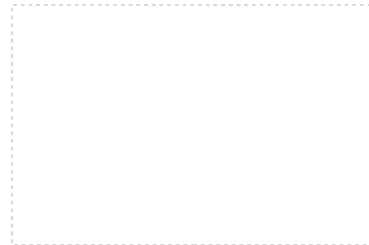


# Salt Lake City JUG

## Java in Tiger and Beyond

Chet Haase  
Senior Staff Engineer  
Java2D  
September 16, 2004



# Rough Agenda

- Tiger Features and Improvements
- Post-Tiger Thoughts
- Java Graphics
- Q&A

# Tiger Features & Improvements

- Tiger's here! (almost...)
- Ease of Development Features
  - Annotations (“metadata”)
  - Generics
  - for() loop enhancements
  - enum
  - Autoboxing
  - printf!
  - varargs
  - concurrency utilities
- Performance

# Ease of Development: Annotations

```
// Part 2: What does this produce?

class Parent {
    public void blah() {
        System.out.println("Parent");
    }
}

public class Child extends Parent
{
    public void blahh() {
        System.out.println("Child.blah");
    }
    public static void main(String args[]) {
        (new Child()).blah();
    }
}
```

# Ease of Development: **Annotations**

```
% javac Child.java  
% java Child  
Parent
```

# Ease of Development: **Annotations**

```
// Part 2: How about this version?

class Parent {
    public void blah() {
        System.out.println("Parent");
    }
}

public class Child extends Parent
{
    @Override
    public void blahh() {
        System.out.println("Child.blah");
    }
    public static void main(String args[]) {
        (new Child()).blah();
    }
}
```

# Ease of Development: **Annotations**

```
% javac Child.java
% Child.java:9: method does not override a
method from its superclass
    @Override
     ^
```

# Ease of Development: Generics

```
// This HashMap maps Strings to Mammals
HashMap m = new HashMap();
m.put("wombat", new Mammal());
Mammal w = (Mammal)m.get("wombat");
```

# Ease of Development: Generics

```
// This HashMap maps Strings to Mammals  
HashMap m = new HashMap();  
m.put("wombat", new Mammal());  
Mammal w = (Mammal)m.get("wombat");
```

```
m.put("gecko", new Lizard());  
Mammal x = (Mammal)m.get("gecko");
```

**Runtime error: ClassCastException**

# Ease of Development: Generics

```
// This HashMap maps Strings to Mammals
HashMap<String,Mammal> m
    = new HashMap<String,Mammal>();
m.put("wombat", new Mammal());
Mammal w = (Mammal)m.get("wombat");

m.put("gecko", new Lizard());
```

**Compile-time error: Type mismatch**

# Ease of Development: **for()** loop enhancements

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

# Ease of Development: **for()** loop enhancements

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

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

# Ease of Development: **enum**

```
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; }
}
```

# Ease of Development: **enum**

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

# Ease of Development: **Autoboxing**

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

# Ease of Development: **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);

# Ease of Development: **formatting and scanning**

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

# Ease of Development: **formatting and scanning**

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

# Ease of Development: **formatting and scanning**

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

```
% java Conv
Fahrenheit: 32.1
Celsius: 0.05555555555555556344
%
```

# Ease of Development: **formatting and scanning**

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

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

# Ease of Development: **formatting and scanning (printf!)**

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

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

# Ease of Development: **varargs**

```
public class PrintStream {  
    public void printf(String fmt, Object[] args);  
}
```

# Ease of Development: **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  
});
```

# Ease of Development: **varargs**

```
public class PrintStream {  
    public void printf(String fmt, Object...  
args);  
}
```

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

# Ease of Development: **concurrency utilities**

```
ServerSocketChannel ssc
    = ServerSocketChannel.open();

...

