Drawing lines examples with Java Graphics2D

Written by Nam Ha Minh
Last Updated on 10 August 2019 | Print Email

In this Java graphics tutorial, you will learn how to draw lines with various code examples.

A line is a graphics primitive that connects two points. In Java, to draw a line between two points `(x1, y1)` and `(x2, y2)` onto graphics context represented by a `Graphics`object, use the following method:

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

If a `Graphics2D` object is used, the following method is more object-oriented:

```
draw(Line2D)
```

With two implementations of `Line2D`: `Line2D.Double` (with double coordinates) and `Line2D.Float` (with float coordinates).

Both methods draw a line using the current graphic attributes (paint color, stroke, rendering hints, etc). Here is a quick example (suppose that `g2d` is a reference of a `Graphics2D` object):

```
// draw a line in integer coordinates
g2d.drawLine(20, 50, 460, 50);

// draw a line in double coordinates
g2d.draw(new Line2D.Double(21.5d, 199.8d, 459.5d, 199.8d));

// draw a line in float coordinates
g2d.draw(new Line2D.Float(21.50f, 232.50f, 459.50f, 232.50f));
```

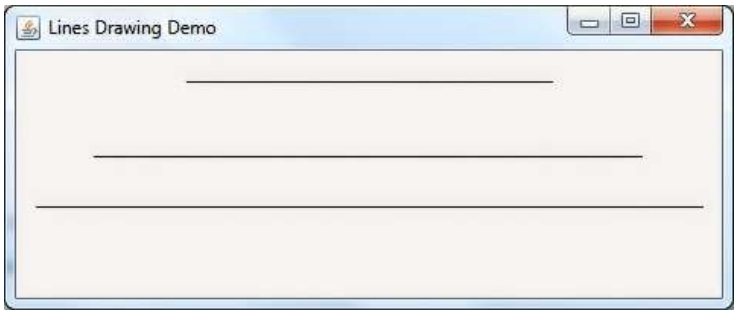
The following source code is a Swing program that draws three lines onto the graphics context of the `JFrame` window:

```
package net.codejava.graphics;

import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.geom.Line2D;

import javax.swing.JFrame;
import javax.swing.SwingUtilities;

/**
 * This program demonstrates how to draw lines using Graphics2D object.
 * @author www.codejava.net
 *
 */
public class LinesDrawingExample extends JFrame {

    public LinesDrawingExample() {
        super("Lines Drawing Demo");

        setSize(480, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
    }

    void drawLines(Graphics g) {
        Graphics2D g2d = (Graphics2D) g;

        g2d.drawLine(120, 50, 360, 50);

        g2d.draw(new Line2D.Double(59.2d, 99.8d, 419.1d, 99.8d));

        g2d.draw(new Line2D.Float(21.50f, 132.50f, 459.50f, 132.50f));

    }

    public void paint(Graphics g) {
        super.paint(g);
        drawLines(g);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                new LinesDrawingExample().setVisible(true);
            }
        });
    }
}
```

Here, the key point is to override the `paint()` method from the superclass (`JFrame`) so that we can obtain the graphics context:

```
public void paint(Graphics g) {
    super.paint(g);
    drawLines(g);
}
```

And here is the screenshot of the program:



Now, let's see more complex examples in which we will alter the graphics attributes in order to decorate the lines drawn. Note that the graphics attributes must be set before drawing the lines.

Setting paint color and thickness for the lines

To specify a specific color for the line, call **`setColor(Color)`** method before drawing, for example:

```
g2d.setColor(Color.RED);
```

To specify thickness for the line, we can create a basic stroke with a specified width as follows:

```
// creates a solid stroke with line width is 2
Stroke stroke = new BasicStroke(2f);
```

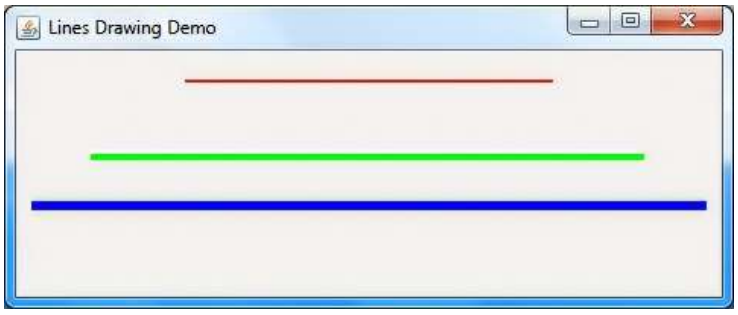
Then set this stroke for the graphics context:

```
g2d.setStroke(stroke);
```

