**Лабораторная работа 15 «Паттерн "Наблюдатель"».**

**Наблюдатель** — это поведенческий паттерн проектирования, который создаёт механизм подписки, позволяющий одним объектам следить и реагировать на события, происходящие в других объектах.

Рассмотрим простой пример. Некая форма играет роль источника разных ситуаций. Для простоты будем считать, что их всего две: событие «А» и событие «В».

Остальные формы проекта (а их число может быть любым) могут реагировать на данные события разными способами (а могут и не реагировать). Назовём эти формы наблюдателями. Для простоты будем считать, что разных реакций будет всего три:

1 – «Временно окрасить» – на 3 секунды форма-наблюдатель меняет свой цвет на красный.

2 – «Переставить влево» – форма-наблюдатель перемещается влево на 3 пикселя.

3 – «Временно обезглавить» – на 5 заголовок формы-наблюдателя «исчезает», а потом восстанавливается.

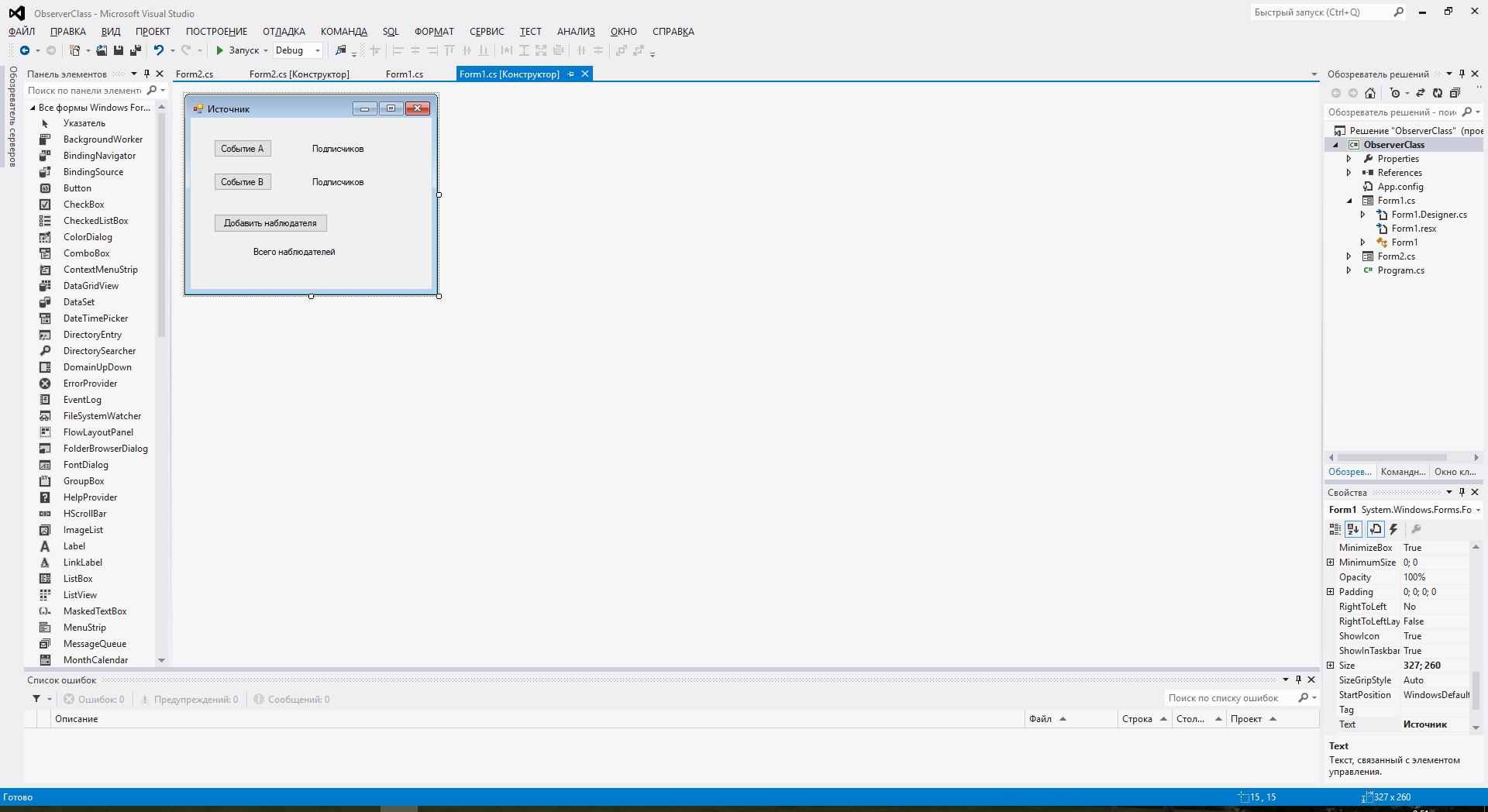
При этом у одного наблюдателя может быть несколько реакций на одно и то же событие, причём некоторые реакции могут быть одинаковыми.

