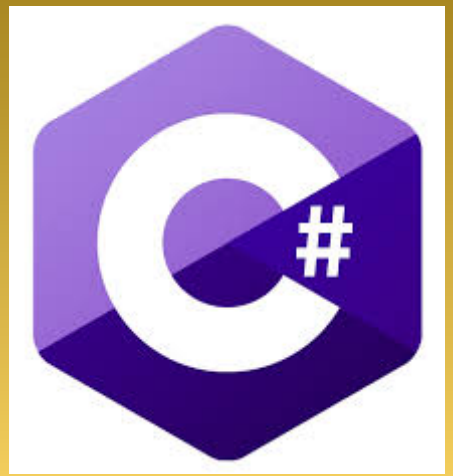
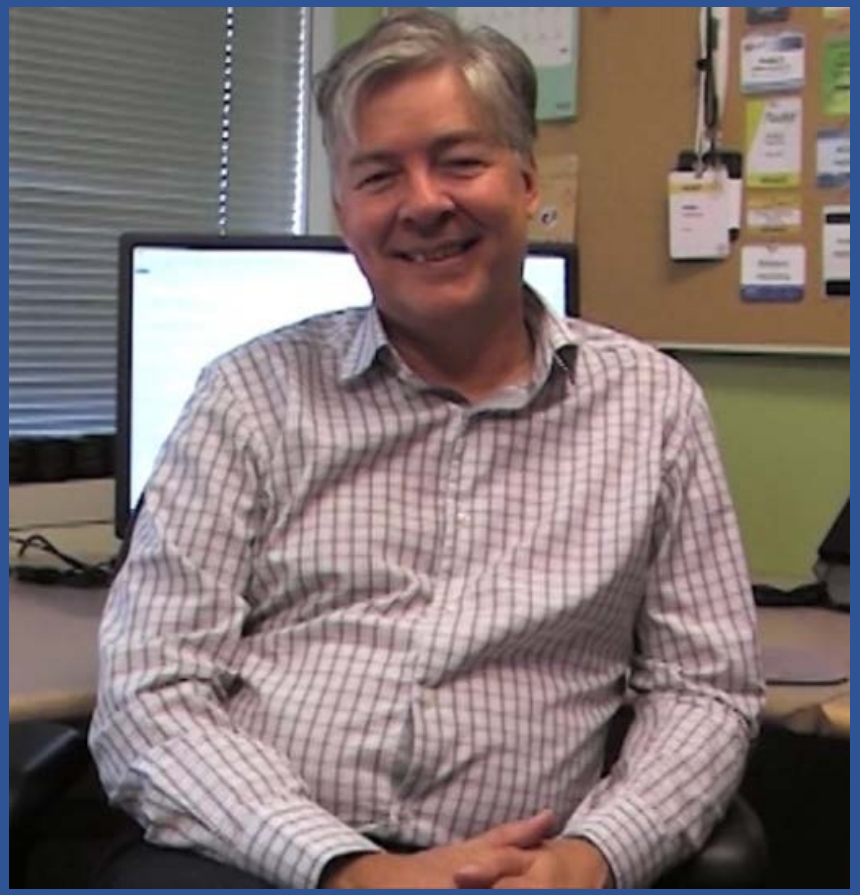
Introduction to C#



Loy Vanich (laploy@gmail.com 084 007 5544)

Creator

Anders Hejlsberg
Distinguished Engineer
Developer Division
Microsoft Corporation



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

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

A component oriented language

- 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

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 aspects of language design

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");
    }
}
```

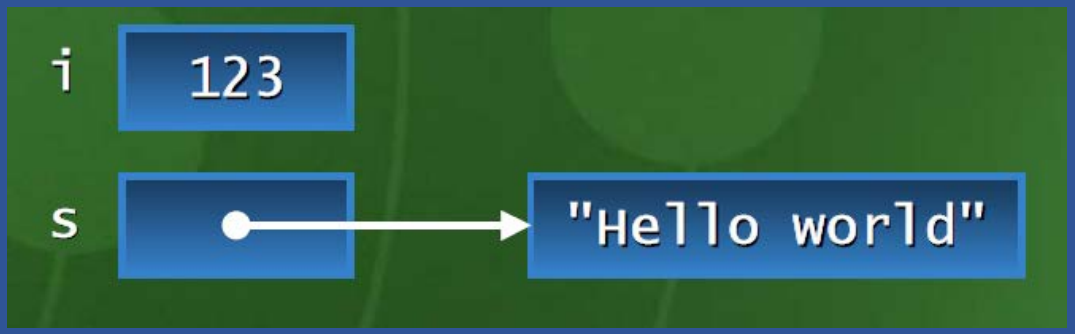
Program Structure

- 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

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"
```



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();`

Predefined Types

- C# predefined types
  - Reference — object, string
  - Signed — sbyte, short, int, long
  - Unsigned — byte, ushort, uint, ulong
  - Character — char
  - Floating-point — float, double, decimal
  - Logical — bool
- Predefined types are simply aliases for system-provided types
  - For example, int == System.Int32

Class

- Single inheritance
- Multiple interface implementation
- Class members
  - Constants, fields, methods, properties, indexers, events, operators, constructors, destructors
  - Static and instance members
  - Nested types
- Member access
  - public, protected, internal, private