Web Technology 14

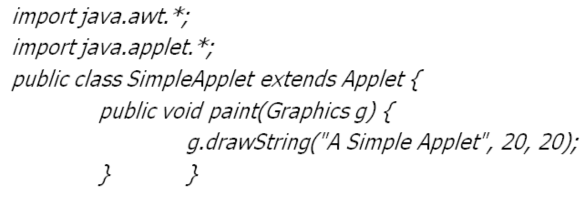
Applet

Chittaranjan Pradhan
School of Computer Engineering,
KIIT University

Applet

Applet

- Applets are small applications that are accessed on an Internet server, transported over the Internet, automatically installed, and run as part of a Web document



```
import java.awt.*;
import java.applet.*;
public class SimpleApplet extends Applet {
            public void paint(Graphics g) {
                        g.drawString("A Simple Applet", 20, 20);
            }           }
```

- Applets interact with the user through the AWT(Abstract Window Toolkit)
- paint() method is defined by the AWT and must be overridden by the applet. It is called each time that the applet must redisplay its output
- drawString() is a member of the Graphics class. It outputs a string beginning at the specified X,Y location

Applet...

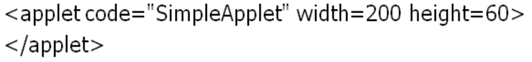
Applet...

- Notice that the applet does not have a **main()** method. Unlike Java programs, applets do not begin execution at main().In fact, most applets don't even have a main() method
- An applet begins execution when the name of its class is passed to an appletviewer or to a web browser
- Compile the source code in the same way that you have been compiling programs
- java.lang.Object → java.awt.Component → java.awt.Container → java.awt.Panel → java.applet.Applet
- An applet's class must be public
- Running of applet involves different process:
    - Executing the applet within a Java-compatible Web browser
    - Using an applet viewer, such as the standard SDK tool, appletviewer. An appletviewer executes your applet in a window. This is generally the fastest and easiest way to test your applet

Applet...

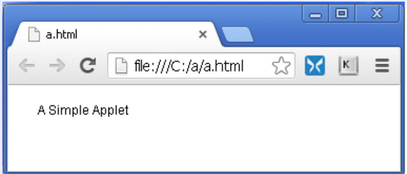
Executing an applet in a Web browser

- Here, you need to write a short HTML text file that contains the appropriate APPLET tag

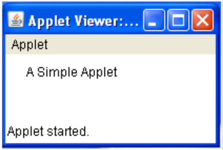


```
<applet code="SimpleApplet" width=200 height=60>
</applet>
```

- After you create this file, you can execute your browser and then load this file, which causes SimpleApplet to be executed
- To execute SimpleApplet with an appletviewer, the command is:
  appletviewer a.html



A Simple Applet



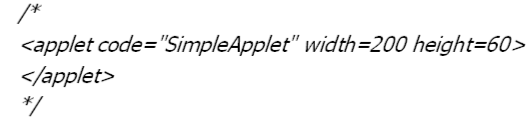
A Simple Applet

Applet started.

Applet...

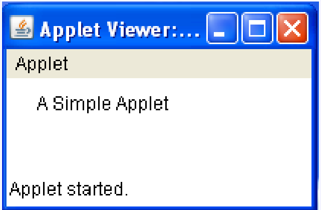
Executing an applet using appletviewer

- Include a comment at the head of your Java source code file that contains the APPLET tag



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

- By doing so, your code is documented with a prototype of the necessary HTML statements, and you can test your compiled applet merely by starting the applet viewer with your Java source code file like
  appletviewer SimpleApplet.java



```
A Simple Applet

Applet started.
```

