Course: Object Based Modeling
Code: CS-33105
Branch: MCA-3

Lecture 16: The Applet Class

Dr. J Sathish Kumar (JSK)
(Faculty & Coordinator)

Department of Computer Science and Engineering

Motilal Nehru National Institute of Technology Allahabad,
Prayagraj-211004

Two Types of Applets

- The first are those based directly on the **Applet** class that use the Abstract Window Toolkit (AWT) to provide the graphical user interface
- The second type of applets are those based on the Swing class **JApplet**, which inherits **Applet**.
- Swing applets use the Swing classes to provide the GUI.
- Swing offers a richer and often easier-to-use user interface than does the AWT.
- Thus, Swing-based applets are now the most popular.

Applet Basics

- AWT-based applets are subclasses of **Applet**.
- Applets are not stand-alone programs.
- Instead, they run within either a web browser or an applet viewer.
- Execution of an applet does not begin at **main( )**.
- Only, few applets even have **main( )** methods.
- Output to your applet's window is not performed by **System.out.println( )**.
- Rather, in an AWT-based applet, output is handled with various AWT methods, such as **drawString( )**, which outputs a string to a specified X,Y location.
- Input is also handled differently than in a console application.

APPLETS

- The use of the APPLET tag offers a secondary advantage when developing applets because it enables you to easily view and test the applet.
- To do so, simply include a comment at the head of your Java source code file that contains the APPLET tag.
- This way, your code is documented with the necessary HTML statements needed by your applet, and you can test the compiled applet by starting the applet viewer with your Java source code file specified as the target.
- Here is an example of such a comment:

```
/*
<applet code="MyApplet" width=200 height=60>
</applet>
*/
```

Example #1

SimpleApplet.java

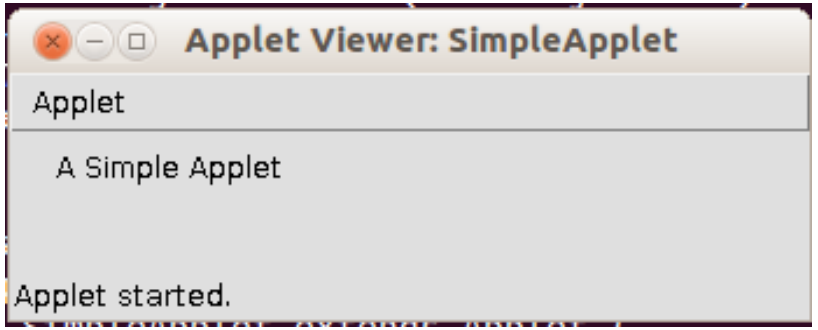
```
import java.awt.*;
import java.applet.*;
/*
<applet code="SimpleApplet" width=200 height=60>
</applet>
*/
public class SimpleApplet extends Applet {
 public void paint(Graphics g) {
 g.drawString("A Simple Applet", 20, 20);
 }
}
```

SimpleApplet.html

```
<applet code="SimpleApplet" width=200 height=60>
</applet>
```

javac SimpleApplet.java

appletviewer SimpleApplet.html



The Applet Class

Method	Description
void destroy( )	Called by the browser just before an applet is terminated. Your applet will override this method if it needs to perform any cleanup prior to its destruction.
AccessibleContext getAccessibleContext( )	Returns the accessibility context for the invoking object.
AppletContext getAppletContext( )	Returns the context associated with the applet.
String getAppletInfo( )	Overrides of this method should return a string that describes the applet. The default implementation returns **null**.
AudioClip getAudioClip(URL *url*)	Returns an **AudioClip** object that encapsulates the audio clip found at the location specified by *url*.
AudioClip getAudioClip(URL *url*, String *clipName*)	Returns an **AudioClip** object that encapsulates the audio clip found at the location specified by *url* and having the name specified by *clipName*.
URL getCodeBase( )	Returns the URL associated with the invoking applet.
URL getDocumentBase( )	Returns the URL of the HTML document that invokes the applet.
Image getImage(URL *url*)	Returns an **Image** object that encapsulates the image found at the location specified by *url*.
Image getImage(URL *url*, String *imageName*)	Returns an **Image** object that encapsulates the image found at the location specified by *url* and having the name specified by *imageName*.
Locale getLocale( )	Returns a **Locale** object that is used by various locale-sensitive classes and methods.
String getParameter(String *paramName*)	Returns the parameter associated with *paramName*. **null** is returned if the specified parameter is not found.

