Table of Contents

Geometry Recipes

The Geometry classes are in the **geoscript.geom** package.

Creating Geometries

Point

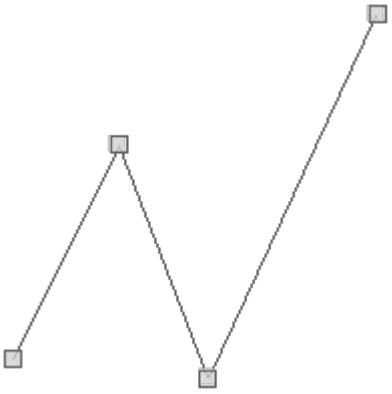
Create a Point with an XY

```
Point point = new Point(-123,46)
```

LineString

Create a LineString from Coordinates

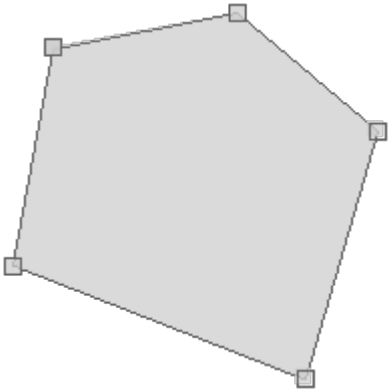
```
LineString lineString = new LineString(
        [3.1982421875, 43.1640625],
        [6.7138671875, 49.755859375],
        [9.7021484375, 42.5927734375],
        [15.3271484375, 53.798828125]
)
```



Polygon

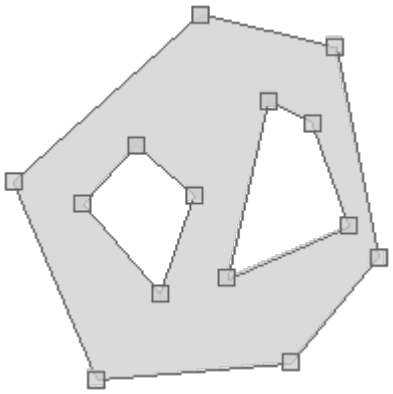
Create a Polygon from a List of Coordinates

```
Polygon polygon = new Polygon([[
       [-101.35986328125, 47.754097979680026],
       [-101.5576171875, 46.93526088057719],
       [-100.12939453125, 46.51351558059737],
       [-99.77783203125, 47.44294999517949],
       [-100.45898437499999, 47.88688085106901],
       [-101.35986328125, 47.754097979680026]
]])
```



Create a Polygon with holes

```
Polygon polygonWithHoles = new Polygon(
    // Exterior Ring
    new LinearRing(
        [-122.39138603210449, 47.58659965790016],
        [-122.41250038146973, 47.57681522195182],
        [-122.40305900573729, 47.56523364515569],
        [-122.38117218017578, 47.56621817878201],
        [-122.3712158203125, 47.57235661809739],
        [-122.37602233886717, 47.584747123985615],
        [-122.39138603210449, 47.58659965790016]
    ),
    // Holes
    [
        new LinearRing(
            [-122.39859580993652, 47.578957532923376],
            [-122.40468978881836, 47.57548347095205],
            [-122.39593505859376, 47.570271945800094],
            [-122.3920726776123, 47.57606249728773],
            [-122.39859580993652, 47.578957532923376]
        ),
        new LinearRing(
            [-122.3836612701416, 47.58156292813543],
            [-122.38829612731934, 47.57114056934196],
            [-122.37456321716309, 47.57420959047542],
            [-122.37868309020995, 47.58023129789275],
            [-122.3836612701416, 47.58156292813543]
        )
    ]
)
```



MultiPoint

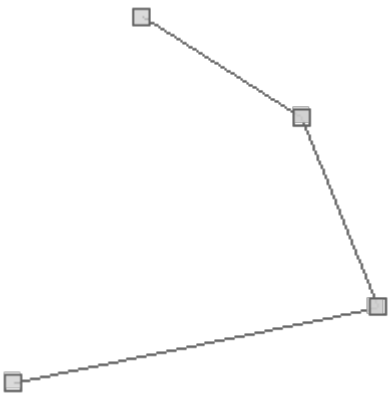
Create a MultiPoint with a List of Points

```
MultiPoint multiPoint = new MultiPoint([
        new Point(-122.3876953125, 47.5820839916191),
        new Point(-122.464599609375, 47.25686404408872),
        new Point(-122.48382568359374, 47.431803338643334)
])
```

MultiLineString

Create a MultiLineString with a List of LineStrings

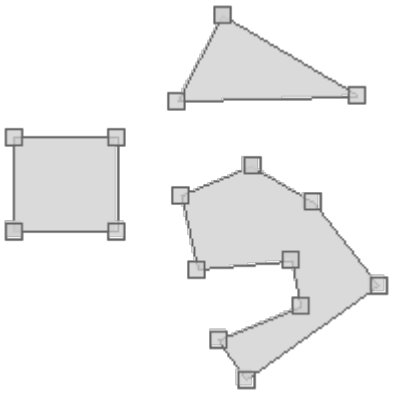
```
MultiLineString multiLineString = new MultiLineString([
        new LineString (
                [-122.3822021484375, 47.57837853860192],
                [-122.32452392578125, 47.48380086737799]
        ),
        new LineString (
                [-122.32452392578125, 47.48380086737799],
                [-122.29705810546874, 47.303447043862626]
        ),
        new LineString (
                [-122.29705810546874, 47.303447043862626],
                [-122.42889404296875, 47.23262467463881]
        )
])
```



MultiPolygon

Create a MultiPolygon with a List of Polygons

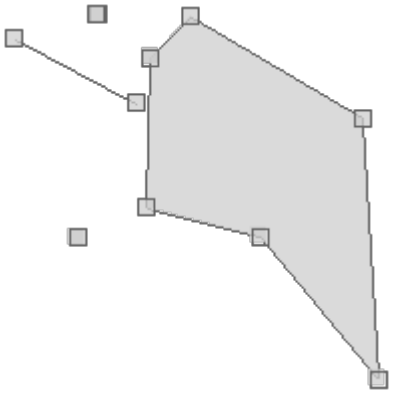
```
MultiPolygon multiPolygon = new MultiPolygon(
    new Polygon ([[
            [-122.2723388671875,  47.818687628247105],
            [-122.37945556640624, 47.66168780332917],
            [-121.95373535156249, 47.67093619422418],
            [-122.2723388671875,  47.818687628247105]
    ]]),
    new Polygon ([[
            [-122.76672363281249, 47.42437092240516],
            [-122.76672363281249, 47.59505101193038],
            [-122.52227783203125, 47.59505101193038],
            [-122.52227783203125, 47.42437092240516],
            [-122.76672363281249, 47.42437092240516]
    ]]),
    new Polygon ([[
            [-122.20367431640624, 47.543163654317304],
            [-122.3712158203125,  47.489368981370724],
            [-122.33276367187499, 47.35371061951363],
            [-122.11029052734374, 47.3704545156932],
            [-122.08831787109375, 47.286681888764214],
            [-122.28332519531249, 47.2270293988673],
            [-122.2174072265625,  47.154237057576594],
            [-121.904296875,      47.32579231609051],
            [-122.06085205078125, 47.47823216312885],
            [-122.20367431640624, 47.543163654317304]
    ]])
)
```



GeometryCollection

Create a GeometryCollection with a List of Geometries

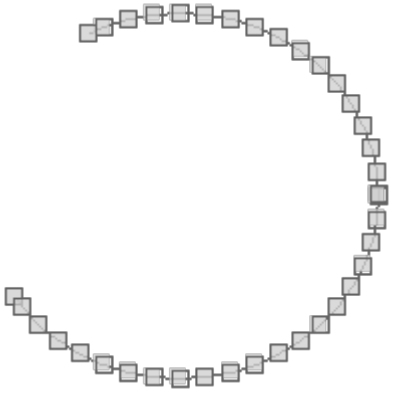
```
GeometryCollection geometryCollection = new GeometryCollection(
    new LineString ([-157.044, 58.722], [-156.461, 58.676]),
    new Point(-156.648, 58.739),
    new Polygon([[
         [-156.395, 58.7083],
         [-156.412, 58.6026],
         [-155.874, 58.5825],
         [-155.313, 58.4822],
         [-155.385, 58.6655],
         [-156.203, 58.7368],
         [-156.395, 58.7083]
    ]]),
    new Point(-156.741, 58.582)
)
```



CircularString

Create a CircularString with a List of Points

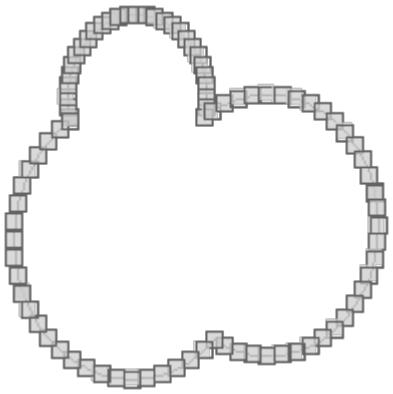
```
CircularString circularString = new CircularString([
        [-122.464599609375, 47.247542522268006],
        [-122.03613281249999, 47.37789454155521],
        [-122.37670898437499, 47.58393661978134]
])
```



CircularRing

Create a CircularRing with a List of Points

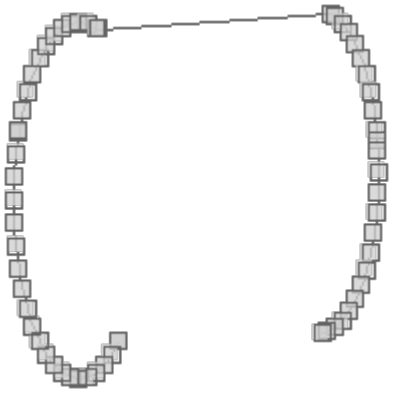
```
CircularRing circularRing = new CircularRing([
        [-118.47656249999999, 41.508577297439324],
        [-109.6875, 57.51582286553883],
        [-93.8671875, 42.032974332441405],
        [-62.57812500000001, 30.14512718337613],
        [-92.10937499999999, 7.36246686553575],
        [ -127.265625, 14.604847155053898],
        [-118.47656249999999, 41.508577297439324]
])
```



CompoundCurve

Create a CompoundCurve with a List of CircularStrings and LineStrings

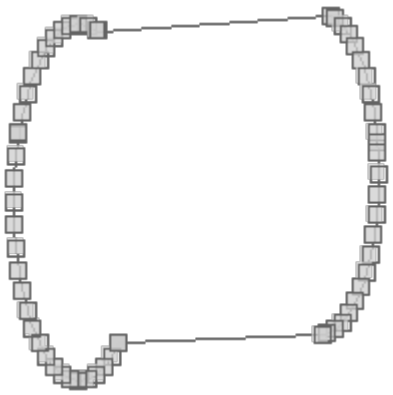
```
CompoundCurve compoundCurve = new CompoundCurve([
    new CircularString([
            [27.0703125, 23.885837699862005],
            [5.9765625, 40.17887331434696],
            [22.5, 47.98992166741417],
    ]),
    new LineString([
            [22.5, 47.98992166741417],
            [71.71875, 49.15296965617039],
    ]),
    new CircularString([
            [71.71875, 49.15296965617039],
            [81.5625, 39.36827914916011],
            [69.9609375, 24.5271348225978]

    ])
])
```



CompoundRing

Create a CompoundRing with a connected List of CircularStrings and LineStrings

```
CompoundRing compoundRing = new CompoundRing([
        new CircularString([
                [27.0703125, 23.885837699862005],
                [5.9765625, 40.17887331434696],
                [22.5, 47.98992166741417],
        ]),
        new LineString([
                [22.5, 47.98992166741417],
                [71.71875, 49.15296965617039],
        ]),
        new CircularString([
                [71.71875, 49.15296965617039],
                [81.5625, 39.36827914916011],
                [69.9609375, 24.5271348225978]

        ]),
        new LineString([
                [69.9609375, 24.5271348225978],
                [27.0703125, 23.885837699862005],
        ])
])
```



Points

Get x, y, and z values from a Point

```
Point point = new Point(-122.38632, 47.58208, 101.45)
println "X = ${point.x}"
println "Y = ${point.y}"
println "Z = ${point.z}"
```

```
X = -122.38632
Y = 47.58208
Z = 101.45
```

Add two Points together to create a MultiPoint

```
Point point1 = new Point(-122.38632, 47.58208)
Point point2 = new Point(-122.37001, 47.55868)
MultiPoint points = point1 + point2
```


Add a Point to a MultiPoint

```
MultiPoint multiPoint = new MultiPoint(
        new Point(-122.83813,47.05141),
        new Point(-122.38220,47.58023)
)
println multiPoint.wkt
MultiPoint newMultiPoint = multiPoint + new Point(-122.48657, 47.271775)
println newMultiPoint.wkt
```

```
MULTIPOINT ((-122.83813 47.05141), (-122.3822 47.58023))
MULTIPOINT ((-122.83813 47.05141), (-122.3822 47.58023), (-122.48657 47.271775))
```

MultiPoint

MultiPoint with extra Point

Calculate the angle between two points

```
Point point1 = new Point(-122.29980, 47.65058)
Point point2 = new Point(-120.54199, 46.64943)
double angleInDegrees = point1.getAngle(point2, "degrees")
println "Angle in degrees = ${angleInDegrees}"

double angleInRadians = point1.getAngle(point2, "radians")
println "Angle in radians = ${angleInRadians}"
```

```
Angle in degrees = -29.663413013476646
Angle in radians = -0.5177242244641005
```

Calculate the azimuth between two points

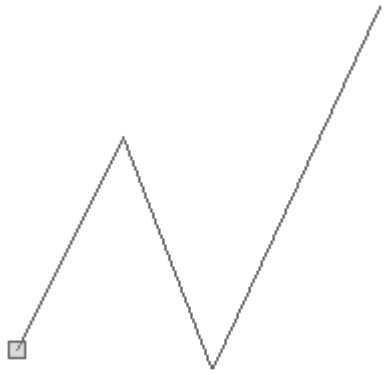
```
Point point1 = new Point(-122.29980, 47.65058)
Point point2 = new Point(-120.54199, 46.64943)
double azimuth = point1.getAzimuth(point2)
println "Azimuth = ${azimuth}"
```

```
Azimuth = 129.21026122904846
```

LineStrings

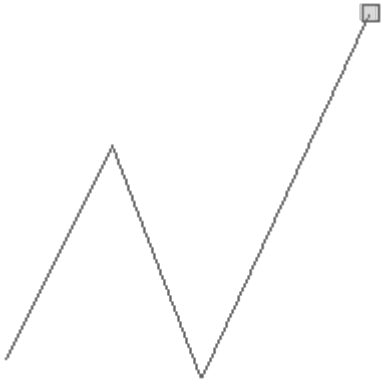
Get the start Point from a LineString

```
LineString lineString = new LineString(
        [3.1982421875, 43.1640625],
        [6.7138671875, 49.755859375],
        [9.7021484375, 42.5927734375],
        [15.3271484375, 53.798828125]
)
Point startPoint = lineString.startPoint
```



Get the end Point from a LineString

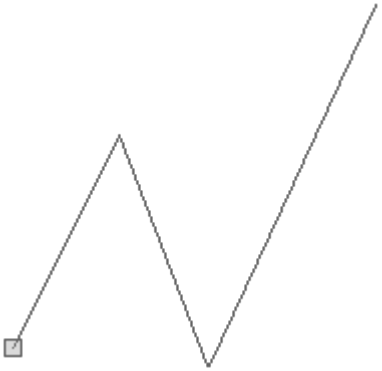
```
LineString lineString = new LineString(
        [3.1982421875, 43.1640625],
        [6.7138671875, 49.755859375],
        [9.7021484375, 42.5927734375],
        [15.3271484375, 53.798828125]
)
Point endPoint = lineString.endPoint
```



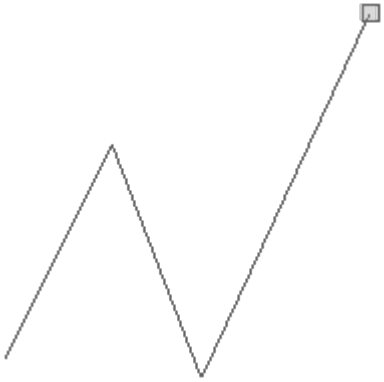
Reverse a LineString

```
LineString lineString = new LineString(
        [3.1982421875, 43.1640625],
        [6.7138671875, 49.755859375],
        [9.7021484375, 42.5927734375],
        [15.3271484375, 53.798828125]
)
Point startPoint = lineString.startPoint

LineString reversedLineString = lineString.reverse()
Point reversedStartPoint = reversedLineString.startPoint
```

