

# Table of Contents

Projection Recipes .....	1
Creating Projections .....	1
Getting Projection Properties .....	3
Using Projections .....	4
Using Geodetic .....	5
Using Decimal Degrees .....	6

## Projection Recipes

The Projection classes are in the [geoscript.proj](#) package.

### 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["False easting", 0.0],  
  PARAMETER["False northing", 0.0],  
  PARAMETER["Longitude of natural origin", 0.0],  
  UNIT["m", 1.0],  
  AXIS["Easting", EAST],  
  AXIS["Northing", NORTH]]
```

*Get a List of all supported Projections (this is really slow)*

```
List<Projection> projections = Projection.projections()
```

```
EPSG:4326  
EPSG:4269  
EPSG:26918  
EPSG:2263  
EPSG:2927  
...
```

# Getting Projection Properties

*Get the id*

```
Projection proj = new Projection("EPSG:4326")
String id = proj.id
```

EPSG:4326

*Get the srs*

```
String srs = proj.srs
```

EPSG:4326

*Get the epsg code*

```
int epsg = proj.epsg
```

4326

*Get the WKT*

```
String wkt = 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"]]
```

*Get the Bounds in the native Projection*

```
Bounds bounds = proj.bounds
```

(-180.0,-90.0,180.0,90.0, EPSG:4326)

```
Bounds geoBounds = proj.geoBounds
```

```
(-180.0, -90.0, 180.0, 90.0, EPSG:4326)
```

## Using Projections

*Transform a Geometry from one projection to another using the Projection static method with strings*

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

```
POINT (1158609.2040371664 703068.0661327793)
```

*Transform a Geometry from one projection to another using the Projection static method with Projections*

```
Projection epsg4326 = new Projection("EPSG:4326")
Projection epsg2927 = new Projection("EPSG:2927")
Geometry epsg4326Geom = new Point(-122.440, 47.245)
Geometry epsg2927Geom = Projection.transform(epsg4326Geom, epsg4326, epsg2927)
println epsg2927Geom
```

```
POINT (1158609.2040371664 703068.0661327793)
```

*Transform a Geometry from one projection to another using two Projections*

```
Projection fromProj = new Projection("EPSG:4326")
Projection toProj = new Projection("EPSG:2927")
Geometry geom = new Point(-122.440, 47.245)
Geometry projectedGeom = fromProj.transform(geom, toProj)
println projectedGeom
```

```
POINT (1158609.2040371664 703068.0661327793)
```

*Transform a Geometry from one projection to another using a Projections and a String*

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

```
POINT (1158609.2040371664 703068.0661327793)
```

## Using Geodetic

*Create a Geodetic object with an ellipsoid*

```
Geodetic geodetic = new Geodetic("wgs84")  
println geodetic
```

```
Geodetic [SPHEROID["WGS 84", 6378137.0, 298.257223563]]
```

*Calculate the forward and back azimuth and distance between the given two Points.*

```
Geodetic geodetic = new Geodetic("clrk66")  
Point bostonPoint = new Point(-71.117, 42.25)  
Point portlandPoint = new Point(-123.683, 45.52)  
Map results = geodetic.inverse(bostonPoint, portlandPoint)  
double forwardAzimuth = results.forwardAzimuth  
println forwardAzimuth
```

```
-66.52547810974724
```

```
double backAzimuth = results.backAzimuth  
println backAzimuth
```

```
75.65817457195088
```

```
double distance = results.distance  
println distance
```

```
4164050.459880065
```

*Calculate a new Point and back azimuth given the starting Point, azimuth, and distance.*

```
Geodetic geodetic = new Geodetic("clrk66")  
Point bostonPoint = new Point(-71.117, 42.25)  
Map results = geodetic.forward(bostonPoint, -66.531, 4164192.708)  
Point point = results.point  
println point
```

```
POINT (-123.6835797667373 45.51642779589723)
```

```
double azimuth = results.backAzimuth  
println azimuth
```

```
75.65337425050723
```

*Place the given number of points between starting and ending Points*

```
Geodetic geodetic = new Geodetic("c1rk66")  
Point bostonPoint = new Point(-71.117, 42.25)  
Point portlandPoint = new Point(-123.683, 45.52)  
List<Point> points = geodetic.placePoints(bostonPoint, portlandPoint, 10)  
points.each { Point point ->  
    println point.wkt  
}
```

```
POINT (-75.41357382496236 43.52791689304304)  
POINT (-79.8828640042499 44.63747566950249)  
POINT (-84.51118758826816 45.565540142641005)  
POINT (-89.27793446221685 46.300124344169255)  
POINT (-94.15564606698499 46.83102721803566)  
POINT (-99.11079892605703 47.15045006457598)  
POINT (-104.10532353179985 47.25351783423774)  
POINT (-109.09873812691617 47.13862709798196)  
POINT (-114.05062990603696 46.80756425557422)  
POINT (-118.92312608779855 46.26537395700513)
```

## Using Decimal Degrees

*Create a new DecimalDegrees from a longitude and latitude*

