namespace StockApplication

{

// declared a delagate under the namespace of StockApplication to reference to a function

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

class Stock

{

// Stock attributes

.

.

.

// delegated an event that is to be raised

public event StockNotification RaiseStockEvent;

/// <summary>

/// constructor that runs a thread

/// </summary>

.

.

.

/// <summary>

/// method that controls how the method is to be run

/// </summary>

public void ChangeStockValue()

{

for(int i = 0; i < 10; i++)

{

Thread.Sleep(500); //0.5 seconds

\_cValue += \_random.Next(1, \_mChange);

\_numberChange++;

\_valueChange = \_cValue - \_sValue;

// if the value change is greater than or equals to the threshold, raise the event

if (\_valueChange >= \_threshold)

{

RaiseStockEvent(this.\_sName, this.\_cValue, this.\_numberChange);

}

}

}

}

}

namespace StockApplication

{

class StockBroker

{

// StockBroker attributes

.

.

.

// .txt file path

.

.

.

// for locking a thread to make sure it is not interrupted

private static Object lockThis = new Object();

/// <summary>

/// constructor of StockBroker

/// </summary>

.

.

.

/// <summary>

/// add a stock to the stocks list

/// </summary>

public void AddStock(Stock stockValue)

{

stocks.Add(stockValue);

// the event is called

stockValue.RaiseStockEvent += Notify;

}

/// <summary>

/// used to ensure that a block of code runs to completion without interruption by other threads

/// and prints to the .txt file

/// </summary>

public void Notify(string name, int value, int changeValue)

{

lock (lockThis)

{

StreamWriter outText = new StreamWriter(file, true);

outText.WriteLine(\_sBroker + "\t" + name + "\t" + value + "\t" + changeValue + "\n");

Console.WriteLine("{0} \t {1} \t {2} \t {3}", \_sBroker, name, value, changeValue);

outText.Close();

}

}

}

}