Programovací jazyk C#: niekoľko zaujímavých čŕt

Poznámky k prednáškam z predmetu Objektovo-orientované programovanie

Valentino Vranić
http://fiit.sk/~vranic/, vranic@stuba.sk
Ústav informatiky, informačných systémov a softvérového inžinierstva
Fakulta informatiky a informačných technológií
Slovenská technická univerzita v Bratislave

25. apríl 2016

Obsah

1 Programovací jazyk C#

Programovací jazyk C#

- Programovací jazyk C# vytvorila spoločnosť Microsoft
- Dostupný je od roku 2002 – súčasná verzia je 6.0
- Na vývoj v C# sa používa prostredie Visual Studio – jazyk v spojení s týmto prostredím sa označuje ako Visual C#
- Využíva rámec .NET
- Kompilovaný kód sa vykonáva na CLR (Common Language Runtime)
- C# je v mnohom podobný Jave
- Často je vnímaný ako odpoveď Microsoftu na Javu
- Má však rad špecifických čŕt

2 Vlastnosti

Vlastnosti (properties)

- Bola identifikovaná potreba transparentne pristupovať k atribútom a metódam objektov (Eiffel)
- C# poskytuje mechanizmus označený ako vlastnosť – `property`
- Tento mechanizmus umožňuje oddeliť vnútornú reprezentáciu atribútu od toho, ako sa javí navonok
- Triviálny prípad:[1]

```
class TimePeriod
{
    private double _seconds;
    public double Seconds
    {
        get { return _seconds; }
        set { _seconds = value; }
    }
}
```

Autoimplementácia vlastností

- Pre triviálny prípad stačí aj automiplementácia

```
class TimePeriod2
{
    public double Hours { get; set; }
}
```

[1] get (C# Reference), http://msdn.microsoft.com/en-us/library/ms228503.aspx

Odlišná vnútorná reprezentácia vlastností

```
class TimePeriod
{
    private double seconds;

    public double Hours
    {
        get { return seconds / 3600; }
        set { seconds = value * 3600; }
    }
}

class Program
{
    static void Main()
    {
        TimePeriod t = new TimePeriod();

        // Assigning the Hours property causes the 'set' accessor to be called.
        t.Hours = 24;

        // Evaluating the Hours property causes the 'get' accessor to be called.
        System.Console.WriteLine("Time in hours: " + t.Hours);
    }
}
// Output: Time in hours: 24
```

Properties (C# Programming Guide), http://msdn.microsoft.com/en-us/library/x9fsa0sw.aspx

3 Zástupcovia

Zástupcovia (delegates) (1)

- V C# je možné definovať zástupcu metódy – `delegate`
- Zástupcovi možno dynamický priradiť skutočnú metódu
- Príklad na nasledujúcich slajdoch[2]

Zástupcovia (delegates) (2)

```
// Declare delegate -- defines required signature:
delegate double MathAction(double num);

class DelegateTest
{
    // Regular method that matches signature:
    static double Double(double input)
    {
        return input * 2;
    }

    static void Main()
    {
        // Instantiate delegate with named method:
        MathAction ma = Double;
```

[2] delegate (C# Reference), http://msdn.microsoft.com/en-us/library/900fyy8e.aspx

```
        // Invoke delegate ma:
        double multByTwo = ma(4.5);
        Console.WriteLine("multByTwo: {0}", multByTwo);

        // Instantiate delegate with anonymous method:
        MathAction ma2 = delegate(double input)
        {
            return input * input;
        };

        double square = ma2(5);
        Console.WriteLine("square: {0}", square);

        // Instantiate delegate with lambda expression
        MathAction ma3 = s => s * s * s;
        double cube = ma3(4.375);

        Console.WriteLine("cube: {0}", cube);
    }
    // Output:
    // multByTwo: 9
    // square: 25
    // cube: 83.740234375
}
```

Anonymné metódy

- Inštancia zástupcu môže byť vytvorená priamo kódom bez jeho pomenovania ako metódy
- Príklad:[3]

```
// Create a delegate.
delegate void Del(int x);

// Instantiate the delegate using an anonymous method.
Del d = delegate(int k) { /* ... */ };
```

- Iný príklad:

```
// Create a handler for a click event.
button1.Click += delegate(System.Object o, System.EventArgs e)
                 { System.Windows.Forms.MessageBox.Show("Click!"); };
```

4 Rozšírujúce metódy (extension methods)

- C# umožňuje definovať statickú metódu pre typ mimo tohto typu
- Typ je sprostredkovaný prostredníctvom parametra tejto metódy, ktorému predchádza modifikátor **this**
- Príklad:[4]

```
namespace ExtensionMethods
{
    public static class MyExtensions
    {
```

