# **Table of Contents**

G	eometry Recipes	1
	Creating Geometries	1
	Points.	9
	LineStrings.	. 11
	Polygons	. 24
	MultiLineStrings	. 30
	MultiPolygons	. 32
	Geometry Collections	. 35
	Circular Strings	
	Circular Rings	. 41
	Compound Curves	. 46
	Compound Rings	. 51
	Processing Geometries	. 57
	Prepared Geometry	114
	Reading and Writing Geometries	125

# **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

```
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 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**

```
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



### 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



## **CircularString**

Create a CircularString with a List of Points



## **CircularRing**

Create a CircularRing with a List of Points



## CompoundCurve

Create a CompoundCurve with a List of CircularStrings and LineStrings

```
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 = 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
```

```
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



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

Open LineString



### Closed LinearRing



Add a LineString to another LineString to create a MultiLineString





Add a Point to a LineString at a specific index



Set a Point on a LineString at a specific index



Remove a Point on a LineString at a specific index





### LineString lineString2 = -lineString1



### LineString lineString3 = -lineString2





Locate the position of Points on a LineString

```
LineString lineString = new LineString([
    [-122.39423990249632, 47.57926150237904],
    [-122.3918581008911, 47.58121554959838],
    [-122.38657951354979, 47.58121554959838],
    [-122.38638639450075, 47.58535499390333],
    [-122.38374710083008, 47.58535499390333]
1)
Point point1 = new Point(-122.39423990249632, 47.57926150237904)
Double position1 = lineString.locatePoint(point1)
println "Position of ${point1} is ${position1}"
Point point2 = new Point(-122.38736758304911, 47.58121554959838)
Double position2 = lineString.locatePoint(point2)
println "Position of ${point2} is ${position2}"
Point point3 = new Point(-122.38374710083008, 47.58535499390333)
Double position3 = lineString.locatePoint(point3)
println "Position of ${point3} is ${position3}"
```



```
Position of POINT (-122.39423990249632 47.57926150237904) is 0.0
Position of POINT (-122.38736758304911 47.58121554959838) is 0.50000000000004425
Position of POINT (-122.38374710083008 47.58535499390333) is 1.0
```

Place Points on a LineString

Points near LineString

-----

Point on LineString

#### Extract part of a LineString

```
LineString lineString = new LineString([
       [-122.39423990249632, 47.57926150237904],
       [-122.3918581008911, 47.58121554959838],
       [-122.38657951354979, 47.58121554959838],
       [-122.38638639450075, 47.58535499390333],
       [-122.38638639450075, 47.58535499390333]
])
LineString subLine = lineString.subLine(0.33, 0.66)
```





## **Polygons**

```
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()
```





```
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)

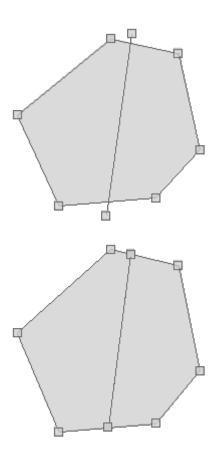






Split a Polygon with a LineString

```
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]
1)
MultiPolygon multiPolygon = polygon.split(lineString)
```



## MultiLineStrings

Add a LineString to a MultiLineString to get a new MultiLineString



Merge the LineStrings in a MultiLineString

```
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]
    )
])
MultiLineString mergedMultiLineString = multiLineString.merge()
println "Original MultiLineString = ${multiLineString}"
println "Merged MultiLineString = ${mergedMultiLineString}"
```

```
Original MultiLineString = MULTILINESTRING ((-122.3822021484375 47.57837853860192, -122.32452392578125 47.48380086737799), (-122.32452392578125 47.48380086737799, -122.29705810546874 47.303447043862626), (-122.29705810546874 47.303447043862626, -122.42889404296875 47.23262467463881))

Merged MultiLineString = MULTILINESTRING ((-122.3822021484375 47.57837853860192, -122.32452392578125 47.48380086737799, -122.29705810546874 47.303447043862626, -122.42889404296875 47.23262467463881))
```



Create Points along a MultiLineString



## MultiPolygons

```
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]
    ]])
)
Polygon polygon = new Polygon ([[
    [-122.32177734375, 47.54501765940571],
    [-122.27645874023438, 47.36673410912714],
    [-122.03887939453125, 47.44480754169437],
    [-122.15972900390624, 47.55150616084034],
    [-122.32177734375, 47.54501765940571]
]])
MultiPolygon newMultiPolygon = multiPolygon + polygon
```

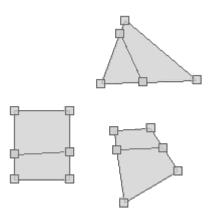


```
MultiPolygon multiPolygon = new MultiPolygon (
    new Polygon([[
             [-122.2723388671875, 47.818687628247105],
             [-122.37945556640624, 47.66168780332917],
             [-121.95373535156249, 47.67093619422418],
             [-122.2723388671875, 47.818687628247105]
    11),
    new Polygon ([[
          [-122.76672363281249, 47.42437092240516],
          [-122.76672363281249, 47.59505101193038],
          [-122.52227783203125, 47.59505101193038],
          [-122.52227783203125, 47.42437092240516],
          [-122.76672363281249, 47.42437092240516]
    ]]),
    new Polygon ([[
          [-122.32177734375, 47.54501765940571],
          [-122.27645874023438, 47.36673410912714],
          [-122.03887939453125, 47.44480754169437],
          [-122.15972900390624, 47.55150616084034],
          [-122.32177734375, 47.54501765940571]
    ]])
)
LineString lineString = new LineString([
    [-122.84362792968749, 47.484728927366504],
    [-122.05810546875, 47.50421439972969],
    [-122.35748291015625, 47.85832433461554]
])
Geometry splitGeometry = multiPolygon.split(lineString)
```

#### Before



After

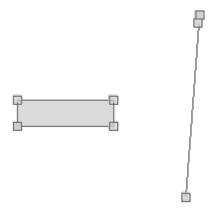


## **Geometry Collections**

Add a Geometry to a Geometry Collection

```
GeometryCollection geometryCollection = new GeometryCollection([
    new Point(-122.38654196262358, 47.581211931059826),
    new LineString([
        [-122.3865446448326, 47.58118841055313],
        [-122.38657146692276, 47.58067638459562]
    ]),
    new Polygon(new LinearRing([
        [-122.38693356513977, 47.58088445228483],
        [-122.38672703504562, 47.58088445228483],
        [-122.38672703504562, 47.58096225129535],
        [-122.38693356513977, 47.58096225129535],
        [-122.38693356513977, 47.58088445228483]
    ]))
1)
GeometryCollection newGeometryCollection = geometryCollection + new Point(-
122.38718032836913, 47.58121374032914)
```

Original Geometry Collection



New Geometry Collection



Get a subset of Geometries from a Geometry Collection



```
Polygon slicedGeometryCollection1 = geometryCollection.slice(2)
```



```
GeometryCollection slicedGeometryCollection2 = geometryCollection.slice(0,2)
```



Get a Geometry from a GeometryCollection that is the most type specific as possible.

```
GeometryCollection geometryCollection1 = new GeometryCollection([
    new Point(-122.38654196262358, 47.581211931059826),
    new Point(-122.38718032836913, 47.58121374032914),
])
Geometry narrowedGeometry1 = geometryCollection1.narrow()
println "Narrow Geometry #1 = ${narrowedGeometry1.wkt}"
```

