CS6308- Java Programming

V P Jayachitra
Assistant Professor
Department of Computer Technology
MIT Campus
Anna University

Applets

- Swing defines two types of containers.
- The first are top-level containers:
  - **JFrame, JApplet, JWindow, and JDialog.**
  - A top-level container is not contained within any other container.
  - The one most commonly used for applications is JFrame.
  - In the past, the one used for applets was JApplet.
    - beginning with JDK 9 applets have been deprecated. As a result, JApplet is deprecated.
    - Furthermore, beginning with JDK 11, applet support has been removed.
- The second type of containers supported by Swing are lightweight containers.
  - **JPanel, which is a general-purpose container.**

What is Applet?

- The applet is a dynamic, self-executing program.
- **Applets are programs stored on a web server, similar to web pages.**
- An **applet** is a Java class that run in the browser's environment on the **client host** which is downloaded from the **web server**.
- **An applet program is a written as a subclass of the java.Applet class or the javax.swing.Japplet class.**
  - **There is no main method**
  - **Applet uses AWT for graphics.**
  - **JApplet uses SWING for graphics.**

Why Applets?

- Ensures security and portability.
  - The execution of an applet is normally subject to restrictions:
    - applets cannot access files in the file system on the client host
      - As the java class runs in java execution environment
      - an applet cannot ordinarily read or write files on the computer that it's executing on.
    - an applet cannot make network connections except to the host that it came from.
- The same application code works on all computers.

```
import java.awt.Graphics;
  // import class Graphics
import javax.swing.JApplet;
  // import class Japplet
public class WelcomeApplet extends
JApplet {
    // draw text on applet's background
   public void paint( Graphics g ){
       // draw a String at x-coordinate 25 and y-coordinate 25
   g.drawString( "Java Programming!", 25, 25
);
  }
 }
```

- **`import`** the **`JApplet`** class (package **`javax.swing`**)
- **`import`** the **`Graphics`** class (package **`java.awt`**) to draw graphics
  - Can draw lines, rectangles ovals, strings of characters
- Methods **`paint`**, **`init`**, and **`start`**
  Guaranteed to be called automatically
- Body of **`paint`**
  Method **`drawString`** (of class **`Graphics`**)

```
<html>
  <applet code = "WelcomeApplet.class"
width = "300" height = "40">
  </applet>
</html>
```

<APPLET specifies the beginning of the HTML applet code

CODE="demoxx.class" is the actual name of the applet (usually a 'class' file)
CODEBASE="demos/" is the location of the applet (relative as here, or a full URL)
NAME="smily" the name you want to give to this instance of the applet on this page
WIDTH="100" the physical width of the applet on your page
HEIGHT="50" the physical height of the applet on your page
ALIGN="Top" where to align the applet within its page space (top, bottom, center)>

<PARAM specifies a parameter that can be passed to the applet (applet specific)
NAME="name2" the name known internally by the applet in order to receive this parameter
VALUE="ffffff" the value you want to pass for this parameter

> end of this parameter

</APPLET> specifies the end of the HTML applet code

```
import java.awt.Graphics;
import javax.swing.JApplet;
public class WelcomeApplet extends JApplet {
  public void paint( Graphics g ){
      // draw horizontal line from (25, 20) to (310, 20)
         g.drawLine( 25, 20, 310, 20 );
         g.drawString( "Java Programming!", 35, 35 );
   // draw horizontal line from (25, 40) to (310, 40)
        g.drawLine( 25, 40, 310, 40 );
  }
}
```