Реализаций данной задачи может быть несколько, причём могут быть использованы очень непохожие подходы. Например, можно организовать так, чтобы наблюдатель помнил, какие операции и в каком порядке он должен запускать в ответ на то или иное событие. И при этом постоянно отсматривал некую доску объявлений, куда источник событий выкладывал бы сообщение о наступлении определённого события.

Паттерн «наблюдатель (observer)» или «источник - наблюдатель» подразумевает, что наблюдатель вопреки своему названию не следит за источником, а просто подписывает (назначает) свои методы в качестве автоматически запускаемых при возникновении события. При этом запускать эти методы (и, соответственно, хранить их адреса) будет не наблюдатель, а источник. Для хранения адресов запускаемых методов источник использует по одной переменной типа «делегат» на каждое событие.

Естественно, разработка идёт в режиме «Windows Forms», с использованием двух типов форм (Form1, Form2).

Form1 представляет «источник».



public Action EventA; int CountA = 0;

public Action EventB; int CountB = 0;

Поле **EventA** представляет делегат типа **Action**, в котором накапливается список методов, запускаемых при генерации события «А». В поле хранится число таких **CountA** методов. Поля **EventВ** и **CountВ** имеют аналогичный смысл.

Кнопка «Добавить наблюдателя» создаёт и открывает новую форму «наблюдатель».

Form2 f2 = new Form2(this);

Observers.Add(f2);

ObsCount++;

f2.Text = "Наблюдатель " + ObsCount;

f2.Show();

ShowInfo();

**Observers** – это переменная формы

List<Form2> Observers = new List<Form2>();

которая хранит список форм «наблюдателей».

**ObsCount** – это переменная формы

int ObsCount = 0;

которая хранит число созданных форм «наблюдателей».

При создании формы «наблюдатель» ей передаётся ссылка на форму «источник». Далее номер наблюдателя выводится в заголовок. Внешний вид формы обновляется и выводится текущая информация:

private void ShowInfo()

{

label1.Text = "Подписчиков " + CountA;

label2.Text = "Подписчиков " + CountB;

label3.Text = "Наблюдателей " + ObsCount;

}

При нажатии на кнопки «Событие А» и «Событие В» после проверки на «не null» запускаются все методы представленные соответствующим делегатом.

private void button1\_Click(object sender, EventArgs e)

{

if (EventA != null) EventA(); }

private void button2\_Click(object sender, EventArgs e)

{

if (EventB != null) EventB(); }

Регистрация подписки выполняется методом

public void Register(string Name, object o, Action Method)

{

bool Done = false;

switch (Name)

{

case "A": EventA += Method; CountA++; Done = true; break;

case "B": EventB += Method; CountB++; Done = true; break;

}

if (Done)

{

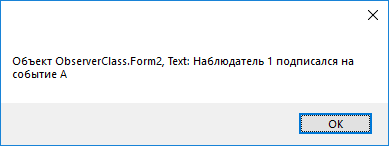
MessageBox.Show(string.Format("Объект {0} подписался на событие {1}", o, Name));

ShowInfo();

}

}

Здесь определяется какому делегату следует подписать метод, увеличивается соответствующий счётчик, и, если это успешно сделано, появляется соответствующее окно-сообщение.



Для отмены подписки применяется аналогичный метод.

public void UnRegister(string Name, object o, Action Method)

{

bool Done = false;

switch (Name)

{

case "A": EventA -= Method; CountA--; Done = true; break;

case "B": EventB -= Method; CountB--; Done = true; break;

}

if (Done)

