

Sistemi Distribuiti

Corso di Laurea in Ingegneria



Prof. Paolo Nesi

PARTE 9: C#



Department of Systems and Informatics
University of Florence
Via S. Marta 3, 50139, Firenze, Italy
tel: +39-055-4796523, fax: +39-055-4796363

Lab: DISIT, Sistemi Distribuiti e Tecnologie Internet

nesi@dsi.unifi.it nesi@computer.org
www: <http://www.dsi.unifi.it/~nesi>





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 1



C# – Introduction

- The first component oriented language in the C/C++ family
- Everything really is an object
- Next generation robust and durable software
- Preservation of investment





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 2

Microsoft
.net

C# – A component oriented language

- C# is the first “component oriented” language in the C/C++ family
- Component concepts are first class:
 - Properties, methods, events
 - Design-time and run-time attributes
 - Integrated documentation using XML
- Enables one-stop programming
 - No header files, IDL, etc.
 - Can be embedded in web pages





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006

3

Microsoft
.net


C# – Everything is an Object

- Traditional views
 - C++, Java: Primitive types are **magic** and do not interoperate with objects
 - Smalltalk, Lisp: Primitive types are objects, but at great performance cost
- C# unifies with no performance cost
 - Deep simplicity throughout system
- Improved extensibility and reusability
 - New primitive types: Decimal, SQL...
 - Collections, etc., work for **all** types





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006

4




C# – Features

- Garbage collection
 - ♣ No memory leaks and stray pointers
- Exceptions
 - ♣ Error handling is not an afterthought
- Type-safety
 - ♣ No uninitialized variables, unsafe casts
- Versioning
 - ♣ Pervasive versioning considerations in all aspects of language design





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006

5




C# – Learning from the others

- C++ heritage
 - ♣ Namespaces, enums, unsigned types, pointers (in unsafe code), etc.
 - ♣ No unnecessary sacrifices
- Interoperability
 - ♣ What software is increasingly about
 - ♣ MS C# implementation talks to XML, SOAP, COM, DLLs, and any .NET language





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006

6




C# and OOP

- C# is designed for the .NET Framework
 - ♣ The .NET Framework is Object Oriented
- In C#
 - ♣ Your access to the OS is through objects
 - ♣ You have the ability to create first class objects
 - ♣ The FCL is designed for extension and integration by your code



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006



7



Hello World

```
using System;

class Hello
{
    static void Main() {
        Console.WriteLine("Hello world");
    }
}
```





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006

8

Microsoft .net

C# Program Structure

- Namespaces
 - Contain types and other namespaces
- Type declarations
 - Classes, structs, interfaces, enums, and delegates
- Members
 - Constants, fields, methods, properties, indexers, events, operators, constructors, destructors
- Organization
 - No header files, code written "in-line"
 - No declaration order dependence



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006

9

Microsoft .net



C# Program Structure

```
using System;

namespace System.Collections
{
    public class Stack
    {
        Entry top;

        public void Push(object data) {
            top = new Entry(top, data);
        }

        public object Pop() {
            if (top == null) throw new InvalidOperationException();
            object result = top.data;
            top = top.next;
            return result;
        }
    }
}
```





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006

10

Microsoft .net

C# - Namespaces

- Code is structured in namespaces
 - ♣ Orthogonal to code-files and assemblies
 - ♣ Namespaces can be nested
- Full name of a type: namespace.typename
 - ♣ **MySpace.Subset1.HelloWorld**



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006

11

Microsoft .net



Namespaces: Example

```
• using System;
• namespace MySpace.Subset1
{
  public class HelloWorld
  {
    public static void Main(string[] argv)
    {
      Console.WriteLine("Hello World!");
    }
  }
}
```

import the System namespace

Same as :
namespace MySpace {
 namespace Subset1 {

from the System namespace





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006

12

Microsoft .net

.NET Types

- All types are compatible with `object` (`System.Object`)
- Reference types (classes, arrays, delegates)
 - ♣ Stored on heap
 - ♣ Assignment copies reference
 - ♣ Initialized with `null`
- Value types (simple types, structs, enums)
 - ♣ Stored on stack
 - ♣ Assignment copies value
 - ♣ Initialized with `0`, `false`, `'\0'`



