SEG 2105, Assignment 1

1. Description of How the Tests were Performed

For the tests, we created a method that would take in a point and a number of tests (n) to perform, make an array of longs which correlated to the point's class methods, then call each method n times and increment the correlated long by the amount of time it took to perform the operation each time. The method would then return the array of longs and we would print it out.

2. Test Sample Outputs
3.

	Design 2: speed (ns)	Design 3: speed (ns)
getX	5655.292	5469.597
getY	5655.292	5471.191
getRho	5514.923	5522.475
getTheta	5518.267	5744.054
getDistance	5711.575	5481.032
rotatePoint	5553.124	5827.807

Table 1: Sample Test Results

4. Predictions and Results

When implementing Design5:

a. Do you still need var typeCoord? No
b. Do you still need the third argument in the construructor? No

	Design 2: predictions	Design 2: findings	Design 3: predictions	Design 3: findings	Design 5:
getX	slowest	slowest	fastest	fastest	Because the cost is determined by the subclass, the speed both in theory and practice reflected the results of the base class used.
getY	slowest	slowest	fastest	fastest	
getRho	fastest	fastest	slowest	slowest	
getTheta	fastest	fastest	slowest	slowest	
getDistance	slowest	slowest	fastest	fastest	
rotatePoint	fastest	fastest	slowest	slowest	

Table 2: predictions and findings of the results

4. PointCPTest Discussion

The PointCPTest suffers from Deep Nesting. Layering if requirements and loops until the developer has trouble remembering what requirements are needed for the current code to be run.

The comments are often used to delimenate sections of the code. This is helpful in mederation, but was used heavy-handedly.

Nesting ternary operators can reduce cole legibility.

5. Discussion about the Designs

	Design 2	Design 3	Design 5
Simplicity of the Code	Very Readable	Very Readable	Very Readable
Efficiency when creating instances	Efficient. Uses inputs as starting variables, but converts degrees to rads to avoid the need to do that when getX and getY are called. This may be division which is more costly.	Very efficient. Uses inputs as starting variables.	See base class used.
Efficiency when doing computations that require both coordinate systems	By precomputing the value of theta in radians, we only need the cos/sin respectively and to multiply by rho.	getRho requires the use of multiplication which is cheep and the sqrt operation which expensive. getTheta requires both sin and cos as well as a conversion to degrees which may imply division.	See base class used.
Amount of memory used	2 doubles per instance.	2 doubles per instance.	2 doubles per instance.
Other considerations	getTheta requires division to convert back. getRho is a simple getter that does not alter the value of rho.	N/A	See base class used.

Table 3: The advantages and disadvantages of each design alternative.