Object Oriented Programming using Java

Prepared By:
Suyel, PhD
Assistant Professor
Dept. of CSE, NIT Patna

Outline

1. Basics of Applet
2. Applet Class
3. Applet Architecture
4. An Applet Skeleton
5. Applet Display Method
6. Requesting Repainting
7. Using the Status Window
8. HTML APPLET Tag
9. Passing Parameters to Applet
10. getDocumentBase( ) and getCodeBase( )
11. AppletContext and showDocument( )

Basics of Applet

- Applets are not stand-alone programs. They run within either a web browser or an applet viewer (appletviewer).
- Applet class contained in the java.applet package.
- There are two types of Applet:
  - **Abstract Window Toolkit (AWT):** The first type is directly based on the Applet class. These applets use the AWT to provide Graphical User Interface (GUI).
  - **Swing:** These applets are those based on the Swing class JApplet. Swing applets use the Swing classes to provide GUI.
- As JApplet inherits Applet, all the features of Applet are also available in Japplet.

Basics of Applet (Cont...)

- Execution of an applet does not begin at main( ). However, few applets even have main( ) methods.
- Output to the applet's window is not performed by System.out.println( ).
- In non-Swing applets, 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 a console application.

Basics of Applet (Cont...)

- To use an applet, we write some HTML codes. One way to do this is by using the APPLET tag.
- Applet is executed by a Java-enabled web browser, when it encounters the APPLET tag within the HTML file. Here, an example of such a comment is given below:

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

The above mentioned comment contains an APPLET tag that runs an applet called MyApplet in a window that is 200 pixels wide and 60 pixels high.

Applet Class

- This class provides all necessary supports for applet execution, such as starting and stopping.
- It also provides methods that load and display images, and methods that load and play audio clips.
- Applet extends the AWT class Panel. In turn, Panel extends Container, which extends Component. These classes provide support for Java's window-based graphical interface.
- **Some methods** of Applet class mentioned in the next slide.

Applet Class (Cont...)

Method	Description
void destroy( )	Called by the browser just before an applet is terminated. Applet overrides this method, if it needs to perform any cleanup prior to its destruction.
void init( )	Called, when an applet begins execution. It is the first method called for any applet.
void start( )	Called by the browser, when an applet should start or resume execution. It is automatically called after init( ).
void stop( )	Called by the browser to suspend execution of the applet. Once stopped, an applet is restarted, when the browser calls start( ).
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*.

Applet Architecture

- An applet is a window-based program and its architecture is different from the console-based programs for the following two reasons:
  - First, applets are event driven.
    - An applet waits until an event occurs.
    - The run-time system notifies the applet about an event by calling an event handler, which is provided by the applet. Once this happens, the applet take proper action and return control to the run time system.
  - Second, the user initiates interaction with an applet.
    - We can interact with the applet as we want, when we want.
    - These interactions are sent to the applet as events to which the applet must respond.

An Applet Skeleton

- Applets override a set of methods that provide the basic mechanism by which the browser or applet viewer interfaces to the applet and controls its execution.
- Four of these methods, init( ), start( ), stop( ) and destroy( ) apply to all applets and are defined by Applet.
- Applets do not need to override those methods they do not use.
- Many AWT-based applets override the paint( ) method, which is defined by the AWT Component class. This method is called, when the applet's output must be redisplayed.
- These five methods are presented in the **next slide.**

An Applet Skeleton (Cont...)

```
import java.awt.*;
import java.applet.*;

/*
<applet code="AppletSkel" width=300 height=100>
</applet>
*/

public class AppletSkel extends Applet
{
  public void init()   //Called first
  {
                       //Initialization
  }
  public void start()  /*Called second, after init(). Also called
                         whenever the applet is restarted */
  {
                      //Start or resume execution
  }
  public void stop()  //Called when the applet is stopped
  {
                      //suspends execution
  }
  public void destroy() /* Called when applet is terminated. This is
                           the last method executed */

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

An Applet Skeleton (Cont...)

- **Output**

An Applet Skeleton (Cont...)

- **A Simple Applet Program**

