Java vs. .NET

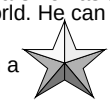
by
Venkat Subramaniam
venkats@durasoftcorp.com
Dura*Soft*, *Inc.*
http://www.durasoftcorp.com/download

at
ACM Houston Chapter – April 2003

This presentation available for download at above URL

Abstract

- The Microsoft .NET framework (and C#) is gaining usage and popularity. Coming from the knowledge of Java (and C++), the speaker's first reaction, like most people, was "Why another language similar to Java...?" This talk compares Java with some of the features of C# and .NET. Similarities and differences between the languages/platforms are presented. The speaker's experience and opinion based on his work with Java for over 7 years and with .NET for over 18 months will be presented. Insights into what is important to focus for those interested in learning C# or .NET is also provided. This talk assumes the audience is familiar with Java.

- Dr. Venkat Subramaniam, president and co-founder of Dura*Soft*, is a trainer and mentor of software developers in the area of object- and web-based technologies. He is experienced in developing large scale object-oriented applications for engineering companies. He is also an adjunct professor in the Department of Computer Science at the University of Houston, where he teaches and works with graduate students on their research projects. Venkat is the instructor for the Professional Software Development Series at Rice Technology Education Center. Over the past 10 years he has trained over 2500 software professionals in the Houston area and around the world. He can be reached at venkats@durasoftcorp.com.
- Any page with a  has an example attached

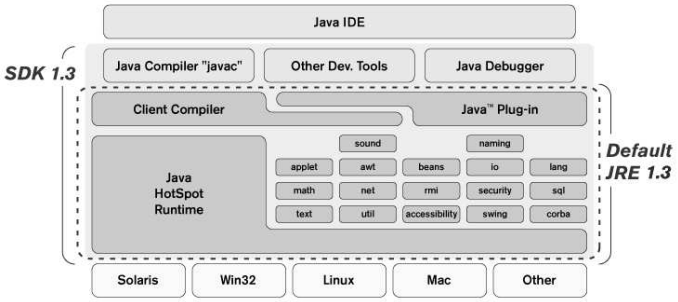
Agenda

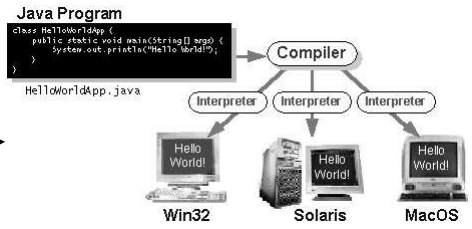
- Introduction
- Java Platform
- .NET Framework
- Technology Comparison
- Java vs. C#
- Servlet/JSP vs. ASP.NET
- EJB vs. .NET
- Summary

Introduction

- Java
  - Introduced in 1995
  - Derives root from C++, Smalltalk, Objective-C
  - Removed number of pitfalls of C++
    - avoids surprises
  - Popularity triggered by Web front end development
    - sustained popularity for back end development
- .NET
  - Introduced Feb. 2002 (betas introduced 1999-2001)
  - Derives root from C++, VB, COM, COM+, MTS, Java!
  - Removed number of pitfalls from interoperability between languages & pitfalls of COM
    - brings ease of GUI development to non-VB developers
    - better server side programming model

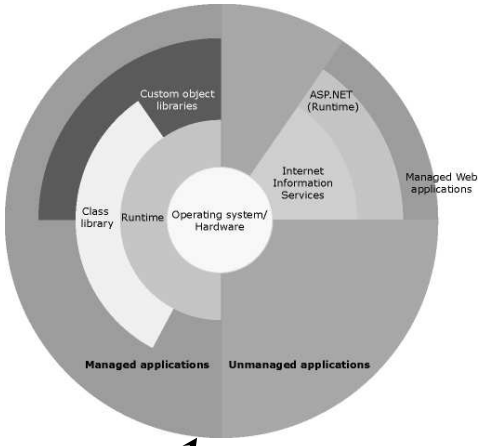
Java Platform

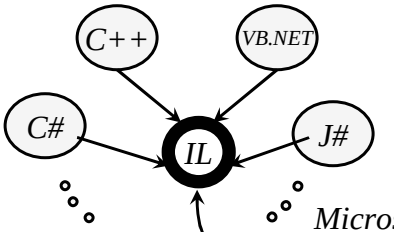




Sources:
http://java.sun.com/j2se/1.3/
http://java.sun.com/docs/books/tutorial/getStarted/intro/definition.html

.NET Framework





Microsoft Intermediate Language (MSIL or IL)

