Stock.cs

```
using System;
using System.Threading;

namespace Stock
{
    public class Stock
    {
        public event EventHandler<StockNotification> StockEvent;

        private string _name;
        private int _initialValue;
        private int _maxChange;
        private int _threshold;
        private int _numChanges;
        private int _currentValue;
        private readonly Thread _thread;

        public string StockName { get => _name; set => _name = value; }
        public int InitialValue { get => _initialValue; }
        public int CurrentValue { get => _currentValue; }
        public int MaxChange { get => _maxChange; }
        public int Threshold { get => _threshold; }
        public int NumChanges { get => _numChanges; }

        public Stock(string name, int startingValue, int maxChange, int
threshold)
        {
            _name = name;
            _initialValue = startingValue;
            _currentValue = InitialValue;
            _maxChange = maxChange;
            _threshold = threshold;
            _numChanges = 0;

            _thread = new Thread(new ThreadStart(Activate));
            _thread.Start();
        }

        public void Activate()
```

```
        {
            for (int i = 0; i < 25; i++)
            {
                Thread.Sleep(500);
                ChangeStockValue();
            }
        }

        public void ChangeStockValue()
        {
            var rand = new Random();
            _currentValue += rand.Next(1, _maxChange + 1);
            _numChanges++;

            if (Math.Abs(_currentValue - _initialValue) > _threshold)
            {
                OnStockEvent(new StockNotification(_name, _currentValue,
_numChanges));
            }
        }

        protected virtual void OnStockEvent(StockNotification e)
        {
            StockEvent?.Invoke(this, e);
        }
    }
}
```

StockBroker.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Threading;

namespace Stock
{
    public class StockBroker
    {
        public string BrokerName { get; set; }
        public List<Stock> stocks = new List<Stock>();
```

```
        public static ReaderWriterLockSlim myLock = new
ReaderWriterLockSlim();
        readonly string destPath =
Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "Lab1_output.txt");
        string titles = "Broker".PadRight(16) + "Stock".PadRight(16) +
"Value".PadRight(16) + "Changes".PadRight(10) + "Date and Time";

        public StockBroker(string brokerName)
        {
            BrokerName = brokerName;
        }

        public void AddStock(Stock stock)
        {
            stocks.Add(stock);
            stock.StockEvent += EventHandler;
        }

        void EventHandler(object sender, StockNotification e)
        {
            try
            {
                myLock.EnterWriteLock();

                Stock newStock = (Stock)sender;


Console.WriteLine($"{BrokerName.PadRight(16)}{e.StockName.PadRight(16)}{e.
CurrentValue.ToString().PadRight(16)}{e.NumChanges.ToString().PadRight(10)
}{DateTime.Now}");

                using (StreamWriter outputFile = new
StreamWriter(destPath, true))
                {

outputFile.WriteLine($"{BrokerName.PadRight(16)}{e.StockName.PadRight(16)}
{e.CurrentValue.ToString().PadRight(16)}{e.NumChanges.ToString().PadRight(
10)}{DateTime.Now}");
                }
            }
```

```
            finally
            {
                myLock.ExitWriteLock();
            }
        }
    }
}
```

StockNotification.cs

```
using System;

namespace Stock
{
    public class StockNotification : EventArgs
    {
        public string StockName { get; set; }
        public int CurrentValue { get; set; }
        public int NumChanges { get; set; }

        public StockNotification(string stockName, int currentValue, int
numChanges)
        {
            StockName = stockName;
            CurrentValue = currentValue;
            NumChanges = numChanges;
        }
    }
}
```

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Stock
{
    internal class Program
    {
        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);
        }
    }
}
```