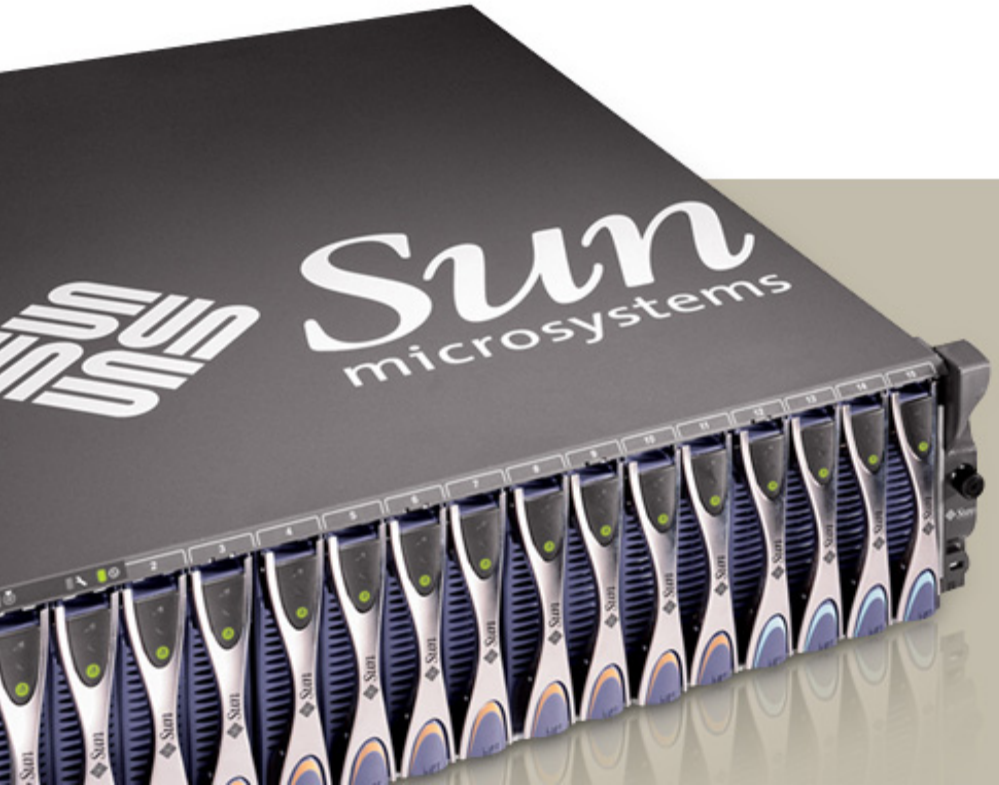
What is New on the Java Platform?

SUN TECH DAYS 2008–2009
A Worldwide Developer Conference

Simon Ritter
Technology Evangelist

Java SE 6 u10



Three Deployment Routes

There are three major ways a Java platform program can be distributed today

- Applets (Java Plug-In software)
- JNLP (Java Web Start software)
- Standalone Programs (Custom Installers)

All three paths share similar challenges

Deployment Challenges

- What versions of Java Technology are installed?
- How do I launch Java Technology?
- What's the best way to install a new version?

When you consider the number of platform and browser combinations Java technology supports, none of these questions have simple answers

Present-Day Solutions

- Possible to detect JRE version 1.4.2+ using JavaScript™ Technology
- Exact version details not available
- Auto-install too complex
  - Only available for Internet Explorer on Windows
- Sun only provides sample scripts
  - No complete, working solution
- Outside the browser, you're on your own
  - Fun with registry manipulation
- Installation could be smoother

Next-Generation Installer

- New Windows look-and-feel
- More streamlined
- Less intimidating
- Improved messaging
- More visually appealing

Deployment Toolkit

- ActiveX Control and NPAPI Plugin included in JRE
- Script will use the plugins if available
  - Fine grained JRE detection
    - Pure javascript can only detect at the family granularity
  - Install any available JRE version
    - Pure javascript will only install latest JRE version
  - Select installation type (kernel or online installer)
  - Declare application or applet required packages
- Plugins remain installed after JRE uninstallation

