**Отчет**

**Лабораторная работа №3**

**С дисциплины: «C#»**

**Тема: «** **Шаблоны проектирования. Поведенческие шаблоны »**

**Автор: Ярцев Д.А.**

**Группа: ИС61**

**Цель**: ознакомиться с основными шаблонами проектирования, научиться применять их при проектировании и разработке ПО.

**Паттерн** — это не законченное архитектурное решение, которое можно напрямую преобразовать в исходный или машинный код. Это описание подхода к решению проблемы, который можно применять в разных ситуациях

**Вариант №7**

Заказ на получение гранта для обучения может находиться в нескольких состояниях:

1. создан,
2. рассматривается,
3. отложен,
4. отклонен,
5. подтвержден,
6. отозван

Определить логику изменения состояний

и разработать модель системы.

**Ход работы**

Пусть есть люди что подают заявки в фонт на грант – авторы.

Пусть есть люди, что рассматривают гранты, где – сотрудники фонда.

Нам нужно уведомлять авторов о изменении состояния их заявок.

Тут подходит паттерн observer, где авторы – наблюдатели, сотрудники – издатели, а состояние заявки – индикатор обновления информации у наблюдателя.

Общий алгоритм работы программы таков:

1. создаем список заявок на грант
2. Назначаем по каждому гранту ответственного за расмотрение данной заявки.
3. Ответсвенный за заявку сотрудник меняет состояние заявки.
4. Наблюдатель проверяет получает обновление состояния его заявки
5. Завершается программа

**Реализация: вывод**

c# lab\_3 author: Yarcev D.A. IS-61

variant - 7

================================================================================

menu:

1) add grant

2) edit state grants

3) show state grants

4) close program

to do: 1

--------------------------------------------------------------------------------

input grand name :g\_1

input autor name :a\_1

input worker name:w\_1

--------------------------------------------------------------------------------

menu:

1) add grant

2) edit state grants

3) show state grants

4) close program

to do: 3

--------------------------------------------------------------------------------

list observers:

0 ~ a\_1 ~ создан

1 ~ w\_1 ~ создан

--------------------------------------------------------------------------------

menu:

1) add grant

2) edit state grants

3) show state grants

4) close program

to do: 2

--------------------------------------------------------------------------------

list grands:

0 - g\_1

list states:

0 - создан

1 - рассматривается

2 - отложен

3 - отклонен

4 - подтвержден

5 - отозван

input num grand:0

input num status:3

--------------------------------------------------------------------------------

menu:

1) add grant

2) edit state grants

3) show state grants

4) close program

to do: 3

--------------------------------------------------------------------------------

list observers:

0 ~ a\_1 ~ отклонен

1 ~ w\_1 ~ отклонен

--------------------------------------------------------------------------------

menu:

1) add grant

2) edit state grants

3) show state grants

4) close program

to do: 4

--------------------------------------------------------------------------------

--------------------------------------------------------------------------------

Program is ending ...

**Реализация: код**

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Yarcev\_c\_lab\_3

{

/// <summary>

/// в нашем случае

/// у наблюдателя всегда полный список

/// издателей (заказов на грант)

/// </summary>

class Program

{

static void lineCh(char s)

{

for (int i = 0; i < 80; i++)

Console.Write(s);

Console.WriteLine();

}

static void Main(string[] args)

{

Console.WriteLine(" c# lab\_3 author: Yarcev D.A. IS-61");

Console.WriteLine(" variant - 7");

lineCh('=');

//список заявок на грант

List<ConcreteSubject> subjects = new List<ConcreteSubject>();

List<ConcreteObserver> observers = new List<ConcreteObserver>();

//list stam

List<string> status = new List<string>() {

"создан",

"рассматривается",

"отложен",

"отклонен",

"подтвержден",

"отозван "

};

bool keyLoop = true;

while (keyLoop)

{

//subjects.Add(new ConcreteSubject());

//observers.Add(new ConcreteObserver(subjects.Last()));

//observers.Add(new ConcreteObserver(subjects.Last()));

//subjects.Last().Attach(observers[observers.Count - 2]);

//subjects.Last().Attach(observers[observers.Count - 1]);

//subjects.Last().State = "Some State ...";

//subjects.Last().Notify();

//Console.WriteLine(observers[observers.Count - 2].observerState);

//Console.WriteLine(observers[observers.Count - 1].observerState);

Console.WriteLine(" menu:");

Console.WriteLine(" 1) add grant");

Console.WriteLine(" 2) edit state grants");

Console.WriteLine(" 3) show state grants");

Console.WriteLine(" 4) close program");

bool key = true;

while (key)

{

key = false;

Console.Write(" to do: ");

int chose = Convert.ToInt32(Console.ReadLine().ToString());

if (chose < 1 || chose > 4)

{

key = true;

Console.WriteLine(" Erorr!");

}

lineCh('-');

switch (chose)

{

case 1://create

//sub name

ConcreteSubject new\_Subject = new ConcreteSubject();

Console.Write(" input grand name :");

new\_Subject.Name = Console.ReadLine().ToString();

new\_Subject.State = status[0];

subjects.Add(new\_Subject);

//obs name

observers.Add(new ConcreteObserver(subjects.Last()));

observers.Add(new ConcreteObserver(subjects.Last()));

Console.Write(" input autor name :");

observers[observers.Count - 2].Name = Console.ReadLine().ToString();

Console.Write(" input worker name:");

observers[observers.Count - 1].Name = Console.ReadLine().ToString();

//add

subjects.Last().Attach(observers[observers.Count - 2]);

subjects.Last().Attach(observers[observers.Count - 1]);

//add

break;

case 2://edit

Console.WriteLine("list grands:");

for (int ii = 0; ii < subjects.Count; ii++)