Narrow Geometry #1 = MULTIPOINT ((-122.38654196262358 47.581211931059826), (-122.38718032836913 47.58121374032914))



```
Narrow Geometry #2 = GEOMETRYCOLLECTION (POINT (-122.38654196262358 47.581211931059826), POLYGON ((-122.38693356513977 47.58088445228483, -122.38672703504562 47.58088445228483, -122.38672703504562 47.58096225129535, -122.38693356513977 47.58088445228483)))
```

## **Circular Strings**

Get Curved WKT from a Circular Strings



```
WKT = LINESTRING (-122.464599609375 47.247542522268006, -122.4550237920096
47.23410579860605, -122.4348780833765 47.21113402059009, -122.41190630536055
47.19098831195699, -122.38650151145254 47.17401337136692, -122.3590983847198
47.16049964475972, -122.33016580025833 47.15067835574435, -122.30019880260986
47.144717549298825, -122.26971013541258 47.1427192164741, -122.2392214682153
47.144717549298825, -122.20925447056683 47.15067835574435, -122.18032188610536
47.16049964475972, -122.15291875937262 47.17401337136692, -122.12751396546462
47.19098831195699, -122.10454218744866 47.21113402059009, -122.08439647881556
47.23410579860605, -122.06742153822549 47.259510592514054, -122.05390781161829
47.28691371924678, -122.04408652260291 47.31584630370827, -122.0381257161574
47.34581330135674, -122.03612738333267 47.37630196855401, -122.03613281249999
47.37789454155521, -122.0381257161574 47.40679063575128, -122.04408652260291
47.43675763339975, -122.05390781161829 47.46569021786124, -122.06742153822549
47.493093344593966, -122.08439647881556 47.51849813850197, -122.10454218744866
47.54146991651793, -122.12751396546462 47.56161562515103, -122.15291875937262
47.5785905657411, -122.18032188610536 47.5921042923483, -122.20925447056683
47.60192558136367, -122.2392214682153 47.607886387809195, -122.26971013541258
47.60988472063392, -122.30019880260986 47.607886387809195, -122.33016580025833
47.60192558136367, -122.3590983847198 47.5921042923483, -122.37670898437499
47.58393661978134)
Curnved WKT = CIRCULARSTRING (-122.464599609375 47.247542522268006,
-122.03613281249999 47.37789454155521, -122.37670898437499 47.58393661978134)
```

### Get control points from a Circular Strings



```
POINT (-122.464599609375 47.247542522268006)
POINT (-122.03613281249999 47.37789454155521)
POINT (-122.37670898437499 47.58393661978134)
```

### Convert a Circural Strings to a Linear Geometry



```
LINESTRING (-122.464599609375 47.247542522268006, -122.4550237920096
47.23410579860605, -122.4348780833765 47.21113402059009, -122.41190630536055
47.19098831195699, -122.38650151145254 47.17401337136692, -122.3590983847198
47.16049964475972, -122.33016580025833 47.15067835574435, -122.30019880260986
47.144717549298825, -122.26971013541258 47.1427192164741, -122.2392214682153
47.144717549298825, -122.20925447056683 47.15067835574435, -122.18032188610536
47.16049964475972, -122.15291875937262 47.17401337136692, -122.12751396546462
47.19098831195699, -122.10454218744866 47.21113402059009, -122.08439647881556
47.23410579860605, -122.06742153822549 47.259510592514054, -122.05390781161829
47.28691371924678, -122.04408652260291 47.31584630370827, -122.0381257161574
47.34581330135674, -122.03612738333267 47.37630196855401, -122.03613281249999
47.37789454155521, -122.0381257161574 47.40679063575128, -122.04408652260291
47.43675763339975, -122.05390781161829 47.46569021786124, -122.06742153822549
47.493093344593966, -122.08439647881556 47.51849813850197, -122.10454218744866
47.54146991651793, -122.12751396546462 47.56161562515103, -122.15291875937262
47.5785905657411, -122.18032188610536 47.5921042923483, -122.20925447056683
47.60192558136367, -122.2392214682153 47.607886387809195, -122.26971013541258
47.60988472063392, -122.30019880260986 47.607886387809195, -122.33016580025833
47.60192558136367, -122.3590983847198 47.5921042923483, -122.37670898437499
47.58393661978134)
```

# **Circular Rings**

Get Curved WKT from a Circular Ring



```
WKT = LINEARRING (-118.47656249999999 41.508577297439324, -118.58869123015452
41.901774579059115, -118.91479847512254 43.54122641035517, -119.02412442253558
45.209218040904574, -118.91479847512252 46.877209671453976, -118.58869123015451
48.516661502750026, -118.05138247300694 50.09952205941734, -117.31206570548474
51.59870815847947, -116.38339084245754 52.98856831012856, -115.28124776830906
54.245321621827856, -114.02449445660976 55.347464695976335, -112.63463430496067
56.27613955900354, -111.13544820589854 57.01545632652574, -109.6875 57.51582286553883,
-109.55258764923123 57.552765083673314, -107.91313581793517 57.87887232864133,
-106.24514418738576 57.98819827605437, -104.57715255683635 57.87887232864133,
-102.9377007255403 57.552765083673314, -101.35484016887298 57.01545632652574,
-99.85565406981085 56.27613955900354, -98.46579391816176 55.347464695976335,
-97.20904060646247 54.245321621827856, -96.10689753231398 52.98856831012856,
-95.17822266928678 51.59870815847947, -94.43890590176458 50.09952205941733,
-93.901597144617 48.51666150275002, -93.575489899649 46.87720967145396,
-93.46616395223595 45.20921804090456, -93.575489899649 43.54122641035516, -93.8671875
42.032974332441405, -92.52944589377746 42.87620255408929, -90.13998424943787
44.0545546346726, -87.61715897053445 44.91093841996455, -85.00413629704853
45.43070094596436, -82.34562577139025 45.60494893174677, -79.68711524573196
45.43070094596437, -77.07409257224604 44.910938419964566, -74.55126729334262
44.05455463467261, -72.16180564900301 42.876202554089296, -69.94659199044581
41.396044109014326, -67.9435292375422 39.639405220820365, -66.18689034934823
37.63634246791675, -64.70673190427325 35.42112880935955, -63.528379823689946
33.03166716501995, -62.67199603839799 30.50884188611652, -62.57812500000001
30.14512718337613, -62.15223351239818 27.895819212630613, -61.97798552661578
25.237308686972327, -62.15223351239818 22.578798161314037, -62.671996038397985
19.965775487828132, -63.52837982368993 17.442950208924703, -64.70673190427325
15.0534885645851, -66.18689034934822 12.838274906027902, -67.94352923754218
10.835212153124278, -69.9465919904458 9.078573264930323, -72.16180564900299
7.598414819855346, -74.55126729334259 6.42006273927203, -77.07409257224602
5.563678953980077, -79.68711524573193 5.0439164279802675, -82.34562577139022
4.869668442197863, -85.00413629704852 5.043916427980264, -87.61715897053442
5.5636789539800695, -90.13998424943784 6.420062739272023, -92.10937499999999
7.36246686553575, -94.00832592648507 5.722586433241059, -96.3693387982898
4.1450080684486466, -98.91606817456314 2.88910012519991, -101.60493880959022
1.9763515366073037, -104.38994338130999 1.4223796840833636, -107.22342968935115
1.2366631796147907, -110.05691599739232 1.4223796840833813, -112.84192056911208
1.9763515366073356, -115.53079120413916 2.8891001251999597, -118.07752058041248
4.1450080684487105, -120.43853345221721 5.722586433241133, -122.57343223472037
7.594842416368293, -124.4456882178475 9.729741198871439, -126.02326658263992
12.09075407067618, -127.265625 14.604847155053898, -127.27917452588866
14.637483446949503, -128.19192311448128 17.326354081976593, -128.7458949670052
20.111358653696357, -128.9316114714738 22.944844961737523, -128.7458949670052
25.77833126977869, -128.19192311448126 28.56333584149845, -127.27917452588864
31.252206476525537, -126.02326658263989 33.798935852798856, -124.44568821784746
36.159948724603595, -122.57343223472031 38.29484750710673, -120.43853345221717
40.16710349023388, -118.47656249999999 41.508577297439324)
Curnved WKT = CIRCULARSTRING (-118.47656249999999 41.508577297439324, -109.6875
57.51582286553883, -93.8671875 42.032974332441405, -62.57812500000001
30.14512718337613, -92.1093749999999 7.36246686553575, -127.265625
14.604847155053898, -118.47656249999999 41.508577297439324)
```



```
POINT (-118.47656249999999 41.508577297439324)

POINT (-109.6875 57.51582286553883)

POINT (-93.8671875 42.032974332441405)

POINT (-62.57812500000001 30.14512718337613)

POINT (-92.10937499999999 7.36246686553575)

POINT (-127.265625 14.604847155053898)

POINT (-118.47656249999999 41.508577297439324)
```