Browser Plug-Ins

- Use native code to perform tests
- JavaScript solution will check for the plug-in
  - If found, delegate to plugin
- Deployment unchanged; call to JavaScript function
  - Improved accuracy on which JRE versions are installed

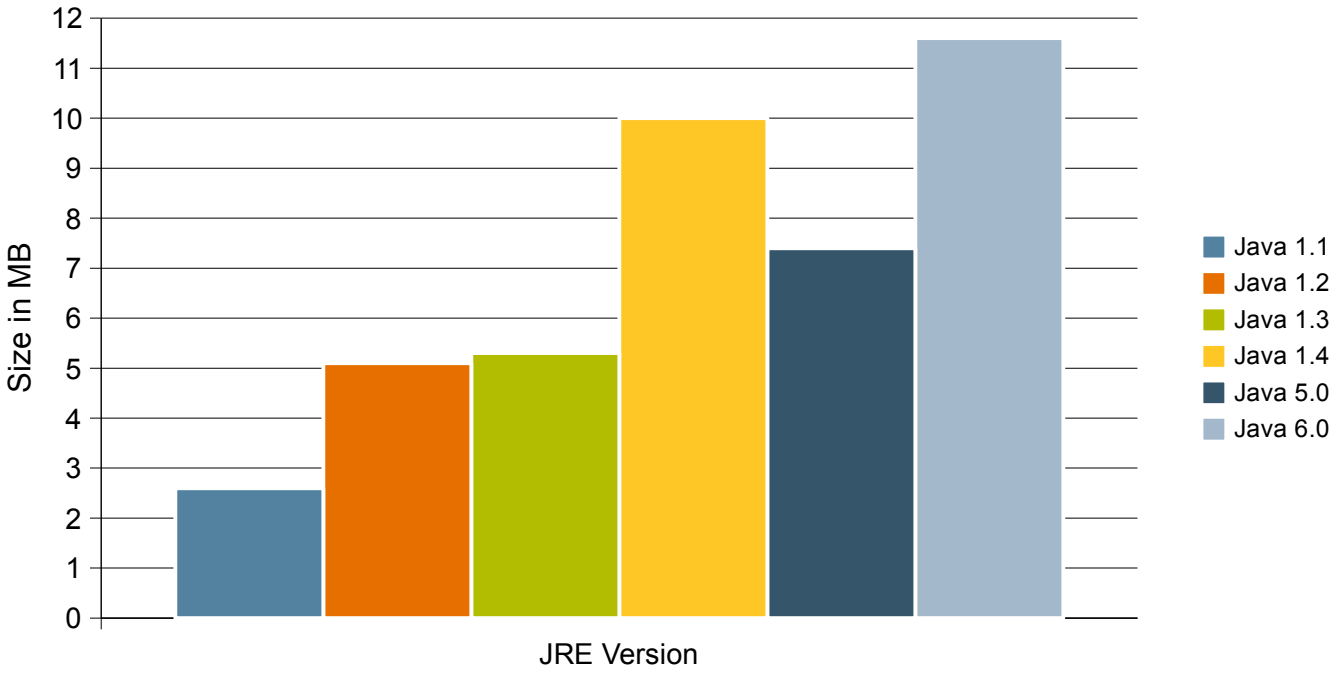
Problem: Too Many Java Versions

- Every Java Update version is installed as a separate program.
- Java Auto-Update cause many versions to be installed.
- Add / Remove Programs is filled with many versions of the JRE.

Solution: Patch in Place

- Patch in Place allows updating an existing Java version to a later update of the same family.
- Static installation will be available for enterprises that need to rely on static versioning.
- Add / Remove Programs will only see one non-static Java Runtime installation in each family.
  - Existing installations are all Static
  - New versions explicitly installed statically will also be shown.

JRE Release Size



Solution: Incremental Update

- Download only the incremental changes from the version that the client machine actually has.
  - Separate patches depending on existing install.
    - For example, for 6 update 8, this would include :
      6u5 -> 6u8,
      6u6 -> 6u8, and 6u7 -> 6u8.
- No longer need to copy and install base images during initial install of the first version of a family.
  - Faster installation of initial version
- Occupy less disk space by not retaining base images.