// draw a String at x-coordinate 25 and y-coordinate 25

```
<html>
  <applet code = "WelcomeApplet.class"
width = "300" height = "40">
 </applet>
</html>
```

```
import java.awt.Graphics;
import javax.swing.JApplet;
public class WelcomeApplet extends JApplet {
double sum;
public void init()   {
   String firstNumber;
   String secondNumber;
    double number1;
    double number2;
    firstNumber = JOptionPane.showInputDialog(   "Enter first value" );      string!!
    secondNumber = JOptionPane.showInputDialog(  "Enter second value" );
    number1 = Double.parseDouble( firstNumber );    double!!
    number2 = Double.parseDouble( secondNumber );
    sum = number1 + number2;
 }
  public void paint( Graphics g ){
     // draw horizontal line from (25, 20) to (310, 20)
        g.drawLine( 25, 20, 310, 20 );
        g.drawString( "The sum is "+ sum, 35, 35 );    // draw a String at x-coordinate 35 and y-coordinate 35
  // draw horizontal line from (25, 40) to (310, 40)
       g.drawLine( 25, 40, 310, 40 );
 }
}
```

```
<html>
  <applet code = "WelcomeApplet.class" width = "300" height = "40">
 </applet>
</html>
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class AddNumbersApplet extends JApplet implements ActionListener {
private JTextField textField1, textField2;
   private JButton addButton;
   private JLabel resultLabel;
   public void init() {
      setLayout(new FlowLayout());
      textField1 = new JTextField(10);
      textField2 = new JTextField(10);
      addButton = new JButton("Add");
      resultLabel = new JLabel("Result: ");
      add(new JLabel("Enter first number:"));       add(textField1);
      add(new JLabel("Enter second number:"));    add(textField2);
      add(addButton);
      add(resultLabel);

      addButton.addActionListener(this) ;
       }
        public void actionPerformed(ActionEvent e) {
          try {
                double number1 = Double.parseDouble(textField1.getText());
                double number2 = Double.parseDouble(textField2.getText());
                double sum = number1 + number2;
               resultLabel.setText("Result: " + sum);
            } catch (NumberFormatException ex) {
                  resultLabel.setText("Please enter valid numbers.");              }
          }
    }
```

The init() method is a special method in the JApplet class (or the deprecated Applet class).
It is called by the Java Applet Lifecycle when the applet is initialized (e.g., when embedded in a browser or applet viewer).

Yes, actionPerformed(ActionEvent e) is a built-in method. It is defined in the ActionListener interface, which is part of the java.awt.event package.

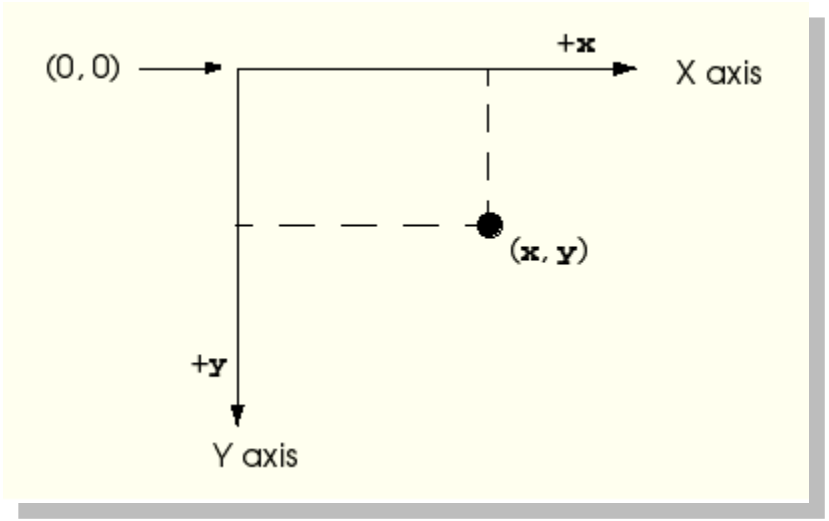
```
import java.awt.*;
import java.applet.*;

public class hello extends Applet{
        public void init( ){
                setBackground(Color.yellow);
        }
public void paint(Graphics g){
    final int FONT_SIZE = 42;
    Font font = new Font("Serif", Font.BOLD, FONT_SIZE);
    g.setFont(font);
    g.setColor(Color.blue);
   g.drawString("Hello,    world!",250,150);
        }
}
```

```
<html>
  <applet code = "hello.class" width = "300" height = "40">
 </applet>
</html>
```

