**13 События**

Задание 1 и задание 2. Создать 3 метода для вывода трех различных сообщений. Создать делегат. В программе вызывать все три метода при помощи делегата. На основе своей программы создать метод, в котором делегат будет параметром.

Листинг программы:

class TestStrAndDelegate

{

public void StrPrint()

{

Console.WriteLine("Тест делегата в качестве параметра");

}

public void StrPrintx2()

{

Console.WriteLine("Тест делегата в качестве параметра");

}

public void DelegatePrint(Message msg) //делегат в качестве параметра

{

msg(); //выполняем делегат (выполняем методы, которые мы занесли в делегат)

}

}

class MyEvent

{

public event MyEventHandler activate; //public event имя\_делегата имя\_объекта

public void fire() //запуск события

{

if(activate != null)

{

activate();

}

}

}

class X

{

public void Xhandler()

{

Console.WriteLine("Res = X");

}

}

class Y

{

public void Yhandler()

{

Console.WriteLine("Res = Y");

}

}

class EventDemo

{

static void Handler()

{

Console.WriteLine("Res = Event");

}

public static void Main()

{

try

{

MyEvent evt = new MyEvent();

X xObj = new X();

Y yObj = new Y();

evt.activate += new MyEventHandler(xObj.Xhandler); //в событие передаем делгат MyEventHandler, в скобках которого указываем метод определенного объекта

evt.activate += new MyEventHandler(yObj.Yhandler);

evt.activate += new MyEventHandler(Handler);

evt.fire(); //активируем событие, тем самым запуская методы объектов, переданных в него (в событие)

Console.WriteLine();

TestStrAndDelegate testStr = new TestStrAndDelegate();

Message msg = testStr.StrPrint;

msg += testStr.StrPrintx2;

testStr.DelegatePrint(msg);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

}

}

delegate void MyEventHandler();

delegate void Message();

Таблица 13.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Res = X  Res = Y  Res = Event  Тест делегата в качестве параметра  Тест делегата в качестве параметра |

Источник: собственная разработка

Анализ результатов представлен на рисунке 13.1.

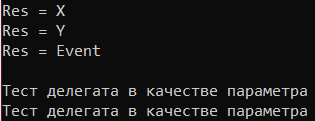


Рисунок 13.1 – Результат работы программы

Источник: собственная разработка

Задание 3. Для класса MyInfo содержащего поле name (Ваше имя) создать событие выводящее оповещение в случае изменения значения поля name.

Листинг программы:

class TestStrAndDelegate

{

public void StrPrint()

{

Console.WriteLine("Тест делегата в качестве параметра");

}

public void StrPrintx2()

{

Console.WriteLine("Тест делегата в качестве параметра");

}

public void DelegatePrint(Message msg) //делегат в качестве параметра

{

msg(); //выполняем делегат (выполняем методы, которые мы занесли в делегат)

}

}

class MyEvent

{

public event MyEventHandler activate; //public event имя\_делегата имя\_объекта

public void fire() //запуск события

{

if(activate != null)

{

activate();

}

}

}

class X

{

public void Xhandler()

{

Console.WriteLine("Res = X");

}

}

class Y

{

public void Yhandler()

{

Console.WriteLine("Res = Y");

}

}

class EventDemo

{

static void Handler()

{

Console.WriteLine("Res = Event");

}

public static void Main()

{

try

{

MyEvent evt = new MyEvent();

X xObj = new X();

Y yObj = new Y();

evt.activate += new MyEventHandler(xObj.Xhandler); //в событие передаем делгат MyEventHandler, в скобках которого указываем метод определенного объекта

evt.activate += new MyEventHandler(yObj.Yhandler);

evt.activate += new MyEventHandler(Handler);

evt.fire(); //активируем событие, тем самым запуская методы объектов, переданных в него (в событие)

Console.WriteLine();

TestStrAndDelegate testStr = new TestStrAndDelegate();

Message msg = testStr.StrPrint;

msg += testStr.StrPrintx2;

testStr.DelegatePrint(msg);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

}

}

delegate void MyEventHandler();

delegate void Message();

Таблица 13.2 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Имя изменено на Max  Имя изменено на Art  Имя изменено на Vlad |

Источник: собственная разработка

Анализ результатов представлен на рисунке 13.2.



Рисунок 13.2 – Результат работы программы

Источник: собственная разработка

Задание 4. Создать класс, в нем делегат и соответствующее ему событие. Создать два класса-наблюдателя, в методах которых будет описана их реакция на событие. Добавить 2 обработчика к событию из первого класса и один из второго. Вывести результат на экран. Удалить один обработчик события и вывести результат на экран.

Листинг программы:

try

{

Console.Write("Введите имя: ");

string name = Console.ReadLine();

MyInfo inf = new MyInfo();

X xObj = new X();

Y yObj = new Y();

inf.Notify += xObj.XHandler;

inf.Notify += yObj.YHandler;

inf.ChangeName(name);

Console.WriteLine();

inf.Notify -= yObj.YHandler;

inf.ChangeName(name);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

delegate void Info(string message);

class MyInfo

{

public string name = "Tom";

public event Info? Notify;

public void ChangeName(string name)

{

this.name = name;

Console.WriteLine($"Имя изменено на {name}");

Notify?.Invoke(name);

}

}

class X

{

public void XHandler(string name) => Console.WriteLine($"{name} - хорошее имя");

}

class Y

{

public void YHandler(string name) => Console.WriteLine($"{name} - прекрасное имя");

}

Таблица 13.3 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Привет | Имя изменено на Привет  Привет - хорошее имя  Привет - прекрасное имя  Имя изменено на Привет  Привет - хорошее имя |

Источник: собственная разработка

Анализ результатов представлен на рисунке 13.3.

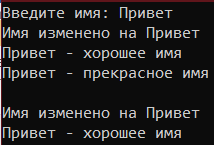


Рисунок 13.3 – Результат работы программы

Источник: собственная разработка