

## Code Review

The code starts from the `Server.Startup.cs`

```
0 references
public Startup(IConfiguration configuration, IWebHostEnvironment env) {
    this._config = configuration;
    _env = env;
    List<Chart> charts = Chart.NewChartList;
    ChartService chartService = new ChartService();
    chartService.InitializePrices(charts);
}
```

When the NewChartList is called, it creates 3 different securities dummies in singleton.

Singleton design pattern is used to improve performance, avoid wasteful computation, and reduce program memory requirements.

The list is stored in `charts` then calls `chartService.InitializePrices(charts)`.

```

18 public class ChartService : IChartService {
19     static List<Chart> charts_;
20     static Random random = new Random();
21
22     2 references
23     public List<Chart> FetchPrices() {
24         return charts_;
25     }
26
27     2 references
28     public async void InitializePrices(List<Chart> charts)
29     {
30         while (true)
31         {
32             for (int i = 0; i < charts.Count(); i++)
33             {
34                 int num = random.Next(1000, 9000);
35                 charts[i].Qt1 = num;
36                 charts[i].Change = charts[i].Qt0 - num;
37             }
38             charts_ = charts;
39             await Task.Delay(1000);
40         }
41     }
42 }

```

InitializePrices method runs a while loop every second and updates the quantities and displays the changes of quantities. The updates are stored in charts\_, where it's declared as static so that any new client can fetch the same data.

```

5
6 1 reference
7 private async void RefreshChartList(bool fromDB = true, Chart nChart = null)
8 {
9     int a = 0;
10    while (true)
11    {
12        this.ChartList.ItemsSource = null;
13        charts = UoW.BE.ChartRepo.FetchPrices();
14        if (a == 0)
15        {
16            int milliseconds = charts[0].Time.Millisecond;
17            await Task.Delay(1000 - milliseconds);
18            a++;
19        }
20
21        if (this.charts != null && this.charts.Count > 0)
22        {
23            this.ChartList.ItemsSource = this.charts
24        }
25        await Task.Delay(400);
26    }
27 }

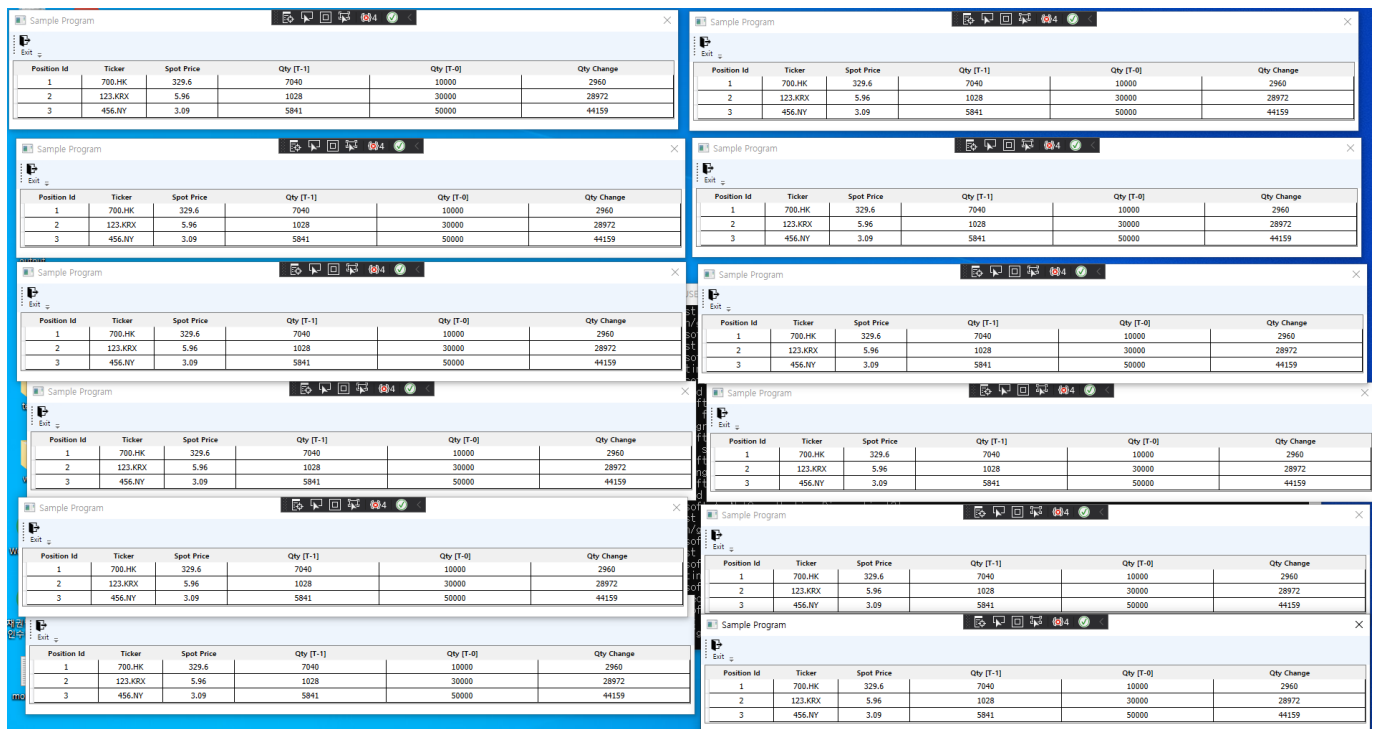
```

When the client is executed, RefreshCharList method is called to update the price. In order to sync the changes in prices from the server and update the prices with multiple clients, each client fetches DateTime from the server in millisecond. and waits until the millisecond reaches 00. This delay happens only once.

Then it renders the data in the window and refetches the newly update data from the server.

| Position Id | Ticker  | Spot Price | Qty [T-1] | Qty [T-0] | Qty Change |
|-------------|---------|------------|-----------|-----------|------------|
| 1           | 700.HK  | 329.6      | 3170      | 10000     | 6830       |
| 2           | 123.KRX | 5.96       | 1915      | 30000     | 28085      |
| 3           | 456.NY  | 3.09       | 7376      | 50000     | 42624      |

In order to run a single client,  
 Either press F5 or  
 Right click on Server -> Debug -> Start New Instance  
 Then Right click on Client -> Debug -> Start New Instance



To run multiple clients,  
 Right click on Server -> Debug -> Start New Instance  
 Then Right click on Client -> Debug -> Start New Instance based on the number of clients the team wants to execute.  
 The screenshot displays 12 clients running at the same time.

## Improvements

Theoretically, the clients should update the prices at the exact same time between other running clients because the DateTime is fetched from the server. The client delays to render the price until the DateTime reaches 00 milliseconds but each client renders and updates at a slight variation.

The RefreshChartList method contains a logic that delays for a second but this logic could have been taken out as an independent method to improve the readability.

In case the program doesn't run on your computer, try running Visual Studio 2019 as administrator.

I hope this document helps your team to understand the code.

Thank you for reading this and have a nice day.

Sincerely,  
 Moog