

Enhanced Digital Clock Applet

Setting applet parameters in the web page.

The applet tag in HTML:

```
<applet code=DigitalClock2.class
width=250 height=80>
  <param name=color value=blue>
</applet>
```

Syntax:

```
<applet code= applet_class_file
width= width_in_pixel height= height_in_pixel >
  <param name= param_name1 value= param_value1 >
  ...
  <param name= param_namen value= param_valuen >
</applet>
```

Getting Applet Parameters

```
import java.awt.Color;

public class DigitalClock2 extends DigitalClock {
    public void init () {
        String param = getParameter("color");
        if ("red".equals(param)) {
            color = Color.red;
        } else if ("blue".equals(param)) {
            color = Color.blue;
        } else if ("yellow".equals(param)) {
            color = Color.yellow;
        } else if ("orange".equals(param)) {
            color = Color.orange;
        } else {
            color = Color.green;
        }
    }
}
```

The java.awt.Graphics Class

- ◆ Represent *Graphics Context*.
- ◆ A *graphics context* is an abstraction of various *drawing surfaces*:
 - ◆ screen
 - ◆ printer
 - ◆ off-screen image (an image stored in memory)
- ◆ Provide a rich set of graphics methods.

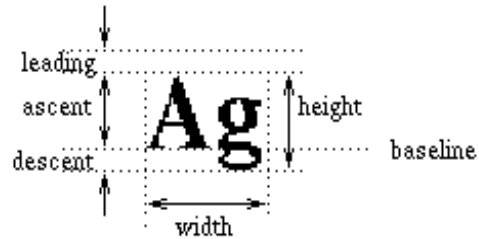
```
drawString()    drawLine()
drawArc()       fillArc()
drawOval()      fillOval()
drawPolygon()   fillPolygon()
drawRect()      fillRect()
drawRoundRect() fillRoundRect()
```

The java.awt.Graphics Class (cont'd)

Methods:

setColor(color)	set the current color
setFont(font)	set the current font
setPaintMode()	set the paint, or overwrite mode
setXORMode(color)	set the XOR mode
getColor()	get the current color
getFont()	get the current font
getFontMetrics()	get the font metrics of the current font
getFontMetrics(font)	get the font metrics for the specified font

The java.awt.FontMetrics Class



Methods:

```
getAscent()  
getDescent()  
getHeight()  
getLeading()  
stringWidth(s)
```

Scrolling Banner Applet

```
public class ScrollingBanner  
    extends java.applet.Applet  
    implements Runnable {  
  
    <field declarations>  
  
    public void init() { ... }  
    public void paint(Graphics g) { ... }  
    public void run() { ... }  
    public void start() { ... }  
    public void stop() { ... }  
}
```

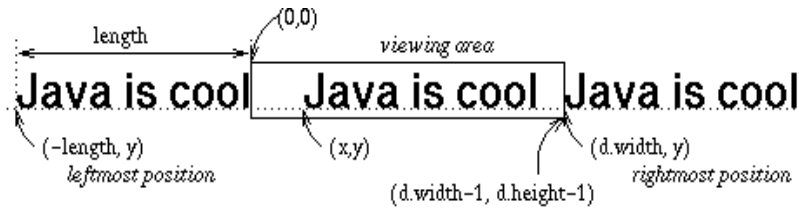
Field Declarations

```
protected Thread bannerThread;  
protected String text;  
protected Font font =  
    new java.awt.Font("Sans-serif", Font.BOLD, 24);  
  
protected int x, y;  
protected int delay = 100;  
protected int offset = 1;  
protected Dimension d;
```

Initialization

```
public void init() {  
    // get parameters "delay" and "text"  
    String att = getParameter("delay");  
    if (att != null) {  
        delay = Integer.parseInt(att);  
    }  
    att = getParameter("text");  
    if (att != null) {  
        text = att;  
    } else {  
        text = "Scrolling banner.";  
    }  
    // set initial position of the text  
    d = getSize();  
    x = d.width;  
    y = font.getSize();  
}
```

Drawing The Scrolling Banner



Paint the Current Frame

```
public void paint(Graphics g) {
    // get the font metrics to determine the length of the text
    g.setFont(font);
    FontMetrics fm = g.getFontMetrics();
    int length = fm.stringWidth(text);
    // adjust the position of text from the previous frame
    x -= offset;
    // if the text is completely off to the left end
    // move the position back to the right end
    if (x < -length)
        x = d.width;
    // set the pen color and draw the background
    g.setColor(Color.black);
    g.fillRect(0,0,d.width,d.height);
    // set the pen color, then draw the text
    g.setColor(Color.green);
    g.drawString(text, x, y);
}
```

The start(), stop(), and run() Methods

```
public void start() {
    bannerThread = new Thread(this);
    bannerThread.start();
}

public void stop() {
    bannerThread = null;
}

public void run() {
    while (Thread.currentThread() == bannerThread) {
        try {
            Thread.currentThread().sleep(delay);
        }
        catch (InterruptedException e){}
        repaint();
    }
}
```

How to Avoid Flickering?