for (;;) {
    SocketChannel sc = ssc.accept();
    new Thread(new Handler(sc)).start();
}
```

# Ease of Development: **concurrency utilities**

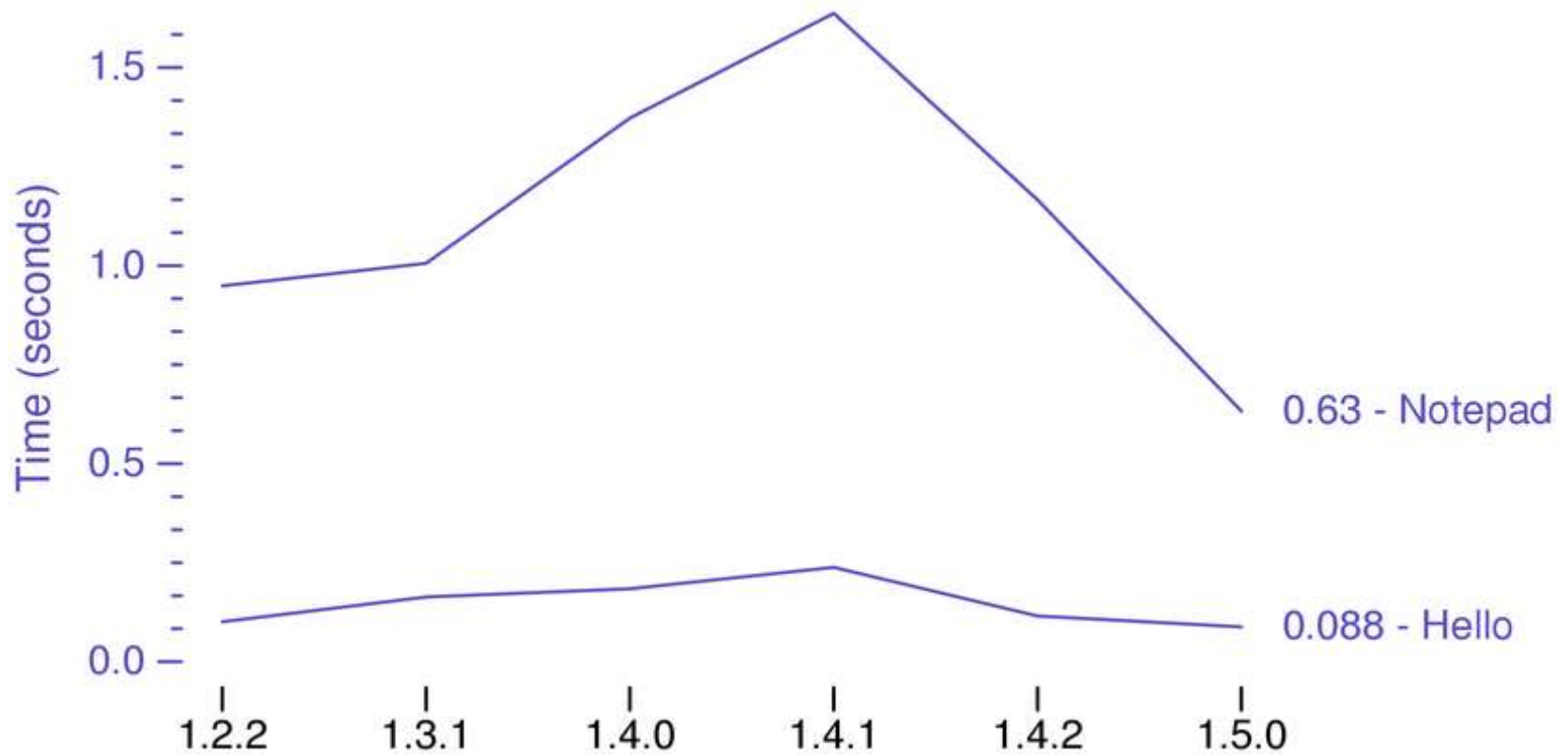
```
import java.util.concurrent.*;

ServerSocketChannel ssc
    = ServerSocketChannel.open();

Executor xec = Executors.newCachedThreadPool();