The previous example is updated as follows:

```
g2d.setColor(Color.RED);
// creates a solid stroke with line width is 2
Stroke stroke = new BasicStroke(2f);
g2d.setStroke(stroke);
g2d.drawLine(120, 50, 360, 50);

g2d.setColor(Color.GREEN);
g2d.setStroke(new BasicStroke(4f));
g2d.draw(new Line2D.Double(59.2d, 99.8d, 419.1d, 99.8d));

g2d.setColor(Color.BLUE);
g2d.setStroke(new BasicStroke(6f));
g2d.draw(new Line2D.Float(21.50f, 132.50f, 459.50f, 132.50f));
```

Result:



Drawing dashed lines:

To draw a dashed line, specify a dashing pattern when creating the stroke. For example:

```
float[] dashingPattern1 = {2f, 2f};
Stroke stroke1 = new BasicStroke(2f, BasicStroke.CAP_BUTT,
        BasicStroke.JOIN_MITER, 1.0f, dashingPattern1, 2.0f);

g2d.setStroke(stroke1);
g2d.drawLine(120, 50, 360, 50);
```

The dashing pattern is formed by alternating between opaque and transparent sections with the widths specified by the float array:

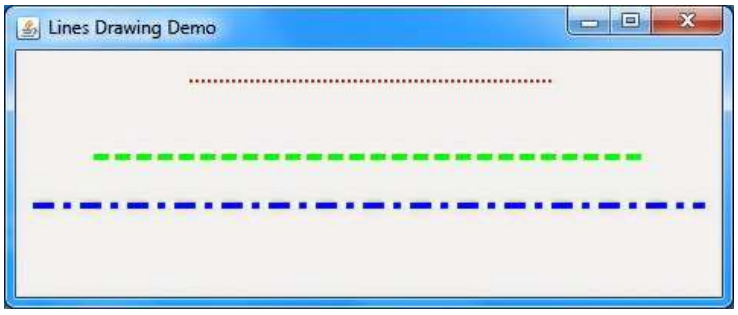
```
float[] dashingPattern1 = {2f, 2f};
```

This pattern specifies that the first two pixels are opaque; the next two are transparent; the next two are opaque; and so on…. Here's the result:

The original example is updated to draw dashed lines as follows:

```
g2d.setColor(Color.RED);

float[] dashingPattern1 = {2f, 2f};
Stroke stroke1 = new BasicStroke(2f, BasicStroke.CAP_BUTT,
        BasicStroke.JOIN_MITER, 1.0f, dashingPattern1, 2.0f);

g2d.setStroke(stroke1);
g2d.drawLine(120, 50, 360, 50);

g2d.setColor(Color.GREEN);

float[] dashingPattern2 = {10f, 4f};
Stroke stroke2 = new BasicStroke(4f, BasicStroke.CAP_BUTT,
        BasicStroke.JOIN_MITER, 1.0f, dashingPattern2, 0.0f);

g2d.setStroke(stroke2);
g2d.draw(new Line2D.Double(59.2d, 99.8d, 419.1d, 99.8d));

g2d.setColor(Color.BLUE);

float[] dashingPattern3 = {10f, 10f, 1f, 10f};
Stroke stroke3 = new BasicStroke(4f, BasicStroke.CAP_SQUARE,
        BasicStroke.JOIN_MITER, 1.0f, dashingPattern3, 0.0f);

g2d.setStroke(stroke3);
g2d.draw(new Line2D.Float(21.50f, 132.50f, 459.50f, 132.50f));
```

Here's the result:



Decorating line end caps

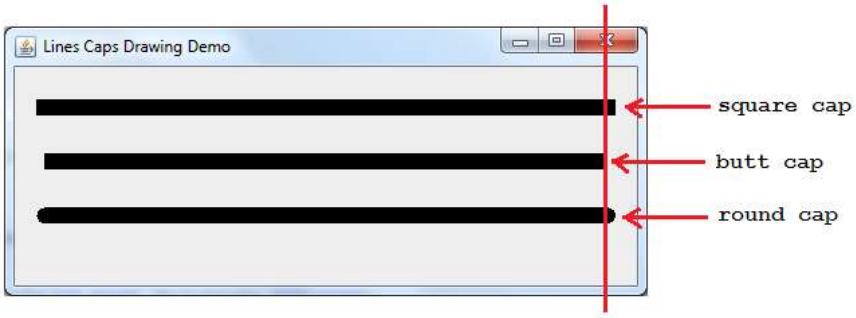
Caps are decorations applied at both ends of a line (solid, unclosed-path line) or dash segments in a dashed line. The `BasicStroke` class provides three cap styles: `CAP_SQUARE` (default), `CAP_BUTT` and `CAP_ROUND`. The following example creates three strokes with different cap styles:

```
// this stroke with default CAP_SQUARE and JOIN_MITER
Stroke stroke1 = new BasicStroke(12f);

// this stroke with CAP_BUTT
Stroke stroke2 = new BasicStroke(12f, BasicStroke.CAP_BUTT, BasicStroke.

// this stroke with CAP_ROUND
Stroke stroke3 = new BasicStroke(12f, BasicStroke.CAP_ROUND, BasicStroke
```