Original LineString showing start point

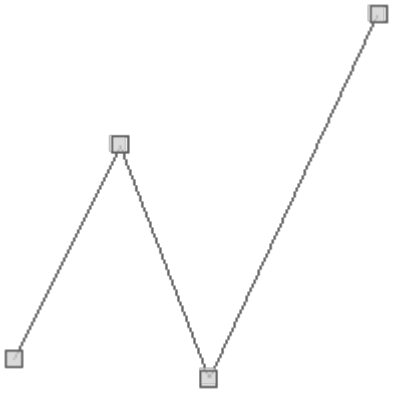


Reversed LineString showing start point

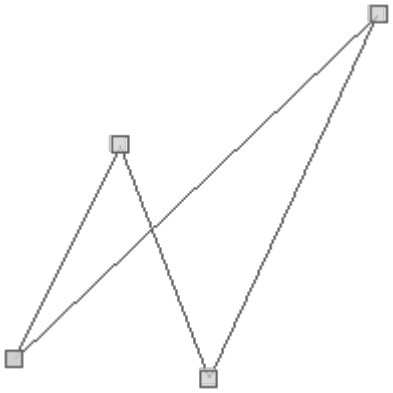


Determine if a LineString is closed or not

```
LineString lineString1 = new LineString(
        [3.1982421875, 43.1640625],
        [6.7138671875, 49.755859375],
        [9.7021484375, 42.5927734375],
        [15.3271484375, 53.798828125]
)
boolean isClosed1 = lineString1.closed
println "Is ${lineString1.wkt} closed? ${isClosed1}"

LineString lineString2 = new LineString(
        [3.1982421875, 43.1640625],
        [6.7138671875, 49.755859375],
        [9.7021484375, 42.5927734375],
        [15.3271484375, 53.798828125],
        [3.1982421875, 43.1640625]
)
boolean isClosed2 = lineString2.closed
println "Is ${lineString2.wkt} closed? ${isClosed2}"
```



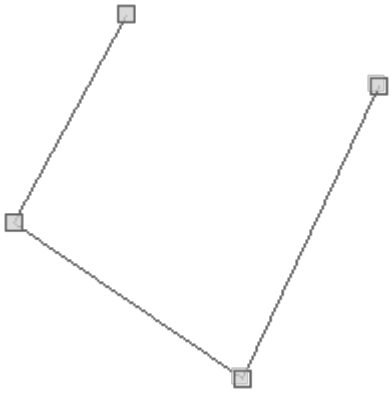
```
Is LINESTRING (3.1982421875 43.1640625, 6.7138671875 49.755859375, 9.7021484375
42.5927734375, 15.3271484375 53.798828125) closed? false
```



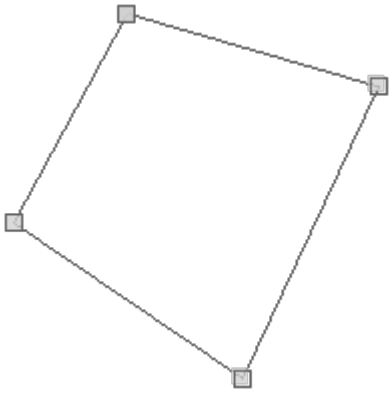
```
Is LINESTRING (3.1982421875 43.1640625, 6.7138671875 49.755859375, 9.7021484375
42.5927734375, 15.3271484375 53.798828125, 3.1982421875 43.1640625) closed? true
```

Determine if a LineString is a Ring or not

```
LineString lineString1 = new LineString(
    [-122.391428, 47.563300],
    [-122.391836, 47.562793],
    [-122.391010, 47.562417],
    [-122.390516, 47.563126]
)
boolean isRing1 = lineString1.ring
println "Is ${lineString1.wkt} a ring? ${isRing1}"

LineString lineString2 = new LineString(
    [-122.391428, 47.563300],
    [-122.391836, 47.562793],
    [-122.391010, 47.562417],
    [-122.390516, 47.563126],
    [-122.391428, 47.563300]
)
boolean isRing2 = lineString2.ring
println "Is ${lineString2.wkt} a ring? ${isRing2}"
```



```
Is LINESTRING (-122.391428 47.5633, -122.391836 47.562793, -122.39101 47.562417,
-122.390516 47.563126) a ring? false
```

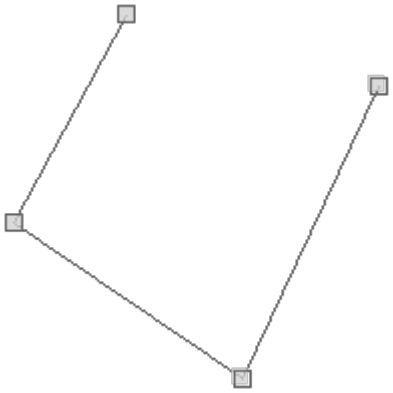


```
Is LINESTRING (-122.391428 47.5633, -122.391836 47.562793, -122.39101 47.562417,
-122.390516 47.563126, -122.391428 47.5633) a ring? true
```

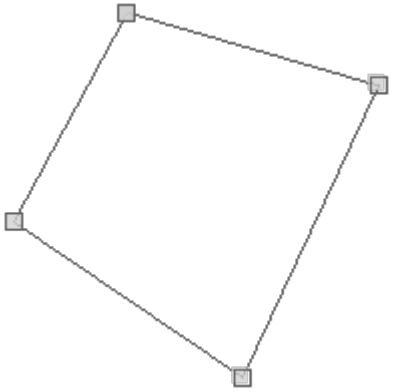
Close an open LineString to create a LinearRing

```
LineString lineString = new LineString(
        [-122.391428, 47.563300],
        [-122.391836, 47.562793],
        [-122.391010, 47.562417],
        [-122.390516, 47.563126]
)

LinearRing linearRing = lineString.close()
```

Open LineString



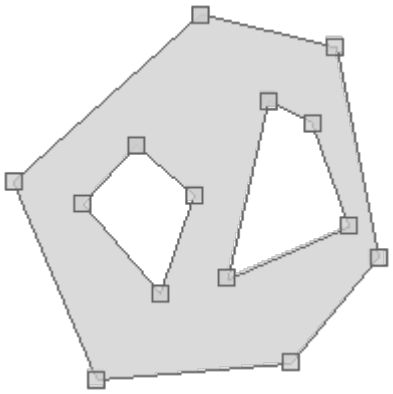
Closed LinearRing

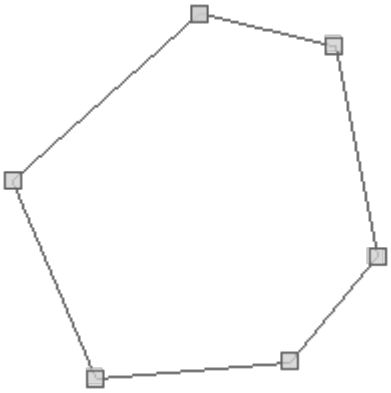


Polygons

Get a Polygon's exterior LinearRing

```
Polygon polygon = new Polygon(
    // Exterior Ring
    new LinearRing(
        [-122.39138603210449, 47.58659965790016],
        [-122.41250038146973, 47.57681522195182],
        [-122.40305900573729, 47.56523364515569],
        [-122.38117218017578, 47.56621817878201],
        [-122.3712158203125, 47.57235661809739],
        [-122.37602233886717, 47.584747123985615],
        [-122.39138603210449, 47.58659965790016]
    ),
    // Holes
    [
        new LinearRing(
            [-122.39859580993652, 47.578957532923376],
            [-122.40468978881836, 47.57548347095205],
            [-122.39593505859376, 47.570271945800094],
            [-122.3920726776123, 47.57606249728773],
            [-122.39859580993652, 47.578957532923376]
        ),
        new LinearRing(
            [-122.3836612701416, 47.58156292813543],
            [-122.38829612731934, 47.57114056934196],
            [-122.37456321716309, 47.57420959047542],
            [-122.37868309020995, 47.58023129789275],
            [-122.3836612701416, 47.58156292813543]
        )
    ]
)
LinearRing exteriorRing = polygon.getExteriorRing()
```

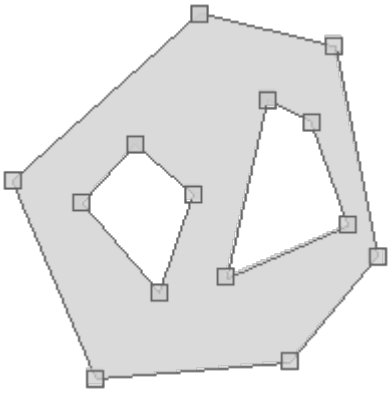


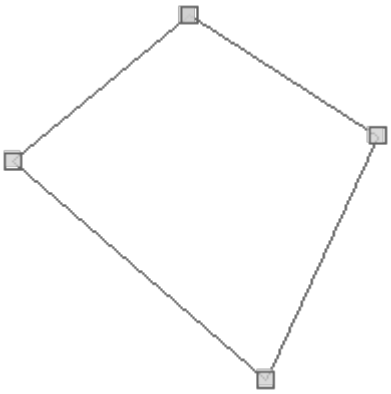


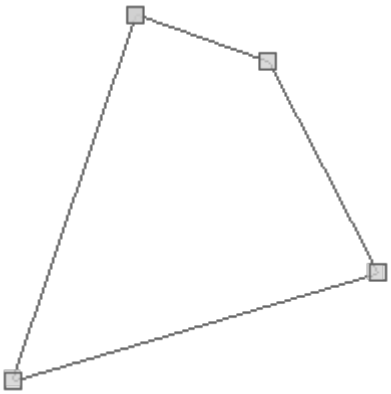
Get a Polygon's interior LinearRings

```
Polygon polygon = new Polygon(
        // Exterior Ring
        new LinearRing(
                [-122.39138603210449, 47.58659965790016],
                [-122.41250038146973, 47.57681522195182],
                [-122.40305900573729, 47.56523364515569],
                [-122.38117218017578, 47.56621817878201],
                [-122.3712158203125, 47.57235661809739],
                [-122.37602233886717, 47.584747123985615],
                [-122.39138603210449, 47.58659965790016]
        ),
        // Holes
        [
                new LinearRing(
                        [-122.39859580993652, 47.578957532923376],
                        [-122.40468978881836, 47.57548347095205],
                        [-122.39593505859376, 47.570271945800094],
                        [-122.3920726776123, 47.57606249728773],
                        [-122.39859580993652, 47.578957532923376]
                ),
                new LinearRing(
                        [-122.3836612701416, 47.58156292813543],
                        [-122.38829612731934, 47.57114056934196],
                        [-122.37456321716309, 47.57420959047542],
                        [-122.37868309020995, 47.58023129789275],
                        [-122.3836612701416, 47.58156292813543]
                )
        ]
)

println "# Interior Rings = ${polygon.numInteriorRing}"
(0..<polygon.numInteriorRing).each { int i ->
    println "  ${polygon.getInteriorRingN(i)}"
}

println "Interior Rings"
polygon.interiorRings.each { LinearRing ring ->
    println "  ${ring}"
}
```

```
# Interior Rings = 2
  LINEARRING (-122.39859580993652 47.578957532923376, -122.40468978881836
47.57548347095205, -122.39593505859376 47.570271945800094, -122.3920726776123
47.57606249728773, -122.39859580993652 47.578957532923376)
  LINEARRING (-122.3836612701416 47.58156292813543, -122.38829612731934
47.57114056934196, -122.37456321716309 47.57420959047542, -122.37868309020995
47.58023129789275, -122.3836612701416 47.58156292813543)

Interior Rings
  LINEARRING (-122.39859580993652 47.578957532923376, -122.40468978881836
47.57548347095205, -122.39593505859376 47.570271945800094, -122.3920726776123
47.57606249728773, -122.39859580993652 47.578957532923376)
  LINEARRING (-122.3836612701416 47.58156292813543, -122.38829612731934
47.57114056934196, -122.37456321716309 47.57420959047542, -122.37868309020995
47.58023129789275, -122.3836612701416 47.58156292813543)
```

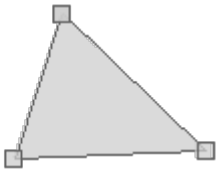


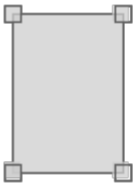




Add a Polygon to another Polygon to create a MultiPolygon

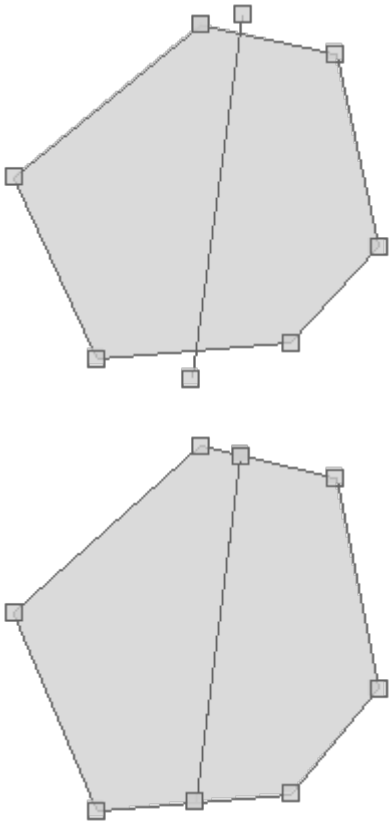
```
Polygon polygon1 = new Polygon ([[
  [-122.2723388671875,  47.818687628247105],
  [-122.37945556640624, 47.66168780332917],
  [-121.95373535156249, 47.67093619422418],
  [-122.2723388671875,  47.818687628247105]
]])
Polygon polygon2 = new Polygon ([[
  [-122.76672363281249, 47.42437092240516],
  [-122.76672363281249, 47.59505101193038],
  [-122.52227783203125, 47.59505101193038],
  [-122.52227783203125, 47.42437092240516],
  [-122.76672363281249, 47.42437092240516]
]])
MultiPolygon multiPolygon = polygon1 + polygon2
```





Split a Polygon with a LineString

```
Polygon polygon = new Polygon(
    new LinearRing(
        [-122.39138603210449, 47.58659965790016],
        [-122.41250038146973, 47.57681522195182],
        [-122.40305900573729, 47.56523364515569],
        [-122.38117218017578, 47.56621817878201],
        [-122.3712158203125, 47.57235661809739],
        [-122.37602233886717, 47.584747123985615],
        [-122.39138603210449, 47.58659965790016]
    )
)

LineString lineString = new LineString([
    [-122.3924160003662, 47.56395951534652],
    [-122.38649368286131, 47.58729434121508]
])

MultiPolygon multiPolygon = polygon.split(lineString)
```



Processing Geometries

Get the geometry type (Point, LineString, Polygon, ect...) from a Geometry

```
Geometry geom = Geometry.fromString("POINT (-124.80 48.92)")
String type = geom.geometryType
println type
```

```
Point
```

Determine if one Geometry exactly equal another Geometry.

```
Point point1 = new Point(-121.915, 47.390)
Point point2 = new Point(-121.915, 47.390)
Point point3 = new Point(-121.409, 47.413)

boolean does1equal2 = point1.equals(point2)
println "Does ${point1} equal ${point2}? ${does1equal2 ? 'Yes' : 'No'}"

boolean does1equal3 = point1.equals(point3)
println "Does ${point1} equal ${point3}? ${does1equal3 ? 'Yes' : 'No'}"

boolean does2equal3 = point2.equals(point3)
println "Does ${point2} equal ${point3}? ${does2equal3 ? 'Yes' : 'No'}"
```

```
Does POINT (-121.915 47.39) equal POINT (-121.915 47.39)? Yes
Does POINT (-121.915 47.39) equal POINT (-121.409 47.413)? No
Does POINT (-121.915 47.39) equal POINT (-121.409 47.413)? No
```

Determine if one Geometry equals another Geometry topologically.

```
Point point1 = new Point(-121.915, 47.390)
Point point2 = new Point(-121.915, 47.390)
Point point3 = new Point(-121.409, 47.413)

boolean does1equal2 = point1.equalsTopo(point2)
println "Does ${point1} equal ${point2}? ${does1equal2 ? 'Yes' : 'No'}"

boolean does1equal3 = point1.equalsTopo(point3)
println "Does ${point1} equal ${point3}? ${does1equal3 ? 'Yes' : 'No'}"

boolean does2equal3 = point2.equalsTopo(point3)
println "Does ${point2} equal ${point3}? ${does2equal3 ? 'Yes' : 'No'}"
```

```
Does POINT (-121.915 47.39) equal POINT (-121.915 47.39)? Yes
Does POINT (-121.915 47.39) equal POINT (-121.409 47.413)? No
Does POINT (-121.915 47.39) equal POINT (-121.409 47.413)? No
```

Determine if one Geometry equals another Geometry when both are normalized.

```
Geometry geom1 = Geometry.fromWKT("POLYGON ((2 4, 1 3, 2 1, 6 1, 6 3, 4 4, 2 4))")
Geometry geom2 = Geometry.fromWKT("POLYGON ((1 3, 2 4, 4 4, 6 3, 6 1, 2 1, 1 3))")
Geometry geom3 = Geometry.fromWKT("POLYGON ((1 1, 1 4, 4 4, 4 1, 1 1))")

boolean does1equal2 = geom1.equalsNorm(geom2)
println "Does ${geom1} equal ${geom2}? ${does1equal2 ? 'Yes' : 'No'}"

boolean does1equal3 = geom1.equalsNorm(geom3)
println "Does ${geom1} equal ${geom3}? ${does1equal3 ? 'Yes' : 'No'}"

boolean does2equal3 = geom2.equalsNorm(geom3)
println "Does ${geom2} equal ${geom3}? ${does2equal3 ? 'Yes' : 'No'}"
```

```
Does POLYGON ((2 4, 1 3, 2 1, 6 1, 6 3, 4 4, 2 4)) equal POLYGON ((1 3, 2 4, 4 4, 6 3,
6 1, 2 1, 1 3))? Yes
Does POLYGON ((2 4, 1 3, 2 1, 6 1, 6 3, 4 4, 2 4)) equal POLYGON ((1 1, 1 4, 4 4, 4 1,
1 1))? No
Does POLYGON ((1 3, 2 4, 4 4, 6 3, 6 1, 2 1, 1 3)) equal POLYGON ((1 1, 1 4, 4 4, 4 1,
1 1))? No
```

