Software components

- **software re-use**
  - libraries, etc.
  - inter-language linkage
- **the Microsoft way**
  - COM: the Component Object Model
  - Visual Basic: scripting, embedding, viruses
  - .NET
  - C#
- **other approaches to components**
  - CORBA, Java RMI, JavaBeans, ...

Software re-use

- **how do we re-use code written by others?**
  - "If I have seen further than others, it is because I have stood on the shoulders of giants."
- **source code**
  - e.g., open source
- **libraries of compiled code**
  - e.g., archives of object files on Unix, DLL's on Windows, Java packages, ...
- **classes**
  - C++ Standard Template Library
  - Java Collection framework
  - ...
- **objects**
- **components**
- **mashups**
- **application program interfaces (APIs)**

Libraries

- **linking to previously compiled code**
- **static linking: all called routines are included in executable**
- **dynamic linking**
  - called routines located and linked in on demand
    shared libraries on Unix (.so == "shared object")
    dynamic link libraries (DLL's) on Windows
    plug-ins in browsers
- **advantages of dynamic linking**
  - no cost if a particular routine is not called
  - minor startup cost for initialization when called
  - minimal cost when running (extra indirection for call)
  - library code is shared among all simultaneous uses
  - can update libraries without updating entire program
- **some disadvantages**
  - DLL hell on Windows: inconsistencies among versions
    especially after install then uninstall

COM: Microsoft's component object model

- **binary standard for creating & using components**
  - components can be written in any language
    - IDL (interface definition language) to describe arguments and return values, generate necessary code
  - components can be in same process, separate process on same machine, or on some other machine (DCOM)
    - DCOM transports include TCP/IP and HTTP
  - supporting libraries marshal arguments, call functions, retrieve results
    - all happens transparently to process that uses it
  - integral part of Microsoft systems
    - available on non-MS operating systems (sort of?)

- **COM components are objects with interfaces**
  - interface: functions that provides access to methods
    - based on C++ virtual function calls, but implementable in any language
  - 128-bit GUID (globally unique identifiers)
    - stored in Windows registry so others can find it

ActiveX

- **marketing name for technologies and services based on COM**
- **ActiveX components are COM objects**
  - executable code that packages an object as
    - .EXE (standalone executable)
    - .DLL (dynamic link library)
    - .OCX (visual interface control)
- **ActiveX controls**
  - COM components with user-interface aspects
  - written in C++, Java, VB, …
  - can be used in web pages (analogous to applets, but less restricted)
  - can be controlled with VBScript, WScript and other scripting languages
- **ActiveX documents**
  - lets users view and edit non-HTML documents through the browser
  - integrates existing documents into browser or any other application
    "embedding"

Calling a COM object

- **conceptually, what happens when a COM object is called from a program...**
- **first time**
  - find its code
    - look up in Windows registry
    - registered during install or when created or by explicit call
  - do any initialization
    - Windows needs to keep track of what DLLs are in use
  - link it into current program (if a DLL)
    - fill in calls with pointer to real code: vtbl
- **each subsequent method call**
  - collect arguments into proper form ("marshalling")
  - call function
  - convert return value and output arguments into proper form
- **when done**
  - do any finalization
  - release resources
    - last user tells Windows that DLL is no longer in use

Alternative approaches

- **CORBA (Common Object Request Broker Architecture)**
  - industry consortium (OMG: Object Management Group)
  - client-server model, using objects
  - object-request broker (ORB)
    - communicates client requests to target objects, finds object implementation, activates it if necessary, delivers request, and returns response
  - IDL (interface definition language) and compiler for specifying and implementing interfaces
- **Java RMI (Remote Method Invocation)**
  - a remote procedure call mechanism
  - call objects located (usually) on other systems
  - very loosely equivalent to (D)COM
  - can pass objects, not just primitive types
- **Java Beans (marketing name for Java components)**
  - an API for writing component software in Java
  - components expose features (methods & events)
  - visual application builder tools determine properties by "introspection" or "reflection": can query an object about its properties
  - loosely analogous to ActiveX components
  - attempting to solve same problems as COM and CORBA, but within Java

Visual Basic

- **a programming language**
  - modern dialect of Basic (John Kemeny ('47, *49) and Tom Kurtz (*56), 1964)
  - reasonable control flow, data types, arrays, structures
- **a toolkit**
  - standard library for math, file I/O, text manipulation
  - user interface components: buttons, text, menus, ...
  - extensible: easy access to entire Windows API and existing objects

    can add own C/C++ code and create new controls
  - a "glue" language for assembling from pre-built components
- **an integrated development environment**
  - interactive system for building and testing VB programs

    draw interface by dragging and dropping components

    fill in behaviors in code templates, set properties like size, color, position, …

    manage/edit source code and other resources

    run in controlled environment for test and debug, compile and export as .EXE file
- **an extension mechanism**
  - embedded (as VBA) in many other programs, including Word, Excel, Powerpoint, Outlook; can easily extend their capabilities
  - a vehicle for distributing viruses

Component scripting

- **component exposes what it can do as an object interface: methods, properties, events**
  - can control object from a programming language that can access objects
- **VBScript is a scripting version of VB for controlling scriptable objects**
  - can use it to control scriptable programs
  - also CScript, WScript, PowerShell, ...
- **Visual Basic for Applications (VBA) is a version of VB that lives inside some programs**
  - notably Word, Excel, other Office programs, Outlook, …
  - can use it to control them and other scriptable programs
- **in general, can do anything from a program that is possible from keyboard and mouse**
  - macro recorder to create command sequences
  - shell escape to run other processes
  - network libraries to access other systems

