AWT GRAPHICS CLASS

Introduction

The Graphics class is the abstract super class for all graphics contexts which allow an application to draw onto components that can be realized on various devices, or onto off-screen images as well.

A Graphics object encapsulates all state information required for the basic rendering operations that Java supports. State information includes the following properties.

- The Component object on which to draw.
- A translation origin for rendering and clipping coordinates.
- The current clip.
- The current color.
- The current font.
- The current logical pixel operation function.
- The current XOR alternation color

Class declaration

Following is the declaration for **java.awt.Graphics** class:

```
public abstract class Graphics
   extends Object
```

Class constructors

S.N.	Constructor & Description
1	**Graphics** Constructs a new Graphics object.

Class methods

S.N.	Method & Description
1	**abstract void clearRect***intx, inty, intwidth, intheight* Clears the specified rectangle by filling it with the background color of the current drawing surface.
2	**abstract void clipRect***intx, inty, intwidth, intheight* Intersects the current clip with the specified rectangle.
3	

abstract void copyArea*intx, inty, intwidth, intheight, intdx, intdy*

Copies an area of the component by a distance specified by dx and dy.

4

abstract Graphics create

Creates a new Graphics object that is a copy of this Graphics object.

5

Graphics create*intx, inty, intwidth, intheight*

Creates a new Graphics object based on this Graphics object, but with a new translation and clip area.

6

abstract void dispose

Disposes of this graphics context and releases any system resources that it is using.

7

void draw3DRect*intx, inty, intwidth, intheight, booleanraised*

Draws a 3-D highlighted outline of the specified rectangle.

8

abstract void drawArc*intx, inty, intwidth, intheight, intstartAngle, intarcAngle*

Draws the outline of a circular or elliptical arc covering the specified rectangle.

9

void drawBytes*byte[]data, intoffset, intlength, intx, inty*

Draws the text given by the specified byte array, using this graphics context's current font and color.

10

void drawChars*char[]data, intoffset, intlength, intx, inty*

Draws the text given by the specified character array, using this graphics context's current font and color.

11

abstract boolean drawImage*Imageimg, intx, inty, Colorbgcolor, ImageObserverobserver*

Draws as much of the specified image as is currently available.

12

abstract boolean drawImage*Imageimg, intx, inty, ImageObserverobserver*

Draws as much of the specified image as is currently available.

13

abstract boolean drawImage
Imageimg, intx, inty, intwidth, intheight, Colorbgcolor, ImageObserverobserver

Draws as much of the specified image as has already been scaled to fit inside the specified rectangle.

14

abstract boolean drawImage*Imageimg, intx, inty, intwidth, intheight, ImageObserverobserver*

Draws as much of the specified image as has already been scaled to fit inside the specified rectangle.

15

abstract boolean drawImage
Imageimg, intdx*1*, intdy*1*, intdx*2*, intdy*2*, intsx*1*, intsy*1*, intsx*2*, intsy*2*, Colorbgcolor, ImageObserverobserver

Draws as much of the specified area of the specified image as is currently available, scaling it on the fly to fit inside the specified area of the destination drawable surface.

16

abstract boolean drawImage
Imageimg, intdx*1*, intdy*1*, intdx*2*, intdy*2*, intsx*1*, intsy*1*, intsx*2*, intsy*2*, ImageObserverobserver

Draws as much of the specified area of the specified image as is currently available, scaling it on the fly to fit inside the specified area of the destination drawable surface.

17

abstract void drawLine*intx*1*, inty*1*, intx*2*, inty*2

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

18

abstract void drawOval*intx, inty, intwidth, intheight*

Draws the outline of an oval.

19

abstract void drawPolygon*int*[]*xPoints, int*[]*yPoints, intnPoints*

Draws a closed polygon defined by arrays of x and y coordinates.

20

void drawPolygon*Polygonp*

Draws the outline of a polygon defined by the specified Polygon object.

21

abstract void drawPolyline*int*[]*xPoints, int*[]*yPoints, intnPoints*

Draws a sequence of connected lines defined by arrays of x and y coordinates.

22

void drawRect*intx, inty, intwidth, intheight*

Draws the outline of the specified rectangle.

23

abstract void drawRoundRect*intx, inty, intwidth, intheight, intarcWidth, intarcHeight*

Draws an outlined round-cornered rectangle using this graphics context's current color.

