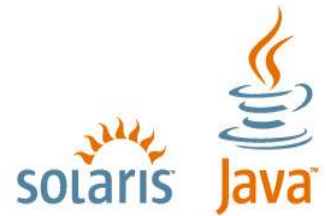




# Java SE 6.0 New Features

**Leon Fan**  
Senior Software Engineer  
Sun Microsystems



**UNLOCK  
OPPORTUNITY**

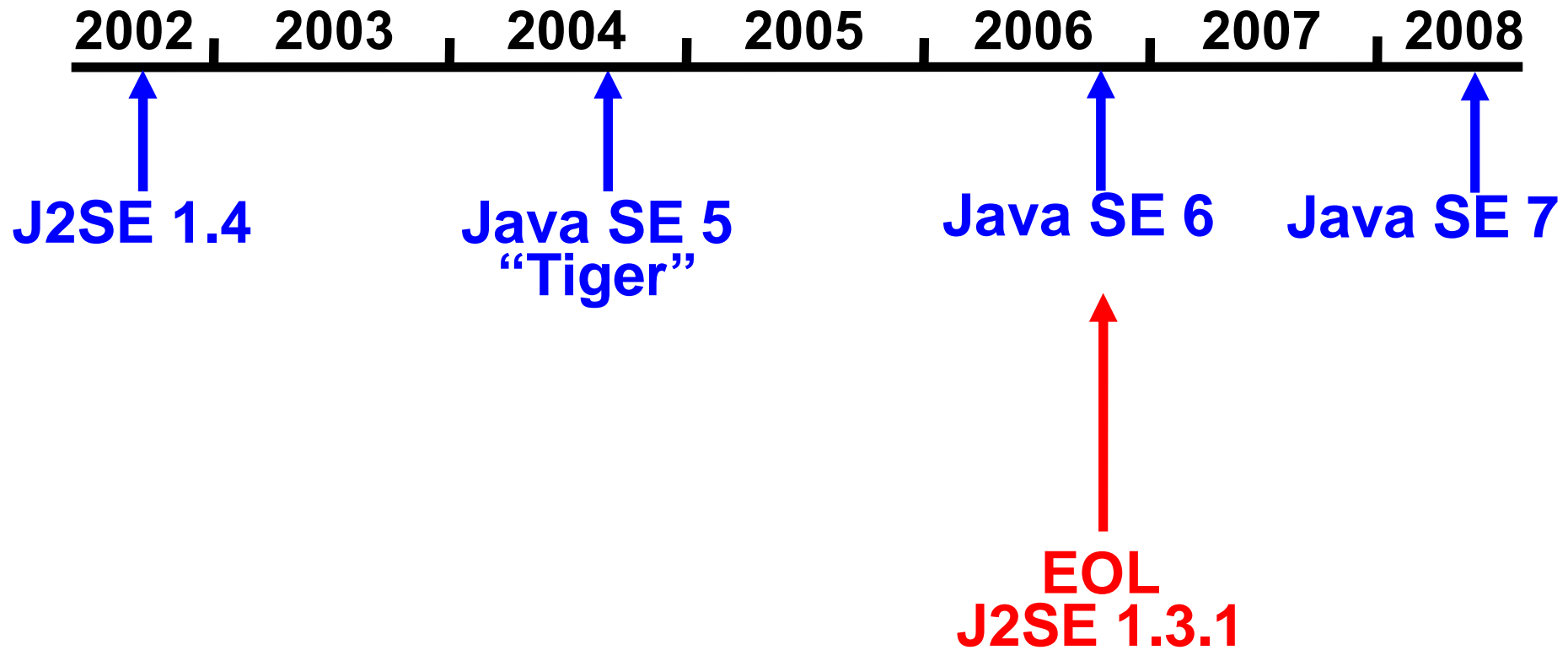
What will you open?

