| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/CubicCurve2D.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/awt/geom/Area.html)   [**NEXT CLASS**](http://docs.google.com/java/awt/geom/CubicCurve2D.Double.html) | [**FRAMES**](http://docs.google.com/index.html?java/awt/geom/CubicCurve2D.html)    [**NO FRAMES**](http://docs.google.com/CubicCurve2D.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | FIELD | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: FIELD | [CONSTR](#4d34og8) | [METHOD](#17dp8vu) |

## **java.awt.geom**

Class CubicCurve2D

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 **java.awt.geom.CubicCurve2D**

**All Implemented Interfaces:** [Shape](http://docs.google.com/java/awt/Shape.html), [Cloneable](http://docs.google.com/java/lang/Cloneable.html) **Direct Known Subclasses:** [CubicCurve2D.Double](http://docs.google.com/java/awt/geom/CubicCurve2D.Double.html), [CubicCurve2D.Float](http://docs.google.com/java/awt/geom/CubicCurve2D.Float.html)

public abstract class **CubicCurve2D**extends [Object](http://docs.google.com/java/lang/Object.html)implements [Shape](http://docs.google.com/java/awt/Shape.html), [Cloneable](http://docs.google.com/java/lang/Cloneable.html)

The CubicCurve2D class defines a cubic parametric curve segment in (x,y) coordinate space.

This class is only the abstract superclass for all objects which store a 2D cubic curve segment. The actual storage representation of the coordinates is left to the subclass.

**Since:** 1.2

| **Nested Class Summary** | |
| --- | --- |
| static class | [**CubicCurve2D.Double**](http://docs.google.com/java/awt/geom/CubicCurve2D.Double.html)            A cubic parametric curve segment specified with double coordinates. |
| static class | [**CubicCurve2D.Float**](http://docs.google.com/java/awt/geom/CubicCurve2D.Float.html)            A cubic parametric curve segment specified with float coordinates. |

| **Constructor Summary** | |
| --- | --- |
| protected | [**CubicCurve2D**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#CubicCurve2D())()            This is an abstract class that cannot be instantiated directly. |