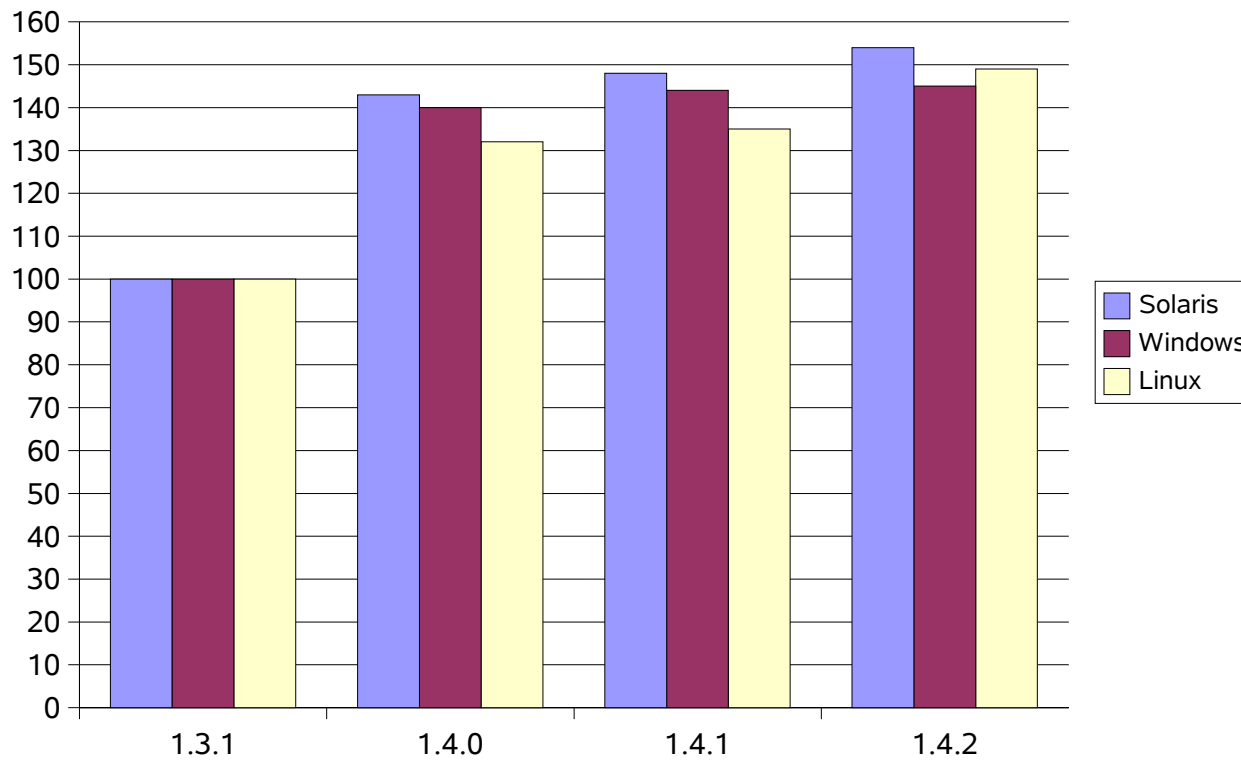
for (;;) {
    SocketChannel sc = ssc.accept();
    xec.execute(new Handler(sc));
}
```

# Tiger Performance: **startup**



1.5GHz Athlon, 512MB, Linux 2.4.20

# Tiger Performance: runtime



SwingMark (bigger is better)

# Tiger Performance: **garbage collection**

- New collectors (serial collector still default):
  - Parallel collector for higher throughput:  
`% java -XX:+UseParallelGC`
  - Concurrent collector for low pauses:  
`% java -XX:+UseConcMarkSweepGC -XX:+UseParNewGC`
- Ergonomics:
  - Instead of:  
`java -server -XX:UseParallelGC -Xmx1G BigFatApp`
    - the VM does the right thing with this:  
`java BigFatApp`
  - ***Note: only on “server” class machines or by using “-server” VM parameter***

For more information:

<http://java.sun.com/docs/hotspot/gc1.4.2/>

# Post-Tiger: **Timeframes**

- 1.4.0      Merlin      2002/2/13
- 1.4.1      Hopper      2002/10/16
- 1.4.2      Mantis      2003/5/29
- **1.5.0**      **Tiger**      **2004/9/30**
- 1.5.1      Dragonfly      2005/Q1
- 1.6.0      Mustang      2006/Q1
- 1.7.0      Dolphin      2007/Q3

# Potential Post-Tiger Themes

Compatibility & Stability  
Diagnosability, Monitoring, Management  
XML & Web Services  
Ease of Development  
Performance  
Enterprise Desktop

# Potential Post-Tiger Themes

## Compatibility & Stability

Diagnosability, Monitoring, Management

XML & Web Services

Ease of Development

Performance

Enterprise Desktop

# Potential Post-Tiger Themes

Compatibility & Stability

Diagnosability, Monitoring, Management

Troubleshooting guides

Annotated MBeans

JMX/SOAP

Tracing

XML & Web Services

Ease of Development

Performance

Enterprise Desktop

# Potential Post-Tiger Themes

Compatibility & Stability

Diagnosability, Monitoring, Management

**XML & Web Services**

JAX-RPC client & lightweight server

XML Encryption & Signature

JAXB (XML data binding)

JAXP.next

Ease of Development