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006



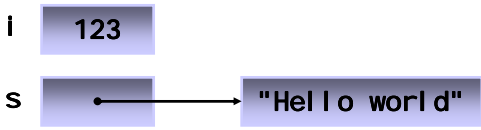
13

Microsoft .net

Type System

- Value types
 - ♣ Directly contain data
 - ♣ Cannot be null
- Reference types
 - ♣ Contain references to objects
 - ♣ May be null

```
int i = 123;  
string s = "Hello world";
```





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006

14

Microsoft .net

Type System

- Value types
 - ♣ Primitives `int i;`
 - ♣ Enums `enum State { Off, On }`
 - ♣ Structs `struct Point { int x, y; }`
- Reference types
 - ♣ Classes `class Foo: Bar, IFoo {...}`
 - ♣ Interfaces `interface IFoo: IBar {...}`
 - ♣ Arrays `string[] a = new string[10];`
 - ♣ Delegates `delegate void Empty();`





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 15

Microsoft .net

Classes

- Single inheritance
- Multiple interface implementation
- Use of ":" for both extends and implements
- Class members
 - ♣ Constants, fields, methods, properties, indexers, events, operators, constructors, destructors
 - ♣ Static and instance members
 - ♣ Nested types
- Member access
 - ♣ public, protected, internal, private



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 16



Microsoft .net

Defining Classes

```
class Name: BaseType{  
    // Members  
}
```

```
Namespace NameName{  
    class Name: BaseType{  
    }  
}
```

```
class MyType{  
    public static String someTypeState;  
    public Int32 x;  
    public Int32 y;  
}
```



 

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 17

Microsoft .net

Classes

- Are reference types
- System.Object (**object**) is the base class of all classes
- Inheritance
 - ♣ Single for implementation
 - ♣ Multiple for interfaces
- Methods are non-virtual by default!



 

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 18

Microsoft .net

Example: Classes

- `public interface IFoo`
- {
- `void Bar(int x);`
- }
- `public class A : IFoo`
- {
- `public void Bar(int x) { ... }`
- }
- `public class B : A`
- {
- ...
- }





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006

19

Microsoft .net

Classes Accessibility

- In C#, `private` is the default accessibility
- Accessibilities options
 - `public` – Accessible to all
 - `private` – Accessible to containing class
 - `protected` – Accessible to containing or derived classes
 - `internal` – Accessible to code in same assembly
 - `protected internal` – means `protected` or `internal`
- Classes can be marked as `public` or `internal`
 - By default they are `private`
 - Accessible only to code in the same source module





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006

20

Microsoft .net

Type Members in C#

- Fields
 - ♣ The state of an object or type
- Methods
 - ♣ Constructors
 - ♣ Functions
 - ♣ Properties (smart fields)
- Members come in two basic forms
 - ♣ Instance – per object data and methods
 - Default
 - ♣ Static – per type data and methods
 - Use the `static` keyword



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006



21

Microsoft .net

Methods

- Declared inline with type definition

```
class MyType{  
    public Int32 SomeMethod(){  
        return x;  
    }  
  
    public static void StaticMethod(){  
        // Do something  
    }  
}
```



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006

22

Microsoft .net

Methods: Parameters I

- Call-by-value
 - ♣ Formal parameter is copy of actual parameter
 - ♣ `int Double(int i) { return 2*i; }`
- Call-by-reference
 - ♣ Formal parameter is alias (address, ref.) for the actual parameter
 - ♣ `void Double(ref int i) { i = 2*i; }`
 - ♣ `int a = 5; Double(ref a);`

Aliasing must be done explicitly.

i is an alias. The result is assigned to the variable that the alias points to.

23


Microsoft .net

Methods: Parameters II

- Out-parameters
 - ♣ Same as call-by-reference but parameter may not be initialized
 - ♣ `void Double(int i, out int d) { d = 2*i; }`
 - ♣ `int a = 5; Double(a, out a);`
- CbV and CbR are orthogonal to value-types
- CbR is handy when methods yield more than one result
 - ♣ `void ParseNameString(string name, out string first, out string last) { ... }`



The value of d cannot be accessed before something has been assigned to the aliased variable.

