Applets

Murray Cole

Portability and Security

- JVM and bytecode make portability "easy"
- facilitates sharing of programs, which can be good and bad
- can we make sharing safer?

Applets

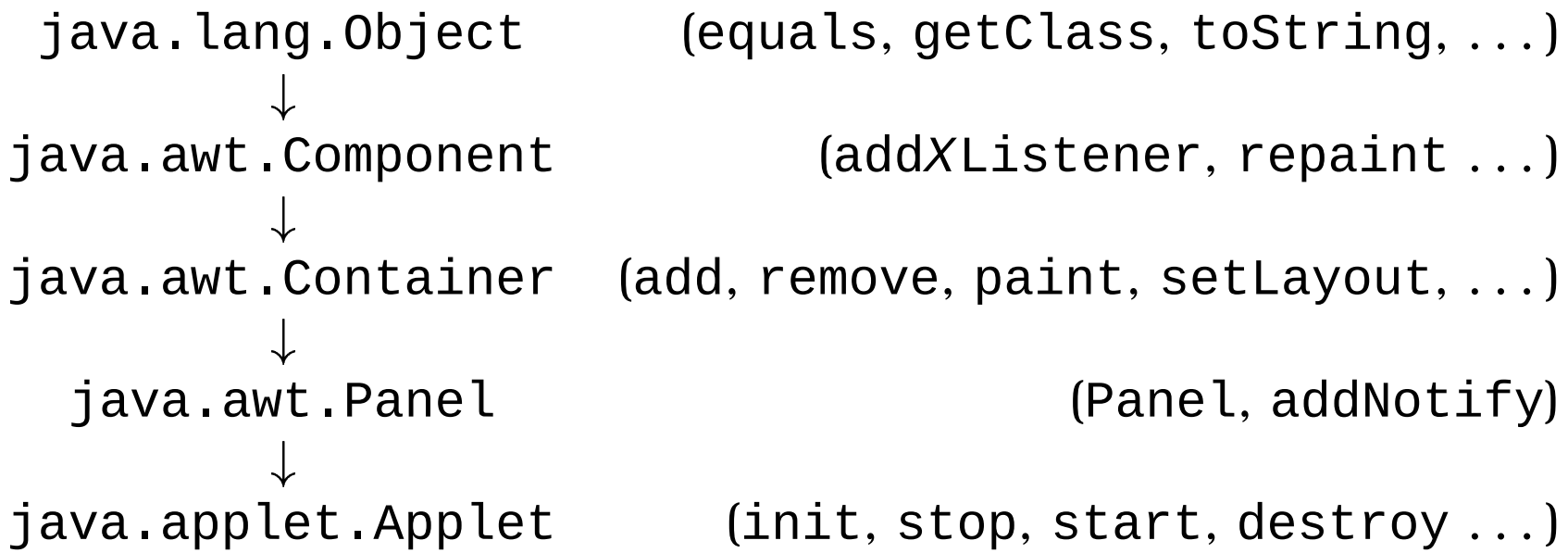
A sub-class of Java programs which

- run from within web pages (byte code is provided by the web site (server) but run in your browser (client))
- are restricted in what they can do (no local file access, no networking (eg e-mail), no "messing with the local system"). This is called sandboxing.

The restrictions are enforced by the JVM through its security manager.

Slide 2

on board illustration of client server nature of web activity, download byte code & run on client, not server. Think of some little graphical app. Also reduced load on server.


java.lang.Object (equals, getClass, toString, ...)
java.awt.Component (addXListener, repaint ...)
java.awt.Container (add, remove, paint, setLayout, ...)
java.awt.Panel (Panel, addNotify)
java.applet.Applet (init, stop, start, destroy ...)

Class `java.applet.Applet`

The applet interface (in the Java sense, not the GUI sense) specifies a number of new methods, including

- `void init()`, invoked when the applet is first downloaded
- `void start()`, invoked whenever the applet is started (or re-started)

Since it extends `Component` it also has a `paint` method.

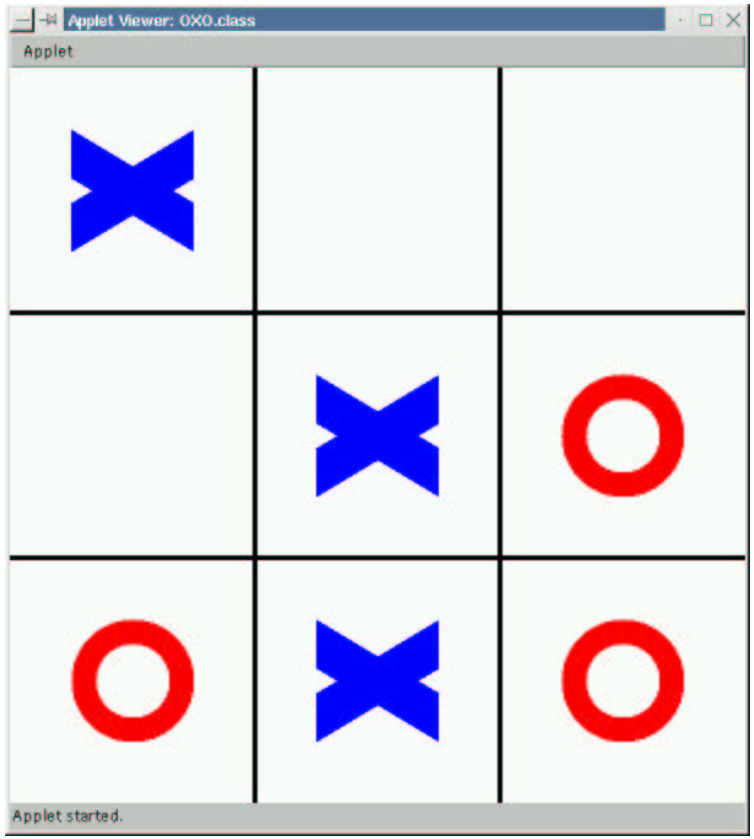
```
public class Clicker extends Applet {
  private int clicks; // Click counter
  public void paint(Graphics g) {
    g.setColor(Color.white); g.fillRect(0, 0, 400, 400);
    g.setColor(Color.black);
    g.setFont(new Font("TimesRoman", Font.BOLD, 64));
    g.drawString("Clicks: " + clicks, 60, 240);
  }
  public void init() {
    addMouseListener(new MouseAdapter() {
      public void mouseClicked(MouseEvent e) {
        clicks++; repaint();
      }
    });
  }}
```