```
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);
  }
}
```

An Applet Skeleton (Cont...)

- **A Simple Applet Program (Cont…)**
  - **Output**

Applet Display Method

- To output a string to an applet, we use drawString( ) method that is a member of the Graphics class.
- Typically, drawString() is called from within either update( ) or paint( ) method. It has the following general form:

  void drawString(String *message*, int *x*, int *y*)

- drawString( ) does not recognize newline characters.
- To set the background color of an applet's window, we use setBackground( ), and for the foreground color, we use setForeground( ). These methods are defined by Component:

  void setBackground(Color *newColor*) //**Ex:** setBackground(Color.green);

  void setForeground(Color *newColor*) //**Ex:** setForeground(Color.red);

Applet Display Method (Cont…)

- We can obtain the background and foreground colors by calling getBackground( ) and getForeground( ), respectively.

- They are also defined by Component and are shown here:

Color getBackground( )

Color getForeground( )

Applet Display Method (Cont…)

```
import java.awt.*;
import java.applet.*;

/*
<applet code="Sample" width=300 height=50>
</applet>
*/

public class Sample extends Applet
{
  String msg;

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

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

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

Applet Display Method (Cont…)

- **Output**

Requesting Repainting

- repaint( ) method is defined by the AWT and it executes a call to the applet's update( ) method, which in its default implementation, calls paint( ).
- To get output for a part of our applet to its window, simply store the output, and then, call repaint( ).
- AWT then call paint( ) to display the stored information.
- repaint( ) method has four forms:

```
repaint( )                                          //The entire window is repainted
void repaint(int left, int top, int width, int height)     //Specify
void repaint(long maxDelay)      //Milliseconds to elapse before update( )
void repaint(long maxDelay, int x, int y, int width, int height)
```

Requesting Repainting (Cont..)

```
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;
  boolean stopFlag;
  public void init()     //Set colors and initialize thread
  {
    setBackground(Color.cyan);
    setForeground(Color.red);
  }
  public void start()    //Start thread
  {
    t = new Thread(this);
    stopFlag = false;
    t.start();
  }
  public void run()    //Entry point for the thread that runs the banner
  {
    char ch;
    for( ; ; )         // Display banner
    {
      try {
        repaint();
        Thread.sleep(250);
        ch = msg.charAt(0);
        msg = msg.substring(1, msg.length());
        msg += ch;
        if(stopFlag)
          break;
      }catch(InterruptedException e) {}
    }
  }
  public void stop()    //Pause the banner
  {
    stopFlag = true;
    t = null;
  }
  public void paint(Graphics g)  //Display the banner
  {
    g.drawString(msg, 50, 30);
  }
}
```

Requesting Repainting (Cont...)

- **Output**

Using the Status Window

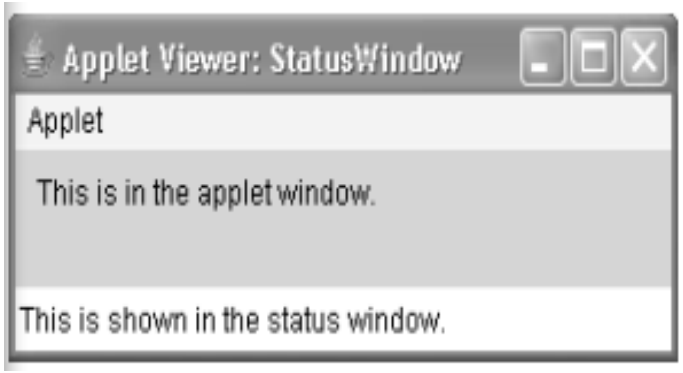
- An applet can also output a message to the status window of the browser or applet viewer on which it is running.
- We can call showStatus( ) with the string that we want to display.
- The status window is a good place to give the user feedback about what is occurring in the applet, suggest options, etc.
- The status window also makes an excellent debugging aid as it gives an easy way to output information about the applet.

Using the Status Window (Cont...)

```
import java.awt.*;
import java.applet.*;

/*
<applet code="StatusWindow" width=300 height=50>
</applet>
*/

public class StatusWindow extends Applet
{
  public void init()
  {
    setBackground(Color.cyan);
  }

  public void paint(Graphics g)   //Display msg in applet window
  {
    g.drawString("This is in the applet window.", 10, 20);
    showStatus("This is shown in the status window.");
  }
}
```

Using the Status Window (Cont...)