24




Methods: Parameters III

- Variable parameter list
 - ♣ Array at the end of the parameter-list
 - ♣ `void ChargePhaserBanks(params int[] banks) {
 foreach (int b in banks) Charge(b);
}`
 - ♣ `ChargePhaserBanks(1, 7, 9);` is the same as
 - ♣ `ChargePhaserBanks(new int[] {1, 7, 9});`
- Extremely useful:
 - ♣ `Console.WriteLine("{0}, {1}", x, y);`

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 25



Instance Constructors



- Constructors are used to initialize fields
- You can implement simpler constructors in terms of more complex ones with the `this` keyword (suggested)

```
class Point{
    Int32 x;
    Int32 y;

    public Point(): this(0, 0){}

    public Point(Int32 x, Int32 y){
        this.x = x;
        this.y = y;
    }
}
```

- You can indicate which base constructor to call
 - ♣ Use the `base` keyword






Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 26

Microsoft
.net

Type (static) Constructors

- Type constructors are used to initialize `static` fields for a type
- Only one static constructor per type
 - Called by the Common Language Runtime
 - Guaranteed to be called before any reference to the type or an instance of the type
 - Must have no parameters
- Use the `static` keyword to indicate a type constructor





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 27

Microsoft
.net

Structs

- Like classes, except
 - Stored in-line, not heap allocated
 - Assignment copies data, not reference
 - No inheritance
- Ideal for light weight objects
 - Complex, point, rectangle, color
 - int, float, double, etc., are all structs
- Benefits
 - No heap allocation, less GC pressure
 - More efficient use of memory



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 28

Microsoft .net

Classes And Structs

```

class CPoint { int x, y; ... }
struct SPoint { int x, y; ... }

CPoint cp = new CPoint(10, 20);
SPoint sp = new SPoint(10, 20);
    
```

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 29

Microsoft .net

Interfaces

- Multiple inheritance
- Can contain methods, properties, indexers, and events
- Private interface implementations
- Your types can implement interfaces
 - Must implement all methods in the interface
- Interfaces can contain methods but no fields
- Constructors are not supported in interfaces

```

interface IDataBound
{
    void Bind(IDataBinder binder);
}

class EditTextBox: Control, IDataBound
{
    void IDataBound.Bind(IDataBinder binder) {...}
}
    
```

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 30

Microsoft
.net

Enums

- Strongly typed
 - No implicit conversions to/from int
 - Operators: +, -, ++, --, &, |, ^, ~
- Can specify underlying type
 - Byte, short, int, long

```
enum Color: byte
{
    Red    = 1,
    Green  = 2,
    Blue   = 4,
    Black  = 0,
    White  = Red | Green | Blue,
}
```

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 31

Microsoft
.net

Enums - Example

```
enum WhiskeyKind {
    Scotch, Irish, Bourbon, Canadian }
enum WhiskeyMode : byte
{
    OnTheRocks = 1,
    WithWater  = 2,
    WithTonic  = 4,
    WithCola   = 8
}
```

Usage:

```
WhiskeyKind k = WhiskeyKind.Irish;
WhiskeyMode m =
    WhiskeyMode.OnTheRocks | WhiskeyMode.WithCola;
```

Default base-type is int (0, 1, 2, ...)

Enumeration base-type must be integral

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 32

Microsoft .net

Delegates

- Object oriented function pointers
- Multiple receivers
 - ✦ Each delegate has an invocation list
 - ✦ Thread-safe + and - operations
- Foundation for events

```

delegate void MouseEvent(int x, int y);

delegate double Func(double x);

Func func = new Func(Math.Sin);
double x = func(1.0);
    
```

33

Microsoft .net

Callback Methods (Delegates)

Delegates.cs

```

using System;
delegate void MyDelegate(String message);
class App{
    public static void Main(){
        MyDelegate call = new MyDelegate(FirstMethod);
        call += new MyDelegate(SecondMethod);
        call("Message A");
        call("Message B");
    }
    static void FirstMethod(String str){
        Console.WriteLine("1st method: "+str);
    }
    static void SecondMethod(String str){
        Console.WriteLine("2nd method: "+str);
    }
}
    
```

34

Microsoft .net

Delegates I

- Typed method references
- Delegate type
 - Type name
 - Delegate arguments
- `delegate void HullBreach(Deck d, Section s);`
 - Delegate return type
- Delegate variables
 - `HullBreach hullFatality;`
- Delegate invocation
 - `hullFatality(10, Section.Forward);`

