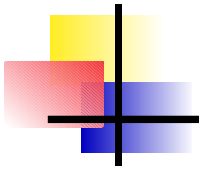
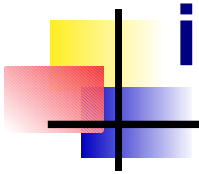


The Rest of *Tiger*

Mark Reinhold <*mr@sun.com*>
J2SE Chief Engineer
Sun Microsystems, Inc.



Much of Tiger Is Covered in Other Sessions This Week...

Annotations (JSR 175)	–	Dave Landers
Generics (JSR 14)	–	Dave Landers
BigDecimal (JSR 13)	–	Mike Cowlishaw
JAXP 1.3 (JSR 206)	–	Neil Graham
JDBC 3.0 (JSR 114)	–	Donald Smith

This talk is about the *rest* of Tiger

Monitoring & management
Observation, profiling, & debugging
Minor language enhancements
Formatting & scanning
Performance
Networking
Security
Miscellanea

Monitoring & management

Architecture

jconsole demo

All about MBeans

Observation, profiling, & debugging

Minor language enhancements

Formatting & scanning

Performance

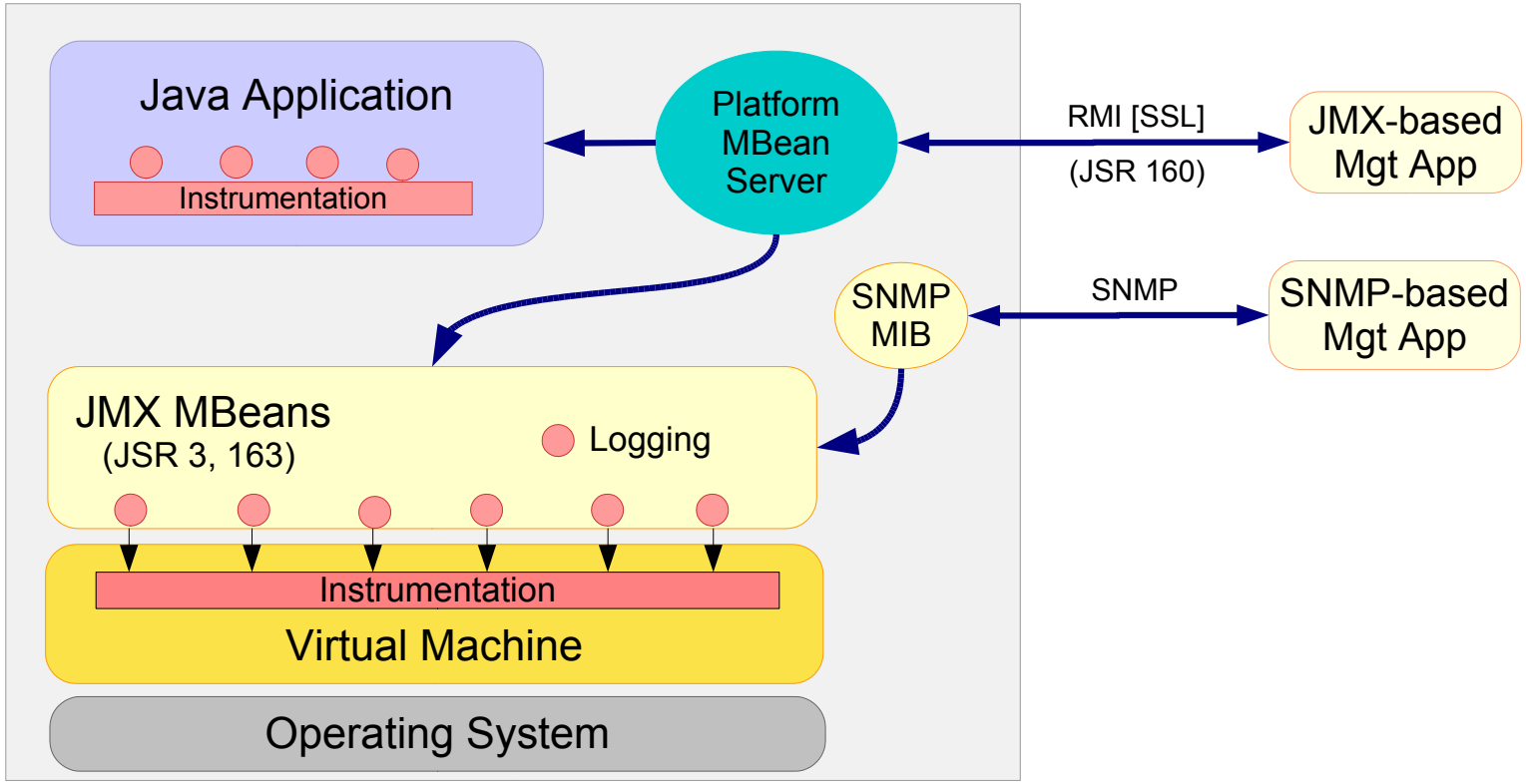
Networking

Security

Miscellanea

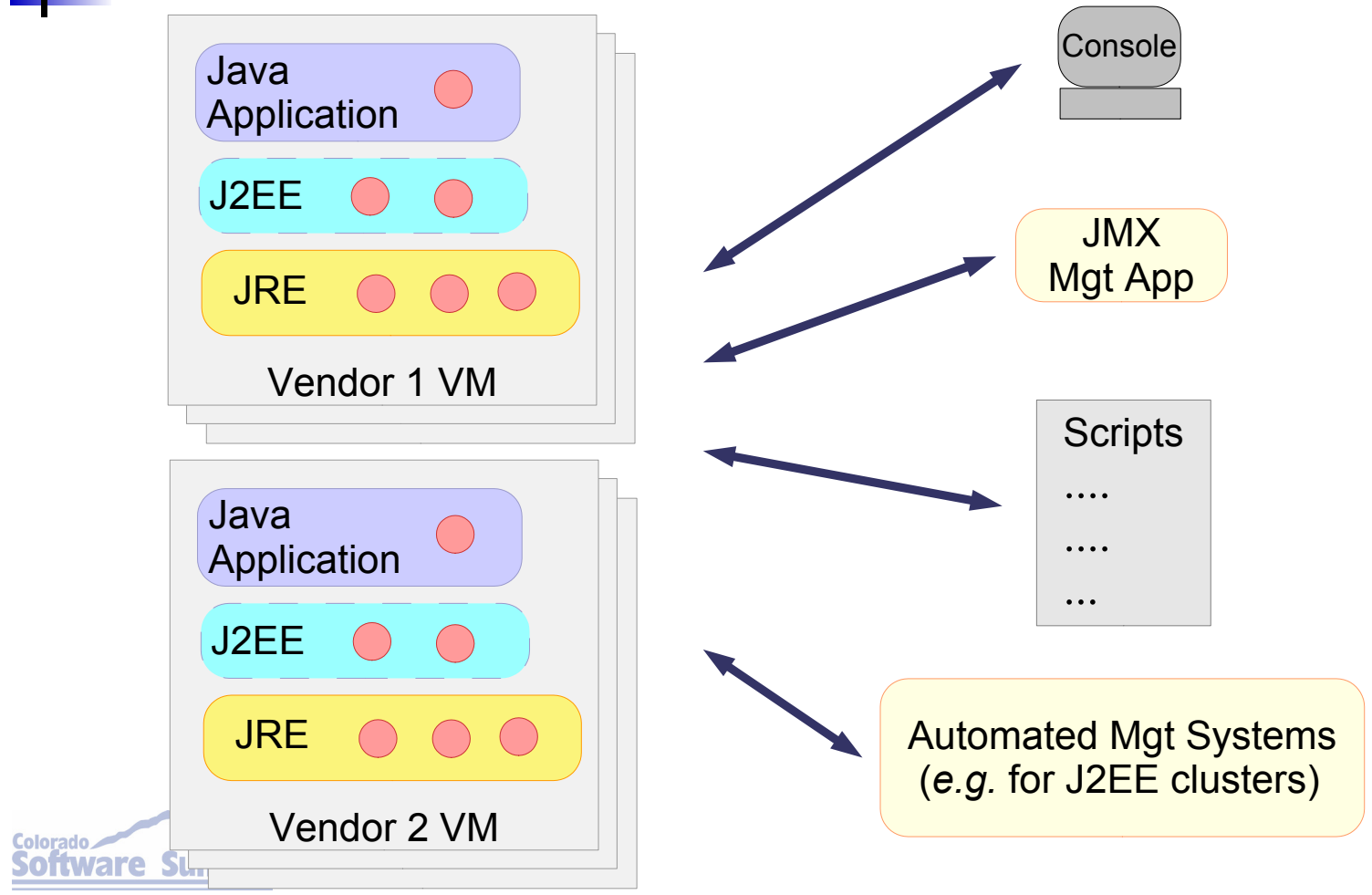
Monitoring & Management

Architecture

