5.5 Applications – Animation Applets

- Example 5.1 Digital Clock Applet – an Enhancement
  - To illustrate the use of parameters in applet with the applet tag
    - without modifying and recompiling the Digital Clock (previous version)
  - modify to allow the foreground color to be set as a parameter
    - init() method.. Override

Enhanced digital clock applet: DigitalClock2.java

```
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 GetParameter() ……. A method of the Applet class

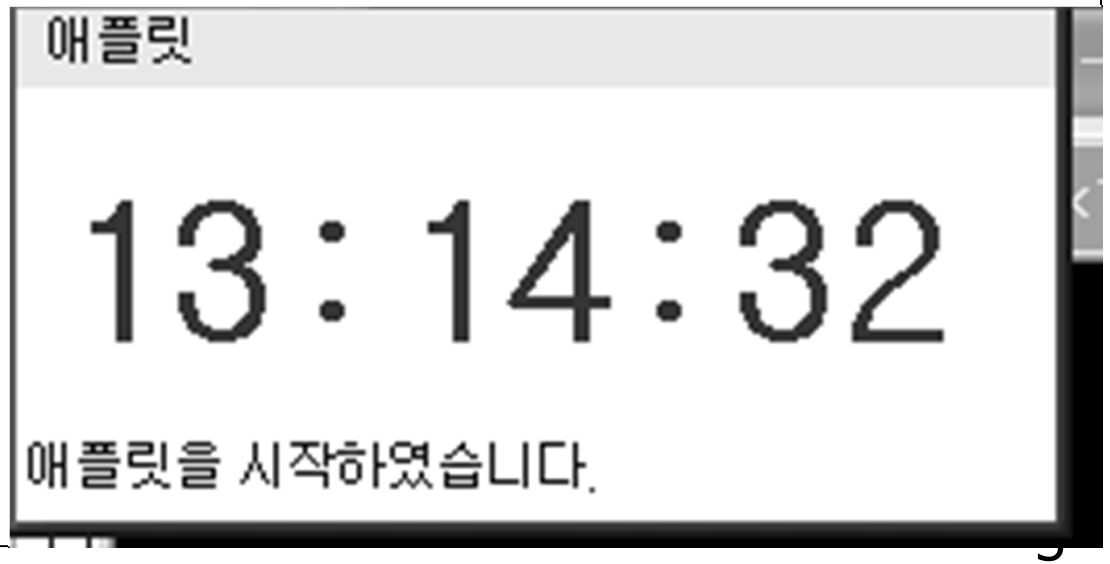
```
<applet code=DigitalClock2.class width=250 height=80>
<param name=color value=blue>
</applet>
```

- Form of the applet tag with parameters

```
<applet code=class_filename
        width=pixels height=pixels>
    <param name=param_name1 value=param_value1>
            :
    <param name=param_namen value=param_valuen>
</applet>
```

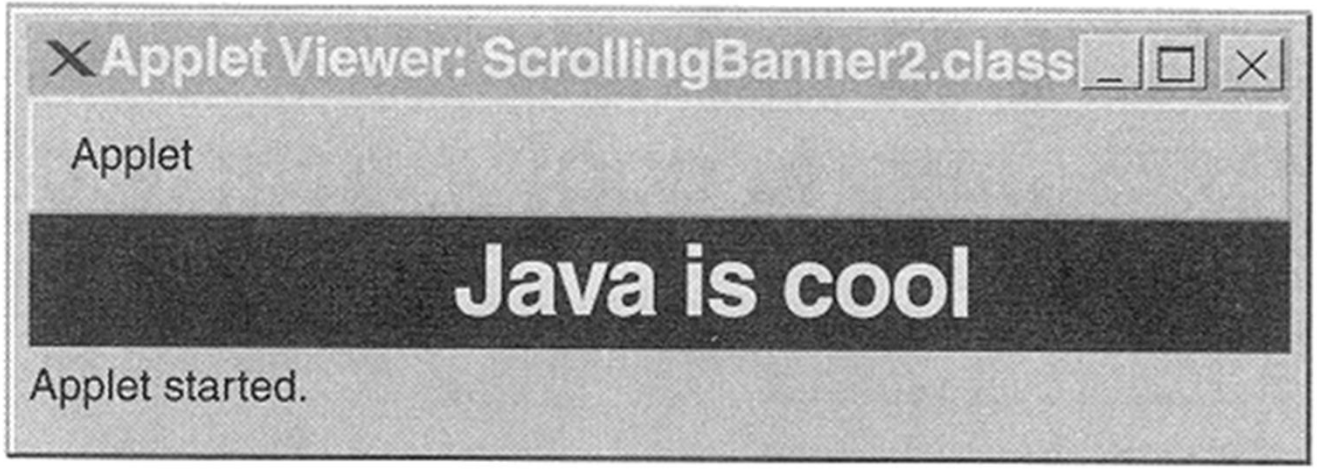
```
// Example 5.1 The Digiatl Clock Applet - An Enhancement