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 35

Microsoft .net

Delegates II

- Creating delegates
- `public class ShipFatalityHandler`



```
{
    public void OnHullBreach(Deck d, Section s)
    {
        structuralIntegrity.PowerLevel++;
    }
    public ShipFatalityHandler(Ship ship)
    {
        ship.hullFatality = new
            HullBreach(this.OnHullBreach);
    }
}
```

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 36

Microsoft .net

Delegates III

- `new DelegateType(target.method);`
 - ♣ In C# 2.0: `target.method;`
- Method may be **static** (target is a class)
- Method may be **virtual, override, or new**
- Method must not be **abstract**
- Method signature and delegate type must match
 - ♣ Same number of parameters
 - ♣ Same parameter types (including return type)
 - ♣ Same parameter kinds (CbV, CbR)
 - ♣ Method name can be freely chosen



 

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 37

Microsoft .net

Delegates IV

- Are first class objects
 - ♣ Reference type
 - ♣ Can be passed around or stored in arrays/collections
 - ♣ Value can be `null` (exception on invocation)
- Store methods and their receivers
 - ♣ **Target** property to query receiver
 - ♣ As long as the delegate is alive target will not be collected
- Are equal if they have the same method *and* target



 

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 38

Microsoft
.net

Delegates V

- Delegate variable can hold multiple values → **multicast**
- Adding/Removing a delegate to a variable
 - ♣ `ship.hullFatality += new HullBreach(hullHandler.OnHullBreach);`
`ship.hullFatality += new HullBreach(evacuationHandler.Evacuate);`
 - ♣ `ship.hullFatality -= new HullBreach(evacuationHandler.Evacuate);`
- Invocation calls all delegates
- What about return or out values?
 - ♣ Last call determines returned values

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 39

Microsoft
.net



Polymorphism and Virtual Functions

- Use the `virtual` keyword to make a method virtual
- In derived class, override method is marked with the `override` keyword
- Example
 - ♣ `ToString()` method in Object class
 - ♣ Example derived class overriding `ToString()`

```
public virtual string ToString();
```

```
class SomeClass: Object{
    public override string ToString(){
        return "Some String Representing State";
    }
}
```

Polymorphism.cs






Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 40

Microsoft .net

Component Development

- What defines a component?
 - Properties, methods, events
 - Integrated help and documentation
 - Design-time information
- C# has first class support
 - Not naming patterns, adapters, etc.
 - Not external files
- Components are easy to build and consume



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 41

Microsoft .net



Properties

- Properties are “smart fields”
 - Natural syntax, accessors, inlining

```
public class Button: Control
{
    private string caption;

    public string Caption {
        get {
            return caption;
        }
        set {
            caption = value;
            Repaint();
        }
    }
}
```

```
Button b = new Button();
b.Caption = "OK";
String s = b.Caption;
```





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 42

Microsoft
.net

Properties

- Methods that look like fields (smart fields)

```
class Point{
    Int32 x;
    Int32 y;
    public Int32 X{
        get{return x;}
        set{x = value;}
    }
    public Int32 Y{
        get{return y;}
        set{y = value;}
    }
}
```

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 43

Microsoft
.net



Properties III

- Properties can be declared in interfaces


```
interface IShip {
    string Captain { get; set; }
}
```
- Properties can be abstract


```
public abstract class GalaxyClass : IShip {
    abstract Captain { get; set; }
}
```
- Properties can be static


```
public sealed class Universe {
    public static ulong GalaxyCount { get {...}
    }
}
```
- Getter or setter can be omitted (read-only or write-only property)

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 44

Microsoft .net



Indexers

- Indexers are “smart arrays”
 - Can be overloaded

```
public class ListBox: Control
{
    private string[] items;

    public string this[Int index] {
        get {
            return items[index];
        }
        set {
            items[index] = value;
            RePaint();
        }
    }
}
```

```
ListBox listBox = new ListBox();
listBox[0] = "hello";
Console.WriteLine(listBox[0]);
```





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006

45

Microsoft .net

C# and Events

- C# has built in support for events
- Great for dealing with objects in an event-driven operating system
- Improved performance and flexibility over an all-virtual-function solution
- More than one type can register interest in a single event
- A single type can register interest in any number of events