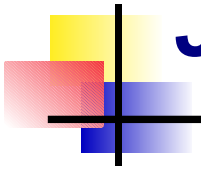


Monitoring & Management

Use cases



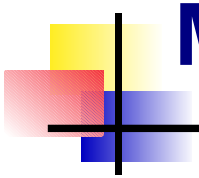
Monitoring & Management



Java Management Extensions (JMX)

- Defined by
  - JSR 3 (JMX API)
  - JSR 160 (JMX Remote API)
- Standard component of J2SE 5.0 and J2EE 1.4
- Can be used for *monitoring*...
  - *e.g.,* obtaining statistics and error notifications
- ... and *management*...
  - *e.g.,* changing configuration settings
- ... of running applications
  - From big server applications to embedded device controllers

Monitoring & Management

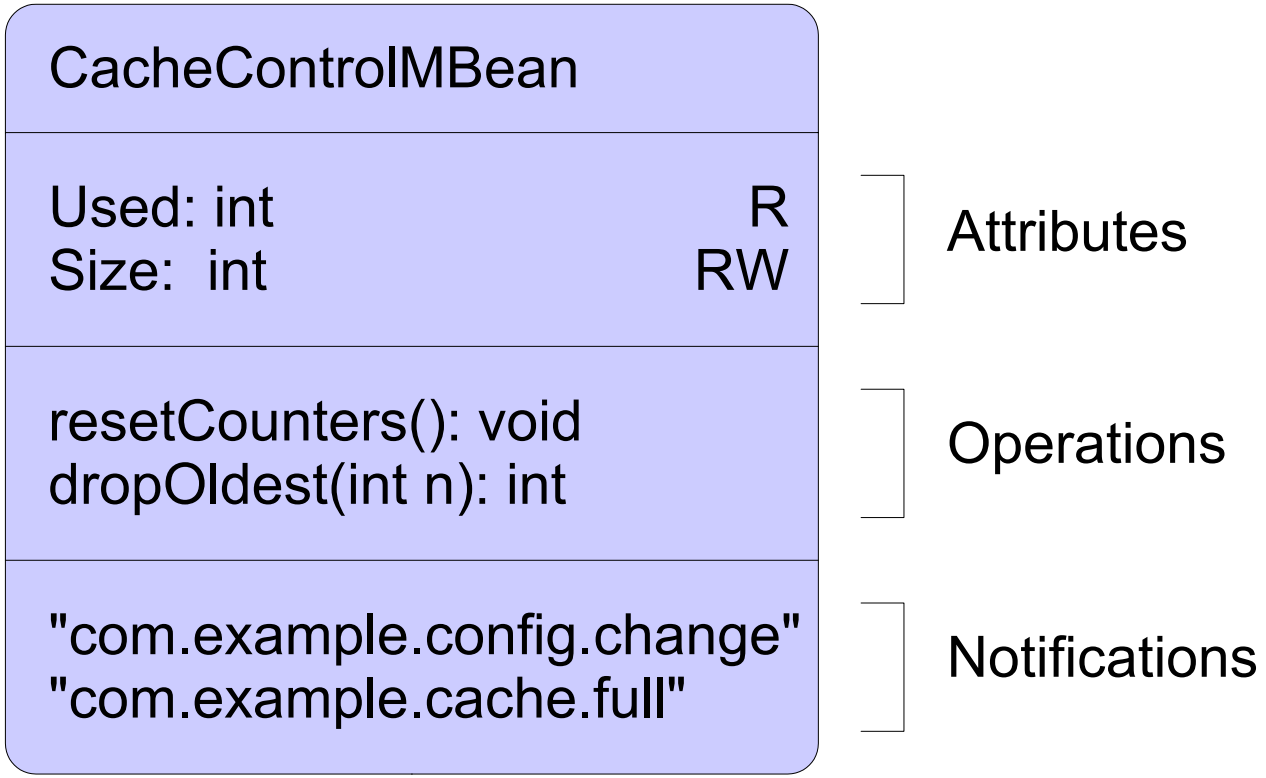


MBeans

- An MBean is a named *managed object* representing a *resource*
  - Application configuration setting
  - Program module
  - User identity
  - Device
  - *etc.*
- An MBean can have:
  - *Attributes,* which can be read and/or written
  - *Operations,* which can be invoked
  - *Notifications,* which can be sent