**SUN TECH DAYS 2006-2007**  
A Worldwide Developer Conference

# Agenda

- **Overview**
- **New Features Overview in Java SE 5**
- **New Features in Java SE 6**
- **Resources and Summary**

# Java SE road map



Compiler API Longhorn Look & Feel MBeans metadata JTable upgrades  
Splash screens Split Verifier Windows system tray Unicode Normalizer Services  
Parallelize Concurrent GC Attach on demand chmod  
JConsole upgrades Core JVM SwingWorker  
Annotation processors performance Parallel old-space GC  
Web Services Stack Password prompting  
JVM DTrace Docs in JDBC 4.0 GroupLayout LCD fonts  
Chinese More gfx acceleration Faster JNI JAXB 2.0 Free disk space  
JVM & CLR Co-Existence Improved OOM diagnosability  
FireFox support Pluggable Locales More desktop integration  
Native L&F Fidelity Scripting Languages  
More GC Ergonomics XAWT HTTP cookie manager  
XML digital signatures Improved text rendering JavaScript™ engine

- **New Features Overview in Java SE 5**
- New Features in Java SE 6

# Seven Major New Features

- Generics
- Autoboxing/Unboxing
- Enhanced for loop (“foreach”)
- Type-safe enumerations
- Varargs
- Static import
- Annotation (Metadata)

# Generics: Problem

```
Vector v = new Vector();  
v.add(new Integer(4));  
OtherClass.expurgate(v);  
...  
static void expurgate(Collection c) {  
    for (Iterator it = c.iterator(); it.hasNext();) {  
        /* ClassCastException */  
        if (((String)it.next()).length() == 4)  
            it.remove();  
    }  
}
```

Problem: Collection element types

- Compiler is unable to verify types
- Assignment must use the cast operator
- This can generate runtime errors (ClassCastException)

# Solution: Generics

- Tell the compiler what type the collection is
- Let the compiler fill in the cast
- Guaranteed to succeed

# Using Generic Classes:

- Instantiate a generic class to create type specific object
- Example

```
Vector<String> x = new Vector<String>();  
x.add(new Integer(5)); // Compiler error!
```

# Autoboxing of Primitive Types

- Problem:
  - > Conversion between primitive types and wrapper objects (and vice-versa)
  - > Needed when adding primitives to a collection

- Solution: Let the compiler do it

```
Byte byteObj = 22;           // Boxing conversion
int i = byteObj              // Unboxing conversion
```

```
ArrayList al = new ArrayList();
al.add(22); // Boxing conversion
```

# Enhanced for Loop Example

- Old code

```
void cancelAll(Collection c) {  
    for (Iterator i = c.iterator(); i.hasNext(); ){  
        TimerTask task = (TimerTask)i.next();  
        task.cancel();  
    }  
}
```

- New Code

```
void cancelAll(Collection<TimerTask> c) {  
    for (TimerTask task : c)  
        task.cancel();  
}
```

# Type-safe Enumerations

- Problem:
  - > Variable needs to hold limited set of values
  - > e.g. card suit can only be spade, diamond, club, heart
- Solution: New type of class declaration
  - > enum type has public, self-typed members for each enum constant
  - > New keyword, enum
  - > Works with switch statement

# Enumeration Example: 1

```
public enum Suit { spade, diamond, club, heart };  
public enum Value { ace, two, three, four, five,  
                  six, seven, eight, nine, ten,  
                  jack, queen, king };
```

```
List<Card> deck = new ArrayList<Card>();
```

```
for (Suit suit : Suit.values())  
    for (Value value : Value.values())  
        deck.add(new Card(suit, value));
```

```
Collections.shuffle(deck);
```

**Think how much JDK1.4 code this would require!**

# Annotation (JSR-175)

- Provide standardised way of adding annotations to Java code
- Like **Serializable** interface, javadoc comments and Xdoclets, but better
- Annotations are used by tools that work with Java code:
  - > Compiler
  - > IDE
  - > Runtime tools

# Example of Annotation Usage

```
public class PayrollBean implements javax.ejb.SessionBean {
    SessionContext ctx;
    DataSource empDB;
    public void setSessionContext(SessionContext ctx) {
        this.ctx = ctx;
    }
    public void ejbCreate() {
        empDB = (DataSource)ctx.lookup("jdbc/empDB");
    }
    public void ejbActivate() { }
    public void ejbPassivate() { }
    public void ejbRemove() { }
    public void setBenefitsDeduction(int empId,
                                     double deduction) {
        Connection conn = empDB.getConnection();
        ...
    }
}
```