- **Output**



HTML APPLET Tag

- An applet viewer can execute each APPLET tag that it finds in a separate window, while web browsers allow many applets on a single page.
- The syntax for a fuller form of the APPLET tag is shown here. Here, bracketed items are optional.

```
< APPLET
  [CODEBASE = codebaseURL]
  CODE = appletFile
  [ALT = alternateText]
  [NAME = appletInstanceName]
  WIDTH = pixels HEIGHT = pixels
  [ALIGN = alignment]
  [VSPACE = pixels] [HSPACE = pixels]
>
[< PARAM NAME = AttributeName VALUE = AttributeValue>]
[< PARAM NAME = AttributeName2 VALUE = AttributeValue>]
. . .
[HTML Displayed in the absence of Java]
</APPLET>
```

HTML APPLET Tag (Cont...)

- **CODEBASE:** It is an optional attribute that specifies the base URL of the applet code, which is the directory that is searched for the applet's executable class file. The HTML document's URL directory is used as the CODEBASE, if this attribute is not specified.
- **CODE:** It is a required attribute that gives the name of the file containing our applet's compiled .class file.
- **ALT:** ALT tag is an optional attribute used to display a short text message, if the browser recognizes the APPLET tag, but can't currently run Java applets.
- **NAME:** It is an optional attribute used to specify a name for the applet instance. Applets must be named in order for other applets on the same page to find them by name and communicate with them. To obtain an applet by name, we use getApplet( ) defined by AppletContext interface.

HTML APPLET Tag (Cont...)

- **WIDTH and HEIGHT:** WIDTH and HEIGHT are required attributes that give the size (in pixels) of the applet display area.
- **ALIGN:** ALIGN is an optional attribute that specifies the alignment of the applet. This attribute is treated the same as the HTML IMG tag with the possible values: LEFT, RIGHT, TOP, etc.
- **VSPACE and HSPACE:** These attributes are optional. VSPACE specifies the space in pixels above and below the applet. HSPACE specifies the space in pixels on each side of the applet.
- **PARAM NAME and VALUE:** The PARAM tag allows us to specify applet-specific arguments in an HTML page. Applets access their attributes with the getParameter( ).

  Other valid APPLET attributes include ARCHIVE that allows to specify one or more archive files, and OBJECT that specifies a saved version.

Passing Parameters to Applet

- As just discussed in the previous slide, the APPLET tag in HTML allows us to pass parameters to the applet.

- To retrieve a parameter, we use the getParameter( ) method. It returns the value of the specified parameter in the form of a String object.

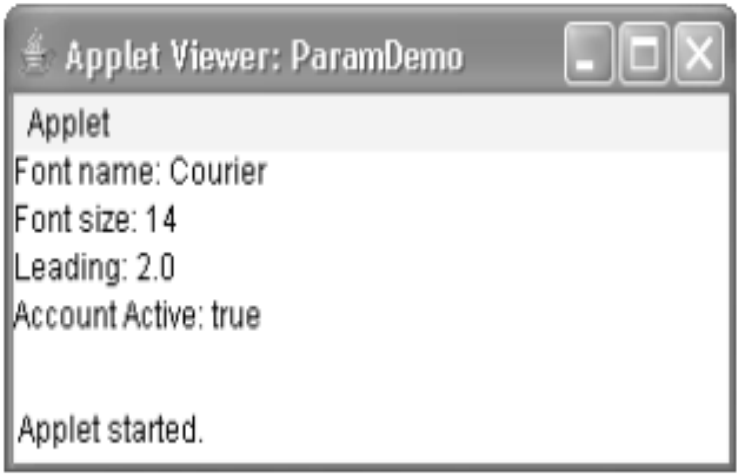
- Therefore, for numeric and boolean values, we need to convert their string representations into their internal formats.