24

abstract void drawString*AttributedCharacterIteratoriterator, intx, inty*

Renders the text of the specified iterator applying its attributes in accordance with the

specification of the TextAttribute class.

25

abstract void drawString*Stringstr, intx, inty*

Draws the text given by the specified string, using this graphics context's current font and color.

26

void fill3DRect*intx, inty, intwidth, intheight, booleanraised*

Paints a 3-D highlighted rectangle filled with the current color.

27

abstract void fillArc*intx, inty, intwidth, intheight, intstartAngle, intarcAngle*

Fills a circular or elliptical arc covering the specified rectangle.

28

abstract void fillOval*intx, inty, intwidth, intheight*

Fills an oval bounded by the specified rectangle with the current color.

29

abstract void fillPolygon*int[]xPoints, int[]yPoints, intnPoints*

Fills a closed polygon defined by arrays of x and y coordinates.

30

void fillPolygon*Polygonp*

Fills the polygon defined by the specified Polygon object with the graphics context's current color.

31

abstract void fillRect*intx, inty, intwidth, intheight*

Fills the specified rectangle.

32

abstract void fillRoundRect*intx, inty, intwidth, intheight, intarcWidth, intarcHeight*

Fills the specified rounded corner rectangle with the current color.

33

void finalize

Disposes of this graphics context once it is no longer referenced.

34

abstract Shape getClip

Gets the current clipping area.

35

abstract Rectangle getClipBounds

Returns the bounding rectangle of the current clipping area.

36

Rectangle getClipBounds*Rectangler*

Returns the bounding rectangle of the current clipping area.

37

Rectangle getClipRect

Deprecated. As of JDK version 1.1, replaced by getClipBounds.

38

abstract Color getColor

Gets this graphics context's current color.

39

abstract Font getFont

Gets the current font.

40

FontMetrics getFontMetrics

Gets the font metrics of the current font.

41

abstract FontMetrics getFontMetrics*Fontf*

Gets the font metrics for the specified font.

42

boolean hitClip*intx, inty, intwidth, intheight*

Returns true if the specified rectangular area might intersect the current clipping area.

43

abstract void setClip*intx, inty, intwidth, intheight*

Sets the current clip to the rectangle specified by the given coordinates.

44

abstract void setClip*Shapeclip*

Sets the current clipping area to an arbitrary clip shape.

45

abstract void setColor*Colorc*

Sets this graphics context's current color to the specified color.

46

abstract void setFont*Fontfont*

Sets this graphics context's font to the specified font.

47

abstract void setPaintMode

Sets the paint mode of this graphics context to overwrite the destination with this graphics context's current color.

48

abstract void setXORMode*Colorc1*

Sets the paint mode of this graphics context to alternate between this graphics context's current color and the new specified color.

49

String toString

Returns a String object representing this Graphics object's value.

50

abstract void translate*intx, inty*

Translates the origin of the graphics context to the point x, y in the current coordinate system.

Methods inherited

This class inherits methods from the following classes:

- java.lang.Object

Graphics Example

Create the following java program using any editor of your choice in say **D:/ > AWT > com > tutorialspoint > gui >**

AWTGraphicsDemo.java

```
package com.tutorialspoint.gui;

import java.awt.*;
import java.awt.event.*;
import java.awt.geom.*;

public class AWTGraphicsDemo extends Frame {

   public AWTGraphicsDemo(){
      super("Java AWT Examples");
      prepareGUI();
   }

   public static void main(String[] args){
      AWTGraphicsDemo  awtGraphicsDemo = new AWTGraphicsDemo();
      awtGraphicsDemo.setVisible(true);
   }

   private void prepareGUI(){
      setSize(400,400);
      addWindowListener(new WindowAdapter() {
         public void windowClosing(WindowEvent windowEvent){
            System.exit(0);
         }
      });
   }

   @Override
   public void paint(Graphics g) {
      g.setColor(Color.GRAY);
      Font font = new Font("Serif", Font.PLAIN, 24);
      g.setFont(font);
      g.drawString("Welcome to TutorialsPoint", 50, 150);
   }
```

```
}
```

Compile the program using command prompt. Go to **D:/ > AWT** and type the following command.

```
D:\AWT>javac com\tutorialspoint\gui\AWTGraphicsDemo.java
```

If no error comes that means compilation is successful. Run the program using following command.

```
D:\AWT>java com.tutorialspoint.gui.AWTGraphicsDemo
```

Verify the following output