Modularization

- Modularize the JRE software
- A core part of the JRE is defined as the kernel
  - Enough functionality to run basic program like 'Hello World'
- The remaining JRE components are downloaded on demand, or lazily downloaded

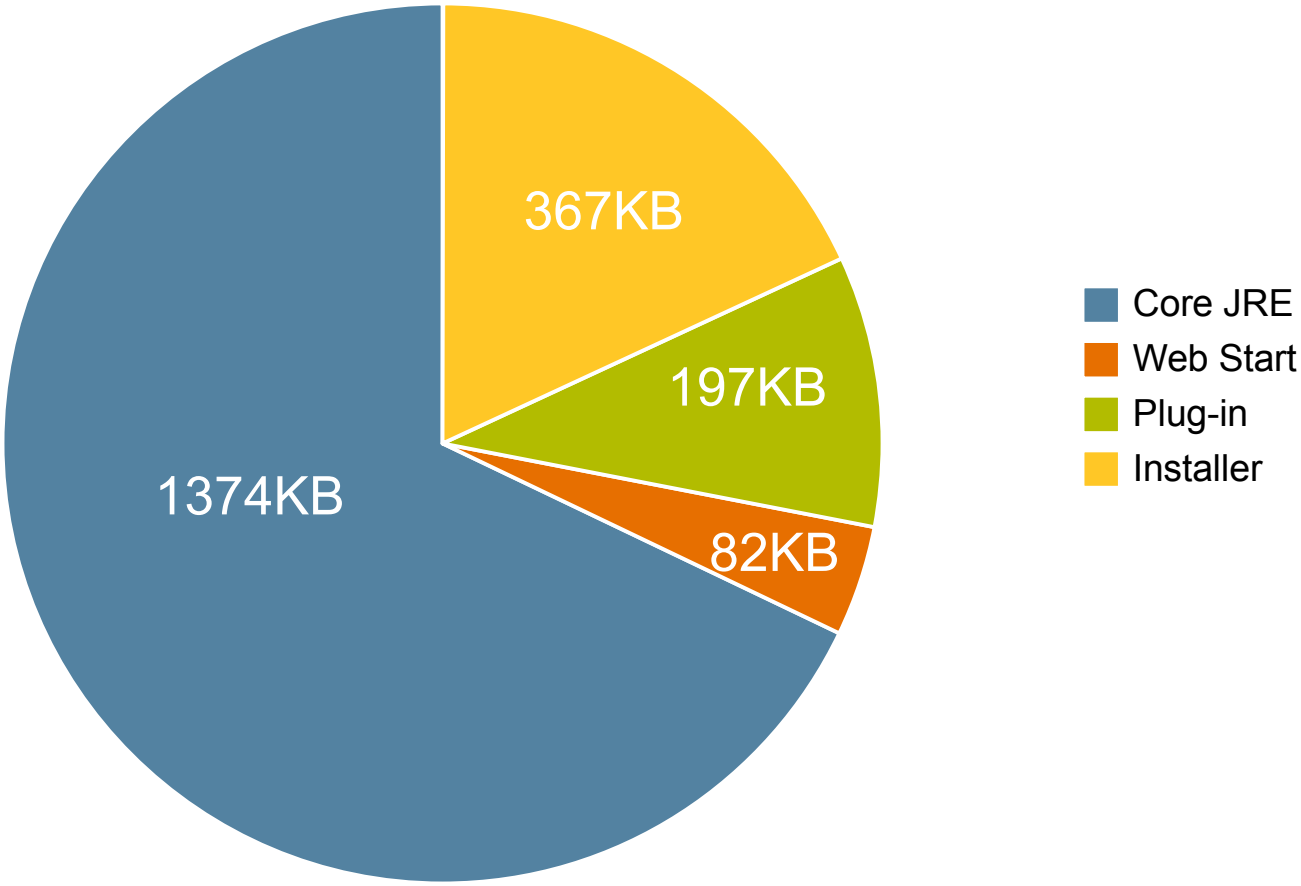
Java Kernel

- Every app needs some core functionality
  - VM, networking, security, classloader
- … plus other stuff on demand
  - Swing, AWT, 2D
- Kernel downloads and installs:
  - Bare essentials immediately
  - Additional dependencies on demand
    - Referencing a class
    - Class.getResource() or equivalent
    - System.loadLibrary() or equivalent
  - Everything else in the background