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006

46

Microsoft
.net

Handling an Event

```

EventHand.cs
using System;
using System.Windows.Forms;
class MyForm: Form{
    MyForm(){
        Button button = new Button();
        button.Text = "Button";
        button.Click += new EventHandler(HandleClick);
        Controls.Add(button);
    }
    void HandleClick(Object sender, EventArgs e){
        MessageBox.Show("The Click event fired!");
    }
    public static void Main(){
        Application.Run(new MyForm());
    }
}

```

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 47

Microsoft
.net

Defining an Event

- Based on a callback mechanism called a **delegate**

```

class EventInt{
    Int32 val;
    public Int32 Value{
        get{return val;}
        set{
            if(Changed != null)
                Changed(val, val);
            val = val;
        }
    }
    public event Callback Changed;
    public delegate
        void Callback(Int32 newVal, Int32 oldVal);
}

```

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 48

Microsoft
.net



Events - Firing

```
public delegate void EventHandler(object sender, EventArgs e);
```

- Define the event and firing logic

```
public class Button
{
    public event EventHandler Click;

    protected void OnClick(EventArgs e) {
        if (Click != null) Click(this, e);
    }
}
```

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 49



Microsoft
.net

Events - Handling


```
public class MyForm: Form
{
    Button okButton;

    public MyForm() {
        okButton = new Button(...);
        okButton.Caption = "OK";
        okButton.Click += new EventHandler(OkButtonClick);
    }

    void OkButtonClick(object sender, EventArgs e) {
        ShowMessage("You pressed the OK button");
    }
}
```



 

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 50




Attributes

- How do you associate information with types and members?
 - Documentation URL for a class
 - Transaction context for a method
 - XML persistence mapping
- Traditional solutions
 - Add keywords or pragmas to language
 - Use external files, e.g., .IDL, .DEF
- C# solution: Attributes

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 51



Attributes - Example



```

public class OrderProcessor
{
    [WebMethod]
    public void SubmitOrder(PurchaseOrder order) {...}
}

[XmlRoot("Order", Namespace="urn:acme.b2b-schema.v1")]
public class PurchaseOrder
{
    [XmlElement("shipTo")] public Address ShipTo;
    [XmlElement("billTo")] public Address BillTo;
    [XmlElement("comment")] public string Comment;
    [XmlElement("items")] public Item[] Items;
    [XmlAttribute("date")] public DateTime OrderDate;
}

public class Address {...}

public class Item {...}
  
```






Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 52

Microsoft .net

Attributes - Features

- Attributes can be
 - Attached to types and members
 - Examined at run-time using reflection
- Completely extensible
 - Simply a class that inherits from System.Attribute
- Type-safe
 - Arguments checked at compile-time
- Extensive use in .NET Framework
 - XML, Web Services, security, serialization, component model, COM and P/Invoke interop, code configuration...





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006

53

Microsoft .net

XML Comments

```
class XmlElement
{
    /// <summary>
    /// Returns the attribute with the given name and
    /// namespace</summary>
    /// <param name="name">
    /// The name of the attribute</param>
    /// <param name="ns">
    /// The namespace of the attribute, or null if
    /// the attribute has no namespace</param>
    /// <return>
    /// The attribute value, or null if the attribute
    /// does not exist</return>
    /// <seealso cref="GetAttr(string)"/>
    ///
    public string GetAttr(string name, string ns) {
        ...
    }
}
```



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006

54



Microsoft .net

Statements And Expressions

- High C++ fidelity
- If, while, do require bool condition
- goto can't jump into blocks
- Switch statement
 - ♣ No fall-through, "goto case" or "goto default"
- foreach statement
- Checked and unchecked statements
- Expression statements must do work

```

void Foo() {
    | == 1;    // error
}
    
```






Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 55

Microsoft .net

Arrays I

- Fixed size collection of homogeneous items
 - ♣ Items can be both value-types or reference types
- Arrays are reference types
- `int[] numbers = new int[3];`
creates an uninitialized array with 3 elements
- `int[] numbers = { 7, 8, 75 };`
creates an initialized array
- Element access with 0-based index (index-type is `int`):
`numbers[1] = 42;`
`Console.WriteLine("{0}", numbers[2]);`
- `numbers.Length` yields number of elements in the array (array-size)

