Модуль 10: Пространства имен, модификатор доступа internal и сборки

Обзор

- Использование пространств имен
- Использование внутренних (internal) классов, методов и данных
- Использование модулей и сборок

Использование пространств имен

- Область видимости
- Разрешение конфликтов имен
- Объявление пространств имен
- Полностью определенные имена
- Объявление директивы using
- Использование альтернативных имен в директиве using
- Рекомендации по именованию пространств имен

Пространство имен

- **Пространство имен определяет декларативную область, которая позволяет отдельно хранить множества имен.**
  - платформа .NET Framework использует пространства имен для организации большинства классов.
    - ✓Пример: библиотека .NET Framework использует пространство имен **System**.

```
System.Console.WriteLine("Hello World!");
```

  - объявление собственного пространства имен поможет в управлении областью действия имен классов и методов в крупных программных проектах

Область видимости

- **Область видимости имени определяется той частью программы, в которой вы можете использовать имя без уточнения**

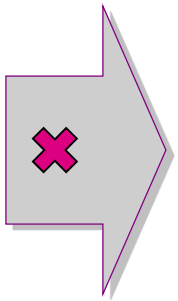
```
public class Bank
{
    public class Account
    {
        public void Deposit(decimal amount)
        {
            balance += amount;
        }
        private decimal balance;
    }
    public Account OpenAccount( ) { ... }
}
```

Разрешение конфликтов имен

- Реальные проекты могут насчитывать в своем составе тысячи классов
- Что будет, если имена каких-то двух классов совпадут?
- Не добавляйте приставку ко всем именам классов

```
// From Vendor A
public class Widget
{ ... }

// From Vendor B
public class Widget
{ ... }
```



```
public class VendorAWidget
{ ... }

public class VendorBWidget
{ ... }
```

Объявление пространств имен

```
namespace VendorA
{
    public class Widget
    { ... }
}
```

```
namespace VendorB
{
    public class Widget
    { ... }
}
```

```
namespace Microsoft
{
   namespace Office
   {
      ...
   }
}
```

Краткая запись

```
namespace Microsoft.Office
{

}
```

Полностью определенные имена

- Полностью определенное имя класса включает в себя пространство имен
- Неопределенные имена классов могут использоваться только в пределах их области видимости

```
namespace VendorA
{
    public class Widget { ... }
    ...
}
class Application
{
    static void Main( )
    {
        Widget w = new Widget( );
        VendorA.Widget w = new VendorA.Widget( );
    }
}
```

Объявление директивы using

- **Позволяет вернуть имена в область их видимости**

```
namespace VendorA.SuiteB
{
    public class Widget { ... }
}
```

```
using VendorA.SuiteB;

class Application
{
    static void Main( )
    {
        Widget w = new Widget( );
    }
}
```

Использование альтернативных имен в директиве using

- Можно создавать альтернативные имена (псевдонимы) для вложенных пространств имен и типов

```
namespace VendorA.SuiteB
{
    public class Widget { ... }
}
```

```
using Widget = VendorA.SuiteB.Widget;

class Application
{
    static void Main( )
    {
        Widget w = new Widget( );
    }
}
```

Рекомендации по именованию пространств имен

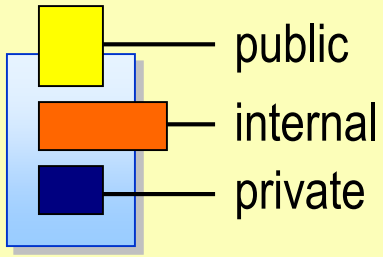
- **Для логического разделения имени используйте технику «Паскаль»**
  - Пример: VendorA.SuiteB
- **Рекомендуется для пространств имен использовать префикс с именем компании**
  - Пример: Microsoft.Office
- **При необходимости используйте имена во множественном числе**
  - Example: System.Collections
- **Избегайте конфликтов имен между пространствами имен и классами**

Использование внутренних (internal) классов, методов и данных

- Зачем нужен модификатор доступа internal?
- Модификатор доступа internal
- Синтаксис
- Пример использования модификатора доступа internal