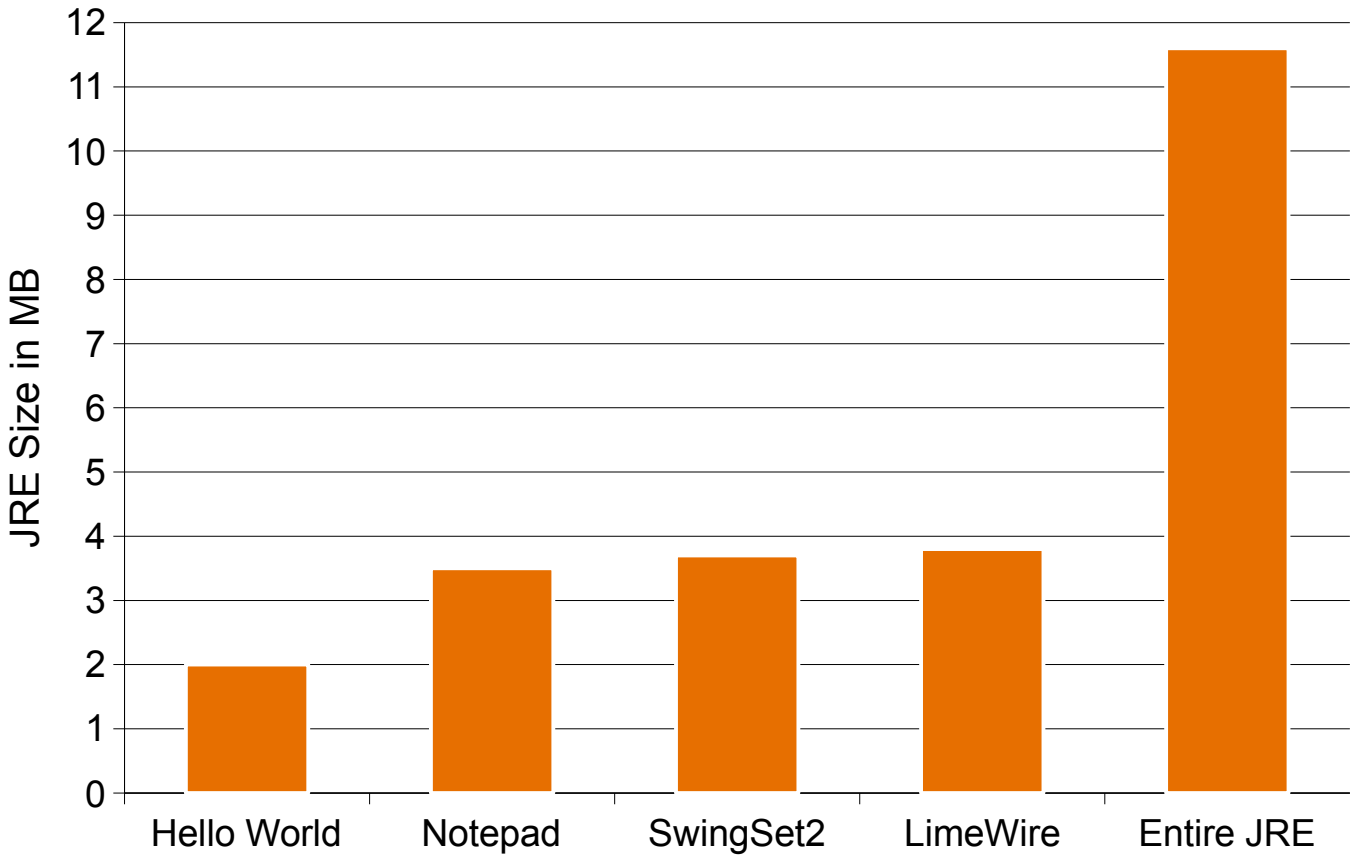
Java Kernel Structure

- Similar to the JRE structure
- Primary differences
  - Much smaller rt.jar file
  - Many files not present
- Missing class files are grouped into logical components
  - javax_swing, java_net, etc
  - Based on package boundaries