#### Convert a Circural Ring to a Linear Geometry



```
LINEARRING (-118.47656249999999 41.508577297439324, -118.58869123015452
41.901774579059115, -118.91479847512254 43.54122641035517, -119.02412442253558
45.209218040904574, -118.91479847512252 46.877209671453976, -118.58869123015451
48.516661502750026, -118.05138247300694 50.09952205941734, -117.31206570548474
51.59870815847947, -116.38339084245754 52.98856831012856, -115.28124776830906
54.245321621827856, -114.02449445660976 55.347464695976335, -112.63463430496067
56.27613955900354, -111.13544820589854 57.01545632652574, -109.6875 57.51582286553883,
-109.55258764923123 57.552765083673314, -107.91313581793517 57.87887232864133,
-106.24514418738576 57.98819827605437, -104.57715255683635 57.87887232864133,
-102.9377007255403 57.552765083673314, -101.35484016887298 57.01545632652574,
-99.85565406981085 56.27613955900354, -98.46579391816176 55.347464695976335,
-97.20904060646247 54.245321621827856, -96.10689753231398 52.98856831012856,
-95.17822266928678 51.59870815847947, -94.43890590176458 50.09952205941733,
-93.901597144617 48.51666150275002, -93.575489899649 46.87720967145396,
-93.46616395223595 45.20921804090456, -93.575489899649 43.54122641035516, -93.8671875
42.032974332441405, -92.52944589377746 42.87620255408929, -90.13998424943787
44.0545546346726, -87.61715897053445 44.91093841996455, -85.00413629704853
45.43070094596436, -82.34562577139025 45.60494893174677, -79.68711524573196
45.43070094596437, -77.07409257224604 44.910938419964566, -74.55126729334262
44.05455463467261, -72.16180564900301 42.876202554089296, -69.94659199044581
41.396044109014326, -67.9435292375422 39.639405220820365, -66.18689034934823
37.63634246791675, -64.70673190427325 35.42112880935955, -63.528379823689946
33.03166716501995, -62.67199603839799 30.50884188611652, -62.57812500000001
30.14512718337613, -62.15223351239818 27.895819212630613, -61.97798552661578
25.237308686972327, -62.15223351239818 22.578798161314037, -62.671996038397985
19.965775487828132, -63.52837982368993 17.442950208924703, -64.70673190427325
15.0534885645851, -66.18689034934822 12.838274906027902, -67.94352923754218
10.835212153124278, -69.9465919904458 9.078573264930323, -72.16180564900299
7.598414819855346, -74.55126729334259 6.42006273927203, -77.07409257224602
5.563678953980077, -79.68711524573193 5.0439164279802675, -82.34562577139022
4.869668442197863, -85.00413629704852 5.043916427980264, -87.61715897053442
5.5636789539800695, -90.13998424943784 6.420062739272023, -92.10937499999999
7.36246686553575, -94.00832592648507 5.722586433241059, -96.3693387982898
4.1450080684486466, -98.91606817456314 2.88910012519991, -101.60493880959022
1.9763515366073037, -104.38994338130999 1.4223796840833636, -107.22342968935115
1.2366631796147907, -110.05691599739232 1.4223796840833813, -112.84192056911208
1.9763515366073356, -115.53079120413916 2.8891001251999597, -118.07752058041248
4.1450080684487105, -120.43853345221721 5.722586433241133, -122.57343223472037
7.594842416368293, -124.4456882178475 9.729741198871439, -126.02326658263992
12.09075407067618, -127.265625 14.604847155053898, -127.27917452588866
14.637483446949503, -128.19192311448128 17.326354081976593, -128.7458949670052
20.111358653696357, -128.9316114714738 22.944844961737523, -128.7458949670052
25.77833126977869, -128.19192311448126 28.56333584149845, -127.27917452588864
31.252206476525537, -126.02326658263989 33.798935852798856, -124.44568821784746
36.159948724603595, -122.57343223472031 38.29484750710673, -120.43853345221717
40.16710349023388, -118.47656249999999 41.508577297439324)
```