Monitoring & Management

MBean example



Monitoring & Management

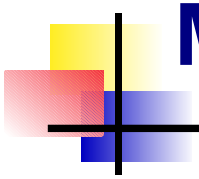
Built-in MBeans

- Runtime subsystem
  - Command-line arguments
  - Uptime, start time
  - System properties
  - Class path
  - *Boot class path*
- Class-loading subsystem
  - Class counts
    - Current
    - Loaded
    - Unloaded
  - Enable/disable verbosity
- Compilation subsystem
  - Compiler name
  - *Time spent in compilation*
- Memory subsystem
  - Length of finalization queue
  - Heap & non-heap metrics
  - Enable/disable verbosity
- Operating system
  - Name, version, architecture
  - Count of available CPUs
- Logging
  - List of logger names
  - Get/set log level of a logger
  - Find parent of a logger

(*Italics* denote optional attributes)

Monitoring & Management

More built-in MBeans

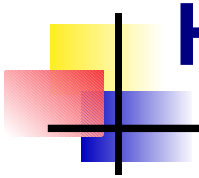


- Thread subsystem
  - Counts of
    - Started threads
    - Running threads
    - Daemon threads
  - List live threads
  - Detect deadlocks
  - *Enable/disable*
    - *Thread CPU time metrics*
    - *Contention monitoring*
- Per-thread information
  - ID & name
  - Stack trace
  - State
  - Monitor blocked on, if any
    - *Monitor owner*
  - Block/wait statistics
    - Count
    - *Time*

(jconsole demo 1)

Monitoring & Management

How to Create a Custom MBean



- Create a ***`FooMBean`*** interface
  - Defines the MBean's attributes and operations
- Create a ***`Foo`*** implementation class
- Instantiate and register your MBean with the local MBean server

Monitoring & Management

CacheControlMBean

```
public interface CacheControlMBean {

    // Read-only attributes
    public int getUsed();
    public int getHitCount();
    public int getMissCount();
    public double getMissRatio();

    // Read/write attributes
    public int getSize();
    public void setSize(int size);

    // Operations
    public void resetCounters();
    public void dropOldest(int n);

}
```

Monitoring & Management

CacheControl

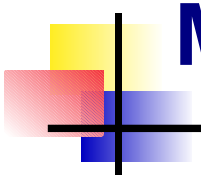
```
public class CacheControl
        implements CacheControlMBean
{
    private Cache<?> cache;

    public CacheControl(Cache<?> cache) {
        this.cache = cache;
    }

    // Attributes

    public int getSize() { return cache.getSize(); }
    public void setSize(int s) { cache.setSize(s); }
    public int getHitCount() { return cache.getHitCount
  (); }

    ...
```

Monitoring & Management

CacheControl // *(Continued)*

```
public class CacheControl
        implements CacheControlMBean
{
    ...

    // Operations

    public void resetCounters() {
        cache.resetCounters();
    }

    public void dropOldest(int n) {
        List<?> el = cache.entries();
        int s = el.size();
        el.removeAll(el.subList(0, Math.min(n, s)));
    }
}
```

Monitoring & Management



MBean server

- To be useful, an MBean must be registered in an *MBean Server*
- The only access to MBeans is through the MBean server (usually)
- You can have more than one MBean server
- Usually just use the *platform MBean server*

```
java.lang.management.ManagementFactory
    .getPlatformMBeanServer()
```

Monitoring & Management

MBean names

- Every MBean has a name
- Instance of **`javax.management.ObjectName`**
- Name = *domain* + one or more *key properties*

```
com.example:type=CacheControl
com.example:type=CacheControl,name=whatsitCache
    java.lang:type=Threading
    java.lang:type=MemoryPool,name=PS Perm Gen
```

Monitoring & Management

Registering Your Custom MBean

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

public class App {

  static void setup() {
    Cache<X> cache = new Cache<X>();
    CacheControlMBean ccmb = new CacheControl(cache);
    ObjectName name
      = new ObjectName("com.example:type=CacheControl");
    MBeanServer mbs
      = ManagementFactory.getPlatformMBeanServer();
    mbs.registerMBean(ccmb, name);
  }

  ...
}
```

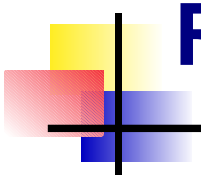
Monitoring & Management

Accessing Your Custom MBean

```
% java -Dcom.sun.management.jmxremote App &
[3] 4783
% jconsole 4783 &
%
```

(jconsole demo 2)

Monitoring & Management



Remote Access to MBeans

- Defined by the JMX Remote API (JSR 160)
- Connector types
  - RMI
  - RMI/IIOP
  - Purpose-built protocol: JMXMP
    - Fits into existing security infrastructures *via* SASL
  - Unrelated to the J2EE Connector Architecture
- Future work: SOAP-based connector

Monitoring & management

Observation, profiling, & debugging

JVMTI & `java.lang.instrument`

Troubleshooting tools

Stack traces & thread state

Minor language enhancements

Formatting & scanning

Performance

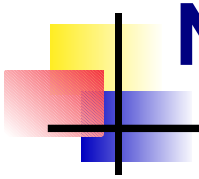
Networking

Security

Miscellanea

Observation, Profiling, & Debugging

New Profiling and Debugging APIs



- JVM Tool Interface (JVMTI)
  - Native interface – builds on JNI
  - In-process
  - Extension of JVMDI
  - Replaces JVMPI & JVMDI
- **`java.lang.instrument`**
  - Java-level interface
  - In-process
  - Based on bytecode instrumentation

These are primarily of interest to IDE and tool implementors; they will not be covered here

Observation, Profiling, & Debugging

Troubleshooting Tools

`jps`	Lists running JREs
`jstat`	Collects & logs performance stats
`jinfo`	Configuration info
`jmap`	Memory/heap maps
`jstack`	Thread-stack dump

} *Can be used locally, remotely, or on a core dump (Unix)*

All of these tools are experimental and specific to Sun's implementations

(demo)

Observation, Profiling, & Debugging: ***`java.lang.Thread`***

Stack Traces

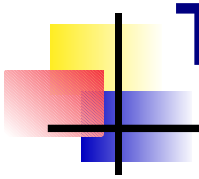
```
class Thread {

    ...

