Yoo Kyung Baek(101282741)

Lab 8 Submission

April 2, 2021

Iterator.cs

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

namespace Lab8_Observer_Iterator_Pattern
{
    interface IIterator
    {
        string FirstItem { get; }
        string NextItem { get; }
        string CurrentItem { get; }
        bool IsDone { get; }
    }
    interface IAggregate
    {
        IIterator GetIterator();
        string this[int itemIndex] { set; get; }
        int Count { get; }
    }
    class MyAggregate : IAggregate
    {
        List<string> values_ = null;
        public MyAggregate()
        {
            values_ = new List<string>();
        }
        #region IAggregate Members
        public IIterator GetIterator()
        {
            return new MyIterator(this);
        }
        #endregion
        public string this[int itemIndex]
        {
            get
            {
                if (itemIndex < values_.Count)
                {
                    return values_[itemIndex];
                }
                else
                {
                    return string.Empty;
                }
            }
```

```
        set
        {
            values_.Add(value);
        }
    }
    public int Count
    {
        get
        {
            return values_.Count;
        }
    }
}
class MyIterator : IIterator
{
    IAggregate aggregate_ = null;
    int currentIndex_ = 0;
    public MyIterator(IAggregate aggregate)
    {
        aggregate_ = aggregate;
    }
    #region IIterator Members
    public string FirstItem
    {
        get
        {
            currentIndex_ = 0;
            return aggregate_[currentIndex_];
        }
    }
    public string NextItem
    {
        get
        {
            currentIndex_ += 1;
            if (IsDone == false)
            {
                return aggregate_[currentIndex_];
            }
            else
            {
                return string.Empty;
            }
        }
    }
    public string CurrentItem
    {
        get
        {
            return aggregate_[currentIndex_];
        }
    }
    public bool IsDone
    {
        get
```

```
            {
                if (currentIndex_ < aggregate_.Count)
                {
                    return false;
                }
                return true;
            }
        }
        #endregion
    }
}
```

Observer.cs

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

namespace Lab8_Observer_Iterator_Pattern
{
    interface ISubject
    {
        void Subscribe(Observer observer);
        void Unsubscribe(Observer observer);
        void Notify();
    }
    interface IObserver
    {
        void Update();
    }
    public class Subject : ISubject
    {
        private List<Observer> observers = new List<Observer>();
        private int _int;
        public int Inventory
        {
            get
            {
                return _int;
            }
            set
            {
                // Just to make sure that if there is an increase in inventory then only we are
notifying the observers.
                if (value > _int)
                    Notify();
                _int = value;
            }
        }
        public void Subscribe(Observer observer)
        {
```

```
            observers.Add(observer);
        }
        public void Unsubscribe(Observer observer)
        {
            observers.Remove(observer);
        }
        public void Notify()
        {
            observers.ForEach(x => x.Update());
        }
    }
    public class Observer : IObserver
    {
        public string ObserverName { get; private set; }
        public Observer(string name)
        {
            this.ObserverName = name;
        }
        public void Update()
        {
            Console.WriteLine("{0}: A new product has arrived at the store",
this.ObserverName);
        }
    }
}
```

Program.cs

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

namespace Lab8_Observer_Iterator_Pattern
{
    class Program
    {
        static void Main(string[] args)
        {
            MyAggregate aggr = new MyAggregate();

            Console.WriteLine("How many Observers?");


            int userinput = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("---------------------------------------");
            Console.WriteLine("This program will create and print: " + userinput + "
Observers");

            for (int i = 0; i < userinput; i++)
            {
                aggr[i] = new Observer("Observer Number:" + i).ObserverName;
```

```
            }

            IIterator iter = aggr.GetIterator();
            for (string s = iter.FirstItem; iter.IsDone == false; s = iter.NextItem)
            {
                Console.WriteLine(s);
            }
            Console.ReadLine(); //add the make the console window stay as it will expect some
user input
        }
    }
}
```

Output