Performance

Enterprise Desktop

# Potential Post-Tiger Themes

Compatibility & Stability

Diagnosability, Monitoring, Management

XML & Web Services

**Ease of Development**

Alternative languages (e.g., Groovy)

New language features (7.0)

Simplified platform view

JDBC 4.0

Performance

Enterprise Desktop

# Potential Post-Tiger Themes

Compatibility & Stability

Diagnosability, Monitoring, Management

XML & Web Services

Ease of Development

**Performance**

Split verifier

Startup & footprint

Still better ergonomics

Improved class-file/jar-file format

Enterprise Desktop

# Potential Post-Tiger Themes

Compatibility & Stability

Diagnosability, Monitoring, Management

XML & Web Services

Ease of Development

Performance

**Enterprise Desktop**

Longhorn & GNOME integration

Simplified GUI construction

JDIC/JDNC

# Potential Post-Tiger Themes

Compatibility & Stability  
Diagnosability, Monitoring, Management  
XML & Web Services  
Ease of Development  
Performance  
Enterprise Desktop  
?

# Java Graphics

- Tiger Improvements
- Tips & Tricks
- Post-Tiger Plans

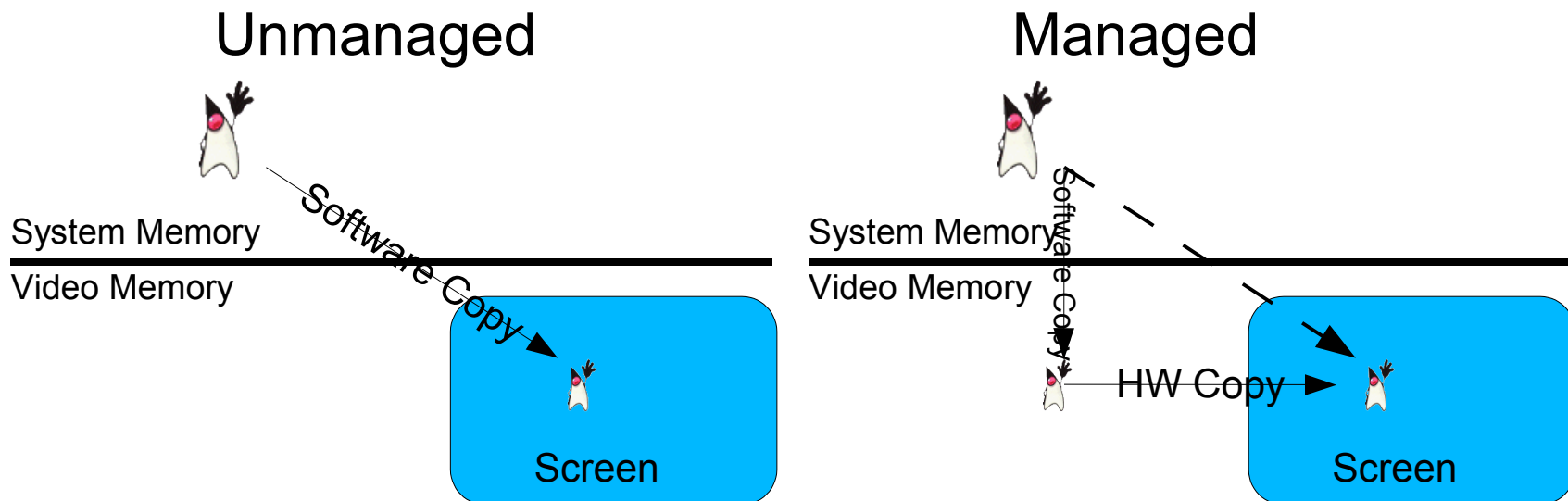
# Tiger Improvements

- Managed Images Everywhere!
- OpenGL acceleration
- Font rearchitecture
- CUPS (Unix printing)
- ImageIO plugins (BMP, WBMP); +performance
- Startup performance
- Misc. Performance Improvements
  - Primitive batching on Windows
  - Complex clip rendering
  - Footprint reduction for Toolkit images

# Graphics Tips & Tricks:

## Managed Images

- Images will get copy-from acceleration under the hood, as appropriate and as possible
- No need to use `VolatileImage` unless you are writing *to* the image frequently
- Avoid grabbing the `DataBuffer` on the object; this causes us to punt on acceleration



# Graphics Tips & Tricks:

## Intermediate Images

- Cache complex/slow rendering in intermediate images
- use simple `drawImage()` from intermediate image

- Instead of:

```
g.drawImage(img, x, y, w, h, null);
```

- Do:

```
if (imImage == null) {  
    imImage = createImage(w, h);  
    Graphics imgG = imImage.getGraphics();  
    imgG.drawImage(img, 0, 0, w, h, null);  
}  
g.drawImage(imImage, x, y, null);
```

<http://java.sun.com/developer/technicalArticles/Media/intimages/>

# Graphics Tips & Tricks:

## Primitive Batching

- Context Switching (Direct3D, DirectDraw, GDI, Software) can be expensive, so:

- Do:

```
for (int i = 0; i < numPrims; ++i)
    g.drawLine(line[i].x0, line[i].y0,
               line[i].x1, line[i].y1);
for (int i = 0; i < numPrims; ++i)
    g.drawString(text[i].string,
                 text[i].x0, text[i].y0);
```

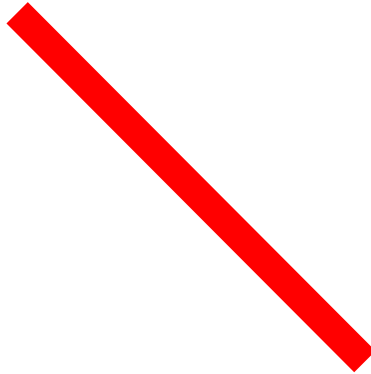
- Not:

```
for (int i = 0; i < numPrims; ++i) {
    g.drawLine(line[i].x0, line[i].y0,
               line[i].x1, line[i].y1);
    g.drawString(text[i].string,
                 text[i].x0, text[i].y0);
}
```

# Graphics Tips & Tricks:

## Draw what you mean

- For:



- Do:

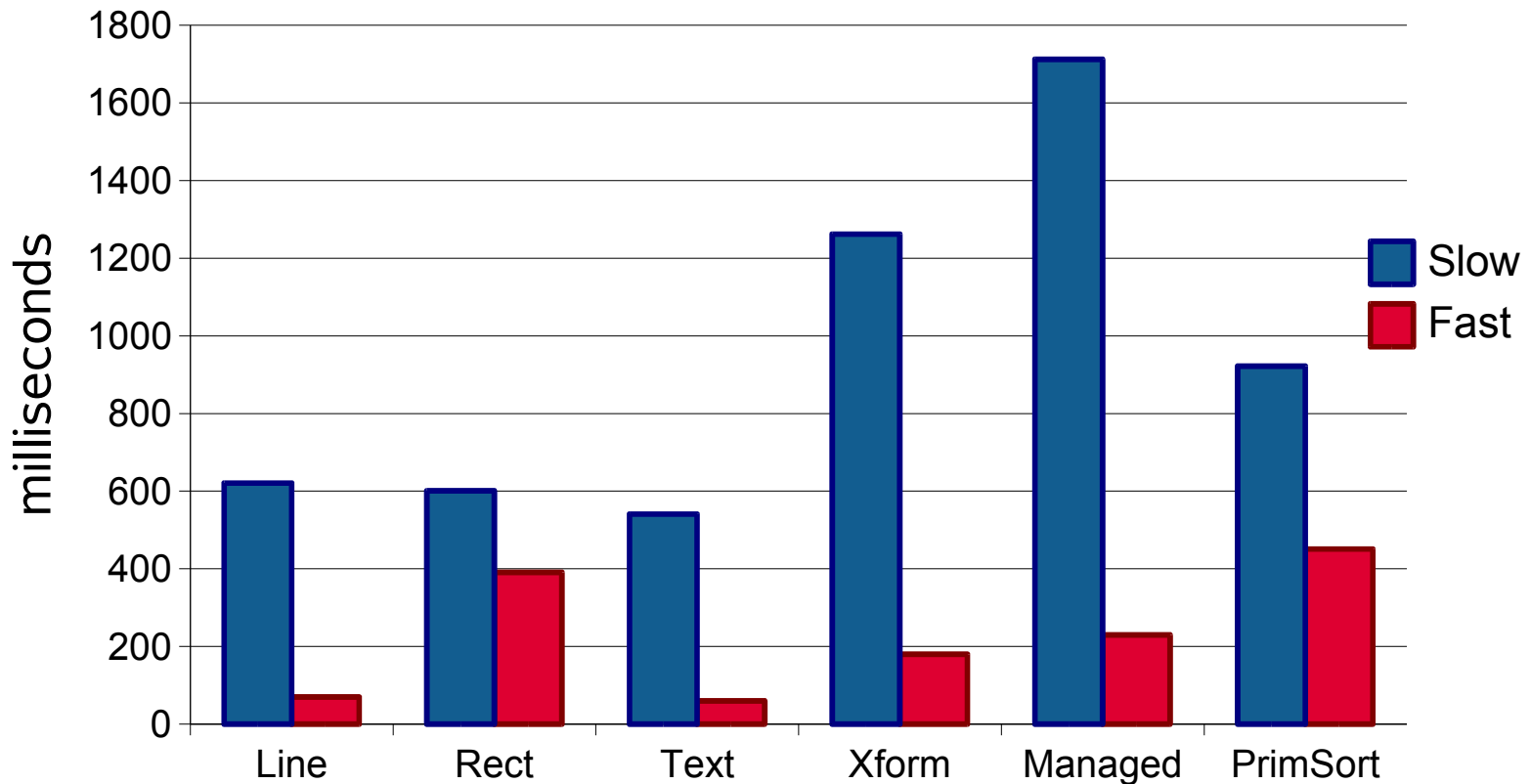
```
Graphics.drawLine(0, 0, 100, 100);
```

- Not:

```
Shape line =  
    new Line2D.Double(0, 0, 100, 100);  
Graphics.fill(line);
```

# Gratuitous Bar Chart (Graphics Tips Benchmark)

Test timings  
(smaller is better)



# Graphics Tips & Tricks:

## Image Creation/Loading: Old vs. New

- Old:
  - Toolkit: `createImage()`, `getImage()`
  - Applet: `getImage()`
  - `ImageIcon.getImage()`
  - All of these load asynchronously (`ImageIcon` uses `MediaTracker` internally to synchronize)
  - All return objects of type `Image`
- New:
  - `ImageIO.read()`
  - `new BufferedImage()`
  - `Component.createImage()`
  - `GraphicsConfiguration.createCompatibleImage()`
  - Synchronous creation, return `BufferedImage` objects

<http://weblogs.java.net/cs/weblog/view/wlg/1739>

# Thoughts on Future Graphics Projects

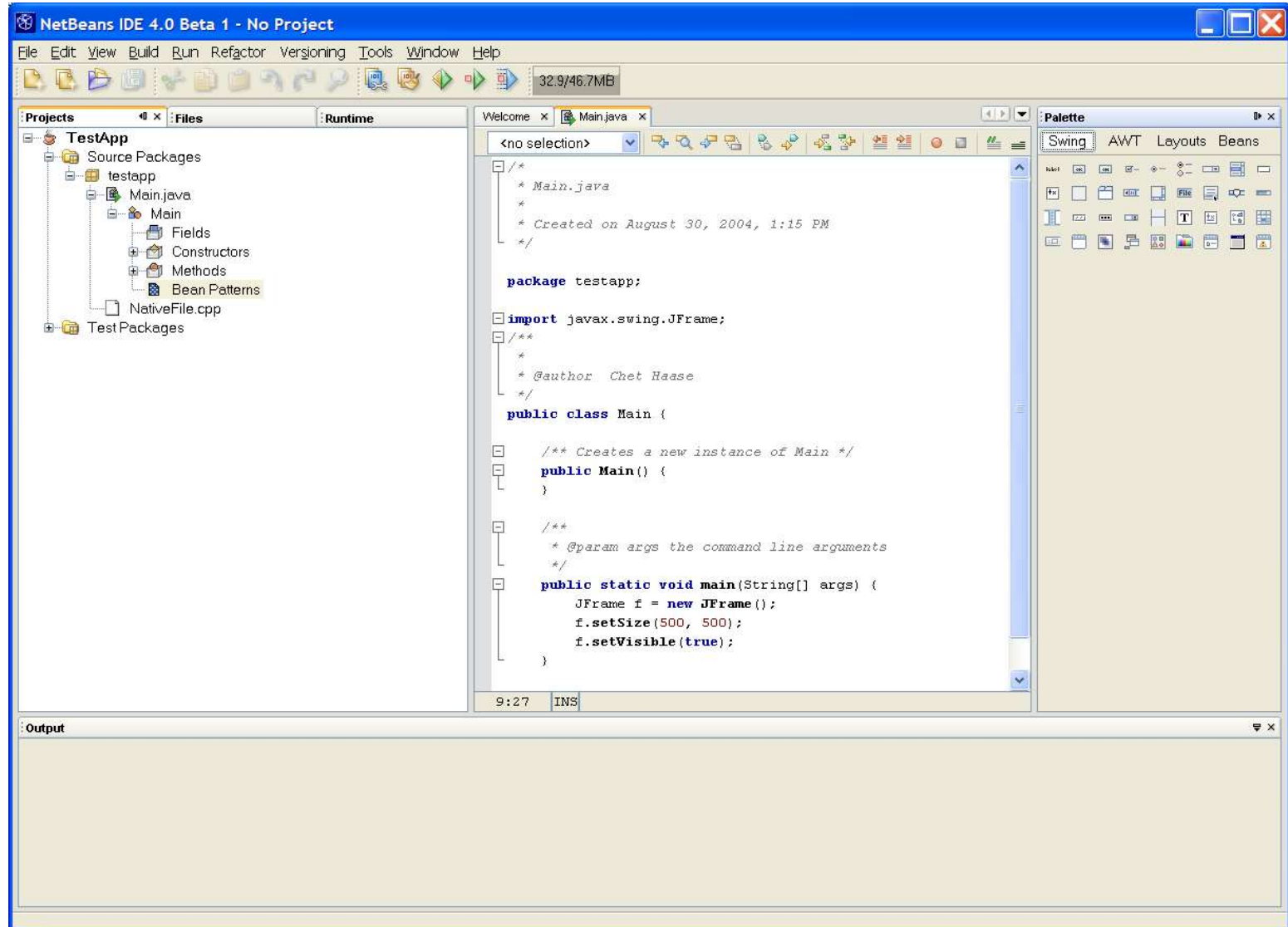
*(No commitments yet...)*

- Longhorn/Avalon Performance and Look&Feel
- Ease of Use (New higher-level APIs?)
- Rendering primitive quality (e.g., ovals)
- Text quality
- Repaint issues (Multithreading, gray rect, ...)
- More hardware acceleration
- More IPP printing support
- java.awt.geom.Area improvements
- JOGL/Swing/2D/Java3D/AWT cooperation
- Fullscreen/DisplayMode on Linux/Solaris
- SVG and SVG-related APIs in core?
- Performance, performance, performance, ...

