

## 1999 APS105S Tutorial

Last Day — solved some recursion problems

Today — solve linked list problems

News — tutorial today

- first half: cover some Java useful in some projects
- second half: Q&A... you ask, I answer...

## Applets

- is an object!
- no 'main' method
- create HTML file:

```
<APPLET code="MyProg.class" width=300
height=300></APPLET>
```
- use "appletviewer MyProg.html"
- imagine your applet:
  - don't use "static" unless it is to be shared with "other running copies of your program" (create new copies via clone)
  - is under the control of appletviewer (built-in main method)
  - when user does something, appletviewer responds
  - if your program provides some special methods, appletviewer will call them
- eg:

```
void init( Graphics g )
void paint( Event e, int x, int y )
boolean mouseDown( Event e, int x, int y )
boolean mouseMove( Event e, int x, int y )
boolean mouseDrag( Event e, int x, int y )
```

- ONE THING HAPPENS AT A TIME
- you should "finish" your work in mouseDown(), return true
- if you don't "finish" you won't get another mouseDown() call

## Getting Text Input

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

public class Text extends Applet implements ActionListener {

    Label    promptA;
    TextField inputA;
    double   a;

    // init() is called once at applet startup time.
    // set up the graphical user interface components
    // add a label (prompt), then add an input box.
    public void init ()
    {
        promptA = new Label( "Enter the value of a:" );
        add( promptA );

        inputA = new TextField ( "0", 10);
        inputA.addActionListener(this);
        add( inputA );
    }

    // whenever the user presses RETURN, this
    // method is called.
    public void actionPerformed(ActionEvent e)
    {
        String stringA = inputA.getText();

        try {
            a = Double.valueOf(stringA).doubleValue();
        }
        catch( NumberFormatException nfe ) {
            // comes here if converting string to double fails.
            a = 0.0;
        }

        // show the accepted value of a on the status line
        showStatus( "Value of a: " + a );
        repaint ();
    }
}
```

## Pausing

- animating... controlling speed
- to pause, don't do this:

```
for( int i=0; i < 10000000; i++ ) {  
    // do nothing  
}
```
- proper way:

```
try {  
    Thread.sleep( 10 );    // 10 milliseconds  
}  
catch( InterruptedException e ) {  
}
```
- this causes your program to “sleep” for 10 ms
- the try { } catch { } block
  - similar to “if... else”
  - try doing first part try { ... }
  - if an exception arises, do the catch{ ... }
  - the (InterruptedException e) is the argument to catch
  - lets you determine what type of error occurred
  - we'll ignore it

## Timing

```
long start = System.currentTimeMillis();  
doSomeMethod();  
long stop = System.currentTimeMillis();  
double elapsedSeconds = (stop-start)/1000.0;
```

## Animation

- animating applets is difficult
- your program does one thing at a time:
  - calls mouseDown(), returns.
  - calls paint(), returns.
  - mouseDown() should always return quickly
- if you're doing a “screen saver”,
  - easiest type of animation
  - no user input to worry about
  - start animation in the mouseDown() method
  - never have to **return** if you don't want too
  - not clean or proper, but “oh well”
- next hardest animation
  - short animation clip reacting to user input
  - eg: firing a missile across the screen
  - when missile goes off screen, animation stops
  - can start animation in mouseDown() method
  - do a return (soon!) when animation stops
  - can now get a new mouseDown() call
- hardest animation
  - eg: games like pong, arkanoid, pacman, space invaders
  - other objects moving “in the background”
  - eg: a ball, pacman ghosts, invading UFOs
  - if you don't click the mouse
    - your program isn't “running”
    - only runs when mouseDown() called, or paint() called
    - called only in response to USER ACTIONS
  - how to work in background?
  - need to use “Thread” class
  - beyond scope of tutorial....

## Threads

```
public class MyThread extends Thread
{
    Applet theApplet;

    public MyThread( Applet a ) {
        theApplet = a; // remember the applet
    }

    public void run() {
        // this method is called when the thread
        // starts running. do something!
        animate();
    }

    private void myPause( int ms ) {
        try { Thread.sleep( ms ); }
        catch( InterruptedException e ) { }
    }

    private void animate() {
        // ... do something long here....
        while( true ) {
            myPause( 100 ); // wait 100ms
            a.repaint(); // ask applet to repaint!
        }
    }
}
```

```
...inside your applet:
MyThread myThread = new MyThread( this );
myThread.start(); // to start it, calls run()
myThread.stop(); // to stop it
```

The hard part about threads is communicating values between them. What if the Applet and the Thread want to both update the same variable (say, increment a counter) at the same time?