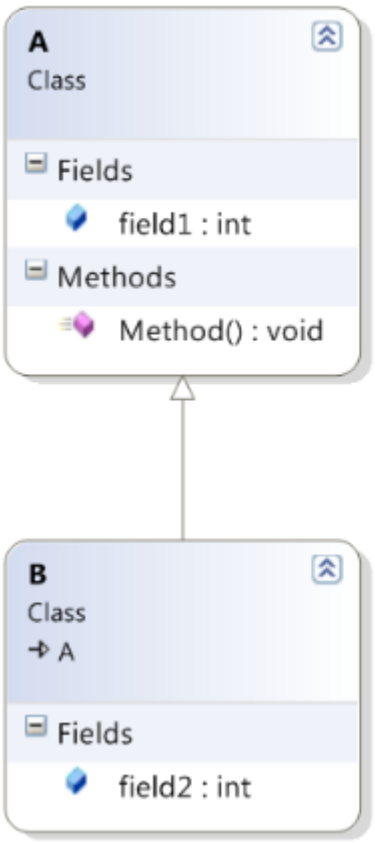
Парадигма ООП

Наследование—механизм объектно-ориентированного программирования (наряду с инкапсуляцией, полиморфизмом и абстракцией), позволяющий описать новый класс на основе уже существующего (родительского), при этом свойства и функциональность родительского класса заимствуются новым классом.




```
class A
{
    public int field1;
    public void Method()
    {
        /* … */
    }
}



class B : A
{
    public int field2;
}
```

```
static void Main()
{
    B b = new B();
    b.field1=5;
    b.field2=8;
    b.Method();
}
```

Недостаток наследования – хрупкий базовый класс.

Хрупкий базовый класс — фундаментальная проблема объектно-ориентированного программирования. Проблема хрупкого базового класса заключается в том, что малейшие правки в деталях реализации базового класса могут привнести ошибку в производные классы. В худшем случае это приводит к тому, что любая успешная модификация базового класса требует предварительного изучения всего дерева наследования, и зачастую невозможна (без создания ошибок) даже в этом случае.

Рекомендуется использовать следующие пары:

- Базовый класс – Производный класс
- Супер класс - Подкласс или (сабкласс)
- Родительский класс - Дочерний класс
- Класс Родитель – Класс Потомок

Обзор и применение модификаторов доступа

Модификаторы доступа – это ключевые слова, задающие доступность члена или типа. При помощи модификаторов доступа можно задавать уровни доступа к членам.

`Public` - доступ к типу или члену возможен из любого другого кода в той же сборке или другой сборке, ссылающейся на него.

`Protected` - доступ к типу или элементу можно получить только из кода в том же классе или структуре, либо в производном классе.

`Private` – доступ к типу или члену можно получить только из кода в том же классе или структуре.

При помощи модификаторов доступа можно задать следующие пять уровней доступности:

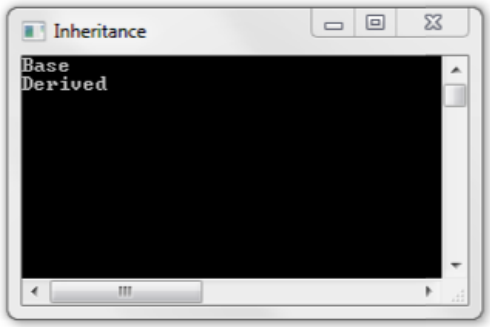
- `public` - доступ к типу или члену возможен из любого другого кода в той же сборке или другой сборке, ссылающейся на него
- `protected` - доступ к типу или элементу можно получить только из кода в том же классе или структуре, либо в производном классе.
- `internal` - доступ к типу или члену возможен из любого кода в той же сборке, но не из другой сборки.
- `protected internal` - доступ ограничен текущей сборкой или типами, которые являются производными от содержащего класса.
- `private` - доступ к типу или члену можно получить только из кода в том же классе или структуре.