<APPLET> Tag

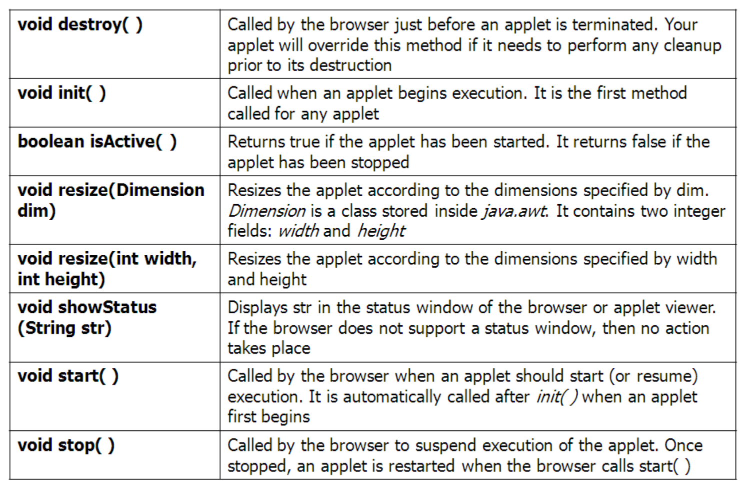
<APPLET> Tag

<APPLET> tag is used to embed an applet into an HTML page

```
<APPLET CODE=äpplet class name"
CODEBASE=äpplet class file path"
HEIGHT=maximum height"WIDTH=maximum width"
ALIGN=älignmentÄLT=älternate text»
<PARAM NAME=parameter name"VALUE="value»
</APPLET>
```

Applet Class

Applet Class



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
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
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
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( )

Applet Architecture

Applet Architecture

- Applets are event driven. An applet waits until an event occurs. The runtime system notifies the applet about an event by calling an event handler that has been provided by the applet. Once this happens, the applet must take appropriate action and the quickly returns
- User initiates interaction with an applet. These interactions are sent to the applet as events to which the applet must respond
- Applet extends the AWT class Panel. In turn, Panel extends Container, which extends Component, which extends Object

Applet Lifecycle

Applet Lifecycle

When an applet begins, the AWT calls the following methods, in this sequence:

- **public void init()**: it is the first method to be called. This is used to initialized the Applet. It is invoked only once
- **public void start()**: it is invoked after the init() method or browser is maximized. It is used to start the Applet. Whereas init() is called once, start() is called each time an applet's HTML document is displayed on screen
- **public void paint(Graphics g)**: it is used to paint the Applet. It provides Graphics class object that can be used for drawing shapes. It is called each time your applet's output must be redrawn. Whenever the applet must redraw its output, paint() is called

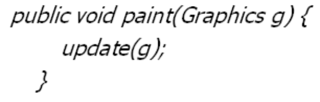
Applet Lifecycle...

Applet Lifecycle...

When an applet is terminated, the following sequence of method calls takes place:

- **public void stop()**: it is used to stop the Applet. It is invoked when Applet is stop or browser is minimized
- **public void destroy()**: it is used to destroy the Applet. It is invoked only once. The stop() method is always called before destroy()

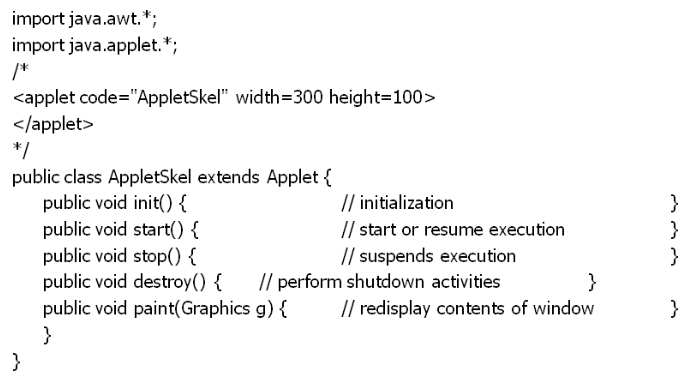
update() method is called when your applet has requested that a portion of its window be redrawn



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

Applet Lifecycle...