Get a Geometry by index in a GeometryCollection

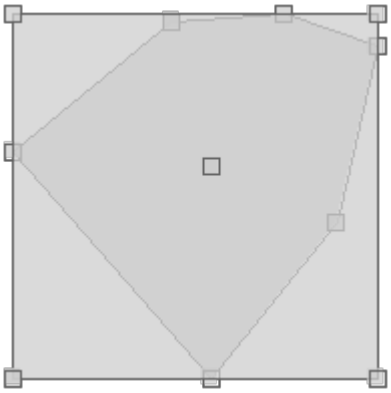
```
MultiPoint multiPoint = new MultiPoint([
        new Point(-122.3876953125, 47.5820839916191),
        new Point(-122.464599609375, 47.25686404408872),
        new Point(-122.48382568359374, 47.431803338643334)
])
Point p1 = multiPoint[0]
println p1

Point p2 = multiPoint[1]
println p2

Point p3 = multiPoint[2]
println p3
```

```
POINT (-122.3876953125 47.5820839916191)
POINT (-122.464599609375 47.25686404408872)
POINT (-122.48382568359374 47.431803338643334)
```

Cast a Geometry to a Bounds or to a Point

```
Geometry geometry = new Polygon([[
    [-121.915, 47.390],
    [-122.640, 46.995],
    [-121.739, 46.308],
    [-121.168, 46.777],
    [-120.981, 47.316],
    [-121.409, 47.413],
    [-121.915, 47.390]
]])

Bounds bounds = geometry as Bounds
println bounds

Point point = geometry as Point
println point
```

```
(-122.64,46.308,-120.981,47.413)
POINT (-121.73789467295867 46.95085967283822)
```



Get the area of a Geometry

```
Polygon polygon = new Polygon([[
    [-124.80, 48.92],
    [-126.21, 45.33],
    [-114.60, 45.08],
    [-115.31, 51.17],
    [-121.99, 52.05],
    [-124.80, 48.92]
]])
double area = polygon.area
println area
```

```
62.4026
```

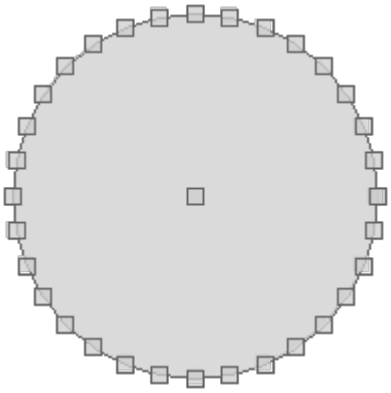
Get the length of a Geometry

```
LineString lineString = new LineString([-122.69, 49.61], [-99.84, 45.33])
double length = lineString.length
println length
```

```
23.24738479915536
```

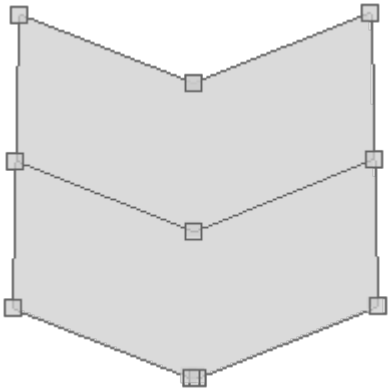
Buffer a Point

```
Point point = new Point(-123,46)
Geometry bufferedPoint = point.buffer(2)
```



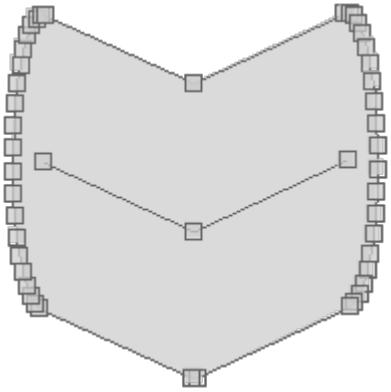
Buffer a LineString with a butt cap

```
LineString line = new LineString([
    [-122.563, 47.576],
    [-112.0166, 46.589],
    [-101.337, 47.606]
])
Geometry bufferedLine1 = line.buffer(2.1, 10, Geometry.CAP_BUTT)
```



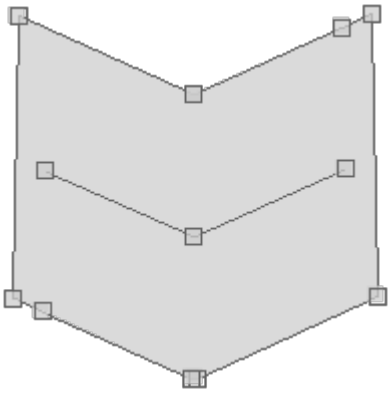
Buffer a LineString with a round cap

```
Geometry bufferedLine2 = line.buffer(2.1, 10, Geometry.CAP_ROUND)
```



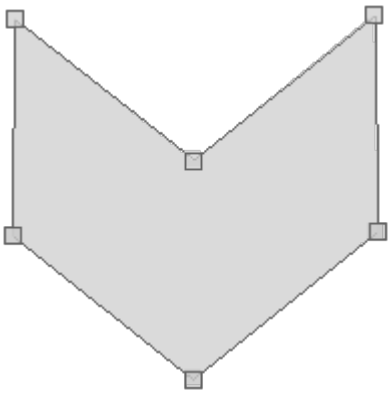
Buffer a LineString with a square cap

```
Geometry bufferedLine3 = line.buffer(2.1, 10, Geometry.CAP_SQUARE)
```



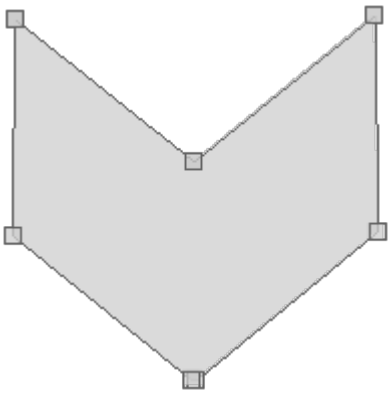
Buffer a LineString on the right side only

```
LineString line = new LineString([
        [-122.563, 47.576],
        [-112.0166, 46.589],
        [-101.337, 47.606]
])
Geometry rightBufferedLine = line.singleSidedBuffer(1.5)
```



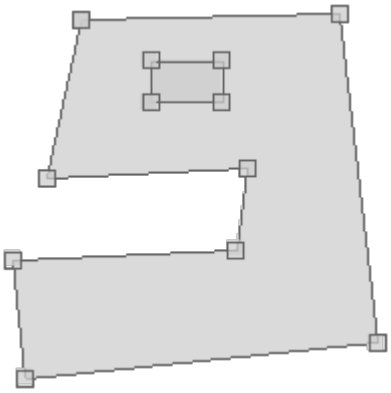
Buffer a LineString on the left side only

```
Geometry leftBufferedLine = line.singleSidedBuffer(-1.5)
```



Check whether a Geometry contains another Geometry

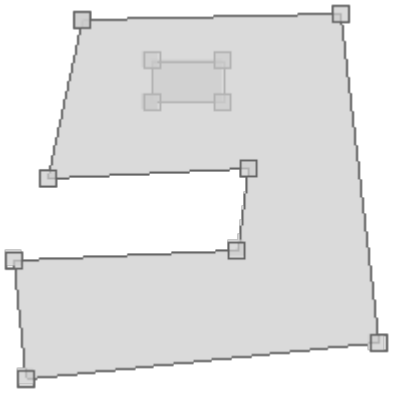
```
Polygon polygon1 = new Polygon([[
    [-120.739, 48.151],
    [-121.003, 47.070],
    [-119.465, 47.137],
    [-119.553, 46.581],
    [-121.267, 46.513],
    [-121.168, 45.706],
    [-118.476, 45.951],
    [-118.762, 48.195],
    [-120.739, 48.151]
]])

Polygon polygon2 = new Polygon([[
    [-120.212, 47.591],
    [-119.663, 47.591],
    [-119.663, 47.872],
    [-120.212, 47.872],
    [-120.212, 47.591]
]])

boolean contains = polygon1.contains(polygon2)
println contains
```



```
true
```

Check whether a Geometry is within another Geometry

```
Polygon polygon1 = new Polygon([[
        [-120.212, 47.591],
        [-119.663, 47.591],
        [-119.663, 47.872],
        [-120.212, 47.872],
        [-120.212, 47.591]
]])

Polygon polygon2 = new Polygon([[
        [-120.739, 48.151],
        [-121.003, 47.070],
        [-119.465, 47.137],
        [-119.553, 46.581],
        [-121.267, 46.513],
        [-121.168, 45.706],
        [-118.476, 45.951],
        [-118.762, 48.195],
        [-120.739, 48.151]
]])

boolean within = polygon1.within(polygon2)
println within
```



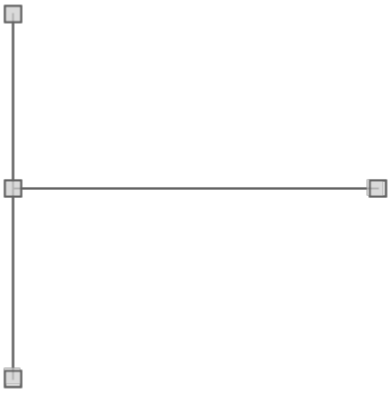
```
true
```

```
Polygon polygon3 = new Polygon([[
        [-120.563, 46.739],
        [-119.948, 46.739],
        [-119.948, 46.965],
        [-120.563, 46.965],
        [-120.563, 46.739]
]])

within = polygon1.within(polygon3)
println within
```

```
false
```

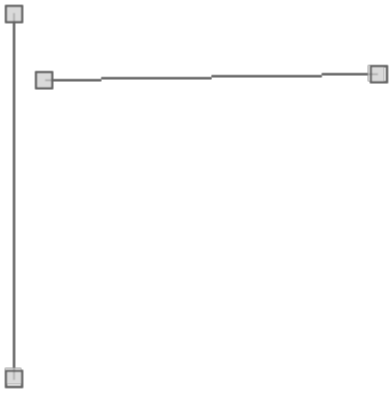
Check whether a Geometry touches another Geometry

```
LineString line1 = new LineString([
    [-122.38651514053345, 47.58219978280006],
    [-122.38651514053345, 47.58020234903306]
])

LineString line2 = new LineString([
    [-122.38651514053345, 47.58124449789785],
    [-122.38333940505981, 47.58124449789785]
])

boolean touches = line1.touches(line2)
```



```
true
```

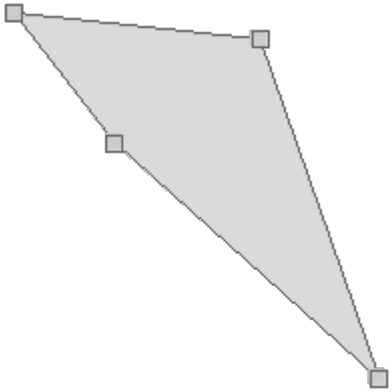
```
LineString line3 = new LineString([
    [-122.386257648468, 47.58183793450921],
    [-122.38348960876465, 47.5818668824645]
])

touches = line1.touches(line3)
```



```
false
```

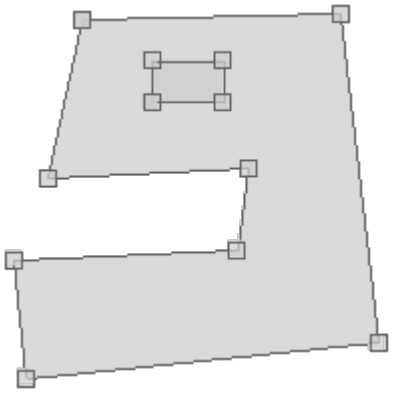
Create a convexhull Geometry around a Geometry

```
Geometry geometry = new MultiPoint(
    new Point(-119.882, 47.279),
    new Point(-100.195, 46.316),
    new Point(-111.796, 42.553),
    new Point(-90.7031, 34.016)
)
Geometry convexHull = geometry.convexHull
```



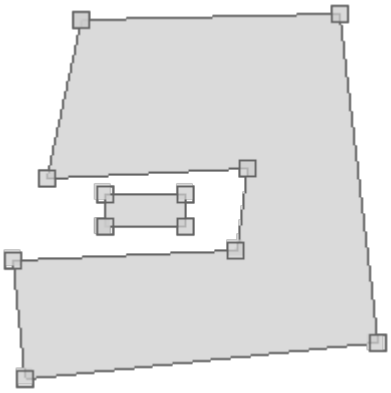
Check whether a Geometry covers another Geometry

```
Polygon polygon1 = new Polygon([[
        [-120.739, 48.151],
        [-121.003, 47.070],
        [-119.465, 47.137],
        [-119.553, 46.581],
        [-121.267, 46.513],
        [-121.168, 45.706],
        [-118.476, 45.951],
        [-118.762, 48.195],
        [-120.739, 48.151]
]])

Polygon polygon2 = new Polygon([[
        [-120.212, 47.591],
        [-119.663, 47.591],
        [-119.663, 47.872],
        [-120.212, 47.872],
        [-120.212, 47.591]
]])

boolean isCovered = polygon1.covers(polygon2)
println isCovered
```



```
true
```

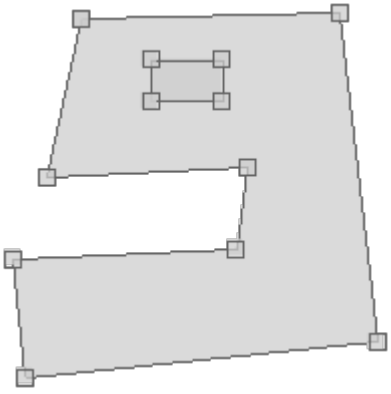
```
Polygon polygon3 = new Polygon([[
        [-120.563, 46.739],
        [-119.948, 46.739],
        [-119.948, 46.965],
        [-120.563, 46.965],
        [-120.563, 46.739]
]])

isCovered = polygon1.covers(polygon3)
println isCovered
```



```
false
```

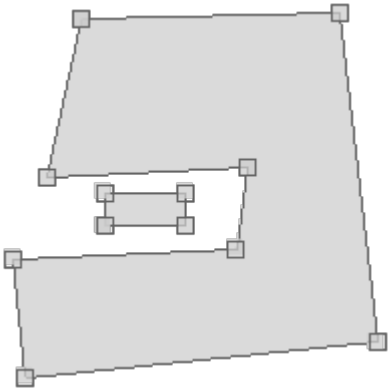
Check whether a Geometry is covered by another Geometry

```
Polygon polygon1 = new Polygon([[
        [-120.739, 48.151],
        [-121.003, 47.070],
        [-119.465, 47.137],
        [-119.553, 46.581],
        [-121.267, 46.513],
        [-121.168, 45.706],
        [-118.476, 45.951],
        [-118.762, 48.195],
        [-120.739, 48.151]
]])

Polygon polygon2 = new Polygon([[
        [-120.212, 47.591],
        [-119.663, 47.591],
        [-119.663, 47.872],
        [-120.212, 47.872],
        [-120.212, 47.591]
]])

boolean isCoveredBy = polygon2.coveredBy(polygon1)
println isCoveredBy
```



```
true
```

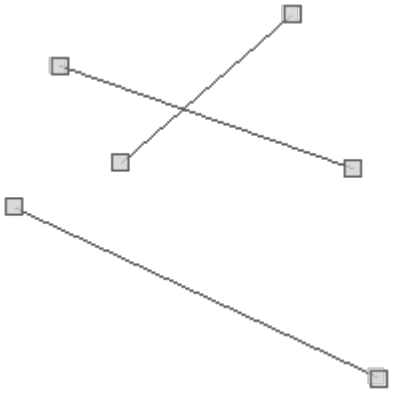
```
Polygon polygon3 = new Polygon([[
        [-120.563, 46.739],
        [-119.948, 46.739],
        [-119.948, 46.965],
        [-120.563, 46.965],
        [-120.563, 46.739]
]])

isCoveredBy = polygon3.coveredBy(polygon1)
println isCoveredBy
```



```
false
```

Check whether one Geometry crosses another Geometry

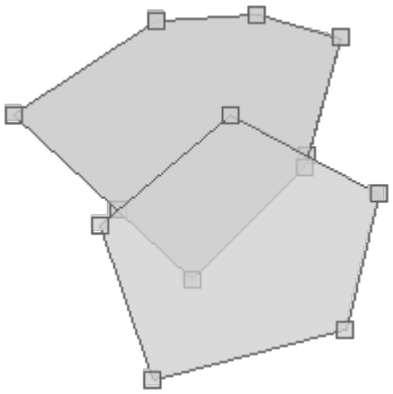
```
LineString line1 = new LineString([[-122.486, 47.256], [-121.695, 46.822]])
LineString line2 = new LineString([[-122.387, 47.613], [-121.750, 47.353]])
LineString line3 = new LineString([[-122.255, 47.368], [-121.882, 47.746]])

boolean doesCross12 = line1.crosses(line2)
println doesCross12

boolean doesCross13 = line1.crosses(line3)
println doesCross13

boolean doesCross23 = line2.crosses(line3)
println doesCross23
```



```
false
false
true
```