Method	Description
String[ ] [ ] getParameterInfo( )	Overrides of this method should return a **String** table that describes the parameters recognized by the applet. Each entry in the table must consist of three strings that contain the name of the parameter, a description of its type and/or range, and an explanation of its purpose. The default implementation returns **null**.
void init( )	Called when an applet begins execution. It is the first method called for any applet.
boolean isActive( )	Returns **true** if the applet has been started. It returns **false** if the applet has been stopped.
boolean isValidateRoot( )	Returns **true**, which indicates that an applet is a validate root.
static final AudioClip newAudioClip(URL *url*)	Returns an **AudioClip** object that encapsulates the audio clip found at the location specified by *url*. This method is similar to **getAudioClip( )** except that it is static and can be executed without the need for an **Applet** object.
void play(URL *url*)	If an audio clip is found at the location specified by *url*, the clip is played.
void play(URL *url*, String *clipName*)	If an audio clip is found at the location specified by *url* with the name specified by *clipName*, the clip is played.
void resize(Dimension *dim*)	Resizes the applet according to the dimensions specified by *dim*. **Dimension** is a class stored inside **java.awt**. It contains two integer fields: **width** and **height**.
void resize(int *width*, int *height*)	Resizes the applet according to the dimensions specified by *width* and *height*.
final void setStub(AppletStub *stubObj*)	Makes *stubObj* the stub for the applet. This method is used by the run-time system and is not usually called by your applet. A *stub* is a small piece of code that provides the linkage between your applet and the browser.
void showStatus(String *str*)	Displays *str* in the status window of the browser or applet viewer. If the browser does not support a status window, then no action takes place.
void start( )	Called by the browser when an applet should start (or resume) execution. It is automatically called after **init( )** when an applet first begins.
void stop( )	Called by the browser to suspend execution of the applet. Once stopped, an applet is restarted when the browser calls **start( )**.

The Applet Class

An Applet Skeleton

- Four of these methods, **init( )**, **start( )**, **stop( )**, and **destroy( )**, apply to all applets and are defined by **Applet**.
- Default implementations for all of these methods are provided.
- Applets do not need to override those methods they do not use.
- However, only very simple applets will not need to define all of them.
- AWT-based applets will also often override the **paint( )** method, which is defined by the AWT **Component** class.
- This method is called when the applet’s output must be redisplayed.

```
// An Applet skeleton.
import java.awt.*;
import java.applet.*;
/*
<applet code="AppletSkel" width=300 height=100>
</applet>
*/

public class AppletSkel extends Applet {
  // Called first.
  public void init() {
    // initialization
  }

  /* Called second, after init().  Also called whenever
     the applet is restarted. */
  public void start() {
    // start or resume execution
  }

  // Called when the applet is stopped.
  public void stop() {
    // suspends execution
  }

  /* Called when applet is terminated.  This is the last
     method executed. */
  public void destroy() {
    // perform shutdown activities
  }

  // Called when an applet's window must be restored.
  public void paint(Graphics g) {
    // redisplay contents of window
  }
}
```

Applet Initialization and Termination

- It is important to understand the order in which the various methods shown in the skeleton are called.
- When an applet begins, the following methods are called, in this sequence:
  1. **init( )**
  2. **start( )**
  3. **paint( )**
- When an applet is terminated, the following sequence of method calls takes place:
  1. **stop( )**
  2. **destroy( )**

  Let’s look more closely at these methods.