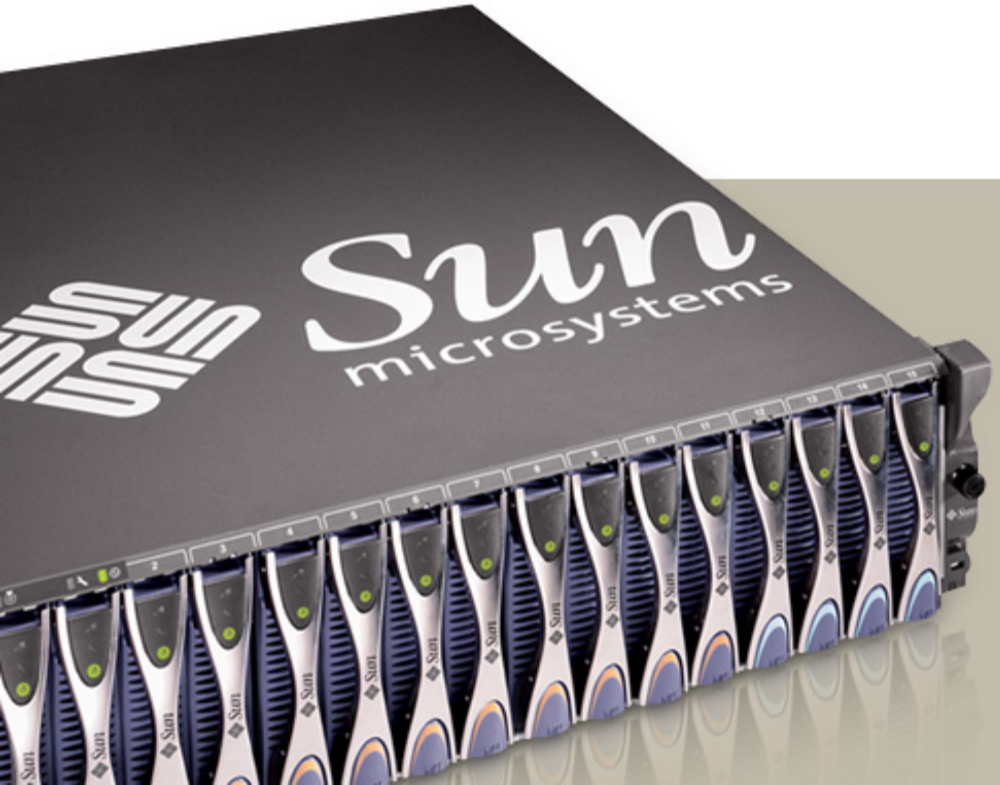
Kernel: Bare Essentials



Estimated Download Sizes



Java SE 7*



"Why don't you add X to Java?"