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 56



Microsoft
.net

Arrays II

- One dimension is good, multiple dimension are better
- Jagged array (array of arrays)

```
float[][] fs = new float[2][];  
fs[0] = new float[15];  
fs[1] = new float[23];  
fs[1][17] = 12f;
```
- Rectangular (more efficient)

```
float[,] fs = new float[5, 4];  
fs[2, 1] = 23f;  
fs.GetLength(0)  
fs.GetLength(1)
```





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 57

Microsoft
.net

Arrays III

- **System.Array** class has a lot of useful methods
- Reverse, Copy, Sort, BinarySearch, Clear
- Read the fine manual for more info





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 58

Microsoft .net

Collections I

- **System.Collection** namespace
- Classes and interfaces for dealing with data collections
- **IList**: variable sized list of heterogeneous elements
 - ♣ **ArrayList, SortedList**
 - ♣ Element-type is **object**
- **IDictionary**: variable sized associative array
 - ♣ **Hashtable**
 - ♣ Element- and key-type are **object**





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 59

Microsoft .net

Collections: List Example

- `IList lst = new ArrayList();`
- `lst.Add("something");`
- `lst.Add(1); // boxing, IList expects a reference`
- `string s = (string)lst[0];`
- `// cast required IList only knows objects`
- `lst.Remove(1);`
- `lst.RemoveAt(0);`
- `lst[0] = "something else";`
- `Console.WriteLine(lst.Count);`
- `lst.Clear();`





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 60

Microsoft .net

Collections: Dictionary Example

- `IDictionary dict = new Hashtable();`
- `dict["something"] = 45; // boxing`
- `string s = (string)dict["something else"];`
- `// s == null → key not in dictionary`
- `object[] keys = dict.Keys;`
- `object[] val = dict.Values;`
- `dict.Remove("something");`
- `Console.WriteLine(dict.Count);`
- `dict.Clear();`

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 61

Microsoft .net



foreach Statement

- Iteration of arrays

```
public static void Main(string[] args) {
    foreach (string s in args) Console.WriteLine(s);
}
```

- Iteration of user-defined collections

```
foreach (Customer c In customers.OrderBy("name")) {
    if (c.Orders.Count != 0) {
        ...
    }
}
```






Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 62

Microsoft .net

Operator Overloading

- First class user-defined data types
- Used in base class library
 - ♣ Decimal, DateTime, TimeSpan
- Used in UI library
 - ♣ Unit, Point, Rectangle
- Used in SQL integration
 - ♣ SQLString, SQLInt16, SQLInt32, SQLInt64, SQLBool, SQLMoney, SQLNumeric, SQLFloat...



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 63

Microsoft .net

Operator Overloading

```
public struct DBInt
{
    public static readonly DBInt Null = new DBInt();



    private int value;
    private bool defined;

    public bool IsNull { get { return !defined; } }

    public static DBInt operator +(DBInt x, DBInt y) {...}

    public static implicit operator DBInt(int x) {...}
    public static explicit operator int(DBInt x) {...}
}
```

```
DBInt x = 123;
DBInt y = DBInt.Null;
DBInt z = x + y;
```





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 64

Microsoft .net

Versioning

- Problem in most languages
 - ♣ C++ and Java produce fragile base classes
 - ♣ Users unable to express versioning intent
- C# allows intent to be expressed
 - ♣ Methods are not virtual by default
 - ♣ C# keywords "virtual", "override" and "new" provide context
- C# can't guarantee versioning
 - ♣ Can enable (e.g., explicit override)
 - ♣ Can encourage (e.g., smart defaults)





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 65

Microsoft .net

Conditional Compilation

- #define, #undef
- #if, #elif, #else, #endif
 - ♣ Simple boolean logic
- Conditional methods

```
public class Debug
{
    [Conditional("Debug")]
    public static void Assert(bool cond, String s) {
        if (!cond) {
            throw new Asserti onExcepti on(s);
        }
    }
}
```





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 66

Microsoft
.net

Unsafe Code

- Platform interoperability covers most cases
- Unsafe code
 - ♣ Low-level code “within the box”
 - ♣ Enables unsafe casts, pointer arithmetic
- Declarative pinning
 - ♣ Fixed statement
- Basically “inline C”

```
unsafe void Foo() {
    char* buf = stackalloc char[256];
    for (char* p = buf; p < buf + 256; p++) *p = 0;
    ...
}
```