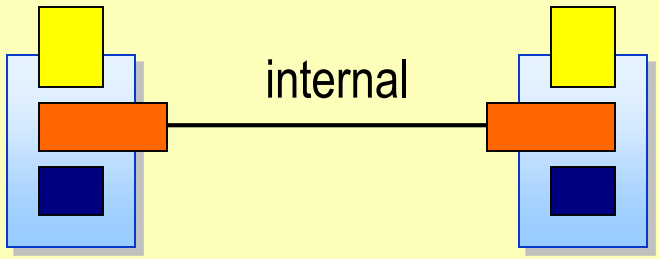
Зачем нужен модификатор доступа internal?

- Маленькие объекты редко используются сами по себе
- Объекты должны взаимодействовать между собой, образуя более крупные объекты
- Необходимо организовать доступ между отдельными объектами



Модификатор доступа internal

- **Сравнение модификаторов доступа**
  - Модификатор доступа public логический
  - Модификатор доступа private логическийl
  - Модификатор доступа internal физический



Синтаксис

```
internal class <outername>
{
    internal class <nestedname> { ... }
    internal <type> field;
    internal <type> Method( ) { ... }

    protected internal class <nestedname> { ... }
    protected internal <type> field;
    protected internal <type> Method( ) { ... }
}
```

protected internal означает **protected** *или* **internal**

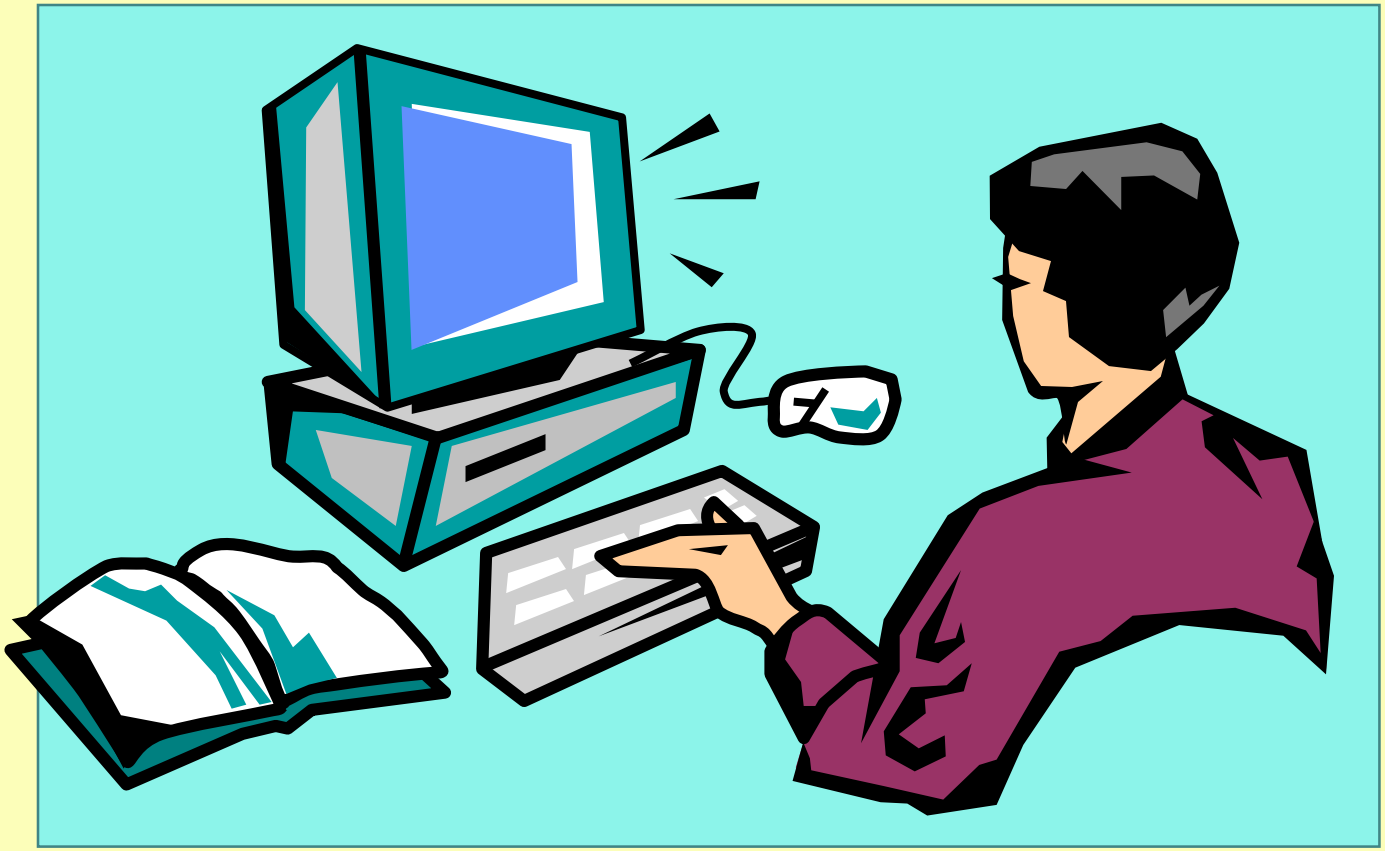
Пример использования модификатора доступа internal

```
public interface IBankAccount { ... }

internal abstract class CommonBankAccount { ... }

internal class DepositAccount: CommonBankAccount,
                               IBankAccount { ... }
public class Bank
{
    public IBankAccount OpenAccount( )
    {
        return new DepositAccount( );
    }
}
```