import java.awt.Color;
/**
 *  This is an ehhanced version of DigitalClock. It takes a parameter: color
 *  It displays the time in the following format:
 *        HH:MM:SS
 */
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 ("yelow".equals(param)) {
      color = Color.yellow;
    } else if ("orange".equals(param)) {
      color = Color.orange;
    } else {
      color = Color.green;
    }
  }
}
```

```
<!--DigitalClock2.html-->
<HTML>
  <HEAD>
    <TITLE> Extended Digital Clock Applet </TITLE>
  </HEAD>
<BODY BGCOLOR=white>
  <CENTER>
    <H1> The <B>Extended</B> Digital Clock Applet</H1>
      <P>
        <APPLET CODE=DigitalClock2.class
              WIDTH=250 HEIGHT=80>
        <PARAM NAME=color VALUE=red>
        </APPLET>
  </CENTER>
</BODY>
</HTML>
```



5.5.2 An Idiom for Animation Applets

- Animation : To Display a sequence of frames
- Motion Pictures : more than 10 frames per second
- Example 5.2 Scrolling Banner Applet – The Initial Version
  - To illustrate animation and text drawing
  - Displays a text banner that moves horizontally from right to left
  - When the banner moves completely off the left end of the viewing area, it reappears at the right end.
- Figure 5.6 The scrolling banner applet



the addition

```
// Example 5.2 The Scrolling Banner Applet - The initial Version (p. 188)

import java.awt.*;
public class ScrollingBanner
        extends java.applet.Applet implements Runnable {

  protected Thread bannerThread;  // the animation thread
  protected String text; // the text to be displayed
  protected Font font = new java.awt.Font("Sans serif", Font.BOLD, 24);
                        // the font used to display the text  p
  rotected int x, y;       // the current position of the text
  protected int delay = 100;     // the interval between two consecutive
frames in milliseconds
  protected int offset = 1;  // the distance moved between two consecutive
frames in pixels
  protected Dimension d;  // the size of the viewing area
```

```
// handles the initialization of the applet
 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();  // html에서의  WIDTH=250 HEIGHT=33
   x = d.width;
   y = font.getSize();
 }
```

```
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.drawStrin
   g(text, x, y);
 }
```

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

  public void stop() {  b
   annerThread = null;
  }

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

}
```

```
<!--ScrollingBanner.html-->
<HTML>
  <HEAD>
    <TITLE> Scrolling Banner Applet </TITLE>
  </HEAD>
<BODY BGCOLOR=black TEXT=white>
  <CENTER>
    <H1> The Scrolling Banner Applet</H1>
     <P>
       <APPLET CODE=ScrollingBanner.class
             WIDTH=250 HEIGHT=33>
       <PARAM NAME=text VALUE="Java is Cool!">
       <PARAM NAME=delay VALUE=50>
       </APPLET>
  </CENTER>
</BODY>
</HTML>
```

5.5.2 An Idiom for Animation Applets

- Idiom
  - a way to represent a template implementation of a recurring problem that may be customized and adapted in different contexts.
- Idiom: Animation Applets

Idiom ***Animation Applet***

Category: Behavioral implementation idiom.
Intent: For an applet to update continuously its appearance without user input or intervention.
Also known as: Active applet.
Applicability: Use the animation applet idiom to animate dynamic processes.