Source:
http://msdn.microsoft.com

Technology Comparison

Java	.NET
JVM	CLR
Byte Code	MSIL
Swing	WinForms
Servlet, JSP, Struts	ASP.NET, WebForms
Micro Edition	Compact Edition
JDBC	ADO.NET
Web Services(JAXRPC)	WebServices
J2EE*	Enterprise Services*
RMI*	.NET Remoting*
JMS*	MSMQ*
JNI	PInvoke/COM Interop
Java Predominates	Multi language support
Multi Platform	Windows Predominates
* Not quite equivalent - discussed later	

Java vs. C#

- Is this Java or C#?

```
public class HelloWorld
{
    public static void Main(String[] args)
    {

    }
}
```

main in Java

Syntax Similar Yet Different

- Java
  - C++, Smalltalk, Objective-C root
  - Removes number of gotcha in C++ ☺
  - Restrictive when it comes to
    - pointers
    - methods are virtual by default
      - Positive: No way to hide a method and shoot in the foot like C++
- C#
  - Tries to be a union of C++ and Java
  - at times irritating in this regard 😐
    - Discussed later
    - Use your judgment in using some of these features

Syntax Similarity

Java

```
public class HelloWorld
{
      public static void main(String[] args)
      {
            System.out.println("Hello World!");
      }
}
```

C#

```
public class HelloWorld
{
      public static void Main(String[] args)
      {
            Console.WriteLine("Hello World!");
      }
}
```

Optional

Syntax Differences

- Access Controls
  - Java
    - private, public, protected ☹, (package friendly)
  - C#
    - private, public, internal, protected, protected internal ☺
- Inheritance
  - Java
    - extends, implements
  - C#
    - : (for both one base class and interfaces)

Syntax Differences Overriding

- Overriding
  - Java
    - final vs. non-final methods ☺
    - You may accidentally override methods
      - Not really an issue most of the time!
  - C#
    - methods not marked as virtual can not be overridden ☺
    - Overriding method needs to be marked with override ☺
      - if you do not, it only gives you warning & assumes you are hiding
        - You got to be kidding! ☹
        - Use the option to treat warnings as errors PLEASE
    - You can mark a method as new with intent to hide
      - I feel sick in my stomach ☹

Java vs. C# Some Differences

Java	C#
package	namespace
import	using
Pass by value only	By value, ref and out
instanceof	is
extends, implements	:
super(…)	: base(…)
super.method()	base.method()
final (class)	sealed
final (field)	readonly
Class	Type
Serialization (binary)	Serialization (XML too)

Some More Differences

- Only in Java
  - Checked Exceptions and throws clause
  - Anonymous Inner classes (coming in future version of C#)
- Only in .NET
  - Attributes
  - Properties
  - Indexer
  - IDisposable
  - foreach
  - Delegate
  - Code based security
  - Pointer access - allows through “unsafe” construct
  - operator overloading

Checked Exceptions and throws

```
public class E1 extends Exception {…}
…
      public void doSomething() throws E1
      {
            if(…)
                  throw new E1(…);
            …
      }
}
```

IDEs like Eclipse use this information to automatically surround code with try-catch block

```
…
      try
      {
            obj.doSomething();
      }
      catch(E1 ex)
      {
      }
```

Checked Exceptions and throws…

- Java supports checked and unchecked exceptions
  - Checked exceptions must be caught using catch
    - or the method itself should declare throws clause
  - otherwise results in compilation error
- A method may not throw any exception unless the method is marked with a throws of that exception or its base
- Exception thrown by a method is part of its interface

Anonymous Inner Classes

```
…
      component.addActionListener(
            new ActionListener()
            {
                  public void actionPerformed(…)
                  {
                        …
                  }
            }
      );
```


Attributes

- Defines characteristics on various subjects
  - assemblies, classes, methods, properties, etc.
- Similar to the attributes in Microsoft IDL
- Appears within [] before the subject

```
[System.Serializable]
public class Sample {
      [System.NonSerialized] private int aField1;
      private int aField2;

      [System.Obsolete("Use increment instead", true)]
      public void inc() {…}

      public void increment() { aField2++; }
```

Properties ☺

- An attribute or field represents some characteristics of the object
- Making it public is undesirable
  - uncontrolled access
- You may make them private and provide access methods
- If intent is to access field, why not expose it as a property
  - intent is clear
  - tools can help us identify properties
  - change is very much controlled – still encapsulated!
- Compiler translates property p into get_p and set_p methods! and provides an illusion to the user

Writing Properties

```
public class Car {
      private int yearOfMake;
      private string bodyColor;
      public Car() {…}
      public int year
      {
            get { return yearOfMake; }
      }
      public virtual string color
      {
            get { return bodyColor; }
            set {
                  if (value.CompareTo("Orange") == 0)
                     throw new ApplicationException(…);
                  bodyColor = value;
            }
      }
}
```

