Microsoft .NET (v1: ~2002; v4: April 2010)

- **a framework for supporting standalone and web-based services**
  - single run-time environment for programs written in a variety of languages
  - web forms for interfaces on web pages
  - support for web services
  - better security than COM
- **development platform**
  - single intermediate language as target for all languages
  - just in time compilation to native instructions
  - common type system
    all languages produce interoperable objects and types
  - common language runtime environment
    base class libraries accessible to all languages
  - control of deployment and versioning
    the end of DLL hell?
  - uniform development environment for programs in multiple languages
  - significant new language, C#
  - major revision of Visual Basic

Why bother / who cares?

- **a major focus of Microsoft software development after COM**
- **interesting comparisons and contrasts with Java**
- **ties in with other topics of 333**
  - evolution of C, C++, Java -> C#
  - object-oriented programming
  - component-based software development
  - user interfaces
  - web services
  - politics and economics of software

Java model

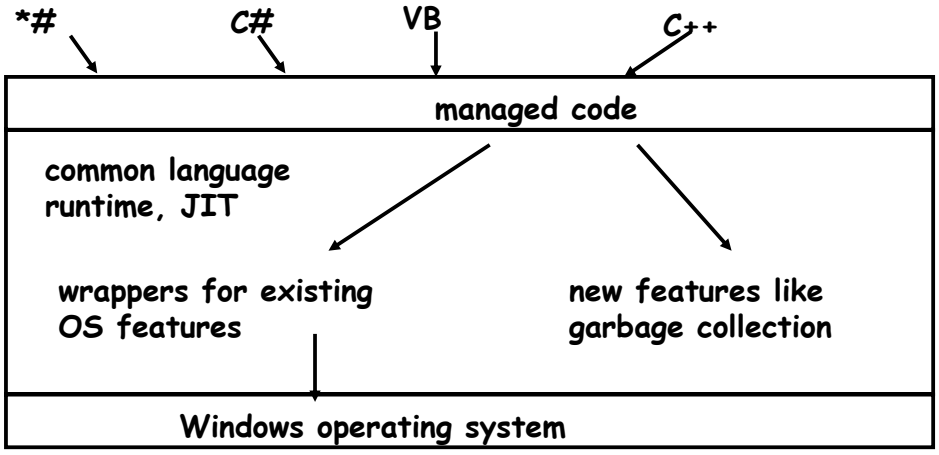
- **Java language**
  - derivative of C and C++
  - strictly object-oriented, strongly typed
  - garbage collection
- **compiled into intermediate language ("byte code")**
  - result stored in .class files
  - packages and JAR files for larger collections
- **interpreted by Java Virtual Machine on host**
  - local services provided by host system
  - enormous set of libraries in JRE
  - can be compiled into native instructions ahead of time or "just in time"
- **largely portable**
  - types completely specified
  - main problems come from making use of services of host environment
  - "write once, run anywhere" is mostly true
- **applets for running code in web pages**
- **Java Server Pages (JSP) for server-based web transactions**

.NET model

- **multiple languages: C#, VB, C++, J#, F#, …**
  - C# is a derivative of C, C++ and Java
  - VB.net is a significantly different version of VB
  - "managed extensions" for C++ that permit safe computation, garbage collection, etc.
- **all are object-oriented**
- **all languages compile into common intermediate language (CIL)**
  - types completely specified by Common Type System (CTS)
  - objects can interoperate if they conform to Common Language Specification (CLS) [a subset of CTS]
- **intermediate language compiled into native machine instructions**
  - just in time compilation, or compilation in advance: no interpretation
  - local services provided by host system (Windows)
  - enormous set of libraries
- **not portable**
  - tightly integrated into Windows environment
- **web forms for GUI components on web pages**
- **ASP.NET for server-based web transactions**

Common Language Runtime (CLR)

- **all languages compile into IL that uses CLR**
- **common services:**
  - memory management / garbage collection
  - exceptions
  - security
  - debugging, profiling
- **access to underlying operating system**



Deployment, versioning

- **prior to .NET, installing an application requires**
  - copying files to multiple directories
  - making entries in registry
  - adding shortcuts to desktop and menus
- **backing up, moving, or removing an application requires an installer program**
- **"DLL Hell"**
  - shared libraries can get out of sync with apps that need them
  - new installation can break existing programs that rely on properties of old DLLs
  - fresh installation can overwrite newer DLL with older one
- **assemblies provide strong internal naming/typing**
  - ensure that the right library is being used
  - assembly can specify versions of external references that it needs to work properly
  - CLR loads proper one
  - can have old and new versions working side by side

Assemblies

- **"fundamental unit of deployment, version control, reuse, activation scoping, and security permissions for a .NET-based application"**
  VS.NET documentation

- **collection of type and resource info**
- **(usually? always?) packaged as a .exe or .dll**
  - may contain other files, including .exe and .dll
  - executable parts are in CIL, not native code

- **each assembly contains a "manifest" with**
  - name, version of the assembly
  - file table: other files in the assembly
  - external dependencies

- **greatly reduce need for Windows registry**
  - program and components self-contained
  - can often remove an application just by removing the files

C# programming language

- **by Anders Hejlsberg (Turbo Pascal, Delphi, ...)**
- **based on C, C++ and Java**
  - Microsoft does not stress the Java contribution
  - "An evolution of Microsoft C and Microsoft C++" **(Visual Studio.NET documentation)**
- **"C# has a high degree of fidelity to C and C++"**
  - everything is a class object (Java)
    no global functions, variables, constants
  - garbage collection; destructors called implicitly (Java)
  - arrays are managed types (Java)
  - updated primitive types (Java)
    char is Unicode character; string is a basic type (Java)
  - single inheritance and interfaces (Java)
  - ref, out parameter modifiers
  - try-catch-finally (Java)
  - delegate type (roughly, function pointers)
  - unsafe mode (pointers permitted)
  - some syntax changes:
    '**.**' instead of **->** and **::** (Java), switches don't fall through, **foreach** statement
  - no headers or **#include** (Java)
  - /// documentation comments (Java)
- **ISO standard in 2003, v4.0 in April 2010**