`JApplet` methods that the applet container calls during an applet's execution.	
Method	purpose
public void `init()`	This method is called once by the applet container when an applet is loaded for execution. It performs initialization of an applet. Typical actions performed here are initializing fields, creating GUI components, loading sounds to play, loading images to display and creating threads.
public void `start()`	This method is called after the `init` method completes execution. In addition, if the browser user visits another Web site and later returns to the HTML page on which the applet resides, method `start` is called again. The method performs any tasks that must be completed when the applet is loaded for the first time and that must be performed every time the HTML page on which the applet resides is revisited.
public void `paint( Graphics g )`	This drawing method is called after the `init` method completes execution and the `start` method has started. It is also called every time the applet needs to be repainted. For example, if the user covers the applet with another open window on the screen and later uncovers the applet, the `paint` method is called.
public void `stop()`	This method is called when the applet should stop executing—normally, when the user of the browser leaves the HTML page on which the applet resides. The method performs any tasks that are required to suspend the applet's execution. Typical actions performed here are to stop execution of animations and threads.
public void `destroy()`	This method is called when the applet is being removed from memory—normally, when the user of the browser exits the browsing session (i.e., closes all browser windows). The method performs any tasks that are required to destroy resources allocated to the applet.

- Running the applet
  - Compile
    - **`javac WelcomeApplet.java`**
    - If no errors, bytecodes stored in **`WelcomeApplet.class`**
  - Create an HTML file
    - Loads the applet into **`appletviewer`** or a browser
    - Ends in **`.htm`** or **`.html`**
  - To execute an applet
    - Create an HTML file indicating which applet the browser (or **`appletviewer`**) should load and execute

JAVA Coordinate spaces



- Coordinates are specified as
  - Integer, float or double
- Coordinate units are in pixels
- Origin is in upper left corner of drawing area i.e., (0,0).
- X value increases to the right.
- Y value increases downward.

Java 2D API

- Two-dimensional
- Extends Graphics class to support enhanced features.
- Supports following capabilities for Java programs.
  - graphics –drawXxx() and fillXxx()
  - text –drawString()
  - Imaging- drawImage()
- Java.awt.Graphics supports
  - draw method
  - fill method
  - set method –setFont() and setColor() define how to draw and fill

Java 2D API in the application

- To employ Java 2D API in the application, cast the Graphics object to a Graphics2D object.

```
public void paint (Graphics g) {
        Graphics2D g2 = (Graphics2D) g;
            ... }
```

- 2D spans
  - `java.awt.image`
  - `java.awt.color`
  - `java.awt.font`
  - `java.awt.geom`

Geometric Primitives

- java.awt.geom package.
- Java 2D API provides set of standard shapes such as
  - points
  - Lines
  - Rectangles
  - Arcs
  - Ellipses
  - curves.
- Point2D,Line2D,Rectangle2D, RoundRectangle2D, Arc2D,Ellipse2D Class.
- Why Xxx2D classes?
  - **Xxx2D.Double and Xxx2D.Float** to support double- and float-precision.
  - For better accuracy.
  - Extends `java.awt.geom` package

Text

- Graphics class
  - using the drawString method.
  - setFont(), getFont(), setColor(),getColor(),drawRect(),fillRect(),
- FONT

  Java SE defines the following five logical *font name* families:
  - Dialog
  - DialogInput
  - Monospaced
  - Serif (i.e TimesNewRoman)
  - SansSerif

  Java SE defines the following *font style*:
  - PLAIN, BOLD,ITALIC
- **Font(FontFace, Style, Size)**
- Font font = new Font("Dialog", Font.PLAIN, 12);
- Font (`setFont()`, `getFont()`)
- Color (`setColor()`, `getColor()`)

