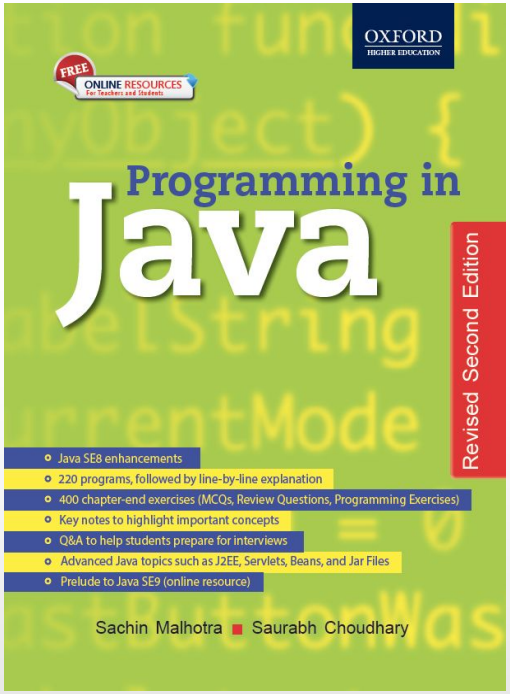
OXFORD
UNIVERSITY PRESS

Programming in Java

Revised 2nd Edition

Sachin Malhotra & Saurabh Choudhary



Chapter 12

Applets

Objectives

- Understand the difference between applet and application
- Understand the lifecycle of an applet
- Learn how applets are created and executed
- Create shapes within applets
- Use images in applets
- Use threads in applet and create a digital clock

Introduction

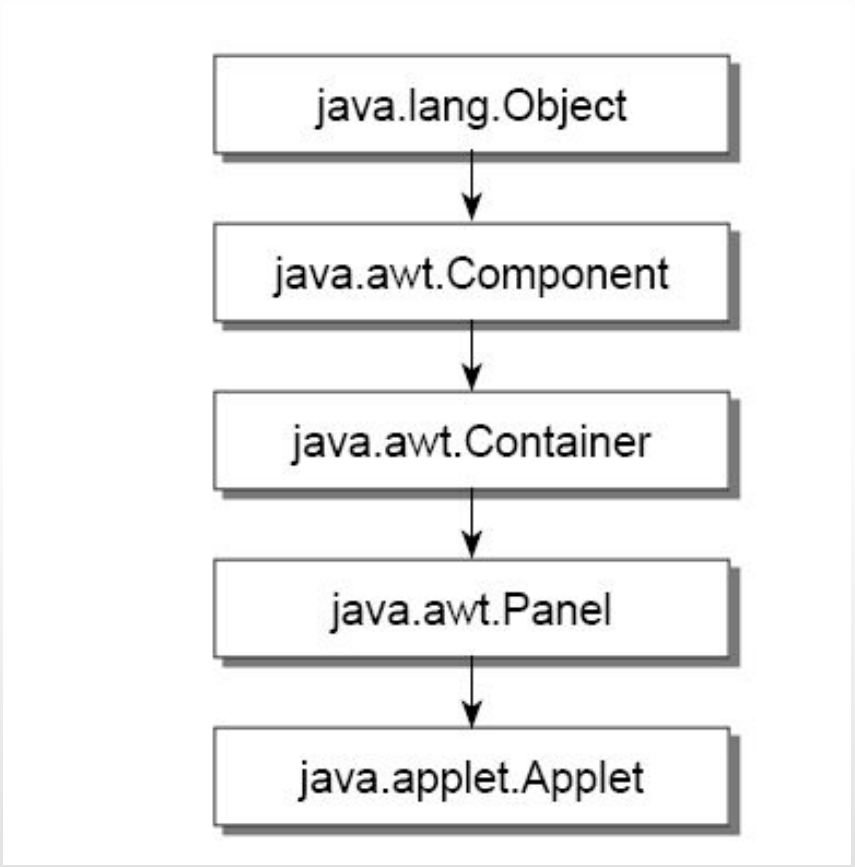
- Applet is small java program, which can be easily transported over the network from one computer to other.
- Used in internet applications.
- Embedded in an html page, can be downloaded from the server and run on the client, so as to do a specific kind of job.

Difference between Applets and Applications

Applet	Application
The execution of the applet does not start from `main()` method, as it does not have one.	The execution of an application program starts from `main()`.
Applets cannot run on their own. They have to be embedded inside a web page to get executed.	These can run on their own. In order to get executed, they need not be embedded inside any web page.
Applets can only be executed inside a browser or appletviewer.	Applications are executed at command line.
Applets execute under strict security limitations that disallow certain operations (sandbox model security).	Applications have no inherent security restrictions.
Applets have their own life cycle `init()→start()→paint()→stop→destroy()`	Applications have their own life cycle. Their execution begins at `main()`.

Applet Class

- **java.applet.Applet** is the super class of all the applets.
- Applet class has a predefined hierarchy



Few Methods of Applet Class

Method	Description
`public void init()`	First method to be called when an applet begins execution.
`public boolean isActive()`	Returns true if the applet is running, otherwise false.
`public URL getDocumentBase()`	Gets the URL of the document in which this applet is embedded.
`public URL getCodeBase()`	Returns the URL of the directory where the class file of the invoking applet exists.
`public String getParameter(String name)`	Return the value of the parameter associated with parameter's name. Null is returned if the parameter is not specified.
`public AppletContext getAppletContext()`	Determines this applet's context, which allows the applet to query and affect the environment in which it runs.
`public void resize (int width, int height)`	Resizes the applet according to the parameters, *width* and *height*.
`public void showStatus(String msg)`	Displays the string, *msg*, in the status window of the browser or appletviewer (only if they support status window).
`public Image getImage(URL url)`	Returns an object of Image, which binds the image found at the *url*, specified as the argument of the method.

Applet Structure

```
import java.awt.*;
//so as to make Graphics class (which is part of this package) available
import java.applet.*;
//so as to make Applet class (which is part of this package) available
……
……
public class NewApplet extends Applet
// new applet with the name, newApplet declared
{
 ……....
 ……..
 public void paint(Graphics g)
 // paint() of Applet class overridden to contain output operations
{
……..
……
}
……
……}
```

Applet Example

```
import java.applet.*;

import java.awt.*;

public class FirstApplet extends Applet {

public void paint(Graphics g) {

g.drawString(“This is my first applet program”, 10,20);

}

}
```

Running an Applet

- There are two ways to run the applet.

1. Add the applet tag within the body of the html tag

```
<HTML><BODY>
<APPLET code = "FirstApplet.class" WIDTH = 200 HEIGHT =
150></APPLET>
</BODY></HTML>
```

You can execute the HTML file by giving *appletviewer FirstApplet.html*

Running an Applet (contd.)