| **Method Summary** | |
| --- | --- |
| [Object](http://docs.google.com/java/lang/Object.html) | [**clone**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#clone())()            Creates a new object of the same class as this object. |
| boolean | [**contains**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#contains(double,%20double))(double x, double y)            Tests if the specified coordinates are inside the boundary of the Shape. |
| boolean | [**contains**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#contains(double,%20double,%20double,%20double))(double x, double y, double w, double h)            Tests if the interior of the Shape entirely contains the specified rectangular area. |
| boolean | [**contains**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#contains(java.awt.geom.Point2D))([Point2D](http://docs.google.com/java/awt/geom/Point2D.html) p)            Tests if a specified [Point2D](http://docs.google.com/java/awt/geom/Point2D.html) is inside the boundary of the Shape. |
| boolean | [**contains**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#contains(java.awt.geom.Rectangle2D))([Rectangle2D](http://docs.google.com/java/awt/geom/Rectangle2D.html) r)            Tests if the interior of the Shape entirely contains the specified Rectangle2D. |
| [Rectangle](http://docs.google.com/java/awt/Rectangle.html) | [**getBounds**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#getBounds())()            Returns an integer [Rectangle](http://docs.google.com/java/awt/Rectangle.html) that completely encloses the Shape. |
| abstract  [Point2D](http://docs.google.com/java/awt/geom/Point2D.html) | [**getCtrlP1**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#getCtrlP1())()            Returns the first control point. |
| abstract  [Point2D](http://docs.google.com/java/awt/geom/Point2D.html) | [**getCtrlP2**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#getCtrlP2())()            Returns the second control point. |
| abstract  double | [**getCtrlX1**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#getCtrlX1())()            Returns the X coordinate of the first control point in double precision. |
| abstract  double | [**getCtrlX2**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#getCtrlX2())()            Returns the X coordinate of the second control point in double precision. |
| abstract  double | [**getCtrlY1**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#getCtrlY1())()            Returns the Y coordinate of the first control point in double precision. |
| abstract  double | [**getCtrlY2**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#getCtrlY2())()            Returns the Y coordinate of the second control point in double precision. |
| double | [**getFlatness**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#getFlatness())()            Returns the flatness of this curve. |
| static double | [**getFlatness**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#getFlatness(double%5B%5D,%20int))(double[] coords, int offset)            Returns the flatness of the cubic curve specified by the control points stored in the indicated array at the indicated index. |
| static double | [**getFlatness**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#getFlatness(double,%20double,%20double,%20double,%20double,%20double,%20double,%20double))(double x1, double y1, double ctrlx1, double ctrly1, double ctrlx2, double ctrly2, double x2, double y2)            Returns the flatness of the cubic curve specified by the indicated control points. |
| double | [**getFlatnessSq**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#getFlatnessSq())()            Returns the square of the flatness of this curve. |
| static double | [**getFlatnessSq**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#getFlatnessSq(double%5B%5D,%20int))(double[] coords, int offset)            Returns the square of the flatness of the cubic curve specified by the control points stored in the indicated array at the indicated index. |
| static double | [**getFlatnessSq**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#getFlatnessSq(double,%20double,%20double,%20double,%20double,%20double,%20double,%20double))(double x1, double y1, double ctrlx1, double ctrly1, double ctrlx2, double ctrly2, double x2, double y2)            Returns the square of the flatness of the cubic curve specified by the indicated control points. |
| abstract  [Point2D](http://docs.google.com/java/awt/geom/Point2D.html) | [**getP1**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#getP1())()            Returns the start point. |
| abstract  [Point2D](http://docs.google.com/java/awt/geom/Point2D.html) | [**getP2**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#getP2())()            Returns the end point. |
| [PathIterator](http://docs.google.com/java/awt/geom/PathIterator.html) | [**getPathIterator**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#getPathIterator(java.awt.geom.AffineTransform))([AffineTransform](http://docs.google.com/java/awt/geom/AffineTransform.html) at)            Returns an iteration object that defines the boundary of the shape. |
| [PathIterator](http://docs.google.com/java/awt/geom/PathIterator.html) | [**getPathIterator**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#getPathIterator(java.awt.geom.AffineTransform,%20double))([AffineTransform](http://docs.google.com/java/awt/geom/AffineTransform.html) at, double flatness)            Return an iteration object that defines the boundary of the flattened shape. |