Paint

- Graphics2D's paint attribute determines the color of the shape.
  - setPaint() method.
- Paint object implements the java.awt.Paint interface.
  - Java 2D built-in Paint objects
    - GradientPaint, LinearGradientPaint, RadialGradientPaint, MultipleGradientPaint, TexturePaint.

```
g.setPaint(new GradientPaint(80, 80, Color.RED, 200, 150, Color.GREEN));
 // set current paint object to a GradientPaint, from (x1, y1) with color1 to (x2, y2) with color2
g.fill(new Rectangle2D.Double(50, 80, 200, 100));
// fill the Shape with the current paint object
g.setColor(Color.CYAN);
g.draw(new Rectangle2D.Double(50, 50, 100, 200));
```

drawLine()

```
public abstract void drawLine(int x1,
                              int y1,
                              int x2,
                              int y2)
```

Draws a line, using the current color, between the points `(x1, y1)` and `(x2, y2)` in this graphics context's coordinate system.

Parameters:

`x1` - the first point's *x* coordinate.

`y1` - the first point's *y* coordinate.

`x2` - the second point's *x* coordinate.

`y2` - the second point's *y* coordinate.

fillRect()

```
public abstract void fillRect(int x,
              int y,
              int width,
              int height)
```

Parameters:

`x` - the *x* coordinate of the rectangle to be filled.

`y` - the *y* coordinate of the rectangle to be filled.

`width` - the width of the rectangle to be filled.

`height` - the height of the rectangle to be filled.

drawRect()

```
public void drawRect(int x,
                     int y,
                     int width,
                     int height)
```

Draws the outline of the specified rectangle.
The left and right edges of the rectangle are at `x` and `x + width`. The top and bottom edges are at `y` and `y + height`.

Parameters:

`x` - the *x* coordinate of the rectangle to be drawn.

`y` - the *y* coordinate of the rectangle to be drawn.

`width` - the width of the rectangle to be drawn.

`height` - the height of the rectangle to be drawn.

fillRect()

```
public abstract void fillRect(int x,
            int y,
            int width,
            int height)
```

Fills the specified rectangle.

The rectangle is filled using the graphics context's current color.

The left and right edges of the rectangle are at `x` and `x + width - 1`. The top and bottom edges are at `y` and `y + height - 1`.

Parameters:

`x` - the *x* coordinate of the rectangle to be filled.

`y` - the *y* coordinate of the rectangle to be filled.

`width` - the width of the rectangle to be filled.

`height` - the height of the rectangle to be filled.

clearRect()

```
public abstract void clearRect(int x,
                               int y,
                               int width,
                               int height)
```

Clears the specified rectangle by filling it with the background color of the current drawing surface.

Parameters:

`x` - the *x* coordinate of the rectangle to clear.

`y` - the *y* coordinate of the rectangle to clear.

`width` - the width of the rectangle to clear.

`height` - the height of the rectangle to clear.

drawArc()

```
public abstract void drawArc(int x,
                             int y,
                             int width,
                             int height,
                             int startAngle,
                             int arcAngle)
```

Parameters:

`x` - the *x* coordinate of the upper-left corner of the arc to be drawn.

`y` - the *y* coordinate of the upper-left corner of the arc to be drawn.

`width` - the width of the arc to be drawn.

`height` - the height of the arc to be drawn.

`startAngle` - the beginning angle.

`arcAngle` - the angular extent of the arc, relative to the start angle.

drawPolyline() and drawPolygon()

```
public abstract void drawPolyline(int[] xPoints,
                                  int[] yPoints,
                                  int nPoints)
```

Draws a sequence of connected lines defined by arrays of *x* and *y* coordinates. Each pair of (*x*, *y*) coordinates defines a point.

Parameters:

`xPoints` - an array of *x* points

`yPoints` - an array of *y* points

`nPoints` - the total number of points

```
public abstract void drawPolygon(int[] xPoints,
                                 int[] yPoints,
                                 int nPoints)
```

Draws a closed polygon defined by arrays of *x* and *y* coordinates. Each pair of (*x*, *y*) coordinates defines a point.

Parameters:

`xPoints` - a an array of `x` coordinates.

`yPoints` - a an array of `y` coordinates.