5.5.2 An Idiom for Animation Applets

- The generic Structure of the top-level class of an animation applet

```
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;
  }

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

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

  (other methods and fields)
}
```

5.5.3 Double-Buffered Animation

- The cause of flickering
  - the animation process is controlled by the while loop in the run method (P. 192)
  - during iteration, the thread sleeps for a short period of time before calling the repaint()
    - repaint() calls update()
  - Default implementation of update()
    - 1) clears background by filling it with the background color (gray or white)
    - 2) set the pen color to the foreground color, and
    - 3) calls the paint() method

5.5.3 Double-Buffered Animation

- In the Paint() methods (P. 190)
  - the background is repainted (black, this ex.)
  - and then the text is drawn
  - because the painting is directly on the screen, it cannot be completed instantaneously.
  - the background painting may take long enough for the human eyes to notice
- Solution
  - first paint it in a temporary buffer in memory.

5.5.3 Double-Buffered Animation

- Example 5.3
  - To illustrate double-buffered animation
  - eliminate the flickering
  - Solution == Double buffering or Off-screen drawing
    - at first, to paint them in a temporary buffer in memory
    - when the frame has been completed, Copy it f rom temporary buffer to screen in a single step
    - The Painting in Progress is notvisible
    - ⓪ Override Update()

5.5.3 Double-Buffered Animation

- Implementation
  - Source Code
  - override the default implementation of the update() method
  - Two additional fields

Field	Description
`image`	An off-screen image that is the same size as the viewing area
`offscreen`	A graphics context associated with the off-screen image

Double-buffered scrolling banner applet

```
import java.awt.*;

public class ScrollingBanner2 extends ScrollingBanner {

   protected Image image;          // The off-screen image
   protected Graphics offscreen;   // The off-screen graphics

   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);
   }
}
```

Scrolling Banner 2

```
// Example 5.3 The Scrolling Banner Applet - Using Double-Buffering (p. 194)


import java.awt.*;

/**
 *  An enhanced version of the scrolling banner applet.
 *  It uses double-buffering to eliminate the flickering.
 */
public class ScrollingBanner2 extends ScrollingBanner {

  protected Image image;          // The off-screen image  prot
  ected Graphics offscreen;       // The off-screen graphics
```