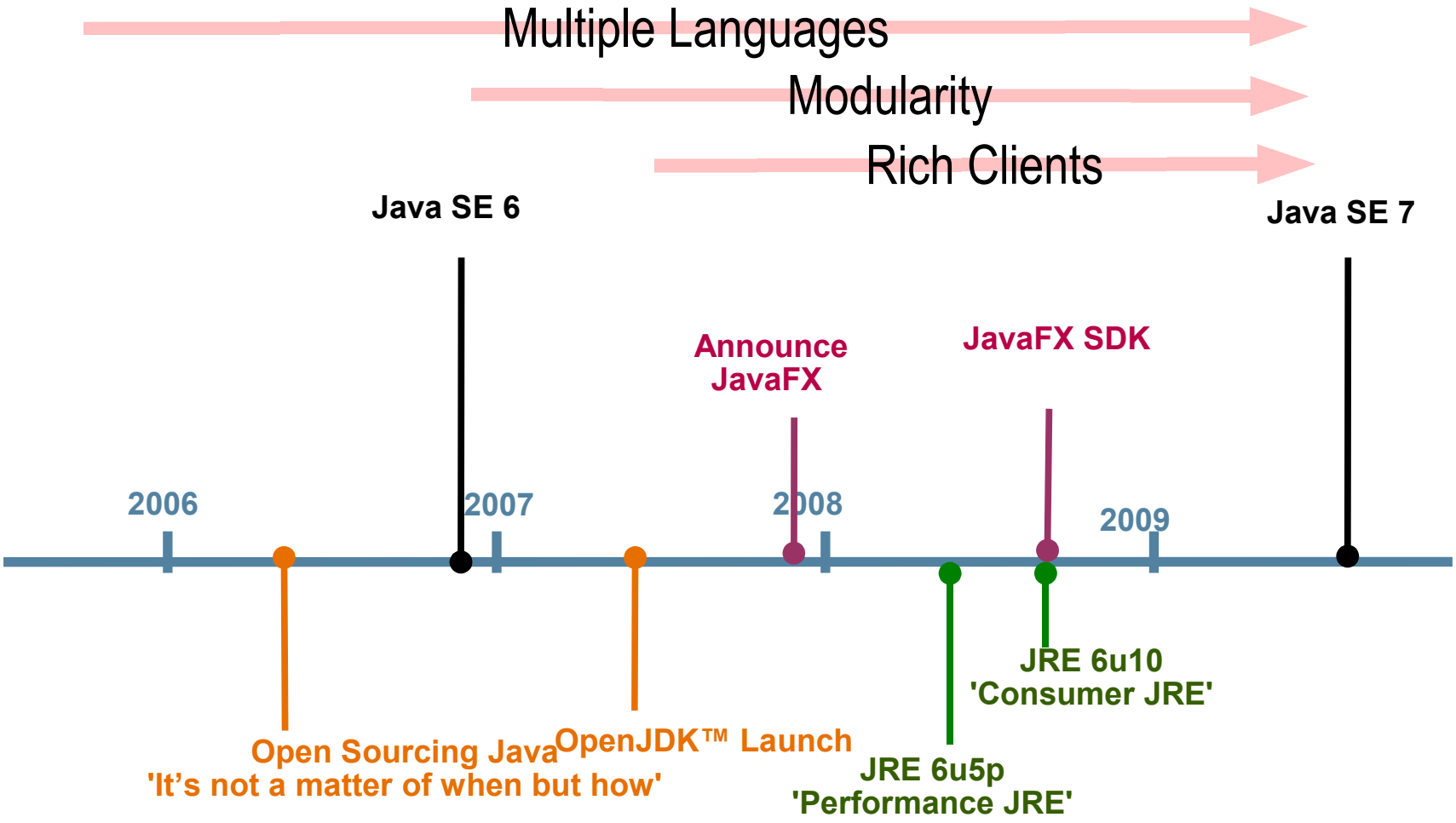
- **Assumption** is that adding features is always good
- Application
  - Application competes on the basis of completeness
  - User cannot do X until application supports it
  - Features *rarely* interacts "intimately" with each other
  - Conclusion: more features are better
- Language
  - Languages (most) are Turing complete
  - Can always do X; question is how elegantly
  - Features *often* interact with each other
  - Conclusion: fewer, more regular features are better

Adding X To Java

- Must be compatible with existing code
  - assert and enum keywords breaks old code
- Must respect Java's abstract model
  - Should we add ability to do inline bytecode like C?
- Must leave room for future expansion
  - Syntax/semantics of new feature should not conflict with syntax/semantics of existing and/or **potential** features
  - Allow consistent evolution eg. keyword parameters

```
@Point(x=3, y=4)          – keyword parameter
new Point(x = 3, y = 4) – assigment
new Point(x:3, y:3)       – possible syntax
@Point(x:3, y:3)          – what about this?
```

Java Platform Roadmap at a Glance



Big New Features From Sun

- Modularization (JSR-294, Project Jigsaw)
- JSR-292: VM support for dynamic languages
- JSR-TBD: Small language changes
- JSR-203: More new IO APIs
- JSR-296: Swing Application Framework