    // Returns stack trace of this thread
    StackTraceElement[] getStackTrace();

    // Returns stack traces of all threads
    Map<Thread,StackTraceElement[]>
        getAllStackTraces();

}
```

Observation, Profiling, & Debugging: ***`java.lang.Thread`***

Stack Traces: Example

```
// Find any thread whose stack trace includes
// the named class
static Thread findThread(String cn) {
    Map<Thread,StackTraceElement[]> tm
        = Thread.getAllStackTraces();
    for (Map.Entry<Thread,StackTraceElement[]> me
         : tm.entrySet())
    {
        StackTraceElement[] st = me.getValue();
        for (int i = 0; i < st.length; i++) {
            if (st[i].getClassName().equals(cn))
                return me.getKey();
        }
    }
    return null;
}
```

Observation, Profiling, & Debugging: ***`java.lang.Thread`***



Thread State

```
class Thread {

    ....

    enum State {
        // These may not correspond precisely
        // to native operating-system states
        NEW,
        RUNNABLE,
        BLOCKED,
        WAITING,
        TIMED_WAITING,
        TERMINATED;
    }

    State getState();

}
```

Observation, Profiling, & Debugging: ***`java.lang.Thread`***

Thread State: Example

```
// Find all blocked threads
static List<Thread> findBlockedThreads() {
    Map<Thread,StackTraceElement[]> tm
        = Thread.getAllStackTraces();
    List<Thread> bts = new ArrayList<Thread>();
    for (Thread t : tm.keySet()) {
        if (t.getState() == Thread.State.BLOCKED)
            bts.add(t);
    }
    return bts;
}
```

Observation, Profiling, & Debugging: ***`java.lang.Thread`***

Uncaught-exception Handlers

```
class Thread {

    ...

    interface UncaughtExceptionHandler {          // ==
UEH
        void uncaughtException(Thread t, Throwable x);
    }

    void setUncaughtExceptionHandler(UEH h);
    UEH getUncaughtExceptionHandler();

    static void setDefaultUncaughtExceptionHandler(UEH
h);
    static UEH getDefaultUncaughtExceptionHandler();

    // Previously you could only catch uncaught
exceptions
    // by mucking around with thread groups (ugh)
}
```

Observation, Profiling, & Debugging: ***`java.lang.Thread`***

Uncaught-exception Handlers

```
Thread t = new Thread() {
    public void run() {
      throw new RuntimeException("Foo!");
    }
});

t.setUncaughtExceptionHandler(new Thread.UEH() {
    public void uncaughtException(Thread t, Throwable x)
{
      System.out.format("%s: %s%n", t, x);
    }
});

t.start();


% java Test
Thread[Thread-0,5,main]: java.lang.RuntimeException: Foo!
%
```

Monitoring & management

Observation, profiling, & debugging

Minor language enhancements

Enhanced `for` loops

Enumerated types

Autoboxing

Varargs

Formatting & scanning

Performance

Networking

Security

Miscellanea

Minor Language Enhancements

Enhanced `for` Loops

```
Collection<String> c = ... ;

for (Iterator<String> i = c.iterator(); i.hasNext();)
{
    String s = i.next();
    ...
}
```

Minor Language Enhancements

Enhanced `for` Loops

```
Collection<String> c = ... ;

for (Iterator<String> i = c.iterator(); i.hasNext();)
{
    String s = i.next();
    ...
}
```

```
for (String s : c) {
    ...
}
```

Minor Language Enhancements

Enhanced `for` Loops

```
interface Collection<E> {
    Iterator<E> iterator();
    boolean add(E o);
    boolean remove(Object o);
    boolean containsAll(Collection<?> c);
    boolean addAll(Collection<? extends E> c);
}
```

Minor Language Enhancements

Enhanced `for` Loops

```
interface Collection<E>
    implements Iterable<E>
{
    Iterator<E> iterator();
    boolean add(E o);
    boolean remove(Object o);
    boolean containsAll(Collection<?> c);
    boolean addAll(Collection<? extends E> c);
}

interface Iterable<E> {
    Iterator<E> iterator();
}
```

Minor Language Enhancements

Enhanced `for` Loops

```
int sum(int[] a) {
    int s = 0;
    for (int i = 0; i < a.length; i++)
        s += a[i];
    return s;
}
```

Minor Language Enhancements

Enhanced `for` Loops

```
int sum(int[] a) {
    int s = 0;
    for (int i = 0; i < a.length; i++)
        s += a[i];
    return s;
}
```

```
int sum(int[] a) {
    int s = 0;
    for (int v : a)
        s += v;
    return s;
}
```

Minor Language Enhancements

Enumerated Types

```
public class Toss {
    private String name;
    private Toss(String n) { name = n; }
    public static final Toss HEADS = new Toss("HEADS");
    public static final Toss TAILS = new Toss("TAILS");
    public String toString() { return name; }
    public static Toss valueOf(String s) {
        if (s.equals("HEADS")) return HEADS;
        if (s.equals("TAILS")) return TAILS;
        throw new IllegalArgumentException(s);
    }
}
```

Minor Language Enhancements

Enumerated Types

```
public class Toss {
    private String name;
    private Toss(String n) { name = n; }
    public static final Toss HEADS = new Toss("HEADS");
    public static final Toss TAILS = new Toss("TAILS");
    public String toString() { return name; }
    public Toss valueOf(String s) {
        if (s.equals("HEADS")) return HEADS;
        if (s.equals("TAILS")) return TAILS;
        throw new IllegalArgumentException(s);
    }
}
```

```
public enum Toss { HEADS, TAILS }
```

Minor Language Enhancements

Enumerated Types