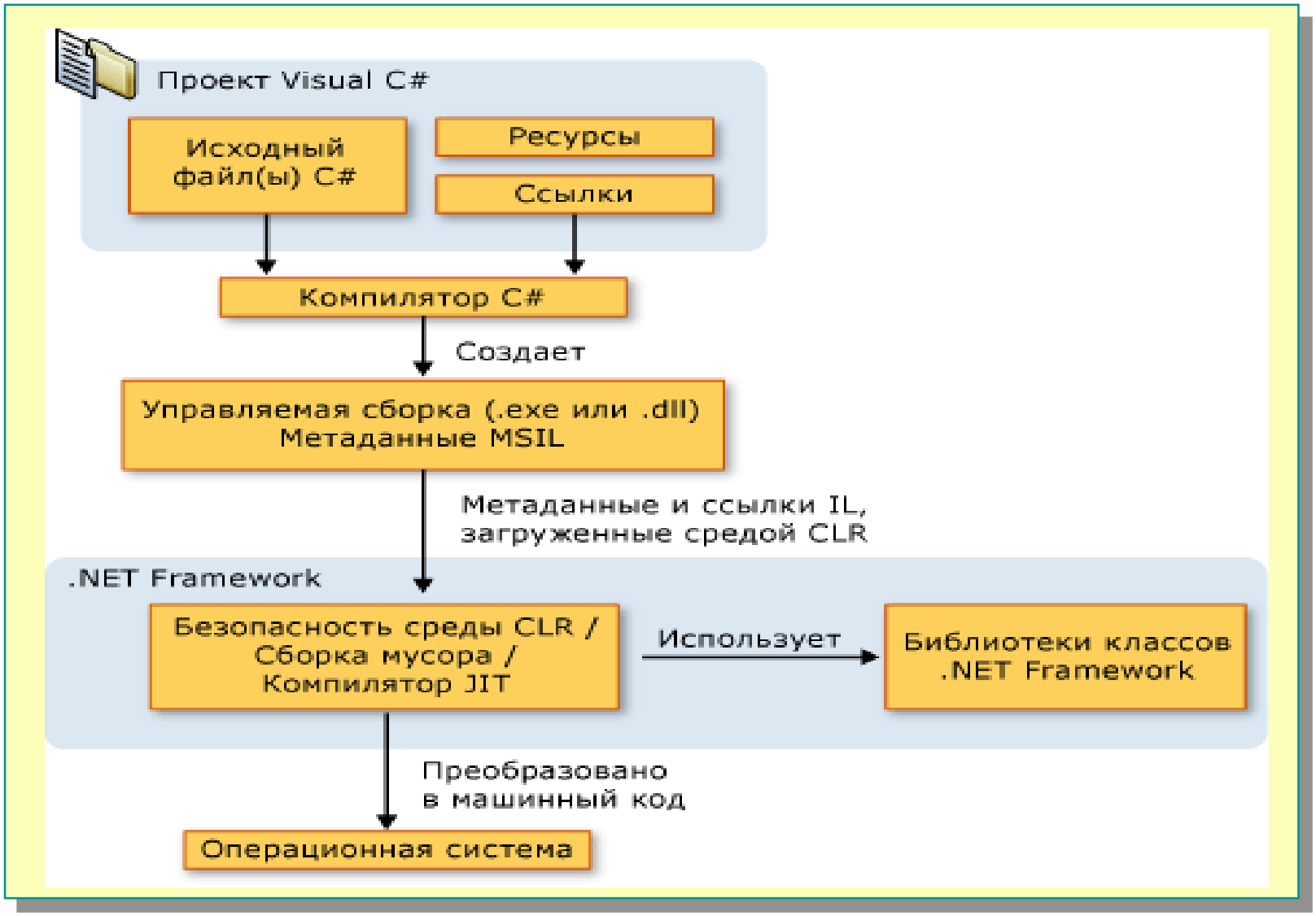
Лабораторная работа 10.1: Использование модификатора доступа internal