| abstract  double | [**getX1**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#getX1())()            Returns the X coordinate of the start point in double precision. |
| abstract  double | [**getX2**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#getX2())()            Returns the X coordinate of the end point in double precision. |
| abstract  double | [**getY1**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#getY1())()            Returns the Y coordinate of the start point in double precision. |
| abstract  double | [**getY2**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#getY2())()            Returns the Y coordinate of the end point in double precision. |
| boolean | [**intersects**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#intersects(double,%20double,%20double,%20double))(double x, double y, double w, double h)            Tests if the interior of the Shape intersects the interior of a specified rectangular area. |
| boolean | [**intersects**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#intersects(java.awt.geom.Rectangle2D))([Rectangle2D](http://docs.google.com/java/awt/geom/Rectangle2D.html) r)            Tests if the interior of the Shape intersects the interior of a specified Rectangle2D. |
| void | [**setCurve**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#setCurve(java.awt.geom.CubicCurve2D))([CubicCurve2D](http://docs.google.com/java/awt/geom/CubicCurve2D.html) c)            Sets the location of the end points and control points of this curve to the same as those in the specified CubicCurve2D. |
| void | [**setCurve**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#setCurve(double%5B%5D,%20int))(double[] coords, int offset)            Sets the location of the end points and control points of this curve to the double coordinates at the specified offset in the specified array. |
| abstract  void | [**setCurve**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#setCurve(double,%20double,%20double,%20double,%20double,%20double,%20double,%20double))(double x1, double y1, double ctrlx1, double ctrly1, double ctrlx2, double ctrly2, double x2, double y2)            Sets the location of the end points and control points of this curve to the specified double coordinates. |
| void | [**setCurve**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#setCurve(java.awt.geom.Point2D%5B%5D,%20int))([Point2D](http://docs.google.com/java/awt/geom/Point2D.html)[] pts, int offset)            Sets the location of the end points and control points of this curve to the coordinates of the Point2D objects at the specified offset in the specified array. |
| void | [**setCurve**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#setCurve(java.awt.geom.Point2D,%20java.awt.geom.Point2D,%20java.awt.geom.Point2D,%20java.awt.geom.Point2D))([Point2D](http://docs.google.com/java/awt/geom/Point2D.html) p1, [Point2D](http://docs.google.com/java/awt/geom/Point2D.html) cp1, [Point2D](http://docs.google.com/java/awt/geom/Point2D.html) cp2, [Point2D](http://docs.google.com/java/awt/geom/Point2D.html) p2)            Sets the location of the end points and control points of this curve to the specified Point2D coordinates. |
| static int | [**solveCubic**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#solveCubic(double%5B%5D))(double[] eqn)            Solves the cubic whose coefficients are in the eqn array and places the non-complex roots back into the same array, returning the number of roots. |
| static int | [**solveCubic**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#solveCubic(double%5B%5D,%20double%5B%5D))(double[] eqn, double[] res)            Solve the cubic whose coefficients are in the eqn array and place the non-complex roots into the res array, returning the number of roots. |
| void | [**subdivide**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#subdivide(java.awt.geom.CubicCurve2D,%20java.awt.geom.CubicCurve2D))([CubicCurve2D](http://docs.google.com/java/awt/geom/CubicCurve2D.html) left, [CubicCurve2D](http://docs.google.com/java/awt/geom/CubicCurve2D.html) right)            Subdivides this cubic curve and stores the resulting two subdivided curves into the left and right curve parameters. |
| static void | [**subdivide**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#subdivide(java.awt.geom.CubicCurve2D,%20java.awt.geom.CubicCurve2D,%20java.awt.geom.CubicCurve2D))([CubicCurve2D](http://docs.google.com/java/awt/geom/CubicCurve2D.html) src, [CubicCurve2D](http://docs.google.com/java/awt/geom/CubicCurve2D.html) left, [CubicCurve2D](http://docs.google.com/java/awt/geom/CubicCurve2D.html) right)            Subdivides the cubic curve specified by the src parameter and stores the resulting two subdivided curves into the left and right curve parameters. |
| static void | [**subdivide**](http://docs.google.com/java/awt/geom/CubicCurve2D.html#subdivide(double%5B%5D,%20int,%20double%5B%5D,%20int,%20double%5B%5D,%20int))(double[] src, int srcoff, double[] left, int leftoff, double[] right, int rightoff)            Subdivides the cubic curve specified by the coordinates stored in the src array at indices srcoff through (srcoff + 7) and stores the resulting two subdivided curves into the two result arrays at the corresponding indices. |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [hashCode](http://docs.google.com/java/lang/Object.html#hashCode()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [toString](http://docs.google.com/java/lang/Object.html#toString()), [wait](http://docs.google.com/java/lang/Object.html#wait()), [wait](http://docs.google.com/java/lang/Object.html#wait(long)), [wait](http://docs.google.com/java/lang/Object.html#wait(long,%20int)) |

