Comp-303 : Programming Techniques
Lecture 23

Alexandre Denault
Computer Science
McGill University
Winter 2004

Last lecture . . .

- The Command design pattern allows you to seperate the Invoker from the Receiver, thus allowing you to create parametrizable frameworks.
- The Command design pattern also allows you to do nifty things like queing commands, undo, redo, transactions, etc.

Why is it called J2SE?

- Because there exists three different types of Java:
  - J2ME is the Java 2 Micro Edition that contains a reduce core of Java classes for developers that code to portable devices. Use this to write code for a PDA.
  - J2SE is the Java 2 Standard Edition that contains the basic core Java classes. Use this to to write your standard applets and applications.
  - J2EE is the Java 2 Enterprise Edition that contains the extented core Java classes (such as Security API, Java Mail API, XML Parsers etc.). Use this to write your server components (such as Servlets).

Java Community Process

- Java is an open standard. Sun does not control the evolution of the Java specification.
- The Java Community Process guides the development and approval of Java technical specifications.
- Anyone can join the JCP and have a part in its process, and you don't even have to join to contribute as a public participant.
- The work of the JCP helps to ensure Java technology's standard of stability and cross-platform compatibility:
  - desktop computers
  - consumer electronics
  - industrial robots

Java Specification Requests

- Java Specification Requests (JSRs) are the actual descriptions of proposed and final specifications for the Java platform.
- At any one time there are numerous JSRs moving through the review and approval process.
- A JSR can take up to 200 days before it is approved by the JCP.

Life and Death of a JSR?

- JSR Review : A specification is initiated by community members and approved for development by the Executive Committee.
- Community Review : Once a JSR is approved, a group of experts is formed to develop a first draft of the specification.
- Public Review : The JSR draft goes out for review by the public where anyone with an Internet connection can read and comment on the draft.
- Proposed Final Draft : The version of the draft specification that will be used as the basis for the RI and TCK.
- Final Release : The leader of the Expert Group then sees that the reference implementation and its associated Technology Compatibility Kit are completed before sending the specification to the Executive Committee for final approval.

Executive Members of JCP for J2SE/J2EE

Apache Software Foundation	Apple
BEA Systems	Borland
Fujitsu Limited	Hewlett-Packard
IBM	IONA Technologies
Doug Lea	Macromedia
Richard Monson-Haefel	Nokia Networks
Oracle	SAP
The SCO Group	Sun Microsystems

Executive Members of JCP for J2ME

Ericsson Mobile Platforms	IBM
Insignia	Intel
Matsushita	Motorola
Nokia	Philips
Research In Motion	Siemens
Sony	Sony-Ericsson
Sun Microsystems	Symbian
Texas Instruments	Vodafone

Java 1.5

- Released February 2004, J2SE 1.5, code-named Tiger, is the first major improvement to the Java programming language since two years ago when version 1.4 was released.
- Improvements include:
  - enumerated types
  - metadata/autoboxing of primitive types
  - enhanced for loops
  - improved diagnostics and for the first time
  - the use of generics

Metadata

- The Metadata feature in J2SE 1.5 provides the ability to associate additional data alongside Java classes, interfaces, methods, and fields.
- This additional data, or annotation, can be read by the javac compiler or other tools, and depending on configuration can also be stored in the class file and can be discovered at runtime using the Java reflection API.
- For example:

```
@Overrides public boolean equals(Foo that) { ... }
```

- This annotation type allows the programmer to declare his belief that a method declaration overrides a superclass method.
- The compiler checks whether method actually overrides a superclass method, and reports an error if it does not, nipping the problem in the bud.

Generic Types

- Generic types enable an API designer to provide common functionality that can be used with multiple data types and which also can be checked for type safety at compile time.
- Before

```
ArrayList list = new ArrayList();
list.add(0, new Integer(42));
int total = ((Integer)list.get(0)).intValue();
```

- After

```
ArrayList<Integer> list =  new ArrayList<Integer>();
list.add(0, new Integer(42));
int total = list.get(0).intValue();
```

- You can't use a primitive as a type variable for your generic collection.

Autoboxing

- Converting between primitive types, like int, boolean, and their equivalent Object-based counterparts like Integer and Boolean, can require unnecessary amounts of extra coding.
- The autoboxing and auto-unboxing of Java primitives produces code that is more concise and easier to follow.
- Before

```
ArrayList<Integer> list = new ArrayList<Integer>();
list.add(0, new Integer(42));
int total = (list.get(0)).intValue();
```

- After

```
ArrayList<Integer> list = new ArrayList<Integer>();
list.add(0, 42);
int total = list.get(0);
```

Enhanced for loop

- The Iterator class is used heavily by the Collections API.
- It provides the mechanism to navigate sequentially through a Collection.
- The new enhanced for loop can replace the iterator when simply traversing through a Collection.
- Before

```
ArrayList<Integer> list = new ArrayList<Integer>();
for (Iterator i = list.iterator(); i.hasNext();) {
  Integer value=(Integer)i.next();
}
```

- After

```
ArrayList<Integer> list = new ArrayList<Integer>();
for (Integer i : list) { ... }
```

Enumerated types

- This type provides enumerated type when compared to using static final constants.

```
public enum StopLight { red, amber, green };
```

Static Import

- The static import feature, implemented as *import static*, enables you to refer to static constants from a class without needing to inherit from it.
- Instead of *BorderLayout.CENTER* each time we add a component, we can simply refer to *CENTER*.

```
import static java.awt.BorderLayout.*;
getContentPane().add(new JPanel(), CENTER);
```

Formatted Output

- Developers now have the option of using printf type functionality to generated formatted output.
- This will help migrate legacy C applications, as the same text layout can be preserved with little or no change.
- Most of the common C *printf* formatters are available, and in addition some Java classes like Date and BigInteger also have formatting rules.

```
System.out.printf("name count\n");
System.out.printf("%s %5d\n", user,total);
```