2. Add the Applet tag as a comment in the java source file

In order to run the applet You have to give the below HTML coding as a comment in the source file :

```
/* <APPLET code = "FirstApplet.class" WIDTH = 200 HEIGHT = 150></APPLET> */
```

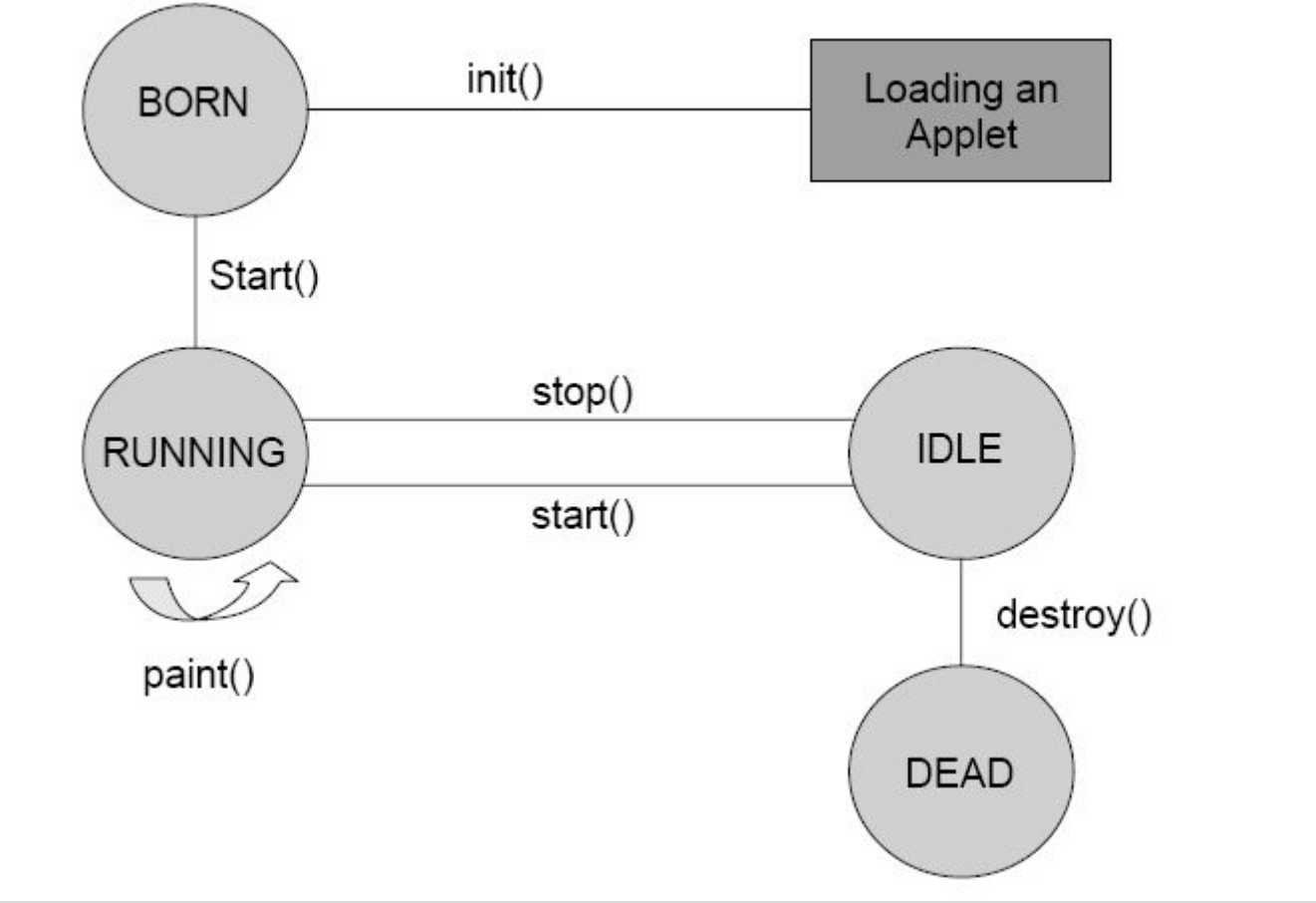
Execute the applet as: appletviewer FirstApplet.java

The Output

Applet Life cycle

- An applet may move from one state to another depending upon a set of default behaviour inherited in the form of methods from ‘Applet’ class.
- These states are
  - Born
  - Running
  - Idle
  - Dead

Applet State Diagram



Life cycle of Applet

- init() –
  - creates the objects needed by the applet
  - sets up initial values, loads font and images, or sets up colors
  - called only once during the lifetime of an Applet
- start()
  - moves to this phase automatically after the initialization state

  -if the applet is stopped or it goes to idle state, start() method must be called in order to force the applet again to the running state
- paint()
  - This method is called each time to draw and redraw the output of an applet
- stop()
  - idle state, once it is stopped from running
- destroy()

  -An applet goes to dead state when it is destroyed by invoking the destroy() method of Applet class
  - It results in complete removal of applet from the memory

Common Methods

- drawString()*:*
  - member of Graphics class, used to output a string to an applet
  - It is typically called from within the **paint()** or **update()** method
  - void drawString(String msg,int a, int b)
- setBackground() & getBackground()
  - belongs to Component class, used to set and get the background color
  - void setBackground(Color anyColor)
  - predefined constants for each color, such as Color.red can be used
- setForeground() & get Foreground()
  - set and gets the color of the text to be displayed on the foreground of the applet window
  - void setForeground(Color anyColor)
- showStatus()
  - display any string in the status window of the browser
  - void showStatus(String text)

Example

```
/* <APPLET code = “ExampleApplet.class” WIDTH = 200 HEIGHT = 150></APPLET>
*/
import java.applet.Applet;
import java.awt.Color;
import java.awt.Graphics;
public class ExampleApplet extends Applet{
String text;
public void init() {
setBackground(Color.white);
setForeground(Color.red);
text = “This is an example applet”;
System.out.println(“....Initialized the applet”);}
public void start() {
System.out.println(“....Starting of the applet”);}
```

Example (contd.)

```
public void stop() {
System.out.println(“....Stopping the applet”);}
public void destroy() {
System.out.println(“....Exiting the applet”);
}
public void paint(Graphics g) {
System.out.println(“....Painting the applet”);
g.drawString(text, 30, 30);
showStatus(“This is status bar”);
}}
```

