Learning C#
What is C#

- A new object oriented language
  - Syntax based on C
    - Similar to C++ and Java
  - Used to write .NET software
    - Software that targets the .NET Framework is called managed code
  - C# gains much from the .NET Framework
    - Internet oriented platform
    - JIT compilation
    - Automatic memory management
    - Security, type-safety
    - Framework Class Library
C#: Rich Software Development

- Provides access to the .NET Framework
  - Great language for targeting .NET
  - Access the features of the framework
    - For example, the FCL
    - Create Web-based apps, GUI, apps, etc.

- Offers access to the underlying OS
  - Full access to Windows (or host OS)
  - Enables creation of rich applications

- Object oriented
  - Create component based applications
  - Gain the benefits of OO design, with no compromises
Defining the .NET Framework

- The .NET Framework is
  - A software development environment
  - A runtime engine for *Managed Code*
  - A platform designed for Internet-Distributed software

- The .NET Framework is an exciting new computing platform
Hello World a-la C#

HelloGUI.cs

```csharp
using System.Windows.Forms;
using System.Drawing;

class MyForm: Form{
    public static void Main() {
        Application.Run(new MyForm());
    }

    protected override void OnPaint(PaintEventArgs e) {
        e.Graphics.DrawString("Hello World!",
            new Font("Arial", 35), Brushes.Blue, 10, 100);
    }
}
```

c:\> csc /target:winexe HelloGui.cs
Types of Applications

- Managed code is packaged as *Assemblies*
- The three kinds of assemblies that you can create with C# are the following.
  - Console applications
  - GUI applications
  - Libraries of Types
- Libraries of Types are especially important because
  - Applications are going to consist of more and more reusable component code
  - Web Forms and Web Service applications are published as libraries
Creating a Console Application

```csharp
using System;

class App
{
    public static void Main(String[] args)
    {
        try
        {
            Int32 iterations = Convert.ToInt32(args[0]);
            if (iterations > 138)
            {
                throw new Exception();
            }
            Decimal lastNum = 1;  
            Decimal secondToLastNum = 0;
            while (iterations-- > 0)
            {
                Decimal newNum = lastNum + secondToLastNum;
                Console.WriteLine(newNum);
                secondToLastNum = lastNum; 
                lastNum = newNum;
            }
        }
        catch
        {
            Console.WriteLine("Usage: Rabbits [Fib Index]\n" + 
"\t[Fib Index] < 139");
        }
    }
}

c:\> csc Rabbits.cs
```
Creating a GUI Application

```csharp
using System;
using System.Drawing;
using System.Windows.Forms;

class App{
    public static void Main(){
        Application.Run(new TribbleForm());
    }
}

class TribbleForm:Form{
    TextBox generationsTextBox;
    ListBox fibList;
    // ...
}

c:> csc /target:winexe Tribbles.cs
```
Creating a Code Library

FibObj.cs

```csharp
using System;
public class Fib{
    Decimal current;
    Decimal last;
    public Fib(){
        current = 1;
        last = 0;
    }
    private Fib(Decimal last, Decimal secondToLast){
        current = last+secondToLast;
        this.last = last;
    }
    public Fib GetNext(){
        return new Fib(current, last);
    }
    public Decimal Value{
        get{return current;}
    }
}
```

c:\> csc /target:library FibObj.cs
using System;
class App{
    public static void Main(){
        Int32 index = 50;
        Fib obj = new Fib();
        do{
            Console.WriteLine(obj.Value);
            obj = obj.GetNext();
        }while(index-- != 0);
    }
}

FibTest.cs

c:\> csc /r:FibOjb.dll FibTest.cs
Language Concepts

- Syntax based on C/C++
  - Case-sensitive
  - White space means nothing
  - Semicolons (;) to terminate statements
  - Code blocks use curly braces ({})

- Some features
  - Can create methods with a variable number of arguments
  - Parameters are passed by value (by default)
    - Can create methods that take parameters by reference
    - Can create methods with out-only parameters
  - Operator overloading and type converters
  - Type-safety and code verification

- Object oriented, code is structured using the class keyword
Primitive Types

- Signed Numeric Primitive Types
  - Int32, Int16, Int64, SByte, Double, Single, Decimal

- Unsigned Numeric Primitive Types
  - UInt32, UInt16, UInt64, Byte

- Other Primitives
  - Boolean, String, Char, Object

- Primitive Types are FCL Types
  - C#Aliases the primitives
  - Example: Int32 == int
Conditional Statements

- C# uses `if`

```csharp
if(y == x){
    Console.WriteLine("y equals x");
} else{
    Console.WriteLine("y does not equal x");
}
```

- C# uses `switch`

```csharp
switch(x){
    case 2:
        Console.WriteLine("x equals 2");
        break;
    default:
        Console.WriteLine("x does not equal 2");
        break;
}
```
C# Loops…

- C# uses `for`

```csharp
for(index = 0; index < 100; index++){
    Console.Write(index);
    Console.Write("\t");
}
```

- C# uses `while`

```csharp
index = 10;
while(index != 0){
    Console.WriteLine(index);
    index--;
}
```
C# Loops (continued)

- C# uses **do-while**

```csharp
index = 0;
do{
    Console.WriteLine("Happens at least once");
}while(index < 0);
```

- C# uses **foreach**

```csharp
Int32[] myArray = new Int32[] {10, 20, 30, 40};
foreach(Int32 i in myArray){
    Console.WriteLine(i);
}
```
C# Error Handling

- C# uses try-catch

```csharp
try{
    Int32 index = 10;
    while(index-- != 0){
        Console.WriteLine(100/index);
    }
}
catch(DivideByZeroException){
    Console.WriteLine("Caught division by zero exception");
}
Console.WriteLine("Caught; code keeps running");
```
C# Assured Cleanup

- C# uses **try-finally**

```csharp
try{
    // Perhaps an exception is thrown or
    // return statement is hit
    return;
}finally{
    Console.WriteLine(
        "Code in finally always runs");
}
```
Using Types

- You will often use types from
  - The Framework Class Library (FCL)
  - Third party libraries

```csharp
using System;
using System.IO;
class App{
    public static void Main(String[] args){
        StreamReader reader =
            new StreamReader(args[0]);
        Console.WriteLine(reader.ReadToEnd());
    }
}
```
Demo C#Pad.cs
Learning C#
Demo
MDLView
Demo Visual Studio.Net
Demo TerraViewer