`nPoints` - a the total number of points.

drawstring()

```
public abstract void drawString(String str,
                                int x,
                                int y)
```

Draws the text given by the specified string, using this graphics context's current font and color. The baseline of the leftmost character is at position (*x*, *y*) in this graphics context's coordinate system.

Parameters:

`str` - the string to be drawn.

`x` - the *x* coordinate.

`y` - the *y* coordinate.

drawImage()

```
public abstract boolean drawImage(Image img,
                                  int x,
                                  int y,
                                  ImageObserver observer)
```

Draws as much of the specified image as is currently available. The image is drawn with its top-left corner at (*x*, *y*) in this graphics context's coordinate space.

Transparent pixels in the image do not affect whatever pixels are already there.

This method returns immediately in all cases, even if the complete image has not yet been loaded, and it has not been dithered and converted for the current output device.

If the image has completely loaded and its pixels are no longer being changed, then `drawImage` returns `true`. Otherwise, `drawImage` returns `false`
process that loads the image notifies the specified image observer.

Parameters:

`img` - the specified image to be drawn. This method does nothing if `img` is null.

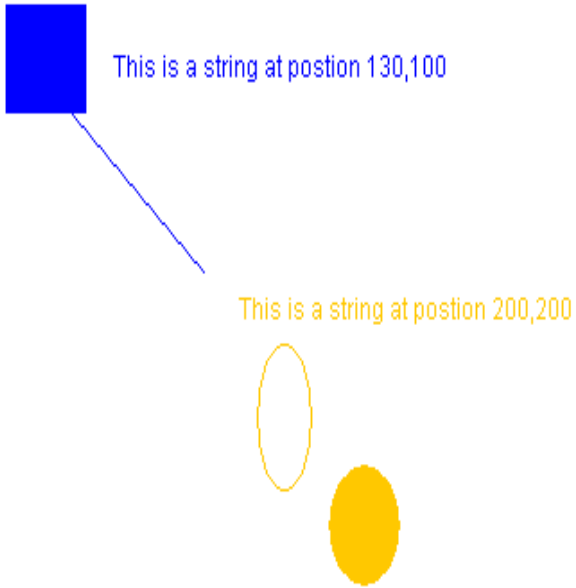
`x` - the *x* coordinate.

`y` - the *y* coordinate.

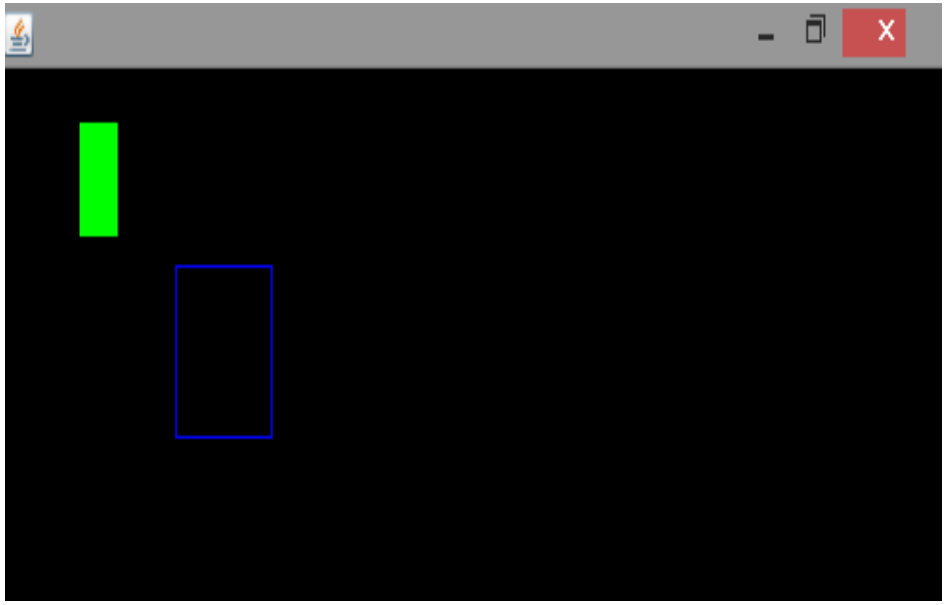
`observer` - object to be notified as more of the image is converted.