Read-only property

Read/Write property

Well Encapsulated

Indexer

```
public class Vector {
      private int[] values;
      private int size;
      public Vector(int rSize) {…}
      public virtual int this[int index]
      {
            get
            {
                  if (index >= size)
                  throw new IndexOutOfRangeException(…);
                        return values[index];
            }
            set
            {…
                   values[index] = value;
            }
      }
}
```

Gives an illusion of being indexed

Translates into get_item and set_item methods

Problem with Finalize in Java

- Java has automatic Garbage collection
  - No need to worry about memory cleanup
  - Still resource cleanup is a concern
- Finalize called when Garbage Collector returns object to heap
- Garbage Collector may be lazy - Finalize will be called sometime in the future
- If program exits fast - Finalized may never be called

Do not depend on the Finalize() method

IDisposable

- .NET has a better handle on this
- Dispose the object by calling Dispose

```
public class Garbage : IDisposable {
      private bool disposed = false;
      public void Dispose() {
            if (disposed == true)
            throw new ObjectDisposedException(…);
            disposed = true;
            // What ever cleanup
            Console.WriteLine("Dispose called");
            GC.SuppressFinalize(this);
      }
      ~Garbage() { Dispose();
            Console.WriteLine("Finalize called");
      }
}
```

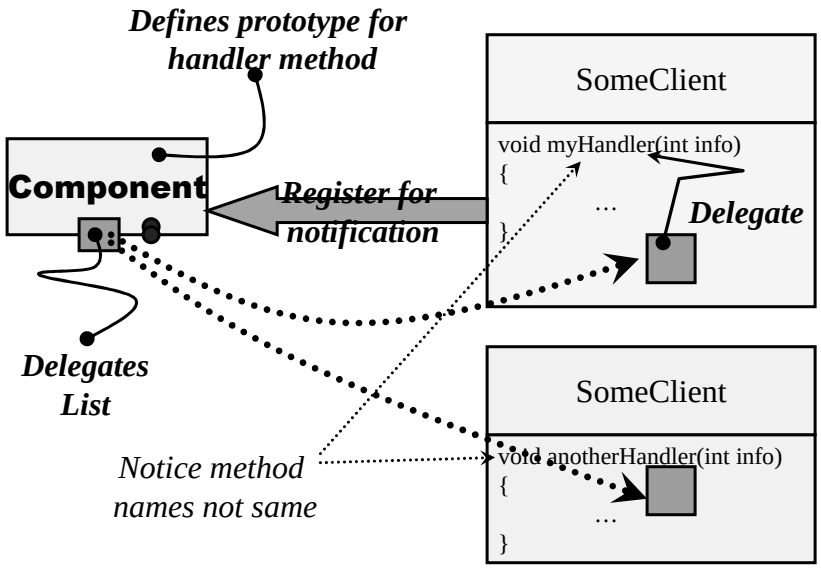
```
using (Garbage obj = new Garbage())
{
      // code to use obj
}//obj.Dispose() called automatically here!
```

foreach

- Any object that implements the IEnumerable can be traversed using a foreach

```
foreach(DataRow row in table.Rows)
{
     int someField = row[“some_field”];
     // row here represents each row in
     // the collection in the respective
     // iteration through the loop
}
```

Event Handling with Delegate



Delegate

- Event handlers traditionally were global or static functions
- In .NET, delegates allow a lot more
- Delegates are data structures (objects) that hold
  - either a pointer to global or static function
  - or a pointer to an object's method and the object itself
  - are object-oriented, type safe and secure
- All Delegates derive from the Delegate class

Writing A Delegate

- Delegate classes are written with special syntax
- Compiler does special processing
  - Writes a constructor and Invoke method when compiled
- You can maintain a list of delegates by simply adding and subtracting
  - myDelegate += anotherDelegate;
    // Adds the delegate
  - myDelegate -= anotherDelegate;
    // Removes the delegate

Writing A Delegate…

```
public delegate void StockQuoteDelegate(double amount);
public class StockQuote {
      public StockQuoteDelegate highDelegate = null;
      public StockQuoteDelegate lowDelegate = null;

      private void newHighReached(double amount)
            {// one way to invoke the handlers
                  Object[] args = new Object[1];
                  args[0] = amount;
                  highDelegate.DynamicInvoke(args);
            }
      private void newLowReached(double amount) {
            //another way of achieving the same result
                  lowDelegate(amount);
            }
}
```

