C# Essential

Использование классов и объектов

После урока обязательно

Повторите этот урок в видео формате на ITVDN.com

Проверьте как Вы усвоили данный материал на TestProvider.com

Тема

Использование классов и объектов

Частичные классы

Partial classes

В C# реализована возможность разделить создание класса или метода (структуры, интерфейса) между двумя или более исходными файлами или модулями. Каждый исходный файл содержит определение типа или метода, и все части объединяются при компиляции приложения.

Для разделения класса на несколько частей, используется ключевое слово `partial`.

```
partial class PartialClass
{
    public void MethodFromPart1()
    {
    }
}
```

```
partial class PartialClass
{
    public void MethodFromPart2()
    {
    }
}
```

```
static void Main()
{
    PartialClass instance = new PartialClass();

    instance.MethodFromPart1();
    instance.MethodFromPart2();
}
```

Частичные методы

Partial methods

Частичные методы – это методы, где «прототип» или сигнатура метода определена при создании частичного класса, а реализация выполняется в любой другой (только одной) части этого класса.

```
partial class PartialClass
{

   partial void PartialMethod();

   partial void MyMethod();

}
```

```
partial class PartialClass
{
   partial void PartialMethod()
   {/* … */}

   public void CallPartialMethod()
   {
      PartialMethod();
   }
}
```

```
static void Main()
{
    PartialClass instance = new PartialClass();

    instance.CallPartialMethod();
}
```

Частичные методы

Правила использования

- Частичные методы должны быть определенны только в частичных классах.
- Частичные методы должны быть помечены ключевым словом `partial`.
- Частичные методы всегда являются `private`, попытка явного использования с ними модификатора доступа приведет к ошибке.
- Частичные методы должны возвращать `void`.
- Частичные методы могут быть нереализованными.

Поля только для чтения

Если поле используется с модификатором `readonly`, то присвоение значений таким полям может происходить только при создании поля или в конструкторе того же класса.

```
public readonly string field = "Hello!";
```

`readonly` – это модификатор, который можно использовать только для полей.

Unified Modeling Language

UML (*Unified Modeling Language* – унифицированный язык моделирования) – язык графического описания для объектного моделирования в области разработки программного обеспечения.

UML – это такой же язык, как и C#, Visual Basic, русский, английский или любой другой язык. **UML** – был разработан в 1997 году.

UML – был создан для того, чтобы участники процесса создания программного обеспечения смогли строить модели для визуализации системы, определения её структуры и поведения, сборки системы и документирования решений, принимаемых в процессе разработки.

UML 2.0 – Спецификация UML 2.0 была окончательно согласована в октябре 2004 года.

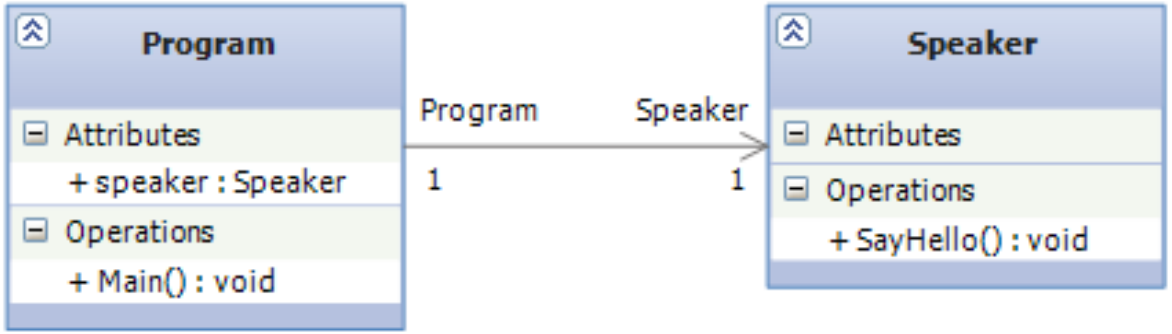
Диаграммы классов

Class diagrams

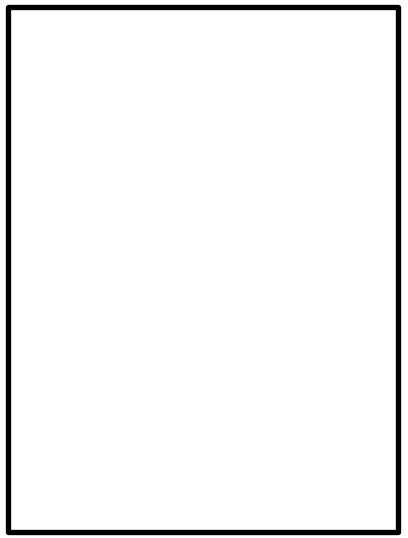
Диаграммы классов используются для изображения классов, а также связей между ними.

