Names: Aisha Khalid(300009084) & Yusra Adinoyi(300018525)

Assignment 1

Java Files Included:

Design 2: PointCPD2.java

Design 3: PointCPD3.java

Design 6: PointD6.java (as the interface), **PointPolarD6.java, CartesianD6**

File to analyse the performance of all methods: PointAnalyseTest.java

This file took over 10 seconds to run. Running this file will output the time it takes for each method of each design to run. This was implemented by creating 3 additional static classes in the same file, one to test each design.

In each class, an array holding many points to be tested is created We have used 5,000,000 points for each test. The array is then tested by 4 methods in the class called testGetters, testDistance, testRotate, and testConversion that print the results.

Note: to test the interface, all points were of type PointD6, however the concrete type of half the points was PointPolarD6, and the other half was of type CartesianD6.

The sample output of one run of the program is included below.

File to test the 3 designs work: PointCPTestQ4.java

Running this file will prompt the user for the type of point it wants to create. It will then run tests on methods to ensure that the designs work. The output generated from 3 tests is included below.

Part two vector/arrays/arraylist file: PartTwo.java

Running this file will print the results.

Names: Aisha Khalid(300009084) & Yusra Adinoyi(300018525)

Part 1: Point classes

Question 4: Sample Ouput: “PointCPTestQ4.java”

```
Design Checker Program
Enter the number for the design you want to test: 2, 3 or 6: 2
Enter the value of Rho using a decimal point(.): 50
Enter the value of Theta using a decimal point(.): 0

You entered:
Stored as Polar: [50.0,0.0]
Computed Cartesian coords [50.0,0.0]

After rotating 142 degrees
Stored as Polar: [50.0,-51.999999999999986]
Computed Cartesian coords [30.783073766282925,-39.400537680336086]

Distance between rotated point and the original: 43.837114678907724

 After converting orignal to Cartesian: Cartesian Point: (50.0, 0.0)
 Computed Polar coord: [50.0, 0.0]
PS C:\Users\aisha\Desktop\SEG2105\Labs\Lab2\Assignment\src\lab1> java PointCPTestQ4
Design Checker Program
Enter the number for the design you want to test: 2, 3 or 6: 3
Enter the value of X using a decimal point(.): 50
Enter the value of Y using a decimal point(.): 50

You entered:
Cartesian Point: (50.0, 50.0)
 Computed Polar coord: [70.71067811865476, 45.0]

After rotating 273 degrees
Cartesian Point: (52.54827454987585, -47.31467892558154)
 Computed Polar coord: [70.71067811865476, -42.00000000000005]

Distance between rotated point and the original: 97.34803766676845

 After converting orignal to Polar: Stored as Polar: [70.71067811865476,45.0]
Computed Cartesian coords [50.00000000000001,50.0]
PS C:\Users\aisha\Desktop\SEG2105\Labs\Lab2\Assignment\src\lab1> java PointCPTestQ4
Design Checker Program
Enter the number for the design you want to test: 2, 3 or 6: 6
Enter the value of Rho using a decimal point(.): 100
Enter the value of Theta using a decimal point(.): 90

You entered:
Stored as Polar: [100.0,90.0]
Computed Cartesian coords [6.123233995736766E-15,100.0]

After rotating 222 degrees
Stored as Polar: [100.00000000000001,138.00000000000003]
Computed Cartesian coords [-74.31448254773944,66.91306063588581]

Distance between rotated point and the original: 81.34732861516007

 After converting orignal to Cartesian: Cartesian Point: (6.123233995736766E-15, 100.0)
 Computed Polar coord: [100.090.0]
PS C:\Users\aisha\Desktop\SEG2105\Labs\Lab2\Assignment\src\lab1
```

Names: Aisha Khalid(300009084) & Yusra Adinoyi(300018525)