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 67



Microsoft
.net

Unsafe Code

```
class FileStream: Stream
{
    int handle;

    public unsafe int Read(byte[] buffer, int index, int count) {
        int n = 0;
        fixed (byte* p = buffer) {
            ReadFile(handle, p + index, count, &n, null);
        }
        return n;
    }

    [DllImport("kernel32", SetLastError=true)]
    static extern unsafe bool ReadFile(int hFile,
        void* lpBuffer, int nBytesToRead,
        int* nBytesRead, Overlapped* lpOverlapped);
}
```



 

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 68

Microsoft .net

Casting

- Change access type by casting values
 - ♣ `IList l = (IList)someObject;`
- Cast may fail
 - ♣ e.g. because `someObject` doesn't implement `IList`
 - ♣ `InvalidCastException`
- **is** operator checks whether an object is instance of a type
 - ♣ `someObject is IList` (either `true` or `false`)
- **as** operator for safe casting (only for reference-types)
 - ♣ `IList l = someObject as IList;`
 - ♣ Yields `null` if cast not possible; no exception
 - ♣ Combination of `is` and `cast`






Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 69

Microsoft .net

Reflection

- Ability of an application “to **examine** and possibly **modify** its **high level structure** at runtime.” (wikipedia.org)
- Use of type information at runtime
 - ♣ Also referred to as Meta-programming
- Uses of reflection
 - ♣ Serialization, remote method invocation, code generation, documentation and analysis, XML-Type mapping, COM Interop, DBMS Integration, dynamic modules (plug-ins)
- “The Case for Reflective Middleware” G. Blair, G. Coulson, 2002






Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 70

Microsoft .net

Reflection: Introspection in .NET

- Examine high-level structure
 - ♣ Types, members (methods, fields, ...), ...
 - ♣ But not loops, statements, expressions (may be supported in some languages through supporting libraries)
- Meta-information is part of the MSIL stored in an assembly
- Type descriptor for every type
 - ♣ Class `System.Type`
 - ♣ `obj.GetType()`
 - ♣ `typeof (typename)`
- Type descriptor is starting point to explore a type





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006

71

Microsoft .net


Reflection: System.Type

- Examine type
 - ♣ `IsPublic`, `IsPrimitive`, `IsEnum`, `IsClass`, `IsValueType`, `Assembly` ...
- Access to type members
 - ♣ `GetMethod`, `GetProperties`, `GetConstructor`
- Inheritance hierarchy
 - ♣ `IsSubtypeOf`, `IsAssignableFrom`, `IsInstanceOf`, `GetInterface`





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006

72




Reflection: Descriptors for other CTS Constructs

- Namespace: **System.Reflection**
- **ConstructorInfo, PropertyInfo, FieldInfo, MethodInfo, EventInfo, ParameterInfo, ...**
- Example: **MethodInfo**
 - **Attributes** (public, static, virtual, ...)
 - **GetParameters** (method parameters)
 - **Invoke** (invokes the reflected method)
- There's far more on reflection in the documentation





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 73



Reflection: Emit

- Examine code at runtime is nice
- But creating code at runtime is way cool
- Namespace: **System.Reflection.Emit**
- CLR is language agnostic → **only MSIL possible!!**



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 74



Microsoft .net

Threading

- Namespace: **System.Threading**
- Usage of the **Thread** class:

```
public class HoloDeckCharacter
{
    void Perform() {
        ...
    }
}

static void RunHoloDeckProgram(){
    HoloDeckCharacter c = new ProfMoriarity();
    Thread t =
        new Thread(new ThreadStart(c.Perform));
    t.Start();
}
```





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006

75

Microsoft .net

Threading: Thread States

- **ThreadState** property
 - ♣ **Aborted, Running, Stopped, Suspended, Unstarted, ...**
- State can be influenced with thread instance methods
 - ♣ **Start, Suspend, Resume, Abort**
- Aborting threads throws **ThreadAbortException**
 - ♣ Can be caught in running thread, which is to be aborted
 - ♣ Can be ignored with **Thread.ResetAbort()**



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006



76

Microsoft .net

Threading: Synchronization I

- Use monitors to protect critical sections

```
Monitor.Enter(someObj);  
try {  
}  
finally {  
    Monitor.Exit(someObj);  
}  
Is the same as  
lock (someObj) {  
  
}
```