The Output

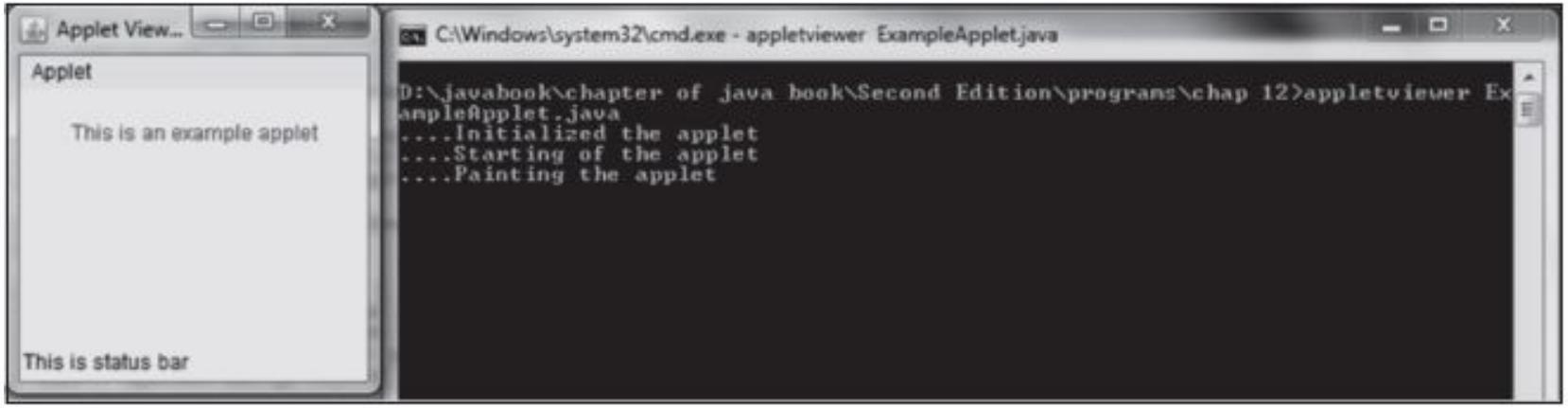


Fig. 12.4(a) Applet Initialized Using Applet Viewer

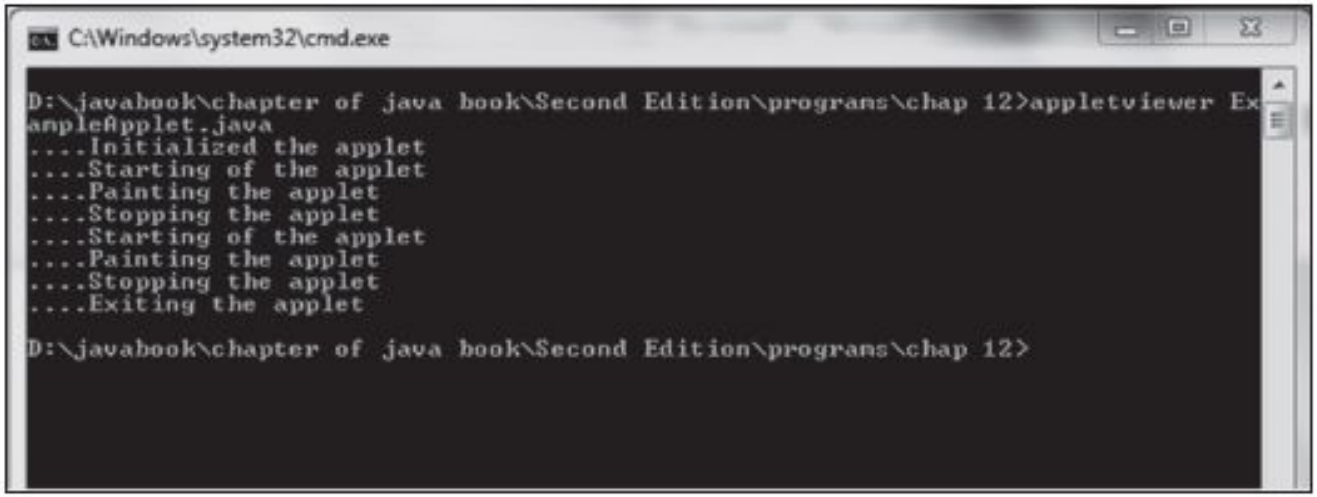


Fig. 12.4(b) Strings Printed by Various Methods of the Applet

Paint, Update and Repaint

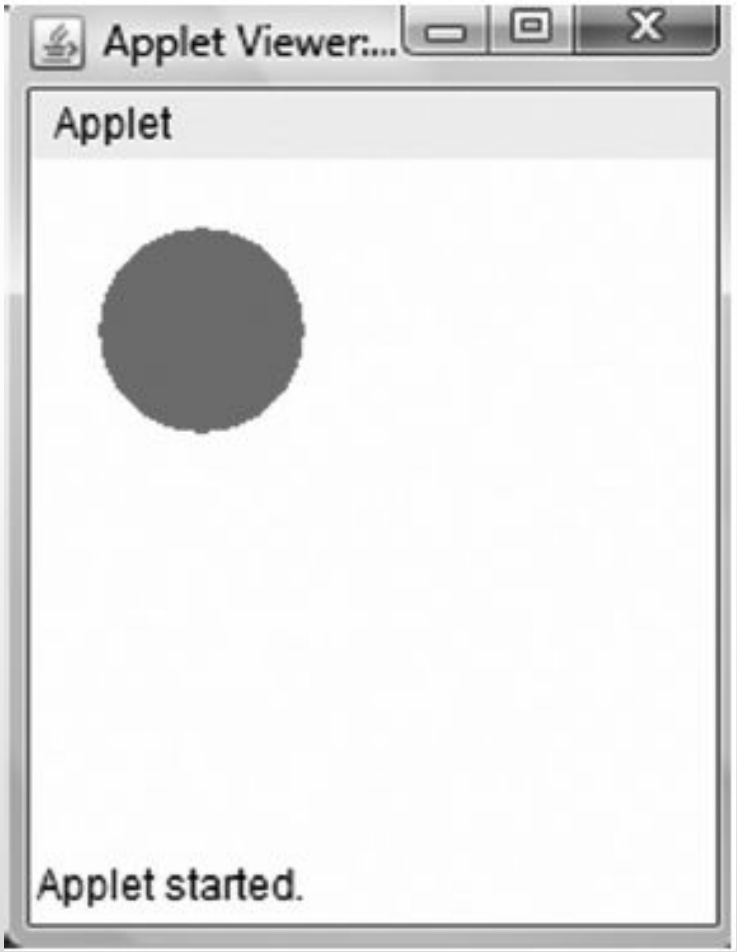
- All components and containers in the JDK have two methods that are called by the system to paint their surface.
  - public void paint(Graphics g);
  - public void update(Graphics g);
- If you wish that a drawing should appear in a window, you shall override either or both of the methods.
- paint()
  - for drawing/redrawing **paint()** method is called.
  - The component draws itself when it first becomes visible.

-The component **paint()** method is also invoked when the window containing it is uncovered, if it is covered by another window.

Example

```
/* <APPLET code = “FillOval.class” WIDTH = 200 HEIGHT = 200></APPLET> */
    import java.applet.Applet;
    import java.awt.Color;
    import java.awt.Graphics;
    public class FillOval extends Applet{
    public void paint(Graphics g) {
        g.setColor(Color.red);
        g.fillOval(20, 20, 60, 60);
    }
    }
```

The Output



update() Method

- It clears the surface of the calling component to its background color and then calls the paint() method to paint the rest of the component.
- It makes the job easier because one does not have to draw the whole component within a paint() method, as the background is already filled. Then, when one overrides paint(), he/she only needs to draw what should appear on the foreground.

repaint() Method

- If you have changed certain properties of a component to reflect its new appearance, you can call the repaint() method.
  - text.setBackground(Color.blue);
  - text.repaint();

- Calling the **repaint()** method causes the whole component to be repainted.

- repaint() in its default implementation calls update() which in turn calls paint().

- repaint() method requests the AWT to call update and it returns. The AWT combines multiple rapid repaint requests into one request (usually this happens when you repaint inside a loop). So the last repaint in the sequence actually causes **paint()**.

Applet Tag

```
<APPLET [CODEBASE= codebasedURL ] CODE= appletFile [ALT= alternateText] [NAME =
appletInstanceName] WIDTH = pixels HEIGHT = pixels [ALIGN = alignment] [VSPACE = pixels]
[HSPACE = pixels]>
[<PARAM NAME = attributeName VALUE = attributeValue>]
[<PARAM NAME = attributeName VALUE = attributeValue>]
.....
</APPLET>
```

Applet Tag

- CODEBASE - specifies the URL of the directory where the executable class file of the applet will be searched for.
- CODE - gives the name of the file containing the applet's compiled class file.
- ALT **-** specifies the alternate short text message that should be displayed in case the browser recognizes the HTML tag but cannot actually run the applet because of some reason.
- NAME **-** give a name to an applet's instance.
- WIDTH - gives the width of the applet display area in terms of pixels.
- HEIGHT - gives the height of the applet display area in terms of pixels.
- ALIGN - sets the alignment of an applet. The alignment can be set as LEFT, RIGHT, TOP, BOTTOM, MIDDLE, BASELINE, TEXTTOP, ABSMIDDLE, and ABSBOTTOM.
- VSPACE - used to specify the space, in pixels, above and below the applet.

Applet Tag (contd.)