Passing Parameters to Applet (Cont...)

```
import java.awt.*;
import java.applet.*;
/*
<applet code="ParamDemo" width=300 height=80>
<param name=fontName value=Courier>
<param name=fontSize value=14>
<param name=leading value=2>
<param name=accountEnabled value=true>
</applet>
*/
public class ParamDemo extends Apple
{
  String fontName;
  int fontSize;
  float leading;
  boolean active;
  public void start()  //Initialize the string to be displayed.
  {
    String param;
    fontName = getParameter("fontName");
    if(fontName == null)
      fontName = "Not Found";
    param = getParameter("fontSize");
    try {
          if(param != null) // if not found
            fontSize = Integer.parseInt(param);
          else
            fontSize = 0;
    } catch(NumberFormatException e) {
        fontSize = -1;
      }
    param = getParameter("leading");
    try {
          if(param != null) // if not found
            leading = Float.valueOf(param).floatValue();
          else
            leading = 0;
    } catch(NumberFormatException e) {
        leading = -1;
      }
    param = getParameter("accountEnabled");
    if(param != null)
       active = Boolean.valueOf(param).booleanValue();
  }
  public void paint(Graphics g)   //Display parameters.
  {
    g.drawString("Font name: " + fontName, 0, 10);
    g.drawString("Font size: " + fontSize, 0, 26);
    g.drawString("Leading: " + leading, 0, 42);
    g.drawString("Account Active: " + active, 0, 58);
  } }
```

Passing Parameters to Applet (Cont...)

- **Output**



getDocumentBase( ) and getCodeBase( )

- Applet can load data from the directory holding the HTML file that started the applet and the directory from which the applet's class file is loaded.
- These directories are returned as URL objects by getDocumentBase( ) and getCodeBase( ) method.
- They can be concatenated with a string that names the file we want to load.

getDocumentBase( ) and getCodeBase( ) (Cont…)

```
import java.awt.*;
import java.applet.*;
import java.net.*;

/*
<applet code="Bases" width=300 height=50>
</applet>
*/

public class Bases extends Applet
{
  public void paint(Graphics g)   //Display code and document bases
  {
    String msg;
    URL url = getCodeBase();      //Get code base
    msg = "Code base: " + url.toString();
    g.drawString(msg, 10, 20);
    url = getDocumentBase();      //Get document base
    msg = "Document base: " + url.toString();
    g.drawString(msg, 10, 40);
  }
}
```

getDocumentBase( ) and getCodeBase( ) (Cont…)

- **Output**

AppletContext and showDocument( )

- AppletContext gets information from the applet's execution environment.
- The context of the currently executing applet is obtained by a call to the getAppletContext( ) method defined by Applet.
- When we want to transfer control of Applet to another URL, we use the showDocument( ) method defined by the AppletContext interface.
- Within an applet, once we have obtained the applet's context, we can bring another document into view by calling the showDocument( ).

AppletContext and showDocument( ) (Cont...)

- There are two showDocument( ) methods:
  - showDocument(URL): It displays the document at the specified URL.
  - showDocument(URL, String): It displays the specified document at the specified location within the browser window.

- We can also specify a name, which causes the document to be shown in a new browser window by that name.

- **Some methods** defined by AppletContext are mentioned in the next slide.

AppletContext and showDocument( ) (Cont...)

Method	Description
Applet getApplet(String *appletName*)	Returns the applet specified by *appletName*, if it is within the current applet context. Otherwise, null is returned.
AudioClip getAudioClip(URL *url*)	Returns an AudioClip object that encapsulates the audio clip found at the location specified by *url*.
void showDocument(URL *url*)	Brings the document at the URL specified by *url* into view. This method may not be supported by applet viewers.
void showDocument(URL *url*, String *where*)	Brings the document at the URL specified by *url* into view. This method may not be supported by applet viewers. The placement of the document is specified by where as described in the text (The *where* argument indicates where to display the frame).
void showStatus(String *str*)	Displays *str* in the status window.

AppletContext and showDocument( ) (Cont...)

```
import java.awt.*;
import java.applet.*;
import java.net.*;

/*
<applet code="ACDemo" width=300 height=50>
</applet>
*/

public class AppletContext extends Applet
{
  public void start()
  {
    AppletContext ac = getAppletContext();
    URL url = getCodeBase();   //Get url of this applet
    try {
          ac.showDocument(new URL(url+"Test.html"));
    } catch(MalformedURLException e) {
          showStatus("URL not found");
      }
  }
}
```


thank
you . . .

Slides are prepared from Book