# Example of Annotation Usage

**@Stateless**

```
public class PayrollBean implements Payroll
{
    @Resource private DataSource empDB;

    public void setBenefitsDeduction(in empId,
                                    double deduction) {
        ...
        Connection conn = empDB.getConnection();
        ...
    }
    ...
}
```

# Defining Annotations

- Defined like an interface
  - > @interface
- Definition contains parameters that can be specified when using the annotation
- Default values can be provided

# Meta-Annotations

- **@Retention**
  - > How long is annotation information kept
  - > **Enum RetentionPolicy**
    - > SOURCE, CLASS, RUNTIME
- **@Target**
  - > Restrictions on use of this annotation
  - > **Enum ElementType**
    - > TYPE, FIELD, METHOD, PARAMETER, CONSTRUCTOR, LOCAL\_VARIABLE, ANNOTATION\_TYPE, PACKAGE

# Example: Defining Annotation

```
@Target(ElementType.FIELD)
@Retention(RetentionPolicy.RUNTIME)
public @interface Accessor {
    String variableName();
    String variableType() default "String";
}
```

# Example: Process Annotation

```
@interface ToDo {
    String
value();
}
@ToDo("Should be package-private")
public class CountingStream
                extends InputStream
{
    @ToDo("Check EOF logic!")
    public int read() throws IOException {
        ...
    }
}
```

# Example: Process Annotation

```
% apt -factory ToDoProcessor CountingStream.java  
read() @ToDo(value=Check EOF logic!)  
CountingStream @ToDo(value=Should be package-private)  
%
```

- **New Features Overview in Java SE 5**
- **New Features in Java SE 6**

# JDK 6 Themes

- XML & Web Services
- Scripting Language Support
- Database
- Diagnosability, Monitoring, & Management
- Security Features and Enhancements
- Desktop API and Deployment
- Compiler Access
- Other features
- Compatibility, Stability, & Quality!

# XML & Web Services

- Streaming API for XML (JSR173)
- Java Architecture for XML Binding (JAXB) 2.0 (JSR222)
- SOAP with Attachments API for Java (SAAJ) 1.3
- Java API for XML Web Services (JAX-WS), version 2.0 (JAX-WS 2.0) (JSR224 & JSR181 & JSR250)

# Streaming API for XML

```
<favorites-list>
  <favorites>
    <who>Laurie</who>
    <color>9a661b</color>
  </favorites>
  <favorites>
    <who>Brian</who>
    <color>43b120</color>
  </favorites>
  <favorites>
    <who>Clara</who>
    <color>888314</color>
  </favorites>
  <favorites>
    <who>Jeff</who>
    <color>61f8ba</color>
  </favorites>
</favorites-list>
```

# Streaming API for XML(Cont.)

```
SAXParserFactory spf = SAXParserFactory.newInstance();
SAXParser sp = spf.newSAXParser();
final int[] s = new int[1];
final int[] n = new int[1];
sp.parse(in, new DefaultHandler() {
    private boolean inColor = false;
    StringBuider color = new StringBuilder();
    public void startElement(String uri, String ln,
                            String qn, Attributes as)
    { inColor = qn.equals("color"); }
    public void endElement(String uri, String ln, String qn)
    {
        if (qn.equals("color")) {
            s[0] += Integer.parseInt(color.toString(), 16);
            n[0]++;
            color.setLength(0);
            inColor = false;
        }
    }
    public void characters(char[] ch, int start, int len)
    { if (inColor) color.append(ch, start, len); }
});
return s[0]/n[0];
```