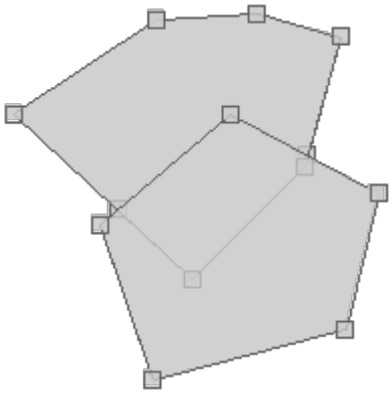
Calculate the difference between two Geometries

```
Polygon polygon1 = new Polygon([[
        [-121.915, 47.390],
        [-122.640, 46.995],
        [-121.739, 46.308],
        [-121.168, 46.777],
        [-120.981, 47.316],
        [-121.409, 47.413],
        [-121.915, 47.390]
]])

Polygon polygon2 = new Polygon([[
        [-120.794, 46.664],
        [-121.541, 46.995],
        [-122.200, 46.536],
        [-121.937, 45.890],
        [-120.959, 46.096],
        [-120.794, 46.664]
]])

Geometry difference = polygon1.difference(polygon2)
```



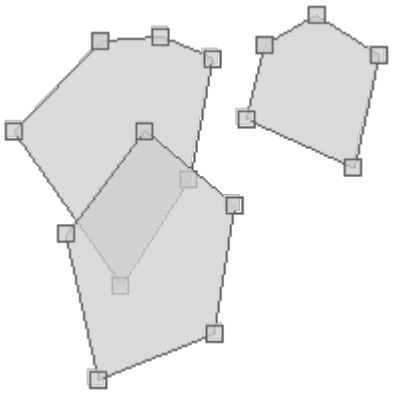
Calculate the symmetric difference between two Geometries

```
Polygon polygon1 = new Polygon([[
        [-121.915, 47.390],
        [-122.640, 46.995],
        [-121.739, 46.308],
        [-121.168, 46.777],
        [-120.981, 47.316],
        [-121.409, 47.413],
        [-121.915, 47.390]
]])

Polygon polygon2 = new Polygon([[
        [-120.794, 46.664],
        [-121.541, 46.995],
        [-122.200, 46.536],
        [-121.937, 45.890],
        [-120.959, 46.096],
        [-120.794, 46.664]
]])

Geometry symDifference = polygon1.symDifference(polygon2)
```



Check whether one Geometry is disjoint from another Geometry

```
Polygon polygon1 = new Polygon([[
        [-121.915, 47.390],
        [-122.640, 46.995],
        [-121.739, 46.308],
        [-121.168, 46.777],
        [-120.981, 47.316],
        [-121.409, 47.413],
        [-121.915, 47.390]
]])

Polygon polygon2 = new Polygon([[
        [-120.794, 46.664],
        [-121.541, 46.995],
        [-122.200, 46.536],
        [-121.937, 45.890],
        [-120.959, 46.096],
        [-120.794, 46.664]
]])

Polygon polygon3 = new Polygon([[
        [-120.541, 47.376],
        [-120.695, 47.047],
        [-119.794, 46.830],
        [-119.586, 47.331],
        [-120.102, 47.509],
        [-120.541, 47.376]
]])

boolean isDisjoint12 = polygon1.disjoint(polygon2)
println isDisjoint12

boolean isDisjoint13 = polygon1.disjoint(polygon3)
println isDisjoint13

boolean isDisjoint23 = polygon2.disjoint(polygon3)
println isDisjoint23
```



```
false
true
true
```

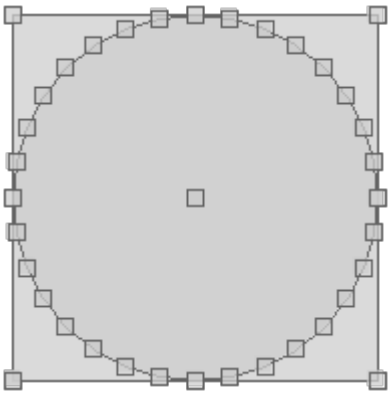
Calculate the distance bewteen two Geometries

```
Point point1 = new Point(-122.442, 47.256)
Point point2 = new Point(-122.321, 47.613)
double distance = point1.distance(point2)
println distance
```

```
0.37694827231332195
```

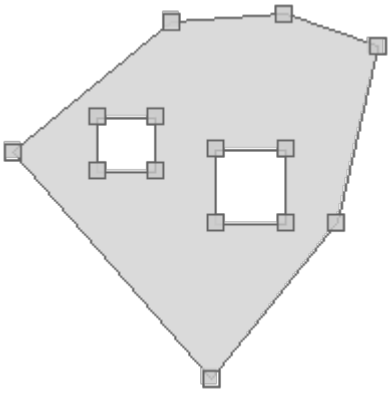
Get Bounds from a Geometry

```
Point point = new Point(-123,46)
Polygon polygon = point.buffer(2)
Bounds bounds = polygon.bounds
```



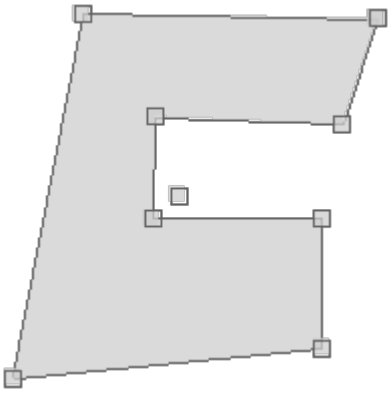
Get the Boundary from a Geometry

```
Polygon polygon = new Polygon([
    [
            [-121.915, 47.390],
            [-122.640, 46.995],
            [-121.739, 46.308],
            [-121.168, 46.777],
            [-120.981, 47.316],
            [-121.409, 47.413],
            [-121.915, 47.390]
    ],
    [
            [-122.255, 46.935],
            [-121.992, 46.935],
            [-121.992, 47.100],
            [-122.255, 47.100],
            [-122.255, 46.935]
    ],
    [
            [-121.717, 46.777],
            [-121.398, 46.777],
            [-121.398, 47.002],
            [-121.717, 47.002],
            [-121.717, 46.777]
    ]
])
Geometry boundary = polygon.boundary
```



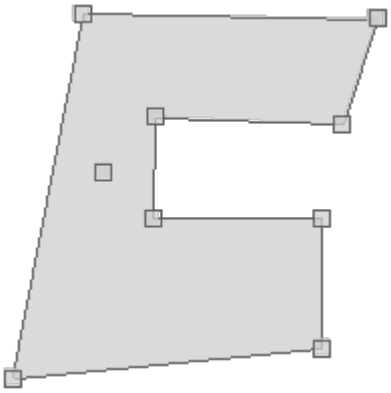
Get the Centroid from a Geometry

```
Polygon polygon = new Polygon([[
       [-118.937, 48.327],
       [-121.157, 48.356],
       [-121.684, 46.331],
       [-119.355, 46.498],
       [-119.355, 47.219],
       [-120.629, 47.219],
       [-120.607, 47.783],
       [-119.201, 47.739],
       [-118.937, 48.327]
]])
Geometry centroid = polygon.centroid
```



Get the Interior Point from a Geometry

```
Polygon polygon = new Polygon([[
       [-118.937, 48.327],
       [-121.157, 48.356],
       [-121.684, 46.331],
       [-119.355, 46.498],
       [-119.355, 47.219],
       [-120.629, 47.219],
       [-120.607, 47.783],
       [-119.201, 47.739],
       [-118.937, 48.327]
]])
Geometry interiorPoint = polygon.interiorPoint
```



Get the number of Geometries

```
MultiPoint multiPoint = new MultiPoint([
        new Point(-122.3876953125, 47.5820839916191),
        new Point(-122.464599609375, 47.25686404408872),
        new Point(-122.48382568359374, 47.431803338643334)
])
int number = multiPoint.numGeometries
println number
```

```
3
```

Get a Geometry by index

```
MultiPoint multiPoint = new MultiPoint([
        new Point(-122.3876953125, 47.5820839916191),
        new Point(-122.464599609375, 47.25686404408872),
        new Point(-122.48382568359374, 47.431803338643334)
])
(0..<multiPoint.getNumGeometries()).each { int i ->
    Geometry geometry = multiPoint.getGeometryN(i)
    println geometry.wkt
}
```

```
POINT (-122.3876953125 47.5820839916191)
POINT (-122.464599609375 47.25686404408872)
POINT (-122.48382568359374 47.431803338643334)
```

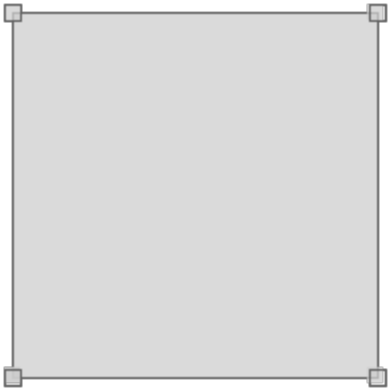
Get a List of Geometries

```
MultiPoint multiPoint = new MultiPoint([
        new Point(-122.3876953125, 47.5820839916191),
        new Point(-122.464599609375, 47.25686404408872),
        new Point(-122.48382568359374, 47.431803338643334)
])
List<Geometry> geometries = multiPoint.geometries
geometries.each { Geometry geometry ->
    println geometry.wkt
}
```

```
POINT (-122.3876953125 47.5820839916191)
POINT (-122.464599609375 47.25686404408872)
POINT (-122.48382568359374 47.431803338643334)
```

Get the number of Points in a Geometry

```
Polygon polygon = new Polygon([[
    [-120.563, 46.739],
    [-119.948, 46.739],
    [-119.948, 46.965],
    [-120.563, 46.965],
    [-120.563, 46.739]
]])
int number = polygon.numPoints
println number
```



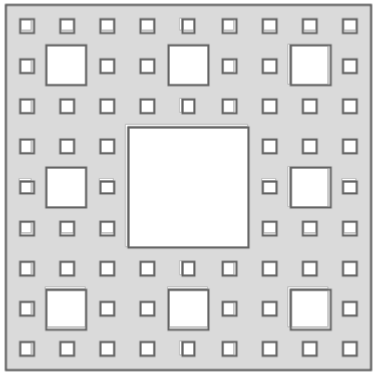
```
5
```

Create a Geometry of a String

```
Geometry geometry = Geometry.createFromText("Geo")
```

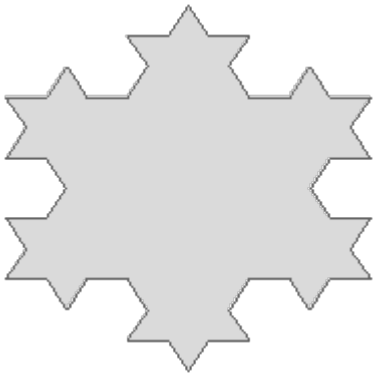
Create a Sierpinski Carpet in a given Bounds and with a number of points

```
Bounds bounds = new Bounds(21.645,36.957,21.676,36.970, "EPSG:4326")
Geometry geometry = Geometry.createSierpinskiCarpet(bounds, 50)
```



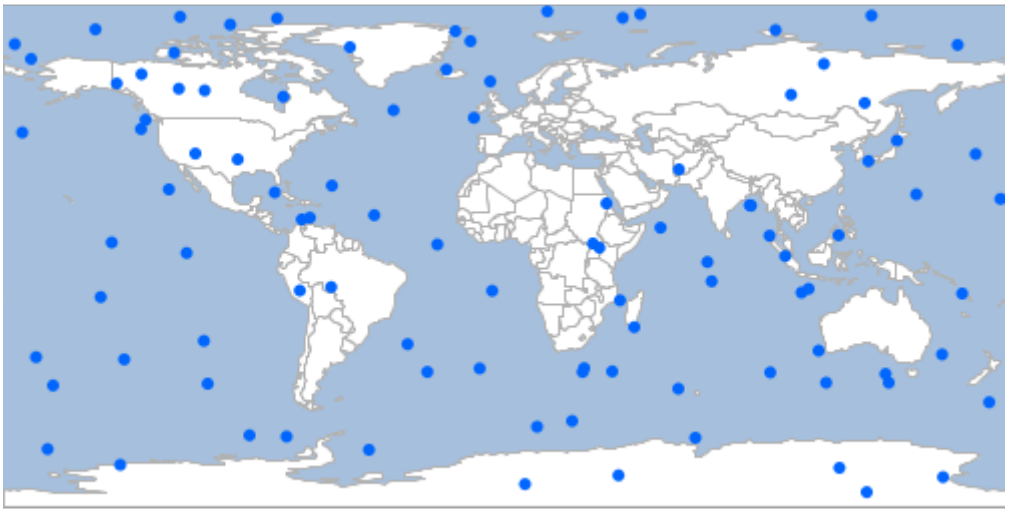
Create a Kock Snowflake in a given Bounds and with a number of points

```
Bounds bounds = new Bounds(21.645,36.957,21.676,36.970, "EPSG:4326")
Geometry geometry = Geometry.createKochSnowflake(bounds, 50)
```



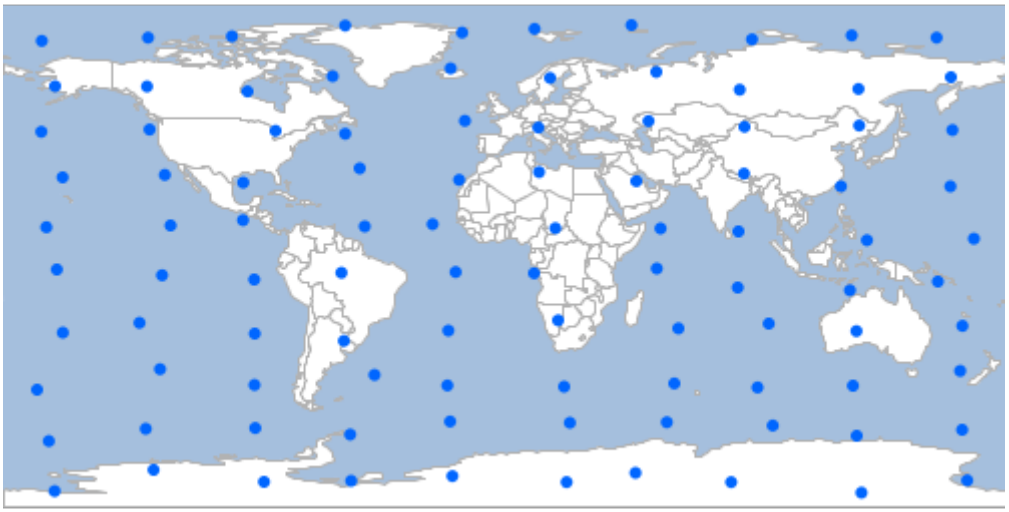
Create a number of random points within a given Geometry

```
Geometry geometry = new Bounds(-180, -90, 180, 90).geometry
MultiPoint randomPoints = Geometry.createRandomPoints(geometry, 100)
```



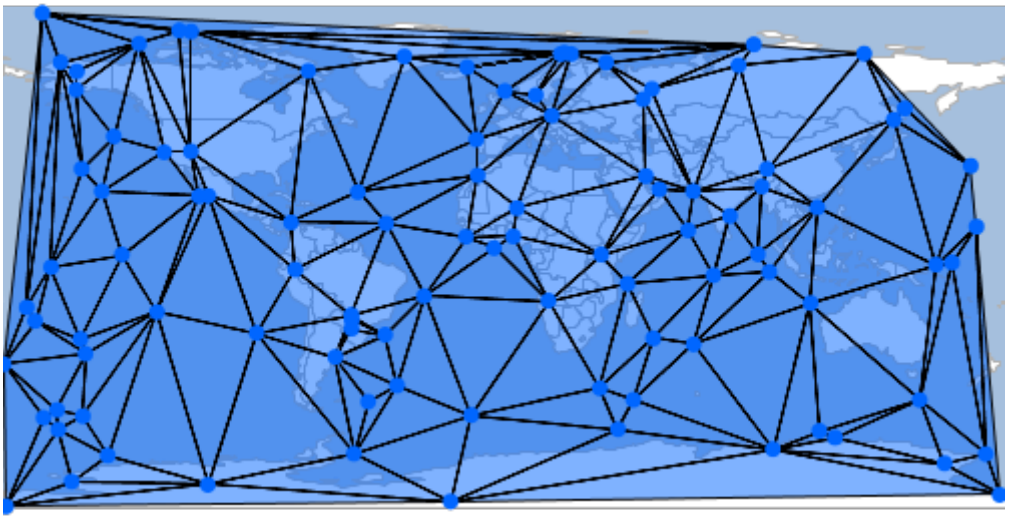
Create a number of random points within a given Geometry where the points are contrained to the cells of a grid

```
Bounds bounds = new Bounds(-180, -90, 180, 90)
MultiPoint randomPoints = Geometry.createRandomPointsInGrid(bounds, 100, true, 0.5)
```