Использование модулей и сборок

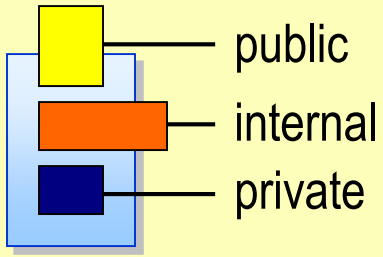
- Создание модулей
- Использование сборок
- Создание сборок
- Сравнение пространств имен и сборок
- Контроль версий

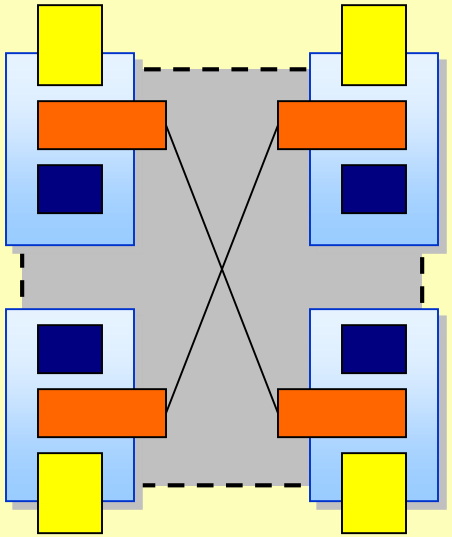
Архитектура платформы .NET Framework



Использование сборок

- **Группа совместно используемых классов**
  - Повторно используемая, безопасная единица развертывания с управляемыми версиями
- **Контроль доступности на уровне сборки**
  - Internal





an assembly
of four classes

Использование модулей

- .cs файлы можно откомпилировать в (.netmodule) управляемый модуль

```
csc /target:module Bank.cs
```

Создание сборок

- **Создание однофайловой сборки**

```
csc /target:library /out:Bank.dll
                    Bank.cs Account.cs
```

- **Создание многофайловой сборки**

```
csc /t:library /addmodule:Account.netmodule
               /out:Bank.dll Bank.cs
```

Сравнение пространств имен со сборками

- **Пространство имен: логическая группировка**
  - Классы из одного пространства имен могут находиться в различных сборках
  - Классы из различных пространств имен могут находиться в одной сборке
- **Сборка: физическая группировка**
  - Сборка MSIL и манифест хранятся вместе
  - Модули сборки и ресурсы могут быть внешними ссылками

Лабораторная работа 10.2: Использование пространств имен и сборок