# And in Other News

- JDNC: JDesktop Networked Components
  - <http://jdnc.dev.java.net/>
  - Easier creation of data-bound, networked rich applications
  - Higher-level Swing components
  - XML and Java APIs for JDNC apps
  - Experiment for now, also feeding ground for future Swing features
- JDIC: JDesktop Integration Components
  - <http://jdic.dev.java.net/>
  - Easier integration of Java apps with native desktop
  - screensavers, toolbar apps, browser, etc.

# And now, a word from our sponsor...



# *And now, a word from our sponsor...*

- NetBeans 4.0: way better this time around...
  - My personal favorite: No “mount” required...
  - Refactoring
  - GUI builder
  - Better startup time, more responsive
- And heck, it's still free!

# Your turn...

- Questions?
- ~~Complaints~~ Issues?
- Feature Requests?

# For More Information

- j2se: <http://java.sun.com/j2se>
- Java2D:
  - Main site (whitepapers, links, notes):
  - Feedback to dev team (link from 2D site)
- Community sites:
  - forums, articles, blogs, projects
  - <http://java.net>
    - for java development in general
  - <http://javadesktop.org>
    - for all things desktop-client related
  - <http://javagaming.org>
    - for game development in particular

# Demos!

Chet Haase  
Senior Staff Engineer  
Java2D  
September 16, 2004