```
DecimalDegrees decimalDegrees = new DecimalDegrees(-122.525619, 47.212023)  
println decimalDegrees
```

```
-122° 31' 32.2284" W, 47° 12' 43.2828" N
```

*Create a new DecimalDegrees from a Point*

```
DecimalDegrees decimalDegrees = new DecimalDegrees(new Point(-122.525619, 47.212023))  
println decimalDegrees
```

```
POINT (-122.52561944444444 47.212022222222224)
```

*Create a new DecimalDegrees from a Longitude and Latitude string*

```
DecimalDegrees decimalDegrees = new DecimalDegrees("-122.525619, 47.212023")  
println decimalDegrees
```

```
-122° 31' 32.2284" W, 47° 12' 43.2828" N
```

*Create a new DecimalDegrees from two strings with glyphs*

```
DecimalDegrees decimalDegrees = new DecimalDegrees("122\u00B0 31' 32.23\" W",  
"47\u00B0 12' 43.28\" N")  
println decimalDegrees
```

```
-122° 31' 32.2300" W, 47° 12' 43.2800" N
```

*Create a new DecimalDegrees from two strings*

```
DecimalDegrees decimalDegrees = new DecimalDegrees("122d 31m 32.23s W", "47d 12m  
43.28s N")  
println decimalDegrees
```

```
-122° 31' 32.2300" W, 47° 12' 43.2800" N
```

*Create a new DecimalDegrees from a single Degrees Minutes Seconds formatted string*

```
DecimalDegrees decimalDegrees = new DecimalDegrees("122d 31m 32.23s W, 47d 12m 43.28s  
N")  
println decimalDegrees
```

```
-122° 31' 32.2300" W, 47° 12' 43.2800" N
```

*Create a new DecimalDegrees from a single Decimal Degree Minutes formatted string with glyphs*

```
DecimalDegrees decimalDegrees = new DecimalDegrees("122\u00B0 31.5372' W, 47\u00B0  
12.7213' N")  
println decimalDegrees
```

```
-122° 31' 32.2320" W, 47° 12' 43.2780" N
```

*Create a new DecimalDegrees from a single Decimal Degree Minutes formatted string*

```
DecimalDegrees decimalDegrees = new DecimalDegrees("122d 31.5372m W, 47d 12.7213m N")  
println decimalDegrees
```

```
-122° 31' 32.2320" W, 47° 12' 43.2780" N
```

*Get degrees minutes seconds from a DecimalDegrees object*

```
DecimalDegrees decimalDegrees = new DecimalDegrees("122d 31m 32.23s W", "47d 12m  
43.28s N")  
Map dms = decimalDegrees.dms  
println "Degrees: ${dms.longitude.degrees}"  
println "Minutes: ${dms.longitude.minutes}"  
println "Seconds: ${dms.longitude.seconds}"
```

```
Degrees: -122  
Minutes: 31  
Seconds: 32.22999999998388
```

```
println "Degrees: ${dms.latitude.degrees}"  
println "Minutes: ${dms.latitude.minutes}"  
println "Seconds: ${dms.latitude.seconds}"
```

```
Degrees: 47  
Minutes: 12  
Seconds: 43.280000000006396
```

*Convert a DecimalDegrees object to a DMS String with glyphs*

```
DecimalDegrees decimalDegrees = new DecimalDegrees("122d 31m 32.23s W", "47d 12m  
43.28s N")  
println decimalDegrees.toDms(true)
```

```
-122° 31' 32.2300" W, 47° 12' 43.2800" N
```

*Convert a DecimalDegrees object to a DMS String without glyphs*

```
println decimalDegrees.toDms(false)
```



```
-122d 31m 32.2300s W, 47d 12m 43.2800s N
```

*Get degrees minutes from a DecimalDegrees object*

```
DecimalDegrees decimalDegrees = new DecimalDegrees("122d 31m 32.23s W", "47d 12m 43.28s N")
Map dms = decimalDegrees.ddm
println "Degrees: ${dms.longitude.degrees}"
println "Minutes: ${dms.longitude.minutes}"
```

```
Degrees: -122
Minutes: 31.537166666666398
```

```
println "Degrees: ${dms.latitude.degrees}"
println "Minutes: ${dms.latitude.minutes}"
```

```
Degrees: 47
Minutes: 12.72133333333344
```

*Convert a DecimalDegrees object to a DDM String with glyphs*

```
DecimalDegrees decimalDegrees = new DecimalDegrees("122d 31m 32.23s W", "47d 12m 43.28s N")
println decimalDegrees.toDdm(true)
```

```
-122° 31.5372' W, 47° 12.7213' N
```

*Convert a DecimalDegrees object to a DDM String without glyphs*

```
println decimalDegrees.toDdm(false)
```

```
-122d 31.5372m W, 47d 12.7213m N
```

*Get a Point from a DecimalDegrees object*

```
DecimalDegrees decimalDegrees = new DecimalDegrees("122d 31m 32.23s W", "47d 12m 43.28s N")
Point point = decimalDegrees.point
```

POINT (-122.52561944444444 47.212022222222224)