Project Jigsaw
The Modular JDK

Hello World

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

$ javac Hello.java
```

Hello World

```
$ time java Hello
Hello world

real 0m0.077s
user 0m0.022s
sys  0m0.000s
$
```

Hello World

```
$ time java Hello
Hello world

real 0m0.077s
user 0m0.022s
sys  0m0.000s
$ time python -c 'print “Hello world”'
Hello World

real 0m0.009s
user 0m0.008s
sys  0m0.000s
```

Hello World

```
$ java -verbose:classes Hello | wc -l
322
```

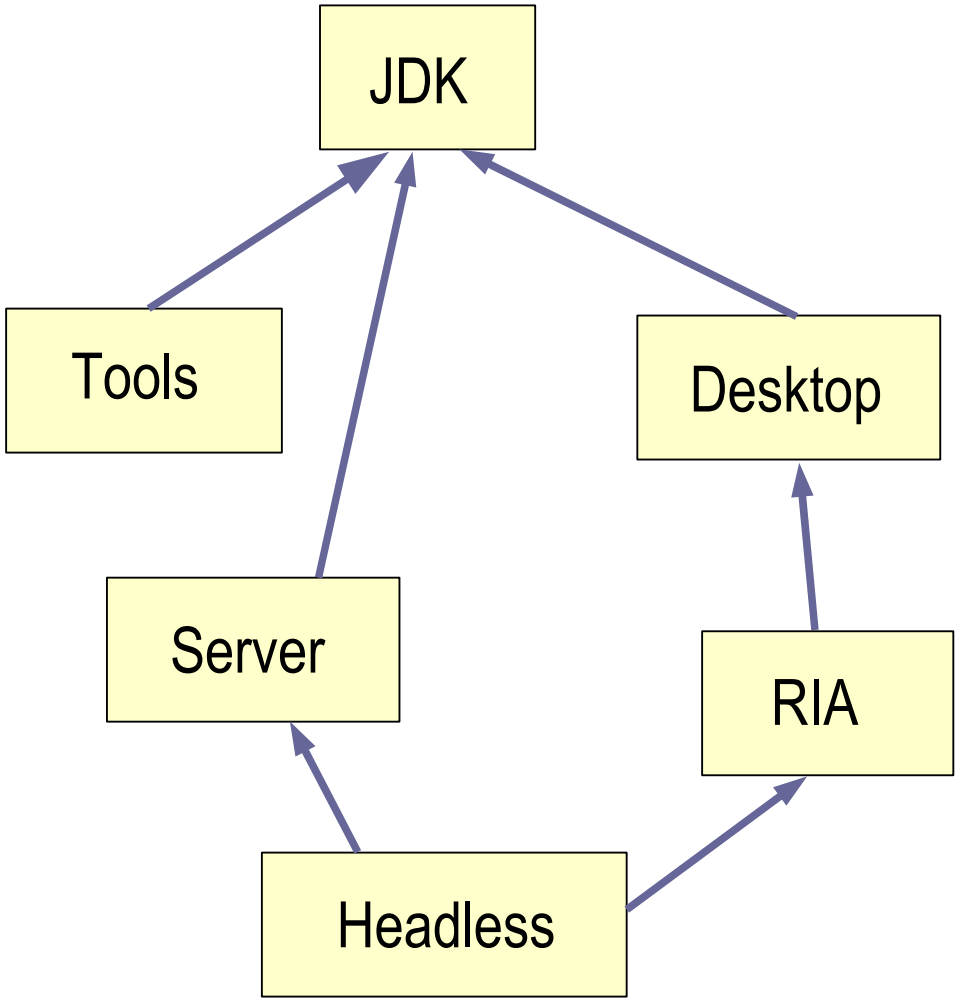
Requirements of a Platform Module System

- Integrate with the VM
- Integrate with the language
- Integrate with native packaging
- Support multi-module packages
- Support “friend” modules

New Module System For Java

- JSR-277: JAM module system
  - Sun has decided to halt development of this JSR until after Java SE 7
- JSR-294: Improved Modularity Support
  - Revived and expanded expert group
- OSGi
  - Looking at integration

The Modular JDK: Project Jigsaw



Small Language Changes

Safe Re-throw

```
void open() throws InvalidClassException
        , FileNotFoundException {
    try { ...
    } catch (Throwable e) {
        logger.log(e);
        throw e;
    }
```

Error: Unreported exception Throwable

- We want to express we are rethrowing the exception

```
void open() throws InvalidClassException
        , FileNotFoundException {
    try { ...
    } catch (final Throwable e) {
        logger.log(e);
        throw e;
    }
```

Compiles OK

Multi Catch