Returns:

`false` if the image pixels are still changing; `true` otherwise.




```
//sample program to draw simple geometric objects
import java.awt.Color;
import java.awt.Font;
import java.awt.Container;
import java.awt.FlowLayout;
import javax.swing.JFrame;
import java.awt.Graphics;
public class Shapes extends JFrame{
  Shapes() {
              //FRAME SETTINGS
             setSize(400,400);
             setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
             setLayout(new FlowLayout());
              setVisible(true);
             pack();
             }
 public void paint(Graphics g){
             g.setColor(Color.BLUE);
             g.drawString("This is a string at postion 130,100",130,100);
             g.drawLine(90,100,180,180);
             g.fillRect(70,70,45,45);

             g.setColor(Color.ORANGE);
             g.drawString("This is a string at postion 200,200",200,200);
             g.drawOval(210,210,30,60);
             g.fillOval(250,260,40,50);
}
public static void main(String args[]){
  Shapes s=new Shapes();}
```



JAVA 2D

- New features added to shapes drawn
- Shading
- Fill with graphic
- Dashed lines

```
//sample program to draw 2D objects
import java.awt.FlowLayout;
import java.awt.geom.*;
import javax.swing.JFrame;
import java.awt.Graphics;
import java.awt.Graphics2D;
public class Shapes1 extends JFrame{
  Shapes1(){
              //FRAME SETTINGS
             setSize(400,400);
             setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
             setLayout(new FlowLayout());
              setVisible(true);
             pack();
             }
             public void paint(Graphics g){
             Graphics2D gg=(Graphics2D)g;
             gg.setColor(Color.BLUE);
             Rectangle2D.Double rect=new Rectangle2D.Double(100,100,50,60);
             gg.draw(rect);
             gg.setColor(Color.GREEN);
             Rectangle2D.Double rect1=new Rectangle2D.Double(50,50,20,40);
             gg.fill(rect1);
}
public static void main(String args[]){
```

Java 3D API

- Three-dimensional
- Extends Graphics class to support enhanced features.
- Supports following capabilities for Java programs.
  - graphics –drawXxx() and fillXxx()
  - text –drawString()
  - Imaging- drawImage()
- Java.awt.Graphics supports
  - draw method
  - fill method
  - set method –setFont() and setColor() define how to draw and fill

Java 3D API in the application

- To employ Java 3D API in the application

  public void paint (Graphics g) {

  g. drawXxx(); ... }

- 3D spans
  - `java.awt.image`
  - `java.awt.color`
  - `java.awt.font`
  - `java.awt.geom`

Jframe

```
setLayout(new FlowLayout());
setSize(400,400);
setVisible(true);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
getContentPane().setBackground(Color.GREEN);
pack();
```

draw3DRect()

```
public void draw3DRect(int x,
                       int y,
                       int width,
                       int height,
                       boolean raised)
```

Draws a 3-D highlighted outline of the specified rectangle. The edges of the rectangle are highlighted so that they appear to be beveled and lit from the upper left corner.

The colors used for the highlighting effect are determined based on the current color. The resulting rectangle covers an area that is `width + 1` pixels wide by `height + 1` pixels tall.

Parameters:

`x` - the *x* coordinate of the rectangle to be drawn.

`y` - the *y* coordinate of the rectangle to be drawn.

`width` - the width of the rectangle to be drawn.

`height` - the height of the rectangle to be drawn.

`raised` - a boolean that determines whether the rectangle appears to be raised above the surface or sunk into the surface.

fill3DRect()

```
public void fill3DRect(int x,
                       int y,
                       int width,
                       int height,
                       boolean raised)
```

Paints a 3-D highlighted rectangle filled with the current color. The edges of the rectangle will be highlighted so that it appears as if the edges were beveled and lit from the upper left corner.

Parameters:

`x` - the *x* coordinate of the rectangle to be filled.

