А.В. Макаров, С.Ю. Скоробогатов, А.М. Чеповский

Учебный курс “CIL и системное программирование в Microsoft .NET”

Лекция 7.

Взаимодействие программных компонентов

Компонентный подход к разработке программного обеспечения

- Программный продукт состоит из взаимодействующих компонентов, которые могут независимо разрабатываться разными группами программистов с применением наиболее подходящих языков
- Негативные явления при отсутствии компонентной технологии:
  - Разработка ведется только на одном языке
  - Программа выглядит как гигантский монолит
  - Появляются чрезмерно универсальные языки программирования, а существующие языки наделяются несвойственными им возможностями
- В этой лекции мы ограничимся рассмотрением взаимодействия компонентов внутри адресного пространства одного процесса

7.1. Обзор компонентных технологий

- Существует множество способов и технологий организации компонентного программирования:
  - Библиотеки подпрограмм
    - Динамические библиотеки
  - Открытые исходные тексты
  - COM и CORBA

7.1.1. Библиотеки подпрограмм

- Библиотеки разрабатываются одной группой разработчиков, а затем могут быть использованы другими программистами в других проектах. Исходный код библиотек может быть закрыт. Недостатки:
  - Двоичный код библиотеки жестко привязан к аппаратной платформе
  - Библиотека рассчитана на использование конкретного компоновщика и может поддерживать ограниченное количество языков
  - Библиотеки плохо подходят для объектно-ориентированных языков, так как не содержат никакой информации о типах
- Динамические библиотеки могут использоваться из большего диапазона языков программирования. В остальном они не лучше, чем обычные библиотеки подпрограмм.

7.1.2. Открытые исходные тексты

- Компоненты распространяются в виде исходных текстов. Их можно переносить с платформы на платформу путем перекомпиляции. Кроме того, они содержат информацию о типах, что позволяет использовать объектно-ориентированные возможности. Недостатки:
  - Если компоненты написаны на разных языках, то соединить их вместе не так-то просто
  - Открытые компоненты почти всегда имеют так называемые "заразные" лицензии (например, GNU Public License – GPL). Такие лицензии требуют, чтобы программы, использующие эти компоненты, сами распространялись в виде открытых исходников

7.1.3. Технологии COM и CORBA

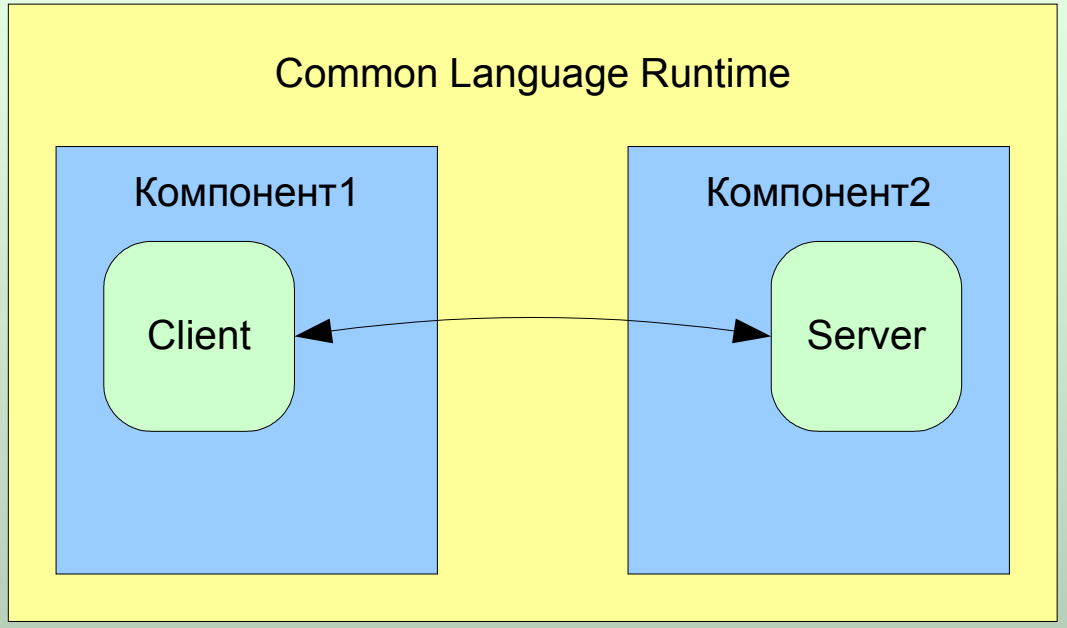
- COM - Component Object Model
- CORBA - Common Object Request Broker Architecture
- Основные особенности:
  - Метаданные описываются на языке IDL (Interface Definition Language)
  - Определен двоичный стандарт для передачи данных между компонентами

