Retail          38              8
Broker 2        Commodity       589             8
Broker 1        Technology      182             8
```

Resources for lab 3

1. Lecture note on lab 3 (pdf)
2. Multithreading in C#
3. Thread synchronization
4. Write to a text file in C#