# Streaming API for XML(Cont.)

```
XMLInputFactory xif = XMLInputFactory.newInstance();
XMLStreamReader xr = xif.createXMLStreamReader(in);
int s = 0;
int n = 0;
while (xr.hasNext()) {
    if (xr.nextTag() == START_ELEMENT) {
        String ln = xr.getLocalName();
        if (ln.equals("who")) {
            xr.getElementText();
        } else if (ln.equals("color")) {
            s += Integer.parseInt(xr.getElementText(), 16);
            n++;
        }
    } else if (xr.getLocalName().equals("favorites-
list")) {
        break;
    }
}
return s/n;
```

# JAXB 2.0

```
<favorites-list>
  <favorites>
    <who>Laurie</who>
    <color>9a661b</color>
  </favorites>
  <favorites>
    <who>Brian</who>
    <color>43b120</color>
  </favorites>
  <favorites>
    <who>Clara</who>
    <color>888314</color>
  </favorites>
  <favorites>
    <who>Jeff</who>
    <color>61f8ba</color>
  </favorites>
</favorites-list>
```



```
Laurie
9a661b
Brian
43b120
Clara
888314
Jeff
61f8ba
```

# JAXB 2.0(Cont.)

```
public class Favorites {  
    private String who;  
    public String getWho() { return who; }  
    public void setWho(String w) { who = w; }  
    private String color;  
    public String getColor() { return color; }  
    public void setColor(String c) { color = c; }  
}
```

# JAXB 2.0(Cont.)

```
import javax.xml.bind.annotation.*;
@XmlType
public class Favorites {
    private String who;
    public String getWho() { return who; }
    public void setWho(String w) { who = w; }
    private String color;
    public String getColor() { return color; }
    public void setColor(String c) { color = c; }
}
```

# JAXB 2.0(Cont.)

```
void summarize(List<Favorites> faves) {  
    for (Favorites fc : faves)  
        System.out.format("%8s %s%n",  
                            fc.getWho(),  
                            fc.getColor());  
}
```

# JAXB 2.0(Cont.)

```
void summarize(List<Favorites> faves) {  
    for (Favorites fc : faves)  
        System.out.format("%8s %s%n",  
                            fc.getWho(),  
                            fc.getColor());  
}
```

```
% java Summarize <favorites.xml
```

```
Laurie 9a661b
```

```
Brian 43b120
```

```
Clara 888314
```

```
Jeff 61f8ba
```

```
%
```

# JAX-WS 2.0

```
public class ColorGenerator {  
    public int generate(String name) {  
        return name.hashCode() & 0xffffffff;  
    }  
}
```

# JSR 224: JAX-WS 2.0 (&181 &250)

```
import javax.jws.*;
```

```
@WebService
```

```
public class ColorGenerator {  
    public int generate(String name) {  
        return name.hashCode() & 0xffffffff;  
    }  
}
```

# JSR 224: JAX-WS 2.0 (&181 &250)

```
import java.rmi.RemoteException;

public static int myFavoriteColor()
    throws WebServiceException, RemoteException
{
    ColorGeneratorService service
        = new ColorGeneratorService();
    ColorGenerator cg = service.getColorGeneratorPort();
    return cg.generate(System.getProperty("user.name"));
}
```

# Scripting Language Support

- Access and control Java technology-based objects from a scripting environment
- Create web content with scripting languages
- Embed scripting environments within Java technology-based applications

# Scripting Language Support (Cont.)

- Scripting API
  - > javax.script package available
- Available Script Engines
  - > Mozilla Rhino engine bundled
  - > Can add more script engines if there have
- Tool
  - > jrunscript

# Scripting API

```
import javax.script.*;
public class Main {
    public static void main(String[] args)
        throws ScriptException {
        // create a script engine manager
        ScriptEngineManager factory = new
            ScriptEngineManager();
        // create JavaScript engine
        ScriptEngine engine =
            factory.getEngineByName("JavaScript")
            ;
        // evaluate JavaScript code from String
        engine.eval("print('hello world')");
    }
}
```

# Tool

- Jrunscript
  - > command line script shell
  - > supports both an interactive (read-eval-print) mode and a batch (-f option) mode of script execution
  - > By default, JavaScript is used
- Access Java Object in javascript

```
C:\>jrunscript
```

```
js> importPackage(javax.swing)
```

```
js> JOptionPane.showMessageDialog(null, 'Hello  
world!');
```