Formatted Input

- The scanner API provides basic input functionality for reading data from the system console or any data stream.
- The Scanner methods like next and nextInt will block if no data is available.
- If you need to process more complex input then there are also pattern matching algorithms, available from the java.util.Formatter class.
- The following example reads a String from standard input and expects a following int value.

```
Scanner s = Scanner.create(System.in);
String param = s.next();
int value = s.nextInt();
s.close();
```

Varargs

- The varargs functionality allows multiple arguments to be passed as parameters to methods.
- It requires the simple ... notation for the method that accepts the argument list and is used to implement the flexible number of arguments required for printf.

```
void argtest(Object ... args) {
  for (int i=0;i <args.length; i++) {
  }
}

argtest("test", "data");
```

Concurrency Utilities

- The concurrency utility library is a special release of the popular concurrency package into the J2SE 1.5 platform.
- It provides powerful, high-level thread constructs, including executors, which are
  - a thread task framework
  - thread safe queues
  - Timers
  - locks (including atomic ones)
  - other synchronization primitives.

Concurrency Utilities : Semaphores

- A semaphore can be used to restrict access to a block of code.
- Semaphores are more flexible and can also allow a number of concurrent threads access, as well as allow you to test a lock before acquiring it.

```
final  private Semaphore s= new Semaphore(1, true);

//for non-blocking version use s.acquire()
s.acquireUninterruptibly();

balance=balance+10; //protected value
s.release(); //return semaphore token
```

Scalability and Performance

- The 1.5 release promises improvements in scalability and performance with a new emphasis on startup time and memory footprint to make it easier to deploy applications running at top speed.
- One of the more significant updates is the introduction of class data sharing in the Hotspot JVM. This technology not only shares read-only data between multiple running JVMs but also improves startup time as core JVM classes are pre-packed.

Monitoring and Manageability

- The JVM Monitoring & Management API specifies a comprehensive set of JVM internals that can be monitored from a running JVM.
- One of the most useful features is a low memory detector or a memory monitor.

```
import java.lang.management.*;
import java.util.*;
import javax.management.*;

public class MemTest {
    public static void main(String args[]) {
      List pools =ManagementFactory.getMemoryPoolMBeans();
      for(ListIterator i = pools.listIterator(); i.hasNext();) {
            MemoryPoolMBean p = (MemoryPoolMBean) i.next();
            System.out.println("Memory type=" + p.getType() +
                  " Memory usage="+p.getUsage());
}}}
```

Improved Diagnostic Ability

- Generating Stack traces has been awkward if no console window has been available.
- Two new APIs, *getStackTrace* and *Thread.getAllStackTraces* provide this information programmatically.

```
StackTraceElement e[]=Thread.currentThread().getStackTrace();

for (int i=0; i <e.length; i++) {
   System.out.println(e[i]);
}

System.out.println("\n"+Thread.getAllStackTraces());
```

Other Improvements

- Network:
  - The InetAddress class now provides an API to allow testing for the reachability of a host. This features provides a ping-like capability in Java.
- Security:
  - This release of J2SE offers significant enhancements for security.
  - Improvements for scalability (SSLEngine) and performance.
- Internationalization
  - Character handling is now based on version 4.0 of the Unicode standard.
- And many improvements/bug fix in Java Sound, AWT and Swing.

Desktop Client

- The Java Desktop client remains a key component of the Java platform and as such has been the focus of many improvements in J2SE 1.5.
- This Beta release contains some of the early improvements in startup time and memory footprint.
- Not only is the release faster but the Swing toolkit enjoys a fresh new theme called Ocean.
- And by building on the updates in J2SE 1.4.2, there are further improvements in the GTK skinnable Look and Feel and the Windows XP Look and Feel.

JSR in Java 1.5

- 003 Java Management Extensions (JMX) Specification
- 013 Decimal Arithmetic Enhancement
- 014 Add Generic Types To The Java Programming Language
- 028 Java SASL Specification
- 114 JDBC Rowset Implementations
- 133 Java Memory Model and Thread Specification Revision
- 160 Java Management Extensions (JMX) Remote API 1.0
- 163 Java Platform Profiling Architecture
- 166 Concurrency Utilities
- 174 Monitoring and Management Specification for the Java Virtual Machine
- 175 A Metadata Facility for the Java Programming Language
- 200 Network Transfer Format for Java Archives
- 201 Extending the Java Programming Language with Enumerations, Autoboxing, Enhanced for Loops and Static Import
- 204 Unicode Supplementary Character Support
- 206 Java API for XML Processing (JAXP) 1.3

Summary

- The JCP controls the evolution of the Java Specification.
- Java 1.5 introduces some interesting new improvements, most of them focused at making life easier for the programmer.

Tool of the day: Eclipse

- The Eclipse Platform is designed for building integrated development environments (IDEs) that can be used to create applications as diverse as web sites, embedded Java programs, C++ programs, and Enterprise JavaBeans.
- The Eclipse Platform is an IDE for anything, and for nothing in particular.
- A plug-in is the smallest unit of Eclipse Platform function that can be developed and delivered separately.
- Everything is built to be generic, even the User Interface tools.
- Proof by demonstration : Java Development Tooling

References

- J2SE 1.5 in a Nutshell
  http://java.sun.com/developer/technicalArticles/releases/j2se15/
- Java 1.5 SDK Documentation
  http://java.sun.com/j2se/1.5.0/docs/index.html
- Sun Lights Up Java 1.5 Beta
  http://www.internetnews.com/dev-news/article.php/3309061
- Java Community Process http://www.jcp.org/en/home/index