```
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; 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; }

        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();
        }

        public void Activate()
        {
            for (int i = 0; i < 25; i++)
            {
                Thread.Sleep(500); // 1/2 second delay
                ChangeStockValue();
            }
        }
        public void ChangeStockValue()
        {
            var rand = new Random();
            CurrentValue += rand.Next(1, MaxChange);
            NumChanges++;
            if (Math.Abs(CurrentValue - InitialValue) > Threshold)
            {
                StockEvent?.Invoke(this, new StockNotification(StockName, CurrentValue,
NumChanges));
            }
        }
    }
}
```