

Java in the real world

Stephen White

8th February 2007

Plan

- About the Company (brief!)
- What we use Java for
- Why we use Java
- What we're looking forward to in Java 6



About DecisionSoft

- Based in Oxford
- Developing XML Tools since 1997
- Provider of XBRL Services & Components
- In partnership with Business Wire, formed CoreFiling Ltd as a dedicated XBRL company
- Customers include HMRC, Companies House, Dutch Government, and others



So ... Java?

- All major current development projects are written in Java
- Includes server and client applications:
 - Desktop XBRL validator
 - Desktop XBRL development tool
 - Enterprise/server XBRL validator
 - Web applications

Why Java? Clean Language

- There are fewer ways to accidentally shoot yourself in the foot in Java than in some of the obvious alternatives (e.g. C/C++)
- Often easier to debug
- This means more time writing productive (and interesting) code
- Why do I think its cleaner ...



Why Java? Clean Language

- High level, OO language
- Runtime bounds & cast checking
- Java doesn't have #define or operator overloading
- C++ has many mis-features (e.g. Problems caused by Array/pointer duality) that cause bugs



Why Java? Memory Management

 Bit of a contentious issue whether garbage collection is a good thing but ... C++ memory management is hard



Why Java? Libraries

- High quality standard library one of the original selling points of Java
- Lots of free and OpenSource libraries available
- For example:
 - Xerces (now a standard part of the JRE)
 - Lucene



Why Java? Tools

- IDEs
 - Eclipse (free)
 - IDEA
- Code coverage tools, debuggers, profilers



Why Java? Portability

- Portability comes for free (almost)
- Java includes portable APIs for:
 - GUIs
 - Networking
 - File handling

- ...



Surely it isn't all good

- There are problems, or perceived problems:
 - Performance
 - JRE isn't properly free yet
 - Lack of language features
 - Isn't actually as portable as C can be
 - Platform integration
 - The .NET framework provides some of the same features

Bad? Performance

True:

- Trying to use SWING applications on a i486 was painful
- Writing nice code can result in lots of tiny Objects, this isn't fast in any language
- Garbage collection delays
- JVM startup overhead
- But the JVM is actually really fast



It isn't free?

- Has always been free to download/install
- Source has been available for a while, but under a restrictive licence
- You can now 'apt-get install sun-java5-jre'
- Sun are working towards releasing under GPL
- There are other JVM implementations

Java 5

- Performance improvements
- Generics
- Annotations
- Dynamic generation of RMI stubs
- JAXP improvements



Java 6

- Performance & compatibility enhancements
- Web services support
- SWING/GUI improvements
- Scripting support
- XML API extensions, including StAX
- Extra security related APIs
- Compiler API

Performance & Compatibility

- Faster, our benchmarks (based on part of our real codebase) run 20% faster on Java
- Vista support



Web Services

- Defines annotations to describe how to convert a class into a web service
- Can be deployed using a built-in web server for testing, or on an application server
- Standard approach should lead to greater interoperability between IDEs



Web Services – Simple Example



Web Services – Real Example

```
package nl.ntp.truenorth.validatiedienst;
@WebService(targetNamespace="http://validatiedienst.ntp.nl")
@SOAPBinding(style=SOAPBinding.Style.RPC,
             use=SOAPBinding.Use.LITERAL)
public class NtpValidator {
  @WebMethod
 public String validateXBRL(@WebParam(name="fileContent") final
String fileContent) throws MalformedURLException, IOException {
```



Web Services – Test & Deploy

Nice and easy to test ...

```
public static void main(String[] args) {
   NtpValidator validator = new NtpValidator();
   Endpoint endpoint =
      Endpoint.publish("http://localhost:8081/ValidatieDienst", validator);
}
```

 Also easy to deploy on an application server using J2EE 5 or by including the JAX-WS libraries



SWING (and other GUI things)

- Improved Look and Feel implementations
- Sortable tables (see SwingSet2 demo)
- SwingWorker
- Rendering improvements
- Splash screen support
- System tray support



SWING - SwingWorker

```
SwingWorker worker = new SwingWorker<String, Void>() {
    @Override
    public String doInBackground() {
       return myReallySlowMethod();
    }

@Override
    public void done() {
       _textField.setText(get());
    }
}
```



Scripting

- Very similar to the Apache BSF, but included as standard
- JavaScript interpreter included as standard
- E.g.