| **Methods inherited from interface java.awt.**[**Shape**](http://docs.google.com/java/awt/Shape.html) |
| --- |
| [getBounds2D](http://docs.google.com/java/awt/Shape.html#getBounds2D()) |

| **Constructor Detail** |
| --- |

### CubicCurve2D

protected **CubicCurve2D**()

This is an abstract class that cannot be instantiated directly. Type-specific implementation subclasses are available for instantiation and provide a number of formats for storing the information necessary to satisfy the various accessor methods below.

**Since:** 1.2 **See Also:**[CubicCurve2D.Float](http://docs.google.com/java/awt/geom/CubicCurve2D.Float.html), [CubicCurve2D.Double](http://docs.google.com/java/awt/geom/CubicCurve2D.Double.html)

| **Method Detail** |
| --- |

### getX1

public abstract double **getX1**()

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

**Returns:**the X coordinate of the start point of the CubicCurve2D.**Since:** 1.2

### getY1

public abstract double **getY1**()

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

**Returns:**the Y coordinate of the start point of the CubicCurve2D.**Since:** 1.2

### getP1

public abstract [Point2D](http://docs.google.com/java/awt/geom/Point2D.html) **getP1**()

Returns the start point.

**Returns:**a Point2D that is the start point of the CubicCurve2D.**Since:** 1.2

### getCtrlX1

public abstract double **getCtrlX1**()

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

**Returns:**the X coordinate of the first control point of the CubicCurve2D.**Since:** 1.2

### getCtrlY1

public abstract double **getCtrlY1**()

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

**Returns:**the Y coordinate of the first control point of the CubicCurve2D.**Since:** 1.2

### getCtrlP1

public abstract [Point2D](http://docs.google.com/java/awt/geom/Point2D.html) **getCtrlP1**()

Returns the first control point.

**Returns:**a Point2D that is the first control point of the CubicCurve2D.**Since:** 1.2

### getCtrlX2

public abstract double **getCtrlX2**()

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

**Returns:**the X coordinate of the second control point of the CubicCurve2D.**Since:** 1.2

### getCtrlY2

public abstract double **getCtrlY2**()

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

**Returns:**the Y coordinate of the second control point of the CubicCurve2D.**Since:** 1.2

### getCtrlP2

public abstract [Point2D](http://docs.google.com/java/awt/geom/Point2D.html) **getCtrlP2**()

Returns the second control point.

**Returns:**a Point2D that is the second control point of the CubicCurve2D.**Since:** 1.2

### getX2

public abstract double **getX2**()

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

**Returns:**the X coordinate of the end point of the CubicCurve2D.**Since:** 1.2

### getY2

public abstract double **getY2**()

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

**Returns:**the Y coordinate of the end point of the CubicCurve2D.**Since:** 1.2

### getP2

public abstract [Point2D](http://docs.google.com/java/awt/geom/Point2D.html) **getP2**()

Returns the end point.

**Returns:**a Point2D that is the end point of the CubicCurve2D.**Since:** 1.2

### setCurve

public abstract void **setCurve**(double x1,  
 double y1,  
 double ctrlx1,  
 double ctrly1,  
 double ctrlx2,  
 double ctrly2,  
 double x2,  
 double y2)

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

**Parameters:**x1 - the X coordinate used to set the start point of this CubicCurve2Dy1 - the Y coordinate used to set the start point of this CubicCurve2Dctrlx1 - the X coordinate used to set the first control point of this CubicCurve2Dctrly1 - the Y coordinate used to set the first control point of this CubicCurve2Dctrlx2 - the X coordinate used to set the second control point of this CubicCurve2Dctrly2 - the Y coordinate used to set the second control point of this CubicCurve2Dx2 - the X coordinate used to set the end point of this CubicCurve2Dy2 - the Y coordinate used to set the end point of this CubicCurve2D**Since:** 1.2

### setCurve

public void **setCurve**(double[] coords,  
 int offset)

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

**Parameters:**coords - a double array containing coordinatesoffset - the index of coords from which to begin setting the end points and control points of this curve to the coordinates contained in coords**Since:** 1.2

### setCurve

public void **setCurve**([Point2D](http://docs.google.com/java/awt/geom/Point2D.html) p1,  
 [Point2D](http://docs.google.com/java/awt/geom/Point2D.html) cp1,  
 [Point2D](http://docs.google.com/java/awt/geom/Point2D.html) cp2,  
 [Point2D](http://docs.google.com/java/awt/geom/Point2D.html) p2)

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

**Parameters:**p1 - the first specified Point2D used to set the start point of this curvecp1 - the second specified Point2D used to set the first control point of this curvecp2 - the third specified Point2D used to set the second control point of this curvep2 - the fourth specified Point2D used to set the end point of this curve**Since:** 1.2

### setCurve

public void **setCurve**([Point2D](http://docs.google.com/java/awt/geom/Point2D.html)[] pts,  
 int offset)

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

**Parameters:**pts - an array of Point2D objectsoffset - the index of pts from which to begin setting the end points and control points of this curve to the points contained in pts**Since:** 1.2

### setCurve

public void **setCurve**([CubicCurve2D](http://docs.google.com/java/awt/geom/CubicCurve2D.html) c)

Sets the location of the end points and control points of this curve to the same as those in the specified CubicCurve2D.

**Parameters:**c - the specified CubicCurve2D**Since:** 1.2

### getFlatnessSq

public static double **getFlatnessSq**(double x1,  
 double y1,  
 double ctrlx1,  
 double ctrly1,  
 double ctrlx2,  
 double ctrly2,  
 double x2,  
 double y2)

Returns the square of the flatness of the cubic curve specified by the indicated control points. The flatness is the maximum distance of a control point from the line connecting the end points.