Applet Lifecycle...



```
import java.awt.*;
import java.applet.*;
/*
<applet code="AppletSkel" width=300 height=100>
</applet>
*/
public class AppletSkel extends Applet {
    public void init() {                // initialization                    }
    public void start() {               // start or resume execution         }
    public void stop() {                // suspends execution                }
    public void destroy() {      // perform shutdown activities         }
    public void paint(Graphics g) {     // redisplay contents of window      }
    }
}
```

Applet Display Methods

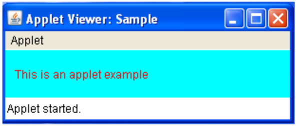
Applet Display Methods

- The drawString() method will not recognize *newline* characters
- setBackground() and setForeground() used to set the color. These methods are defined by Component
  void setBackground(Color newColor)
  void setForeground(Color newColor)

  where, newColor specifies the new color. The class Color defines the constants shown here that can be used to specify colors: {**Color.black, Color.blue, Color.cyan Color.magenta, Color.gray Color.green, Color.lightGray, Color.orange, Color.pink, Color.red, Color.white, Color.yellow**}
- A good place to set the foreground and background colors is in the *init()* method
- You can obtain the current settings for the background and foreground colors by calling **getBackground()** and **getForeground()**

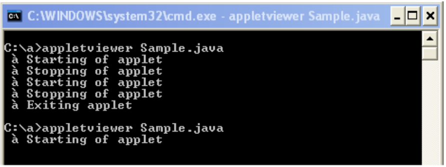
Example 1

```
import java.awt.*;
import java.applet.*;
/*
<applet code="Sample"  width=300 height=50>
</applet>
*/
public class Sample extends Applet{
        String msg;
        public void init() {
                                        setBackground(Color.cyan);
                                        setForeground(Color.red);
                                        msg = “This is an applet example";
                                        }
        public void start() {           System.out.println(“ ... Starting of applet”);  }
   public void stop(){                  System.out.println(“ ... Stopping of applet”); }
  public void destroy(){                System.out.println(“ ... Exiting applet”);       }
  public void paint(Graphics g) {   g.drawString(msg, 10, 30);                   }
        }
```



This is an applet example

Applet started.

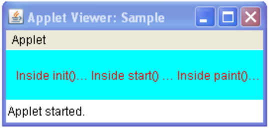


```
C:\a>appletviewer Sample.java
 à Starting of applet
 à Stopping of applet
 à Starting of applet
 à Stopping of applet
 à Exiting applet

C:\a>appletviewer Sample.java
 à Starting of applet
```

Example 2

```
import java.awt.*;
import java.applet.*;
/*
<applet code="Sample"  width=300 height=50>
</applet>
*/
public class Sample extends Applet{
        String msg;
        public void init() {
                                setBackground(Color.cyan);
                                setForeground(Color.red);
                                msg = “Inside init()…";
                                }
        public void start() {
                                msg+= “ Inside start() …”;
                                }
        public void paint(Graphics g) {
                                msg+= “ Inside paint()…”;
                                g.drawString(msg, 10, 30);
                                }
        }
```



```
Inside init()… Inside start() … Inside paint()…
```

Applet started.

Requesting Repainting

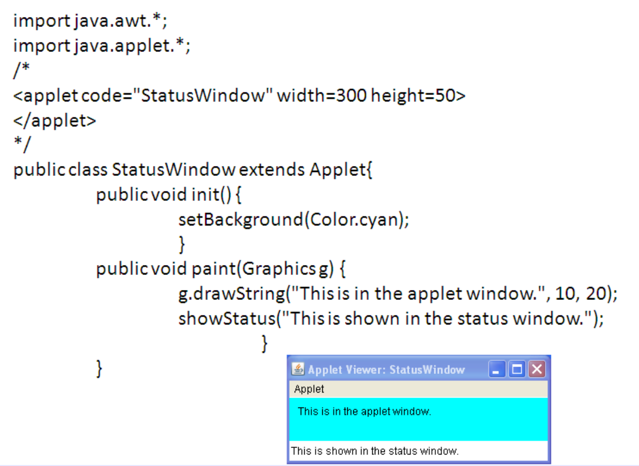
Requesting Repainting

- As a general rule, an applet writes to its window only when its update() or paint() method is called by the AWT
- Whenever your applet needs to update the information displayed in its window, it simply calls *repaint()*
- The repaint() method is defined by the AWT. It causes the AWT run-time system to execute a call to your applet's update() method, which, in its default implementation, calls paint():
    - **void repaint()**: It causes the entire window to be repainted
    - **void repaint(int left, int top, int width, int height)**: It specifies a region that will be repainted. These dimensions are specified in pixels
    - **void repaint(long maxDelay)**: maxDelay specifies the maximum number of milliseconds that can elapse before update() is called
    - **void repaint(long maxDelay, int x, int y, int width, int height)**

