Introduction to C#

Anders Hejlsberg
Distinguished Engineer
Developer Division
Microsoft Corporation

C# – The Big Ideas

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

C# – The Big Ideas
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

C# – The Big Ideas

Everything really 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

C# – The Big Ideas

Robust and durable software

- 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

C# – The Big Ideas

Preservation of Investment

- 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
- Millions of lines of C# code in .NET
  - Short learning curve
  - Increased productivity

Hello World

```
using System;

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

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

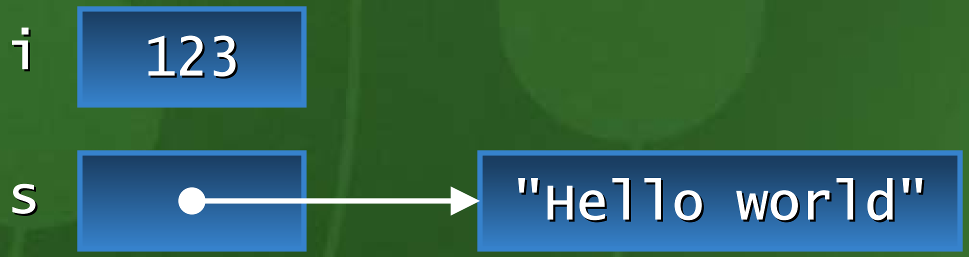
C# Program Structure

```
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;
     }
   }
}
```

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";
```


i
123
s
"Hello world"