Microsoft C# Presentation

Team Members:
Jon Limpalair, David Rak, Casey Long
Relevant Paradigm and Problem Domains

Paradigms:
* Imperative, Object Oriented, Event-Driven, Type-Safe, Generic, Reflective
* Includes Functional programming abilities

> Library that adds Lambda expressions, extension methods, and anonymous types.
> Implemented type safety, garbage collection, and exception handling

* Used for Rapid Program Development

> Largely used for 'General' Programming and applications, though has implementation to help with database management, XML parsing, searching, and more.
> .NET integration hinders it's portable, making it a Windows only solution, though.
> Heavy emphasis on actual programming logic and readability over repetitive boilerplate code.
Context and Evolution

*Lead Designer: Anders Hejlsberg

> Anders was the lead designer of TurboPascal and Delphi
> Created .Net from C#, giving Microsoft a Virtual Machine implementation
> Starter a project called "Cool" (C-Like Object Oriented Language)
> "Cool" later became C#

*Aimed to create a first class modern language for the "Curly-Brace Crowd"

> "Curly-Brace Crowd" = C++ and Java devs- largest "General Applications" programmers

*Obvious heavy influences from Java and C++

> Less obvious influences from Delphi 5- of which C#'s principle designer also designed
> Since it's conception, Java and C# have greatly influenced one another's development
How It Evolved-
Where it is today

*Gained LINQ in 2007
  >LINQ allowed for more functional-style programming

*Has gone through 5 major revisions
  >Currently on C# 5.0, which is backwards compatible with all previous versions
  >2.0 added generics, iterators, 3.0 added Lambda expressions, typed local variables,
    4.0 added dynamic binding, 5.0 added asynchronous methods

*Gained popularity from XNA
  >XNA is Microsoft Toolkit for Game Development made in 2004
  >XNA gained mass popularity after Xbox 360's Xbox Live Arcade gave independent
    developers a chance to release their games for profit easily

*In 2004, signed deal with Novel for Mono
  >Mono is open source compiler for C# implementation.
  >If the code is 'clean' of Windows only code, the C# can be compiled for GNU/Linux
Language Concepts

*Designed to be closest to Microsoft's CLI
  >Common Language Infrastructure (CLI) is an open specification by Microsoft

*Has no global variables/functions
  >Can be substituted with Static members, however

*Syntactically similar to Java
  >Not entirely identical, but close enough to transfer simple code between one to another

*Supports Operator Overloading

*Supports Inheritance
  >Multiple inheritance not supported, but multiple interfaces are

*Supports libraries, methods, classes, etc.
Language Concepts

*Unified Type System

> Called Common Type System (CTS)
> All types, even primitives, use System.Object
> For example, this means all types inherit ToString()

*Supports generics

> Both syntactically and functionally identical to Java generics

*Implements "Boxing" and "Unboxing"

> Boxing is converting value-type object to generic object
> Unboxing is converting through explicit type casting a 'boxed' variable back
> int testingVar = 3001;
> object testingObj = testingVar;
> int testingVar2 = (int)testingObj;
Language Features

*XML based documentation system
  >Generates documentation based on code, much like Javadoc

*Memory Address Pointer security
  >C# does not use a virtual machine
  >Memory pointers can only be used within 'unsafe' code blocks and need special permission to be run
  >CANNOT reference garbage collected block or random memory block

*Supports the 'type' "Dynamic"

  >Dynamic Language Runtime determines a type at runtime

*Garbage Collection

  >Memory cannot be explicitly freed- it must instead be garbage collected
using System;
using System.Linq;
namespace Kodecsharp.Example.Linq
{
    class LinqIntro
    {
        [STAThread]
        public static void Main(string[] args)
        {
            int[] numbers = new int[10] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
            var evenNumbers = from number in numbers where (number % 2) == 0 select number;
            Console.WriteLine("Even numbers: ");
            foreach (int number in evenNumbers)
            {//Console.WriteLine(number + " ");
                var oddNumbers = from number in numbers where (number % 2) != 0 select number;
                Console.WriteLine("Odd numbers: ");
                foreach (int number in oddNumbers)
                {
                    Console.WriteLine(number + " ");
                }
            }
        }
    }
}

Prints out:
Even numbers: 
2 4 6 8 10
Odd numbers: 
1 3 5 7 9
Language Comparisons

*C# and Java

>Syntactically very similar to Java, although C# includes more robust tools
>C# does not naturally use a Virtual Machine (C# is not necessarily .NET)
>Java includes virtual methods, which C# does not have
>C# handles Generics much better than Java
>C# is generally faster than Java and generally uses less code. Does not rely on JITC.

*C# and VB.NET

>VB.NET doesn't rely on curly brackets or semi-colon
>VB.NET is much less robust than C#
>VB.NET compiles projects in the background (advantagous for small projects only)
>VB.NET has no document generator from code comments