Applet Initialization and Termination

- **init( )**
  - The **init( )** method is the first method to be called.
  - This is where you should initialize variables.
  - This method is called only once during the run time of your applet.
- **start( )**
  - The **start( )** method is called after **init( )**.
  - It is also called to restart an applet after it has een stopped.
  - Whereas **init( )** is called once—the first time an applet is loaded—**start( )** is called each time an applet's HTML document is displayed onscreen.
  - So, if a user leaves a web page and comes back, the applet resumes execution at **start( )**.

Applet Initialization and Termination

- **paint( )**
  - The **paint( )** method is called each time an AWT-based applet's output must be redrawn.
  - This situation can occur for several reasons.
  - For example, the window in which the applet is running may be overwritten by another window and then uncovered.
  - Or the applet window may be minimized and then restored.
  - **paint( )** is also called when the applet begins execution.
  - Whatever the cause, whenever the applet must redraw its output, **paint( )** is called.
  - The **paint( )** method has one parameter of type **Graphics**.
  - This parameter will contain the graphics context, which describes the graphics environment in which the applet is running.
  - This context is used whenever output to the applet is required.

Applet Initialization and Termination

- **stop( )**
  - The **stop( )** method is called when a web browser leaves the HTML document containing
    the applet—when it goes to another page, for example.
  - When **stop( )** is called, the applet is probably running.
  - You should use **stop( )** to suspend threads that don't need to run when
    the applet is not visible.
  - You can restart them when **start( )** is called if the user returns to the page.
- **destroy( )**
  - The **destroy( )** method is called when the environment determines that your applet needs
    to be removed completely from memory.
  - At this point, you should free up any resources the applet may be using.
  - The **stop( )** method is always called before **destroy( )**.

Simple Applet Display Methods

- void drawString(String *message*, int *x*, int *y*)
- Here, *message* is the string to be output beginning at *x,y*.
- In a Java window, the upper-left corner is location 0,0.
- The **drawString( )** method will not recognize newline characters.
- If you want to start a line of text on another line, you must do so manually, specifying the precise X,Y location where you want the line to begin.
- To set the background color of an applet's window, use **setBackground( )**.
- To set the foreground color use **setForeground( )**.
- These methods are defined by **Component**, and they have the following general forms:
  void setBackground(Color *newColor*)
  void setForeground(Color *newColor*)

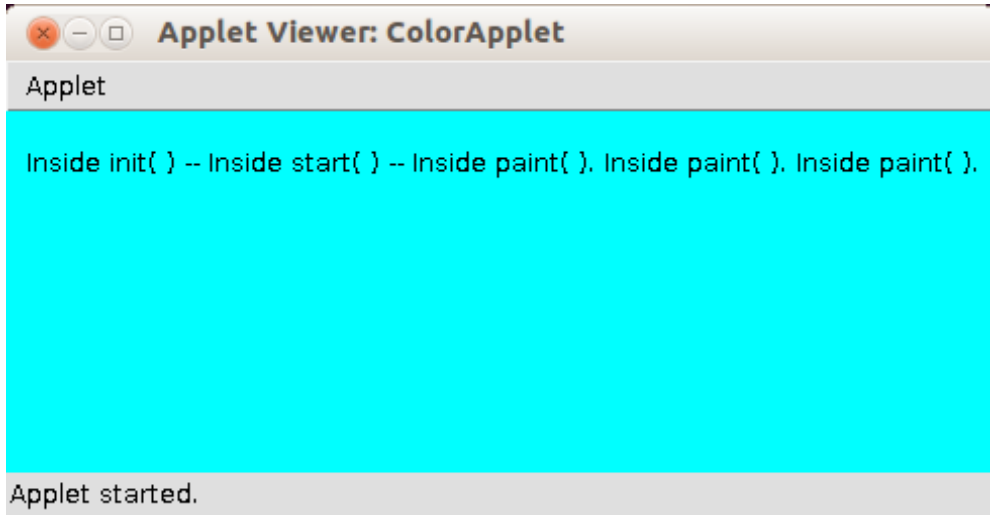
Simple Applet Display Methods

Color.black	Color.magenta
Color.blue	Color.orange
Color.cyan	Color.pink
Color.darkGray	Color.red
Color.gray	Color.white
Color.green	Color.yellow
Color.lightGray	

```
setBackground(Color.green);
setForeground(Color.red);
```

