

Geoscript Groovy Cookbook

Jared Erickson

Table of Contents

Geometry Recipes.....	1
Creating Geometries	1
Procesing Geometries	8
Read and Write Geometries	9
Projection Recipes	12
Creating Projections	12
Using Projections	13
Plot Recipes	14
Creating a Bar Chart	14

Geometry Recipes

Creating Geometries

Create a Point with an XY

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



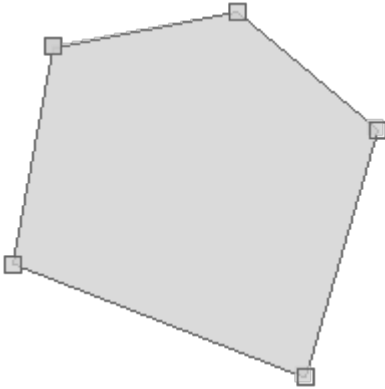
Create a LineString from Coordinates

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



Create a Polygon from a List of Coordinates

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



Create a MultiPoint with a List of Points

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



Create a MultiLineString with a List of LineStrings

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



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]  
    ]])  
    ]])  
)
```



Create a CircularString with a List of Points

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



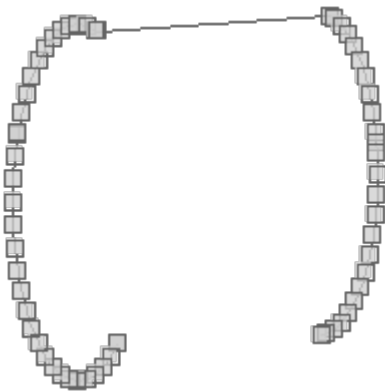
Create a *CircularRing* with a List of Points

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



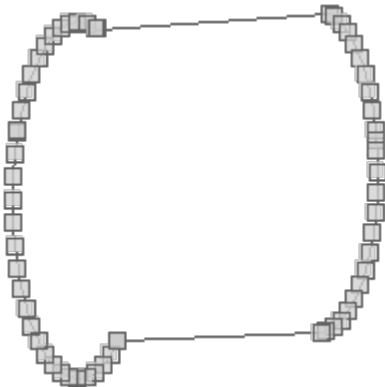
Create a CompoundCurve with a List of CircularStrings and LineStrings

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



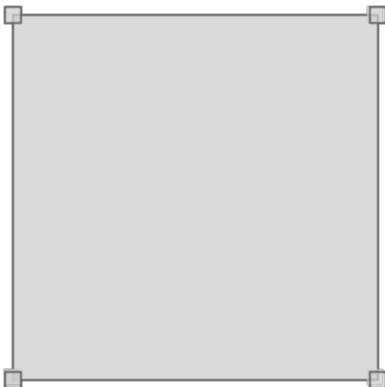
Create a *CompoundRing* with a connected List of *CircularStrings* and *LineStrings*

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



Create a *Bounds* from four coordinates (minx, miny, maxx, maxy) and a projection.

```
Bounds bounds = new Bounds(-127.265, 43.068, -113.554, 50.289, "EPSG:4326")
```



Processing Geometries

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

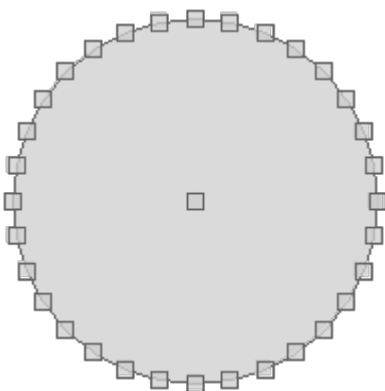
Get the length of a Geometry

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

23.24738479915536

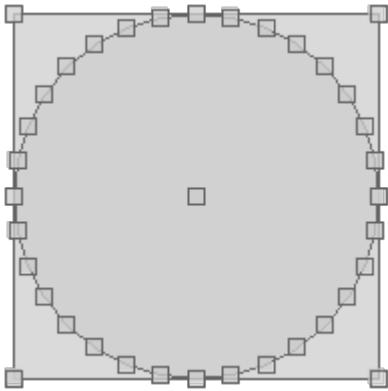
Buffer a Point

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



Get Bounds from a Geometry

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



Read and Write Geometries

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

WKT

Read a Geometry from WKT using the `WktReader`

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



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

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



Get the WKT of a Geometry

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

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

Write a Geometry to WKT using the WktWriter

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

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

GeoJSON

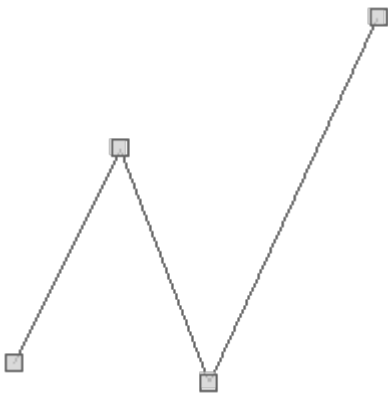
Read a Geometry from GeoJSON using the GeoJSONReader

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



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

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



Get the GeoJSON of a Geometry

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

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

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

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

Projection Recipes

Creating Projections

Create a Projection from an EPSG Code

