| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Path2D.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/NoninvertibleTransformException.html)   [**NEXT CLASS**](http://docs.google.com/java/awt/geom/Path2D.Double.html) | [**FRAMES**](http://docs.google.com/index.html?java/awt/geom/Path2D.html)    [**NO FRAMES**](http://docs.google.com/Path2D.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | [FIELD](#2et92p0) | CONSTR | [METHOD](#tyjcwt) | DETAIL: [FIELD](#4d34og8) | CONSTR | [METHOD](#3rdcrjn) |

## **java.awt.geom**

Class Path2D

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 **java.awt.geom.Path2D**

**All Implemented Interfaces:** [Shape](http://docs.google.com/java/awt/Shape.html), [Cloneable](http://docs.google.com/java/lang/Cloneable.html) **Direct Known Subclasses:** [Path2D.Double](http://docs.google.com/java/awt/geom/Path2D.Double.html), [Path2D.Float](http://docs.google.com/java/awt/geom/Path2D.Float.html)

public abstract class **Path2D**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 Path2D class provides a simple, yet flexible shape which represents an arbitrary geometric path. It can fully represent any path which can be iterated by the [PathIterator](http://docs.google.com/java/awt/geom/PathIterator.html) interface including all of its segment types and winding rules and it implements all of the basic hit testing methods of the [Shape](http://docs.google.com/java/awt/Shape.html) interface.

Use [Path2D.Float](http://docs.google.com/java/awt/geom/Path2D.Float.html) when dealing with data that can be represented and used with floating point precision. Use [Path2D.Double](http://docs.google.com/java/awt/geom/Path2D.Double.html) for data that requires the accuracy or range of double precision.

Path2D provides exactly those facilities required for basic construction and management of a geometric path and implementation of the above interfaces with little added interpretation. If it is useful to manipulate the interiors of closed geometric shapes beyond simple hit testing then the [Area](http://docs.google.com/java/awt/geom/Area.html) class provides additional capabilities specifically targeted at closed figures. While both classes nominally implement the Shape interface, they differ in purpose and together they provide two useful views of a geometric shape where Path2D deals primarily with a trajectory formed by path segments and Area deals more with interpretation and manipulation of enclosed regions of 2D geometric space.

The [PathIterator](http://docs.google.com/java/awt/geom/PathIterator.html) interface has more detailed descriptions of the types of segments that make up a path and the winding rules that control how to determine which regions are inside or outside the path.

**Since:** 1.6

| **Nested Class Summary** | |
| --- | --- |
| static class | [**Path2D.Double**](http://docs.google.com/java/awt/geom/Path2D.Double.html)            The Double class defines a geometric path with coordinates stored in double precision floating point. |
| static class | [**Path2D.Float**](http://docs.google.com/java/awt/geom/Path2D.Float.html)            The Float class defines a geometric path with coordinates stored in single precision floating point. |

| **Field Summary** | |
| --- | --- |
| static int | [**WIND\_EVEN\_ODD**](http://docs.google.com/java/awt/geom/Path2D.html#WIND_EVEN_ODD)            An even-odd winding rule for determining the interior of a path. |
| static int | [**WIND\_NON\_ZERO**](http://docs.google.com/java/awt/geom/Path2D.html#WIND_NON_ZERO)            A non-zero winding rule for determining the interior of a path. |