# **Compound Curves**

Get Curved WKT from a Compound Curves

```
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]
        ])
])
println "WKT = ${compoundCurve.wkt}"
println "Curnved WKT = ${compoundCurve.curvedWkt}"
```



```
WKT = LINESTRING (27.0703125 23.885837699862005, 27.022331449414295 23.8488525605847,
25.524767337284196 22.84821221194479, 23.909405319245263 22.051603821364203,
22.203884687289108 21.472657579267334, 20.43738737228815 21.12127941637069,
18.64013863306764 21.00348151046186, 16.842889893847133 21.1212794163707,
15.076392578846175 21.472657579267356, 13.370871946890022 22.051603821364232,
11.755509928851094 22.848212211944833, 10.257945816720998 23.848852560584746,
8.903803347661494 25.036403633488835, 7.716252274757412 26.390546102548342,
6.715611926117504 27.88811021467844, 5.919003535536911 29.50347223271737,
5.340057293440038 31.20899286467353, 4.988679130543391 32.97549017967449,
4.8708812246345605 34.772738918894994, 4.988679130543396 36.5699876581155,
5.340057293440047 38.336484973116455, 5.9190035355369215 40.04200560507262, 5.9765625
40.17887331434696, 6.715611926117516 41.65736762311154, 7.71625227475743
43.15493173524164, 8.903803347661514 44.50907420430114, 10.257945816721021
45.69662527720523, 11.755509928851117 46.69726562584515, 13.37087194689005
47.49387401642574, 15.076392578846209 48.07282025852261, 16.84288989384717
48.42419842141926, 18.64013863306768 48.5419963273281, 20.437387372288192
48.42419842141926, 22.20388468728915 48.07282025852261, 22.5 47.98992166741417,
71.71875 49.15296965617039, 72.58658076138724 48.953451060440116, 74.12646468490568
48.43073090444654, 75.58494601643167 47.71148750685245, 76.93706973601537
46.80802732160263, 78.15970063193521 45.735808802953194, 79.23191915058464
44.513177907033366, 80.13537933583446 43.16105418744966, 80.85462273342854
41.70257285592367, 81.37734288942212 40.162688932405224, 81.5625 39.36827914916011,
81.69459591702075 38.567750257762206, 81.80095352896302 36.9450466749183,
81.69459591702075 35.3223430920744, 81.37734288942212 33.72740441743138,
80.85462273342854 32.187520493912935, 80.13537933583446 30.729039162386947,
79.23191915058464 29.376915442803238, 78.15970063193521 28.15428454688341,
76.93706973601537 27.082066028233974, 75.58494601643167 26.178605842984155,
74.12646468490568 25.459362445390067, 72.58658076138724 24.936642289396488,
70.99164208674422 24.61938926179787, 69.9609375 24.5271348225978)
Curnved WKT = COMPOUNDCURVE (CIRCULARSTRING (27.0703125 23.885837699862005, 5.9765625
40.17887331434696, 22.5 47.98992166741417), (22.5 47.98992166741417, 71.71875
49.15296965617039), CIRCULARSTRING (71.71875 49.15296965617039, 81.5625
39.36827914916011, 69.9609375 24.5271348225978))
```

```
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]
        ])
])
List<LineString> lineStrings = compoundCurve.components
lineStrings.each { LineString lineString ->
    println lineString
}
```



```
LINESTRING (27.0703125 23.885837699862005, 27.022331449414295 23.8488525605847,
25.524767337284196 22.84821221194479, 23.909405319245263 22.051603821364203,
22.203884687289108 21.472657579267334, 20.43738737228815 21.12127941637069,
18.64013863306764 21.00348151046186, 16.842889893847133 21.1212794163707,
15.076392578846175 21.472657579267356, 13.370871946890022 22.051603821364232,
11.755509928851094 22.848212211944833, 10.257945816720998 23.848852560584746,
8.903803347661494 25.036403633488835, 7.716252274757412 26.390546102548342,
6.715611926117504 27.88811021467844, 5.919003535536911 29.50347223271737,
5.340057293440038 31.20899286467353, 4.988679130543391 32.97549017967449,
4.8708812246345605 34.772738918894994, 4.988679130543396 36.5699876581155,
5.340057293440047 38.336484973116455, 5.9190035355369215 40.04200560507262, 5.9765625
40.17887331434696, 6.715611926117516 41.65736762311154, 7.71625227475743
43.15493173524164, 8.903803347661514 44.50907420430114, 10.257945816721021
45.69662527720523, 11.755509928851117 46.69726562584515, 13.37087194689005
47.49387401642574, 15.076392578846209 48.07282025852261, 16.84288989384717
48.42419842141926, 18.64013863306768 48.5419963273281, 20.437387372288192
48.42419842141926, 22.20388468728915 48.07282025852261, 22.5 47.98992166741417)
LINESTRING (22.5 47.98992166741417, 71.71875 49.15296965617039)
LINESTRING (71.71875 49.15296965617039, 72.58658076138724 48.953451060440116,
74.12646468490568 48.43073090444654, 75.58494601643167 47.71148750685245,
76.93706973601537 46.80802732160263, 78.15970063193521 45.735808802953194,
79.23191915058464 44.513177907033366, 80.13537933583446 43.16105418744966,
80.85462273342854 41.70257285592367, 81.37734288942212 40.162688932405224, 81.5625
39.36827914916011, 81.69459591702075 38.567750257762206, 81.80095352896302
36.9450466749183, 81.69459591702075 35.3223430920744, 81.37734288942212
33.72740441743138, 80.85462273342854 32.187520493912935, 80.13537933583446
30.729039162386947, 79.23191915058464 29.376915442803238, 78.15970063193521
28.15428454688341, 76.93706973601537 27.082066028233974, 75.58494601643167
26.178605842984155, 74.12646468490568 25.459362445390067, 72.58658076138724
24.936642289396488, 70.99164208674422 24.61938926179787, 69.9609375 24.5271348225978)
```

```
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]
        ])
])
Geometry linear = compoundCurve.linear
println linear.wkt
```



```
LINESTRING (27.0703125 23.885837699862005, 27.022331449414295 23.8488525605847,
25.524767337284196 22.84821221194479, 23.909405319245263 22.051603821364203,
22.203884687289108 21.472657579267334, 20.43738737228815 21.12127941637069,
18.64013863306764 21.00348151046186, 16.842889893847133 21.1212794163707,
15.076392578846175 21.472657579267356, 13.370871946890022 22.051603821364232,
11.755509928851094 22.848212211944833, 10.257945816720998 23.848852560584746,
8.903803347661494 25.036403633488835, 7.716252274757412 26.390546102548342,
6.715611926117504 27.88811021467844, 5.919003535536911 29.50347223271737,
5.340057293440038 31.20899286467353, 4.988679130543391 32.97549017967449,
4.8708812246345605 34.772738918894994, 4.988679130543396 36.5699876581155,
5.340057293440047 38.336484973116455, 5.9190035355369215 40.04200560507262, 5.9765625
40.17887331434696, 6.715611926117516 41.65736762311154, 7.71625227475743
43.15493173524164, 8.903803347661514 44.50907420430114, 10.257945816721021
45.69662527720523, 11.755509928851117 46.69726562584515, 13.37087194689005
47.49387401642574, 15.076392578846209 48.07282025852261, 16.84288989384717
48.42419842141926, 18.64013863306768 48.5419963273281, 20.437387372288192
48.42419842141926, 22.20388468728915 48.07282025852261, 22.5 47.98992166741417,
71.71875 49.15296965617039, 72.58658076138724 48.953451060440116, 74.12646468490568
48.43073090444654, 75.58494601643167 47.71148750685245, 76.93706973601537
46.80802732160263, 78.15970063193521 45.735808802953194, 79.23191915058464
44.513177907033366, 80.13537933583446 43.16105418744966, 80.85462273342854
41.70257285592367, 81.37734288942212 40.162688932405224, 81.5625 39.36827914916011,
81.69459591702075 38.567750257762206, 81.80095352896302 36.9450466749183,
81.69459591702075 35.3223430920744, 81.37734288942212 33.72740441743138,
80.85462273342854 32.187520493912935, 80.13537933583446 30.729039162386947,
79.23191915058464 29.376915442803238, 78.15970063193521 28.15428454688341,
76.93706973601537 27.082066028233974, 75.58494601643167 26.178605842984155,
74.12646468490568 25.459362445390067, 72.58658076138724 24.936642289396488,
70.99164208674422 24.61938926179787, 69.9609375 24.5271348225978)
```