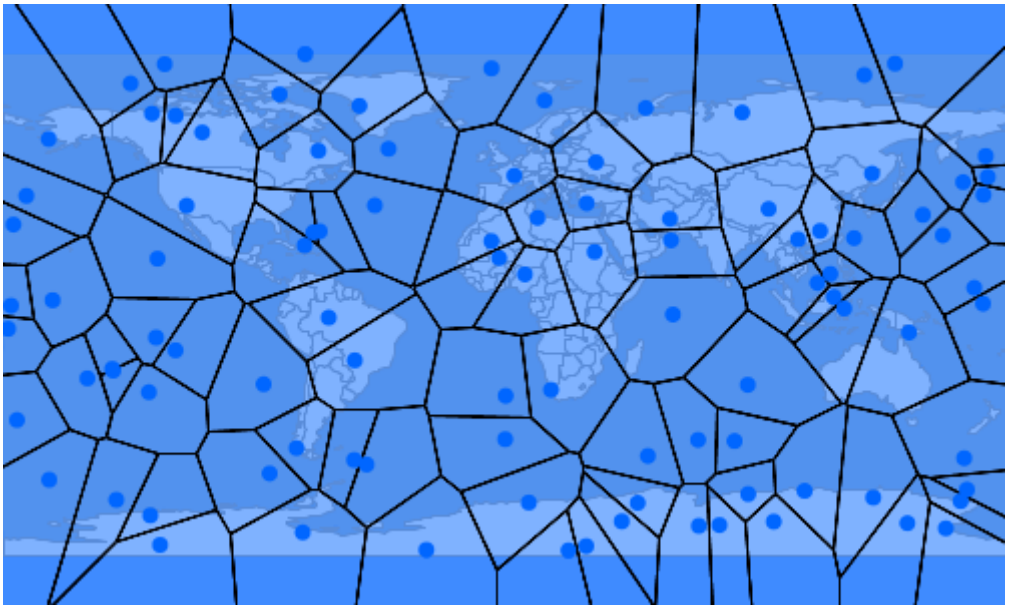
Create a delaunay triangle diagram around a Geometry

```
Geometry points = Geometry.createRandomPoints(new Bounds(-180, -90, 180, 90).geometry,
100)
Geometry delaunayTriangle = points.delaunayTriangleDiagram
```



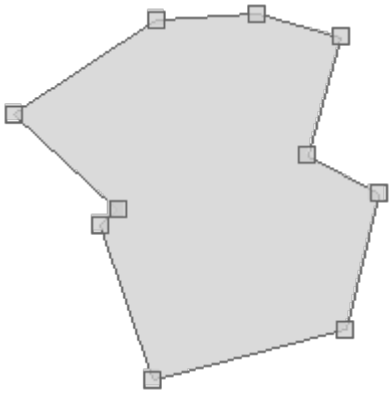
Create a voronoi diagram around a Geometry

```
Geometry points = Geometry.createRandomPoints(new Bounds(-180, -90, 180, 90).geometry,
100)
Geometry voronoiDiagram = points.voronoiDiagram
```



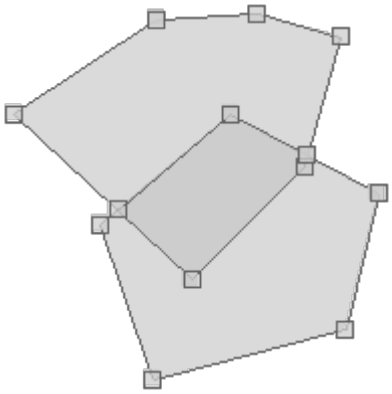
Calculate the union of two Geometries

```
Polygon polygon1 = new Polygon([[
        [-121.915, 47.390],
        [-122.640, 46.995],
        [-121.739, 46.308],
        [-121.168, 46.777],
        [-120.981, 47.316],
        [-121.409, 47.413],
        [-121.915, 47.390]
]])

Polygon polygon2 = new Polygon([[
        [-120.794, 46.664],
        [-121.541, 46.995],
        [-122.200, 46.536],
        [-121.937, 45.890],
        [-120.959, 46.096],
        [-120.794, 46.664]
]])

Geometry union = polygon1.union(polygon2)
```



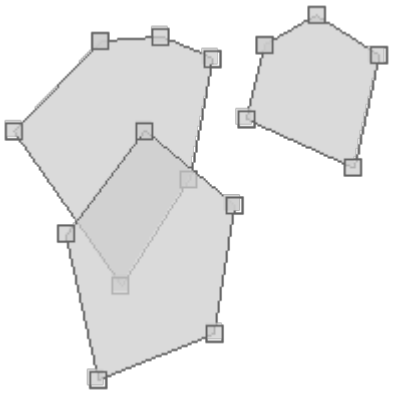
Calculate the intersection between two Geometries

```
Polygon polygon1 = new Polygon([[
        [-121.915, 47.390],
        [-122.640, 46.995],
        [-121.739, 46.308],
        [-121.168, 46.777],
        [-120.981, 47.316],
        [-121.409, 47.413],
        [-121.915, 47.390]
]])

Polygon polygon2 = new Polygon([[
        [-120.794, 46.664],
        [-121.541, 46.995],
        [-122.200, 46.536],
        [-121.937, 45.890],
        [-120.959, 46.096],
        [-120.794, 46.664]
]])

Geometry intersection = polygon1.intersection(polygon2)
```



Check whether one Geometry intersects from another Geometry

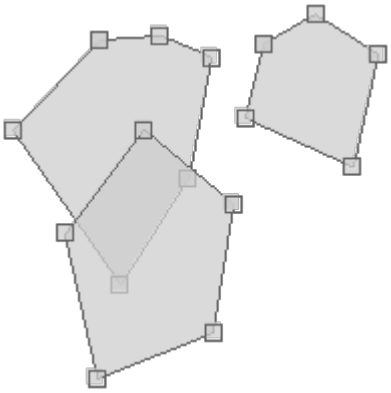
```
Polygon polygon1 = new Polygon([[
        [-121.915, 47.390],
        [-122.640, 46.995],
        [-121.739, 46.308],
        [-121.168, 46.777],
        [-120.981, 47.316],
        [-121.409, 47.413],
        [-121.915, 47.390]
]])

Polygon polygon2 = new Polygon([[
        [-120.794, 46.664],
        [-121.541, 46.995],
        [-122.200, 46.536],
        [-121.937, 45.890],
        [-120.959, 46.096],
        [-120.794, 46.664]
]])

Polygon polygon3 = new Polygon([[
        [-120.541, 47.376],
        [-120.695, 47.047],
        [-119.794, 46.830],
        [-119.586, 47.331],
        [-120.102, 47.509],
        [-120.541, 47.376]
]])

boolean does1intersect2 = polygon1.intersects(polygon2)
println does1intersect2

boolean does1intersect3 = polygon1.intersects(polygon3)
println does1intersect3

boolean does2intersect3 = polygon2.intersects(polygon3)
println does2intersect3
```



```
true
false
false
```

Check whether one Geometry overlaps from another Geometry

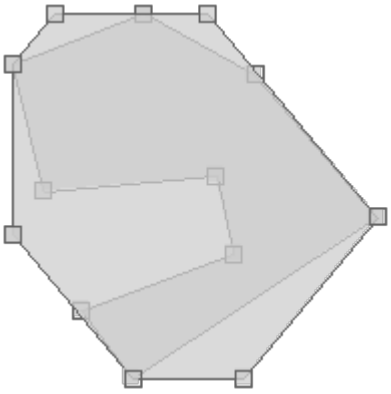
```
Polygon polygon1 = new Polygon([[
        [-121.915, 47.390],
        [-122.640, 46.995],
        [-121.739, 46.308],
        [-121.168, 46.777],
        [-120.981, 47.316],
        [-121.409, 47.413],
        [-121.915, 47.390]
]])

Polygon polygon2 = new Polygon([[
        [-120.794, 46.664],
        [-121.541, 46.995],
        [-122.200, 46.536],
        [-121.937, 45.890],
        [-120.959, 46.096],
        [-120.794, 46.664]
]])

Polygon polygon3 = new Polygon([[
        [-120.541, 47.376],
        [-120.695, 47.047],
        [-119.794, 46.830],
        [-119.586, 47.331],
        [-120.102, 47.509],
        [-120.541, 47.376]
]])

boolean does1overlap2 = polygon1.overlaps(polygon2)
println does1overlap2

boolean does1overlap3 = polygon1.overlaps(polygon3)
println does1overlap3

boolean does2overlap3 = polygon2.overlaps(polygon3)
println does2overlap3
```



```
true
false
false
```

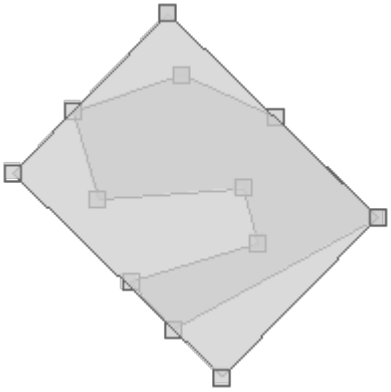
Calculate the octagonal envelope of a Geometry

```
Geometry geometry = new Polygon ([[
    [-122.20367431640624, 47.543163654317304],
    [-122.3712158203125,  47.489368981370724],
    [-122.33276367187499, 47.35371061951363],
    [-122.11029052734374, 47.3704545156932],
    [-122.08831787109375, 47.286681888764214],
    [-122.28332519531249, 47.2270293988673],
    [-122.2174072265625,  47.154237057576594],
    [-121.904296875,      47.32579231609051],
    [-122.06085205078125, 47.47823216312885],
    [-122.20367431640624, 47.543163654317304]
]])
Geometry octagonalEnvelope = geometry.octagonalEnvelope
```



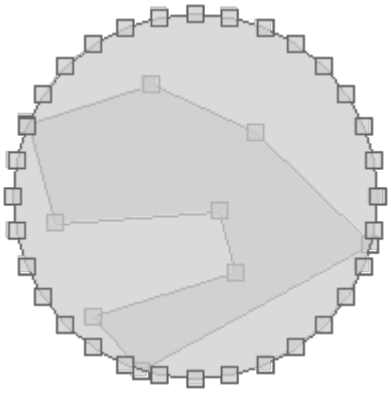
Calculate the minimum rectangle of a Geometry

```
Geometry geometry = new Polygon ([[
  [-122.20367431640624, 47.543163654317304],
  [-122.3712158203125,  47.489368981370724],
  [-122.33276367187499, 47.35371061951363],
  [-122.11029052734374, 47.3704545156932],
  [-122.08831787109375, 47.286681888764214],
  [-122.28332519531249, 47.2270293988673],
  [-122.2174072265625,  47.154237057576594],
  [-121.904296875,      47.32579231609051],
  [-122.06085205078125, 47.47823216312885],
  [-122.20367431640624, 47.543163654317304]
]])
Geometry minimumRectangle = geometry.minimumRectangle
```



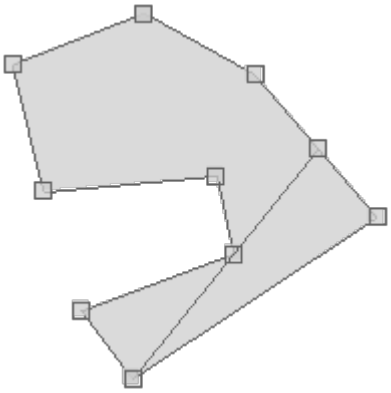
Calculate the minimum circle of a Geometry

```
Geometry geometry = new Polygon ([[
  [-122.20367431640624, 47.543163654317304],
  [-122.3712158203125,  47.489368981370724],
  [-122.33276367187499, 47.35371061951363],
  [-122.11029052734374, 47.3704545156932],
  [-122.08831787109375, 47.286681888764214],
  [-122.28332519531249, 47.2270293988673],
  [-122.2174072265625,  47.154237057576594],
  [-121.904296875,      47.32579231609051],
  [-122.06085205078125, 47.47823216312885],
  [-122.20367431640624, 47.543163654317304]
]])
Geometry minimumBoundingCircle = geometry.minimumBoundingCircle
```



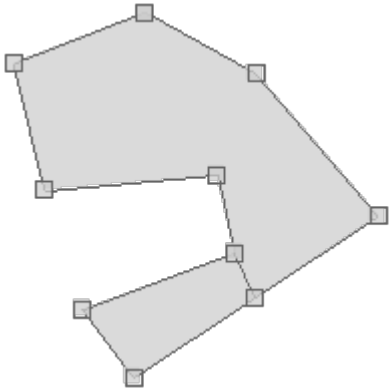
Calculate the minimum diameter of a Geometry

```
Geometry geometry = new Polygon ([[
    [-122.20367431640624, 47.543163654317304],
    [-122.3712158203125,  47.489368981370724],
    [-122.33276367187499, 47.35371061951363],
    [-122.11029052734374, 47.3704545156932],
    [-122.08831787109375, 47.286681888764214],
    [-122.28332519531249, 47.2270293988673],
    [-122.2174072265625,  47.154237057576594],
    [-121.904296875,      47.32579231609051],
    [-122.06085205078125, 47.47823216312885],
    [-122.20367431640624, 47.543163654317304]
]])
Geometry minimumDiameter = geometry.minimumDiameter
```



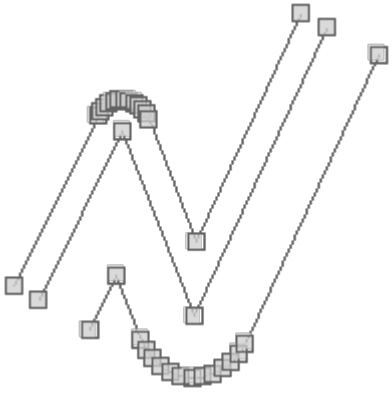
Calculate the minimum clearance of a Geometry

```
Geometry geometry = new Polygon ([[
    [-122.20367431640624, 47.543163654317304],
    [-122.3712158203125,  47.489368981370724],
    [-122.33276367187499, 47.35371061951363],
    [-122.11029052734374, 47.3704545156932],
    [-122.08831787109375, 47.286681888764214],
    [-122.28332519531249, 47.2270293988673],
    [-122.2174072265625,  47.154237057576594],
    [-121.904296875,      47.32579231609051],
    [-122.06085205078125, 47.47823216312885],
    [-122.20367431640624, 47.543163654317304]
]])
Geometry minimumClearance = geometry.minimumClearance
```



Offset a LineString by a given distance. Positive distances will offset to the right. Negative distance will offset to the left.

```
LineString line = new LineString(
        [3.198, 43.164],
        [6.713, 49.755],
        [9.702, 42.592],
        [15.32, 53.798]
)
LineString positive = line.offset(1.2)
LineString negative = line.offset(-2.4)
```



Get the dimension of a Geometry

```
Point point = Geometry.fromWKT("POINT (-122.3437 47.7540)")
println "Point Dimension = ${point.dimension}"

LineString lineString = Geometry.fromWKT("LINESTRING (-122.525 47.256, -122.376
47.595)")
println "LineString Dimension = ${lineString.dimension}"

Polygon polygon = Geometry.fromWKT("POLYGON ((-122.590 47.204, -122.365 47.204,
-122.365 47.312, -122.590 47.312, -122.590 47.204))")
println "Polygon Dimension = ${polygon.dimension}"
```

```
Point Dimension = 0
LineString Dimension = 1
Polygon Dimension = 2
```

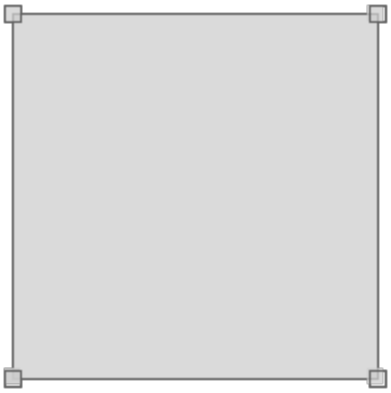
Determine if a Geometry is empty or not

```
Geometry geom1 = Geometry.fromWKT("POINT EMPTY")
boolean isGeom1Empty = geom1.empty
println "Is ${geom1.wkt} empty? ${isGeom1Empty ? 'Yes' : 'No'}"

Geometry geom2 = Geometry.fromWKT("POINT (-122.3437 47.7540)")
boolean isGeom2Empty = geom2.empty
println "Is ${geom2.wkt} empty? ${isGeom2Empty ? 'Yes' : 'No'}"
```

```
Is POINT EMPTY empty? Yes
Is POINT (-122.3437 47.754) empty? No
```

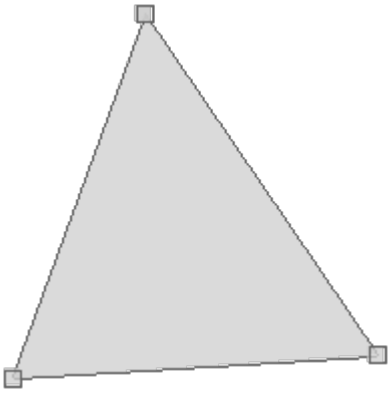
Determine if a Geometry is rectanglular

```
Geometry geom1 = Geometry.fromWKT("POLYGON ((-122.590 47.204, -122.365 47.204,
-122.365 47.312, -122.590 47.312, -122.590 47.204))")
boolean isGeom1Rect = geom1.isRectangle()
println "Is the geometry a rectangle? ${isGeom1Rect ? 'Yes' : 'No'}"
```



```
Is the geometry a rectangle? Yes
```