The HTML file

HyperText Markup Language (HTML) describes web pages.

```
<html>
  <head>
    <title>The Clicker Applet</title>
  </head>
  <body>
    <h1>The Clicker Applet</h1>
    <applet code="Clicker.class" width="400" height="400">
    </applet>
    <hr>
  </body>
</html>
```

A Really Useful Applet



Design Issues (1)

The game itself

- board and current moves
- choosing a computer move
- winning

```
public class OXO extends Applet {
    int movesmade;
    boolean player[]   = new boolean[9];
    boolean computer[] = new boolean[9];
    boolean MoveOK (int m) {
      return !(computer[m] || player[m]);
    }
    void computerMove () {
      int m;
      m = (int) (Math.random()*9);
      while (!MoveOK(m)) {
         m = (int) (Math.random()*9);
      }
      movesmade++; computer[m] = true;
    }
```

```
// Check all the possible ways of winning
boolean won (boolean []moves) {
    return (moves[0] && moves[1] && moves[2]) ||
           (moves[3] && moves[4] && moves[5]) ||
           (moves[6] && moves[7] && moves[8]) ||
           (moves[0] && moves[3] && moves[6]) ||
           (moves[1] && moves[4] && moves[7]) ||
           (moves[2] && moves[5] && moves[8]) ||
           (moves[0] && moves[4] && moves[8]) ||
           (moves[2] && moves[4] && moves[6]);
}
```

```
public void start() {
    movesmade = 0;
    for (int i=0; i<9; i++) {
      computer[i] = false;
      player[i]   = false;
    }
```

```
this.addMouseListener(new MouseAdapter() {
   public void mouseReleased(MouseEvent e) {
     int x = e.getX();  int y = e.getY();
     Dimension d = getSize();
     int r = (y * 3) / d.height; int c = (x * 3) / d.width;
     if (MoveOK(r*3 + c) && !won(player) && !won(computer))
        player[r*3+c] = true;   // Make the move
        movesmade++;
        repaint();
        if (!won(player) && (movesmade < 9)) {
           computerMove();
           repaint();
        }
     }
}});
```

Design Issues (2)

Representing the game on screen

- capturing the player's move
- drawing the grid
- drawing moves

```
    public void paint(Graphics g) {
        Dimension d = getSize(); setBackground(Color.white);
        g.setColor(Color.black);
        int xoff = d.width / 3; int yoff = d.height / 3;
        g.fillRect(xoff-2,   0, 4, d.height);
        g.fillRect(2*xoff-2, 0, 4, d.height);
        g.fillRect(0, yoff-2, d.width, 4);
        g.fillRect(0, 2*yoff-2, d.width, 4);
        for (int r = 0 ; r < 3 ; r++)
          for (int c = 0 ; c < 3 ; c++)
            if (player[r*3+c])
              drawX(g, r, c, d.width/3, d.height/3);
            else if (computer[r*3+c])
              drawO(g, r, c, d.width/3, d.height/3);
}
```

```
void drawO (Graphics g, int r, int c, int boxw, int boxh) {
 g.setColor(Color.red);
 g.fillOval(c*boxw + boxw/4, r*boxh + boxh/4, boxw/2, boxh/2)
 g.setColor(Color.white);
 g.fillOval(c*boxw + boxw/4 + boxw/10, r*boxh + boxh/4 +
             boxw/10, boxw/2 - boxw/5, boxh/2 - boxw/5);
}
```

```
void drawX (Graphics g, int r, int c, int boxw, int boxh) {
    Polygon p1 = new Polygon(); Polygon p2 = new Polygon();

    // Top left to bottom right swish
    p1.addPoint(c*boxw + boxw/4,  r*boxh + boxh/4);
    p1.addPoint(c*boxw + boxw/4,  r*boxh + boxh/4 + boxw/5);
    p1.addPoint((c+1)*boxw - boxw/4,  (r+1)*boxh - boxh/4);
    p1.addPoint((c+1)*boxw - boxw/4,  (r+1)*boxh - boxh/4
                                      - boxw/5);

    // Similarly bottom left to top right swish

    g.setColor(Color.blue);
    g.fillPolygon(p1); g.fillPolygon(p2);
}
```