`y` - the *y* coordinate of the rectangle to be filled.

`width` - the width of the rectangle to be filled.

`height` - the height of the rectangle to be filled.

`raised` - a boolean value that determines whether the rectangle appears to be raised above the surface or etched into the surface

drawstring()

```
public abstract void drawString(String str,
                                int x,
                                int y)
```

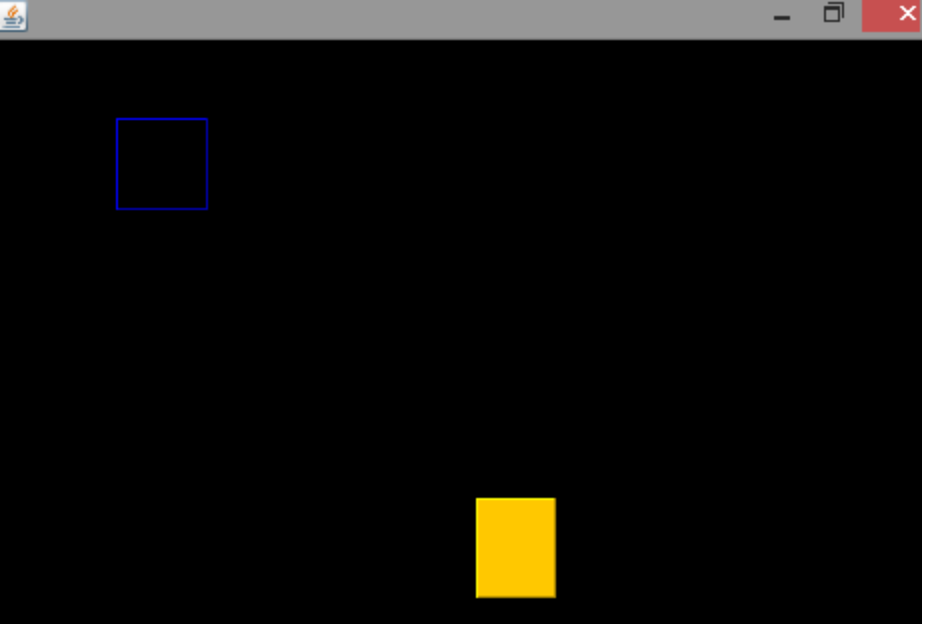
Draws the text given by the specified string, using this graphics context's current font and color. The baseline of the leftmost character is at position (*x*, *y*) in this graphics context's coordinate system.

Parameters:

`str` - the string to be drawn.

`x` - the *x* coordinate.

`y` - the *y* coordinate.



```
import java.awt.Color;
import java.awt.Font;
import java.awt.Container;
import java.awt.FlowLayout;
import javax.swing.JFrame;
import java.awt.Graphics;
public class Shapes1 extends JFrame{
  Shapes1(){
            //FRAME SETTINGS
             setSize(400,400);
             setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
             setLayout(new FlowLayout());
              setVisible(true);
             pack();
             }
             public void paint(Graphics g){
             g.setColor(Color.BLUE);
             g.draw3DRect(70,70,45,45,true);
             g.setColor(Color.ORANGE);
             g.fill3DRect(250,260,40,50,true);
}
public static void main(String args[]){
  Shapes1 s=new Shapes1();}
}
```

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class AnAppletWithTextFields extends Applet implements ActionListener {
        TextField textField1, textField2;
        Button swapEm;
         public void init() {
                Label label1 = new Label("First text field: ");
                textField1 = new TextField("Hello");
                swapEm = new Button("Swap");
                swapEm.addActionListener(this);
                Label label2 = new Label("Second text field: ");
                textField2 = new TextField("Goodbye");
                textField2.setEditable(false);
                add(label1); add(textField1);
                add(label2); add(textField2); add(swapEm);
        }
        public void actionPerformed(ActionEvent e) {
                String temp = textField1.getText();
                textField1.setText(textField2.getText());
                textField2.setText(temp);
        }
        }
```