GIVE IT A TRY #1

* Start with the code found in Exercise Starters\ ⮨  
  01\_EmployeeManager.
* This program will build a collection of employee records. You will be implementing sort logic for employee names and IDs. Then you will be able to test this with a delegate.
* Locate the // TODO comments throughout the code and provide implementation for each.
* How to view the // TODO comments:
  + For VS 2008, open the View menu then click Task List (alternatively you can press CTRL + \ then CTRL + T). The task list should appear with the // TODO comments. If it’s not docked, you can dock the window.

GIVE IT A TRY #2

* Create a console application that will simulate a timer.
  + Create a class called TimerEventArgs that inherits from EventArgs. Have the constructor accept an int and store it. Provide a read-only property to return the stored value.
  + Create a delegate called UpdateClockHandler with parameters of object and TimerEventArgs.
  + Create a class called ConsoleClock that has two members: an internal event called OnUpdateClock of type UpdateClockHandler and a private int called counter.
  + Create a method in ConsoleClock called RunTheClock(). It has a loop that will check if counter has reached 100. In the loop, sleep for 1 second using System.Threading.Thread.Sleep (1000). Check to see that there are subscribers to the event. Create a new TimerEventArgs object passing in the counter and then signal the event. Increment the counter by 1.
  + Create a Tester class with a method called UpdateClockHandler() that takes an object and a TimerEventArgs parameter. Display the counter value followed by the current time using DateTime.Now.
  + In Main() create a new Tester object and a new ConsoleClock object. Subscribe to the event and then call RunTheClock().
* What other handlers could be used for the timer?

EXERCISE #1:

|  |  |
| --- | --- |
| WHAT TO DO | HOW TO DO IT |
| Create an application that will use a delegate to perform two different types of calculations. | Create a class called Computer. This class will have a constructor in it that will take and int and save it in a private member variable.  The Computer class will contain a public delegate called ComputeAnswer that will take an int argument and return a double.  The Computer class will also contain a public method called GetAnswer() which will take a single argument which is the delegate and return void. This method will call the delegate and display the computed answer when the delegate returns.  Create a class called FactorialNumbers. It will contain a single method in it called ComputeValue() that will take an int argument and return a double. This method will compute the factorial using the int passed in. A factorial of 4 is 4! which equals 1 \* 2 \* 3 \* 4 or 24. A factorial of 0 equals 0. If the number passed in is less than 0, return a -1.  Create a class called SquareNumbers. It will contain a single method in it called ComputeValue() that will take an integer argument and return a double. This method will compute the square of the number passed in.  Create a class called TestClass that contains the Main() method. First, prompt the user for a number and be sure that the data they supply is valid. Then, create a Computer object and supply it with the user’s number. Next, create two objects, one of type FactorialNumber and one of type SquareNumber. Create two delegates, one for each of the ComputeValue methods, and then call the GetAnswer() method in the Computer class to perform the calculations.  As a bonus, think about other types of calculations that can be performed on a single number (like absolute value, factors that when multiplied equal that number, etc.). Add these in and use the delegates to call these other methods. |

EXERCISE #2:

|  |  |
| --- | --- |
| WHAT TO DO | HOW TO DO IT |
| Create an application that will write messages to the console and to a log file (which will also go to the console) whenever a timer raises and event. | Create a class called TimerEventArgs. It will contain a constructor that will get the current time from the system and save it in a local variable. It will also have a public read-only property that will format the time into HH:MM:SS.mmm and return this new format as a string.  Create a class called CustomTimerClass. It will contain a public delegate called TimerAlarmHandler and a public event called OnTimerAlarm. It will also contain a private member variable called stopFlag of type bool and set it to false.  The CustomTimerClass will contain three methods. The first is a method called Run() that will sleep for 1 second and then call a second method to raise an event. The second method called RaiseTimerAlarm() will check to see if there are any subscribers to the event. If so, it will create a new TimerEventArgs object and raise the event. The final method is actually a read/write property called StopFlag the sets the stopFlag member or returns its value.  Create a class called DisplayMessageClass. It will have two member variables. The first is a private reference to the CustomTimerClass and the second is a static counter of type int. Also, create a constructor that accepts a CustomTimerClass argument and save that argument in the private member variable. Finally, create a method called DisplayMessage that takes two arguments – a sender object and a TimerEventArgs reference and displays the information from the TimerEventArgs object. Increment the counter and then check it. If the counter is >= 0, set the StopFlag property of the CustomTimerClass object to true.  Create a class called LogMessageClass. It will have one member variable, a private reference to the CustomTimerClass object. Create a constructor that takes a CustomTimerClass argument and saves that argument in the private member variable. Create a method called LogMessage() that takes two arguments – a sender object and a TimerEventArgs reference and displays the information from the TimerEventArgs object.  Create a class called TestClass that will contain the Main() method. Create an object of type CustomTimerClass. Then create an object of type DisplayMessageClass and an object of type LogMessageClass. Call the Run() method in the CustomTimerClass object. |