| **Method Summary** | |
| --- | --- |
| abstract  void | [**append**](http://docs.google.com/java/awt/geom/Path2D.html#append(java.awt.geom.PathIterator,%20boolean))([PathIterator](http://docs.google.com/java/awt/geom/PathIterator.html) pi, boolean connect)            Appends the geometry of the specified [PathIterator](http://docs.google.com/java/awt/geom/PathIterator.html) object to the path, possibly connecting the new geometry to the existing path segments with a line segment. |
| void | [**append**](http://docs.google.com/java/awt/geom/Path2D.html#append(java.awt.Shape,%20boolean))([Shape](http://docs.google.com/java/awt/Shape.html) s, boolean connect)            Appends the geometry of the specified Shape object to the path, possibly connecting the new geometry to the existing path segments with a line segment. |
| abstract  [Object](http://docs.google.com/java/lang/Object.html) | [**clone**](http://docs.google.com/java/awt/geom/Path2D.html#clone())()            Creates a new object of the same class as this object. |
| void | [**closePath**](http://docs.google.com/java/awt/geom/Path2D.html#closePath())()            Closes the current subpath by drawing a straight line back to the coordinates of the last moveTo. |
| boolean | [**contains**](http://docs.google.com/java/awt/geom/Path2D.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/Path2D.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. |
| static boolean | [**contains**](http://docs.google.com/java/awt/geom/Path2D.html#contains(java.awt.geom.PathIterator,%20double,%20double))([PathIterator](http://docs.google.com/java/awt/geom/PathIterator.html) pi, double x, double y)            Tests if the specified coordinates are inside the closed boundary of the specified [PathIterator](http://docs.google.com/java/awt/geom/PathIterator.html). |
| static boolean | [**contains**](http://docs.google.com/java/awt/geom/Path2D.html#contains(java.awt.geom.PathIterator,%20double,%20double,%20double,%20double))([PathIterator](http://docs.google.com/java/awt/geom/PathIterator.html) pi, double x, double y, double w, double h)            Tests if the specified rectangular area is entirely inside the closed boundary of the specified [PathIterator](http://docs.google.com/java/awt/geom/PathIterator.html). |
| static boolean | [**contains**](http://docs.google.com/java/awt/geom/Path2D.html#contains(java.awt.geom.PathIterator,%20java.awt.geom.Point2D))([PathIterator](http://docs.google.com/java/awt/geom/PathIterator.html) pi, [Point2D](http://docs.google.com/java/awt/geom/Point2D.html) p)            Tests if the specified [Point2D](http://docs.google.com/java/awt/geom/Point2D.html) is inside the closed boundary of the specified [PathIterator](http://docs.google.com/java/awt/geom/PathIterator.html). |
| static boolean | [**contains**](http://docs.google.com/java/awt/geom/Path2D.html#contains(java.awt.geom.PathIterator,%20java.awt.geom.Rectangle2D))([PathIterator](http://docs.google.com/java/awt/geom/PathIterator.html) pi, [Rectangle2D](http://docs.google.com/java/awt/geom/Rectangle2D.html) r)            Tests if the specified [Rectangle2D](http://docs.google.com/java/awt/geom/Rectangle2D.html) is entirely inside the closed boundary of the specified [PathIterator](http://docs.google.com/java/awt/geom/PathIterator.html). |
| boolean | [**contains**](http://docs.google.com/java/awt/geom/Path2D.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/Path2D.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. |
| [Shape](http://docs.google.com/java/awt/Shape.html) | [**createTransformedShape**](http://docs.google.com/java/awt/geom/Path2D.html#createTransformedShape(java.awt.geom.AffineTransform))([AffineTransform](http://docs.google.com/java/awt/geom/AffineTransform.html) at)            Returns a new Shape representing a transformed version of this Path2D. |
| abstract  void | [**curveTo**](http://docs.google.com/java/awt/geom/Path2D.html#curveTo(double,%20double,%20double,%20double,%20double,%20double))(double x1, double y1, double x2, double y2, double x3, double y3)            Adds a curved segment, defined by three new points, to the path by drawing a Bézier curve that intersects both the current coordinates and the specified coordinates (x3,y3), using the specified points (x1,y1) and (x2,y2) as Bézier control points. |
| [Rectangle](http://docs.google.com/java/awt/Rectangle.html) | [**getBounds**](http://docs.google.com/java/awt/geom/Path2D.html#getBounds())()            Returns an integer [Rectangle](http://docs.google.com/java/awt/Rectangle.html) that completely encloses the Shape. |
| [Point2D](http://docs.google.com/java/awt/geom/Point2D.html) | [**getCurrentPoint**](http://docs.google.com/java/awt/geom/Path2D.html#getCurrentPoint())()            Returns the coordinates most recently added to the end of the path as a [Point2D](http://docs.google.com/java/awt/geom/Point2D.html) object. |
| [PathIterator](http://docs.google.com/java/awt/geom/PathIterator.html) | [**getPathIterator**](http://docs.google.com/java/awt/geom/Path2D.html#getPathIterator(java.awt.geom.AffineTransform,%20double))([AffineTransform](http://docs.google.com/java/awt/geom/AffineTransform.html) at, double flatness)            Returns an iterator object that iterates along the Shape boundary and provides access to a flattened view of the Shape outline geometry. |
| int | [**getWindingRule**](http://docs.google.com/java/awt/geom/Path2D.html#getWindingRule())()            Returns the fill style winding rule. |
| boolean | [**intersects**](http://docs.google.com/java/awt/geom/Path2D.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. |
| static boolean | [**intersects**](http://docs.google.com/java/awt/geom/Path2D.html#intersects(java.awt.geom.PathIterator,%20double,%20double,%20double,%20double))([PathIterator](http://docs.google.com/java/awt/geom/PathIterator.html) pi, double x, double y, double w, double h)            Tests if the interior of the specified [PathIterator](http://docs.google.com/java/awt/geom/PathIterator.html) intersects the interior of a specified set of rectangular coordinates. |
| static boolean | [**intersects**](http://docs.google.com/java/awt/geom/Path2D.html#intersects(java.awt.geom.PathIterator,%20java.awt.geom.Rectangle2D))([PathIterator](http://docs.google.com/java/awt/geom/PathIterator.html) pi, [Rectangle2D](http://docs.google.com/java/awt/geom/Rectangle2D.html) r)            Tests if the interior of the specified [PathIterator](http://docs.google.com/java/awt/geom/PathIterator.html) intersects the interior of a specified [Rectangle2D](http://docs.google.com/java/awt/geom/Rectangle2D.html). |
| boolean | [**intersects**](http://docs.google.com/java/awt/geom/Path2D.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. |
| abstract  void | [**lineTo**](http://docs.google.com/java/awt/geom/Path2D.html#lineTo(double,%20double))(double x, double y)            Adds a point to the path by drawing a straight line from the current coordinates to the new specified coordinates specified in double precision. |
| abstract  void | [**moveTo**](http://docs.google.com/java/awt/geom/Path2D.html#moveTo(double,%20double))(double x, double y)            Adds a point to the path by moving to the specified coordinates specified in double precision. |
| abstract  void | [**quadTo**](http://docs.google.com/java/awt/geom/Path2D.html#quadTo(double,%20double,%20double,%20double))(double x1, double y1, double x2, double y2)            Adds a curved segment, defined by two new points, to the path by drawing a Quadratic curve that intersects both the current coordinates and the specified coordinates (x2,y2), using the specified point (x1,y1) as a quadratic parametric control point. |
| void | [**reset**](http://docs.google.com/java/awt/geom/Path2D.html#reset())()            Resets the path to empty. |
| void | [**setWindingRule**](http://docs.google.com/java/awt/geom/Path2D.html#setWindingRule(int))(int rule)            Sets the winding rule for this path to the specified value. |
| abstract  void | [**transform**](http://docs.google.com/java/awt/geom/Path2D.html#transform(java.awt.geom.AffineTransform))([AffineTransform](http://docs.google.com/java/awt/geom/AffineTransform.html) at)            Transforms the geometry of this path using the specified [AffineTransform](http://docs.google.com/java/awt/geom/AffineTransform.html). |