**Parameters:**x1 - the X coordinate that specifies the start point of a CubicCurve2Dy1 - the Y coordinate that specifies the start point of a CubicCurve2Dctrlx1 - the X coordinate that specifies the first control point of a CubicCurve2Dctrly1 - the Y coordinate that specifies the first control point of a CubicCurve2Dctrlx2 - the X coordinate that specifies the second control point of a CubicCurve2Dctrly2 - the Y coordinate that specifies the second control point of a CubicCurve2Dx2 - the X coordinate that specifies the end point of a CubicCurve2Dy2 - the Y coordinate that specifies the end point of a CubicCurve2D **Returns:**the square of the flatness of the CubicCurve2D represented by the specified coordinates.**Since:** 1.2

### getFlatness

public static double **getFlatness**(double x1,  
 double y1,  
 double ctrlx1,  
 double ctrly1,  
 double ctrlx2,  
 double ctrly2,  
 double x2,  
 double y2)

Returns the flatness of the cubic curve specified by the indicated control points. The flatness is the maximum distance of a control point from the line connecting the end points.

**Parameters:**x1 - the X coordinate that specifies the start point of a CubicCurve2Dy1 - the Y coordinate that specifies the start point of a CubicCurve2Dctrlx1 - the X coordinate that specifies the first control point of a CubicCurve2Dctrly1 - the Y coordinate that specifies the first control point of a CubicCurve2Dctrlx2 - the X coordinate that specifies the second control point of a CubicCurve2Dctrly2 - the Y coordinate that specifies the second control point of a CubicCurve2Dx2 - the X coordinate that specifies the end point of a CubicCurve2Dy2 - the Y coordinate that specifies the end point of a CubicCurve2D **Returns:**the flatness of the CubicCurve2D represented by the specified coordinates.**Since:** 1.2

### getFlatnessSq

public static double **getFlatnessSq**(double[] coords,  
 int offset)

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

**Parameters:**coords - an array containing coordinatesoffset - the index of coords from which to begin getting the end points and control points of the curve **Returns:**the square of the flatness of the CubicCurve2D specified by the coordinates in coords at the specified offset.**Since:** 1.2

### getFlatness

public static double **getFlatness**(double[] coords,  
 int offset)

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

**Parameters:**coords - an array containing coordinatesoffset - the index of coords from which to begin getting the end points and control points of the curve **Returns:**the flatness of the CubicCurve2D specified by the coordinates in coords at the specified offset.**Since:** 1.2

### getFlatnessSq

public double **getFlatnessSq**()

Returns the square of the flatness of this curve. The flatness is the maximum distance of a control point from the line connecting the end points.

**Returns:**the square of the flatness of this curve.**Since:** 1.2

### getFlatness

public double **getFlatness**()

Returns the flatness of this curve. The flatness is the maximum distance of a control point from the line connecting the end points.

**Returns:**the flatness of this curve.**Since:** 1.2

### subdivide

public void **subdivide**([CubicCurve2D](http://docs.google.com/java/awt/geom/CubicCurve2D.html) left,  
 [CubicCurve2D](http://docs.google.com/java/awt/geom/CubicCurve2D.html) right)

Subdivides this cubic curve and stores the resulting two subdivided curves into the left and right curve parameters. Either or both of the left and right objects may be the same as this object or null.

**Parameters:**left - the cubic curve object for storing for the left or first half of the subdivided curveright - the cubic curve object for storing for the right or second half of the subdivided curve**Since:** 1.2

### subdivide

public static void **subdivide**([CubicCurve2D](http://docs.google.com/java/awt/geom/CubicCurve2D.html) src,  
 [CubicCurve2D](http://docs.google.com/java/awt/geom/CubicCurve2D.html) left,  
 [CubicCurve2D](http://docs.google.com/java/awt/geom/CubicCurve2D.html) right)

Subdivides the cubic curve specified by the src parameter and stores the resulting two subdivided curves into the left and right curve parameters. Either or both of the left and right objects may be the same as the src object or null.