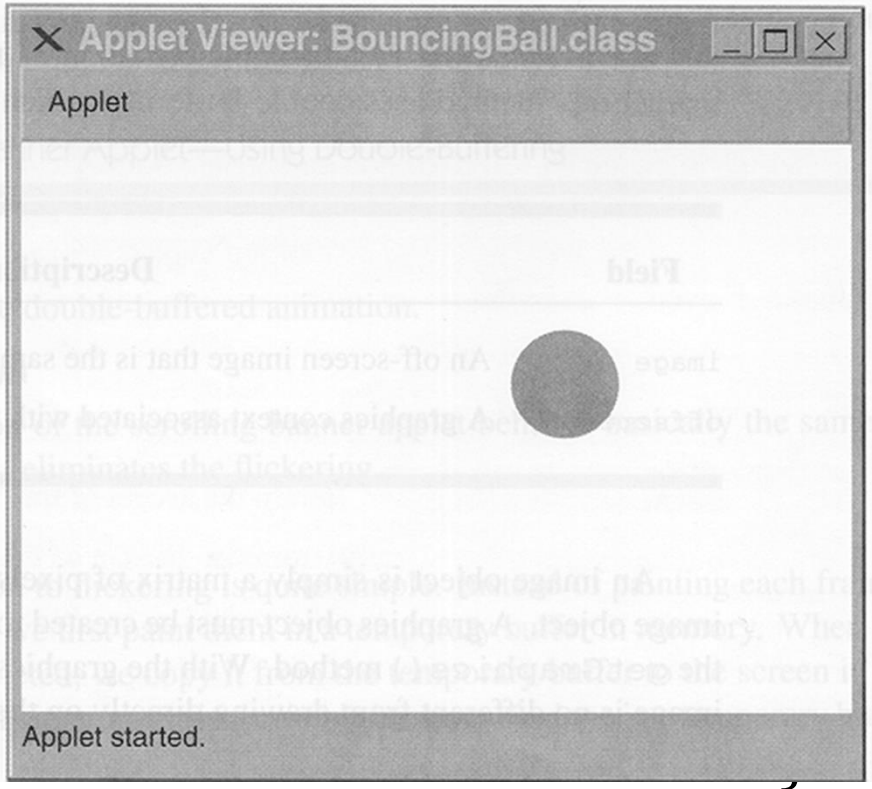
```
    public void update(Graphics g) {
    // create the offscreen image if it is the first time
    if (image == null) {
      image = createImage(d.width, d.height);
      // we cannot directly draw into an image object
      // a graphics object must be created for drawing into the image
      //       by calling getGraphics()  of
      fscreen = 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);
  }

}
```

```
<!--ScrollingBanner2.html-->
<HTML>
  <HEAD>
    <TITLE> The <B>Extended</B> Scrolling Banner Applet </TITLE>
  </HEAD>
<BODY BGCOLOR=black TEXT=white>
  <CENTER>
    <H1> The <B>Extended</B> Scrolling Banner Applet</H1>
      <P>
        <APPLET CODE=ScrollingBanner2.class
              WIDTH=250 HEIGHT=33>
        <PARAM NAME=text VALUE="Java is Cool!">
        <PARAM NAME=delay VALUE=50>
        </APPLET>
  </CENTER>
</BODY>
</HTML>
```

5.5.3 Double-Buffered Animation

- Example 5.4. The Bouncing Ball Applet
  - to illustrate the animation applet Idiom, graphics drawing, double-buffering
- The Fields of the Bouncing Ball. Figure 5.10 The bouncing ball

Field	
color	Color of the ball
radius	Radius of the ball in pixels
	Current position of the ball
dx, dy	Distance moved between two in pixels
image	Off-screen image
offscree	Off-screen graphics
	Size of the viewing area



Bouncing Ball

```
// Example 5.4 The Bouncing Ball Applet (p. 195)

import java.awt.*;

/**
 *  The Bouncing Ball Applet
 */
public class BouncingBall
    extends java.applet.Applet implements Runnable {

  protected Color color = Color.green; // Color of the ball
  protected int radius = 20;   // Radius of the Ball in pixels
  protected int x, y;  // current position of the ball
  protected int dx = -2, dy = -4;    // distance moved between two
consecutive frames
                                                  // in the x and y directions in pixels
  protected Image image;  // off-screen image
  protected Graphics offscreen;   //off-screen graphics
  protected Dimension d;   // Size of the viewing area
```

```
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;
 }

 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.setC
   olor(Color.white);  offscreen.fillRect(0,0
   ,d.width,d.height);
```

```
// 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;

   // draw the ball  offscreen.setColor(col
   or);  offscreen.fillOval(x - radius, y - ra
   dius,
                            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);
 }
```

```
// The animation applet idiom  prote
  cted Thread bouncingThread;  prot
  ected int delay = 100;

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

  public void stop() {  bou
    ncingThread = null;
  }

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

}
```

```
<!--BouncingBall.html-->
<HTML>
  <HEAD>
    <TITLE> The Bouncing Ball Applet </TITLE>
  </HEAD>
<BODY BGCOLOR=black TEXT=white>
  <CENTER>
    <H1> The Bouncing Ball Applet </H1>
      <P>
        <APPLET CODE=BouncingBall.class
              WIDTH=250 HEIGHT=150>
        <PARAM NAME=delay VALUE=50>
        </APPLET>
  </CENTER>
</BODY>
</HTML>
```