```
void foo(Toss t) {
    switch (t) {
        case HEADS:
            doHeads();
            break;
        case TAILS:
            doTails();
            break;
        default:
            assert false;  // Can never happen
    }
}
```

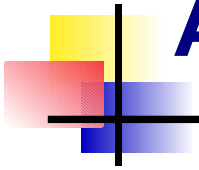
Minor Language Enhancements

Enumerated Types

```
for (Toss t : Toss.values()) {
    System.out.println(t);
}
```

Minor Language Enhancements

Autoboxing



```
Integer x = new Integer(3);
int y = x.intValue();
map.put(new Integer(1),
        new Integer(42));
```

Minor Language Enhancements

Autoboxing

```
Integer x = new Integer(3);
int y = x.intValue();
map.put(new Integer(1),
        new Integer(42));
```

→

```
Integer x = 3;
int y = x;
map.put(1, 42);
```

Minor Language Enhancements

Autoboxing

```
Integer x = new Integer(3);   ⇔  Integer x = 3;
int y = x.intValue();         ⇔  int y = x;
map.put(new Integer(1),       ⇔  map.put(1, 42);
        new Integer(42));
```

```
public static void test() {
    Pair<Integer,String> p
      = new Pair<Integer,String>(new Integer(1), "one");
    int i = p.left().intValue();
    String n = p.right();
}
```

Minor Language Enhancements

Autoboxing

```
Integer x = new Integer(3);   ⇒  Integer x = 3;
int y = x.intValue();         ⇒  int y = x;
map.put(new Integer(1),       ⇒  map.put(1, 42);
        new Integer(42));
```

```
public static void test() {
    Pair<Integer,String> p
      = new Pair<Integer,String>(new Integer(1)1, "one");
    int i = p.left().intValue();
    String n = p.right();
}
```

Minor Language Enhancements

Varargs

```
public class PrintStream {

    public void printf(String fmt, Object[] args) {
        ...
    }

}
```

Minor Language Enhancements

Varargs

```
public class PrintStream {

    public void printf(String fmt, Object[] args) {
        ...
    }

}
```

```
out.printf("%4d/%08X %tY-%<tm-%<tD %s%n",
           new Object[] { new Integer(bits),
                          new Integer(fingerprint),
                          date, user
                        });
```

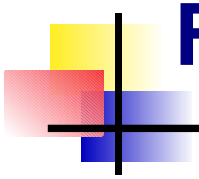
Minor Language Enhancements

Varargs

```
public class PrintStream {

    public void printf(String fmt, Object... args) {
        ...
    }

}
```

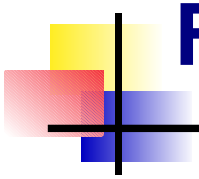
```
out.printf("%4d/%08X %tY-%<tm-%<tD %s%n",
           bits, fingerprint, date, user);
```

Monitoring & management
Observation, profiling, & debugging
Minor language enhancements
Formatting & scanning
Performance
Networking
Security
Miscellanea



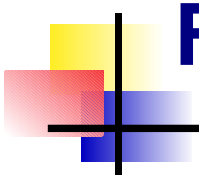
Formatting & Scanning

```
BufferedReader br
    = new BufferedReader(
          new InputStreamReader(System.in));
System.out.print("Farenheit: ");
String ln = br.readLine();
StringTokenizer st = new StringTokenizer(ln);
double f = Double.parseDouble(st.nextToken());
System.out.println("Celsius: " + ((f - 32) / 1.8));
```



Formatting & Scanning

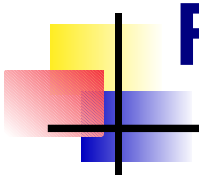
```
BufferedReader br
    = new BufferedReader(
          new InputStreamReader(System.in));
System.out.print("Farenheit: ");
String ln = br.readLine();
double f = Double.parseDouble(ln.split("\\s+")[0]);
System.out.println("Celsius: " + ((f - 32) / 1.8));
```



Formatting & Scanning

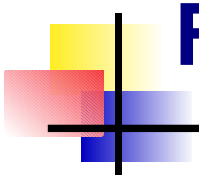
```
BufferedReader br
    = new BufferedReader(
          new InputStreamReader(System.in));
System.out.print("Farenheit: ");
String ln = br.readLine();
double f = Double.parseDouble(ln.split("\\s+")[0]);
System.out.println("Celsius: " + ((f - 32) / 1.8));
```

```
% java Conv
Farenheit: 32.1
Celsius: 0.05555555555556344
%
```



Formatting & Scanning

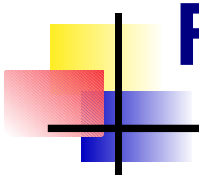
```
BufferedReader br = new BufferedReader(...);
System.out.print("Farenheit: ");
String ln = br.readLine();
double f = Double.parseDouble(ln.split("\\s+")[0]);
MessageFormat mf
    = new MessageFormat("Celsius: {0,number,.##}");
System.out.println(mf.format(new Object[] {
                                 new Double((f - 32)
                                            / 1.8)
                             }));

% java Conv
Farenheit: 32.1
Celsius: 0.06
%
```



Formatting & Scanning

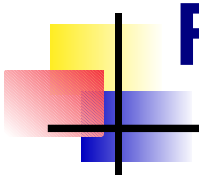
```
System.out.print("Farenheit: ");
Scanner sc = new Scanner(System.in);
double f = sc.nextDouble();
out.printf("Celsius: %.2f%n", (f - 32) / 1.8);
```

```
% java Conv
Farenheit: 32.1
Celsius: .06
%
```



Formatting & Scanning

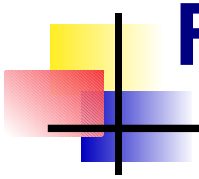
```
% java ShowKey mr
1024/0E9F1EA4 2001-10-12 mr@sun.com
%
```



Formatting & Scanning