```
Projection proj = new Projection("EPSG:4326")  
println proj.wkt
```

```
GEOGCS["WGS 84",  
  DATUM["World Geodetic System 1984",  
    SPHEROID["WGS 84", 6378137.0, 298.257223563, AUTHORITY["EPSG","7030"]],  
    AUTHORITY["EPSG","6326"]],  
  PRIMEM["Greenwich", 0.0, AUTHORITY["EPSG","8901"]],  
  UNIT["degree", 0.017453292519943295],  
  AXIS["Geodetic longitude", EAST],  
  AXIS["Geodetic latitude", NORTH],  
  AUTHORITY["EPSG","4326"]]
```

Create a Projection from a WKT Projection String

```
Projection proj = new Projection("GEOGCS[\"WGS 84\",  
DATUM[\"World Geodetic System 1984\",  
  SPHEROID[\"WGS 84\", 6378137.0, 298.257223563, AUTHORITY[\"EPSG\", \"7030\"]],  
  AUTHORITY[\"EPSG\", \"6326\"]],  
PRIMEM[\"Greenwich\", 0.0, AUTHORITY[\"EPSG\", \"8901\"]],  
UNIT[\"degree\", 0.017453292519943295],  
AXIS[\"Geodetic longitude\", EAST],  
AXIS[\"Geodetic latitude\", NORTH],  
AUTHORITY[\"EPSG\", \"4326\"]\"")
```

```
GEOGCS[\"WGS 84\",  
  DATUM[\"World Geodetic System 1984\",  
    SPHEROID[\"WGS 84\", 6378137.0, 298.257223563, AUTHORITY[\"EPSG\", \"7030\"]],  
    AUTHORITY[\"EPSG\", \"6326\"]],  
  PRIMEM[\"Greenwich\", 0.0, AUTHORITY[\"EPSG\", \"8901\"]],  
  UNIT[\"degree\", 0.017453292519943295],  
  AXIS[\"Geodetic longitude\", EAST],  
  AXIS[\"Geodetic latitude\", NORTH],  
  AUTHORITY[\"EPSG\", \"4326\"]]
```

Create a Projection from well known name

```
Projection proj = new Projection("Mollweide")  
println proj.wkt
```

```
PROJCS[\"Mollweide\",  
  GEOGCS[\"WGS84\",  
    DATUM[\"WGS84\",  
      SPHEROID[\"WGS84\", 6378137.0, 298.257223563]],  
    PRIMEM[\"Greenwich\", 0.0],  
    UNIT[\"degree\", 0.017453292519943295],  
    AXIS[\"Longitude\", EAST],  
    AXIS[\"Latitude\", NORTH]],  
  PROJECTION[\"Mollweide\"],  
  PARAMETER[\"semi-minor axis\", 6378137.0],  
  PARAMETER[\"Longitude of natural origin\", 0.0],  
  UNIT[\"m\", 1.0],  
  AXIS[\"Easting\", EAST],  
  AXIS[\"Northing\", NORTH]]
```

Using Projections

Transform a Geometry from one projection to another

```
Geometry epsg4326Geom = new Point(-122.440, 47.245)
Geometry epsg2927Geom = Projection.transform(epsg4326Geom, "EPSG:4326", "EPSG:2927")
println epsg2927Geom
```

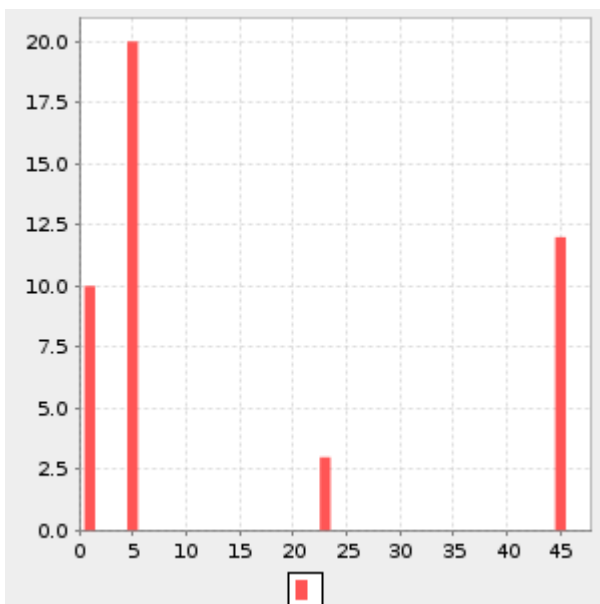
```
POINT (1158609.2040371667 703068.0661327887)
```

Plot Recipes

Creating a Bar Chart

Create a basic bar chart

```
List data = [
    [1,10],[45,12],[23,3],[5,20]
]
Chart chart = Bar.xy(data)
```



Create a bar chart with categories

```
Map data = [
    "A":20,"B":45,"C":2,"D":14
]
Chart chart = Bar.category(data)
```