Documentation

COMPLETED TASK / PENDING TASK / ~~SCRAPPED~~

CirclePolygonizer/CirclePolgonizerTest To Do’s:

* Implement use of Smileys line class (infLine) vs my CartesianLine class, because he define lines base on a slope and intercept (vs slope and pt). This will take care of the POSITIVE\_INFINITY case
* ~~Calculating the intersect (and maybe even tangent?) line… currently only for quadrant 1. Check that~~
* POSITIVE vs NEGATIVE infinity
* Integrate randomness into tests
* Intersection of two lines: how to handle different cases of
  + Two lines the same
* Calc slope: between two of the same point
* ~~Get quandrant~~
  + ~~What if point is on a axis~~

~~No longer necessary because we do all calcs in quadrant one and translate points at the end. We don’t even need to worry about tangent lines in any other quadrant, or really anything really (at least until further notice, when we go Geodesic and need to do calcs for at least quadrant I&IV~~

* Addition of four boundary points
* Change the functions within the class that are there for convenience to ‘protected’
* Put functions in sequential order as they run (rough sequential order)
* Translation of points into all four quadrants
* assertTrue/False in tests
* use assert instead of exception in calctangentline
* impl a reflection method (args = axis and inclusive or exclusive)
* in calccircintersection use DistanceCalculator -- > calcbearing
* stop using circ.getCetner, use axialCenter (need to make two protected methods public)
* lstofpoints -> result points
* X -> x