Самым важным является показ классов и связей между ними с различных сторон таким способом, чтобы передать наиболее важный смысл.

Диаграмма классов представляет собой статическую модель системы. Диаграмма классов не описывает поведение системы, или то, как взаимодействуют экземпляры классов.



Основные элементы

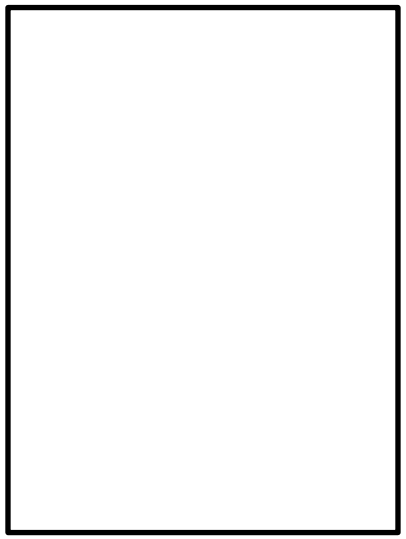


Прямоугольник

Линия

Диаграммы классов

Классификатор



Прямоугольник

Диаграммы классов

Секции классификатора

Секция Имени

Секция Атрибутов

Секция Операций

Секция Обязанностей (необязательная)

Диаграммы классов

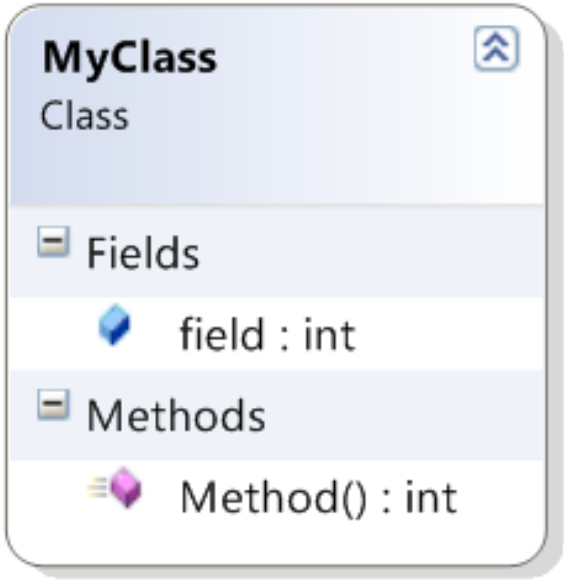
Класс на C#


```
class MyClass
{

    int field;


    int Method() { return 777; }


    // TODO: Вернуть строку "Hello!"

}
```