And the following code snippet draws three lines with these strokes:

```
g2d.setStroke(stroke1);
g2d.drawLine(30, 60, 450, 60);

g2d.setStroke(stroke2);
g2d.drawLine(30, 100, 450, 100);

g2d.setStroke(stroke3);
g2d.drawLine(30, 140, 450, 140);
```

Here's the result:



Decorating line joins

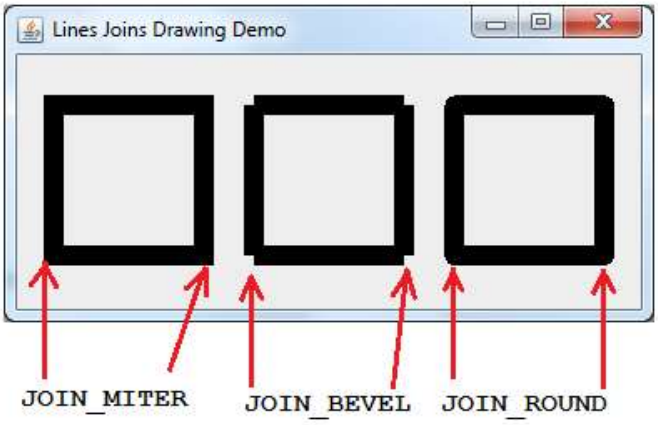
Joins are decorations applied at intersections between lines. The `BasicStroke` class provides three join styles: `JOIN_MITER` (default), `JOIN_BEVEL` and `JOIN_ROUND`. The following example creates three strokes with different join styles:

```
// this stroke with default CAP_SQUARE and JOIN_MITER
Stroke stroke1 = new BasicStroke(12f);

// this stroke with JOIN_BEVEL
Stroke stroke2 = new BasicStroke(12f, BasicStroke.CAP_SQUARE, BasicStrok

// this stroke with JOIN_ROUND
Stroke stroke3 = new BasicStroke(12f, BasicStroke.CAP_SQUARE, BasicStrok
```

And the following code snippet draws three rectangles using lines to demonstrate the concept of join decorations:

```
// draws the first rectangle with a stroke of JOIN_MITER
g2d.setStroke(stroke1);
g2d.drawLine(30, 60, 120, 60);
g2d.drawLine(30, 60, 30, 150);
g2d.drawLine(120, 60, 120, 150);
g2d.drawLine(120, 60, 120, 150);
g2d.drawLine(30, 150, 120, 150);

// draws the second rectangle with a stroke of JOIN_BEVEL
g2d.setStroke(stroke2);
g2d.drawLine(30 + 120, 60, 120 + 120, 60);
g2d.drawLine(30 + 120, 60, 30 + 120, 150);
g2d.drawLine(120 + 120, 60, 120 + 120, 150);
g2d.drawLine(120 + 120, 60, 120 + 120, 150);
g2d.drawLine(30 + 120, 150, 120 + 120, 150);

// draws the third rectangle with a stroke of JOIN_ROUND
g2d.setStroke(stroke3);
g2d.drawLine(30 + 240, 60, 120 + 240, 60);
g2d.drawLine(30 + 240, 60, 30 + 240, 150);
g2d.drawLine(120 + 240, 60, 120 + 240, 150);
g2d.drawLine(120 + 240, 60, 120 + 240, 150);
g2d.drawLine(30 + 240, 150, 120 + 240, 150);
```

Here's the result:



API References:

- Graphics2D Javadoc
- BasicStroke Javadoc
- Line2D.Double Javadoc
- Line2D.Float Javadoc

Other Java Graphics Tutorials:

- How to add watermark for images using Java
- How to resize images using Java
- How to convert image format using Java
- How to draw image with automatic scaling in Java
- How to capture screenshot programmatically in Java
- How to draw text vertically with Java Graphics2D
- How to Create Zoomable User Interface Java Programs with Piccolo2D Framework
- Drawing Rectangles Examples with Java Graphics2D
- Using JFreechart to draw line chart with CategoryDataset
- Using JFreechart to draw XY line chart with XYDataset

About the Author:

Nam Ha Minh is certified Java programmer (SCJP and SCWCD). He started programming with Java in the time of Java 1.4 and has been falling in love with Java since then. Make friend with him on Facebook and watch his Java videos you YouTube.