Draw SVG path

SVG **(Scalable Vector Graphics)** is an XML based language to define vector based graphics. In JavaFX we can construct images by parsing SVG paths. Such shapes are represented by the class named **SVGPath**. This class belongs to the package **javafx.scene.shape**. By instantiating this class, you can create a node which is created by parsing an SVG path in JavaFX.

This class has a property named **content** of String datatype. This represents the SVG Path encoded string, from which the image should be drawn.

To draw a shape by parsing an SVG path, you need to pass values to this property, using the method named **setContent()** of this class as follows −

```
setContent(value);
```

```
package javafxapplication30;

import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.Group;
import javafx.scene.paint.Color;
import javafx.scene.shape.QuadCurve;

public class JavaFXApplication30 extends Application {

   @Override
   public void start(Stage stage) {
 //Creating a QuadCurve
    QuadCurve quadCurve = new QuadCurve();

    //Adding properties to the Quad Curve
    quadCurve.setStartX(100.0);
    quadCurve.setStartY(220.0);
    quadCurve.setEndX(500.0);
    quadCurve.setEndY(220.0);
    quadCurve.setControlX(250.0);
    quadCurve.setControlY(0.0);
    quadCurve.setFill(Color.AQUA);
    quadCurve.setStrokeWidth(12);
    quadCurve.setStroke(Color.BLACK);

    //Creating a Group object
    Group root = new Group(quadCurve);
```

```
        //Creating a scene object
        Scene scene = new Scene(root, 600, 300);

        //Setting title to the Stage
        stage.setTitle("Drawing a Quad curve");

        //Adding scene to the stage
        stage.setScene(scene);

        //Displaying the contents of the stage
        stage.show();
    }
}
```

Properties of 2D Objects

For all the 2-Dimensional objects, you can set various properties like fill, stroke, StrokeType, etc. The following section discusses various properties of 2D objects.

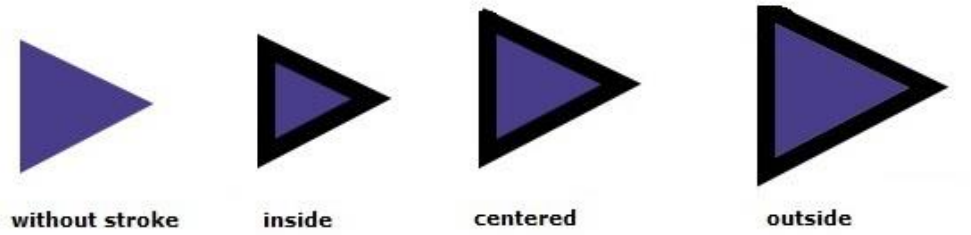
Stroke Type : This property is of the type StrokeType. It represents the position of the boundary line applied to the shape. You can set the type of the stroke using the method **setStrokeType()** as follows −

```
Path.setStrokeType(StrokeType.CENTERED);
```

The stroke type of a shape can be −

- **Inside** − The boundary line will be drawn inside the edge (outline) of the shape (StrokeType.INSIDE).
- **Outside** − The boundary line will be drawn outside the edge (outline) of the shape (StrokeType.OUTSIDE).
- **Centered** − The boundary line will be drawn in such a way that the edge (outline) of the shape passes exactly thorough the center of the line (StrokeType.CENTERED).

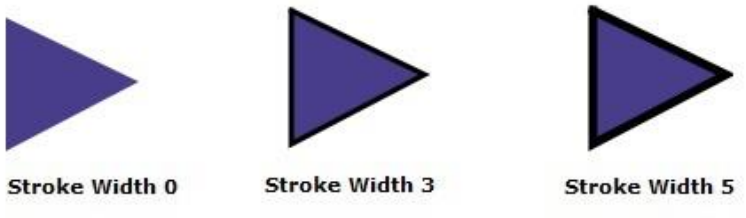
By default, the stroke type of a shape is centered. Following is the diagram of a triangle with different Stroke Types –



Stroke Width : This property is of the type double and it represents the width of the boundary line of the shape. You can set the stroke width using the method **setStrokeWidth()** as follows −

```
Path.setStrokeWidth(3.0)
```

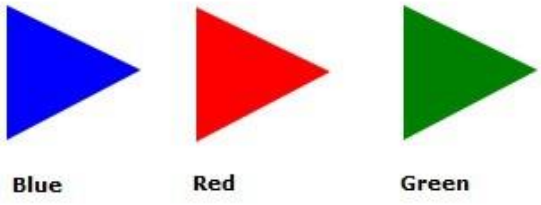
By default, the value of the stroke with of a shape is **1.0**. Following is a diagram of a triangle with different values of stroke width.



Stroke Fill : This property is of the type Paint and it represents the color that is to be filled inside the shape. You can set the fill color of a shape using the method **setFill()** as follows −

```
path.setFill(COLOR.BLUE);
```

By default, the value of the stroke color is **BLACK**. Following is a diagram of a triangle with different colors.