**Parameters:**src - the cubic curve to be subdividedleft - the cubic curve object for storing the left or first half of the subdivided curveright - the cubic curve object for storing the right or second half of the subdivided curve**Since:** 1.2

### subdivide

public static void **subdivide**(double[] src,  
 int srcoff,  
 double[] left,  
 int leftoff,  
 double[] right,  
 int rightoff)

Subdivides the cubic curve specified by the coordinates stored in the src array at indices srcoff through (srcoff + 7) and stores the resulting two subdivided curves into the two result arrays at the corresponding indices. Either or both of the left and right arrays may be null or a reference to the same array as the src array. Note that the last point in the first subdivided curve is the same as the first point in the second subdivided curve. Thus, it is possible to pass the same array for left and right and to use offsets, such as rightoff equals (leftoff + 6), in order to avoid allocating extra storage for this common point.

**Parameters:**src - the array holding the coordinates for the source curvesrcoff - the offset into the array of the beginning of the the 6 source coordinatesleft - the array for storing the coordinates for the first half of the subdivided curveleftoff - the offset into the array of the beginning of the the 6 left coordinatesright - the array for storing the coordinates for the second half of the subdivided curverightoff - the offset into the array of the beginning of the the 6 right coordinates**Since:** 1.2

### solveCubic

public static int **solveCubic**(double[] eqn)

Solves the cubic whose coefficients are in the eqn array and places the non-complex roots back into the same array, returning the number of roots. The solved cubic is represented by the equation:

eqn = {c, b, a, d}  
 dx^3 + ax^2 + bx + c = 0

A return value of -1 is used to distinguish a constant equation that might be always 0 or never 0 from an equation that has no zeroes.

**Parameters:**eqn - an array containing coefficients for a cubic **Returns:**the number of roots, or -1 if the equation is a constant.**Since:** 1.2

### solveCubic

public static int **solveCubic**(double[] eqn,  
 double[] res)

Solve the cubic whose coefficients are in the eqn array and place the non-complex roots into the res array, returning the number of roots. The cubic solved is represented by the equation: eqn = {c, b, a, d} dx^3 + ax^2 + bx + c = 0 A return value of -1 is used to distinguish a constant equation, which may be always 0 or never 0, from an equation which has no zeroes.

**Parameters:**eqn - the specified array of coefficients to use to solve the cubic equationres - the array that contains the non-complex roots resulting from the solution of the cubic equation **Returns:**the number of roots, or -1 if the equation is a constant**Since:** 1.3

### contains

public boolean **contains**(double x,  
 double y)

Tests if the specified coordinates are inside the boundary of the Shape.

**Specified by:**[contains](http://docs.google.com/java/awt/Shape.html#contains(double,%20double)) in interface [Shape](http://docs.google.com/java/awt/Shape.html) **Parameters:**x - the specified X coordinate to be testedy - the specified Y coordinate to be tested **Returns:**true if the specified coordinates are inside the Shape boundary; false otherwise.**Since:** 1.2

### contains

public boolean **contains**([Point2D](http://docs.google.com/java/awt/geom/Point2D.html) p)

Tests if a specified [Point2D](http://docs.google.com/java/awt/geom/Point2D.html) is inside the boundary of the Shape.

**Specified by:**[contains](http://docs.google.com/java/awt/Shape.html#contains(java.awt.geom.Point2D)) in interface [Shape](http://docs.google.com/java/awt/Shape.html) **Parameters:**p - the specified Point2D to be tested **Returns:**true if the specified Point2D is inside the boundary of the Shape; false otherwise.**Since:** 1.2

### intersects

public boolean **intersects**(double x,  
 double y,  
 double w,  
 double h)

Tests if the interior of the Shape intersects the interior of a specified rectangular area. The rectangular area is considered to intersect the Shape if any point is contained in both the interior of the Shape and the specified rectangular area.

The Shape.intersects() method allows a Shape implementation to conservatively return true when:

* there is a high probability that the rectangular area and the Shape intersect, but
* the calculations to accurately determine this intersection are prohibitively expensive.

This means that for some Shapes this method might return true even though the rectangular area does not intersect the Shape. The [Area](http://docs.google.com/java/awt/geom/Area.html) class performs more accurate computations of geometric intersection than most Shape objects and therefore can be used if a more precise answer is required.