- HSPACE
  - These are used to specify the space, in pixels, on each side of the applet.
- PARAM sub tag
  - provides information about parameters, or arguments, to be used by the Java applet.
  - The <PARAM> tag is simple—it NAMES a parameter and provides a VALUE for that parameter.
  - This tag has two attributes
    - **NAME:** attribute name
    - **VALUE:** value of the attribute named by corresponding PARAM NAME.
  - The applets access their attributes using the **getParameter()** method.
  - String getParameter(String name);

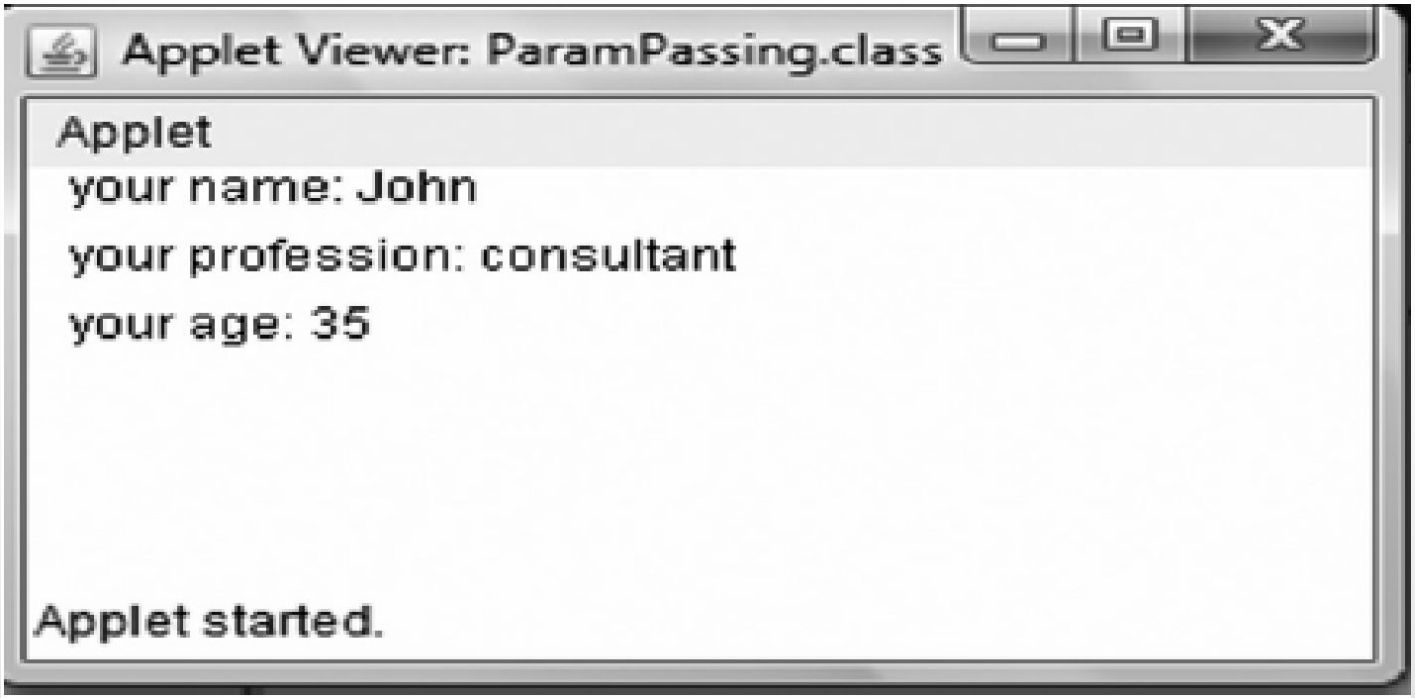
Param Tag Example

```
/*<APPLET CODE = ParamPassing.class WIDTH = 300 HEIGHT = 250>
<param NAME = yourName VALUE = John>
<param NAME = yourProfession VALUE = consultant>
<param NAME = yourAge VALUE = 35>      </applet>*/
import java.awt.*;
import java.applet.*;
public class ParamPassing extends Applet {
String name;     String profession;int age;
public void start() {
String str;
name = getParameter("yourName");
if (name == null)      name = "not found";
str = getParameter("yourProfession");
 if (str != null)          profession = str;
```

Param Tag Example (contd.)

```
else profession = “No job”;
str = getParameter(“yourAge”);
try {
if (str != null)
   age = Integer.parseInt(str);
else age = 0;
} catch (NumberFormatException e) {}
}
public void paint(Graphics g) {
   g.drawString(“your name: “+name, 10, 10);
   g.drawString(“your profession: “+profession, 10, 30);
   g.drawString(“your age: “ +age, 10, 50);
}}
```

The Output



getDocumentBase() and getCodeBase()

- **getDocumentBase()** returns the URL of the directory that holds the HTML file responsible for starting the applet in the form of URL object.
- **getCodeBase()** returns the URL object of the directory from where the class file of the applet is loaded.

AppletContext Interface

- This interface corresponds to an applet's environment: the document containing the applet and other applets in the same document.
- In other words, interface determines applet's context, which allows the applet to query and affect the environment in which it runs.
- This environment of an applet represents the document that contains the applet.
- The methods in this interface can be used by an applet to obtain information about its environment.

Communication Between Applets

- **showDocument()** enables your applet to transfer control to any other URL.
  - public abstract void showDocument(URL url);
  - The url specifies where you want your applet to transfer control to
- You can use this method only with the object of the currently executing applet, which can be obtained by getAppletContext().
- After that, your browser or appletviewer can show another document or web page by using showDocument(url).
  - try {
  - getAppletContext( ).showDocument(new URL(url));
  - } catch (java.net.MalformedURLException e) {
  - System.out.println("URL could not be reached");}
- showDocument() method has one more form
  - public abstract void showDocument(URL url, String target)
  - This argument indicates in which HTML frame the document is to be displayed. The *target* arguments can be used in following forms.

Target Attribute of showDocument()

Target Argument	Description
"_self"	Show in the window and frame that contain the applet, i.e. the current frame.
"_parent"	Show in the applet's parent frame. If the applet's frame has no parent frame, it acts the same as "_self".
"_blank"	Show in a new, unnamed top-level window.
"_top"	Show in the topmost frame of the applet's window. If the applet's frame is the top-level frame, it acts the same as "_self".
Name	You can specify a name which causes the document to be shown in a new browser window by that name.