```
% java ShowKey mr
1024/0E9F1EA4 2001-10-12 mr@sun.com
%
```

```
SimpleDateFormat df
    = new SimpleDateFormat("yyyy-MM-dd");
out.println(bits + "/"
            + Integer.toHexString(fingerprint)
                     .toUpperCase()
            + " " + df.format(date) + " " + user);
```



Formatting & Scanning

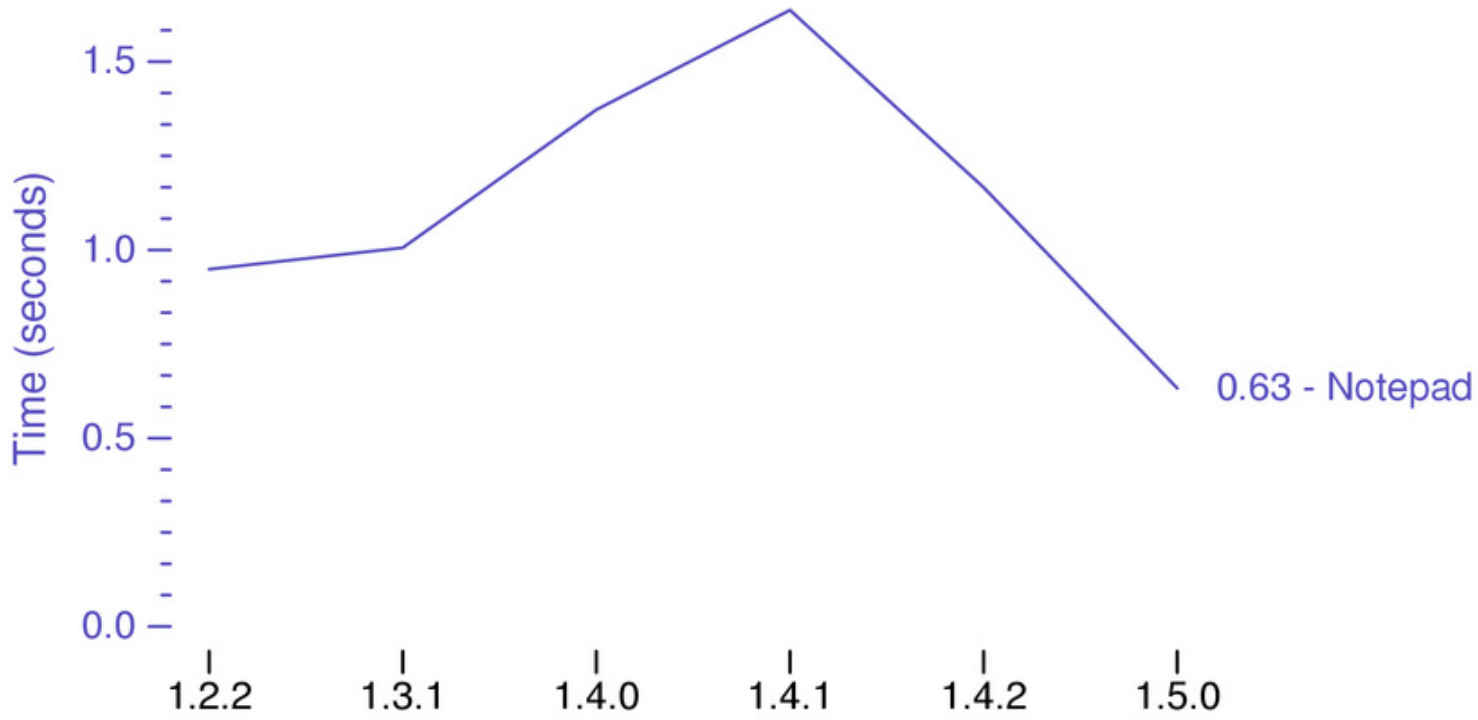
```
% java ShowKey mr
1024/0E9F1EA4 2001-10-12 mr@sun.com
%
```

```
SimpleDateFormat df
    = new SimpleDateFormat("yyyy-MM-dd");
out.println(bits + "/"
            + Integer.toHexString(fingerprint)
                     .toUpperCase()
            + " " + df.format(date) + " " + user);

out.printf("%4d/%08X %tY-%<tm-%<tD %s%n",
           bits, fingerprint, date, user);
```

Monitoring & management
Observation, profiling, & debugging
Minor language enhancements
Formatting & scanning
Performance
Startup time
Ergonomics
Networking
Security
Miscellanea

Performance

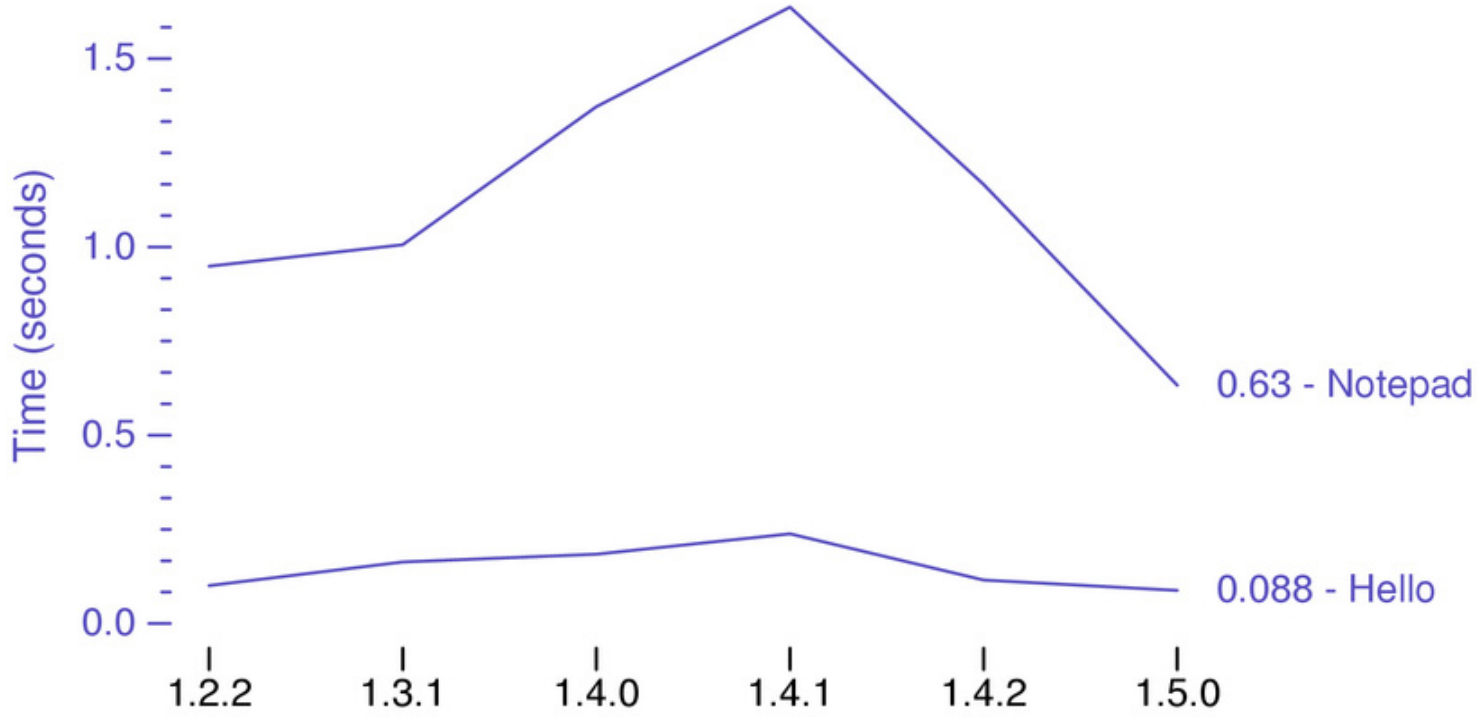
Startup Time



1.5GHz Athlon, 512MB, Linux 2.4.20

Performance

Startup Time



1.5GHz Athlon, 512MB, Linux 2.4.20

Performance

Ergonomics

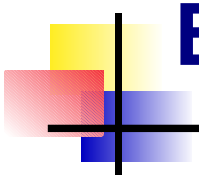
```
% # Uses client compiler, starts w/ small heap
% java BigFatApp
```

Performance

Ergonomics