Взаимодействие двух объектов через COM или CORBA

7.2. Взаимодействие компонентов в среде .NET

- Особенности платформы .NET, позволяющие среде выполнения автоматически обеспечивать взаимодействие компонентов вне зависимости от того, на каком языке они написаны:
  - Формат сборок .NET определяется спецификацией CLI и не зависит от языка программирования
  - Представление данных в памяти определяется CTS
  - Выполнение любой программы управляется средой выполнения CLR

Взаимодействие двух объектов в среде .NET



- Передача сообщения от объекта Client объекту Server сводится к простому вызову соответствующего метода объекта Server, то есть в среде .NET не нужны никакие объекты-заглушки

Взаимодействия компонентов как импорт сборок

- Компоненты на платформе .NET представляют собой сборки .NET
- Сборка .NET может статически импортировать любую другую сборку и свободно использовать типы, экспортируемые из этой сборки. Для этого информация об импортируемой сборке заносится в таблицу метаданных AssemblyRef, информация о каждом импортируемом типе заносится в таблицу TypeRef, а информация о каждом импортируемом методе и поле заносится в таблицу MemberRef
- Кроме того, сборка может импортироваться динамически через рефлексию.

7.2.1. Видимость и контроль доступа

- Метаданные в сборках .NET содержат полную информацию о типах. При этом каждый тип может экспортироваться или не экспортироваться из сборки, а каждый член типа (метод, поле, свойство и т.д.) должен быть объявлен с определенным значением флага доступа