- ◆ Flickering is caused by `repaint()`
 - ◆ `repaint()` calls the `update()` method.
 - ◆ The default `update()` method does the following:
 1. paint the whole area with the background color;
 2. set the foreground color;
 3. call the `paint()` method.
 - ◆ The `update()` method is also called by the system to update windows.
- ◆ Solution:
 - ◆ override the `update()` method
 - ◆ use an off-screen image

Using An Off-Screen Image

A.k.a. *double-buffering*

```
import java.awt.*;

public class ScrollingBanner2
    extends ScrollingBanner {
    protected Image image;
        // The off-screen image
    protected Graphics offscreen;
        // The off-screen graphics

    public update(Graphics g) { ... }
    public paint(Graphics g) { ... }
}
```

Using An Off-Screen Image(cont'd)

```
public void update(Graphics g) {
    // create the offscreen image if it is the first time
    if (image == null) {
        image = createImage(d.width, d.height);
        offscreen = image.getGraphics();
    }
    // draw the current frame into the off-screen image
    // using the paint method of the superclass
    super.paint(offscreen);

    // copy the off-screen image to the screen
    g.drawImage(image, 0, 0, this);
}

public void paint(Graphics g) {
    update(g);
}
```

Animation Applet Idiom

Category

Behavioral implementation idiom.

Intent

For an applet to continuously update its appearance without user input or intervention.

Also known as

Active Applet.

Applicability

Use the Animation Applet Idiom to animate dynamic processes.

Animation Applet Idiom (cont'd)

```
public class AnimationApplet
    extends java.applet.Applet implements Runnable {

    Thread mainThread = null;
    int delay;

    public void start() {
        if (mainThread == null) {
            mainThread = new Thread(this);
            mainThread.start();
        }
    }

    public void stop() {
        mainThread = null;
    }
}
```

Animation Applet Idiom (cont'd)

```
public void run(){
    while (Thread.currentThread() == mainThread) {
        repaint();
        try{
            Thread.currentThread().sleep(delay);
        } catch (InterruptedException){}
    }
}

public void paint(java.awt.Graphics g) {
    <paint the current frame>
}

<other methods and fields>
}
```

Another Applet -- Bouncing Ball

```
import java.awt.*;

public class BouncingBall
    extends java.applet.Applet implements Runnable {

    protected Color color = Color.green;
    protected int radius = 20;
    protected int x, y;
    protected int dx = -2, dy = -4;
    protected Image image;
    protected Graphics offscreen;
    protected Dimension d;

    // ...
}
```

Bouncing Ball (cont'd)

```
public void init() {
    String att = getParameter("delay");
    if (att != null) {
        delay = Integer.parseInt(att);
    }
    d = getSize();
    x = d.width * 2 / 3 ;
    y = d.height - radius;
}
```

Bouncing Ball (cont'd)

```
public void update(Graphics g) {
    // create the off-screen image buffer
    // if it is invoked the first time
    if (image == null) {
        image = createImage(d.width, d.height);
        offscreen = image.getGraphics();
    }

    // draw the background
    offscreen.setColor(Color.white);
    offscreen.fillRect(0,0,d.width,d.height);
}
```

Bouncing Ball (cont'd)

(method update() continued.)

```
// adjust the position of the ball
// reverse the direction if it touches
// any of the four sides
if (x < radius || x > d.width - radius) {
    dx = -dx;
}
if (y < radius || y > d.height - radius) {
    dy = -dy;
}
x += dx;
y += dy;
```

Bouncing Ball (cont'd)

(method update() continued.)

```
// draw the ball
offscreen.setColor(color);
offscreen.fillOval(x - radius, y - radius,
                  radius * 2, radius * 2);

// copy the off-screen image to the screen
g.drawImage(image, 0, 0, this);
}

public void paint(Graphics g) {
    update(g);
}
```

Bouncing Ball (cont'd)

```
// The animation applet idiom
protected Thread bouncingThread;
protected int delay = 100;

public void start() {
    bouncingThread = new Thread(this);
    bouncingThread.start();
}

public void stop() {
    bouncingThread = null;
}
```

Bouncing Ball (cont'd)

```
public void run() {
    while (Thread.currentThread() ==
           bouncingThread) {
        try {
            Thread.currentThread().sleep(delay);
        } catch (InterruptedException e){}
        repaint();
    }
}
```