| **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()), [getPathIterator](http://docs.google.com/java/awt/Shape.html#getPathIterator(java.awt.geom.AffineTransform)) |

| **Field Detail** |
| --- |

### WIND\_EVEN\_ODD

public static final int **WIND\_EVEN\_ODD**

An even-odd winding rule for determining the interior of a path.

**Since:** 1.6 **See Also:**[PathIterator.WIND\_EVEN\_ODD](http://docs.google.com/java/awt/geom/PathIterator.html#WIND_EVEN_ODD), [Constant Field Values](http://docs.google.com/constant-values.html#java.awt.geom.Path2D.WIND_EVEN_ODD)

### WIND\_NON\_ZERO

public static final int **WIND\_NON\_ZERO**

A non-zero winding rule for determining the interior of a path.

**Since:** 1.6 **See Also:**[PathIterator.WIND\_NON\_ZERO](http://docs.google.com/java/awt/geom/PathIterator.html#WIND_NON_ZERO), [Constant Field Values](http://docs.google.com/constant-values.html#java.awt.geom.Path2D.WIND_NON_ZERO)

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

### moveTo

public abstract void **moveTo**(double x,  
 double y)

Adds a point to the path by moving to the specified coordinates specified in double precision.

**Parameters:**x - the specified X coordinatey - the specified Y coordinate**Since:** 1.6

### lineTo

public abstract void **lineTo**(double x,  
 double y)

Adds a point to the path by drawing a straight line from the current coordinates to the new specified coordinates specified in double precision.