**Specified by:**[intersects](http://docs.google.com/java/awt/Shape.html#intersects(double,%20double,%20double,%20double)) in interface [Shape](http://docs.google.com/java/awt/Shape.html) **Parameters:**x - the X coordinate of the upper-left corner of the specified rectangular areay - the Y coordinate of the upper-left corner of the specified rectangular areaw - the width of the specified rectangular areah - the height of the specified rectangular area **Returns:**true if the interior of the Shape and the interior of the rectangular area intersect, or are both highly likely to intersect and intersection calculations would be too expensive to perform; false otherwise.**Since:** 1.2 **See Also:**[Area](http://docs.google.com/java/awt/geom/Area.html)

### intersects

public boolean **intersects**([Rectangle2D](http://docs.google.com/java/awt/geom/Rectangle2D.html) r)

Tests if the interior of the Shape intersects the interior of a specified Rectangle2D. The Shape.intersects() method allows a Shape implementation to conservatively return true when:

* there is a high probability that the Rectangle2D and the Shape intersect, but
* the calculations to accurately determine this intersection are prohibitively expensive.

This means that for some Shapes this method might return true even though the Rectangle2D does not intersect the Shape. The [Area](http://docs.google.com/java/awt/geom/Area.html) class performs more accurate computations of geometric intersection than most Shape objects and therefore can be used if a more precise answer is required.

**Specified by:**[intersects](http://docs.google.com/java/awt/Shape.html#intersects(java.awt.geom.Rectangle2D)) in interface [Shape](http://docs.google.com/java/awt/Shape.html) **Parameters:**r - the specified Rectangle2D **Returns:**true if the interior of the Shape and the interior of the specified Rectangle2D intersect, or are both highly likely to intersect and intersection calculations would be too expensive to perform; false otherwise.**Since:** 1.2 **See Also:**[Shape.intersects(double, double, double, double)](http://docs.google.com/java/awt/Shape.html#intersects(double,%20double,%20double,%20double))

### contains

public boolean **contains**(double x,  
 double y,  
 double w,  
 double h)

Tests if the interior of the Shape entirely contains the specified rectangular area. All coordinates that lie inside the rectangular area must lie within the Shape for the entire rectanglar area to be considered contained within the Shape.

The Shape.contains() method allows a Shape implementation to conservatively return false when:

* the intersect method returns true and
* the calculations to determine whether or not the Shape entirely contains the rectangular area are prohibitively expensive.

This means that for some Shapes this method might return false even though the Shape contains the rectangular area. The [Area](http://docs.google.com/java/awt/geom/Area.html) class performs more accurate geometric computations than most Shape objects and therefore can be used if a more precise answer is required.

**Specified by:**[contains](http://docs.google.com/java/awt/Shape.html#contains(double,%20double,%20double,%20double)) in interface [Shape](http://docs.google.com/java/awt/Shape.html) **Parameters:**x - the X coordinate of the upper-left corner of the specified rectangular areay - the Y coordinate of the upper-left corner of the specified rectangular areaw - the width of the specified rectangular areah - the height of the specified rectangular area **Returns:**true if the interior of the Shape entirely contains the specified rectangular area; false otherwise or, if the Shape contains the rectangular area and the intersects method returns true and the containment calculations would be too expensive to perform.**Since:** 1.2 **See Also:**[Area](http://docs.google.com/java/awt/geom/Area.html), [Shape.intersects(double, double, double, double)](http://docs.google.com/java/awt/Shape.html#intersects(double,%20double,%20double,%20double))

### contains

public boolean **contains**([Rectangle2D](http://docs.google.com/java/awt/geom/Rectangle2D.html) r)

Tests if the interior of the Shape entirely contains the specified Rectangle2D. The Shape.contains() method allows a Shape implementation to conservatively return false when:

* the intersect method returns true and
* the calculations to determine whether or not the Shape entirely contains the Rectangle2D are prohibitively expensive.

This means that for some Shapes this method might return false even though the Shape contains the Rectangle2D. The [Area](http://docs.google.com/java/awt/geom/Area.html) class performs more accurate geometric computations than most Shape objects and therefore can be used if a more precise answer is required.