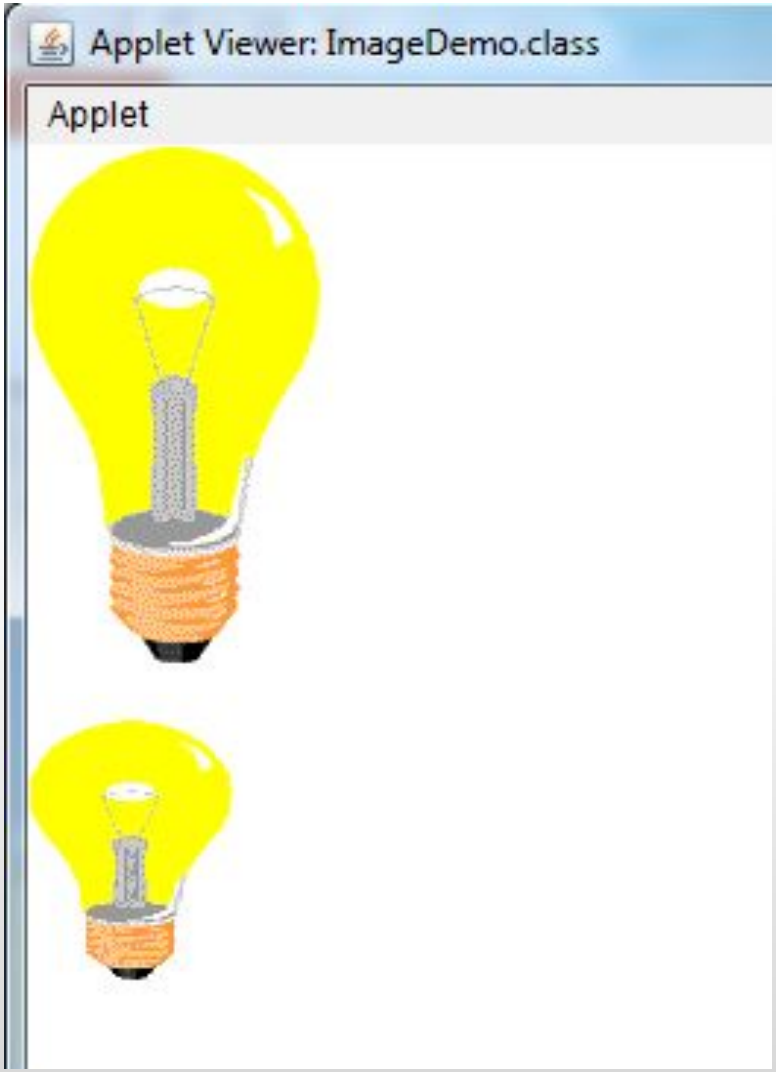
Audio Clip

- You can load a clip using the method getAudioclip() of the Applet class, which returns an AudioClip object. The signature of this method is,
  - AudioClip getAudioClip(URL url)
- This object actually encapsulates the audio clip found at the specified *url,* passed as the argument to the method.
- These methods can be used by the audio clip object (returned by getAudioClip()) to either play, stop, or loop an audio clip.
  - play(): plays a clip from the beginning
  - stop(): stops playing the clip
  - loop(): plays the clip in loop continuously

Images in Applet

```
public class ImageDemo extends Applet{
Image image;
public void init() {
image = getImage(getDocumentBase(),"bulbon.gif");}
public void paint(Graphics g) {
//drawImage(Image img,int x,int y,int width, int height, //
ImageObserver observer)
g.drawImage(image, 0, 0,
image.getWidth(this),image.getHeight(this), this);
g.drawImage(image, 0, 200, 70,90, this); }}
```

The Output



MediaTracker Class

```
import java.applet.Applet;

import java.awt.*;

public class MediaTrackerDemo extends Applet implements
Runnable{

// image array to hold images

Image[] imageArray = null;

// to track the images

MediaTracker m = null;

int current = 0;

Thread t=null;
```

MediaTracker Class

```
public void init(){
m = new MediaTracker(this);
// Create an array of three images
imageArray = new Image[2];
// Start downloading the fi rst images into the image array
imageArray[0] = getImage(getDocumentBase(), "bulboff.gif");
// Register it with media tracker
m.addImage(imageArray[0], 0);
// second image
imageArray[1] = getImage(getDocumentBase(), "bulbon.gif");
m.addImage(imageArray[1], 1);
```

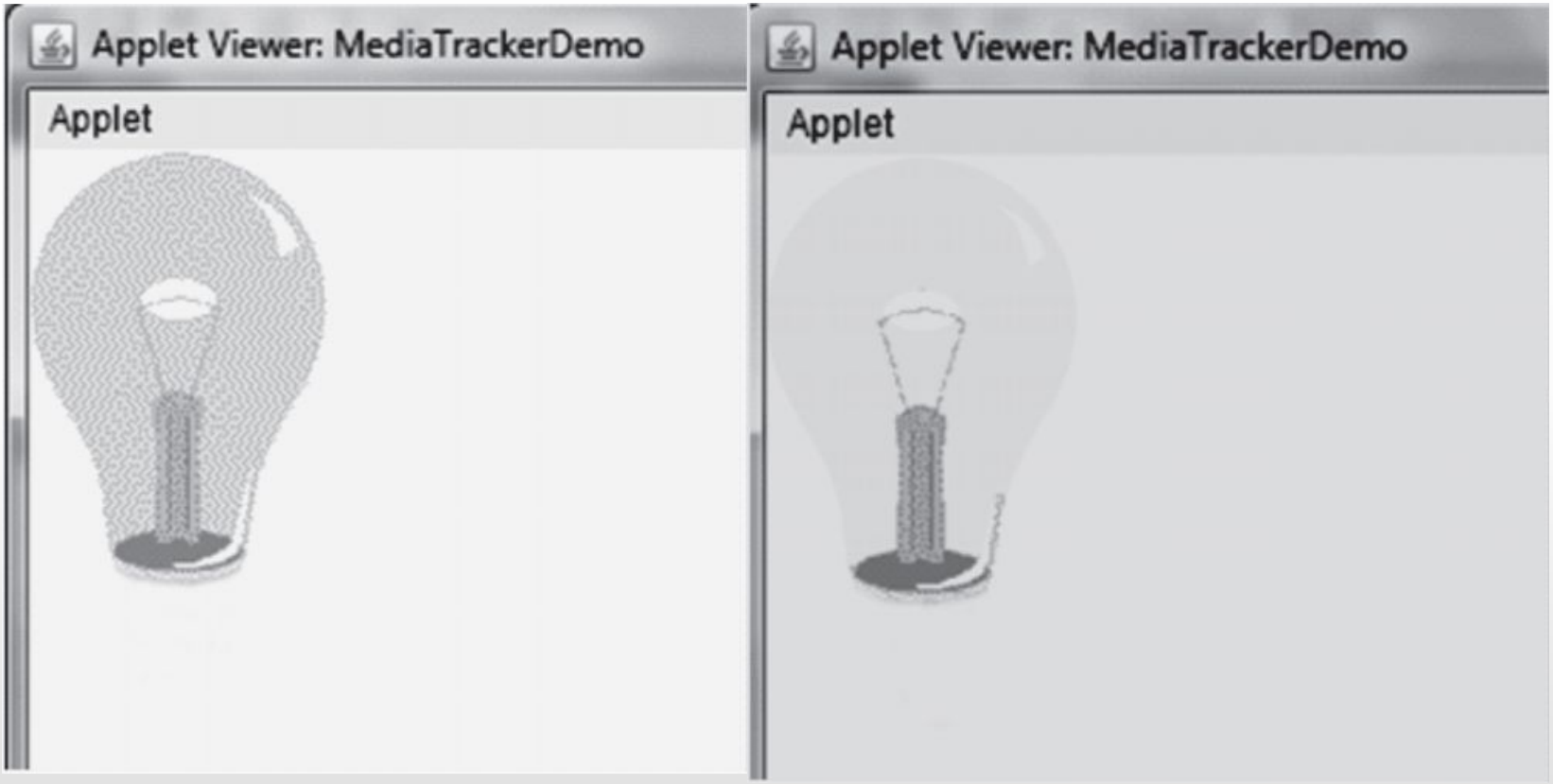
MediaTracker Class

```
// Start thread to begin blinking of images
t = new Thread(this);
t.start();}
public void paint (Graphics g){
g.setColor(Color.white);
// Sets the background with White Color
g.fillRect(0,0, 400, 400);
// Sets the Color as black, so that any text written will be black
g.setColor(Color.black);
// Check to see if images have loaded
if (m.checkAll()){
g.drawImage(imageArray[current++], 0, 0, this);
```

MediaTracker Class

```
if (current >= imageArray.length)
current = 0;}
else {// Still loading
g.drawString ("Images are still loading...", 20,20);}}
public void run(){
try{
// waits until all the images have fi nished loading
m.waitForAll();
for (;;){  // Repaint the images
repaint();
Thread.sleep(2000);}}
catch (InterruptedException ie){}}}
```