```
try { ...
} catch (InvalidClassException e) { foo(); }
} catch (InvalidObjectException e) { foo(); }
} catch (FileNotFoundException e) { bar(); }
```

- Longstanding request to allow catching Ex1 and Ex2 together

```
try { ...
} catch (InvalidClassException,
         InvalidObjectException e1) { foo(); }
} catch (FileNotFoundException e2) { bar(); }
```

- Members of e1 and e2 have direct common superclass
  - `ObjectStreamException`

Null Dereference Expression

```
T x = null;
if (a != null) {
  B b = a.b;
  if (b != null) {
    C c = b.c;
    if (c != null)
      x = c.x;
  }
}
```

Null Dereference Expression

```
T x = null;
if (a != null) {
  B b = a.b();
  if (b != null) {
    C c = b.c();
    if (c != null)
      x = c.x();
  }
}
```

```
T x = a?.b()?.c()?.x();
```

Better Type Inference

```
Map<String, Integer> foo =
   new HashMap<String, Integer>();
```

Better Type Inference

```
Map<String, Integer> foo =
   new HashMap<String, Integer>();

Map<String, Integer> foo =
   new HashMap<>();
```

Small Features From Sun

- SCTP Stream Control Transport Protocol
- SDP Sockets Direct Protocol
- Upgrade class loader architecture
- Method to close URLClassLoader
- Unicode 5.0 support
- XRender pipeline for Java2D
- Swing updates
  - JXLayer, DatePicker, CSS Styling (maybe)

New Features Not From Sun

- JSR-308: Annotations on Java types
  - Prof. Michael Ernst, Mahmood Ali
- Concurrency and collections updates
  - Doug Lea, Josh Bloch, etc
  - Fork/join framework
  - Phasers – generalised barriers
  - LinkedTransferQueue – Generalised queue
  - ConcurrentReferenceHashMap
  - Fences: Fine grained read/write ordering

Annotations Today

- Annotations on declarations only
  - Classes

```
@Deprecated class Signer { ...
```

  - Methods

```
@Override boolean equals(...
```

  - Fields

```
@Id String customerId;
```

  - Locals

```
@SuppressWarnings("unchecked")
        List<String> = new ArrayList();
```

- Allow annotations on type uses

JSR 308 – Annotations on Java Types

- Type checking prevents many bugs, but does not prevent enough bugs
- Cannot express important properties about code
  - Non null, interned, immutable, encrypted, ...

```
getValue().toString() //Potential NPE
```

Example of JSR 308

```
List<@NonNull String> stringList;

@NonEmpty List<String> stringList;

Graph g = new Graph();
...
//Now g2 will not be change
@Immutable Graph g2 = (@Immutable Graph)g;

//Method does not modify the object(fred)
fred.marshall(...)
void marshall(@Readonly Object jaxbElement
      , @Mutable Writer writer) @Readonly
```

File System API

- What's wrong with java.io.File?
  - No concept of file systems, attributes, 'link', storages, ...
- Proposed new API (main classes only)
  - FileSystem – factory for objects to access file and other objects in file system
  - FileRef – reference to a file or directory, contains methods to operate on then
  - Path – a FileRef that locates a file by a system dependent path
  - FileStore – underlying storage pool, device, partition, etc
- http://openjdk.java.net/projects/nio

JSR 166y.forkjoin

- "Free lunch is over"
  - Rely on faster CPUs to compensate for sloppy coding
- Multicore CPUs
  - Next speed bump will come by exploiting these
- Concurrency and parallelism techniques are no longer confined to HPC realm
- java.util.concurrent.forkjoin package
- Processor hints
  - Runtime.availableProcessors()

What Will Not Be In Java SE 7

- Closures
- Other language features
  - Reified generic types
  - First class properties
  - Operator overloading
  - BigDecimal syntax
- JSR-295: Bean binding

Summary

- Lots of new things coming
- Will make Java applications smaller, more concise, easier to read (and understand), less errors
- Lots of nice libraries that are going to exploit the hardware
- Platform will be more robust and scalable
- Expect Java SE 7 early 2010


Sun
microsystems
THANK YOU
Simon Ritter
Technology Evangelist
simon.ritter@sun.com
SUN TECH DAYS 2008–2009
A Worldwide Developer Conference