**Specified by:**[contains](http://docs.google.com/java/awt/Shape.html#contains(java.awt.geom.Rectangle2D)) in interface [Shape](http://docs.google.com/java/awt/Shape.html) **Parameters:**r - The specified Rectangle2D **Returns:**true if the interior of the Shape entirely contains the Rectangle2D; false otherwise or, if the Shape contains the Rectangle2D and the intersects method returns true and the containment calculations would be too expensive to perform.**Since:** 1.2 **See Also:**[Shape.contains(double, double, double, double)](http://docs.google.com/java/awt/Shape.html#contains(double,%20double,%20double,%20double))

### getBounds

public [Rectangle](http://docs.google.com/java/awt/Rectangle.html) **getBounds**()

Returns an integer [Rectangle](http://docs.google.com/java/awt/Rectangle.html) that completely encloses the Shape. Note that there is no guarantee that the returned Rectangle is the smallest bounding box that encloses the Shape, only that the Shape lies entirely within the indicated Rectangle. The returned Rectangle might also fail to completely enclose the Shape if the Shape overflows the limited range of the integer data type. The getBounds2D method generally returns a tighter bounding box due to its greater flexibility in representation.

**Specified by:**[getBounds](http://docs.google.com/java/awt/Shape.html#getBounds()) in interface [Shape](http://docs.google.com/java/awt/Shape.html) **Returns:**an integer Rectangle that completely encloses the Shape.**Since:** 1.2 **See Also:**[Shape.getBounds2D()](http://docs.google.com/java/awt/Shape.html#getBounds2D())

### getPathIterator

public [PathIterator](http://docs.google.com/java/awt/geom/PathIterator.html) **getPathIterator**([AffineTransform](http://docs.google.com/java/awt/geom/AffineTransform.html) at)

Returns an iteration object that defines the boundary of the shape. The iterator for this class is not multi-threaded safe, which means that this CubicCurve2D class does not guarantee that modifications to the geometry of this CubicCurve2D object do not affect any iterations of that geometry that are already in process.

**Specified by:**[getPathIterator](http://docs.google.com/java/awt/Shape.html#getPathIterator(java.awt.geom.AffineTransform)) in interface [Shape](http://docs.google.com/java/awt/Shape.html) **Parameters:**at - an optional AffineTransform to be applied to the coordinates as they are returned in the iteration, or null if untransformed coordinates are desired **Returns:**the PathIterator object that returns the geometry of the outline of this CubicCurve2D, one segment at a time.**Since:** 1.2

### getPathIterator

public [PathIterator](http://docs.google.com/java/awt/geom/PathIterator.html) **getPathIterator**([AffineTransform](http://docs.google.com/java/awt/geom/AffineTransform.html) at,  
 double flatness)

Return an iteration object that defines the boundary of the flattened shape. The iterator for this class is not multi-threaded safe, which means that this CubicCurve2D class does not guarantee that modifications to the geometry of this CubicCurve2D object do not affect any iterations of that geometry that are already in process.

**Specified by:**[getPathIterator](http://docs.google.com/java/awt/Shape.html#getPathIterator(java.awt.geom.AffineTransform,%20double)) in interface [Shape](http://docs.google.com/java/awt/Shape.html) **Parameters:**at - an optional AffineTransform to be applied to the coordinates as they are returned in the iteration, or null if untransformed coordinates are desiredflatness - the maximum amount that the control points for a given curve can vary from colinear before a subdivided curve is replaced by a straight line connecting the end points **Returns:**the PathIterator object that returns the geometry of the outline of this CubicCurve2D, one segment at a time.**Since:** 1.2

### clone

public [Object](http://docs.google.com/java/lang/Object.html) **clone**()

Creates a new object of the same class as this object.

**Overrides:**[clone](http://docs.google.com/java/lang/Object.html#clone()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**a clone of this instance. **Throws:** [OutOfMemoryError](http://docs.google.com/java/lang/OutOfMemoryError.html) - if there is not enough memory.**Since:** 1.2 **See Also:**[Cloneable](http://docs.google.com/java/lang/Cloneable.html)

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/CubicCurve2D.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/awt/geom/Area.html)   [**NEXT CLASS**](http://docs.google.com/java/awt/geom/CubicCurve2D.Double.html) | [**FRAMES**](http://docs.google.com/index.html?java/awt/geom/CubicCurve2D.html)    [**NO FRAMES**](http://docs.google.com/CubicCurve2D.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | FIELD | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: FIELD | [CONSTR](#4d34og8) | [METHOD](#17dp8vu) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).