{

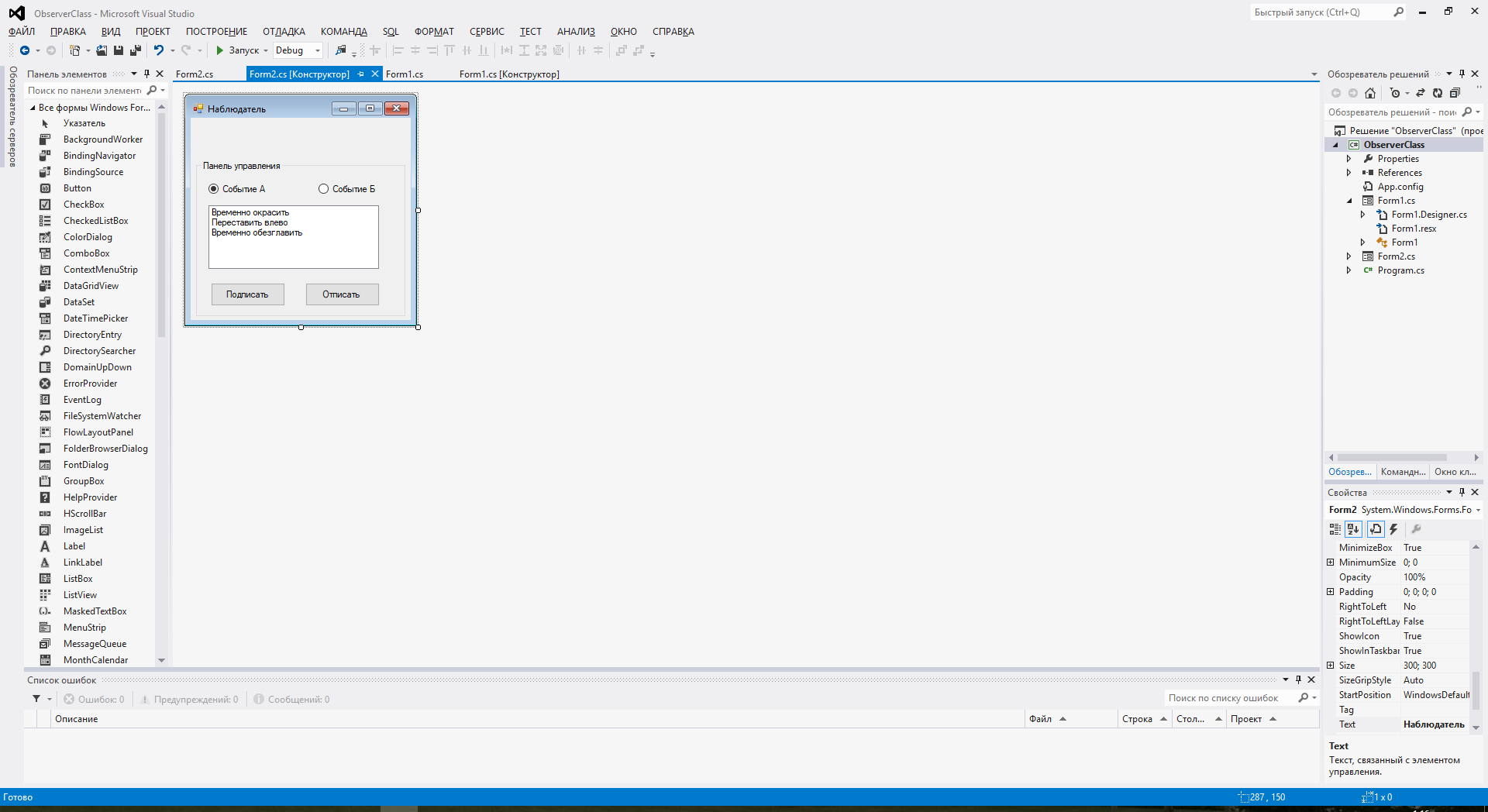
MessageBox.Show(string.Format("Объект {0} отподписался от события {1}", o, Name));

ShowInfo();

}

}

Возможности наблюдателя реализованы в классе Form2.



Таких форм может быть довольно много. Создание экземпляра формы включает сохранение адреса главной формы («источника») для дальнейшей работы с ней.

Form1 mainF;

public Form2(Form f)

{

InitializeComponent();

mainF = (Form1)f;

}

Методы, которые предполагается назначать в качестве реакции данной формы на события в источнике:

1 – «Временно окрасить» – на 3 секунды форма-наблюдатель меняет свой цвет на красный.

public void Action1()

{

Color c = BackColor;

BackColor = Color.Red;

Refresh();

Thread.Sleep(1000);

BackColor = c;

Refresh();

Thread.Sleep(500);

}

2 – «Переставить влево» – форма-наблюдатель перемещается влево на 3 пикселя.

public void Action2()

{

Left-=20;

}

3 – «Временно обезглавить» – на 5 заголовок формы-наблюдателя «исчезает», а потом восстанавливается.

public void Action3()

{

string t = Text;

Text = "";

Refresh();

Thread.Sleep(5000);

Text = t;

Refresh();

Thread.Sleep(500);