# **Compound Rings**

```
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],
       ])
])
println "WKT = ${compoundRing.wkt}"
println "Curnved WKT = ${compoundRing.curvedWkt}"
```



```
WKT = LINEARRING (27.0703125 23.885837699862005, 27.022331449414295 23.8488525605847,
25.524767337284196 22.84821221194479, 23.909405319245263 22.051603821364203,
22.203884687289108 21.472657579267334, 20.43738737228815 21.12127941637069,
18.64013863306764 21.00348151046186, 16.842889893847133 21.1212794163707,
15.076392578846175 21.472657579267356, 13.370871946890022 22.051603821364232,
11.755509928851094 22.848212211944833, 10.257945816720998 23.848852560584746,
8.903803347661494 25.036403633488835, 7.716252274757412 26.390546102548342,
6.715611926117504 27.88811021467844, 5.919003535536911 29.50347223271737,
5.340057293440038 31.20899286467353, 4.988679130543391 32.97549017967449,
4.8708812246345605 34.772738918894994, 4.988679130543396 36.5699876581155,
5.340057293440047 38.336484973116455, 5.9190035355369215 40.04200560507262, 5.9765625
40.17887331434696, 6.715611926117516 41.65736762311154, 7.71625227475743
43.15493173524164, 8.903803347661514 44.50907420430114, 10.257945816721021
45.69662527720523, 11.755509928851117 46.69726562584515, 13.37087194689005
47.49387401642574, 15.076392578846209 48.07282025852261, 16.84288989384717
48.42419842141926, 18.64013863306768 48.5419963273281, 20.437387372288192
48.42419842141926, 22.20388468728915 48.07282025852261, 22.5 47.98992166741417,
71.71875 49.15296965617039, 72.58658076138724 48.953451060440116, 74.12646468490568
48.43073090444654, 75.58494601643167 47.71148750685245, 76.93706973601537
46.80802732160263, 78.15970063193521 45.735808802953194, 79.23191915058464
44.513177907033366, 80.13537933583446 43.16105418744966, 80.85462273342854
41.70257285592367, 81.37734288942212 40.162688932405224, 81.5625 39.36827914916011,
81.69459591702075 38.567750257762206, 81.80095352896302 36.9450466749183,
81.69459591702075 35.3223430920744, 81.37734288942212 33.72740441743138,
80.85462273342854 32.187520493912935, 80.13537933583446 30.729039162386947,
79.23191915058464 29.376915442803238, 78.15970063193521 28.15428454688341,
76.93706973601537 27.082066028233974, 75.58494601643167 26.178605842984155,
74.12646468490568 25.459362445390067, 72.58658076138724 24.936642289396488,
70.99164208674422 24.61938926179787, 69.9609375 24.5271348225978, 27.0703125
23.885837699862005)
Curnved WKT = COMPOUNDCURVE (CIRCULARSTRING (27.0703125 23.885837699862005, 5.9765625
40.17887331434696, 22.5 47.98992166741417), (22.5 47.98992166741417, 71.71875
49.15296965617039), CIRCULARSTRING (71.71875 49.15296965617039, 81.5625
39.36827914916011, 69.9609375 24.5271348225978), (69.9609375 24.5271348225978,
27.0703125 23.885837699862005))
```

```
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],
        ])
1)
List<LineString> lineStrings = compoundRing.components
lineStrings.each { LineString lineString ->
    println lineString
}
```



```
LINESTRING (27.0703125 23.885837699862005, 27.022331449414295 23.8488525605847,
25.524767337284196 22.84821221194479, 23.909405319245263 22.051603821364203,
22.203884687289108 21.472657579267334, 20.43738737228815 21.12127941637069,
18.64013863306764 21.00348151046186, 16.842889893847133 21.1212794163707,
15.076392578846175 21.472657579267356, 13.370871946890022 22.051603821364232,
11.755509928851094 22.848212211944833, 10.257945816720998 23.848852560584746,
8.903803347661494 25.036403633488835, 7.716252274757412 26.390546102548342,
6.715611926117504 27.88811021467844, 5.919003535536911 29.50347223271737,
5.340057293440038 31.20899286467353, 4.988679130543391 32.97549017967449,
4.8708812246345605 34.772738918894994, 4.988679130543396 36.5699876581155,
5.340057293440047 38.336484973116455, 5.9190035355369215 40.04200560507262, 5.9765625
40.17887331434696, 6.715611926117516 41.65736762311154, 7.71625227475743
43.15493173524164, 8.903803347661514 44.50907420430114, 10.257945816721021
45.69662527720523, 11.755509928851117 46.69726562584515, 13.37087194689005
47.49387401642574, 15.076392578846209 48.07282025852261, 16.84288989384717
48.42419842141926, 18.64013863306768 48.5419963273281, 20.437387372288192
48.42419842141926, 22.20388468728915 48.07282025852261, 22.5 47.98992166741417)
LINESTRING (22.5 47.98992166741417, 71.71875 49.15296965617039)
LINESTRING (71.71875 49.15296965617039, 72.58658076138724 48.953451060440116,
74.12646468490568 48.43073090444654, 75.58494601643167 47.71148750685245,
76.93706973601537 46.80802732160263, 78.15970063193521 45.735808802953194,
79.23191915058464 44.513177907033366, 80.13537933583446 43.16105418744966,
80.85462273342854 41.70257285592367, 81.37734288942212 40.162688932405224, 81.5625
39.36827914916011, 81.69459591702075 38.567750257762206, 81.80095352896302
36.9450466749183, 81.69459591702075 35.3223430920744, 81.37734288942212
33.72740441743138, 80.85462273342854 32.187520493912935, 80.13537933583446
30.729039162386947, 79.23191915058464 29.376915442803238, 78.15970063193521
28.15428454688341, 76.93706973601537 27.082066028233974, 75.58494601643167
26.178605842984155, 74.12646468490568 25.459362445390067, 72.58658076138724
24.936642289396488, 70.99164208674422 24.61938926179787, 69.9609375 24.5271348225978)
LINESTRING (69.9609375 24.5271348225978, 27.0703125 23.885837699862005)
```

```
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],
        ])
])
Geometry linear = compoundRing.linear
println linear.wkt
```



```
LINEARRING (27.0703125 23.885837699862005, 27.022331449414295 23.8488525605847,
25.524767337284196 22.84821221194479, 23.909405319245263 22.051603821364203,
22.203884687289108 21.472657579267334, 20.43738737228815 21.12127941637069,
18.64013863306764 21.00348151046186, 16.842889893847133 21.1212794163707,
15.076392578846175 21.472657579267356, 13.370871946890022 22.051603821364232,
11.755509928851094 22.848212211944833, 10.257945816720998 23.848852560584746,
8.903803347661494 25.036403633488835, 7.716252274757412 26.390546102548342,
6.715611926117504 27.88811021467844, 5.919003535536911 29.50347223271737,
5.340057293440038 31.20899286467353, 4.988679130543391 32.97549017967449,
4.8708812246345605 34.772738918894994, 4.988679130543396 36.5699876581155,
5.340057293440047 38.336484973116455, 5.9190035355369215 40.04200560507262, 5.9765625
40.17887331434696, 6.715611926117516 41.65736762311154, 7.71625227475743
43.15493173524164, 8.903803347661514 44.50907420430114, 10.257945816721021
45.69662527720523, 11.755509928851117 46.69726562584515, 13.37087194689005
47.49387401642574, 15.076392578846209 48.07282025852261, 16.84288989384717
48.42419842141926, 18.64013863306768 48.5419963273281, 20.437387372288192
48.42419842141926, 22.20388468728915 48.07282025852261, 22.5 47.98992166741417,
71.71875 49.15296965617039, 72.58658076138724 48.953451060440116, 74.12646468490568
48.43073090444654, 75.58494601643167 47.71148750685245, 76.93706973601537
46.80802732160263, 78.15970063193521 45.735808802953194, 79.23191915058464
44.513177907033366, 80.13537933583446 43.16105418744966, 80.85462273342854
41.70257285592367, 81.37734288942212 40.162688932405224, 81.5625 39.36827914916011,
81.69459591702075 38.567750257762206, 81.80095352896302 36.9450466749183,
81.69459591702075 35.3223430920744, 81.37734288942212 33.72740441743138,
80.85462273342854 32.187520493912935, 80.13537933583446 30.729039162386947,
79.23191915058464 29.376915442803238, 78.15970063193521 28.15428454688341,
76.93706973601537 27.082066028233974, 75.58494601643167 26.178605842984155,
74.12646468490568 25.459362445390067, 72.58658076138724 24.936642289396488,
70.99164208674422 24.61938926179787, 69.9609375 24.5271348225978, 27.0703125
23.885837699862005)
```