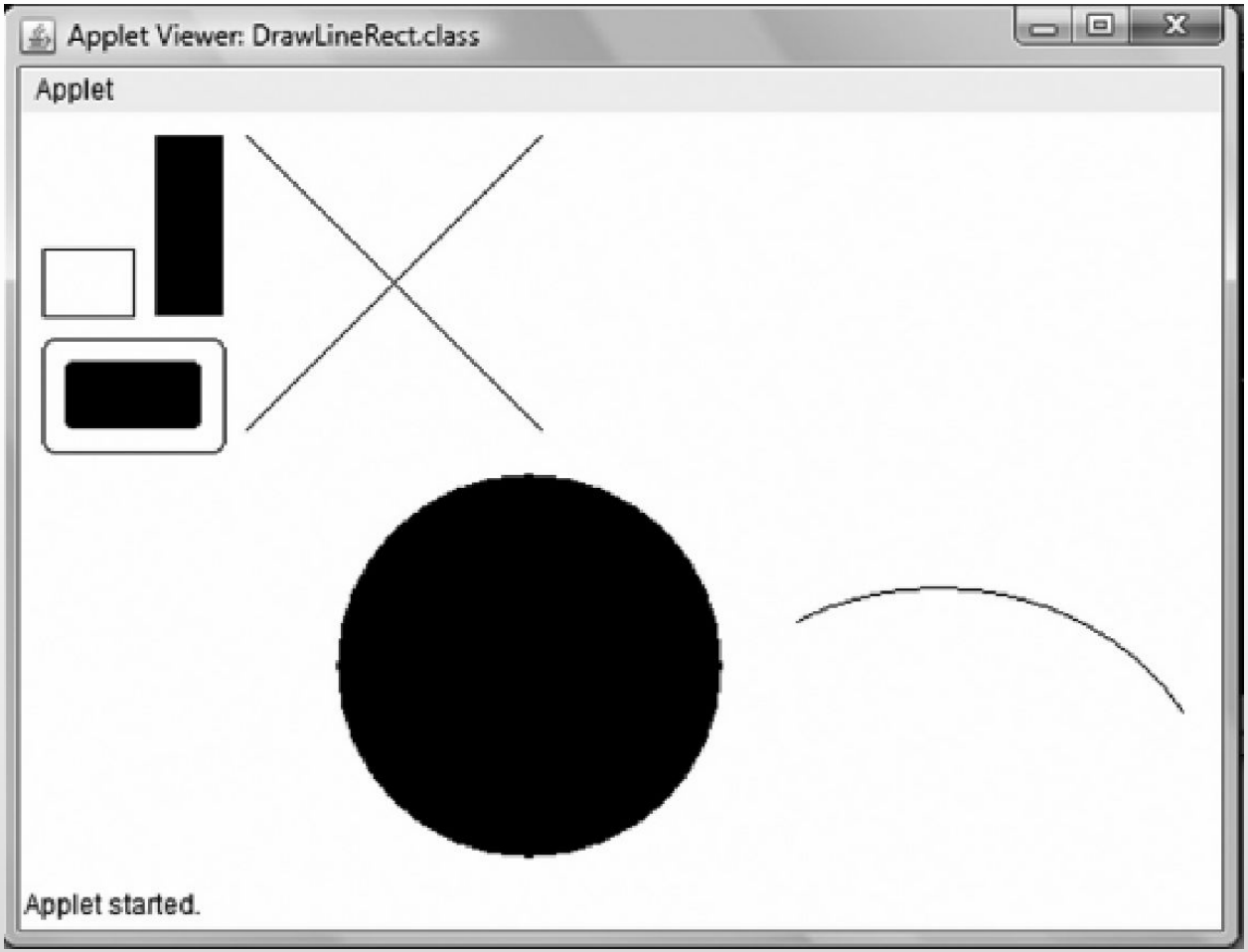
The Output



Graphics Class

```
import java.awt.* ;
   import java.applet.* ;
   public class DrawLineRect extends Applet {
   public void paint(Graphics g){
      g.drawRect(10,60,40,30);
      g.fillRect(60,10,30,80);
      g.fillOval(140,160,170,170);
      g.drawRoundRect(10,100,80,50,10,10);
      g.fillRoundRect(20,110,60,30,5,5);
      g.drawArc(280,210,250,220,30,90);
      g.drawLine(100,10,230,140);
      g.drawLine(100,140,230,10);
   }}
```

The Output



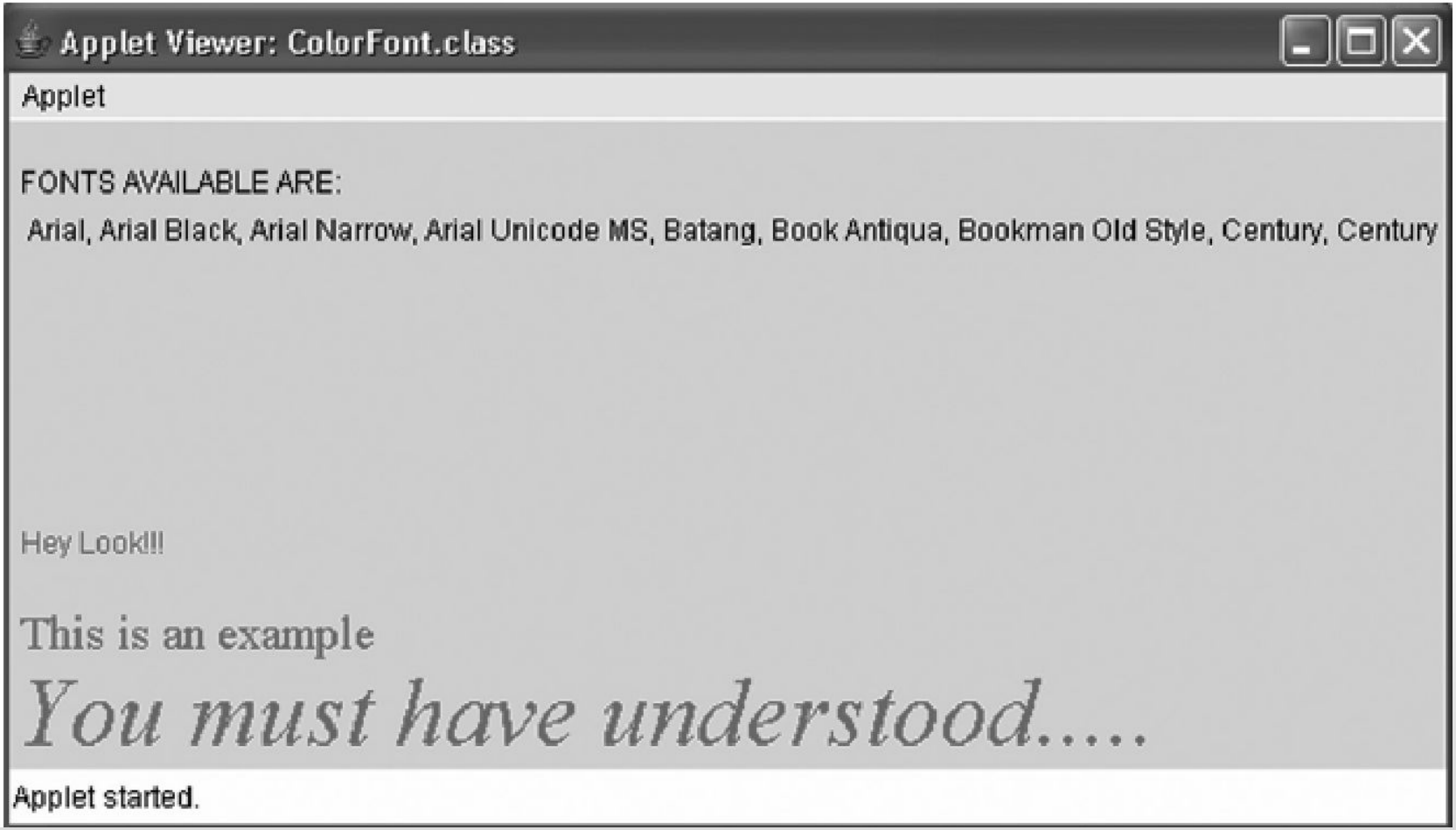
Font and Color Class

```
/*<applet code=ColorFont.class width=600 height = 270 > < / applet >*/
import java.awt.*;
public class ColorFont extends java.applet.Applet{
public void init() {
Color color1 = new Color(230, 220, 0);
setBackground(color1);}
public void paint(Graphics g) {
String str = “”;
String FontList[];
GraphicsEnvironment ge = GraphicsEnvironment.getLocalGraphicsEnvironment();
FontList = ge.getAvailableFontFamilyNames();
for (int i =0; i<FontList.length; i++) {
g.drawString(“FONTS AVAILABLE ARE:”, 5, 30);
str += FontList[i] + “, “;
```