Example #2

```
/* A simple applet that sets the foreground and
   background colors and outputs a string. */
import java.awt.*;
import java.applet.*;
/*
<applet code="Sample" width=300 height=50>
</applet>
*/

public class Sample extends Applet{
  String msg;

  // set the foreground and background colors.
  public void init() {
    setBackground(Color.cyan);
    setForeground(Color.red);
    msg = "Inside init( ) --";
  }

  // Initialize the string to be displayed.
  public void start() {
    msg += " Inside start( ) --";
  }

  // Display msg in applet window.
  public void paint(Graphics g) {
    msg += " Inside paint( ).";
    g.drawString(msg, 10, 30);
  }
}
```



Requesting Repainting

- Whenever your applet needs to update the information displayed in its window, it simply calls **repaint( ).**
- It causes the AWT run-time system to execute a call to your applet's **update( )** method, which, in its default implementation, calls **paint( )**.
- Thus, for another part of your applet to output to its window, simply store the output and then call **repaint( )**.

  void repaint( ) – Simplest Form

  void repaint(int *left*, int *top*, int *width*, int *height*) – Dimensions are specified in pixels.

  void repaint(long *maxDelay*) --

  void repaint(long *maxDelay*, int *x*, int *y*, int *width*, int *height*)

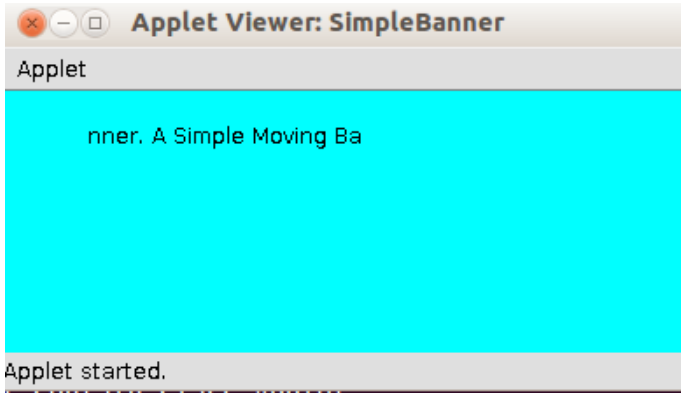
Here, *maxDelay* specifies the maximum number of milliseconds that can elapse before **update( )** is called.

A Simple Banner Applet – Example#3

```
/* A simple banner applet.

   This applet creates a thread that scrolls
   the message contained in msg right to left
   across the applet's window.
*/
import java.awt.*;
import java.applet.*;
/*
<applet code="SimpleBanner" width=300 height=50>
</applet>
*/

public class SimpleBanner extends Applet implements Runnable {
  String msg = " A Simple Moving Banner.";
  Thread t = null;
  int state;
  volatile boolean stopFlag;

  // Set colors and initialize thread.
  public void init() {
    setBackground(Color.cyan);
    setForeground(Color.red);
  }
```

A Simple Banner Applet – Example#3

```
// Start thread
public void start() {
  t = new Thread(this);
  stopFlag = false;
  t.start();
}

// Entry point for the thread that runs the banner.
public void run() {

  // Redisplay banner
  for( ; ; ) {
    try {
      repaint();
      Thread.sleep(250);
      if(stopFlag)
        break;
    } catch(InterruptedException e) {}
  }
}
```




```
// Pause the banner.
public void stop() {
  stopFlag = true;
  t = null;
}

// Display the banner.
public void paint(Graphics g) {
  char ch;

  ch = msg.charAt(0);
  msg = msg.substring(1, msg.length());
  msg += ch;

  g.drawString(msg, 50, 30);
  }
}
```