```
% # Uses client compiler, starts w/ small heap
% java BigFatApp

% # More realistically...
% java -server -XX:UseParallelGC -Xmx1G BigFatApp
```

Performance

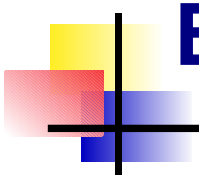


Ergonomics

```
% # Uses client compiler, starts w/ small heap
% java BigFatApp

% # More realistically...
% java -server -XX:UseParallelGC -Xmx1G BigFatApp

% # With Tiger, on a server-class machine,
% # you can just say
% java BigFatApp
```

Performance

Ergonomics



- What is a server-class machine?
  - At least 2 CPUs
  - At least 2GB of memory
  - Not running Windows
    - For complicated reasons
- Effective flags on a server-class machine
  - **`-server`**
  - **`-XX:+UseParallelGC`**
  - **`-Xms`***max(1/64 physical memory, default)*
  - **`-Xmx`***min(1/4 physical memory, 1GB)*

Performance

Ergonomics

```
% # Goal-oriented GC tuning
% java -XX:UseParallelGC \
       -XX:MaxGCPauseMillis=ms \
       -XX:GCTimeRatio=n \
       Foo
```

Monitoring & management
Observation, profiling, & debugging
Minor language enhancements
Formatting & scanning
Performance
Networking
HTTP enhancements
Features for containers
Inherited channels
Reachability
Security
Miscellanea

Networking

`URLConnection` timeouts

```
class URLConnection { ...

    void setConnectTimeout(int
millis);
    int getConnectTimeout();

    void setReadTimeout(int millis);
    int getReadTimeout();

}
```

- Causes **`SocketTimeoutException`** to be thrown by connect or read operations
- Works for both HTTP & FTP connections

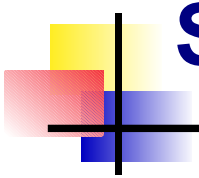
Networking

Streaming HTTP Output

```
class HttpURLConnection { ...

  void setFixedLengthStreamingMode(int
length);
  void setChunkedStreamingMode(int chunkSize);

}
```

- Allows application to stream large POST data without buffering
- Trade-off: Redirection and authentication not automatic when streaming
- New exception: **`HttpRetryException`**

Networking



Streaming HTTP Output: Example

```
// Streaming a known amount of data
HttpURLConnection uc = url.openConnection();
uc.setRequestMetod("POST");
uc.setDoOutput(true);
File f = new File("file2send.txt");
uc.setFixedLengthStreamingMode(f.length());
InputStream is = new FileInputStream(f);
OutputStream os = uc.getOutputStream();
byte[] buffer = new byte[10 * 1024];
while ((n = is.read(buffer)) > 0)
    os.write(buffer, 0, n);
os.close();
try {
    int response = uc.getResponseCode();
} catch (HttpRetryException x) {
    ...
}
```

Networking

Streaming HTTP Output: Example

```
// Streaming an unknown amount of data
InputStream is = ... ;
HttpURLConnection uc = url.openConnection();
uc.setRequestMetod("POST");
uc.setDoOutput(true);
uc.setChunkedStreamingMode(0); // 0 => default size
OutputStream os = uc.getOutputStream();
byte[] buffer = new byte[10 * 1024];
while ((n = is.read(buffer)) > 0)
   os.write(buffer, 0, n);
os.close();
is.close();
try {
    int response = uc.getResponseCode();
} catch (HttpRetryException x) {
    ...
}
```

Networking

Proxy Specification

```
class Proxy {

    enum Type { DIRECT, HTTP, SOCKS };

    static final Proxy NO_PROXY;

    Proxy(Type t, SocketAddress sa);

    Type type();
    SocketAddress address();

}
```

Networking

Proxy Selection

```
class ProxySelector {

    // Retrieve a list of proxies for the given URI
    List<Proxy> select(URI uri);

    // Report that an attempt to connect to the
    // given URI via the given proxy socket failed,
    // allowing the implementation to adjust
    void connectFailed(URI uri,
                       SocketAddress sa,
                       IOException x);

    // The system-wide default proxy selector
    static ProxySelector getDefault();
    static void setDefault(ProxySelector ps);
}
```

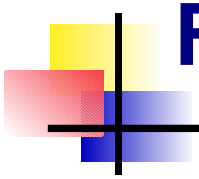
Networking

Proxy Selection

```
class Socket {

    // The existing constructors now use
    // the system-wide proxy selector, if any
    Socket();
    Socket(String host, int port);
    Socket(InetAddress ia, int port);
    ...

