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

## **java.awt**

Interface Shape

**All Known Implementing Classes:** [Arc2D](http://docs.google.com/java/awt/geom/Arc2D.html), [Arc2D.Double](http://docs.google.com/java/awt/geom/Arc2D.Double.html), [Arc2D.Float](http://docs.google.com/java/awt/geom/Arc2D.Float.html), [Area](http://docs.google.com/java/awt/geom/Area.html), [BasicTextUI.BasicCaret](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.BasicCaret.html), [CubicCurve2D](http://docs.google.com/java/awt/geom/CubicCurve2D.html), [CubicCurve2D.Double](http://docs.google.com/java/awt/geom/CubicCurve2D.Double.html), [CubicCurve2D.Float](http://docs.google.com/java/awt/geom/CubicCurve2D.Float.html), [DefaultCaret](http://docs.google.com/javax/swing/text/DefaultCaret.html), [Ellipse2D](http://docs.google.com/java/awt/geom/Ellipse2D.html), [Ellipse2D.Double](http://docs.google.com/java/awt/geom/Ellipse2D.Double.html), [Ellipse2D.Float](http://docs.google.com/java/awt/geom/Ellipse2D.Float.html), [GeneralPath](http://docs.google.com/java/awt/geom/GeneralPath.html), [Line2D](http://docs.google.com/java/awt/geom/Line2D.html), [Line2D.Double](http://docs.google.com/java/awt/geom/Line2D.Double.html), [Line2D.Float](http://docs.google.com/java/awt/geom/Line2D.Float.html), [Path2D](http://docs.google.com/java/awt/geom/Path2D.html), [Path2D.Double](http://docs.google.com/java/awt/geom/Path2D.Double.html), [Path2D.Float](http://docs.google.com/java/awt/geom/Path2D.Float.html), [Polygon](http://docs.google.com/java/awt/Polygon.html), [QuadCurve2D](http://docs.google.com/java/awt/geom/QuadCurve2D.html), [QuadCurve2D.Double](http://docs.google.com/java/awt/geom/QuadCurve2D.Double.html), [QuadCurve2D.Float](http://docs.google.com/java/awt/geom/QuadCurve2D.Float.html), [Rectangle](http://docs.google.com/java/awt/Rectangle.html), [Rectangle2D](http://docs.google.com/java/awt/geom/Rectangle2D.html), [Rectangle2D.Double](http://docs.google.com/java/awt/geom/Rectangle2D.Double.html), [Rectangle2D.Float](http://docs.google.com/java/awt/geom/Rectangle2D.Float.html), [RectangularShape](http://docs.google.com/java/awt/geom/RectangularShape.html), [RoundRectangle2D](http://docs.google.com/java/awt/geom/RoundRectangle2D.html), [RoundRectangle2D.Double](http://docs.google.com/java/awt/geom/RoundRectangle2D.Double.html), [RoundRectangle2D.Float](http://docs.google.com/java/awt/geom/RoundRectangle2D.Float.html)

public interface **Shape**

The Shape interface provides definitions for objects that represent some form of geometric shape. The Shape is described by a [PathIterator](http://docs.google.com/java/awt/geom/PathIterator.html) object, which can express the outline of the Shape as well as a rule for determining how the outline divides the 2D plane into interior and exterior points. Each Shape object provides callbacks to get the bounding box of the geometry, determine whether points or rectangles lie partly or entirely within the interior of the Shape, and retrieve a PathIterator object that describes the trajectory path of the Shape outline.

**Definition of insideness:** A point is considered to lie inside a Shape if and only if:

* it lies completely inside theShape boundary *or*
* it lies exactly on the Shape boundary *and* the space immediately adjacent to the point in the increasing X direction is entirely inside the boundary *or*
* it lies exactly on a horizontal boundary segment **and** the space immediately adjacent to the point in the increasing Y direction is inside the boundary.

The contains and intersects methods consider the interior of a Shape to be the area it encloses as if it were filled. This means that these methods consider unclosed shapes to be implicitly closed for the purpose of determining if a shape contains or intersects a rectangle or if a shape contains a point.

**Since:** 1.2 **See Also:**[PathIterator](http://docs.google.com/java/awt/geom/PathIterator.html), [AffineTransform](http://docs.google.com/java/awt/geom/AffineTransform.html), [FlatteningPathIterator](http://docs.google.com/java/awt/geom/FlatteningPathIterator.html), [GeneralPath](http://docs.google.com/java/awt/geom/GeneralPath.html)

| **Method Summary** | |
| --- | --- |
| boolean | [**contains**](http://docs.google.com/java/awt/Shape.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/Shape.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/Shape.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/Shape.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/Shape.html#getBounds())()            Returns an integer [Rectangle](http://docs.google.com/java/awt/Rectangle.html) that completely encloses the Shape. |
| [Rectangle2D](http://docs.google.com/java/awt/geom/Rectangle2D.html) | [**getBounds2D**](http://docs.google.com/java/awt/Shape.html#getBounds2D())()            Returns a high precision and more accurate bounding box of the Shape than the getBounds method. |
| [PathIterator](http://docs.google.com/java/awt/geom/PathIterator.html) | [**getPathIterator**](http://docs.google.com/java/awt/Shape.html#getPathIterator(java.awt.geom.AffineTransform))([AffineTransform](http://docs.google.com/java/awt/geom/AffineTransform.html) at)            Returns an iterator object that iterates along the Shape boundary and provides access to the geometry of the Shape outline. |
| [PathIterator](http://docs.google.com/java/awt/geom/PathIterator.html) | [**getPathIterator**](http://docs.google.com/java/awt/Shape.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. |
| boolean | [**intersects**](http://docs.google.com/java/awt/Shape.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/Shape.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. |

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

### getBounds

[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.

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

### getBounds2D

[Rectangle2D](http://docs.google.com/java/awt/geom/Rectangle2D.html) **getBounds2D**()

Returns a high precision and more accurate bounding box of the Shape than the getBounds method. Note that there is no guarantee that the returned [Rectangle2D](http://docs.google.com/java/awt/geom/Rectangle2D.html) is the smallest bounding box that encloses the Shape, only that the Shape lies entirely within the indicated Rectangle2D. The bounding box returned by this method is usually tighter than that returned by the getBounds method and never fails due to overflow problems since the return value can be an instance of the Rectangle2D that uses double precision values to store the dimensions.

**Returns:**an instance of Rectangle2D that is a high-precision bounding box of the Shape.**Since:** 1.2 **See Also:**[getBounds()](http://docs.google.com/java/awt/Shape.html#getBounds())

### contains

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

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

**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

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.

**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

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.

**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

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.

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

### contains

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.

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

### contains

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.

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

### getPathIterator

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

Returns an iterator object that iterates along the Shape boundary and provides access to the geometry of the Shape outline. If an optional [AffineTransform](http://docs.google.com/java/awt/geom/AffineTransform.html) is specified, the coordinates returned in the iteration are transformed accordingly.

Each call to this method returns a fresh PathIterator object that traverses the geometry of the Shape object 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.

**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:**a new PathIterator object, which independently traverses the geometry of the Shape.**Since:** 1.2

### getPathIterator

[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.

**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:** 1.2

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

[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).