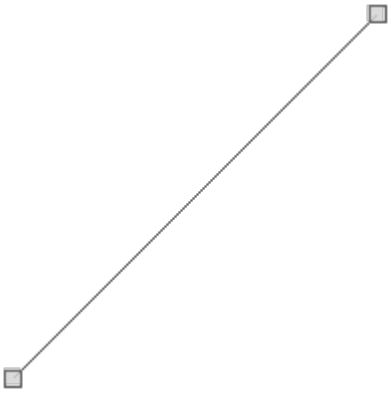
```
Geometry geom2 = Geometry.fromWKT("POLYGON ((-122.360 47.215, -122.656 46.912,
-121.838 46.931, -122.360 47.215))")
boolean isGeom2Rect = geom2.isRectangle()
println "Is the geometry a rectangle? ${isGeom2Rect ? 'Yes' : 'No'}"
```



```
Is the geometry a rectangle? No
```

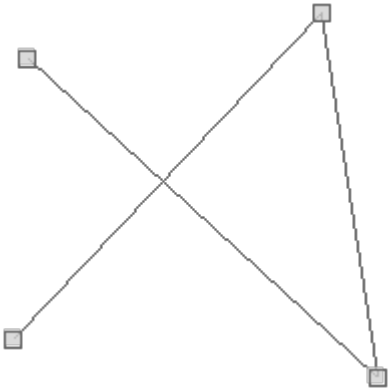
Determine if a Geometry is simple

```
Geometry geom1 = new LineString(
        [-122.323, 47.599],
        [-122.385, 47.581]
)
boolean isGeom1Simple = geom1.simple
println "Is the Geometry simple? ${isGeom1Simple}"
```



```
Is the Geometry simple? true
```

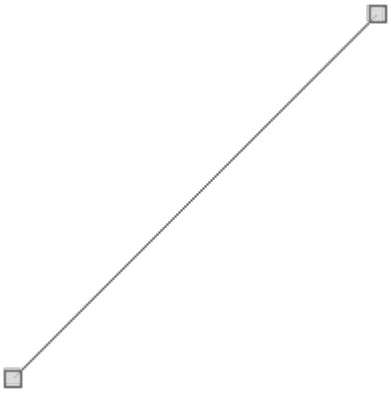
```
Geometry geom2 = new LineString(
        [-122.356, 47.537],
        [-122.295, 47.580],
        [-122.284, 47.532],
        [-122.353, 47.574]
)
boolean isGeom2Simple = geom2.simple
println "Is the Geometry simple? ${isGeom2Simple}"
```



```
Is the Geometry simple? false
```

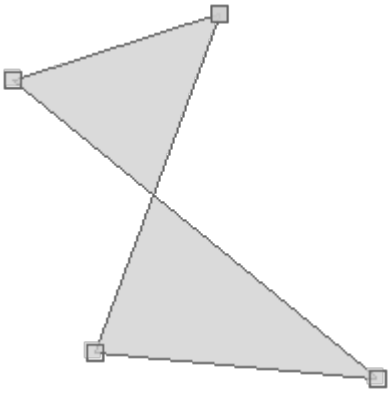
Determine if a Geometry is valid

```
Geometry geom1 = new LineString(
        [-122.323, 47.599],
        [-122.385, 47.581]
)
boolean isGeom1Valid = geom1.valid
println "Is the Geometry valid? ${isGeom1Valid}"
```



```
Is the Geometry valid? true
```

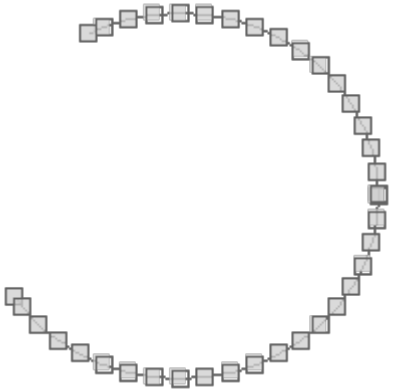
```
Geometry geom2 = new Polygon(new LinearRing([
    [48.16406, 42.29356],
    [35.15625, 25.79989],
    [64.33593, 24.52713],
    [26.71875, 39.09596],
    [48.16406, 42.29356],
]))
boolean isGeom2Valid = geom2.valid
println "Is the Geometry valid? ${isGeom2Valid}"
println geom2.validReason
```



```
Is the Geometry valid? false
Self-intersection
```

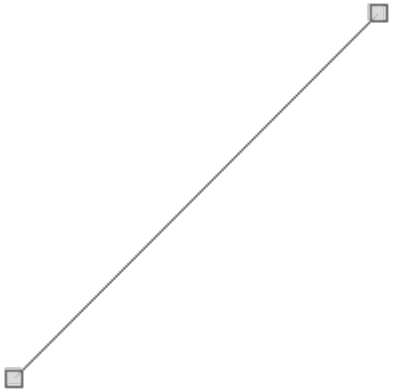
Determine if a Geometry is curved

```
Geometry geom1 = new CircularString([
        [-122.464599609375, 47.247542522268006],
        [-122.03613281249999, 47.37789454155521],
        [-122.37670898437499, 47.58393661978134]
])

boolean isGeom1Curved = geom1.curved
println "Is the Geometry valid? ${isGeom1Curved}"
```



```
Is the Geometry curved? true
```

```
Geometry geom2 = new LineString(
        [-122.323, 47.599],
        [-122.385, 47.581]
)
boolean isGeom2Curved = geom2.curved
println "Is the Geometry valid? ${isGeom2Curved}"
```



```
Is the Geometry curved? false
```

Determine if a Geometry is within a given distance of another Geometry

```
Geometry geom1 = new Point(-88.945, 41.771)
Geometry geom2 = new Point(-113.906, 37.160)

double distance1 = 26.0
boolean isWithin1 = geom1.isWithinDistance(geom2, distance1)
println "Is ${geom1} within ${distance1} of ${geom2}? ${isWithin1 ? 'Yes' : 'No'}"
```

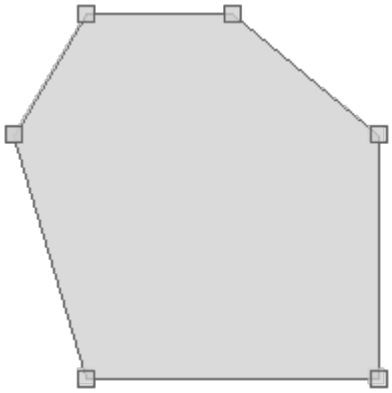
```
Is POINT (-88.945 41.771) within 26.0 of POINT (-113.906 37.16)? Yes
```

```
double distance2 = 15.5
boolean isWithin2 = geom1.isWithinDistance(geom2, distance2)
println "Is ${geom1} within ${distance2} of ${geom2}? ${isWithin2 ? 'Yes' : 'No'}"
```

```
Is POINT (-88.945 41.771) within 15.5 of POINT (-113.906 37.16)? No
```

Normalizing a Geometry changes the Geometry in place.

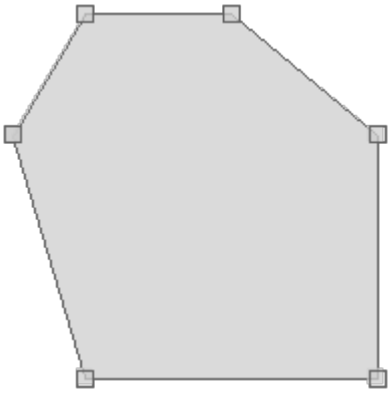
```
Geometry geometry = Geometry.fromWKT("POLYGON((2 4, 1 3, 2 1, 6 1, 6 3, 4 4, 2 4))")
geometry.normalize()
println "Normalized Geometry = ${geometry}"
```



```
Normalized Geometry = POLYGON ((1 3, 2 4, 4 4, 6 3, 6 1, 2 1, 1 3))
```

Calculating a normalized Geometry from a Geometry does not change the original Geometry.

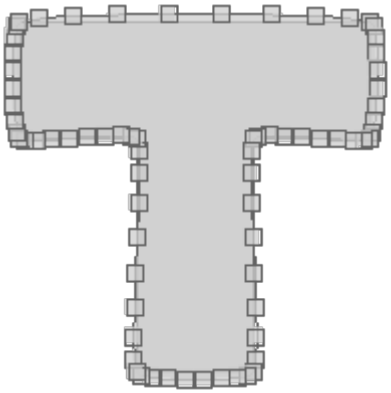
```
Geometry geometry = Geometry.fromWKT("POLYGON((2 4, 1 3, 2 1, 6 1, 6 3, 4 4, 2 4))")
Geometry normalizedGeometry = geometry.norm
println "Un-normalized Geometry = ${geometry}"
println "Normalized Geometry    = ${normalizedGeometry}"
```



```
Un-normalized Geometry = POLYGON ((2 4, 1 3, 2 1, 6 1, 6 3, 4 4, 2 4))
Normalized Geometry = POLYGON ((1 3, 2 4, 4 4, 6 3, 6 1, 2 1, 1 3))
```

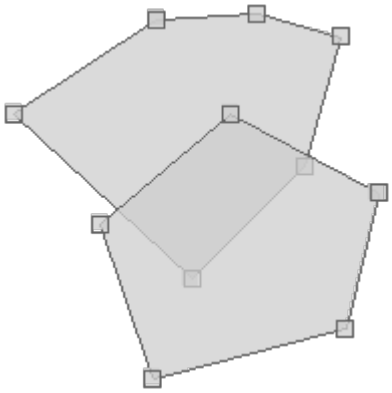
Smooth a Geometry

```
Geometry geometry = Geometry.fromWKT("POLYGON((10 0, 10 20, 0 20, 0 30, 30 30, 30 20,
20 20, 20 0, 10 0))")
Geometry smoothed = geometry.smooth(0.75)
```



Calculate the DE-9IM Intersection Matrix between two geometries.

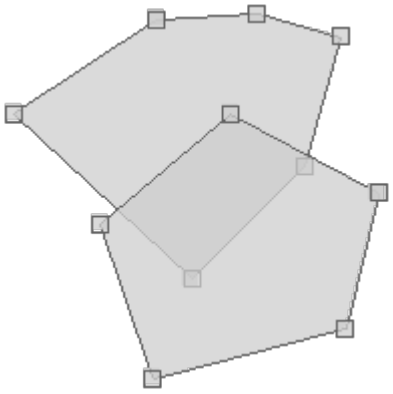
```
Polygon polygon1 = new Polygon([[
        [-121.915, 47.390],
        [-122.640, 46.995],
        [-121.739, 46.308],
        [-121.168, 46.777],
        [-120.981, 47.316],
        [-121.409, 47.413],
        [-121.915, 47.390]
]])

Polygon polygon2 = new Polygon([[
        [-120.794, 46.664],
        [-121.541, 46.995],
        [-122.200, 46.536],
        [-121.937, 45.890],
        [-120.959, 46.096],
        [-120.794, 46.664]
]])

IntersectionMatrix matrix = polygon1.relate(polygon2)
println "Intersection Matrix = ${matrix}"
println "Contains = ${matrix.contains}"
println "Covered By = ${matrix.coveredBy}"
println "Covers = ${matrix.covers}"
println "Disjoint = ${matrix.disjoint}"
println "Intersects = ${matrix.intersects}"
println "Within = ${matrix.within}"
```



```
Intersection Matrix = 212101212
Contains = false
Covered By = false
Covers = false
Disjoint = false
Intersects = true
Within = false
```

Determine if a Geometry relates to another Geometry according to the given DE-9IM Intersection Matrix string

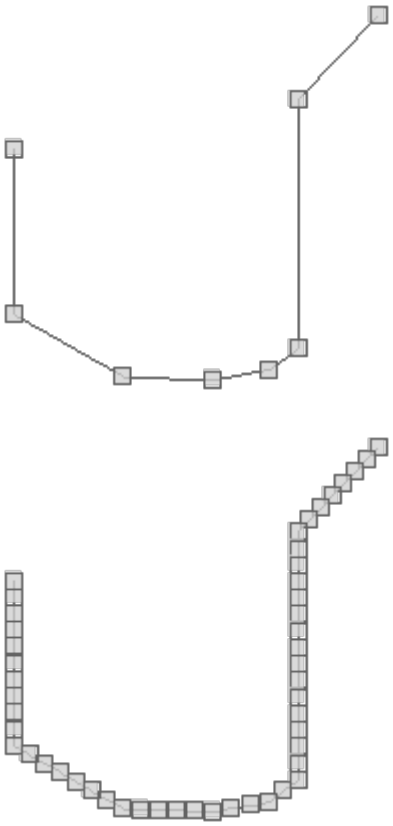
```
Polygon polygon1 = new Polygon([[
        [-121.915, 47.390],
        [-122.640, 46.995],
        [-121.739, 46.308],
        [-121.168, 46.777],
        [-120.981, 47.316],
        [-121.409, 47.413],
        [-121.915, 47.390]
]])

Polygon polygon2 = new Polygon([[
        [-120.794, 46.664],
        [-121.541, 46.995],
        [-122.200, 46.536],
        [-121.937, 45.890],
        [-120.959, 46.096],
        [-120.794, 46.664]
]])

println polygon1.relate(polygon2, "212101212")
println polygon1.relate(polygon2, "111111111")
println polygon1.relate(polygon2, "222222222")
```



```
true
false
false
```

Densify a Geometry by adding points at a given interval.

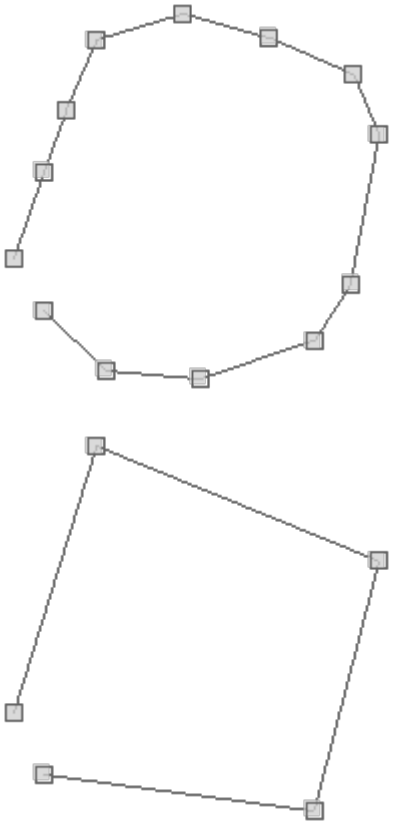
```
Geometry geometry = new LineString([
    [-122.28062152862547, 47.12986316579223],
    [-122.2809863090515, 47.12935221617075],
    [-122.2809863090515, 47.12786313499169],
    [-122.28111505508421, 47.127731743474406],
    [-122.28137254714966, 47.127673347140345],
    [-122.28178024291992, 47.12768794622986],
    [-122.28227376937865, 47.128067521151195],
    [-122.28227376937865, 47.12906024275466]
])
Geometry densified = geometry.densify(0.0001)
println "# of points in original geometry = ${geometry.numPoints}"
println "# of points in densified geometry = ${densified.numPoints}"
```



```
# of points in original geometry = 8
# of points in densified geometry = 50
```

Simplify a Geometry by removing points at a given interval.

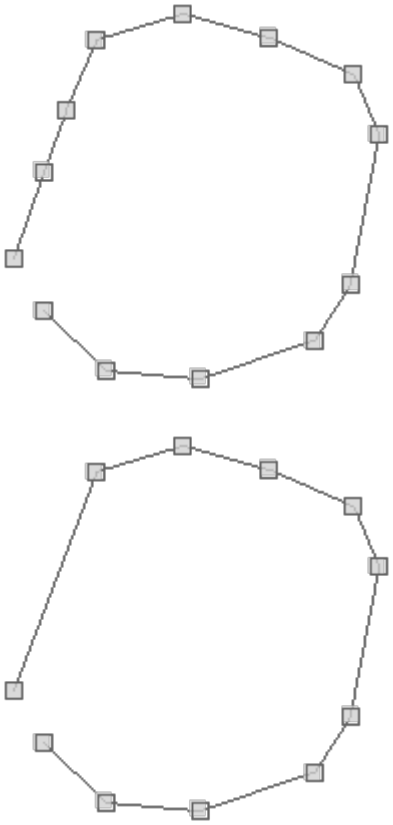
```
Geometry geometry = new LineString([
    [-123.59619140625001, 47.338822694822],
    [-123.04687499999999, 47.010225655683485],
    [-122.2119140625, 46.965259400349275],
    [-121.201171875, 47.17477833929903],
    [-120.87158203125, 47.487513008956554],
    [-120.62988281249999, 48.31242790407178],
    [-120.84960937499999, 48.647427805533546],
    [-121.59667968749999, 48.850258199721495],
    [-122.36572265625, 48.980216985374994],
    [-123.134765625, 48.83579746243093],
    [-123.3984375, 48.44377831058802],
    [-123.59619140625001, 48.10743118848039],
    [-123.85986328124999, 47.62097541515849]
])
Geometry simplified = geometry.simplify(0.5)
println "# of points in original geometry = ${geometry.numPoints}"
println "# of points in simplified geometry = ${simplified.numPoints}"
```



```
# of points in original geometry = 13
# of points in simplified geometry = 5
```

Simplify a Geometry by removing points at a given interval but preserve the topological structure.