# Database

- Early Access version of Java DB
  - > built from Apache Derby 10.2.1.7
  - > Can be run under in an embedded environment or in a server environment
  - > Use JDBC API to operate it
- JDBC 4.0
  - > Auto-loading of JDBC driver class
  - > Connection management enhancements
  - > Support for RowId SQL type
  - > Chained exceptions

# Database (cont.)

- JDBC 4.0 (cont.)
  - > Enhanced Support for Large Objects (BLOBs and CLOBs)

# JDBC 4.0 API

```
private Connection connect(String user, String passwd)
    throws SQLException
{
    String url = "jdbc:mysql://javadb.sfbay/jplan";
    String driver = "com.mysql.jdbc.Driver";
    try {
        Class.forName(driver);
        return DriverManager.getConnection(url,
                                         user,
                                         passwd);
    } catch (ClassNotFoundException x) {
        throw new SQLException("Can't load Driver " + x);
    }
}
```

# JDBC 4.0 API (Cont.)

```
private Connection connect(String user, String passwd)
    throws SQLException
{
    String url = "jdbc:mysql://javadb.sfbay/jplan";
    String driver = "com.mysql.jdbc.Driver";
    try {
        Class.forName(driver);
        return DriverManager.getConnection(url,
                                           user,
                                           passwd);
    } catch (ClassNotFoundException x) {
        throw new SQLException("Can't load Driver " + x);
    }
}
```

# Diagnosability, Monitoring & Management

- Common Problems in Java SE App.
- Tools provided to diag., monitoring & management
- How to Diagnosis with JDK tools

# Common Problems

- Insufficient memory
  - > OutOfMemoryError
- Memory leaks
  - > Growing use of memory; Frequent garbage collection
- Finalizers
  - > Objects are pending for finalization
- Deadlocks
  - > Threads block on object monitor or `java.util.concurrent` locks

# Common Problems(Cont.)

- Looping threads
  - > Thread CPU time is continuously increasing
- High lock contention
  - > Thread with high contention statistics

# Tools

- Jconsole (from 5.0)
- jstat (from 5.0)
  - > various VM statistics including memory usage, garbage collection time, class loading, and the just-in-time compiler statistics
- jmap
  - > obtain a heap histogram and a heap dump at runtime
- jhat
  - > analyze a heap dump
- Jstack
  - > obtain a thread stack trace

# Improved JConsole



- Enhanced UI
- New Overview and VM Summary tabs
- Attach On Demand
- JConsole Plugin Support
- Improved presentation of MBeans

# Diagnosis

- Ways to Diagnose a Memory Leak
  - > JConsole to monitor whether the memory usage is growing continuously
  - > Or jstat command to monitor the memory usage and garbage collection statistics
  - > jmap command will help you get a heap histogram
  - > Dump heap by jmap
  - > Heap Analysis by jhat
- Ways to pending finalization
  - > Show in Jconsole or dump report in jhat

# Diagnosis (Cont.)

- Ways to Diagnose Deadlocks
  - > Deadlock tab display in Jconsole
  - > Or directly deadlock information in jstack output
- Ways to Diagnose Looping Threads
  - > JTop - JDK demo shows an application's usage of CPU time per thread
- Ways to Diagnose High Lock Contention
  - > Thread tab in Jconsole will help you to determining which locks are the bottleneck

# Security Features and Enhance.

- Microsoft CryptoAPI Access
- PKCS#11 Cryptographic Services Access
- Native GSS-API Access
- PKCS#12 Keystores Import and Export
- XML Digital Signature

# Microsoft CryptoAPI Access

- MS CryptoAPI (CAPI)
  - > a standard interface for performing cryptographic operations as well as accessing the user keys and certificates that Windows manages
- SunMSCAPI provider
  - > Access private keys and certificates stored in CAPI
  - > Use CAPI's cryptographic algorithm implementations