Question 6:Sample Output: "PointAnalyseTest.java"

```
Testing Version 2...

time to getX of 5000000 points: 583322.478 milliseconds.

time to getY of 5000000 points: 537510.964 milliseconds.

time to getRho of 5000000 points: 13132.641 milliseconds.

time to getTheta of 5000000 points: 12704.087 milliseconds.

time to do Distance of 5000000 points: 1717196.054 milliseconds.

time to do rotate on 5000000 points: 3855089.734 milliseconds.

time to do conversion on 5000000 points: 1203257.089 milliseconds.


Testing Version 3...

time to getX of 5000000 points: 7407.005 milliseconds.

time to getY of 5000000 points: 12680.299 milliseconds.

time to getRho of 5000000 points: 14543.661 milliseconds.

time to getTheta of 5000000 points: 625419.728 milliseconds.

time to do Distance of 5000000 points: 14778.517 milliseconds.

time to do rotate on 5000000 points: 1423069.59 milliseconds.

time to do conversion on 5000000 points: 684914.774 milliseconds.


Testing Version 6...

time to getX of 5000000 points: 427484.643 milliseconds.

time to getY of 5000000 points: 398742.051 milliseconds.

time to getRho of 5000000 points: 35141.455 milliseconds.

time to getTheta of 5000000 points: 367403.969 milliseconds.

time to do Distance of 5000000 points: 919500.506 milliseconds.

time to do rotate on 5000000 points: 2439311.987 milliseconds.

time to do conversion to opposite type on 5000000 points: 17306.043 milliseconds.
```

Names: Aisha Khalid(300009084) & Yusra Adinoyi(300018525)

E.26 Hypothesis for Advantages and Disadvantages of each design

Design	**Simplicity of Code**	**Creating instances**	**Computations' Efficiency**	**Memory used**
2 (PointPolar)	Just as simple as D3	Same as D3	getX, getY, and getDistance will be slower as more conversions must be done. getRho, and getTheta are faster as they are stored values.	Same as D3
3 (PointCoord)	Jut as simple as D3	Same as D2	getX, getY, and getDistance methods will be faster as X and Y coordinates are already stores. getRho and getTheta are slower as converesions must be done	Same as D2
6 (interface +Designs 2,3)	Will require an extra class, the interface to be created	Can create as interface type, then intialize as concrete type	Average efficiency for all methods should be in between design 2 and design 3, as it depends on what concrete class was used.	Same or a bit more due to implementing interface

For the following exercises we created 5 000 000 instances of each class and tested each method:

E.28

Magnitude of the differences in efficiency of designs can be seen from the table made for E30, the conclusions for each method are:

GetX and getY: Was approximately 30-40 times faster than both design 3 and 6

GetRho: Design 6 was approximately 5 times slower than both design 2 and 3 which were comparable

getTheta: Design 2 was approx. 50 times faster than design 3, design 6 was in the middle

getDistance: Design 3 approximately 13 times faster than design 2, design 6 was in the middle

rotatePoint: Design 3 was approx. 4 times faster than design 2, and second fastest on design 6

Converting to the other type: fastest on design 6 with designs 2 and 3 being comparable

Comparison to hypotheses from E.26: All the hypothesis for efficiency were correct, except converting to the opposite type was faster for design 6. Also, this analysis leads to the conclusion that for overall performance, design 3 would be the most efficient.

Names: Aisha Khalid(300009084) & Yusra Adinoyi(300018525)

E.29

Performance analysis on all methods ran using PointAnalyseTest.java. Results are summarized in E.30

E.30

Average Computations speeds for the operations on different designs in milliseconds

Operations	Design 2			Design 3			Design 6		
	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg
getX()	488446	450113	472360	10297	7122	7298	615097	304282	427262
getY()	538501	388100	414600	22909	9215	10812	337502	308001	319133
getRho()	16663	8998	14045	19809	12255	15418	127161	39847	72336
getTheta()	15386	10099	13890	822294	665832	723651	502745	435635	462982
getDistance()	1840178	1721042	1756519	19338	10370	13667	1184053	1131692	1163763
rotatePoint()	4465855	3940196	4159307	2061578	1577395	1699163	3078761	2922002	2991639
Converting to Opposite	1088670	817011	925711	858905	666803	804557	25556	19921	22772

Part 2: Vector/ArraysLists/Arrays

Time	ArrayList	Vector	Array
Test 1	0.122947988	3.260702445	0.079015661
Test 2	0.175345685	3.750741664	0.095158389
Test 3	0.143885242	3.377636697	0.091348591
Average	0.147392972	3.463027026	0.088507320

The table displays the time taken in seconds for each collection to sum up all the integers stored. We chose to test this collection with 80 000 000 randomly generated integers, ranging from 0-9. To do so, a vector and arraylist of unspecified size were created(therefore they must increase their size automatically as more integers are added) as well as an array(with the final size) and filled with the numbers, before being summed up in a for loop seperately.

From these results, we recommend that developers always use arrays when the size is known as the array was approximately 1.67 time faster than the arraylist. If the size is unknown, then we would recommend arrayList over vectors for performance as the vector was approximately 23 times slower than the arraylist.