<u>Assignment 1</u>

Felix Singerman - 7970742

Eyob Tilaye - 7754646

Output Generated in the test cases for the 3 designs.

<u>Design 1 Test:</u>

```
Cartesian-Polar 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(.): 5.0

Enter the value of Y using a decimal point(.): 30.0


You entered:

Stored as Cartesian  (5.0,30.0)



After asking to store as Cartesian:

Stored as Cartesian  (5.0,30.0)



After asking to store as Polar:

Stored as Polar [30.4138126514911,80.53767779197439]
```

<u>Design 3 test:</u>

```
Cartesian-Polar Coordinates Conversion Program

Enter the X Coordinate using a decimal point(.):5.0

Enter the Y Coordinate using a decimal point(.):24.0

Stored as Cartesian  (5.0,24.0)
```

```
Do you want to convert to polar (YES/NO):

yes
```

After asking to convert to Polar:

```
Converted to  (24.515301344262525,78.23171106797936)
```

Design 5 Test:

```
Cartesian-Polar 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/NO):

yes
```

After asking to convert to Cartesian:

```
Converted to  (49.23035293872719,63.087952012735805)
```

Output Generated by the Performance Test:

```
Time for getting Cartesian Point  in design 1: 101millis
Time for getting Cartesian Point  in design 3: 29millis
Time for getting Cartesian Point  in design 5: 22millis
Time for getting Polar Point  in design 1: 24millis
Time for getting Polar Point  in design 3: 91millis
```

Time for getting Polar Point in design 5: 20millis

Time for instantiating a cartesian point PointCP in design 1: 13millis

Time for instantiating a polar point PointCP in design 1: 8millis

Time for instantiating a cartesian point in PointCP in design 3: 11millis

Time for instantiating a cartesian point in PointCP in design 5: 18millis

Time for instantiating a polar point in PointCP in design 5: 12millis

E 26.

Design Number	Advantages	Disadvantages
1	- Stores the entered coordinates in Polar and Cartesian coordinates	- Has to convert between Polar and Cartesian depending on the input
3	- Small number of lines of code - Efficient in returning Cartesian coordinates - Only stores Cartesian, saving memory	- Takes longer to compute Polar coordinates -
5	- Only stores one type of coordinate, saving memory -	-

E 28 & 29.

	Design 1	Design 3	Design 5
getting Cartesian Point.	31 (16/32)	31(16/32)	31(15/47)
getting Polar Point	31(15/32)	47(46/63)	16(15/46)
instantiating Cartesian Coord.	31(15/32)	16(15/31)	31(16/32)
instantiating polar Coord.	16(15/32)		16(15/32)

A test class was implemented for each of these designs in order to test if each design was functioning properly. After checking the design, we ran a performance test in order to measure which of the designs was fastest. This class created a large number of instances to check the speed of each method as well as the initialization of the points.

E 30.

After running our tests, the results we received confirmed our hypothesis that design 5 would be the most time efficient of the 3 designs implemented. Design 1 had the ability to accept coordinates of either Polar or Cartesian providing it with a steady initialization time for both types of coordinates. The time recorded for design 1 initialization was almost an identical initialization time between both cartesian and polar coordinates. However, when compared with Design 3, which only accepted cartesian points, design 1 was significantly slower for initializing cartesian points as expected.

Design 3 has the advantage over design 1 in the initialization of cartesian points since design 3 can only accept cartesian points to be stores. However, the big drawback of design 3 compared to design 1, was that design 3 could not instantiate a polar point. Design 3 could only convert a cartesian point to a polar point making retrieving a polar point very time consuming, compared to design 1 which was able to instantiate a polar point in quite a reasonable amount of time.

Design 5 was the the most efficient design and easily beat out design 1 and 3. Design 5 could accept both cartesian and polar coordinates and initialize them in a very quick amount of time. By having both types of coordinates being subclasses of an abstract superclass, storing and choosing which coordinate to initialize was much faster than design 1, which had to store only one type of coordinate while waiting for the flag to say which type it was, and then having only one pair of instance variables initialize which point to create.