## **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

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

```
(-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



Buffer a LineString on the left side only

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



Buffer a LineString with a beginning and end distance.



Buffer a LineString with a beginning, middle, and end distance.



Buffer a LineString with a list of distances.



```
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
```



```
Polygon polygon1 = new Polygon([[
        [-120.212, 47.591],
        [-119.663, 47.591],
        [-119.663, 47.872],
        [-120.212, 47.872],
        [-120.212, 47.591]
11)
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
```







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)
```



```
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
```



```
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
```





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



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 doesCross13 = line1.crosses(line3)
println 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)
```



```
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)
```



```
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
```



```
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 Interior Point from a Geometry



## Get the number of Geometries

3

## Get a Geometry by index

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

## Get a List of Geometries

```
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)
```



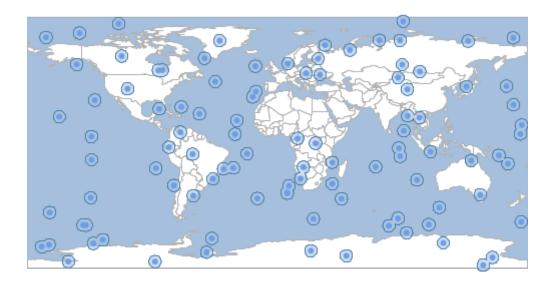
Create a Kock Snowflake in a given Bounds and with a number of points

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



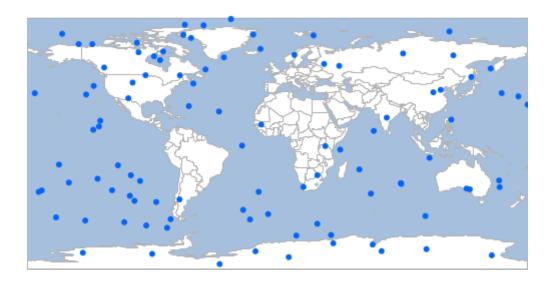
Perform a cascaded union on a List of Polygons.

```
Geometry randomPoints = Geometry.createRandomPoints(new Bounds(-180,-90,180,90,
"EPSG:4326").geometry, 100)
List<Geometry> bufferedPoints = randomPoints.collect{Geometry geom -> geom.buffer(
4.5)}
Geometry unionedGeometry = Geometry.cascadedUnion(bufferedPoints)
```



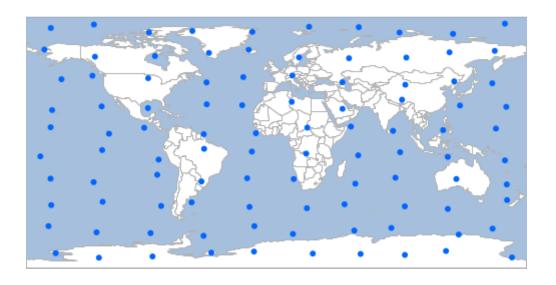
Create a number of random points within a given Geometry

```
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

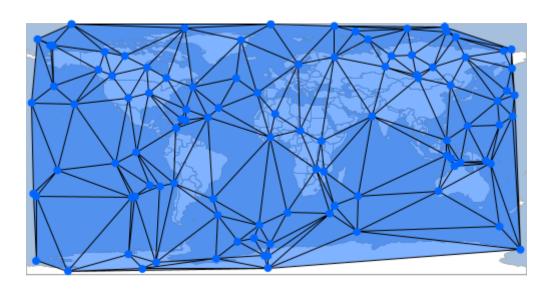
```
Bounds bounds = new Bounds(-180, -90, 180, 90)
MultiPoint randomPoints = Geometry.createRandomPointsInGrid(bounds, 100, true, 0.5)
```



## Create a delaunay triangle diagram around a Geometry

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

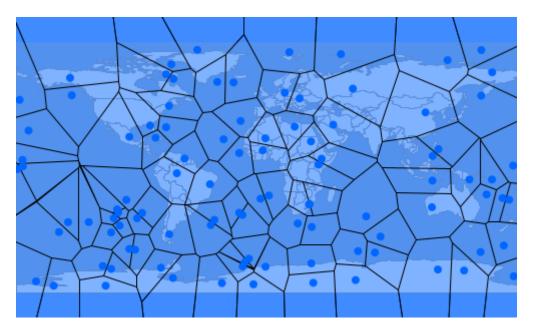
Geometry delaunayTriangle = points.delaunayTriangleDiagram



## 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)
```



```
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)
```



```
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]
11)
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]
11)
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





#### Calculate the minimum circle of a Geometry



## Calculate the minimum diameter of a Geometry





Calculate the largest empty circle within a Geometry





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)
```



```
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



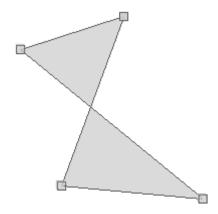
Is the Geometry simple? false

Determine if a Geometry is valid



Is the Geometry valid? true

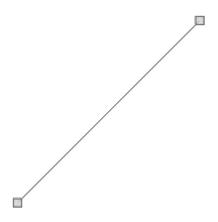
```
Geometry geom2 = new Polygon(new LinearRing([
       [48.16406, 42.29356],
       [35.15625, 25.79989],
       [64.33593, 24.52713],
       [26.71875, 39.09596],
       [48.16406, 42.29356],
]))
boolean isGeom2Valid = geom2.valid
println "Is the Geometry valid? ${isGeom2Valid}"
println geom2.validReason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



Is the Geometry valid? false Self-intersection

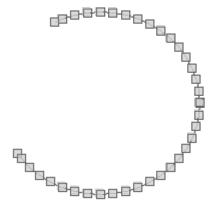
#### Fix a Geometry

```
Geometry line = new LineString([[0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [1, 1]])
println "LineString with duplicated Points = ${line.wkt}"
Geometry fixedLine = line.fix()
println "Fixed LineString = ${fixedLine}"
```

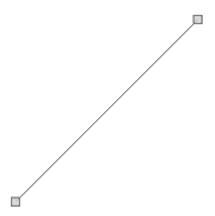


```
LineString with duplicated Points = LINESTRING (0 0, 0 0, 0 0, 0 0, 0 1 1)
Fixed LineString = LINESTRING (0 0, 1 1)
```