Вызов конструктора базового класса

Использование ключевого слово `base` для вызова конструктора базового класса.

```
public class BaseClass
{
    public BaseClass()
    {
        Console.WriteLine("Base");
    }
}

public class DerivedClass : BaseClass
{
    public DerivedClass()
        : base()
    {
        Console.WriteLine("Derived");
    }
}
```



Приведение к базовому типу

Приведение к базовому типу используется для сокрытия реализации членов производного класса.

```
BaseClass instance = new DerivedClass();
```

Переменная **instance** типа `BaseClass` хранит ссылку на экземпляр класса `DerivedClass`.

UpCast и DownCast

UpCast - приведение экземпляра производного класса к базовому типу.

```
BaseClass up = new DerivedClass();
```

DownCast - приведение экземпляра базового типа к производному типу.

```
DerivedClass down = (DerivedClass) up;
```

DownCast невозможен без предварительного UpCast.

Полиморфизм

Полиморфизм —возможность объектов с одинаковой спецификацией иметь различную реализацию.

- Полиморфизм предоставляет подклассу способ определения собственной версии метода, определенного в его базовом классе, с использованием процесса, который называется переопределением метода (method overriding).

- Базовые классы могут определять и реализовывать виртуальные методы, а производные классы могут переопределять их. Это означает, что они предоставляют свои собственные определение и реализацию.

- Во время выполнения, когда клиентский код вызывает метод, среда CLR ищет тип времени выполнения объекта и вызывает это переопределение виртуального метода. Таким образом, в исходном коде можно вызвать метод в базовом классе и вызвать выполнение метода с версией производного класса.

- Если производный класс наследует от базового класса, то он приобретает все методы, поля, свойства и события базового класса. Проектировщик производного класса может выбирать из следующих возможностей:
    1) переопределить виртуальные члены в базовом классе
    2) наследовать метод последнего базового класса без его переопределения
    3) определить новую не виртуальную реализацию этих членов, которая скрывает реализации базового класса.
    - Поля не могут быть виртуальными.
    - Виртуальными могут быть только **методы**, **свойства**, **события** и **индексаторы**.
    - Если в производном классе виртуальный метод переопределяется, то этот член вызывается даже в том случае, если доступ к экземпляру этого класса осуществляется как к экземпляру базового класса.
    - Виртуальные методы и свойства дают возможность производным классам расширять базовый класс, без необходимости использования реализации метода базового класса.
    - Если необходимо, чтобы производный член имел то же имя, что и член базового класса, но не нужно, чтобы он участвовал в виртуальном вызове, можно использовать ключевое слово `new`. Ключевое слово `new` располагается перед возвращаемым типом замещаемого члена класса.
    - Оператор `is` - проверяет совместимость объекта с заданным типом.
    - Если предоставленный объект может быть приведен к предоставленному типу не вызывая исключение, выражение `is` принимает значение `true`.
    - Оператор `as` используется для выполнения преобразований между совместимыми ссылочными типами

- Оператор `as` подобен оператору приведения. Однако, если преобразование невозможно, `as` возвращает значение `null`, а не вызывает исключение
- В общем виде логика работы оператора `as` представляет собой механизм использования оператора `is`, только в сокращенном виде

- Ad-hoc полиморфизм
- Классический (принудительный) полиморфизм:

При применении к классу, модификатор `sealed` запрещает другим классам наследоваться от этого класса.

Модификатор `sealed` можно использовать с методами или свойствами. Это позволяет запретить переопределять виртуальные методы или свойства в производных классах.

MSDN: Полиморфизм

http://msdn.microsoft.com/ru-ru/library/z165t2xk(v=VS.90).aspx

MSDN: Модификаторы доступа (Справочник по C#)

http://msdn.microsoft.com/ru-ru/library/wxh6fsc7(v=VS.90).aspx

MSDN: Наследование (Руководство по программированию на C#)

http://msdn.microsoft.com/ru-ru/library/ms173149.aspx