AWT QUADCURVE2D CLASS

Introduction

The QuadCurve2D class states a quadratic parametric curve segment in x, y coordinate space.

Class declaration

Following is the declaration for **java.awt.geom.QuadCurve2D** class:

```
public abstract class QuadCurve2D
   extends Object
      implements Shape, Cloneable
```

Class constructors

S.N.	Constructor & Description
1	**protected QuadCurve2D** This is an abstract class that cannot be instantiated directly.

Class methods

S.N.	Method & Description
1	**Object clone** Creates a new object of the same class and with the same contents as this object.
2	**boolean contains*doublex, doubley*** Tests if the specified coordinates are inside the boundary of the Shape.
3	**boolean contains*doublex, doubley, doublew, doubleh*** Tests if the interior of the Shape entirely contains the specified rectangular area.
4	**boolean contains*Point2Dp*** Tests if a specified Point2D is inside the boundary of the Shape.
5	**boolean contains*Rectangle2Dr*** Tests if the interior of the Shape entirely contains the specified Rectangle2D.
6	**Rectangle getBounds**

Returns an integer Rectangle that completely encloses the Shape.

7

abstract Point2D getCtrlPt

Returns the control point.

8

abstract double getCtrlX

Returns the X coordinate of the control point in double precision.

9

abstract double getCtrlY

Returns the Y coordinate of the control point in double precision.

10

doublegetFlatness

Returns the flatness, or maximum distance of a control point from the line connecting the end points, of this QuadCurve2D.

11

static double getFlatness*double***[]***coords, intoffset*

Returns the flatness, or maximum distance of a control point from the line connecting the end points, of the quadratic curve specified by the control points stored in the indicated array at the indicated index.

12

static double getFlatness*double***x1, *doubley*1, *doublectrlx*, *doublectrly*, *double***x2, *doubley*2**

Returns the flatness, or maximum distance of a control point from the line connecting the end points, of the quadratic curve specified by the indicated control points.

13

double getFlatnessSq

Returns the square of the flatness, or maximum distance of a control point from the line connecting the end points, of this QuadCurve2D.

14

static double getFlatnessSq*double***[]***coords, intoffset*

Returns the square of the flatness, or maximum distance of a control point from the line connecting the end points, of the quadratic curve specified by the control points stored in the indicated array at the indicated index.

15

static double getFlatnessSq*double***x1, *doubley*1, *doublectrlx*, *doublectrly*, *double***x2, *doubley*2**

Returns the square of the flatness, or maximum distance of a control point from the line connecting the end points, of the quadratic curve specified by the indicated control points.

16

abstract Point2D getP1

Returns the start point.

17 **abstract Point2D getP2**

Returns the end point.

18 **PathIterator getPathIterator***AffineTransformat*

Returns an iteration object that defines the boundary of the shape of this QuadCurve2D.

19 **PathIterator getPathIterator***AffineTransformat, doubleflatness*

Returns an iteration object that defines the boundary of the flattened shape of this QuadCurve2D.

20 **abstract double getX1**

Returns the X coordinate of the start point in double in precision.

21 **abstract double getX2**

Returns the X coordinate of the end point in double precision.

22 **abstract double getY1**

Returns the Y coordinate of the start point in double precision.

23 **abstract double getY2**

Returns the Y coordinate of the end point in double precision.

24 **boolean intersects***doublex, doubley, doublew, doubleh*

Tests if the interior of the Shape intersects the interior of a specified rectangular area.

25 **boolean intersects***Rectangle2Dr*

Tests if the interior of the Shape intersects the interior of a specified Rectangle2D.

26 **void setCurve***double***[]***coords, intoffset*

Sets the location of the end points and control points of this QuadCurve2D to the double coordinates at the specified offset in the specified array.

27 **abstract void setCurve***doublex***1,** *doubley***1,** *doublectrlx, doublectrly, doublex***2,** *doubley***2**

Sets the location of the end points and control point of this curve to the specified double coordinates.

28

void setCurve*Point2D*[]*pts, intoffset*

Sets the location of the end points and control points of this QuadCurve2D to the coordinates of the Point2D objects at the specified offset in the specified array.

29

void setCurve*Point2Dp*1, *Point2Dcp*, *Point2Dp*2

Sets the location of the end points and control point of this QuadCurve2D to the specified Point2D coordinates.

30

void setCurve*QuadCurve2Dc*

Sets the location of the end points and control point of this QuadCurve2D to the same as those in the specified QuadCurve2D.

31

static int solveQuadratic*double*[]*eqn*

Solves the quadratic whose coefficients are in the eqn array and places the non-complex roots back into the same array, returning the number of roots.

32

static int solveQuadratic*double*[]*eqn*, *double*[]*res*

Solves the quadratic whose coefficients are in the eqn array and places the non-complex roots into the res array, returning the number of roots.

33

static void subdivide*double*[]*src*, *intsrcoff*, *double*[]*left*, *intleftoff*, *double*[]*right*, *intrightoff*

Subdivides the quadratic curve specified by the coordinates stored in the src array at indices srcoff through srcoff + 5 and stores the resulting two subdivided curves into the two result arrays at the corresponding indices.

34

void subdivide*QuadCurve2Dleft*, *QuadCurve2Dright*

Subdivides this QuadCurve2D and stores the resulting two subdivided curves into the left and right curve parameters.

35

static void subdivide*QuadCurve2Dsrc*, *QuadCurve2Dleft*, *QuadCurve2Dright*

Subdivides the quadratic curve specified by the src parameter and stores the resulting two subdivided curves into the left and right curve parameters.

Methods inherited

This class inherits methods from the following classes:

- java.lang.Object

QuadCurve2D Example

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

AWTGraphicsDemo

```
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) {
      QuadCurve2D shape = new QuadCurve2D.Double();
      shape.setCurve(250D,250D,100D,100D,200D,150D);
      Graphics2D g2 = (Graphics2D) g;
      g2.draw (shape);
      Font font = new Font("Serif", Font.PLAIN, 24);
      g2.setFont(font);
      g.drawString("Welcome to TutorialsPoint", 50, 70);
      g2.drawString("QuadCurve2D.Curve", 100, 120);
   }
}
```

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