#### Determine if a Geometry is curved



Is the Geometry curved? true



```
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'}"
```

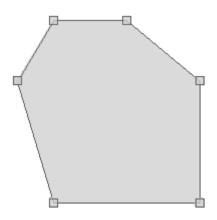
```
Is POINT (-88.945 41.771) within 26.0 of POINT (-113.906 37.16)? Yes
```

```
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.

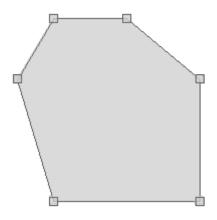
```
Geometry geometry = Geometry.fromWKT("POLYGON((2 4, 1 3, 2 1, 6 1, 6 3, 4 4, 2 4))")
geometry.normalize()
println "Normalized Geometry = ${geometry}"
```



```
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}"
```

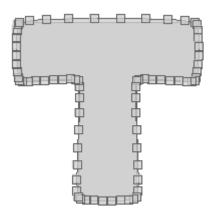


```
Un-normalized Geometry = POLYGON ((2 4, 1 3, 2 1, 6 1, 6 3, 4 4, 2 4))
Normalized Geometry = POLYGON ((1 3, 2 4, 4 4, 6 3, 6 1, 2 1, 1 3))
```

#### Smooth a Geometry

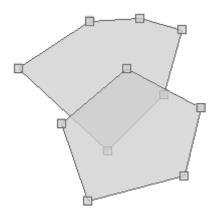
```
Geometry geometry = Geometry.fromWKT("POLYGON((10 0, 10 20, 0 20, 0 30, 30 30, 30 20, 20 20, 20 0, 10 0))")

Geometry smoothed = geometry.smooth(0.75)
```



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

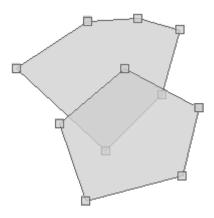
```
Polygon polygon1 = new Polygon([[
        [-121.915, 47.390],
        [-122.640, 46.995],
        [-121.739, 46.308],
        [-121.168, 46.777],
        [-120.981, 47.316],
        [-121.409, 47.413],
        [-121.915, 47.390]
]])
Polygon polygon2 = new Polygon([[
        [-120.794, 46.664],
        [-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
```

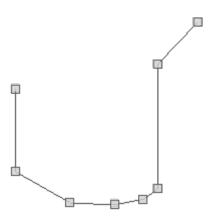
Determine if a Geometry relates to another Geometry according to the given DE-9IM Intersection Matrix string

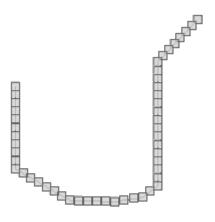
```
Polygon polygon1 = new Polygon([[
        [-121.915, 47.390],
        [-122.640, 46.995],
        [-121.739, 46.308],
        [-121.168, 46.777],
        [-120.981, 47.316],
        [-121.409, 47.413],
        [-121.915, 47.390]
]])
Polygon polygon2 = new Polygon([[
        [-120.794, 46.664],
        [-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.

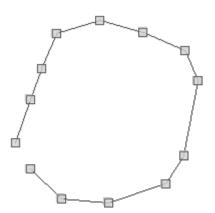


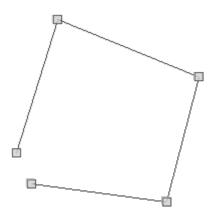


```
# of points in original geometry = 8
# of points in densified geometry = 50
```

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],
    \hbox{[-123.59619140625001, } 48.10743118848039],
    [-123.85986328124999, 47.62097541515849]
1)
Geometry simplified = geometry.simplify(0.5)
println "# of points in original geometry = ${geometry.numPoints}"
println "# of points in simplified geometry = ${simplified.numPoints}"
```

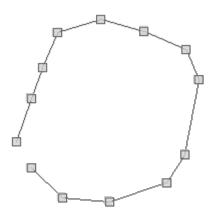


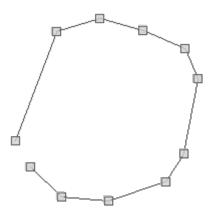


```
# of points in original geometry = 13
# of points in simplified geometry = 5
```

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]
1)
Geometry simplified = geometry.simplifyPreservingTopology(0.1)
println "# of points in original geometry = ${geometry.numPoints}"
println "# of points in simplified geometry = ${simplified.numPoints}"
```

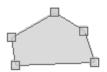




```
# of points in original geometry = 13
# of points in simplified geometry = 11
```

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



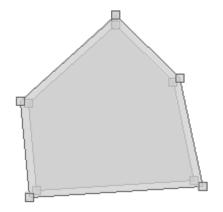


Scale a geometry a given amount in an x and y direction around the origin

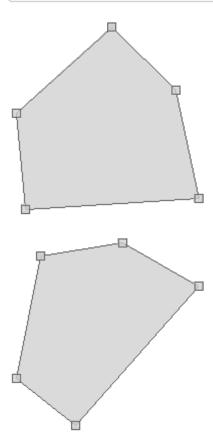




Scale a geometry a given amount in an x and y direction around a point

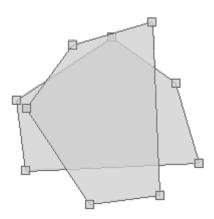


Rotate a Geometry around it's origin by a given angle theta (in radians).



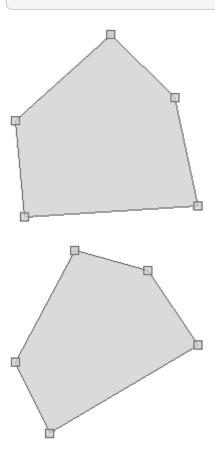
Rotate a Geometry around an XY coordinate by a given angle theta (in radians).

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



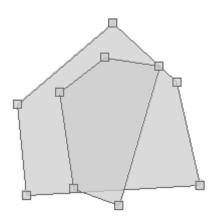
Rotate a Geometry around it's origin by a given angle sine and cosine (in radians).

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

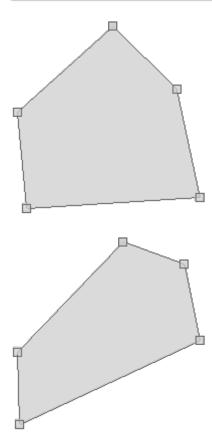


Rotate a Geometry around an XY coordinate by a given angle sine and cosine (in radians).

```
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.

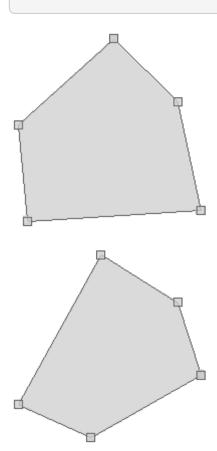


Reflect a Geometry around an XY coordinate for given distance along the x and y axis



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



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 = 85874081 nanoseconds
Time with PreparedGeometry = 84245266 nanoseconds
```

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

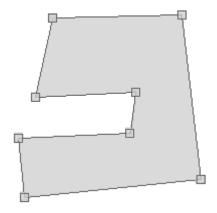
```
Geometry geometry = new Polygon([[
     [-121.915, 47.390],
     [-122.640, 46.995],
     [-121.739, 46.308],
     [-121.168, 46.777],
     [-120.981, 47.316],
     [-121.409, 47.413],
     [-121.915, 47.390]
]])
PreparedGeometry preparedGeometry = Geometry.prepare(geometry)
Closure timer = { Closure 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 = 75915658 nanoseconds
Time with PreparedGeometry = 60494850 nanoseconds
```

# **Prepared Geometry**

PreparedGeometry are more effecient for spatial queries.