# PKCS#11 Cryptographic Services Access

- PKCS#11 - Cryptographic Token Interface Standard
- New Java provider gain benefit
  - > Cryptographic smart cards for added security
  - > Hardware cryptographic accelerators for better performance
  - > Software implementations for more algorithms or for meeting certification requirements

# Native GSS-API Access

- GSS-API - a generic security API
  - > authenticate a principal
  - > delegate its rights to a peer,
  - > apply security services such as confidentiality and integrity on a per-message basis
- Using native GSS-API
  - > the native credentials and settings in users' environment will be picked up automatically
  - > seamless integration with native applications

# PKCS#12 Keystores Import and Export

- PKCS#12 - Personal Information Exchange Syntax Standard
  - > a portable format for storing or transporting personal identity information, including private keys, certificates, and so on
- Two to import and Export
  - > Using keytool
  - > Programmatically through the `java.security.KeyStore` APIs

# Conclusion

- MS CryptoAPI Access
  - > JDK6: 32-bit MS Windows
- PKCS#11 Access
  - > JDK5.0: 32/64 bit Solaris, 32bit Linux, 32bit MS Windows
  - > PKCS#11 v2.0 or later
- Native GSS-API
  - > JKD6: Solaris, Linux
- PKCS#12 Import/Export
  - > JDK1.4: All
  - > Interoperable with MS IE, Netscape, NSS and OpenSSL

# Desktop

- Better platform look-and-feel in Swing technology
  - > match the underlying platform's look and feel
- LCD text rendering
- Access to the platform's System Tray and Start menu
- Splash-Screen Functionality
- Desktop Integration

# System Tray

```
import java.awt.SystemTray;
...
if (SystemTray.isSupported()) {
    SystemTray tray =SystemTray.getSystemTray();
    ...
    trayIcon = new TrayIcon(image, "Tray Demo", popup);
    trayIcon.addMouseListener(mouseListener);
    try {
        tray.add(trayIcon);
    } catch (AWTException e) {
        System.err.println("TrayIcon could not be
added.");
    }
}
```

# Splash-Screen Functionality

- Supported image format
  - > GIF, PNG, or JPEG
- Ways to show the native splash screen
  - > `java -splash:filename.gif <java program>`
  - > Put following content in manifest.mf file of jar file:
    - Manifest-Version: 1.0
    - Main-Class: <main class>
    - SplashScreen-Image: filename.gif
- `SplashScreen splash = SplashScreen.getSplashScreen()`

# Splash-Screen Functionality(Cont.)

- A SplashScreen class provided
  - > close the splash screen
  - > change the splash-screen image,
  - > get the image position or size,
  - > paint in the splash screen
- Only can get a instance if application has not created the splash screen at startup by following code:

```
SplashScreen splash =  
    SplashScreen.getSplashScreen()
```

# Java Desktop integration

- java.awt.Desktop API
  - > Launching the default browser
  - > Launching the default email client
  - > Launching the default application to handle file open, file edit and file print

```
if (!Desktop.isDesktopSupported()) { //exit }
```

```
Desktop desktop = Desktop.getDesktop();
```

```
if (!desktop.isSupported(Desktop.PRINT)  
    { //print not supported }
```

```
desktop.browse(new URI(urlField.getText()));
```

# Compiler Access

```
import javax.tools.*;

String sourceFile = "Hello.java";
JavaCompiler compiler = ToolProvider.getSystemJavaCompiler();
StandardJavaFileManager fileManager =
    compiler.getStandardFileManager(null, null, null);
Iterable compilationUnits =
    fileManager.getJavaFileObjects(sourceFile);
JavaCompiler.CompilationTask task =
    compiler.getTask(null, fileManager, null, null, null,
    compilationUnits);

task.call();
fileManager.close();
```

# Other features

- Annotation Processors
- Free Disk Space
- ClassPath Wildcards
- Improved out of memory Handling

# Annotation Processors

- Annotation Processing library move to javax package
- Javac offers a -processor option instead of apt
- Removal of the Processor factory