Stroke : This property is of the type Paint and it represents the color of the boundary line of the shape. You can set a value to this property using the method **setStroke()** as shown below −

```
path.setStroke(Color.RED)
```

By default, the color of the stroke is black. Following is a diagram of a triangle with different stroke colors.

Stroke Line : This property is of the type **StrokeLineJoin**, it represents the type of joining that is used at the edges of the shape. You can set the line join of the stroke using the method **setStrokeLineJoin()** as follows −

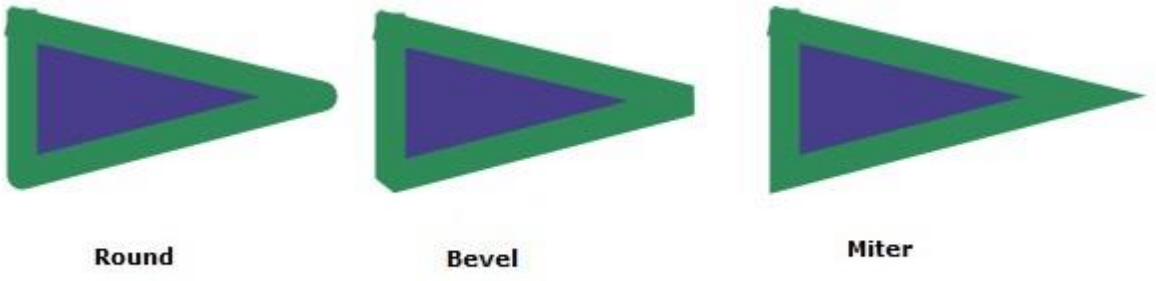
```
path.setStrokeLineJoin(StrokeLineJoin.BEVEL);
```

The stroke line join can be −

- **Bevel** − The bevel join is applied to the joining of the edges of the shape (StrokeLineJoin.BEVEL).
- **Miter** − The miter join is applied to the joining of the edges of the shape (StrokeLineJoin.MITER).

- **Round** – The round join is applied to the joining of the edges of the shape (StrokeLineJoin.ROUND).

By default, the Stroke Line Joining a shape is miter. Following is a diagram of a triangle with different line join types –

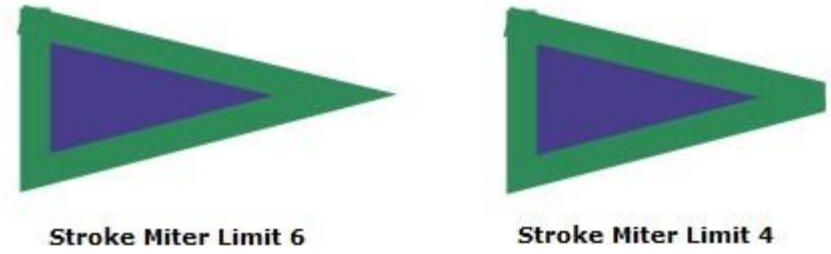


Stroke Miter Limit : This property is of the type double. It represents the limit for the distance between the inside point of the joint and the outside point of the joint. If the distance between these two points exceeds the given limit, the miter is cut at the edge.

You can set value to this property using the method **setStroke()** as follows –

```
path.setStrokeMiterLimit(4);
```

By default, the stroke miter limit value id 10 of the stroke is black. Following is a diagram of a triangle with different stroke limits.



Stroke Line Cap : This property is of the type **StrokeLineCap** It represents the end cap style of the line. You can set the line cap stroke using the method **setStrokeLineCap()** as shown in the following code block –

```
line.setStrokeLineCap(StrokeLineCap.SQUARE);
```

The stroke line cap can be –

- **Butt** – The butt line cap is applied at the end of the lines (StrokeLineCap.BUTT).
- **Square** – The square line cap is applied at the end of the lines (StrokeLineCap.SQUARE).
- **Round** – The round line cap is applied at the end of the lines (StrokeLineCap.ROUND).

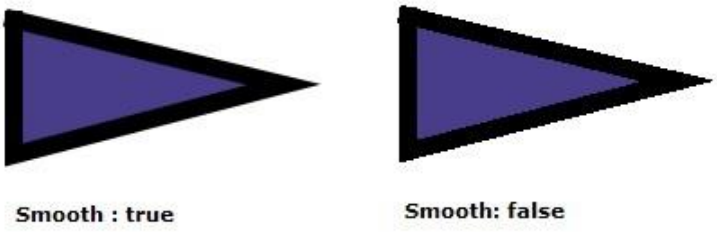
By default, the Stroke Line cap a shape is square. Following is the diagram of a triangle with different line cap types.

Smooth : This property is of the type Boolean. If this value is true, then the edges of the shape will be smooth.

You can set value to this property using the method **setSmooth()** as follows –

```
path.setSmooth(false);
```

By default, the smooth value is true. Following is a diagram of a triangle with both smooth values.



Smooth : true

Smooth: false