}

Также функционал формы позволяет подписываться на обработку событий источника;

private void button1\_Click(object sender, EventArgs e)

{

string Nam = "A";

if (radioButton2.Checked) Nam = "B";

Action Act = null;

switch (listBox1.SelectedIndex)

{

case 0: Act = Action1; break;

case 1: Act = Action2; break;

case 2: Act = Action3; break;

}

mainF.Register(Nam, this, Act);

}

и отписываться от обработки:

private void button2\_Click(object sender, EventArgs e)

{

string Nam = "A";

if (radioButton2.Checked) Nam = "B";

Action Act = null;

switch (listBox1.SelectedIndex)

{

case 0: Act = Action1; break;

case 1: Act = Action2; break;

case 2: Act = Action3; break;

}

mainF.UnRegister(Nam, this, Act);

}

Запустите приложение и проверьте его работу.

Создайте 3-х наблюдателей и назначьте каждому из событий по 5 обработчиков (от разных наблюдателей).

Самостоятельно, часть 1 (3 балла, за полную реализацию в классе +1балл):

1. Добавьте ещё одно событие (событие «С») и ещё один обработчик (форма «наблюдатель» на 1 секунду становится невидимой).
2. Организуйте подсказки для кнопок «источника»:

а) у каждой кнопки «Событие …» подсказка содержит список методов, подписанных на событие, с указанием номера наблюдателя;

б) у кнопки «Добавить наблюдателя» подсказка содержит список реально существующих наблюдателей (открытых и не закрытых) с количеством подписок каждого из них.

1. Сделайте так, чтобы при закрытии наблюдателя его имя изымалось из списка и его подписки на события также закрывались.

Самостоятельно, часть 2 (на дом, 6 баллов, сдаётся только занятии, через две недели, ни раньше, ни позже).

Реализовать с использованием данного паттерна эмуляцию издателя периодики и подписчиков на периодику.

**Функционал издателя:**

1. Открывает новые издания: газеты (периодичность 1 день) и журналы (периодичность 1 неделя) с немедленным их выпуском. Издания имеют уникальное название.
2. Издаёт открытые издания (изменение дня реализовано через таймер, 1 день = 5 секунд, издания «направляются» подписчикам).
3. Приостанавливает открытые издания (подписчики перестают их получать).
4. Закрывают открытые издания (подписка на них аннулируется у всех подписчиков).
5. Регистрирует нового подписчика, указывая его имя. Для каждого подписчика открывается своя форма.
6. Подписывает подписчиков на издания.
7. Отписывает подписчика от издания.
8. В многострочном texBox ведётся протокол событий с указанием их даты, например:

…

12.03.18 Выпуск газеты «Правда» №14. Получили 3 подписчиков.

12.03.18 Выпуск журнала «Звезда» №2. Получили 1 подписчиков.

13.03.18 Подписчик «Номер 3» подписался на журнал «Звезда»

13.03.18 Подписчик «Номер 5» аннулировал подписку на газету «Правда»

…

1. По щелчку на форме появляется отдельная форма с указанием текущего дня и перечнем актуальных изданий. У каждого издания указывается его порядковый номер, название (с уточнением газета или журнал), дата открытия, статус (издаётся, приостановлено) и списком актуальных на данную дату подписчиков этого издания.
2. В подсказке формы представлен список всех издающихся (не приостановленных) изданий.

**Функционал подписчика:**

1. Имеет интерфейс для выбора издания для подписки на него. В списке выбора присутствуют только открытые издания
2. Имеет интерфейс для выбора издания для закрытия подписки на него.
3. В многострочном texBox ведётся протокол событий с указанием их даты, например:

…

12.03.18 Получена газеты «Правда» №14.

12.03.18 Получен журнал «Звезда» №2.

13.03.18 Аннулирована подписка на газету «Правда»

14.03.18 Оформлена подписка на журнал «Наука и жизнь»

…

Даты всех протоколов должны быть синхронизированы.

1. В подсказке формы представлен список всех подписных изданий данного подписчика.

Приложение оконное.

Сдаётся программа (на паре) и отчёт.

Отчёт в электронном виде отсылается мне на почту до 18-00 дня, предшествующего занятию. Файл имеет имя в формате «Петров, 184-2. Лаб 7\_2. Наблюдатель».

В отчёте;

* привести скриншот экрана, на котором видны протоколы каждой формы и форма актуальных изданий.
* привести список информационных полей, созданных на каждой форме с описанием их назначения.
* для каждого элемента функционала приводится его текст, внятное описание реализации, список полей, свойств и тексты методов, необходимых для реализации с указанием, в каком классе реализованы.