Using A Delegate

```
public class MyClass {
public static void highReport(double amt) { … }
public static void lowReport (double amt) { … }
public void beepReport(double amt) { … } // Non-static
static void Main(string[] args) {
      MyClass obj = new MyClass();
      StockQuote stkQuote = new StockQuote();
      stkQuote.highDelegate +=
            new StockQuoteDelegate(User.highReport);
      stkQuote.lowDelegate +=
            new StockQuoteDelegate(User.lowReport);
      stkQuote.highDelegate +=
            new StockQuoteDelegate(obj.beepReport);
      stkQuote.lowDelegate +=
            new StockQuoteDelegate(obj.beepReport);
      …
      stkQuote.highDelegate -=
            new StockQuoteDelegate(obj.beepReport);
}
```


Code Access Security

- Permission granted based on trust level
- Security Demand
  - Your code (class library) demands that other classes calling your methods or accessing objects of your classes have a certain set of permissions (specified by you)
  - All the callers in the Call Stack are checked to see if ***any*** of the callers lack the required permission
    - SecurityException is thrown if that is the case
- Security Overrides
  - allows you to override code permission explicitly
  - you can further restrict your permission before calling a third party code – a way to use other's untrustworthy code

Code Access Security…

```
public class MyClass {
      public void foo() {
            TextReader reader =
             new StreamReader("myfile.dat");
            …
}
// Security demand may be specified using attributes
[FileIOPermission(SecurityAction.Demand,  Unrestricted=true)]
public void f2() {…}
public void f3() {
// This method applies security overrides.
// It demands that the code being called does
// not access any files.
FileIOPermission perm =
      new FileIOPermission(
            PermissionState.Unrestricted);
perm.Deny();
MyClass obj = new MyClass();
obj.foo();
FileIOPermission.RevertDeny();
```

Unsafe vs. Unmanaged Code

- Unmanaged Code:
  - this is not executed under the tight supervision of CLR
    - no garbage collection
    - limited debugging capabilities
  - Useful to call Platform Specific functions
- Unsafe Code:
  - this is managed code!
  - it simply uses some constructs (like pointer usage) that C# does not encourage

Unsafe Code

- CLR manages memory
- Java does not allow pointer manipulation
- C# derived from C++, wants to allow it, however with caution
- Code that manipulates pointers may lead to memory leaks, etc.
- C# declares the section of code that manipulates pointers as unsafe!
  - your take care of memory management when within this block of code – allows you to use pointers
- To prevent GC use the fixed keyword to ***pin***

Usage of unsafe

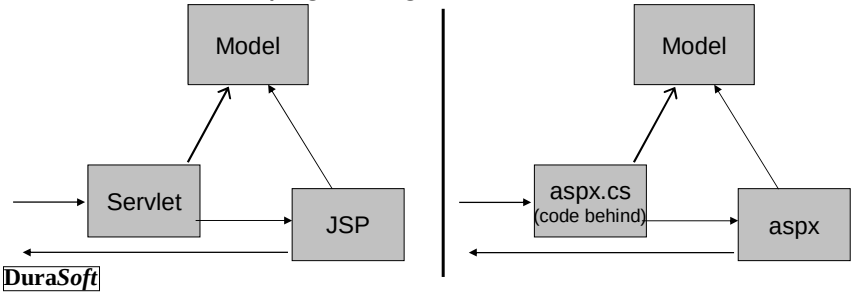
```
unsafe public static void usePointers(int[] array)
{
        fixed(int* ptrArray = array) {
                for (int i = 0; i < array.Length; i++)
                        Console.Write(*(ptrArray + i) + " ");
        }
}
…
{       ValTypeX instOfX;
        ValTypeX[] myXArray = new ValTypeX[2];
        RefTypeY objY = new RefTypeY();
        …
        unsafe
        {
                ValTypeX* ptrX = &instOfX;
                // No need to use fixed for Value type (on the stack, remember!)
                Console.WriteLine(ptrX->val);
                fixed(ValTypeX* ptrX2 = myXArray)  {
                        Console.WriteLine(ptrX2->val);
                }
                fixed(int* pVal = &objY.val)  {
                        Console.WriteLine(*pVal);
                }
                …
```

Operator Overloading

- C++ has operator overloading
  - one can debate if this is a feature or a flaw
- Java smartly avoided this for good reasons
- It is disappointing to see that C# took this up!
- Note operator overloading is not supported across .NET languages
  - If you want your code to be CLI compliant, you must provide a regular method for each overloaded one
- C# has gone overboard with overloading
  - May overload true, false, &&, || (gets pretty messy)
  - This is surely a *feature* to be avoided