Диаграммы классов

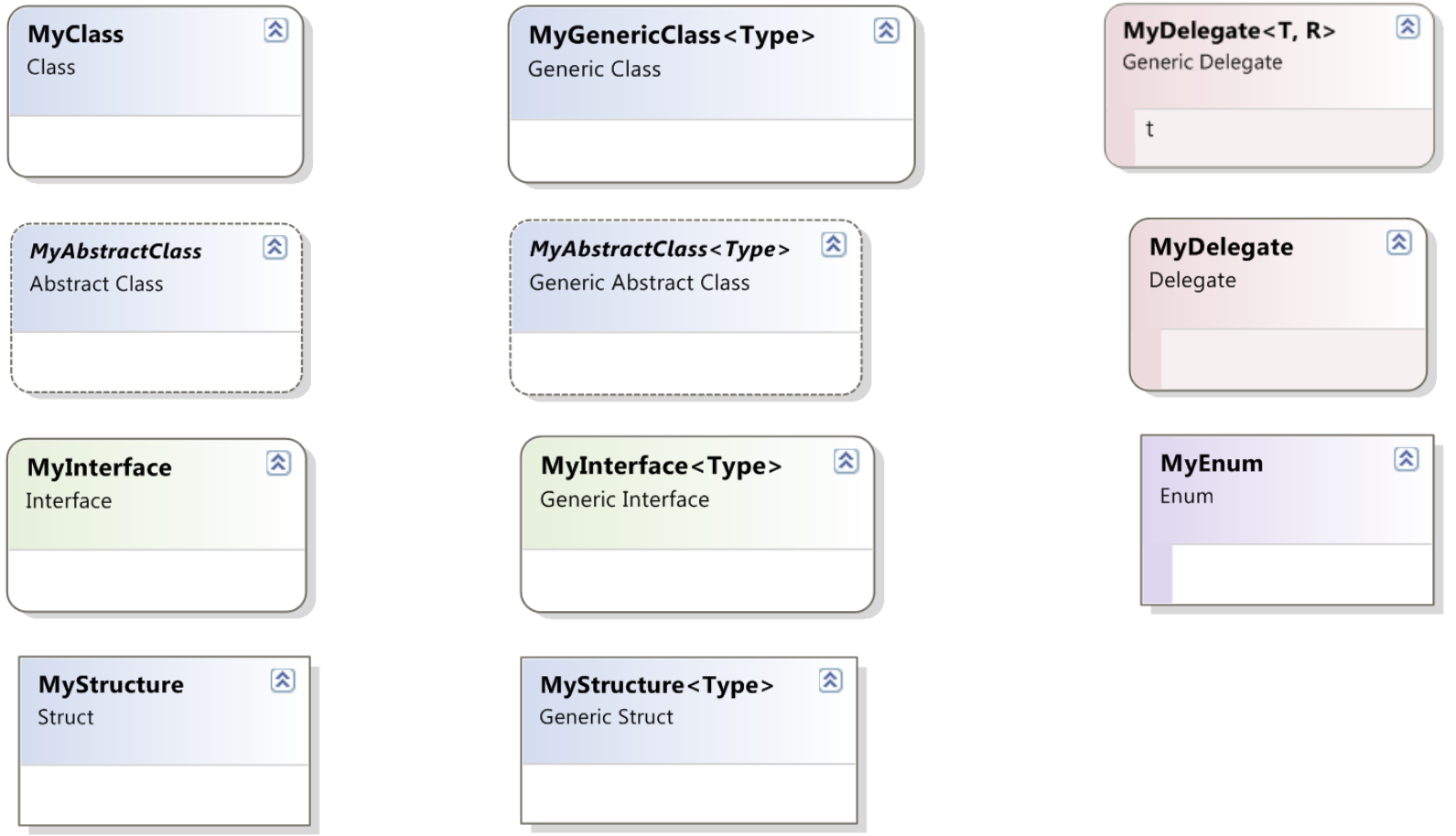
Класс в MS Visual Studio

MyClass
field : int
Method() : int
Обязанности: -Вернуть строку “Hello!” - …



Диаграммы классов

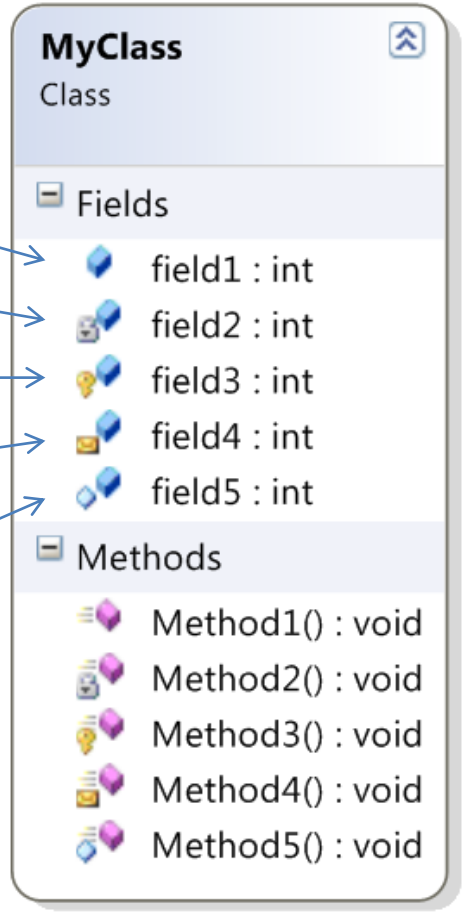
Стереотипы в MS Visual Studio



Диаграммы классов

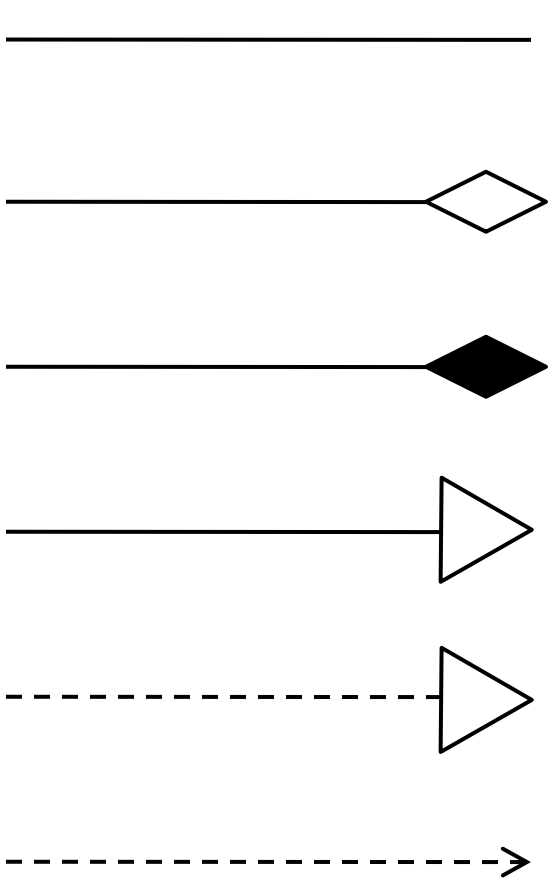
Модификаторы доступа в C# и UML

+	public
-	private
#	protected
~	internal
~#	internal protected



Диаграммы классов

Связи отношений между классами



Ассоциация

Агрегация

Композиция

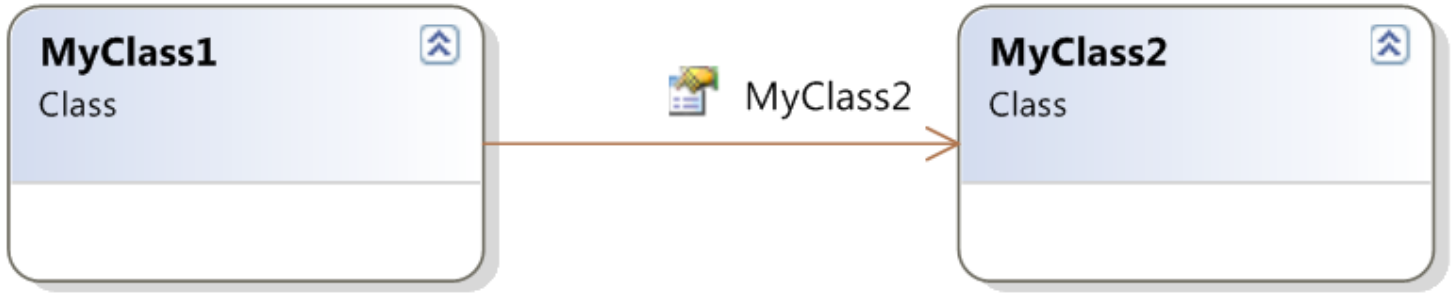
Обобщение

Реализация

Зависимость

Ассоциация

Направленная (Однонаправленная)




```
class MyClass1
{

     public MyClass2 MyObj;

}
```

```
class MyClass2
{



}
```

Ассоциация

Двунаправленная (Ненаправленная)

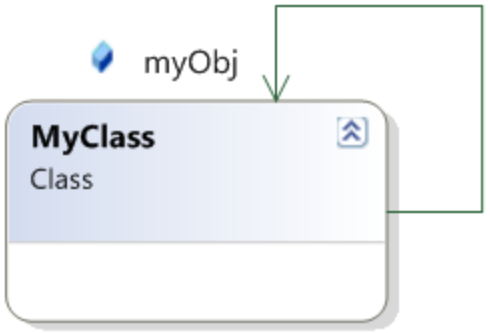

```
class MyClass1
{

     public MyClass2 myObj;

}
```

```
class MyClass2
{

     public MyClass1 myObj;

}
```

Ассоциация

Рефлексивная Ассоциация

```
class MyClass
{

      public MyClass myObj;

}
```



Q&A

Информационный видеосервис для разработчиков программного обеспечения

ITVDN

IT VIDEO DEVELOPERS NETWORK

TestProvider PROMETRIC TEST CENTER CyberBionic systematics Microsoft Partner Silver Learning Windows Azure Cloud Partner EBA European Business Association