Status Window

Status Window

An applet can also output a message to the status window of the browser or applet viewer on which it is running by using showStatus()



```
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) {
                    g.drawString("This is in the applet window.", 10, 20);
                    showStatus("This is shown in the status window.");
                              }
          }
```

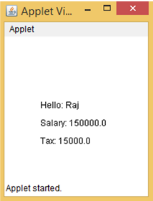
```
This is in the applet window.
```

```
This is shown in the status window.
```

Parameter passing to Applet

APPLET tag allows to pass parameters to the applet

```
import java.awt.*;
import java.applet.*;
/*
<APPLET CODE="Tax" HEIGHT=200 WIDTH=200>
<PARAM name="t1" VALUE="Raj">
<PARAM name="t2" VALUE="150000">
</APPLET>
*/
public class Tax extends Applet{
String name;
double sal, tax;
public void init(){
        name=getParameter("t1");
        sal=Double.parseDouble(getParameter("t2"));
        calc(sal);
        }
public void calc(double sal){
        if(sal<=100000)
                tax=0;
        else if(sal<=500000)
                tax=sal*0.1;
        else
                tax=sal*0.2;
        }
public void paint(Graphics g){
        g.drawString("Hello: "+name,50, 100);
        g.drawString("Salary: "+sal, 50, 125);
        g.drawString("Tax: "+tax, 50, 150);
        }
}
```



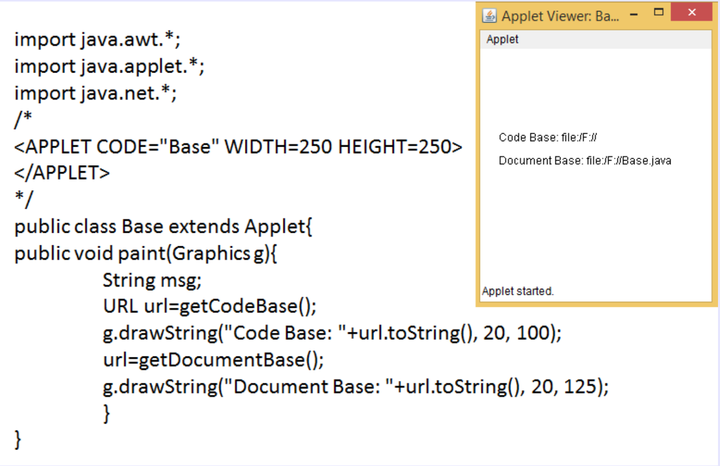
```
Hello: Raj
Salary: 150000.0
Tax: 15000.0
```

Applet started.

getDocumentBase() and getCodeBase()

getDocumentBase() and getCodeBase()

DocumentBase gives the detailed path of the applet’s class file. CodeBase gives the directory from which the applet’s class file was loaded



```
import java.awt.*;
import java.applet.*;
import java.net.*;
/*
<APPLET CODE="Base" WIDTH=250 HEIGHT=250>
</APPLET>
*/
public class Base extends Applet{
public void paint(Graphics g){
        String msg;
        URL url=getCodeBase();
        g.drawString("Code Base: "+url.toString(), 20, 100);
        url=getDocumentBase();
        g.drawString("Document Base: "+url.toString(), 20, 125);
        }
}
```

```
Code Base: file:/F://

Document Base: file:/F://Base.java
```

Applet started.