    // A proxy can also be specified explicitly
    Socket(Proxy p);

}
```

Networking

Features for Containers



- HTTP cookie management
  - Container can intercept and manipulate cookies
  - `java.net.CookieHandler`
- `URLConnection` content caching
  - Container can interpose a system-wide cache underneath existing APIs
  - `java.net.ResponseCache`

Networking

Inherited Channels

```
% grep foo /etc/inetd.conf
foo  stream  tcp  nowait  nobody \
  /usr/j2se/bin/java java -cp /opt/Foo FooServer
%
```

Networking

Inherited Channels

```
% grep foo /etc/inetd.conf
foo  stream  tcp  nowait  nobody \
  /usr/j2se/bin/java java -cp /opt/Foo FooServer
%
```

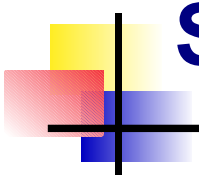
```
public static void main(String[] args) {
    SocketChannel sc
        = (SocketChannel)System.inheritedChannel();
    handleRequest(sc);
}
```

Networking

Reachability ("ping") API

```
class InetAddress {

    ...

    // Uses ICMP ECHO requests if possible,
    // otherwise tries TCP port 7 ("echo")

    boolean isReachable(int timeoutInMillis);

    boolean isReachable(NetworkInterface if,
                        int timeToLive,       // hop count
                        int timeoutInMillis);

}
```

Monitoring & management
Observation, profiling, & debugging
Minor language enhancements
Formatting & scanning
Performance
Networking
Security
Cryptography
Secure sockets
Public-key infrastructure
Miscellanea



Security

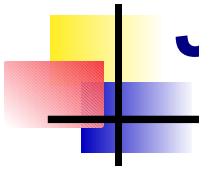
- Cryptography (JCA/JCE)
  - PKCS#11 support for smart cards & accelerators
- Secure sockets (JSSE)
  - SSLEngine for non-blocking NIO SSL
- Public-key infrastructure
- JGSS & Kerberos
- SASL
- Timestamped code signing

Security

Cryptography (JCA/JCE)

- RSA encryption
- RC2, ARCFOUR algorithms
- API ready for Elliptic Curve Cryptography
- Supports features required for XML encryption and XML signature
- NIO integration (`ByteBuffer`s)
- Provider for PKCS#11...

Security



JCE provider for PKCS#11

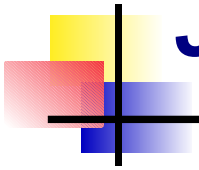
- PKCS#11: Standard 'C' cryptography API
  - Smart cards and USB security tokens
  - Hardware accelerator PCI tokens
  - Software tokens
  - Widely available: Multiple platforms, multiple vendors
- JCE can creates and optionally protect keys
  - Available through KeyStore API as `"PKCS11"`
- Used by JSSE through JCE and KeyStore

Security

Secure sockets (JSSE)

- Full provider pluggability
  - With limitations on supported cipher suites
- PKIX CertPath trust manager
  - Was option in 1.4.2, now the default
- Kerberos cipher suites
  - If supported by underlying OS
- PKCS#11 tokens *via* JCE
- SSLEngine...

Security



JSSE: SSLEngine

- Highly scalable SSL/TLS API
  - Non-blocking design
  - Accommodates different threading models
- Separates SSL/TLS functionality from I/O
- Enables arbitrary transport layers
  - Serial ports
  - Pipes
  - Selectable SocketChannels
  - ...

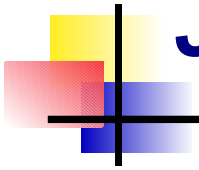
Security

Public Key Infrastructure

- Online Certificate Status Protocol (OCSP)
  - Authority Information Access X.509 Extension
- Minor enhancements to CertPath API
- Passed RFC 3280 compliance test
- JSSE CertPath-based TrustManager
- Enhanced PKCS#12 support (KeyStore)
  - Added ability to write in addition to reading

Security

JGSS & Kerberos



JGSS (Generic Security Services)

- Additional encryption cipher types
  - Triple DES
  - AES coming soon
- Support for IPv6

Kerberos

- TLS cipher suites
- More deployment options
  - TGT renewals
  - TCP *vs*. UDP preference configuration

Security

SASL

SASL = Simple Authentication & Security Layer

- Provides pluggable authentication to network protocols (RFC 2222)
- Standard API and framework (JSR 28)
- Pluggable providers
- Default provider supports client/server:
  - Digest-MD5,
  - CRAM-MD5, and
  - GSS-API/Kerberos mechanisms

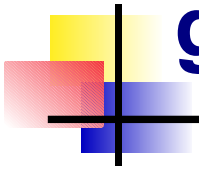
Security

Timestamped Code Signing

- IETF RFC 3161
  - Standard protocol for timestamp requests
- **`jarsigner`**
  - New timestamp command line options
- JRE
  - Parses timestamps in signed JARs
  - Enhanced JAR API for signing information
- Plugin and Java Web Start
  - Can trust signer with expired X.509 certificate

Monitoring & management
Observation, profiling, & debugging
Minor language enhancements
Formatting & scanning
Performance
Networking
Security
Miscellanea
getenv
Multiple JRE support

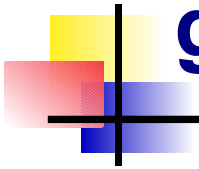
Miscellanea



getenv

```
out.println(System.getenv("PATH"));
```

Miscellanea



getenv

```
out.println(System.getenv("PATH"));
```

```
for (Map.Entry<String,String> me : System.getenv())
    out.printf("%s=%s%n",
               me.getKey(), me.getValue());
```

Miscellanea

Multiple JRE Support

```
% /usr/local/java/1.4.1/bin/java Foo
```

Miscellanea

Multiple JRE Support

```
% /usr/local/java/1.4.1/bin/java Foo
```

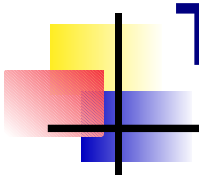
```
% java -version:1.4.1+ Foo
```

Miscellanea

Multiple JRE Support

```
% /usr/local/java/1.4.1/bin/java Foo

% java -version:1.4.1+ Foo

% cat >manifest
Main-Class: Foo
JRE-Version: 1.4.1+
^D
% jar cfm manifest foo.jar Foo.class
% java -jar foo.jar
```

Monitoring & management
Observation, profiling, & debugging
Minor language enhancements
Formatting & scanning
Performance
Networking
Security
Miscellanea



Thanks...

Alan Bateman
Mandy Chung
Robert Field
Jim Holmlund
Michael McMahon
Éamonn McManus
Jeff Nisewanger
Sanjay Radia

Mark Reinhold
mr@sun.com