Testing Discussion for SEG2105 Assignment 1

Hypothesis

Design	Advantages	Disadvantages
2: Store polar coordinates only	Stores Polar coordinates only	Needs to process input and convert to either Polar or Cartesian. Slower for initializing cartesian points
3: Store cartesian coordinates only	Stores Cartesian coordinates only	Needs to process input and convert to either Polar or Cartesian Slower for initializing polar points
5: Abstract class with designs 2 and 3 as subclasses	- Would have the most efficient run time - Only stores 1 coordinate which is efficient on memory	Cannot be instantiated

Performance Analysis

Runtime for Different Method calls			
Methods	**Design 2**	**Design 3**	**Design 5**
getX	Min 570 Median 1759.5 Max 2849	Min 4925 Median 6923 Max 8921	Min 21 Median 43 Max 65
getY	Min 609 Median 1072.5 Max 1536	Min 30 Median 59.5 Max 89	Min 19 Median 28.8 Max 41
getRho	Min 25 Median 64 Max 103	Min 29 Median 74.5 Max 120	Min 24 Median 37 Max 50
getTheta	Min 23 Median 45 Max 67	Min 2822 Median 3906 Max 4990	Min 17 Median 28 Max 39
convertStroragetoPolar	Min 949 Median 1476 Max 2002	Min 22 Median 41.5 Max 61	Min 1743 Median 1346.5 Max 950
convertStroragetoCartesian	Min 24 Median 57 Max 90	Min 2589 Median 3613.4 Max 4879	Min 20 Median 75 Max 130
getDistance	Min 1690 Median 2840.5 Max 3991	Min 23 Median 51.5 Max 80	Min 27 Median 56 Max 85
rotatePoint	Min 6504 Median 8201.5 Max 9899	Min 2910 Median 5760 Max 8610	Min 8445 Median 9536.5 Max 10628

Test Case Results from 3 Different Designs.

Design 1 Test:

```
CartesianPolar Coordinates Conversion Program

Enter the type of Coordinates you are inputting ((C)artesian / (P)olar): C
Enter the value of X using a decimal point(.): 7.0
Enter the value of Y using a decimal point(.): 29.0

You entered: Stored as Cartesian  (7.0,29.0)

After asking to store as Cartesian: Stored as Cartesian  (7.0,29.0)
After asking to store as Polar: Stored as Polar 29.832867780352596,1.333947565847976
```

Design 3 Test:

```
CartesianPolar Coordinates Conversion Program

Enter the X Coordinate using a decimal point(.):2.0
Enter the Y Coordinate using a decimal point(.):10.0
Stored as Cartesian  (2.0,10.0)
Do you want to convert to polar yes
After asking to convert to Polar: Converted to  (10.1980390271855,1.37340076694501)
```

Design 5 Test:

```
CartesianPolar Coordinates Conversion Program

Enter the type of Coordinates you are inputting ((C)artesian / (P)olar): P
Enter the Rho Coordinate using a decimal point(.):80.02323
Enter the Theta Coordinate using a decimal point(.):52.033489753
Stored as Polar  (80.02323,52.033489753)
Do you want to convert to cartesian  yes
After asking to convert to Cartesian: Converted to  (49.23035293872719,63.087952012735805
```

Conclusion from 3 Different Designs.

The outcomes of our tests supported our hypothesis that Design 5 would be the most time-effective of the three executed designs. Design 2 accepts either polar or cartesian coordinates, giving it a constant startup time for both types of coordinates. When polar and cartesian coordinates were used, the initialization time for Design 2 was almost exactly the same. As expected, Design 2 was noticeably slower for initializing cartesian points than Design 3. Since Design 3 can only take cartesian points as stores, it has an advantage over Design 2 in the initialization of cartesian points.The most effective design, Design 5, easily outperformed both Designs 2 and 3. Cartesian and polar coordinates could both be accepted by Design 5 and initialized relatively quickly. It was much quicker to store and decide which coordinate to initialize by having both types of coordinates be subclasses of an abstract superclass compared to Design 2, which had to store only one type of coordinate while it waited for the flag to indicate which type it was, and then had only one pair of instance variables initialize which point to create.