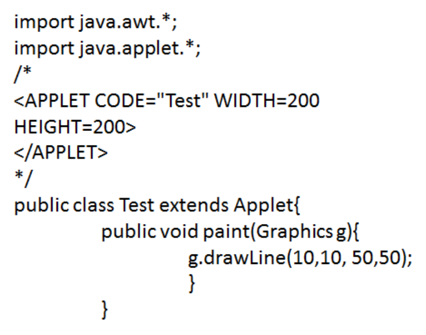
Working with Graphics

Working with Graphics

- AWT supports a rich assortment of graphics methods
- All graphics are drawn relative to a window
- *Graphics* class defines a number of drawing functions

Drawing Lines

void drawLine(int x1, int y1, int x2, int y2)

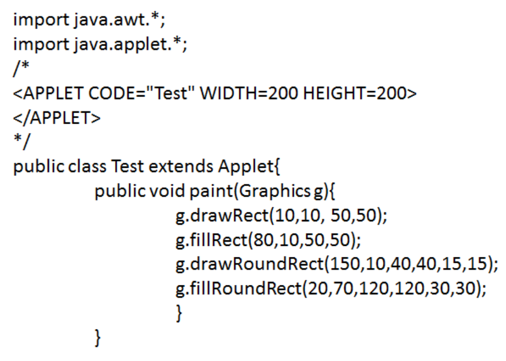


```
import java.awt.*;
import java.applet.*;
/*
<APPLET CODE="Test" WIDTH=200
HEIGHT=200>
</APPLET>
*/
public class Test extends Applet{
            public void paint(Graphics g){
                        g.drawLine(10,10, 50,50);
                        }
            }
```

Working with Graphics...

Drawing Rectangles

void drawRect(int top, int left, int width, int height)
void fillRect(int top, int left, int width, int height)
void drawRoundRect(int top, int left, int width, int height, int arcWidth, int arcHeight)
void fillRoundRect(int top, int left, int width, int height, int arcWidth, int arcHeight)

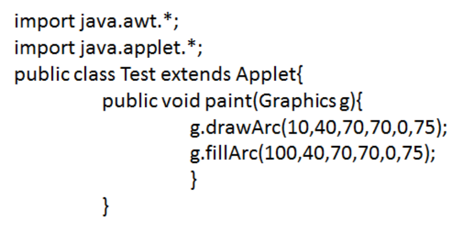


```
import java.awt.*;
import java.applet.*;
/*
<APPLET CODE="Test" WIDTH=200 HEIGHT=200>
</APPLET>
*/
public class Test extends Applet{
        public void paint(Graphics g){
                g.drawRect(10,10, 50,50);
                g.fillRect(80,10,50,50);
                g.drawRoundRect(150,10,40,40,15,15);
                g.fillRoundRect(20,70,120,120,30,30);
                }
        }
```

Working with Graphics...

Drawing Arcs

void drawArc(int top, int left, int width, int height, int startAngle, int sweepAngle)
void fillArc(int top, int left, int width, int height, int startAngle, int sweepAngle)

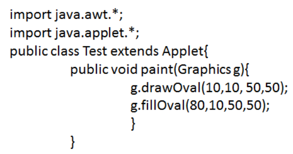


```
import java.awt.*;
import java.applet.*;
public class Test extends Applet{
            public void paint(Graphics g){
                        g.drawArc(10,40,70,70,0,75);
                        g.fillArc(100,40,70,70,0,75);
                        }
            }
```

Working with Graphics...

Drawing Ellipses & Circles

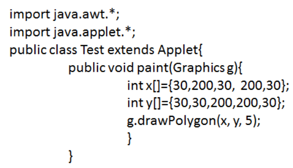
void drawOval(int top, int left, int width, int height)
void fillOval(int top, int left, int width, int height)



```
import java.awt.*;
import java.applet.*;
public class Test extends Applet{
            public void paint(Graphics g){
                        g.drawOval(10,10, 50,50);
                        g.fillOval(80,10,50,50);
                        }
            }
```

Drawing Polygons

void drawPolygon(int x[], int y[], int noPts)
void fillPolygon(int x[], int y[], int noPts)



```
import java.awt.*;
import java.applet.*;
public class Test extends Applet{
            public void paint(Graphics g){
                        int x[]={30,200,30,  200,30};
                        int y[]={30,30,200,200,30};
                        g.drawPolygon(x, y, 5);
                        }
            }
```

Working with Color

Working with Color

- Color is a class which is inherited from Object class and implements Paint and Serializable interfaces