Separated at birth?

```
public class hello {
  public static void main(String[] args)
  {
     System.out.println("hello, world");
  }
}
```

```
public class hello {
  public static void Main(string[] args)
  {
     System.Console.out.WriteLine("hello, world");
  }
}
```

"echo" in Java and C#

```
public class echo {
   public static void main(String[] args) {
      for (int i = 0; i < args.length; i++)
         System.out.println(
            "Arg[" + i + "] = [" + args[i] + "]");
   }
}
```

```
public class echo {
   public static void Main(string[] args) {
      for (int i = 0; i < args.Length; i++)
         System.Console.WriteLine(
            "Arg[{0}] = [{1}]", i, args[i]);
   }
}
```

Properties & accessors

- class data members can have get/set members
- external syntax looks like public class variables
- semantics defined by implicitly calling get and set methods

```
class Thing {
  bool _ok;          // private data member

  public bool ok {  // public property
    get { return _ok; }   // arbitrary computation
    set { _ok = value; }  // value is reserved word
  }
}

  Thing v;

  if (v.ok) {          // calls v's ok get
       v.ok = false;  // calls v's ok set
       ...
  }
```

Indexers (get/set [] members)

- syntax looks like array access (v[i])
- semantics defined by calling get and set members with a subscript
- can overload [ ] with different types

```
public class Awkarray {
  public Hashtable ht = new Hashtable();
  public Awk this[string name] {
    get {
      if (!ht.Contains(name))
        ht.Add(name, new Awk());
      return (Awk) ht[name];
    }
    set { ht.Add(name, value); }
  }

  Awkarray aa = new Awkarray();
  if (aa["whatever"] != null)
    aa["whatever"] = "a string";
```

Other C# odds and ends

- **operator overloading**
  - more like C++
  - but not =, ->, ( ), etc.
- **a goto statement!**
- **pointers (for unsafe code)**
- **structs as a value type**
  - not everything is an object
- **ref, out parameters**
- **lambda expressions, anonymous types**
- **generics**
- ...

- **other .NET languages**
  - VB, F# (sort of like ML / OCaml)
  - PowerShell
  - …

fmt in Java

```
import java.io.*;
import java.util.*;

public class f {
  String line = ""; String space = ""; int maxlen = 60;
  public static void main(String args[]) {
    f t = new f();
    t.runf();
  }
  public void runf()  {
    String s;
    try {
      BufferedReader in = new BufferedReader(new InputStreamReader((Sy
      while ((s = in.readLine()) != null) {
        String wds[] = s.split("[     ]+");
        for (int i = 0; i < wds.length; i++) addword(wds[i]);
      }
    } catch (Exception e) {
      System.err.println(e); //eof
    }
    printline();
  }
  public void addword(String w) {
    if (line.length() + w.length() > maxlen) printline();
    line += space + w;
    space = " ";
  }
  public void printline() {
    if (line.length() > 0) System.out.println(line);
    line = space = "";
  }
```

fmt in C#

```
using System;
using System.IO;

namespace fmtcs {
  class fmt {
    int maxlen = 60; string line = "";

    static void Main(string[] args) {
      new fmt(args[0]);
    }
    fmt(string f) {
      string inline;
      Stream fin = File.OpenRead(f);
      StreamReader sr = new StreamReader(fin);
      for (inline = sr.ReadLine(); inline != null; inline = sr.ReadLine
        string[] inwords = inline.Split(null);
        for (int i = 0; i < inwords.Length; i++)
          if (inwords[i] != String.Empty) addword(inwords[i]);
      }
      printline();
    }
    void addword(string w) {
      if (line.Length + w.Length > maxlen) printline();
        if (line.Length > 0) line += " ";
        line += w;
     }
     void printline() {
        if (line.Length > 0) {
           Console.WriteLine(line);
           line = "";
        }
```

fmt in VB.NET

```
Module Module1
    Dim line As String
    Sub Main(ByVal args As String())
        Dim inline As String, words As String()
        Dim i As Integer
        line = ""
        FileOpen(1, args(0), OpenMode.Input)
        While Not EOF(1)
            inline = LineInput(1)
            words = inline.Split(Nothing)
            For i = 0 To words.Length - 1
                addword(words(i))
            Next i
        End While
        FileClose(1)
        printline()
    End Sub
    Sub addword(ByVal w As String)
        If line.Length + w.Length > 60 Then
            printline()
        End If
        If line.Length > 0 Then
            line = line & " "
        End If
        line = line & w
    End Sub
    Sub printline()
        If line.Length > 0 Then
            Console.WriteLine(line)
            line = ""
        End If
    End Sub
End Module
```

Conclusions

- ***C#***
  - easy to pick up basics if know Java
  - easy to convert Java statements to C#
  - batch mode compilation is easy

- **VB.NET**
  - each new release has made VB more complicated
  - wizard helps upgrade process but doesn't handle everything

- **Visual Studio.NET**
  - all languages are handled in a uniform way
  - good integration of visual and textual descriptions

- **.NET framework**
  - huge download if not already installed
  - not easy to adapt or upgrade most existing programs to .NET
    COM not likely to go away in the near future