First, a recap

- We’ve seen that modern computer systems are organized using object-based structures and remote method invocation.
- These architectures:
  - Are layered
  - Are typically described using pictures with lots of boxes and arrows labeled with 3 or 4 letter acronyms
  - Suffer from major performance issues…
  - … which stem primarily from latency
  - … and can be attacked using caching
- We also saw the complexity of the network layer. The sophisticated application developer is forced to be very aware of options at the network layer.
Today?

- Major vendors are trying to help
  - New languages are intended to improve productivity and reduce errors
  - Better integration of language and environment can ease your job
- But while this structures a complex world, it doesn’t make it “simple”
  - So they encourage a style of learning by doing…

History of Java and C#

- Java was introduced by Sun almost ten years ago
  - C was not sufficiently object oriented, had unsafe pointers, and code was felt to be hopelessly buggy
  - C++ was worse, sharing the problems for which C was infamous and adding new ones associated with inheritance and operator overload
- Java was a breath of fresh air for many!
We teach Java at Cornell

- For teachers, Java is
  - Clean
  - Easy to explain and easy to use
  - Well supported
  - Not all that slow

- But Java is also a very soft and fuzzy language. One can’t “visualize the instructions produced”

Ken’s favorite C fragments

- Code for which you can visualize the instructions generated:

  ```c
  n = N;
ap = dest;
bp = src;
do
    *ap++ = *bp++;
while(--n);
  ```

  ```c
  n = N;
ap = & vec[N];
do
    *--ap = 0
while(- -n);
  ```

- When building real systems, this matters!
Criticism of Java

- Java code compiles to Java byte code, which is a “safe” language
- When ready to run, the runtime uses a JIT to create actual machine code, mixed with calls to the runtime environment
- The result is safe, but often slow. Hard to predict what will be fast and what will crawl… Ken’s favorite fragments can’t be coded without an index variable in Java

C#

- Microsoft and SUN battled for years over Java runtime environment
  - The “framework”, “run time” and standard “class library”
- Eventually, Microsoft offered .NET as a better (but rather proprietary) alternative to J2EE.
  - They support Java on .NET
  - But they don’t support J2EE as a .NET environment
- At the same time, they made some small changes to the Java language
- C# is this new (but very similar) variant of Java
## Differences

<table>
<thead>
<tr>
<th>C# type</th>
<th>Java type</th>
</tr>
</thead>
<tbody>
<tr>
<td>sbyte</td>
<td>byte</td>
</tr>
<tr>
<td>byte</td>
<td>boolean</td>
</tr>
<tr>
<td>bool</td>
<td>Object</td>
</tr>
<tr>
<td>object</td>
<td>unsigned byte</td>
</tr>
<tr>
<td>byte</td>
<td>unsigned short</td>
</tr>
<tr>
<td>ushort</td>
<td>unsigned int</td>
</tr>
<tr>
<td>uint</td>
<td>unsigned long</td>
</tr>
<tr>
<td>ulong</td>
<td>(no special money type)</td>
</tr>
<tr>
<td>decimal</td>
<td></td>
</tr>
</tbody>
</table>

## Differences?

<table>
<thead>
<tr>
<th>In C# you write</th>
<th>In Java you write</th>
</tr>
</thead>
<tbody>
<tr>
<td>int[] iarray;</td>
<td>int iarray [];</td>
</tr>
<tr>
<td>In C# everything is an object</td>
<td>Primitives aren’t objects</td>
</tr>
<tr>
<td>Automatic “boxing”</td>
<td>Manual “boxing”</td>
</tr>
<tr>
<td>Can introduce unsafe code</td>
<td>Can’t do unsafe code at all</td>
</tr>
</tbody>
</table>
Real philosophical difference?

- Java has tended more and more towards a “pure Java” world
  - CORBA can be used to talk to non-Java applications
  - Overheads aren’t much different than Web Services overheads
  - Yet this is discouraged rather explicitly

.NET and interfacing

- Microsoft has embraced a diversity of languages
- .NET and C# abound in examples of
  - Cross language operations
  - Support for weird languages like Eiffel and O’Caml and Cobol
- Microsoft treats legacy software as a first-class member of .NET
Overall model in .NET

The intermediate language has type information but otherwise looks much like any assembler language. Compilers easily ported to it.. But many languages have their own ways to do allocation and representation and for them, some elements of CLR are a challenge!

CLR provides...

- CLI – a form of machine code
- Type system
- Cross-domain access
- Class library
- Heap management, garbage collection
- File format
- Platform access
Java “versus” C#?

- Differences are mostly in the contents of the runtime environment
- But these differences are major
  - Java actually has two runtimes, J2EE and Jini, but Jini has lost steam...
  - J2EE “adaptors” seen by many as a late “catch-up with .NET” effort
- Real issue is that as cross-language stuff expands, safety is being lost

Safety: A holy grail...

- At the outset, “pure” Java was touted as a security solution
  - But more and more problems arose with imported code and special data types
  - Java security demanded that all of these be implemented in Java!
- But cross-domain invocations became the usual case, and now applications have many ways to subvert the local system
- System administrators see security as the biggest challenge of our era!
How does Microsoft “sell” .NET to J2EE users? How does the J2EE community respond?

- SellingdotNET.ppt
- SellingJ2EE.ppt
- Microsoft .NET vs J2EE How Do They Stack Up.htm