**Parameters:**x - the specified X coordinatey - the specified Y coordinate**Since:** 1.6

### quadTo

public abstract void **quadTo**(double x1,  
 double y1,  
 double x2,  
 double y2)

Adds a curved segment, defined by two new points, to the path by drawing a Quadratic curve that intersects both the current coordinates and the specified coordinates (x2,y2), using the specified point (x1,y1) as a quadratic parametric control point. All coordinates are specified in double precision.

**Parameters:**x1 - the X coordinate of the quadratic control pointy1 - the Y coordinate of the quadratic control pointx2 - the X coordinate of the final end pointy2 - the Y coordinate of the final end point**Since:** 1.6

### curveTo

public abstract void **curveTo**(double x1,  
 double y1,  
 double x2,  
 double y2,  
 double x3,  
 double y3)

Adds a curved segment, defined by three new points, to the path by drawing a Bézier curve that intersects both the current coordinates and the specified coordinates (x3,y3), using the specified points (x1,y1) and (x2,y2) as Bézier control points. All coordinates are specified in double precision.

**Parameters:**x1 - the X coordinate of the first Bézier control pointy1 - the Y coordinate of the first Bézier control pointx2 - the X coordinate of the second Bézier control pointy2 - the Y coordinate of the second Bézier control pointx3 - the X coordinate of the final end pointy3 - the Y coordinate of the final end point**Since:** 1.6

### closePath

public final void **closePath**()

Closes the current subpath by drawing a straight line back to the coordinates of the last moveTo. If the path is already closed then this method has no effect.

**Since:** 1.6

### append

public final void **append**([Shape](http://docs.google.com/java/awt/Shape.html) s,  
 boolean connect)

Appends the geometry of the specified Shape object to the path, possibly connecting the new geometry to the existing path segments with a line segment. If the connect parameter is true and the path is not empty then any initial moveTo in the geometry of the appended Shape is turned into a lineTo segment. If the destination coordinates of such a connecting lineTo segment match the ending coordinates of a currently open subpath then the segment is omitted as superfluous. The winding rule of the specified Shape is ignored and the appended geometry is governed by the winding rule specified for this path.

**Parameters:**s - the Shape whose geometry is appended to this pathconnect - a boolean to control whether or not to turn an initial moveTo segment into a lineTo segment to connect the new geometry to the existing path**Since:** 1.6

### append

public abstract void **append**([PathIterator](http://docs.google.com/java/awt/geom/PathIterator.html) pi,  
 boolean connect)

Appends the geometry of the specified [PathIterator](http://docs.google.com/java/awt/geom/PathIterator.html) object to the path, possibly connecting the new geometry to the existing path segments with a line segment. If the connect parameter is true and the path is not empty then any initial moveTo in the geometry of the appended Shape is turned into a lineTo segment. If the destination coordinates of such a connecting lineTo segment match the ending coordinates of a currently open subpath then the segment is omitted as superfluous. The winding rule of the specified Shape is ignored and the appended geometry is governed by the winding rule specified for this path.

**Parameters:**pi - the PathIterator whose geometry is appended to this pathconnect - a boolean to control whether or not to turn an initial moveTo segment into a lineTo segment to connect the new geometry to the existing path**Since:** 1.6

### getWindingRule

public final int **getWindingRule**()

Returns the fill style winding rule.

**Returns:**an integer representing the current winding rule.**Since:** 1.6 **See Also:**[WIND\_EVEN\_ODD](http://docs.google.com/java/awt/geom/Path2D.html#WIND_EVEN_ODD), [WIND\_NON\_ZERO](http://docs.google.com/java/awt/geom/Path2D.html#WIND_NON_ZERO), [setWindingRule(int)](http://docs.google.com/java/awt/geom/Path2D.html#setWindingRule(int))

### setWindingRule

public final void **setWindingRule**(int rule)

Sets the winding rule for this path to the specified value.

**Parameters:**rule - an integer representing the specified winding rule **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if rule is not either [WIND\_EVEN\_ODD](http://docs.google.com/java/awt/geom/Path2D.html#WIND_EVEN_ODD) or [WIND\_NON\_ZERO](http://docs.google.com/java/awt/geom/Path2D.html#WIND_NON_ZERO)**Since:** 1.6 **See Also:**[getWindingRule()](http://docs.google.com/java/awt/geom/Path2D.html#getWindingRule())

### getCurrentPoint

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

Returns the coordinates most recently added to the end of the path as a [Point2D](http://docs.google.com/java/awt/geom/Point2D.html) object.

