C# Presentation

Trey Mack
James Moore
Osa Osar-Emokpae

Introduction

C#, pronounced “C Sharp,” is one of the new languages in the .NET framework being implemented by Microsoft. All .NET languages compile to a common byte code (MSIL) making their integration into programs written in different languages easier.

History

- C
- C++
- Developed by Anders Hejlsberg
  - Turbo Pascal
  - Delphi
  - Visual J++
- Released in 2001-2002

Previous Problems

- Memory Leaks
- Illegal Pointer References
- Overly Complex Multiple-Inheritance
- Static Linking

Resolutions

- Garbage Collection
- Threw out pointers
- Single inheritance with Interfaces
- Dynamic Linking
- Done 5 years ago in Java

What is C#

- Contrary to popular belief, C# is not simply a clone of or replacement for Java
- According to Anders Hejlsberg, Microsoft's Chief Architect, C# is a derivation of C++, C, Java, Modula 2, and Smalltalk

What is C#

- C# combines the best features of these languages and eradicates some of their weaknesses

Why Choose C#?

- C# was designed from scratch with the .net framework in mind
- C# combines the power of C and C++ with the productivity of Visual Basic
- With its familiar syntax the transition for Java and C++ programmers will be an easy one

Why Choose C#?

C# is in sync with current web standards and is easily integrated with existing applications.

In today's society where internet programming is inevitable having a language that already supports this makes the job of the developer easier.

Example of Code

The code looks a lot like Java

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

Features

- OOP

OOP

C# is object oriented. Every class is a subclass of an object. Everything is an object, yes even primitives. This makes generic programming easier.

Example:

```
int n = 3;
string s = n.ToString();
```

Features

- OOP
- Enumerators

Enumerators

Enumerators are a borrowed idea from C/C++. This is a data type consisting of a set of of named integers.

Example:

enum Weekday {Mon, Tues, Wed, Thu,

Fri, Sat, Sun};

Features

- OOP
- Enumerators
- Operator Overloading

Operator Overloading

Operator Overloading is yet another idea borrowed from c++. This makes polymorphism easier with custom data types.

Example:

Currency a, b, c;

c = a + b;

Features

- OOP
- Enumerators
- Operator Overloading
- Windows API Invocation

Windows API Invocation

C# was built with Windows in mind. It was created to allow programmers to create Windows application easily through a wraparound API. Some other technologies supported are COM, COM+.

Features

- OOP
- Enumerators
- Operator Overloading
- Windows API Invocation
- Structured Error Handling

Structured Error Handling

C# introduces new error handling techniques.

Try-catch blocks are used but with more functionality.

To throw an object, it has to be a subclass of System.Exception.

Try-Catch

try-catch blocks could be any of the following;

- ***try{ } catch(SomeException){ }***
- ***try{ } catch(){ } //catches any kind of exception***
- ***try{ } catch(){ } finally{ } //finally is always executed***
- ***try{ } finally{ } //finally is always executed***

Features

- OOP
- Enumerators
- Operator Overloading
- Windows API Invocation
- Structured Error Handling
- Delegates

Delegates

Delegates provide a template for a single method.

Example:

- *public delegate int ArithOp(int a, int b);*
- …
- *public int DoOp(ArithOp ar)*
- *{ return ar(a, b); }*

Features

- OOP
- Enumerators
- Operator Overloading
- Windows API Invocation
- Structured Error Handling
- Delegates
- Namespaces

Namespace

Namespace is a method of organizing similar files together. This is similar in some way to the java package idea. Every program is either explicitly within a namespace or in by default.

Example:

- ***namespace Project{ public class P1{} }***
- ***public class P2{}***

Namespace

To use a namespace, you just simply import by using the keyword ***using*.**

Example:

```
using system;
public class P1{}
```

Future of C#

With C#'s flexibility and support for many languages through the .NET architecture it will definitely become a widely used language in all aspects of programming.

Bibliography

- C# programming, Harvey, Robinson, Templeman, Watson
- http://www.funducode.com/csharp/basics/basics1.htm
- http://www.simonrobinson.com/DotNET/Articles/Languages/IntroCSh.aspx
- http://windows.oreilly.com/news/hejlsberg_0800.html
- http://msdn.microsoft.com/msdnmag/issues/0900/csharp/default.aspx