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 77


Microsoft .net

Threading: Synchronization II

- Use `Monitor.Wait(obj)` to block until `obj` is notified
 - ♣ Thread must be in the monitor of `obj`
 - ♣ Monitor is released on `wait` and regained when `wait` returns
- Use `Monitor.Pulse(obj)` or `PulseAll(obj)` to notify one or all threads blocking on `obj`
 - ♣ Thread must be in the monitor of `obj`
- Example: Producer-consumer scenario
 - ♣ I leave that as an exercise to the reader





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 78




Threading: Synchronization III

- There are more synchronization primitives available
 - ♣ **ManualResetEvent, AutoResetEvent**
 - ♣ **Mutex**
 - ♣ **Interlocked**
- See the documentation for details





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 79



Asynchronous Method Invocation

- Non-blocking method calls
 - ♣ Call method
 - ♣ Return immediately
 - ♣ Get notified when invocation has completed
- Internally mapped to thread-pools (implementation detail)
 - ♣ No need to mess around with threads
- Any delegate object can be invoked asynchronously
 - ♣ **BeginInvoke, EndInvoke, IAsyncResult, AsyncCallback**





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 80

Microsoft
.net

Overview: Input/Output

- Namespace: **System.IO**
- Classes and interfaces for working with files and directories
- **Stream** as base class for all IO operations
 - ♣ **Read, Write, Flush, Seek, ...**
 - ♣ Asynchronous operations: **BeginRead, BeginWrite, ...**
 - ♣ Implementations: **FileStream, MemoryStream, ...**
- Reader and writer classes for specialised IO Operations
 - ♣ **TextReader/Writer, BinaryReader/Writer, ...**





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006

81

Microsoft
.net

Overview: Basic Networking

- Namespace **System.Net**
- Classes to build Internet applications
 - ♣ **IPAddress, IPEndPoint** (host, port), **Dns, ...**
- Low level **socket** interface (similar to the winsock API)
 - ♣ Various domains supported: **Inet, IrDA, IPX, ...**
- High level classes for stream-based networking
 - ♣ **TCPListener, TCPClient, ...**
- High level HTTP handling
 - ♣ **WebRequest, WebResponse, ...**



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006

82



Microsoft .net

XML

- Namespace: `System.Xml`
- There is a rich API for working with XML
 - DOM-based (`XmlDocument`)
 - Pull-based (`XmlReader`)
- Cool feature: XML-serialization

```

XmlSerializer xs = new
    XmlSerializer(typeof(MyClass))
MyClass m = new MyClass();
xs.Serialize(stream, m);
m = xs.Deserialize(stream);
    
```



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 83

Microsoft .net

XML Serialization Example

```

public class StarShip {
    [XmlElement,
    XmlArrayItem(Type=typeof(Officer)),
    XmlArrayItem(Type=typeof(Captain))]
    public Person[] Officers;
}
<StarShip>
  <Officer>Malcolm Reed</Officer>
  <Officer>T'Pol</Officer>
  <Officer>Hoshi Sato</Officer>
  <Captain>
    Jonathan Archer
  </Captain>
</StarShip>
    
```



 

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006 84

Microsoft .net

Exception

```
try {  
    // codice che può portare ad un errore  
}  
catch (Exception Type [ variable ]) {  
    // codice che deve essere eseguito quando avviene  
    l'errore  
}  
finally {  
    // codice da eseguire che avvenga o meno  
    // l'errore  
}
```





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006

85

Microsoft .net

Exception

```
using System;  
using System.IO;  
  
public class App {  
    public static void Main() {  
        FileStream fs = null;  
        try {  
            fs = new FileStream(@"C:\NotThere.txt", FileMode.Open);  
        }  
        catch (Exception e) {  
            Console.WriteLine(e.Message);  
        }  
        finally {  
            if (fs != null) fs.Close();  
        }  
    }  
}
```



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2005-2006

86

Microsoft
.net

References

<http://msdn.microsoft.com/net>

- Download .NET SDK and documentation

<http://msdn.microsoft.com/events/pdc>

- Slides and info from .NET PDC

<news://msnews.microsoft.com>

- microsoft.public.dotnet.csharp.general