Servlet/JSP vs. ASP.NET

- Servlet JSP has come a long way
  - Model 1 architecture mixed presentation layer with business logic – not maintainable
  - Model 2 applies MVC and is elegant
  - Struts framework makes it easy to achieve this
- ASP.NET has come a long way
  - Far superior than ASP – an understatement
  - code behind page brings the best of both worlds



EJB vs. .NET

- Specification vs. Implementation
  - EJB is a specification
  - .NET is an implementation
    - though parts of it open (C# and CLI – ECMA standard!)
- Multiple vendors implement the EJB spec
  - WebLogic, WebSphere, JBoss, to name a few
- Microsoft only vendor (predominantly)
  - though efforts underway to implement on other platforms by other vendors

AppServers and Enterprise Services

- EJB specification implemented by different vendors
- .NET provides similar functionalities (not all) through its Enterprise Services
  - Under the hood it interacts with COM+ (the second generation of MTS) [not COM]
- EJB relies on deployment descriptors to communicate intent on transactions, etc.
- .NET relies on Meta data for that

Stateless Beans vs. Object Pooling

- EJB: The most favored Bean ☺
  - provides the best performance and scalability
  - Control classes in OO Modeling maps to these
  - may access the database using JDBC, etc.
  - Need to use deployment descriptors to specify type of bean
- .NET: Enterprise Services provides Object Pooling and Just-in-time Activation ☺
  - Attributes used to build meta data to specify these
  - Unlike MTS, no separate deployment required. Automatic deployment when accessed.
  - Essentially behaves as (Managed) COM+ Service

Stateful Bean vs. ?

- EJB: Stateful session Beans
  - useful to interact with clients while carrying a conversational state
  - Not optimal from the scalability point of view
  - Not as desirable as Stateless Beans
- .NET: Not directly supported.
  - You can try to achieve the same goal through work arounds

Entity Beans vs. ?

- EJB: Entity Beans
  - Substantial support for managing persistence
  - Two ways to manage persistance: CMPs and BMPs
  - Least favored Beans in EJB however
  - Suffers from poor performance due to comprehensive object life time management and data access mechanism
- .NET: Did not even bother to go this route
  - Some people look at this as deficiency.
  - Some people look at this as efficiency

Persistence Management

- EJB:
  - Entity Beans are really cool if only they have better performance
  - People give up on Entity Beans and use JDBC from session beans
  - JDO is gaining popularity, however, requires third party product support
  - Other proprietary OR mappings
- .NET: No Object Wrapper on data
  - Several improvements made to ADO.NET for data access
  - No effort to provide an object wrapped access

Transaction Management

- EJB:
  - Transaction boundaries and needs can be marked
  - Required, RequiresNew, Supports, NotSupported, …
  - Uses deployment descriptors however
    - XML based descriptors are read by the container
  - EJB Container manages this
- .NET:
  - Transaction boundaries and needs can be marked
  - Required, RequriesNew, Supports, NotSupported, …
  - Uses TransactionAttribute (Meta data) to specify these needs
    - Enterprise Services meta data is ready by COM+ runtime
  - COM+ manages these

Messaging

- EJB:
  - Relies on JMS for messaging
  - JMS is a specification
    - different vendors implement the MOMs
- .NET:
  - Provides Queued Components
  - Well integrated with MSMQ
  - Not a specification, but a product

Tools

- Command Line
  - Java: Strong command line tools support
  - .NET: Not as powerful
- IDE
  - Java:
    - Some what behind in capabilities for Java
    - Leading projects Visual Age, JBuilder
    - Eclipse (open source) is very promising
      - Lacks some nice features present in Visual Studio .NET, however
  - .NET
    - Strong IDE Visual Studio .NET
      - Enhances productivity
      - Makes some difficult tasks almost effort less
      - Exceptional support for development and debugging
      - Lacks some nice features present in Eclipse, however

Future Features

- Java
  - generics
    - good old Templates from C++!
- .NET
  - generics
  - anonymous inner classes (for event handlers)

Summary

- Java
  - Very elegant
  - Superior model, cleaner, you clearly understand what's going on
  - Hard to use for certain tasks/applications
  - Not as good a support for GUI and IDE as .NET
  - Very good server side support – JSP, Struts, EJB
  - Multiple vendors (Pluses and minuses related to this)
- .NET
  - Really cool
  - Not very clear modeling when you develop
    - You have to make an effort to keep your head clear
  - Very easy to develop, high productivity, shorter time to market
  - One vendor (Minuses and Pluses related to this)

Which one should you use?

- You decide which one is best for your application and needs
- Use the one that you think will do the job well for you
- Decide without any prejudice or emotions
- Enjoy the freedom to choose – Good luck ☺