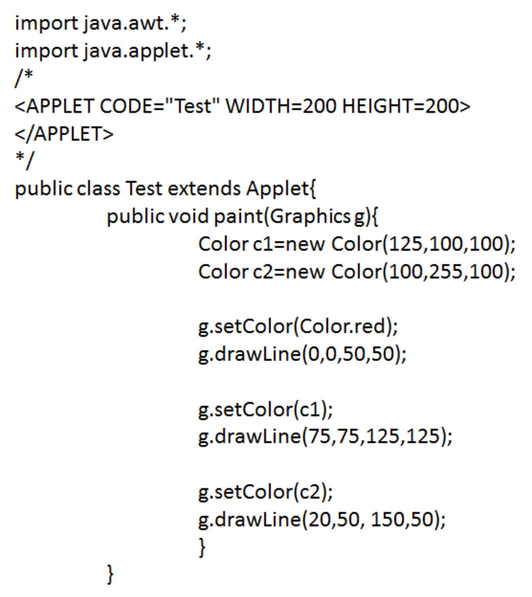
- The default colors are: {Color.black, Color.blue, Color.cyan Color.magenta, Color.gray Color.green, Color.lightGray, Color.orange, Color.pink, Color.red, Color.white, Color.yellow }

- **Color(float r, float g, float b)**
  Color(int r, int g, int b)

  Ex: Color c=new Color(255,100,100);
- **void setColor(Color newColor)**
  Color getColor()

Working with Color...

Working with Color...



```
import java.awt.*;
import java.applet.*;
/*
<APPLET CODE="Test" WIDTH=200 HEIGHT=200>
</APPLET>
*/
public class Test extends Applet{
        public void paint(Graphics g){
                Color c1=new Color(125,100,100);
                Color c2=new Color(100,255,100);

                g.setColor(Color.red);
                g.drawLine(0,0,50,50);

                g.setColor(c1);
                g.drawLine(75,75,125,125);

                g.setColor(c2);
                g.drawLine(20,50, 150,50);
                }
        }
```

Working with Font

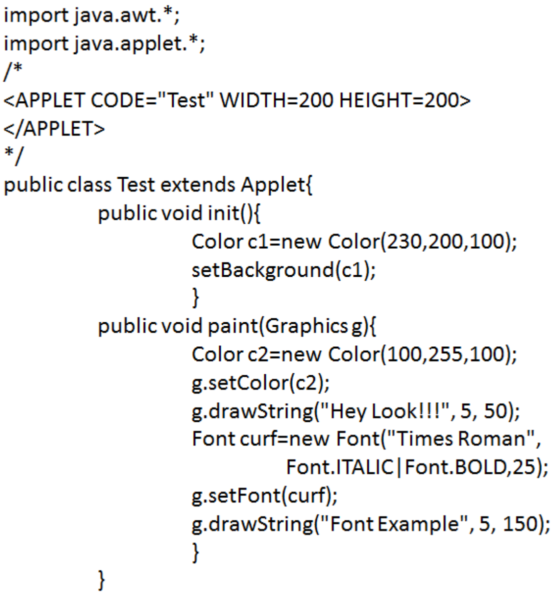
Working with Font

- Font class extends Object class and implements Serializable interface

- The variables of Font are: {Name, PointSize, size, style}

- **Font(Font font)**
  Font(String name, int style, int size)
    - name specifies the name of the desired Font
    - style specifies the style of the desired Font, which can be Font.PLAIN, Font.BOLD, Font.ITALIC or any combination like Font.ITALIC | Font.BOLD
    - size specifies the size of the Font in points
- Once the Font is created, **setFont()** method of Component class is used to select it

Working with Font...



```
import java.awt.*;
import java.applet.*;
/*
<APPLET CODE="Test" WIDTH=200 HEIGHT=200>
</APPLET>
*/
public class Test extends Applet{
        public void init(){
                Color c1=new Color(230,200,100);
                setBackground(c1);
                }
        public void paint(Graphics g){
                Color c2=new Color(100,255,100);
                g.setColor(c2);
                g.drawString("Hey Look!!!", 5, 50);
                Font curf=new Font("Times Roman",
                        Font.ITALIC|Font.BOLD,25);
                g.setFont(curf);
                g.drawString("Font Example", 5, 150);
                }
        }
```