# Annotation Processors (Cont.)

```
public class ToDoProcessor
    implements AnnotationProcessor
{
    ...
    public void process() {
        AnnotationTypeDeclaration todoDecl =
atds.iterator().next();
        for(Declaration decl :
            env.getDeclarationsAnnotatedWith(todoDecl)) {
            System.out.format("%-16s %s%n",
                decl.toString(),
                decl.getAnnotation(ToDo.Class));
        }
    }
}
```

# Annotation Processors(Cont.)

```
public class ToDoProcessor
    extends AbstractProcessor
{
    ...
    public boolean process(Set<? extends TypeElement>
annotations, RoundEnvironment roundEnv) {
        if (!roundEnv.processingOver()) {
            for (TypeElement element : annotations) {
                System.out.println(element.getQualifiedName() +
                    "(" + element.getNestingKind() + ")");
            }
        }
        return false;
    }
}
```

# Free Disk Space

```
void safeCopy(File src, File dstDir)
                throws IOException
{
    if (src.length() > dstDir.getUsableSpace())
        throw new IOException("Insufficient space");
    doCopy(src, new File(dstDir.getParent(),
        src.getName()));
}
```

# Free Disk Space (Cont.)

```
void df(File dir) {
    out.format("Total MB Used Free Use%%\n");
    long t = dir.getTotalSpace();
    long f = dir.getFreeSpace();
    out.format(" %6d %6d %6d %2d%%\n",
        t >> 20,
        (t - f) >> 20,
        f >> 20,
        ((t - f) * 100) / t);
}
```

```
%java df / a
```

Total MB	Used	Free	Use%
32766	28632	4134	87%

# ClassPath Wildcards

```
%javac -cp ../javab/lib/javaxb-api.jar\  
../javaxb/lib/javaxb-impl.jar\  
../javaxb/lib/jsr173_1.0_api.jar\  
../javaxb/lib/activation.jar \  
Something.java
```

# ClassPath Wildcards

```
%javac -cp ../javab/lib/javaxb-api.jar\  
../javaxb/lib/javaxb-impl.jar\  
../javaxb/lib/jsr173_1.0_api.jar\  
../javaxb/lib/activation.jar \  
Something.java
```

```
% javac -cp ' ../jaxb/lib/*.jar' Sum.java
```

# Improved out of memory Handling

```
public class Chew {
    static void chew() {
        List<Object> ls = new ArrayList<Object>();
        for (;;)
            ls.add(new byte[1 << 20]);
    }
    public static void main(String[] args) {
        chew();
    }
}
```

% java Chew

Exception java.lang.OutOfMemoryError: Java heap space

# Improved out of memory Handling (Cont.)

```
public class Chew {
    static void chew() {
        List<Object> ls = new ArrayList<Object>();
        for (;;)
            ls.add(new byte[1 << 20]);
    }
    public static void main(String[] args) {
        chew();
    }
}
```

```
% java Chew
```

```
Exception java.lang.OutOfMemoryError: Java heap space
```

```
    at X.chew(X.java:189)
```

```
    at X.main(X.java:193)
```

# Compatibility, Stability, & Quality

- Compatibility
  - > upwards binary-compatible
  - > 11 minor potential source and binary incompatibilities in release notes
- Stability & Quality
  - > 115 p1& p2 bugs & rfe fixed
  - > More p3,p4&p5 bugs fixed

# Resources and Summary

# Call for Action!

- Download and install JDK 6
- Start using the new language features ASAP

**I NEED**

**YOU**

*for*

**JDK**

**DEVELOPMENT**



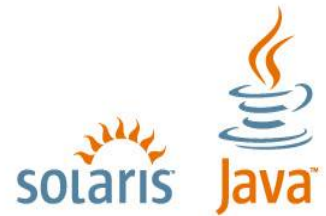
# Resources

- <http://java.sun.com/javase>
- [Java.net](http://java.net)
- <http://jdk.dev.java.net>



# Java SE 6.0 New Features

**Leon Fan**  
Senior Software Engineer  
Sun Microsystems



**UNLOCK  
OPPORTUNITY**

What will you open?

**SUN TECH DAYS 2006-2007**  
A Worldwide Developer Conference