Спостерігач — це поведінковий патерн проектування, який створює механізм підписки, що дає змогу одним об’єктам стежити й реагувати на події, які відбуваються в інших об’єктах.

На даний момент цього патерна в моєму проекті немає, але є в моїй попередній лабі.

Посилання на попередню лабораторну.

Тут цей патерн реалізований у спрощеному вигляді без великої кількості функцій.

В цій лабі є багато методів сортування, вони відіграють роль підписника, а також є основний клас, який відіграє роль видавника.

SortEventHandler – це клас-видавник до якого будуть підписуватися методи сортування в ньому є метод **Sort**, який викликає всі підписанні методи.

```
2 references
public class SortEventHandler<T> where T : IComparable
{
    /// <summary>
    /// Event for subscribing different sort methods.
    /// </summary>
    public event EventHandler<SortingEventArgs<T>> sortMethod;
    /// <summary>
    /// The method for sorting **MyList list.**
    /// </summary>

    1 reference
    public void Sort(MyList<T> list)
    {
        StartSorting(list);

    }
    /// <summary>
    /// Protected method where the **MyList list** connects to the class.
    /// </summary>
    /// <param name="list">List for sorting.</param>
    1 reference
    protected virtual void StartSorting(MyList<T> list)
    {
        sortMethod?.Invoke(this, new SortingEventArgs<T> { myList = list });
    }
}
```

Допоміжний клас **SortingEventArgs**, який вкладає лист для сортування у виклик метода сортування.

```
public class SortingEventArgs<T> : EventArgs where T : IComparable
{
    public MyList<T> myList;


}
```

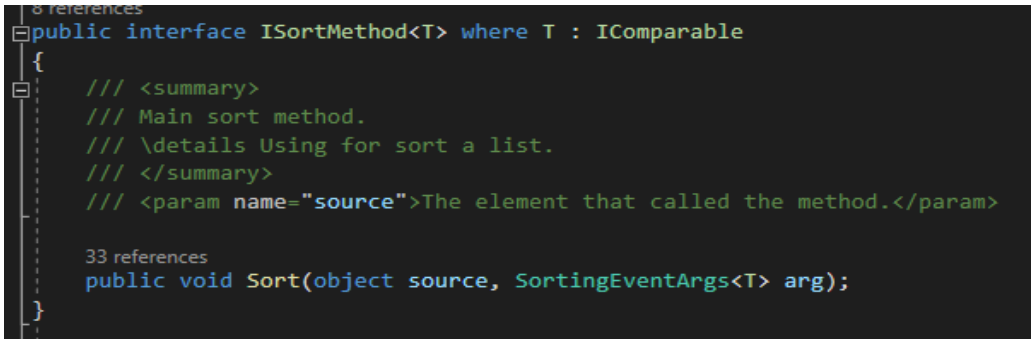
Також є клас **SortMethodsInvoker**, цей клас підписує методи сортування до класу **SortEventHandler**, а також визиває сортування.

```
public class SortMethodsInvoker<T> where T : IComparable
{
    /// <summary>
    /// List to sort. This list are used for adding new elements, sorting by different methods.
    /// </summary>
    public MyList<T> listToSort;
    /// <summary>
    /// Sort event handler. Event for containing different sort methods.
    /// </summary>
    public SortEventHandler<T> subscribeSortMethods = new SortEventHandler<T>();
    public TextBox textBox;
```

Метод **SubscribeAndSort** приймає метод сортування та підписує його до **SortEventHandler**, після чого визиває цей метод у видавця і передає йому для сортування лист, а потім відписує його від видавця.

```
/// <summary>
/// The function that is used to subscribe the sort methods to the sort event.
/// </summary>
/// <param name="methodForSigning">Sort method that would be subscribed.</param>
//function to subscribe sort methods and sort listToSort
7 references
public void SubscribeAndSort(ISortMethod<T> methodForSigning)
{
    subscribeSortMethods.sortMethod += methodForSigning.Sort;
    subscribeSortMethods.Sort(listToSort);
    subscribeSortMethods.sortMethod -= methodForSigning.Sort;
    PrintList();
}
```

Всі методи сортування успадковують інтерфейс **ISortMethod**.



Цей патерн дуже підійшов для моєї лаби, тому що в мене є велика кількість різних об’єктів, яким потрібно створити функціонал прив’язування, що і робить цей патерин.

Посилання на попередню лабораторну.

Класи можете знайти за шляхом:

1a\After\Task1\SortMethodsInvoker.cs

1a\After\Task1\SortingMethods.cs

1a\After\Task1\Sorting.cs

1a\After\Task1\MyList.cs