[3] Anonymous Methods (C# Programming Guide), http://msdn.microsoft.com/en-us/library/0yw3tz5k.aspx
[4] Extension Methods (C# Programming Guide), https://msdn.microsoft.com/en-us/library/bb383977.aspx

```
        public static int WordCount(this String str)
        {
            return str.Split(new char[] { ' ', '.', '?' },
                             StringSplitOptions.RemoveEmptyEntries).Length;
        }
    }
}
```

- Použitie – ako keby metóda bola súčasťou daného typu:

```
using ExtensionMethods;

...

    string s = "Hello Extension Methods";
    int i = s.WordCount();
```

5 Prekonávanie

Explicitné prekonávanie

- Prekonávanie sa musí zapnúť – `virtual`
- Prekonávajúca metóda sa musí explicitne takto označiť – `override`

6 Menné priestory

Menné priestory

- `namespace`
- Analogické k balíkom v Jave
- Prvky jedného menného priestoru môžu byť v rôznych súboroch
- Jeden súbor môže obsahovať viac menných priestorov
- `using`

```
namespace SampleNamespace
{
    class SampleClass
    {
        public void SampleMethod()
        {
            System.Console.WriteLine(
              "SampleMethod inside SampleNamespace");
        }
    }
}
```

Namespaces (C# Programming Guide), http://msdn.microsoft.com/en-us/library/0d941h9d.aspx

7 Parciálne triedy a metódy

- Trieda môže byť implementovaná vo viacerých súboroch – musí byť označená kľúčovým slovom `partial`
- Pri použití kľúčového slova `partial`, možno uviesť len signatúru metódy
- Takáto metóda môže byť inde implementovaná alebo nie (zanikne pri kompilácii)

```
namespace PM
{
    partial class A
    {
        partial void OnSomethingHappened(string s);
    }

    // This part can be in a separate file.
    partial class A
    {
        // Comment out this method and the program
        // will still compile.
        partial void OnSomethingHappened(String s)
        {
            Console.WriteLine("Something happened: {0}", s);
        }
    }
}
```

partial (Method) (C# Reference), http://msdn.microsoft.com/en-us/library/6b0scde8.aspx

8 Udalosti

Udalosti (events)

- Mechanizmus vytvárania a spracovania udalosti
- Kľúčové slovo `event`
- Zástupca zabezpečuje spojenie udalosti a bloku spracovania

Príklad udalosti

```
using System;
namespace wildert
{
    public class Metronome
    {
        public event TickHandler Tick;
        public EventArgs e = null;
        public delegate void TickHandler(Metronome m, EventArgs e);
        public void Start()
        {
            while (true)
            {
                System.Threading.Thread.Sleep(3000);
                if (Tick != null)
```

```
            {
                Tick(this, e);
            }
        }
    }
}

public class Listener
{
    public void Subscribe(Metronome m)
    {
        m.Tick += new Metronome.TickHandler(HeardIt);
    }
    private void HeardIt(Metronome m, EventArgs e)
    {
        System.Console.WriteLine("HEARD IT");
    }
}

class Test
{
    static void Main()
    {
        Metronome m = new Metronome();
        Listener l = new Listener();
        l.Subscribe(m);
        m.Start();
    }
}
}
```

The Simplest C# Events Example Imaginable, http://www.codeproject.com/Articles/11541/The-Simplest-C-Events-Example-Imaginable

9 Ďalšie mechanizmy

Ďalšie mechanizmy nepodporované v Jave

- Preťaženie operátorov (podobne ako v C++)
- Struct – implicitne zapečatená (sealed) trieda (finálna trieda v terminológii jazyka Java)
- Implicitný typ `var`
- Preprocesorové direktívy (preprocessor directives) – používané aj v jazyku C
- Rozsiahla podpora viacnitovosti s uplatnením pokročilých mechanizmov na synchronizáciu

Mechanizmy známe z Javy

- Rozhrania (interfaces) môžu navyše predpisovať vlastnosti (properties) – podobné možnosti predpísať implementáciu atribútu v UML
- Modifikátory prístupu (zapuzdrenie) sú o niečo rozšírené (`internal` a **protected** `internal`)
- Vhniezdené typy vrátane anonymných tried

- Výnimky – ale žiadne nie sú kontrolované
- C# podporoval generickosť a lambda výrazy skôr ako Java

Analogické kľúčové slová k niektorým v Jave

- **sealed** – v Jave **final**
- **using** – v Jave **import**
- **readonly** a **const** – v Jave **final** (**const** je kľúčové slovo v Jave, ale nie je využité)
- Niektoré kľúčové slova sú v C# rezervované len v určitom kontexte[5]

10 Sumarizácia

- C# je pre programátora v Jave pomerne čitateľný
- Niektoré odlišnosti sú na úrovni zámeny kľúčových slov
- C# však poskytuje viacero úplne iných mechanizmov

[5] Contextual Keywords (C# Reference), http://msdn.microsoft.com/en-us/library/the35c6y.aspx