Console.WriteLine(ii + " - " + subjects[ii].Name);

Console.WriteLine("list states:");

Console.WriteLine("0 - создан");

Console.WriteLine("1 - рассматривается");

Console.WriteLine("2 - отложен");

Console.WriteLine("3 - отклонен");

Console.WriteLine("4 - подтвержден");

Console.WriteLine("5 - отозван ");

Console.Write(" input num grand:");

int i = Convert.ToInt32(Console.ReadLine().ToString());

Console.Write(" input num status:");

int j = Convert.ToInt32(Console.ReadLine().ToString());

if (i < 0 || i > subjects.Count - 1) i = 0;

if (j < 0 || j > 5) j = 0;

//edit

//Console.WriteLine(subjects[i].Name);

//Console.WriteLine(subjects[i].State);

//Console.WriteLine(status[j]);

subjects[i].State = status[j];

subjects[i].Notify();

break;

case 3://show

Console.WriteLine("list observers:");

for (int ii = 0; ii < observers.Count; ii++)

Console.WriteLine(" " + ii + " ~ " + observers[ii].Name +" ~ "+ observers[ii].observerState);

break;

case 4:

keyLoop = false;

break;

}

}

lineCh('-');

}

Console.WriteLine(" Program is ending ...");

Console.ReadKey();

}

}

}

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Yarcev\_c\_lab\_3

{

/// <summary>

/// грант с людьми что слядят за ним

/// </summary>

abstract class Subject

{

public List<Observer> observers = new List<Observer>();

public void Attach(Observer observer)

{

observers.Add(observer);

observer.Update();

}

public void Detach(Observer observer)

{

observers.Remove(observer);

}

public void Notify()

{

foreach (Observer observer in observers)

observer.Update();

}

}

}

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Yarcev\_c\_lab\_3

{

/// <summary>

/// сотрудник фонда или

/// автор гранта

/// </summary>

abstract class Observer

{

public abstract void Update();

}

}

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Yarcev\_c\_lab\_3

{

/// <summary>

/// заявление на получение

/// гранта - конкретный издатель

/// </summary>

class ConcreteSubject : Subject

{

//состояние заявления

public string State { get; set; }

//name

public string Name { get; set; }

}

}

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Yarcev\_c\_lab\_3

{

/// <summary>

/// конкретный сотрудник фонда или автор гранта

/// </summary>

class ConcreteObserver : Observer

{

public string observerState;

public string Name;

ConcreteSubject subject;

public ConcreteObserver(ConcreteSubject subject)

{

this.subject = subject;

}

public override void Update()

{

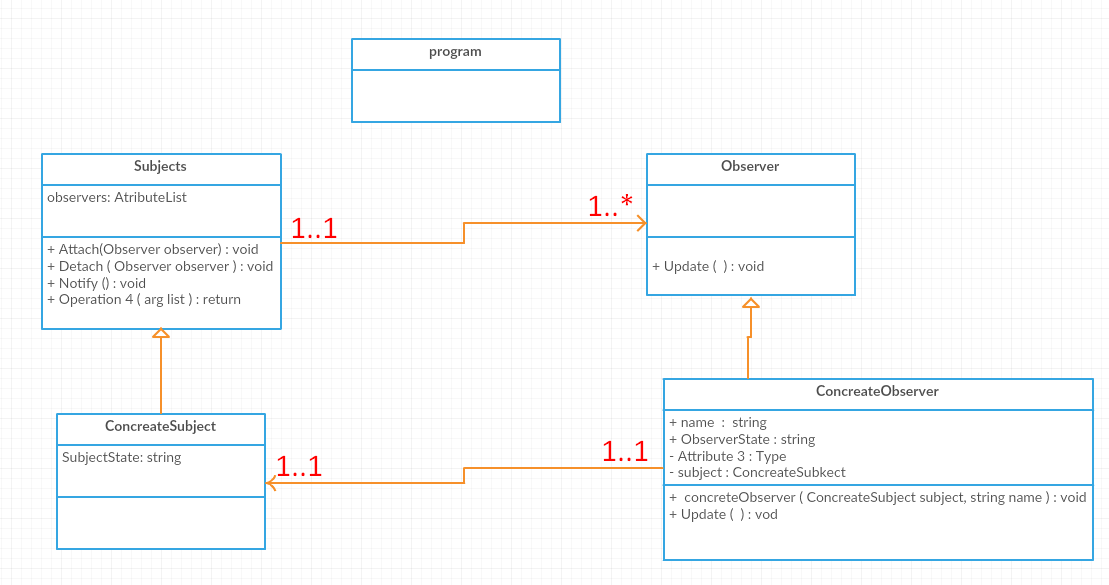
observerState = subject.State;

}

}

}

**Структура программы на языке UML**



**Классы**

|  |  |
| --- | --- |
| **Паттерн Factory method**  **Составные классы и их описание** | |
| **Наименование** | **Описание** |
| Programm | Главный класс программы |
| Subject | Издатель содержит ссылки на своих подписчиков и предоставляет интерфейс (набор методов) для добавления и удаления подписчиков. На издателя может ссылаться любое число подписчиков. |
| ConcreteSubject | Конкретный издатель посылает уведомление своим подписчикам и передает им свое состояние. |
| Observer | Подписчик предоставляет интерфейс (набор методов) для обновления своего состояния при изменении состояния издателя. |
| ConcreteObserver | Реализует интерфейс обновления, предоставляемый абстрактным классом Observer и поддерживает согласованность состояния с издателем. |

**Вивод:**

Паттерны довольно сильно ускоряют процесс разработки ПО. Обеспечивают четко прослеживаемую структуру.