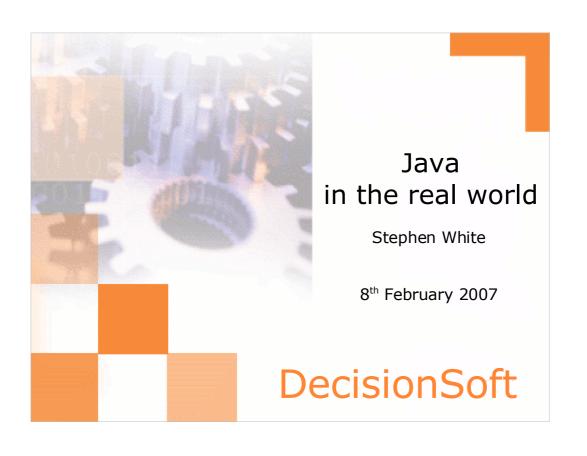
```
alert("Hello in JavaScript");
javax.swing.JOptionPane.showMessageDialog(
  null, "Hello SWING World", "A Title",
  javax.swing.JOptionPane.ERROR_MESSAGE);
```



References

- http://java.sun.com/docs/books/tutorial/uiswing/concurrency/index.html
- http://java.sun.com/webservices/docs/2.0/jaxws/UsersGuide.html
- http://java.sun.com/javase/6/docs/technotes/guides/security/enhancements.html
- /usr/lib/jvm/java-6-sun/demo/





Plan

- About the Company (brief!)
- What we use Java for
- Why we use Java
- What we're looking forward to in Java 6



DecisionSoft

Explain why we're still looking forward to Java 6, despite the fact that it has already been released.

About DecisionSoft

- Based in Oxford
- Developing XML Tools since 1997
- Provider of XBRL Services & Components
- In partnership with Business Wire, formed CoreFiling Ltd as a dedicated XBRL company
- Customers include HMRC, Companies House, Dutch Government, and others



DecisionSoft

XBRL is an XML-based language for "the electronic communication of business and financial data". Accountants get excited by that sort of thing. Since we're not accountants we'll skim quickly over what XBRL is and focus on the software we've been writing.

HMRC run Java application servers, as do various big US financial institutions that I'm not allowed to mention.

So ... Java?

- All major current development projects are written in Java
- Includes server and client applications:
 - Desktop XBRL validator
 - Desktop XBRL development tool
 - Enterprise/server XBRL validator
 - Web applications



DecisionSoft

Swing. SWT. J2EE webapps

Why Java? Clean Language

- There are fewer ways to accidentally shoot yourself in the foot in Java than in some of the obvious alternatives (e.g. C/C++)
- Often easier to debug
- This means more time writing productive (and interesting) code
- Why do I think its cleaner ...



DecisionSoft

Saying you're spending more time being productive will sell an idea to management. How do you sell it to programmers? I don't know many programmers who say "Yay, I've got hours worth of debugging in front of me". I think we call get bored and frustrated when debugging a trivial problem that has occurred because we've stumbled over a language (mis)-feature. We'd much rather be getting on and writing new and useful functionality, implementing interesting optimizations, and so on.

Why Java? Clean Language

- High level, OO language
- Runtime bounds & cast checking
- Java doesn't have #define or operator overloading
- C++ has many mis-features (e.g. Problems caused by Array/pointer duality) that cause bugs



DecisionSoft

Ok, #define has uses – but it can be abused. Even the "good" uses, such as the production of debug and non-debug builds, aren't totally problem free. Do your testers use the debug or non-debug builds? Are you sure your debugging statements don't have any unexpected side effects?

I've rarely seen a use for operator overloading that really helps improve the clarity of the code, and there are some really dumb things that people have done with it.