Security issues

- **VB embedding and scripting is a mixed blessing**
  - useful properties: can easily extend capabilities, customize behaviors
  - lots of not so nice properties: viruses are very easy
- **scripts, plug-ins, applets let others run their code on your machine**
- **how can this be made safe (enough)?**
- **code-signing (Microsoft's "Authenticode")**
  - uses crypto to assure that code comes from who it says it does
  - and that it hasn't been tampered with
  - but NOT that it works properly
    - doesn't protect against bugs, invasion of privacy, ...
- **sandboxing (Java applets, Javascript)**
  - isolate code inside virtual machine or similar
  - limits capabilities (e.g., no access to local file system)
  - doesn't protect against bugs in programs
  - or bugs in the security model and implementation
- **perfect security is not possible**
  - see Doug McIlroy's Virology 101 paper

Microsoft .NET (v1: ~2002; v4.5 preview: Feb 2012)

- **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**

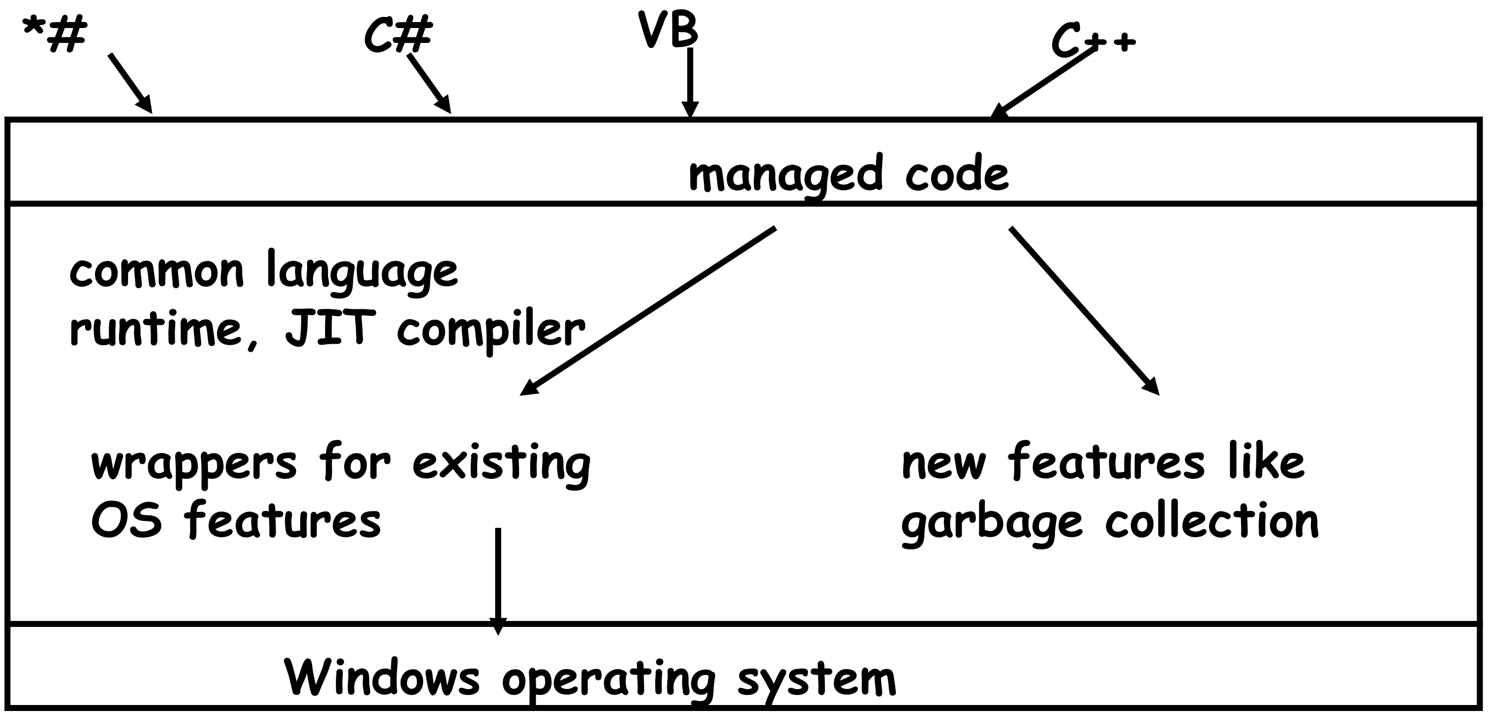
.NET model

- **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**


*#
C#
VB
C++
managed code
common language runtime, JIT compiler
wrappers for existing OS features
new features like garbage collection
Windows 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

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)**
  - "... sort of Java with reliability, productivity and security deleted." (James Gosling)
- **"C# has a high degree of fidelity to C and C++"**
  - everything is a class object; no global functions, variables, constants (Java)
  - 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 (most recent)**

“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

```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((System
      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
    }
    println();
  }
  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]);
      }
      println();
    }
    void addword(string w) {
      if (line.Length + w.Length > maxlen) println();
        if (line.Length > 0) line += " ";
        line += w;
     }
     void println() {
        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)
        println()
    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
  - very 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