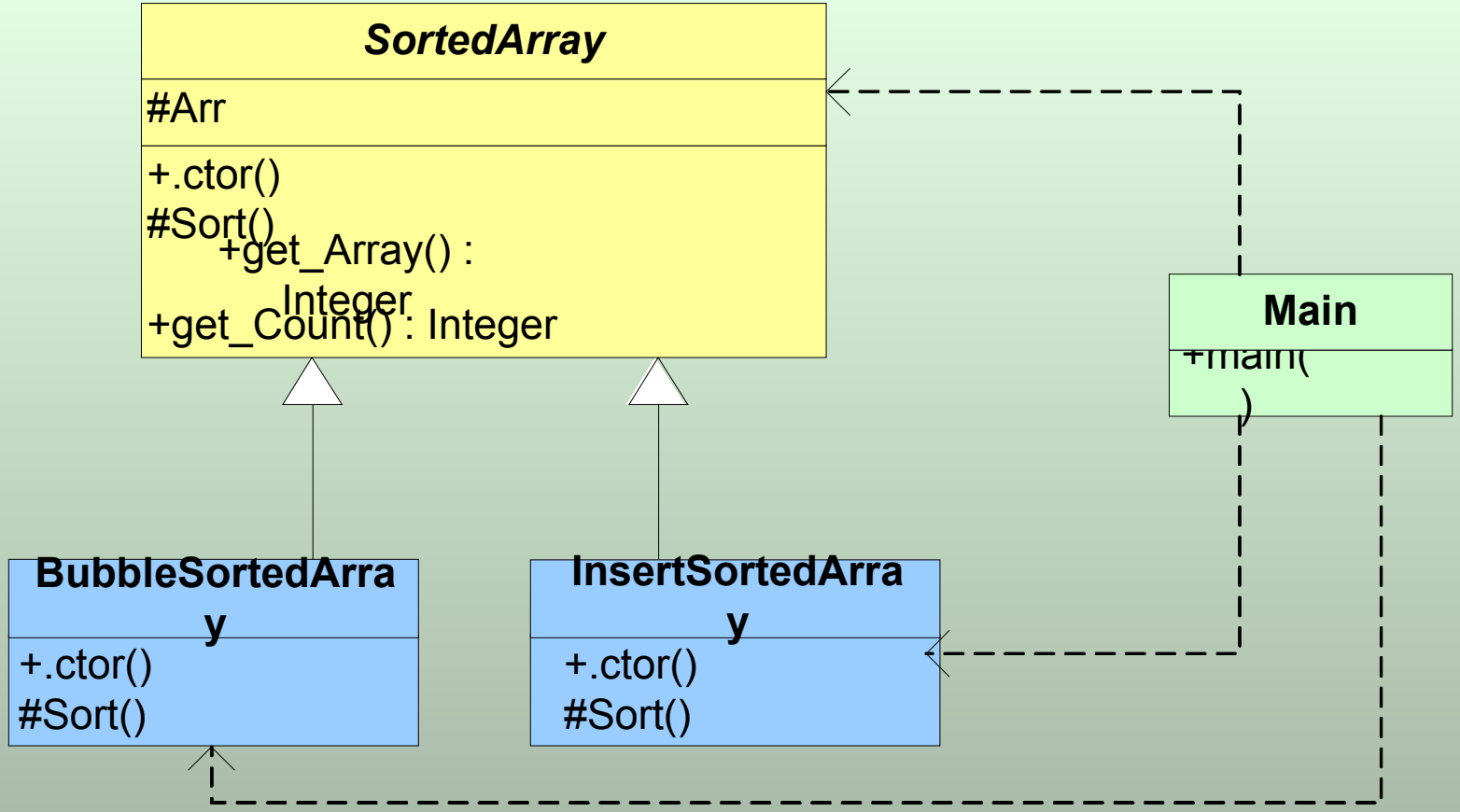
Флаги видимости для типов

Флаг	Значение	Описание
`NotPublic`	`0x00000000`	Тип не экспортируется из сборки.
`Public`	`0x00000001`	Тип экспортируется из сборки.
`NestedPublic`	`0x00000002`	Вложенный тип доступен везде.
`NestedPrivate`	`0x00000003`	Вложенный тип доступен только внутри того типа, в который он вложен.
`NestedFamily`	`0x00000004`	Вложенный тип доступен наследникам того типа, в который он вложен.
`NestedAssembly`	`0x00000005`	Вложенный тип доступен везде внутри сборки.
`NestedFamAndAssem`	`0x00000006`	Вложенный тип доступен наследникам того типа, в который он вложен, но только внутри сборки.
`NestedFamOrAssem`	`0x00000007`	Вложенный тип доступен везде внутри сборки, и, кроме того, наследникам того типа, в который он вложен.

Флаги доступа для членов типа

Флаг	Значение	Описание
`CompilerControlled`	`0x00000000`	Доступ контролируется компилятором.
`Private`	`0x00000001`	Доступен только внутри типа.
`FamAndAssem`	`0x00000002`	Доступен наследникам типа, объявленным внутри сборки.
`Assembly`	`0x00000003`	Доступен только внутри сборки.
`Family`	`0x00000004`	Доступен наследникам типа.
`FamOrAssem`	`0x00000005`	Доступен внутри сборки, а также наследникам типа.
`Public`	`0x00000006`	Доступен везде.