You can create a PreparedGeometry by wrapping an existing Geometry. You can then get the original Geometry with the geometry property.



```
PreparedGeometry polygon1 = new PreparedGeometry(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
```





Check whether a PreparedGeometry properly contains another Geometry. This means that every point in the other geometry doesn't intersect with the first.

```
PreparedGeometry polygon1 = new PreparedGeometry(new Polygon([[
    [-122.50064849853516, 47.22096718353454],
    [-122.41928100585938, 47.22096718353454],
    [-122.41928100585938, 47.277365616965646],
    [-122.50064849853516, 47.277365616965646],
    [-122.50064849853516, 47.22096718353454]
11))
Polygon polygon2 = new Polygon([[
    [-122.44571685791014, 47.24031721435104],
    [-122.42958068847656, 47.24031721435104],
    [-122.42958068847656, 47.253135632244216],
    [-122.44571685791014, 47.253135632244216],
    [-122.44571685791014, 47.24031721435104]
]])
boolean contains = polygon1.containsProperly(polygon2)
println contains
```





Check whether a PreparedGeometry 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]
]])
PreparedGeometry polygon2 = new PreparedGeometry(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
```





Check whether a PreparedGeometry covers another Geometry.

```
PreparedGeometry polygon1 = new PreparedGeometry(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



false

Check whether a PreparedGeometry crosses another Geometry.

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

boolean doesCross12 = line1.crosses(line2)
println doesCross13 = line1.crosses(line3)
println doesCross13
```



true false

```
PreparedGeometry polygon1 = new PreparedGeometry(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]
11)
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]
11)
boolean isDisjoint12 = polygon1.disjoint(polygon2)
println isDisjoint12
boolean isDisjoint13 = polygon1.disjoint(polygon3)
println isDisjoint13
```



```
false
true
```

```
PreparedGeometry polygon1 = new PreparedGeometry(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]
11)
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]
11)
boolean does1intersect2 = polygon1.intersects(polygon2)
println does1intersect2
boolean does1intersect3 = polygon1.intersects(polygon3)
println does1intersect3
```



```
true
false
```

```
PreparedGeometry polygon1 = new PreparedGeometry(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]
11)
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]
11)
boolean does1overlap2 = polygon1.overlaps(polygon2)
println does1overlap2
boolean does1overlap3 = polygon1.overlaps(polygon3)
println does1overlap3
```



```
true
false
```

```
{\it Check\ whether\ a\ Prepared Geometry.}
  PreparedGeometry line1 = new PreparedGeometry(new LineString([
      [-122.38651514053345, 47.58219978280006],
      [-122.38651514053345, 47.58020234903306]
  ]))
 LineString line2 = new LineString([
      [-122.38651514053345, 47.58124449789785],
      [-122.38333940505981, 47.58124449789785]
  1)
  boolean touches = line1.touches(line2)
  true
  LineString line3 = new LineString([
      [-122.386257648468, 47.58183793450921],
```

```
LineString line3 = new LineString([
     [-122.386257648468, 47.58183793450921],
     [-122.38348960876465, 47.5818668824645]
])
touches = line1.touches(line3)
```



```
false
```

```
PreparedGeometry polygon1 = new PreparedGeometry(new Polygon([[
    [-120.212, 47.591],
    [-119.663, 47.591],
    [-119.663, 47.872],
    [-120.212, 47.872],
    [-120.212, 47.591]
11))
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
```



```
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
```





# **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
TWkbReader
YamlReader
```

#### 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
GeoRSSWriter
Gml2Writer
Gml3Writer
GpxWriter
KmlWriter
WkbWriter
WktWriter
GeoPackageWriter
GooglePolylineEncoder
YamlWriter
```

#### 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 = "0000000001C05EC999999999A40471E5604189375"
WkbReader reader = new WkbReader()
Geometry geometry = reader.read(wkb)
```

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

```
String wkb =
"0000000000004400995810624DD2F404594FDF3B645A2401ADA1CAC0831274048E0A3D70A3D71402
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
```

```
000000001C05EC999999999A40471E5604189375
```

Write a Geometry to WKB using the WkbWriter

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

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

#### **TWKB**

Read a Geometry from TWKB hex encoded String

```
TWkbReader reader = new TWkbReader()
Geometry geometry = reader.read("01000204")
println geometry.wkt
```

```
POINT (1 2)
```

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

```
{"type":"LineString","coordinates":[[3.198,43.164],[6.713,49.755],[9.702,42.592],[15.3 2,53.798]]}
```

### GeoPackage

Read a Geometry from GeoPackage hex encoded string using the GeoPackageReader

Read a Geometry from GeoPackage bytes using the GeoPackageReader

```
byte[] bytes =
"47500002000000000c05ec9999999999ac05ec9999999999a40471e560418937540471e560418937500000
00001c05ec9999999999a40471e5604189375".decodeHex()
GeoPackageReader reader = new GeoPackageReader()
Geometry geometry = reader.read(bytes)
```

Write a Geometry to a GeoPackage hex encoded string

```
Geometry geometry = new Point(-123.15, 46.237)
GeoPackageWriter writer = new GeoPackageWriter()
String str = writer.write(geometry)
println str
```

4750000200000000c05ec999999999ac05ec9999999999a40471e560418937540471e5604189375000000000001c05ec99999999a40471e5604189375

Write a Geometry to a GeoPackage byte array

```
Geometry geometry = new Point(-123.15, 46.237)
GeoPackageWriter writer = new GeoPackageWriter()
byte[] bytes = writer.writeBytes(geometry)
println bytes.encodeHex()
```

4750000200000000c05ec999999999ac05ec9999999999a40471e560418937540471e5604189375000000 0001c05ec9999999a40471e5604189375

#### **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 from Kml() 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)
```



```
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

```
<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)
```

```
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 Geobuf Writer

```
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<sub>2</sub>

```
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

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

#### GML<sub>3</sub>

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:Point>
```

Write a Geometry to GML3 using the Gml3Writer

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

```
<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

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

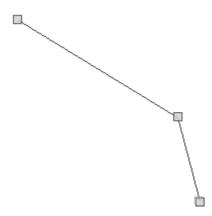
## **Google Polyline**

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@
```

### **YAML**

Get a Geometry from a GeoYaml String

```
String yaml = """---
geometry:
    type: "Point"
    coordinates:
    - -122.23
    - 45.67
"""
Geometry geometry = Geometry.fromYaml(yaml)
```

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

```
geometry:
  type: "Point"
  coordinates:
  - -123.15
  - 46.237
```

Read a Geometry from a GeoYaml String using the YamlReader

```
String yaml = """---
geometry:
    type: "Point"
    coordinates:
    - -122.23
    - 45.67
"""

YamlReader reader = new YamlReader()
    Geometry geometry = reader.read(yaml)
```

Write a Geometry to a GeoYaml String using the YamlWriter

```
geometry:
type: "LineString"
coordinates:
- - 3.198
- 43.164
- - 6.713
- 49.755
- - 9.702
- 42.592
- 15.32
- 53.798
```