The duality of arrays and pointers is really just a deficiency in the type-system. It is an area where you should be able to express to the compiler what you intend and have it statically determine whether this aspect of your code is right (see also delete, delete[]). For that matter the duality of pointers created with new and with malloc is an issue. In fact pointers in general really screw up your compiler's ability to tell you when you've made a mistake in your code.

Why Java? Memory Management

 Bit of a contentious issue whether garbage collection is a good thing but ... C++ memory management is hard



DecisionSoft

Not saying that it isn't possible to introduce memory leaks in Java programs despite the garbage collector. Referenced but never again touched is nearly as bad as totally unreferenced!

If C++ users hate automatic memory management then why does auto_ptr exist? If you're going to reference count why not actually do it properly!

Why Java? Libraries

- High quality standard library one of the original selling points of Java
- Lots of free and OpenSource libraries available
- For example:
 - Xerces (now a standard part of the JRE)
 - Lucene

Why Java? Tools

- IDEs
 - Eclipse (free)
 - IDEA
- Code coverage tools, debuggers, profilers



DecisionSoft

The cleaness of the language helps make it easier to write many of these tools.

Why Java? Portability

- Portability comes for free (almost)
- Java includes portable APIs for:
 - GUIs
 - Networking
 - File handling

- ...



Surely it isn't all good

- There are problems, or perceived problems:
 - Performance
 - JRE isn't properly free yet
 - Lack of language features
 - Isn't actually as portable as C can be
 - Platform integration
 - The .NET framework provides some of the same features

DecisionSoft

Come back to performance on a different slide.

Portability. You need a platform with a JRE. Nearly every platform that has ever existed has a C compiler.

Bad? Performance

- True:
 - Trying to use SWING applications on a i486 was painful
 - Writing nice code can result in lots of tiny Objects, this isn't fast in any language
 - Garbage collection delays
 - JVM startup overhead
- But the JVM is actually really fast

It isn't free?

- Has always been free to download/install
- Source has been available for a while, but under a restrictive licence
- You can now 'apt-get install sun-java5-jre'
- Sun are working towards releasing under GPL
- There are other JVM implementations

Java 5

- Performance improvements
- Generics
- Annotations
- Dynamic generation of RMI stubs
- JAXP improvements



Java 6

- Performance & compatibility enhancements
- Web services support
- SWING/GUI improvements
- Scripting support
- XML API extensions, including StAX
- Extra security related APIs
- Compiler API

Performance & Compatibility

- Faster, our benchmarks (based on part of our real codebase) run 20% faster on Java 6
- Vista support



Web Services

- Defines annotations to describe how to convert a class into a web service
- Can be deployed using a built-in web server for testing, or on an application server
- Standard approach should lead to greater interoperability between IDEs



Web Services – Simple Example



Web Services – Real Example



Web Services – Test & Deploy

Nice and easy to test ...

```
public static void main(String[] args) {
  NtpValidator validator = new NtpValidator();
  Endpoint endpoint =
     Endpoint.publish("http://localhost:8081/ValidatieDienst", validator);
}
```

 Also easy to deploy on an application server using J2EE 5 or by including the JAX-WS libraries



SWING (and other GUI things)

- Improved Look and Feel implementations
- Sortable tables (see SwingSet2 demo)
- SwingWorker
- Rendering improvements
- Splash screen support
- System tray support



SWING - SwingWorker

```
SwingWorker worker = new SwingWorker<String, Void>() {
   @Override
   public String doInBackground() {
     return myReallySlowMethod();
   }

   @Override
   public void done() {
     _textField.setText(get());
   }
}
```



Scripting

- Very similar to the Apache BSF, but included as standard
- JavaScript interpreter included as standard
- E.g.

alert("Hello in JavaScript");
javax.swing.JOptionPane.showMessageDialog(
 null, "Hello SWING World", "A Title",
 javax.swing.JOptionPane.ERROR_MESSAGE);



References

- http://java.sun.com/docs/books/tutorial/uiswing/concurrency/index.html
- http://java.sun.com/webservices/docs/2.0/jaxws/UsersGuide.html
- http://java.sun.com/javase/6/docs/technotes/guides/security/enhancements.html
- /usr/lib/jvm/java-6-sun/demo/