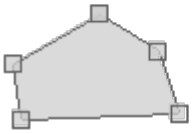
```
Geometry geometry = new LineString([
        [-123.59619140625001, 47.338822694822],
        [-123.04687499999999, 47.010225655683485],
        [-122.2119140625, 46.965259400349275],
        [-121.201171875, 47.17477833929903],
        [-120.87158203125, 47.487513008956554],
        [-120.62988281249999, 48.31242790407178],
        [-120.84960937499999, 48.647427805533546],
        [-121.59667968749999, 48.850258199721495],
        [-122.36572265625, 48.980216985374994],
        [-123.134765625, 48.83579746243093],
        [-123.3984375, 48.44377831058802],
        [-123.59619140625001, 48.10743118848039],
        [-123.85986328124999, 47.62097541515849]
])
Geometry simplified = geometry.simplifyPreservingTopology(0.1)
println "# of points in original geometry = ${geometry.numPoints}"
println "# of points in simplified geometry = ${simplified.numPoints}"
```



```
# of points in original geometry = 13
# of points in simplified geometry = 11
```

Translate or move a geometry a given distance along the x and y axis.

```
Geometry geometry = new Polygon(new LinearRing([
    [-121.83837890625, 47.5913464767971],
    [-122.76123046875, 46.9802523552188],
    [-122.67333984374, 46.3014061543733],
    [-121.00341796874, 46.3772542051002],
    [-121.22314453124, 47.1448974855539],
    [-121.83837890625, 47.5913464767971]
]))
Geometry translatedGeometry = geometry.translate(2.1, 3.2)
```

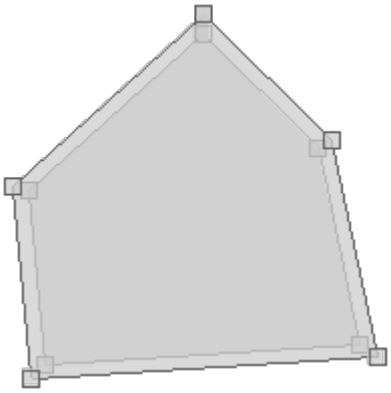


Scale a geometry a given amount in an x and y direction around the origin

```
Geometry geometry = new Polygon(new LinearRing([
        [-121.83837890625, 47.5913464767971],
        [-122.76123046875, 46.9802523552188],
        [-122.67333984374, 46.3014061543733],
        [-121.00341796874, 46.3772542051002],
        [-121.22314453124, 47.1448974855539],
        [-121.83837890625, 47.5913464767971]
]))
Geometry scaledGeometry = geometry.scale(1.1,1.2)
println scaledGeometry
```

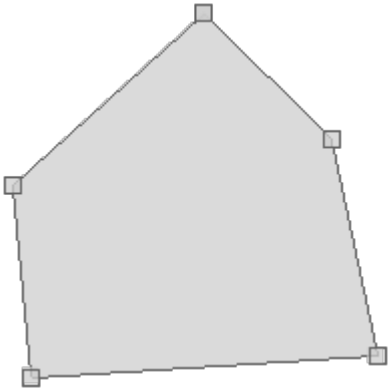
Scale a geometry a given amount in an x and y direction around a point

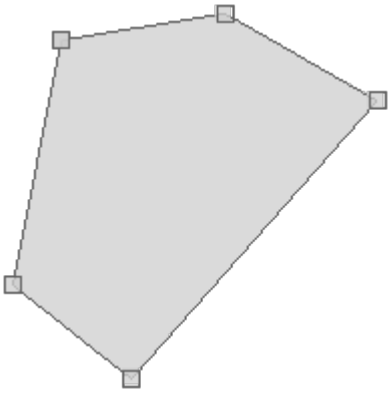
```
Geometry geometry = new Polygon(new LinearRing([
        [-121.83837890625, 47.5913464767971],
        [-122.76123046875, 46.9802523552188],
        [-122.67333984374, 46.3014061543733],
        [-121.00341796874, 46.3772542051002],
        [-121.22314453124, 47.1448974855539],
        [-121.83837890625, 47.5913464767971]
]))
Point centroid = geometry.centroid
Geometry scaledGeometry = geometry.scale(1.1, 1.1, centroid.x, centroid.y)
```



Rotate a Geometry around it's origin by a given angle theta (in radians).

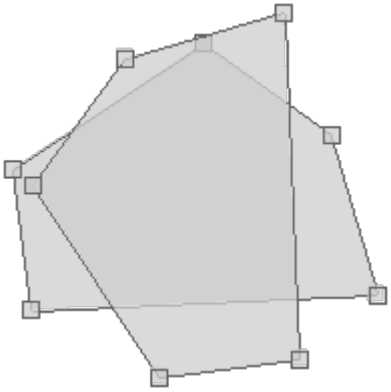
```
Geometry geometry = new Polygon(new LinearRing([
        [-121.83837890625, 47.5913464767971],
        [-122.76123046875, 46.9802523552188],
        [-122.67333984374, 46.3014061543733],
        [-121.00341796874, 46.3772542051002],
        [-121.22314453124, 47.1448974855539],
        [-121.83837890625, 47.5913464767971]
]))
Geometry theta = geometry.rotate(Math.toRadians(45))
```





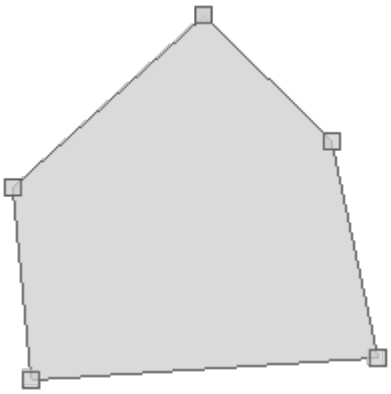
Rotate a Geometry around an XY coordinate by a given angle theta (in radians).

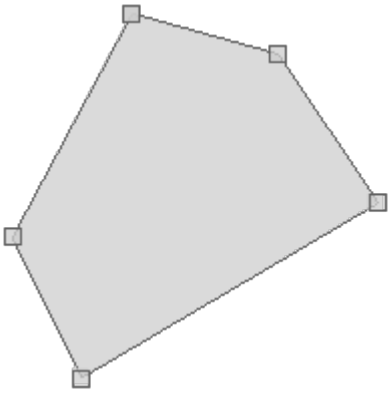
```
Geometry thetaXY = geometry.rotate(Math.toRadians(90), geometry.centroid.x, geometry
.centroid.y)
```



Rotate a Geometry around it's origin by a given angle sine and cosine (in radians).

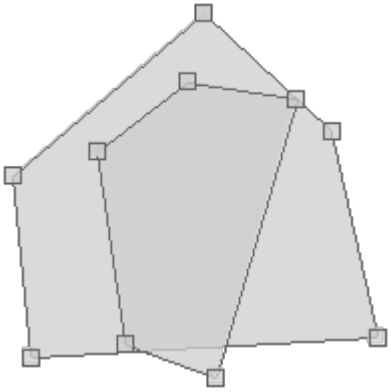
```
Geometry sinCos = geometry.rotate(Math.toRadians(15), Math.toRadians(35))
```





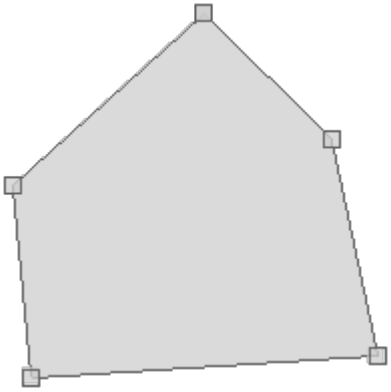
Rotate a Geometry around an XY coordinate by a given angle sine and cosine (in radians).

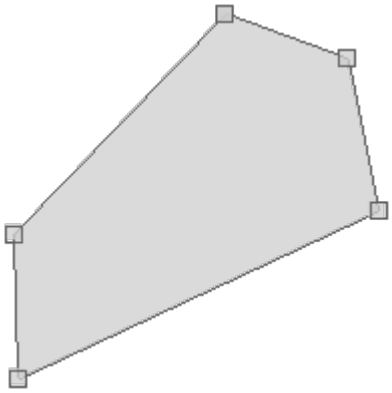
```
Geometry sinCosXY = geometry.rotate(Math.toRadians(15), Math.toRadians(35), geometry
.centroid.x, geometry.centroid.y)
```



Shear a Geometry around it's origin by a given distance along the x and y axis.

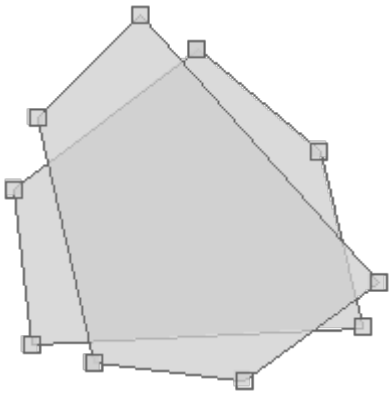
```
Geometry geometry = new Polygon(new LinearRing([
        [-121.83837890625, 47.5913464767971],
        [-122.76123046875, 46.9802523552188],
        [-122.67333984374, 46.3014061543733],
        [-121.00341796874, 46.3772542051002],
        [-121.22314453124, 47.1448974855539],
        [-121.83837890625, 47.5913464767971]
]))
Geometry shearedGeometry = geometry.shear(0.1,0.4)
```





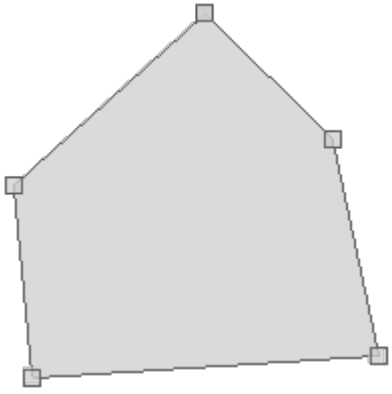
Reflect a Geometry around an XY coordinate for given distance along the x and y axis

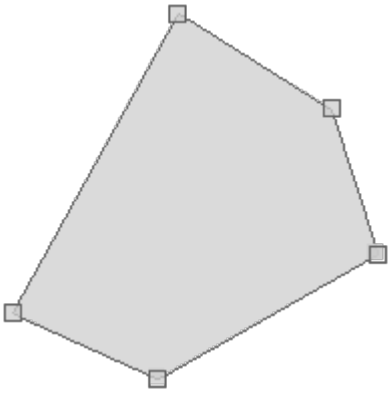
```
Geometry geometry = new Polygon(new LinearRing([
        [-121.83837890625, 47.5913464767971],
        [-122.76123046875, 46.9802523552188],
        [-122.67333984374, 46.3014061543733],
        [-121.00341796874, 46.3772542051002],
        [-121.22314453124, 47.1448974855539],
        [-121.83837890625, 47.5913464767971]
]))
Point centroid = geometry.centroid
Geometry reflectedGeometry = geometry.reflect(0.1,0.4, centroid.x, centroid.y)
```



Reflect a Geometry around the origin for given distance along the x and y axis

```
Geometry reflectedAroundOrigin = geometry.reflect(0.5, 0.34)
```





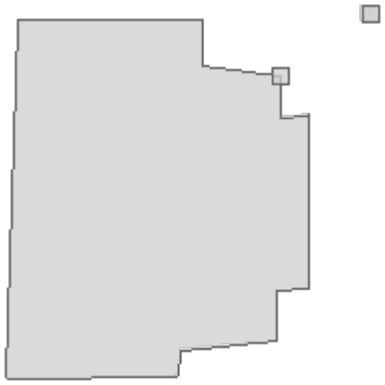
Reduce the precision of a Geometry's coordinates.

```
Geometry g1 = new Point(5.19775390625, 51.07421875)
println "Original Geometry: ${g1.wkt}"

Geometry g2 = g1.reducePrecision()
println "Floating Point Geometry: ${g2.wkt}"

Geometry g3 = g1.reducePrecision("fixed", scale: 100)
println "Fixed Point Geometry: ${g3.wkt}"

Geometry g4 = g1.reducePrecision("floating_single", pointwise: true, removecollapsed:
true)
println "Floating Point Single Geometry: ${g4.wkt}"
```

```
Original Geometry: POINT (5.19775390625 51.07421875)
Floating Point Geometry: POINT (5.19775390625 51.07421875)
Fixed Point Geometry: POINT (5.2 51.07)
Floating Point Single Geometry: POINT (5.19775390625 51.07421875)
```

Find the nearest points in one Geometry to another

```
Geometry point = new Point( -122.3845276236534, 47.58285653105873)
Geometry polygon = Geometry.fromWKT("POLYGON ((-122.3848307132721 47.58285110342092, "
+
        "-122.38484144210814 47.58255620092149, -122.38469392061235
47.582558010144346, " +
        "-122.3846912384033 47.5825797208137, -122.38460808992384 47.58258695770149, "
+
        "-122.38460808992384 47.582628569786834, -122.38458126783371
47.58263037900717, " +
        "-122.38458126783371 47.58277330721735, -122.38460540771483 47.58277149800195,
" +
        "-122.38460540771483 47.582805873084084, -122.38467246294022 47.5828131099406,
" +
        "-122.38467246294022 47.58285110342092, -122.3848307132721
47.58285110342092))")
List<Point> nearestPoints = polygon.getNearestPoints(point)
```



Convert a Geometry to a PreparedGeometry for more effecient spatial queries.

```
Geometry geometry = new Polygon([[
     [-121.915, 47.390],
     [-122.640, 46.995],
     [-121.739, 46.308],
     [-121.168, 46.777],
     [-120.981, 47.316],
     [-121.409, 47.413],
     [-121.915, 47.390]
]])
PreparedGeometry preparedGeometry = geometry.prepare()

Closure timer = { Closure action ->
    long start = System.nanoTime()
    action.call()
    long end = System.nanoTime()
    end - start
}

MultiPoint points = Geometry.createRandomPoints(new Bounds(-180, -90, 180, 90
).geometry, 100000)


long timeWithGeometry = timer({ ->
    points.geometries.each { Point point ->
        geometry.contains(point)
    }
})
println "Time with Geometry         = ${timeWithGeometry} nanoseconds"

long timeWithPreparedGeometry = timer({ ->
    points.geometries.each { Point point ->
        preparedGeometry.contains(point)
    }
})

println "Time with PreparedGeometry = ${timeWithPreparedGeometry} nanoseconds"
```

```
Time with Geometry         = 362587089 nanoseconds
Time with PreparedGeometry = 357857993 nanoseconds
```

Convert a Geometry to a PreparedGeometry using a static method for more effecient spatial queries.

```
Geometry geometry = new Polygon([[
     [-121.915, 47.390],
     [-122.640, 46.995],
     [-121.739, 46.308],
     [-121.168, 46.777],
     [-120.981, 47.316],
     [-121.409, 47.413],
     [-121.915, 47.390]
]])
PreparedGeometry preparedGeometry = Geometry.prepare(geometry)

Closure timer = { Closure action ->
    long start = System.nanoTime()
    action.call()
    long end = System.nanoTime()
    end - start
}

MultiPoint points = Geometry.createRandomPoints(new Bounds(-180, -90, 180, 90
).geometry, 100000)


long timeWithGeometry = timer({ ->
    points.geometries.each { Point point ->
        geometry.contains(point)
    }
})
println "Time with Geometry         = ${timeWithGeometry} nanoseconds"

long timeWithPreparedGeometry = timer({ ->
    points.geometries.each { Point point ->
        preparedGeometry.contains(point)
    }
})

println "Time with PreparedGeometry = ${timeWithPreparedGeometry} nanoseconds"
```

```
Time with Geometry         = 185469923 nanoseconds
Time with PreparedGeometry = 170308489 nanoseconds
```

Reading and Writing Geometries

The **geoscript.geom.io** package has several Readers and Writers for converting geoscript.geom.Geometry to and from strings.

