Буниф Мажда

18-КБ-ПР2

**Отчет КПО Лаб 8**

**4 Задание**

Задание1) Задание выполняется совместно несколькими студентами (2-3  
человека в группе).  
2) Разработать приложение с использованием паттерна Observer.  
Реализовать форму, содержащую кнопки активации/изменения  
состояния субъекта (Subject) и индикаторы состояния каждого  
из наблюдателей (Observer). Осуществлять выбор активных  
наблюдателей через CheckBox’ы. Наблюдателей реализовать в  
соответствии с вариантами их пункта 4. (При реализации состояний рекомендуется использовать конечные автоматы).  
3) Составить тесты к методам класса, проверяющие корректность  
их работы.  
4) Запустить тестирование и проверить результаты работы.  
5) Написать программу, демонстрирующую работу класса, осуществляющее визуальное представление результатов.  
6) Составить отчет.

**Листинг :**

**Interface IObserver**

public interface IObserver

{

void update(Subject sub);

}

**Interface ISubject**

public interface ISubject

{

void registerObserver(Observer observer);

void unregisterObserver(Observer observer);

void notifyObservers();

}

**Class Implements IObserver Interface**

public class Observer : IObserver

{

private string NameObserver;

public Observer(string Name)

{

NameObserver = Name;

}

public void update(Subject sub)

{

GeInfo(sub);

}

public string GeInfo(Subject sub)

{

string info = "Hello " + NameObserver + "!\n" + sub.Name + " is now available at " + sub.Price + " with Discount = " + sub.discount +"%";

return info;

}

}

}

**Class Implements ISubject Interface**

public class Subject : ISubject

{

private int count = 0;

private int sum = 0;

private string name;

private float basePrice;

private float currentPrice;

private List<Observer> Observers = new List<Observer>();

public Subject(string name, float basePrice)

{

this.name = name;

this.basePrice = basePrice;

this.currentPrice = basePrice;

}

public float Price

{

get

{

return basePrice;

}

set

{

basePrice = value;

if (value > currentPrice)

{

//notifyObservers();

}

}

}

public void registerObserver(Observer observer)

{

if (!Observers.Contains(observer))

{

Observers.Add(observer);

}

}

public int CountObserver(Observer observer)

{

return Observers.Count();

}

public void unregisterObserver(Observer observer)

{

Observers.Remove(observer);

}

public void notifyObservers()

{

foreach (var observer in Observers)

{

observer.update(this);

}

}

public string Name

{

get { return name; }

}

public float discount

{

get { return (basePrice - currentPrice) \* 100 / basePrice; }

}

public float CurrentPrice

{

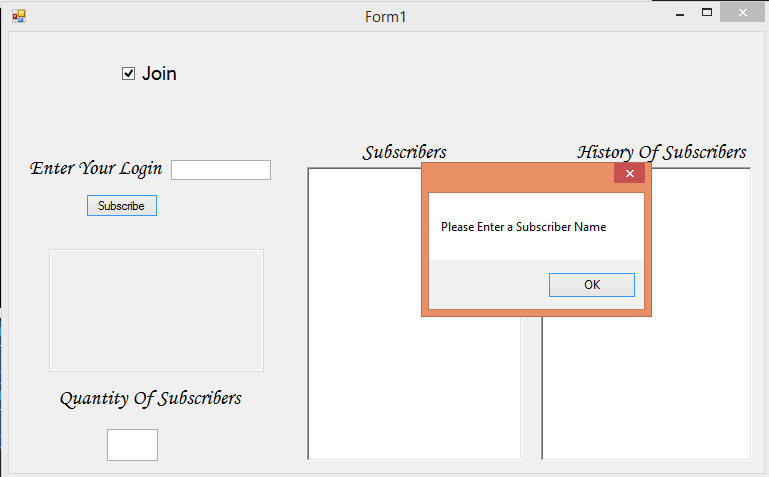
get { return currentPrice; }

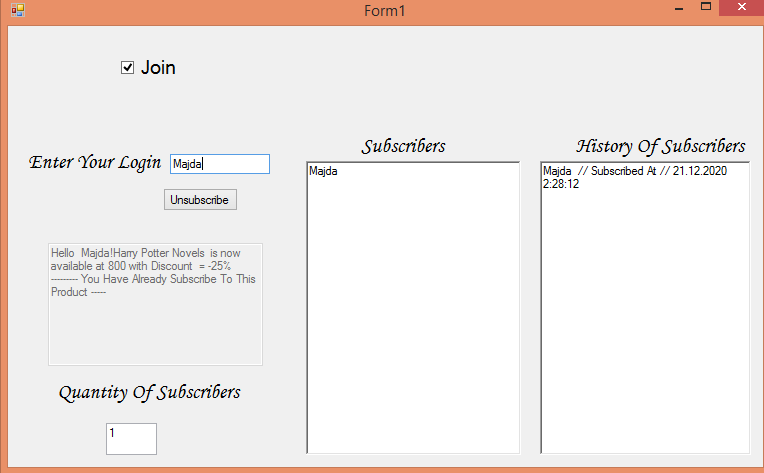
}

}

}

**By running the program the output is:**





**By Unsubscribing**

