

**CS 126 Lecture S2:  
Introduction to Java Applets**

# Outline

- **Introductions**
- Your first applet and more tools of trade
- Life cycle of an applet
- Simple drawing and events
- Conclusions

# Applets: Beyond Animated Clowns

- What can you do when you can slurp code over the net?
- Extensibility
  - Bill Joy: “No more protocols; just code!”
  - No need for hard wired network protocols
  - No need for hard wired information content protocols
- A brave new world
  - New way of structuring applications (local or distributed)
  - New way of structuring operating systems (local or distributed)
- Today is only an introduction to the bare basics
  - Encourage interested people to explore on their own
  - It's fun and there's nothing hard

# Learning About Applets

- Again, take advantage of on-line resources
  - Go through tutorials
  - Always look for existing code to steal
  - Read online documentations to learn about library functionalities
- A warning
  - The GUI stuff is most vulnerable to version confusions
  - “AWT”, “JFC”, “Swing”, .....?!
  - The GUI stuff is also most buggy and least compatible
- (Don't get scared: you need to know very little to survive this class, so the advice is mostly for people who want more.)

# Outline

- Introductions
- Your first applet and more tools of trade
- Life cycle of an applet, “funny” part
  - You have to write a whole bunch of methods you don't call
  - You call a whole bunch of methods that you didn't write
- Simple drawing and events
- Conclusions

# Your First Java Applet

```
import java.applet.Applet;
import java.awt.Graphics;

public class Hello extends Applet {
    public void paint(Graphics g) {
        g.drawString("Hello world!", 125, 95);
    }
}
```

Hello.java

hello.html

```
<HTML><BODY>
<APPLET CODE=Hello.class WIDTH=300 HEIGHT=200></APPLET>
</BODY></HTML>
```

- To try it
  - Compile: **javac Hello.java**
  - Test: **appletviewer hello.html**
  - Or: put all these files in a publicly accessible directory (such as `~/public_html` and view using **netscape**)
- What happens
  - .html and .class files are slurped over the net
  - The browser has a virtual machine (interpreter) in it
  - It checks for security violations and runs it if ok.

# Life Cycle of an Applet

```
import java.applet.Applet;
import java.awt.Graphics;

public class Simple extends Applet {
    StringBuffer buffer;

    public void init() {
        buffer = new StringBuffer();
        addItem("initializing... ");
    }

    public void start() {
        addItem("starting... ");
    }

    public void stop() {
        addItem("stopping... ");
    }
}
```

```
public void destroy() {
    addItem("preparing for unloading...");
}

void addItem(String newWord) {
    System.out.println(newWord);
    buffer.append(newWord);
    repaint();
}

public void paint(Graphics g) {
    g.drawString(buffer.toString(), 5, 15);
}
```

- **init()**: browser calls it when applet first loaded
- **start()**: start execution (eg. after becoming visible)
- **stop()**: stop execution (eg. after switching to different page)
- **destroy()**: clean up after final exit
- **paint()**: browser tells it it's time to redraw

# A Slightly Larger Example

```
import java.applet.Applet;  
import java.awt.*;  
import java.awt.event.*;
```

```
class Spot {  
    public int size;  
    public int x, y;  
  
    public Spot(int size) {  
        this.size = size;  
        this.x = -1;  
        this.y = -1;  
    }  
}
```

```
public class ClickMe extends Applet  
    implements MouseListener {  
    private Spot spot = null;  
    private static final int RADIUS = 7;
```

A helper class for the dot

Later

A constant that can't be changed

## Example (cont.) -- Drawing

```
public void paint(Graphics g) {
    // draw a black border and a white background
    g.setColor(Color.white);
    g.fillRect(0, 0, getSize().width - 1,
               getSize().height - 1);
    g.setColor(Color.black);
    g.drawRect(0, 0, getSize().width - 1,
               getSize().height - 1);

    // draw the spot
    g.setColor(Color.red);
    if (spot != null) {
        g.fillOval(spot.x - RADIUS,
                  spot.y - RADIUS,
                  RADIUS * 2, RADIUS * 2);
    }
}
```

## Example (cont.) -- Event Handling

```
public class ClickMe extends Applet  
implements MouseListener {
```

```
...
```

“this” is the reference to this instance of the class.

MouseListener is an interface. ClickMe promises to implement everything specified by the interface. (Kind of like multiple inheritance in C++)

```
public void init() {  
    addMouseListener(this);  
}
```

As long as ClickMe promises to implement the interface, it can now accept mouse events.

```
public void mousePressed(MouseEvent event) {  
    if (spot == null) {  
        spot = new Spot(RADIUS);  
    }
```

The browser calls the applet through this method when the mouse is pressed.

```
        spot.x = event.getX();  
        spot.y = event.getY();  
        repaint();  
}
```

Figure out where the mouse is and trigger a paint() through repaint().

Don't need these, but a promise is a promise.

```
public void mouseClicked(MouseEvent event) {}  
public void mouseReleased(MouseEvent event) {}  
public void mouseEntered(MouseEvent event) {}  
public void mouseExited(MouseEvent event) {}  
}
```

# Outline

- Introductions
- ~~Your first applet and more tools of trade~~
- ~~Life cycle of an applet~~
- ~~Simple drawing and events~~
- **Conclusions**

# The “Truth”

- “KISS”
  - Large number of complicated features of C++ gone
  - The language is incredibly small
  - Flip side: huge number of libraries and you can't be a serious Java programmer without knowing a lot about them
- “Modern”
  - Garbage collection, strongly typed, exceptions, support for multi-threading and networking
  - Flip side: ideas have been around in the research community for ages: Modula-3, Smalltalk, Lisp, C++, Object C
- “Secure”
  - A nice three-tier protection system: verifier, class loader, and security manager.
  - Can reason about it formally
  - Flip side: bugs

## The “Truth” (cont.)

- “Productive”
  - Much less debugging headaches: no pointer probs, exceptions
  - Stealing has never been easier: the net, portability, reusability
  - Excellent documentation
  - Large and growing body of libraries to help: utilities, media, GUI, networking, threads, databases, cryptography...
  - Flip side: versions, large libraries
- “Slow”
  - Interpreted, too many tiny objects and methods
  - Flip side: just-in-time compiling can make things almost as fast as native code
- “Hype”
  - Important for momentum which translates into community expertise and support, applications, tools, and libraries
  - Flip side: hasty decision-making to feed the frenzy
- Only game in town?
  - Unprecedented roles for scripting languages on the net