**Returns:**a Point2D object containing the ending coordinates of the path or null if there are no points in the path.**Since:** 1.6

### reset

public final void **reset**()

Resets the path to empty. The append position is set back to the beginning of the path and all coordinates and point types are forgotten.

**Since:** 1.6

### transform

public abstract void **transform**([AffineTransform](http://docs.google.com/java/awt/geom/AffineTransform.html) at)

Transforms the geometry of this path using the specified [AffineTransform](http://docs.google.com/java/awt/geom/AffineTransform.html). The geometry is transformed in place, which permanently changes the boundary defined by this object.

**Parameters:**at - the AffineTransform used to transform the area**Since:** 1.6

### createTransformedShape

public final [Shape](http://docs.google.com/java/awt/Shape.html) **createTransformedShape**([AffineTransform](http://docs.google.com/java/awt/geom/AffineTransform.html) at)

Returns a new Shape representing a transformed version of this Path2D. Note that the exact type and coordinate precision of the return value is not specified for this method. The method will return a Shape that contains no less precision for the transformed geometry than this Path2D currently maintains, but it may contain no more precision either. If the tradeoff of precision vs. storage size in the result is important then the convenience constructors in the [Path2D.Float](http://docs.google.com/java/awt/geom/Path2D.Float.html#Path2D.Float(java.awt.Shape,%20java.awt.geom.AffineTransform)) and [Path2D.Double](http://docs.google.com/java/awt/geom/Path2D.Double.html#Path2D.Double(java.awt.Shape,%20java.awt.geom.AffineTransform)) subclasses should be used to make the choice explicit.

**Parameters:**at - the AffineTransform used to transform a new Shape. **Returns:**a new Shape, transformed with the specified AffineTransform.**Since:** 1.6

### getBounds

public final [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:** 1.6 **See Also:**[Shape.getBounds2D()](http://docs.google.com/java/awt/Shape.html#getBounds2D())

### contains

public static boolean **contains**([PathIterator](http://docs.google.com/java/awt/geom/PathIterator.html) pi,  
 double x,  
 double y)

Tests if the specified coordinates are inside the closed boundary of the specified [PathIterator](http://docs.google.com/java/awt/geom/PathIterator.html).

This method provides a basic facility for implementors of the [Shape](http://docs.google.com/java/awt/Shape.html) interface to implement support for the [Shape.contains(double, double)](http://docs.google.com/java/awt/Shape.html#contains(double,%20double)) method.

**Parameters:**pi - the specified PathIteratorx - the specified X coordinatey - the specified Y coordinate **Returns:**true if the specified coordinates are inside the specified PathIterator; false otherwise**Since:** 1.6

### contains

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

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

This method provides a basic facility for implementors of the [Shape](http://docs.google.com/java/awt/Shape.html) interface to implement support for the [Shape.contains(Point2D)](http://docs.google.com/java/awt/Shape.html#contains(java.awt.geom.Point2D)) method.

**Parameters:**pi - the specified PathIteratorp - the specified Point2D **Returns:**true if the specified coordinates are inside the specified PathIterator; false otherwise**Since:** 1.6

### contains

public final 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:** 1.6

### contains

public final 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:** 1.6

### contains

public static boolean **contains**([PathIterator](http://docs.google.com/java/awt/geom/PathIterator.html) pi,  
 double x,  
 double y,  
 double w,  
 double h)

Tests if the specified rectangular area is entirely inside the closed boundary of the specified [PathIterator](http://docs.google.com/java/awt/geom/PathIterator.html).

This method provides a basic facility for implementors of the [Shape](http://docs.google.com/java/awt/Shape.html) interface to implement support for the [Shape.contains(double, double, double, double)](http://docs.google.com/java/awt/Shape.html#contains(double,%20double,%20double,%20double)) method.

This method object may conservatively return false in cases where the specified rectangular area intersects a segment of the path, but that segment does not represent a boundary between the interior and exterior of the path. Such segments could lie entirely within the interior of the path if they are part of a path with a [WIND\_NON\_ZERO](http://docs.google.com/java/awt/geom/Path2D.html#WIND_NON_ZERO) winding rule or if the segments are retraced in the reverse direction such that the two sets of segments cancel each other out without any exterior area falling between them. To determine whether segments represent true boundaries of the interior of the path would require extensive calculations involving all of the segments of the path and the winding rule and are thus beyond the scope of this implementation.

