Finishing an Applet

Before You Ship That Applet

Stop! Before you let the whole world know about your applet, make sure the answer to all of the following questions is **yes**.

1. Have you removed or disabled debugging output?

Debugging output (generally created with `System.out.println()`), while useful to you, is generally confusing or annoying to users. If you need to give textual feedback to the user, try to do it inside the applet's display area or in the status area at the bottom of the window (see the Applet `showStatus()` method).

2. Does the applet stop running when it's offscreen?

Most applets should not use CPU resources when the browser is iconified or is displaying a page that doesn't contain the applet. If your applet code doesn't launch any threads explicitly, then you're OK.

If your applet code launches any threads, then unless you have a REALLY GOOD excuse not to, you should implement the `stop()` method so that it stops and then destroys (by setting to null) the threads you launched. For example:

```
public synchronized void stop() {
    if (refresh != null) {
        refresh.stop();
        refresh = null;
    }
}
```

3. If the applet does something that might get annoying –– play sounds or animation, for example –– does it give the user a way of stopping the annoying behavior?

Be kind to your users. Give them a way to stop the applet in its tracks, without leaving the page. In an applet that otherwise doesn't respond to mouse clicks, you can do this by implementing the `mouseDown()` method so that a mouse click suspends or resumes the annoying thread. For example:

```
boolean threadSuspended = false; //an instance variable

public boolean mouseDown(Event e, int x, int y) {
    if (threadSuspended) {
        myThread.resume();
```

```
        } else {
            myThread.suspend();
        }
        threadSuspended = !threadSuspended;
        return true;
    }
```

Finishing an Applet