Example (contd.)

```
g.drawString(str,5, 50);}
Color color2 = new Color(235, 50, 50);
g.setColor(color2);
g.drawString("Hey Look!!!", 5, 180);
Font currentFont = new Font("TimesRoman", Font.PLAIN, 20);
g.setFont(currentFont);
g.drawString("This is an example", 5, 220);
currentFont = new Font("TimesRoman", Font.ITALIC, 40);
g.setFont(currentFont);
g.drawString("You must have understood.....", 5, 260);}}
```

The Output



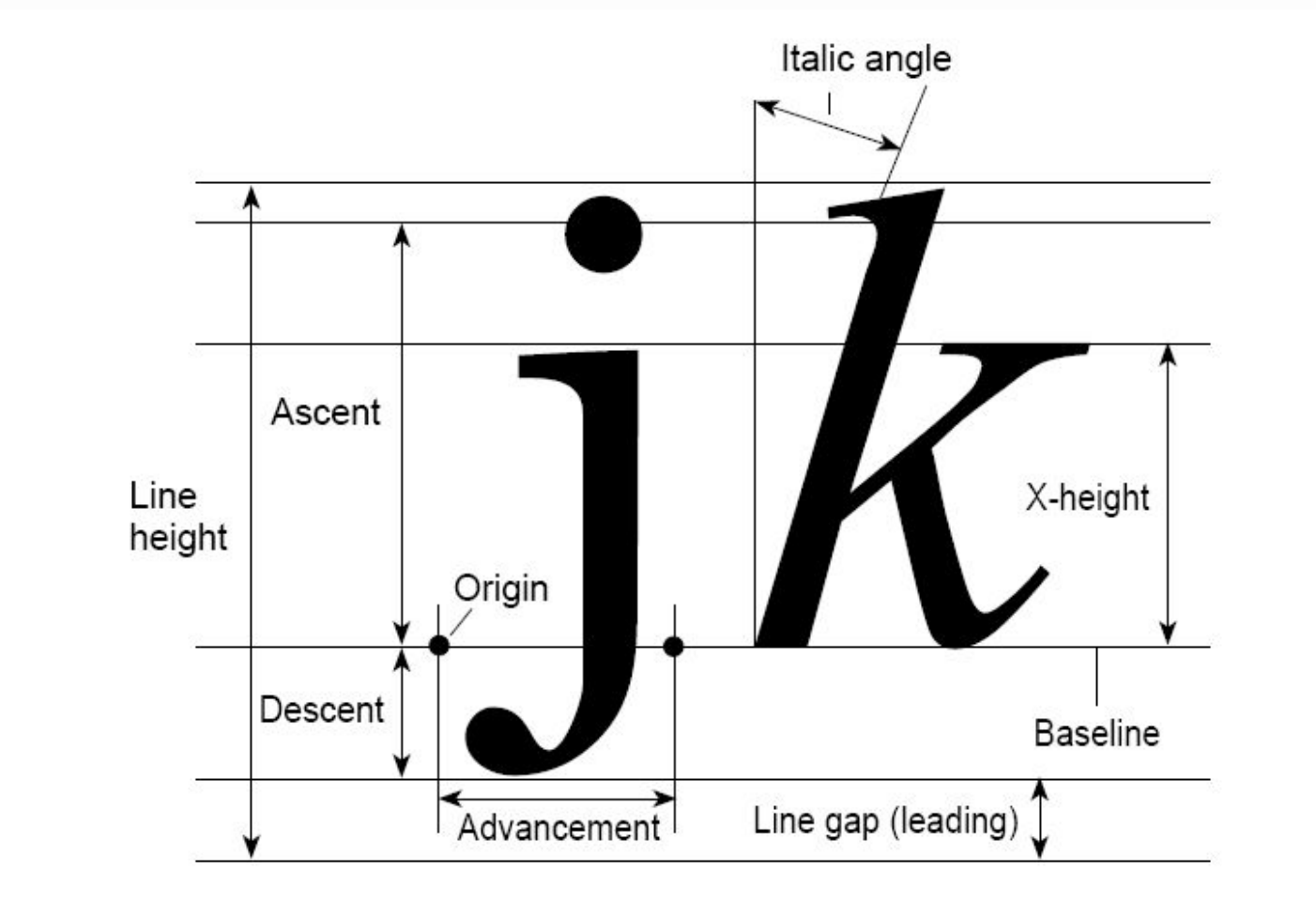
FontMetrics Class

- Font metrics objects are defined by FontMetrics class which extends Object class and implements Serializable interface.
- You must be aware with the term ‘metrics’. Its literal meaning is ‘measurements’.
- The **java.awt.FontMetrics** class is helpful in determining the measurements associated with characters and strings (group of characters).
- The texts can be positioned precisely using these measurements.

FontMetrics Class

Baseline	It is the position at the bottom of characters, i.e. the horizontal line at which the characters sit. The argument specifying the y coordinate in `drawString()` method specifies the baseline position.
Advance width	It is the horizontal width of a character or string.
Ascent	It is the distance above the baseline of a character or string.
Descent	It is the distance below the baseline of a character or string.
Leading	It is the vertical spacing between descent of one line and ascent of the next line. The origin of this term is actually from the strips of leads used to separate lines.