**Parameters:**pi - the specified PathIteratorx - the specified X coordinatey - the specified Y coordinatew - the width of the specified rectangular areah - the height of the specified rectangular area **Returns:**true if the specified PathIterator contains the specified rectangluar area; false otherwise.**Since:** 1.6

### contains

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

Tests if the specified [Rectangle2D](http://docs.google.com/java/awt/geom/Rectangle2D.html) is entirely inside the closed boundary of the specified [PathIterator](http://docs.google.com/java/awt/geom/PathIterator.html).

This method provides a basic facility for implementors of the [Shape](http://docs.google.com/java/awt/Shape.html) interface to implement support for the [Shape.contains(Rectangle2D)](http://docs.google.com/java/awt/Shape.html#contains(java.awt.geom.Rectangle2D)) method.

This method object may conservatively return false in cases where the specified rectangular area intersects a segment of the path, but that segment does not represent a boundary between the interior and exterior of the path. Such segments could lie entirely within the interior of the path if they are part of a path with a [WIND\_NON\_ZERO](http://docs.google.com/java/awt/geom/Path2D.html#WIND_NON_ZERO) winding rule or if the segments are retraced in the reverse direction such that the two sets of segments cancel each other out without any exterior area falling between them. To determine whether segments represent true boundaries of the interior of the path would require extensive calculations involving all of the segments of the path and the winding rule and are thus beyond the scope of this implementation.

**Parameters:**pi - the specified PathIteratorr - a specified Rectangle2D **Returns:**true if the specified PathIterator contains the specified Rectangle2D; false otherwise.**Since:** 1.6

### contains

public final 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.

This method object may conservatively return false in cases where the specified rectangular area intersects a segment of the path, but that segment does not represent a boundary between the interior and exterior of the path. Such segments could lie entirely within the interior of the path if they are part of a path with a [WIND\_NON\_ZERO](http://docs.google.com/java/awt/geom/Path2D.html#WIND_NON_ZERO) winding rule or if the segments are retraced in the reverse direction such that the two sets of segments cancel each other out without any exterior area falling between them. To determine whether segments represent true boundaries of the interior of the path would require extensive calculations involving all of the segments of the path and the winding rule and are thus beyond the scope of this implementation.

**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:** 1.6 **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 final 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.

This method object may conservatively return false in cases where the specified rectangular area intersects a segment of the path, but that segment does not represent a boundary between the interior and exterior of the path. Such segments could lie entirely within the interior of the path if they are part of a path with a [WIND\_NON\_ZERO](http://docs.google.com/java/awt/geom/Path2D.html#WIND_NON_ZERO) winding rule or if the segments are retraced in the reverse direction such that the two sets of segments cancel each other out without any exterior area falling between them. To determine whether segments represent true boundaries of the interior of the path would require extensive calculations involving all of the segments of the path and the winding rule and are thus beyond the scope of this implementation.

**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:** 1.6 **See Also:**[Shape.contains(double, double, double, double)](http://docs.google.com/java/awt/Shape.html#contains(double,%20double,%20double,%20double))

### intersects

public static boolean **intersects**([PathIterator](http://docs.google.com/java/awt/geom/PathIterator.html) pi,  
 double x,  
 double y,  
 double w,  
 double h)

Tests if the interior of the specified [PathIterator](http://docs.google.com/java/awt/geom/PathIterator.html) intersects the interior of a specified set of rectangular coordinates.

This method provides a basic facility for implementors of the [Shape](http://docs.google.com/java/awt/Shape.html) interface to implement support for the [Shape.intersects(double, double, double, double)](http://docs.google.com/java/awt/Shape.html#intersects(double,%20double,%20double,%20double)) method.

This method object may conservatively return true in cases where the specified rectangular area intersects a segment of the path, but that segment does not represent a boundary between the interior and exterior of the path. Such a case may occur if some set of segments of the path are retraced in the reverse direction such that the two sets of segments cancel each other out without any interior area between them. To determine whether segments represent true boundaries of the interior of the path would require extensive calculations involving all of the segments of the path and the winding rule and are thus beyond the scope of this implementation.