7.2.2. Пример межъязыкового взаимодействия



Абстрактный класс SortedArray (Visual Basic .NET)

```
Public MustInherit Class SortedArray
    Protected Arr() As Integer
    Protected MustOverride Sub Sort()
    Public Sub New(ByVal A() As Integer)
        Dim i As Integer
        Arr = New Integer(A.Length - 1) {}
        For i = 0 To A.Length - 1
            Arr(i) = A(i)
        Next
        Sort()
    End Sub
    Default Public ReadOnly Property Array(ByVal Index As Integer) As Integer
        Get
            Return Arr(Index)
        End Get
    End Property
    Public ReadOnly Property Count() As Integer
        Get
            Return Arr.Length
        End Get
    End Property
End Class
```

Класс BubbleSortedArray (Visual C++ with Managed Extensions)

```
using namespace VBLib;

public __gc class BubbleSortedArray: public SortedArray
{
protected:
    void Sort()
    {
        for (int i = Arr->Length, flag = 1; i > 1 && flag; i--)
        {
            flag = 0;
            for (int j = 0; j < i-1; j++)
                if (Arr[j] < Arr[j+1])
                {
                    int tmp = Arr[j];
                    Arr[j] = Arr[j+1];
                    Arr[j+1] = tmp;
                    flag = 1;
                }
        }
    }
public:
    BubbleSortedArray(int A __gc []): SortedArray(A) { }
};
```

Класс InsertSortedArray (Visual C#)

```
using VBLib;

public class InsertSortedArray: SortedArray
{
    protected override void Sort()
    {
        for (int i = 0; i < Arr.Length-1; i++)
        {
            int max = i;
            for (int j = i+1; j < Arr.Length;
j++)
                if (Arr[j] > Arr[max])
                    max = j;
            int tmp = Arr[i];
            Arr[i] = Arr[max];
            Arr[max] = tmp;
        }
    }

    public InsertSortedArray(int[] A): base(A)
{ }
}
```

Главная программа (Visual J#)

```
package JsApp;

import VBLib.SortedArray;

public class Main
{
    public static void main(String[] args)
    {
        int A[] = new int[] { 5, 1, 6, 0, -4, 3 };
        SortedArray SA1 = new BubbleSortedArray(A),
                    SA2 = new InsertSortedArray(A);

        for (int i = 0; i < SA1.get_Count(); i++)
            System.out.print(""+SA1.get_Array(i)+" ");
        System.out.println();

        for (int i = 0; i < SA2.get_Count(); i++)
            System.out.print(""+SA2.get_Array(i)+" ");
        System.out.println();
    }
}
```

7.3. Общая спецификация языков

- Для того чтобы любую библиотеку классов можно было использовать из любого языка платформы .NET, разработчики .NET придумали общую спецификацию языков (Common Language Specification – CLS)
- В спецификации CLS оговариваются правила, которым должны следовать разработчики языков и библиотек. Она описывает некоторое подмножество общей системы типов, и если некий язык реализует хотя бы это подмножество, а библиотека использует только входящие в это подмножество типы, то такая библиотека может быть использована из этого языка

Терминология CLS

- Библиотеки, соответствующие спецификации CLS, называются средами (frameworks), но мы будем называть их CLS-библиотеками
- Компиляторы, которые генерируют код, из которого можно получить доступ к CLS-библиотекам, называются потребителями (consumers)
- Компиляторы, которые являются потребителями, но, к тому же, способны создавать новые CLS-библиотеки, называются расширителями (extenders)

Основные правила CLS

- Спецификация распространяется только на доступные извне части экспортируемых из библиотеки типов
- Упакованные типы-значения, неуправляемые указатели и типизированные ссылки не должны использоваться
- Регистр букв в идентификаторах не имеет значения
- Методы не должны иметь переменного количества параметров.
- Глобальные поля и методы не поддерживаются спецификацией.
- Объекты исключений должны наследовать от System.Exception.