FontMetrics Class



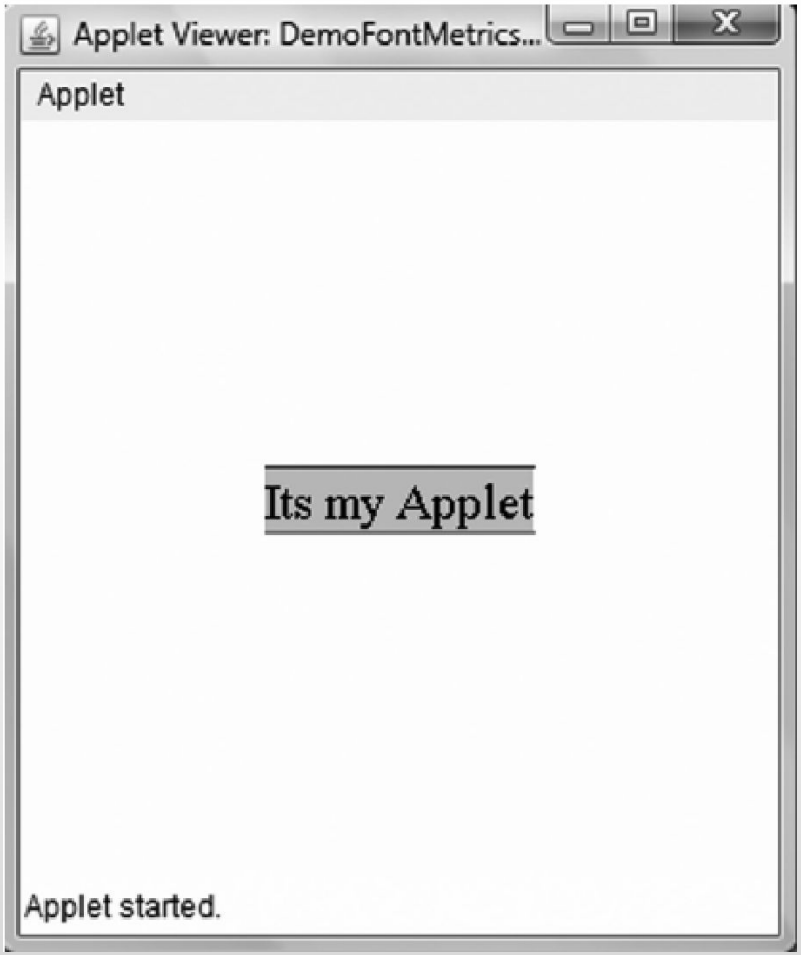
Example

```
/*<applet code=DemoFontMetrics.class width=300 height=300></applet>*/
import java.awt.*;
import java.applet.*;
public class DemoFontMetrics extends Applet {
public void init() {
Color color1 = new Color(255, 255, 250);
setBackground(color1);}
public void paint(Graphics g) {
int fontSize = 20;
g.setFont(new Font("TimesRoman", Font.PLAIN,fontSize));
FontMetrics fm = g.getFontMetrics();
String s = "Its my Applet";
int sw = fm.stringWidth(s);
int w =300;
int h = 300;
int x = (w - sw) / 2;
```

Example (contd.)

```
int baseline = fm.getMaxAscent() + (h - (fm.getAscent() +
fm.getMaxDescent()))/2;
int ascent = fm.getMaxAscent();
int descent = fm.getMaxDescent();
int fontHeight = fm.getMaxAscent() + fm.getMaxDescent();
g.setColor(Color.pink);
g.fillRect(x, baseline-ascent , sw, fontHeight);
g.setColor(Color.red);
g.drawLine(x, baseline+descent, x+sw, baseline+descent);
g.setColor(Color.blue);
g.drawLine(x, baseline-ascent, x+sw, baseline- ascent);
g.setColor(Color.black);
g.drawString(s, x, baseline);}}
```

The Output



Digital Clock Example

```
import java.util.*;
import java.applet.*;
import java.awt.*;
import java.text.*;
/*<applet code=DigitalClock.class width=450 height=100></applet>*/
public class DigitalClock extends Applet implements Runnable{
Thread t;
Calendar c;
Date d;
DateFormat df;
```

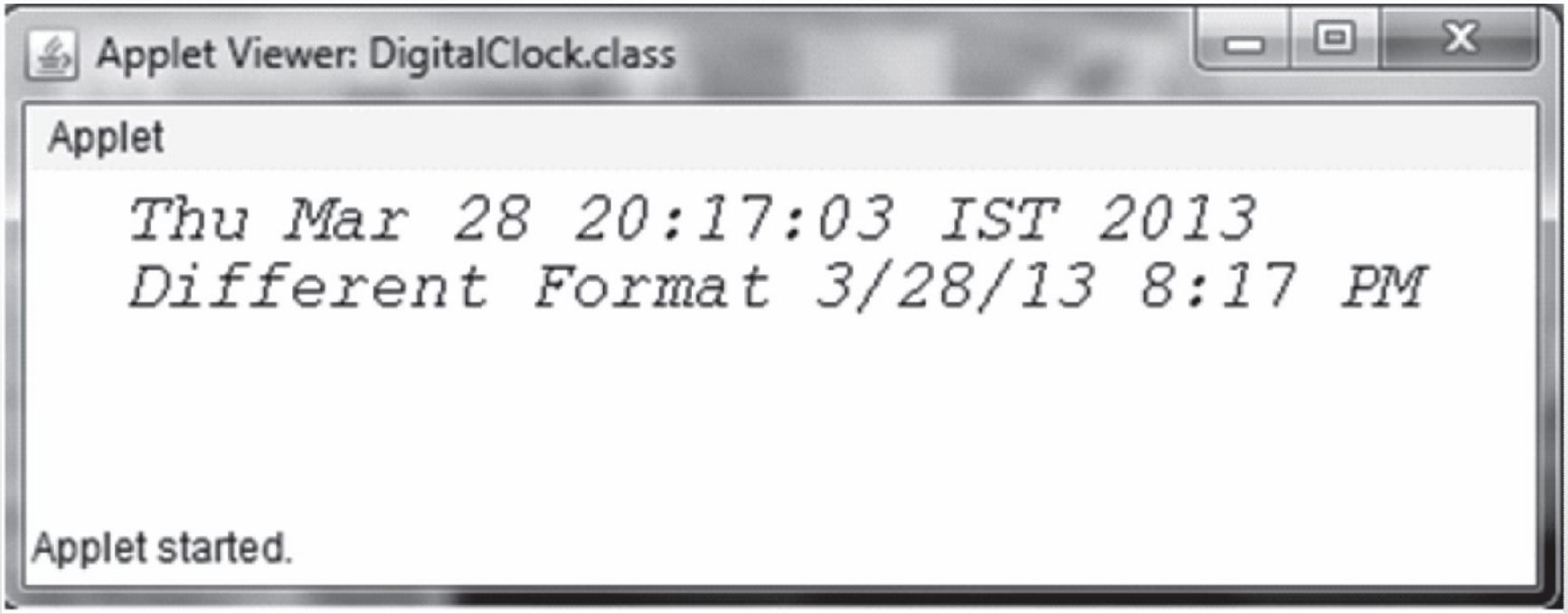
Digital Clock Example

```
public void init(){
t=new Thread(this,"Time Thread");
t.start();
df=DateFormat.getInstance();}
public void run(){
for(;;){
try{
c=Calendar.getInstance();
d=c.getTime();
Thread.sleep(1000);
}
```

Digital Clock Example

```
catch(Exception e){}
repaint();
 }}
public void paint(Graphics g){
g.setFont(new Font("Courier New", Font.ITALIC, 20));
g.drawString(d.toString(),30,20);
g.drawString("Different Format " +df.format(d),30,40);
}}
```

The Output



Summary

- Applets are small programs which can be downloaded from a remote server in its bytecode form and executed on the client, to do a specific job.
- In Java, applets can be dealt in two ways.
- Conventional applet, which uses Abstract Window Toolkit (AWT) to get the GUI features.
- Other uses Swings. These applets can be executed on the clients, with the help of either a Java enabled browser or a utility known as appletviewer.
- Applets have a proper life cycle in which an applet moves from one state to other.
- These states of applet life cycle are: Born, Running, Idle, and Dead.
- Methods such as init(), start(), stop(), and destroy() are respectively called to force an applet to different state.
- We have given you an insight of how to handle images and audio files and have a basic understanding of graphics.
- We have also seen How to use Threads in Applets.