**Parameters:**pi - the specified PathIteratorx - the specified X coordinatey - the specified Y coordinatew - the width of the specified rectangular coordinatesh - the height of the specified rectangular coordinates **Returns:**true if the specified PathIterator and the interior of the specified set of rectangular coordinates intersect each other; false otherwise.**Since:** 1.6

### intersects

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

Tests if the interior of the specified [PathIterator](http://docs.google.com/java/awt/geom/PathIterator.html) intersects the interior of a specified [Rectangle2D](http://docs.google.com/java/awt/geom/Rectangle2D.html).

This method provides a basic facility for implementors of the [Shape](http://docs.google.com/java/awt/Shape.html) interface to implement support for the [Shape.intersects(Rectangle2D)](http://docs.google.com/java/awt/Shape.html#intersects(java.awt.geom.Rectangle2D)) method.

This method object may conservatively return true in cases where the specified rectangular area intersects a segment of the path, but that segment does not represent a boundary between the interior and exterior of the path. Such a case may occur if some set of segments of the path are retraced in the reverse direction such that the two sets of segments cancel each other out without any interior area between them. To determine whether segments represent true boundaries of the interior of the path would require extensive calculations involving all of the segments of the path and the winding rule and are thus beyond the scope of this implementation.

**Parameters:**pi - the specified PathIteratorr - the specified Rectangle2D **Returns:**true if the specified PathIterator and the interior of the specified Rectangle2D intersect each other; false otherwise.**Since:** 1.6

### intersects

public final 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.

This method object may conservatively return true in cases where the specified rectangular area intersects a segment of the path, but that segment does not represent a boundary between the interior and exterior of the path. Such a case may occur if some set of segments of the path are retraced in the reverse direction such that the two sets of segments cancel each other out without any interior area between them. To determine whether segments represent true boundaries of the interior of the path would require extensive calculations involving all of the segments of the path and the winding rule and are thus beyond the scope of this implementation.

**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:** 1.6 **See Also:**[Area](http://docs.google.com/java/awt/geom/Area.html)

### intersects

public final 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.

This method object may conservatively return true in cases where the specified rectangular area intersects a segment of the path, but that segment does not represent a boundary between the interior and exterior of the path. Such a case may occur if some set of segments of the path are retraced in the reverse direction such that the two sets of segments cancel each other out without any interior area between them. To determine whether segments represent true boundaries of the interior of the path would require extensive calculations involving all of the segments of the path and the winding rule and are thus beyond the scope of this implementation.

**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:** 1.6 **See Also:**[Shape.intersects(double, double, double, double)](http://docs.google.com/java/awt/Shape.html#intersects(double,%20double,%20double,%20double))

### 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)

Returns an iterator object that iterates along the Shape boundary and provides access to a flattened view of the Shape outline geometry.

Only SEG\_MOVETO, SEG\_LINETO, and SEG\_CLOSE point types are returned by the iterator.

If an optional AffineTransform is specified, the coordinates returned in the iteration are transformed accordingly.

The amount of subdivision of the curved segments is controlled by the flatness parameter, which specifies the maximum distance that any point on the unflattened transformed curve can deviate from the returned flattened path segments. Note that a limit on the accuracy of the flattened path might be silently imposed, causing very small flattening parameters to be treated as larger values. This limit, if there is one, is defined by the particular implementation that is used.

Each call to this method returns a fresh PathIterator object that traverses the Shape object geometry independently from any other PathIterator objects in use at the same time.

It is recommended, but not guaranteed, that objects implementing the Shape interface isolate iterations that are in process from any changes that might occur to the original object's geometry during such iterations.

The iterator for this class is not multi-threaded safe, which means that this Path2D class does not guarantee that modifications to the geometry of this Path2D 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 distance that the line segments used to approximate the curved segments are allowed to deviate from any point on the original curve **Returns:**a new PathIterator that independently traverses a flattened view of the geometry of the Shape.**Since:** 1.6

### clone

public abstract [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:** 1.6 **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/Path2D.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/NoninvertibleTransformException.html)   [**NEXT CLASS**](http://docs.google.com/java/awt/geom/Path2D.Double.html) | [**FRAMES**](http://docs.google.com/index.html?java/awt/geom/Path2D.html)    [**NO FRAMES**](http://docs.google.com/Path2D.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | [FIELD](#2et92p0) | CONSTR | [METHOD](#tyjcwt) | DETAIL: [FIELD](#4d34og8) | CONSTR | [METHOD](#3rdcrjn) |

[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).