```
// Program.cs
using Stock;

Stock.Stock stock1 = new Stock.Stock("Technology", 160, 5, 15);
Stock.Stock stock2 = new Stock.Stock("Retail", 30, 2, 6);
Stock.Stock stock3 = new Stock.Stock("Banking", 90, 4, 10);
Stock.Stock stock4 = new Stock.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);

// Stock.cs
using System.Collections.Generic;
using System.Reflection.Metadata.Ecma335;
using System.Text;
using System.Threading;


namespace Stock
{
  //------------------------------------------------------------------------------
  public class Stock
  {
    public event EventHandler<StockNotification> StockEvent;
    //Name of our stock.
    private string _name;
    //Starting value of the stock.
    private int _initialValue;
    //Max change of the stock that is possible.
    private int _maxChange;
    //Threshold value where we notify subscribers to the event.
    private int _threshold;
```

```
//Amount of changes the stock goes through.
private int _numChanges;
//Current value of the stock.
private int _currentValue;
private readonly Thread _thread;
public string StockName { get => _name; set => _name = value; }
public int InitialValue { get => _initialValue; set => _initialValue = value; }
public int CurrentValue { get => _currentValue; set => _currentValue = value; }
public int MaxChange { get => _maxChange; set => _maxChange = value; }
public int Threshold { get => _threshold; set => _threshold = value; }
public int NumChanges { get => _numChanges; set => _numChanges = value; }

//---------------------------------------------------------------------------
/// <summary>
/// Stock class that contains all the information and changes of the stock
/// </summary>
/// <param name="name">Stock name</param>
/// <param name="startingValue">Starting stock value</param>
/// <param name="maxChange">The max value change of the stock</param>
/// <param name="threshold">The range for the stock</param>
public Stock(string name, int startingValue, int maxChange, int threshold)
{
  _name = name;
  _initialValue = startingValue;
  _currentValue = InitialValue;
  _maxChange = maxChange;
  _threshold = threshold;
  _thread = new Thread(new ThreadStart(Activate));
  _thread.Start();
}
//-----------------------------------------------------------------------------
/// <summary>
/// Activates the threads synchronizations
/// </summary>
public void Activate()
{
  for (int i = 0; i < 25; i++)
  {
    Thread.Sleep(500); // 1/2 second
    ChangeStockValue();
  }
}
//-------------------------------------------------------------------------------
// delegate
```

```
    //public delegate void StockNotification(String stockName, int currentValue, int
numberChanges);
    // event
    //public event StockNotification ProcessComplete;
    //-----------------------------------------------------------------------------
    /// <summary>
    /// Changes the stock value and also raising the event of stock value changes
    /// </summary>
    public void ChangeStockValue()
    {
      var rand = new Random();
      CurrentValue += rand.Next(1, MaxChange);
      NumChanges++;

      if ((CurrentValue - InitialValue) > Threshold)
      {
        StockEvent?.Invoke(this, new StockNotification(StockName, CurrentValue, NumChanges));
      }
    }
    //-----------------------------------------------------------------------------------
  }
}

// StockBroker.cs
using System;
using System.Collections.Generic;
using System.Text;
using System.IO;
using System.Threading;

namespace Stock
{
  public class StockBroker
  {
    public string BrokerName { get; set; }
    // The broker holds a list of Stocks (though not strictly needed in this

    public List<Stock> stocks = new List<Stock>();
    // We'll write to "Lab1_output.txt" in the same folder as the .exe
    readonly string destPath =
      System.IO.Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "Lab1_Output.txt");
    // We'll print this header in both console & file
    public string titles =
      "Broker".PadRight(10) +
```

```
    "Stock".PadRight(15) +
    "Value".PadRight(10) +
    "Changes".PadRight(10) +
    "Date and Time";

  public static bool printedHeaderToConsole = false;
  public StockBroker(string brokerName)
  {
    BrokerName = brokerName;
    // Print the header to console
    if (!StockBroker.printedHeaderToConsole)
    {
      Console.WriteLine(titles);
      StockBroker.printedHeaderToConsole = true;
    }

    // Overwrite (false) the file with this same header once
    using (StreamWriter outputFile = new StreamWriter(destPath, false))
    {
      outputFile.WriteLine(titles);
    }
  }
  public void AddStock(Stock stock)
  {
    stocks.Add(stock);
    // Subscribe to the stock's event using our event handler
    stock.StockEvent += EventHandler;
  }
   private void EventHandler(object sender, StockNotification e)
   {
     if (sender is not null)
       // The second parameter needs to be cast to StockNotification
       Helper(sender, e);
   }

  public void Helper(object sender, StockNotification e)
  {
    // We could cast the sender back to Stock if we needed more info
    Stock newStock = (Stock)sender;
    // Construct the output line
    string message =
      $"{BrokerName.PadRight(10)}" +
      $"{e.StockName.PadRight(15)}" +
      $"{e.CurrentValue.ToString().PadRight(10)}" +
```

```
        $"{e.NumChanges.ToString().PadRight(10)}" +
        $"{DateTime.Now}";

      try
      {
        // Append this line to the output file
        using (StreamWriter outputFile = new StreamWriter(destPath, true))
        {
          outputFile.WriteLine(message);
        }
        // Also write to console
        Console.WriteLine(message);
      }
      catch (IOException ex)
      {
        Console.WriteLine(ex.Message);
      }
    }
  }
}

// StockNotification.cs
using System;
using System.Collections.Generic;
using System.Text;
namespace Stock
{
  public class StockNotification : EventArgs
  {
    public string StockName { get; set; }
    public int CurrentValue { get; set; }
    public int NumChanges { get; set; }
    /// <summary>
    /// Stock notification attributes that are set and changed
    /// </summary>
    /// <param name="stockName">Name of stock</param>
    /// <param name="currentValue">Current vallue of the stock</param>
    /// <param name="numChanges">Number of changes the stock goes     through</param>
    public StockNotification(string stockName, int currentValue, int numChanges)
    {
      // !NOTE!: Fill in below of what the notification will do using the
      this.StockName = stockName;
      this.CurrentValue = currentValue;
      this.NumChanges = numChanges;
```

```
    }
  }
}
```