Readers and Writers

Find all Geometry Readers

```
List<Reader> readers = Readers.list()
readers.each { Reader reader ->
    println reader.class.simpleName
}
```

```
GeobufReader
GeoJSONReader
GeoRSSReader
Gml2Reader
Gml3Reader
GpxReader
KmlReader
WkbReader
WktReader
GeoPackageReader
GooglePolylineEncoder
```

Find a Geometry Reader

```
String wkt = "POINT (-123.15 46.237)"
Reader reader = Readers.find("wkt")
Geometry geometry = reader.read(wkt)
```

Find all Geometry Writers

```
List<Writer> writers = Writers.list()
writers.each { Writer writer ->
    println writer.class.simpleName
}
```

```
GeobufWriter
GeoJSONWriter
GeoRSSWriter
Gml2Writer
Gml3Writer
GpxWriter
KmlWriter
WkbWriter
WktWriter
GeoPackageWriter
GooglePolylineEncoder
```

Find a Geometry Writer

```
Geometry geometry = new Point(-122.45, 43.21)
Writer writer = Writers.find("geojson")
String geojson = writer.write(geometry)
println geojson
```

```
{"type":"Point","coordinates":[-122.45,43.21]}
```

Create a Geometry from a String. The string will be parse by each Geometry Reader.

```
Geometry geom1 = Geometry.fromString('POINT (-123.15 46.237)')
println geom1

Geometry geom2 = Geometry.fromString
('{"type":"LineString","coordinates":[[3.198,43.164],[6.713,49.755],[9.702,42.592],[15
.32,53.798]]}')
println geom2


Geometry geom3 = Geometry.fromString('<Point><coordinates>-
123.15,46.237</coordinates></Point>')
println geom3
```

```
POINT (-123.15 46.237)
LINESTRING (3.198 43.164, 6.713 49.755, 9.702 42.592, 15.32 53.798)
POINT (-123.15 46.237)
```

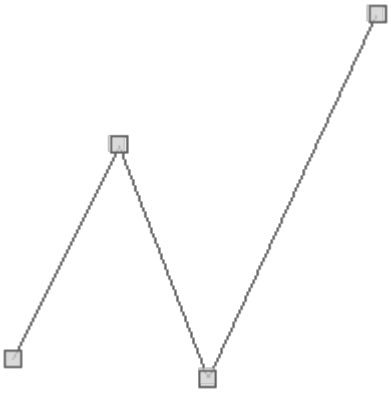
WKB

Read a Geometry from WKB using the WkbReader

```
String wkb = "0000000001C05EC9999999999A40471E5604189375"
WkbReader reader = new WkbReader()
Geometry geometry = reader.read(wkb)
```

Read a Geometry from WKB using the Geometry.fromWKB() static method

```
String wkb =
"0000000002000000044009958106 24DD2F404594FDF3B645A2401ADA1CAC0831274048E0A3D70A3D71402
3676C8B43958140454BC6A7EF9DB2402EA3D70A3D70A4404AE624DD2F1AA0"
Geometry geometry = Geometry.fromWKB(wkb)
```



Get the WKB of a Geometry

```
Geometry geometry = new Point(-123.15, 46.237)
String wkb = geometry.wkb
println wkb
```

```
0000000001C05EC9999999999A40471E5604189375
```

Write a Geometry to WKB using the WkbWriter

```
Geometry geometry = new LineString(
        [3.198, 43.164],
        [6.713, 49.755],
        [9.702, 42.592],
        [15.32, 53.798]
)
WkbWriter writer = new WkbWriter()
String wkb = writer.write(geometry)
println wkb
```

```
000000000200000004400995810624DD2F404594FDF3B645A2401ADA1CAC0831274048E0A3D70A3D714023
676C8B43958140454BC6A7EF9DB2402EA3D70A3D70A4404AE624DD2F1AA0
```

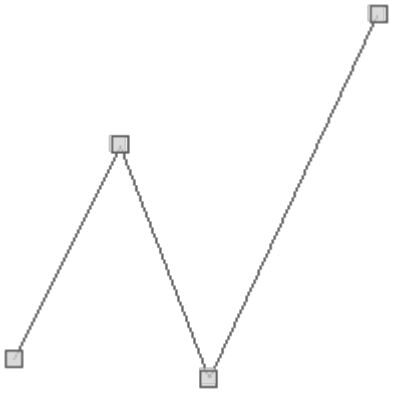
WKT

Read a Geometry from WKT using the WktReader

```
String wkt = "POINT (-123.15 46.237)"
WktReader reader = new WktReader()
Geometry geometry = reader.read(wkt)
```

Read a Geometry from WKT using the Geometry.fromWKT() static method

```
String wkt = "LINESTRING (3.198 43.164, 6.7138 49.755, 9.702 42.592, 15.327 53.798)"
Geometry geometry = Geometry.fromWKT(wkt)
```



Get the WKT of a Geometry

```
Geometry geometry = new Point(-123.15, 46.237)
String wkt = geometry.wkt
println wkt
```

```
POINT (-123.15 46.237)
```

Write a Geometry to WKT using the WktWriter

```
Geometry geometry = new LineString(
        [3.198, 43.164],
        [6.713, 49.755],
        [9.702, 42.592],
        [15.32, 53.798]
)
WktWriter writer = new WktWriter()
String wkt = writer.write(geometry)
println wkt
```

```
LINESTRING (3.198 43.164, 6.713 49.755, 9.702 42.592, 15.32 53.798)
```

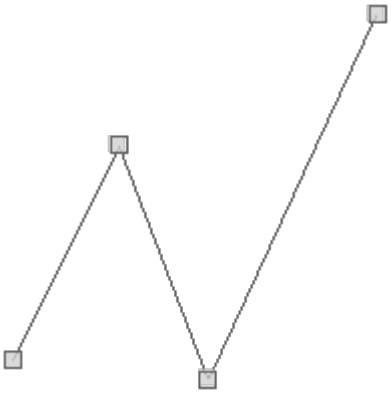
GeoJSON

Read a Geometry from GeoJSON using the GeoJSONReader

```
String json = '{"type":"Point","coordinates":[-123.15,46.237]}'
GeoJSONReader reader = new GeoJSONReader()
Geometry geometry = reader.read(json)
```

Read a Geometry from GeoJSON using the Geometry.fromGeoJSON() static method

```
String json =
'{"type":"LineString","coordinates":[[3.198,43.164],[6.713,49.755],[9.702,42.592],[15.
32,53.798]]}'
Geometry geometry = Geometry.fromGeoJSON(json)
```



Get the GeoJSON of a Geometry

```
Geometry geometry = new Point(-123.15, 46.237)
String json = geometry.geoJSON
println json
```

```
{"type":"Point","coordinates":[-123.15,46.237]}
```

Write a Geometry to GeoJSON using the GeoJSONWriter

```
Geometry geometry = new LineString(
        [3.198, 43.164],
        [6.713, 49.755],
        [9.702, 42.592],
        [15.32, 53.798]
)
GeoJSONWriter writer = new GeoJSONWriter()
String json = writer.write(geometry)
println json
```

```
{"type":"LineString","coordinates":[[3.198,43.164],[6.713,49.755],[9.702,42.592],[15.32,53.798]]}
```

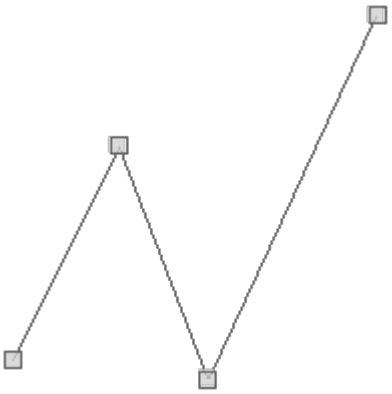
KML

Read a Geometry from KML using the KmlReader

```
String kml = "<Point><coordinates>-123.15,46.237</coordinates></Point>"
KmlReader reader = new KmlReader()
Geometry geometry = reader.read(kml)
```

Read a Geometry from KML using the Geometry.fromKml() static method

```
String kml = "<LineString><coordinates>3.198,43.164 6.713,49.755 9.702,42.592
15.32,53.798</coordinates></LineString>"
Geometry geometry = Geometry.fromKml(kml)
```



Get the KML of a Geometry

```
Geometry geometry = new Point(-123.15, 46.237)
String kml = geometry.kml
println kml
```

```
<Point><coordinates>-123.15,46.237</coordinates></Point>
```

Write a Geometry to KML using the KmlWriter

```
Geometry geometry = new LineString(
        [3.198, 43.164],
        [6.713, 49.755],
        [9.702, 42.592],
        [15.32, 53.798]
)
KmlWriter writer = new KmlWriter()
String kml = writer.write(geometry)
println kml
```

```
<LineString><coordinates>3.198,43.164 6.713,49.755 9.702,42.592
15.32,53.798</coordinates></LineString>
```

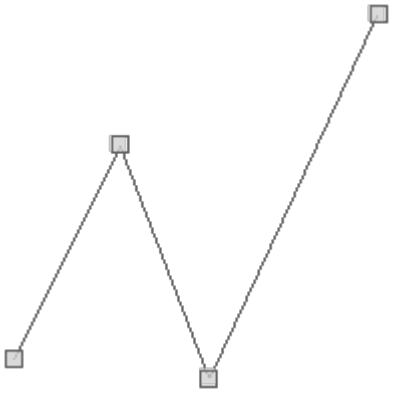
Geobuf

Read a Geometry from Geobuf using the GeobufReader

```
String geobuf = "10021806320c08001a08dffab87590958c2c"
GeobufReader reader = new GeobufReader()
Geometry geometry = reader.read(geobuf)
```

Read a Geometry from Geobuf using the Geometry.fromGeobuf() static method

```
String geobuf =
"10021806322408021a20e0b08603c0859529f089ad03b0c8a40690efec02efb1ea06a0e5ad05e0f5d70a"
Geometry geometry = Geometry.fromGeobuf(geobuf)
```



Get the Geobuf of a Geometry

```
Geometry geometry = new Point(-123.15, 46.237)
String geobuf = geometry.geobuf
println geobuf
```

```
10021806320c08001a08dffab87590958c2c
```

Write a Geometry to Geobuf using the GeobufWriter

```
Geometry geometry = new LineString(
        [3.198, 43.164],
        [6.713, 49.755],
        [9.702, 42.592],
        [15.32, 53.798]
)
GeobufWriter writer = new GeobufWriter()
String geobuf = writer.write(geometry)
println geobuf
```

```
10021806322408021a20e0b08603c0859529f089ad03b0c8a40690efec02efb1ea06a0e5ad05e0f5d70a
```

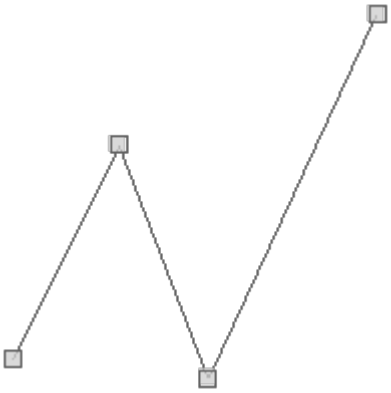
GML 2

Read a Geometry from GML2 using the Gml2Reader

```
String gml2 = "<gml:Point><gml:coordinates>-
123.15,46.237</gml:coordinates></gml:Point>"
Gml2Reader reader = new Gml2Reader()
Geometry geometry = reader.read(gml2)
```

Read a Geometry from GML2 using the Geometry.fromGML2() static method

```
String gml2 = "<gml:LineString><gml:coordinates>3.198,43.164 6.713,49.755 9.702,42.592
15.32,53.798</gml:coordinates></gml:LineString>"
Geometry geometry = Geometry.fromGML2(gml2)
```



Get the GML2 of a Geometry

```
Geometry geometry = new Point(-123.15, 46.237)
String gml2 = geometry.gml2
println gml2
```

```
<gml:Point><gml:coordinates>-123.15,46.237</gml:coordinates></gml:Point>
```

Write a Geometry to GML2 using the Gml2Writer

```
Geometry geometry = new LineString(
        [3.198, 43.164],
        [6.713, 49.755],
        [9.702, 42.592],
        [15.32, 53.798]
)
Gml2Writer writer = new Gml2Writer()
String gml2 = writer.write(geometry)
println gml2
```

```
<gml:LineString><gml:coordinates>3.198,43.164 6.713,49.755 9.702,42.592
15.32,53.798</gml:coordinates></gml:LineString>
```

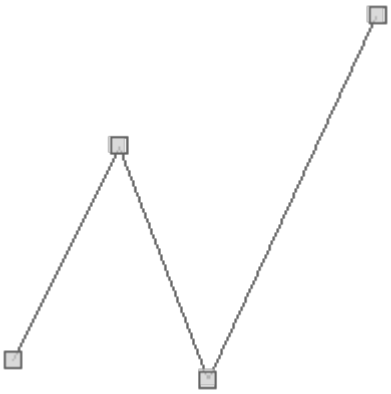
GML 3

Read a Geometry from GML3 using the Gml3Reader

```
String gml3 = "<gml:Point><gml:pos>-123.15 46.237</gml:pos></gml:Point>"
Gml3Reader reader = new Gml3Reader()
Geometry geometry = reader.read(gml3)
```

Read a Geometry from GML3 using the Geometry.fromGML3() static method

```
String gml3 = "<gml:LineString><gml:posList>3.198 43.164 6.713 49.755 9.702 42.592
15.32 53.798</gml:posList></gml:LineString>"
Geometry geometry = Geometry.fromGML3(gml3)
```



Get the GML3 of a Geometry

```
Geometry geometry = new Point(-123.15, 46.237)
String gml3 = geometry.gml3
println gml3
```

```
<gml:Point><gml:pos>-123.15 46.237</gml:pos></gml:Point>
```

Write a Geometry to GML3 using the Gml3Writer

```
Geometry geometry = new LineString(
        [3.198, 43.164],
        [6.713, 49.755],
        [9.702, 42.592],
        [15.32, 53.798]
)
Gml3Writer writer = new Gml3Writer()
String gml3 = writer.write(geometry)
println gml3
```

```
<gml:LineString><gml:posList>3.198 43.164 6.713 49.755 9.702 42.592 15.32
53.798</gml:posList></gml:LineString>
```

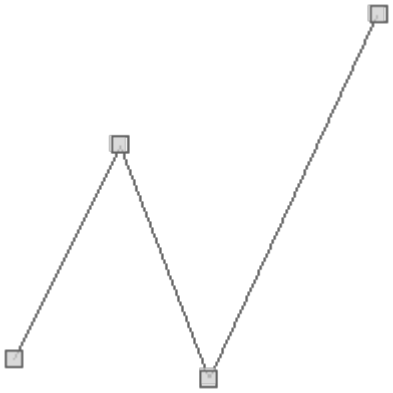
GPX

Read a Geometry from GPX using the GpxReader

```
String gpx = "<wpt lat='46.237' lon='-123.15'/>"
GpxReader reader = new GpxReader()
Geometry geometry = reader.read(gpx)
```

Read a Geometry from GPX using the Geometry.fromGPX() static method

```
String gpx = "<rte><rtept lat='43.164' lon='3.198' /><rtept lat='49.755' lon='6.713'
/><rtept lat='42.592' lon='9.702' /><rtept lat='53.798' lon='15.32' /></rte>"
Geometry geometry = Geometry.fromGpx(gpx)
```



Get the GPX of a Geometry

```
Geometry geometry = new Point(-123.15, 46.237)
String gpx = geometry.gpx
println gpx
```

```
<wpt lat='46.237' lon='-123.15'/>
```

Write a Geometry to GPX using the GpxWriter

```
Geometry geometry = new LineString(
        [3.198, 43.164],
        [6.713, 49.755],
        [9.702, 42.592],
        [15.32, 53.798]
)
GpxWriter writer = new GpxWriter()
String gpx = writer.write(geometry)
println gpx
```

```
<rte><rtept lat='43.164' lon='3.198' /><rtept lat='49.755' lon='6.713' /><rtept
lat='42.592' lon='9.702' /><rtept lat='53.798' lon='15.32' /></rte>
```

GeoRSS

Read a Geometry from GeoRSS using the GeoRSSReader

```
String georss = "<georss:point>46.237 -123.15</georss:point>"
GeoRSSReader reader = new GeoRSSReader()
Geometry geometry = reader.read(georss)
```

Write a Geometry to GeoRSS using the GeoRSSWriter

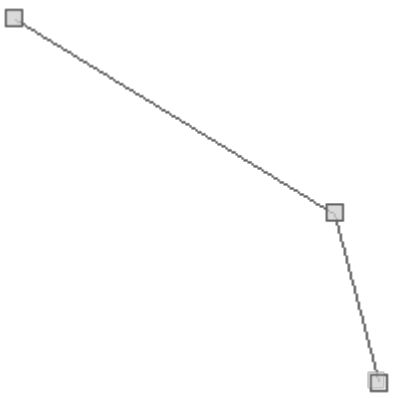
```
Geometry geometry = new LineString(
        [3.198, 43.164],
        [6.713, 49.755],
        [9.702, 42.592],
        [15.32, 53.798]
)
GeoRSSWriter writer = new GeoRSSWriter()
String georss = writer.write(geometry)
println georss
```

```
<georss:line>43.164 3.198 49.755 6.713 42.592 9.702 53.798 15.32</georss:line>
```

Google Polyline

Read a Geometry from a Google Polyline Encoded String using the GeoRSSReader

```
String str = "_p~iF~ps|U_ulLnnqC_mqNvxq`@"
GooglePolylineEncoder encoder = new GooglePolylineEncoder()
Geometry geometry = encoder.read(str)
```



Write a Geometry to a Google Polyline Encoded String using the GeoRSSWriter

```
Geometry geometry = new LineString(
        [3.198, 43.164],
        [6.713, 49.755],
        [9.702, 42.592],
        [15.32, 53.798]
)
GooglePolylineEncoder encoder = new GooglePolylineEncoder()
String str = encoder.write(geometry)
println str
```

```
_nmfGoroRwhfg@womTv_vj@gxfQotkcAogha@
```