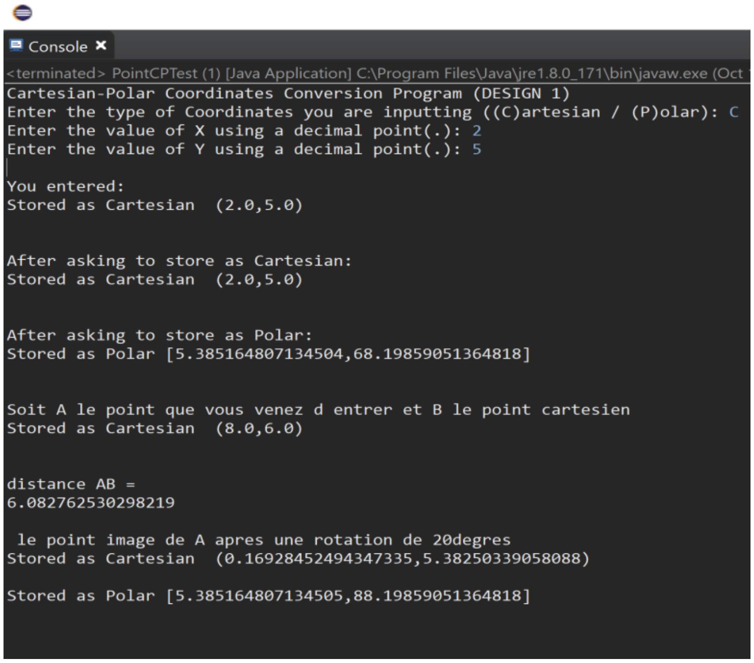
ASSIGNMENT 1:

Table of Contents

Design test compilations

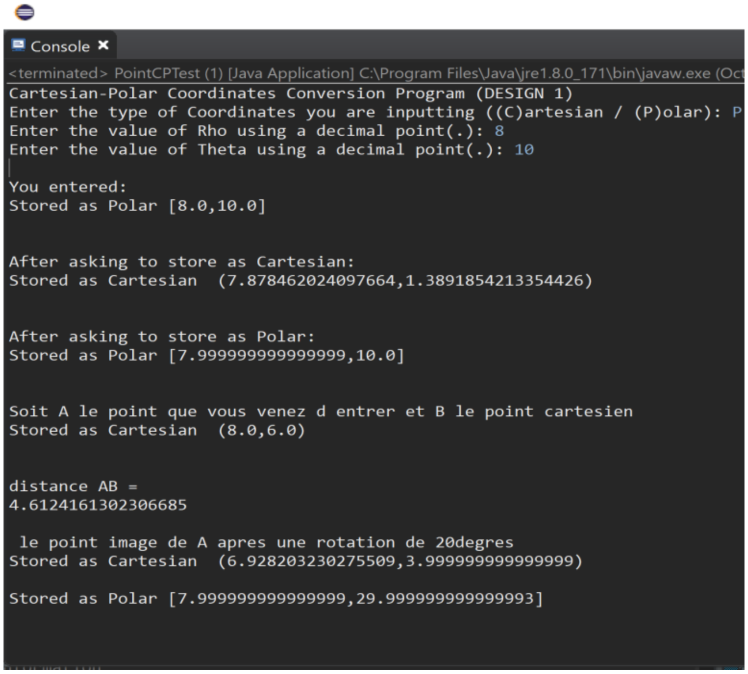
Test design 1

- With Cartesian coordinates



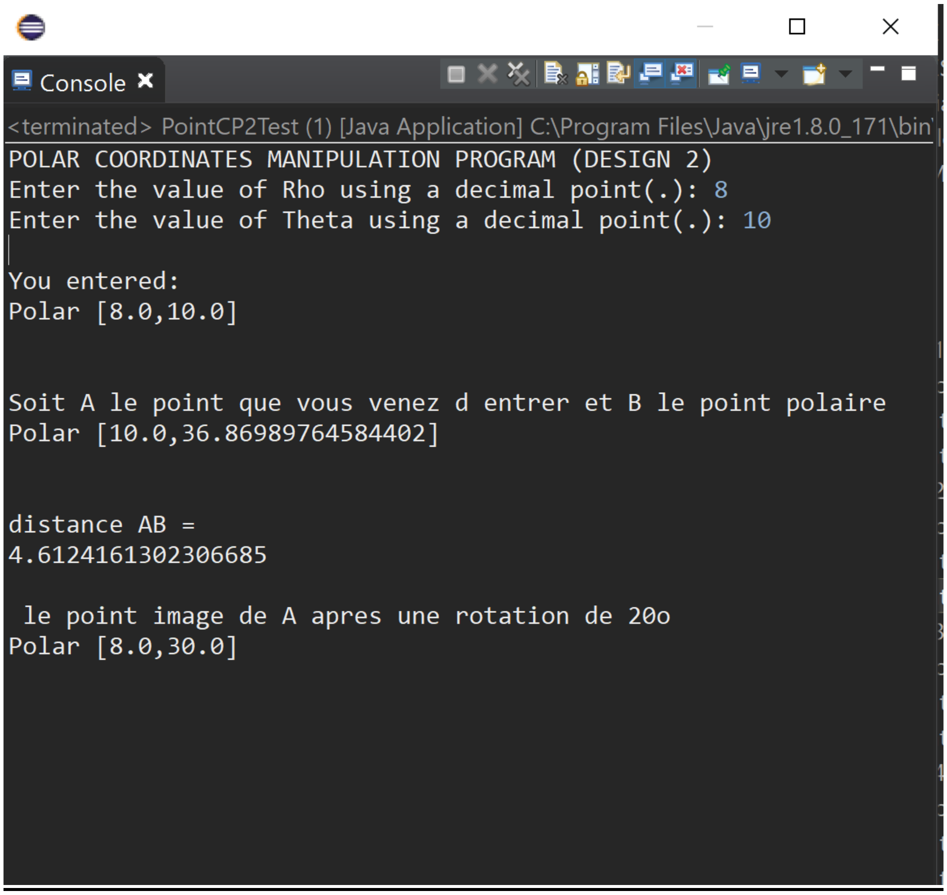
```
<terminated> PointCPTest (1) [Java Application] C:\Program Files\Java\jre1.8.0_171\bin\javaw.exe (Oct
Cartesian-Polar Coordinates Conversion Program (DESIGN 1)
Enter the type of Coordinates you are inputting ((C)artesian / (P)olar): C
Enter the value of X using a decimal point(.): 2
Enter the value of Y using a decimal point(.): 5

You entered:
Stored as Cartesian  (2.0,5.0)


After asking to store as Cartesian:
Stored as Cartesian  (2.0,5.0)


After asking to store as Polar:
Stored as Polar [5.385164807134504,68.19859051364818]


Soit A le point que vous venez d entrer et B le point cartesien
Stored as Cartesian  (8.0,6.0)


distance AB =
6.082762530298219

 le point image de A apres une rotation de 20degres
Stored as Cartesian  (0.16928452494347335,5.38250339058088)

Stored as Polar [5.385164807134505,88.19859051364818]
```

- With Polar coordinates



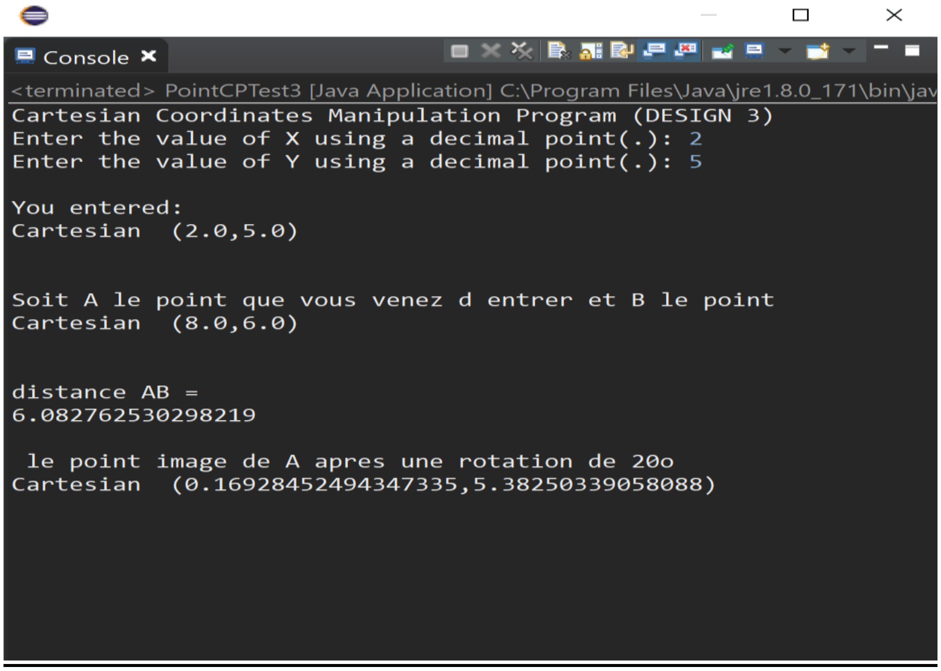
```
<terminated> PointCPTest (1) [Java Application] C:\Program Files\Java\jre1.8.0_171\bin\javaw.exe (Oct
Cartesian-Polar Coordinates Conversion Program (DESIGN 1)
Enter the type of Coordinates you are inputting ((C)artesian / (P)olar): P
Enter the value of Rho using a decimal point(.): 8
Enter the value of Theta using a decimal point(.): 10

You entered:
Stored as Polar [8.0,10.0]


After asking to store as Cartesian:
Stored as Cartesian  (7.878462024097664,1.3891854213354426)


After asking to store as Polar:
Stored as Polar [7.999999999999999,10.0]


Soit A le point que vous venez d entrer et B le point cartesien
Stored as Cartesian  (8.0,6.0)


distance AB =
4.6124161302306685

 le point image de A apres une rotation de 20degres
Stored as Cartesian  (6.928203230275509,3.999999999999999)

Stored as Polar [7.999999999999999,29.999999999999993]
```

Test design 2



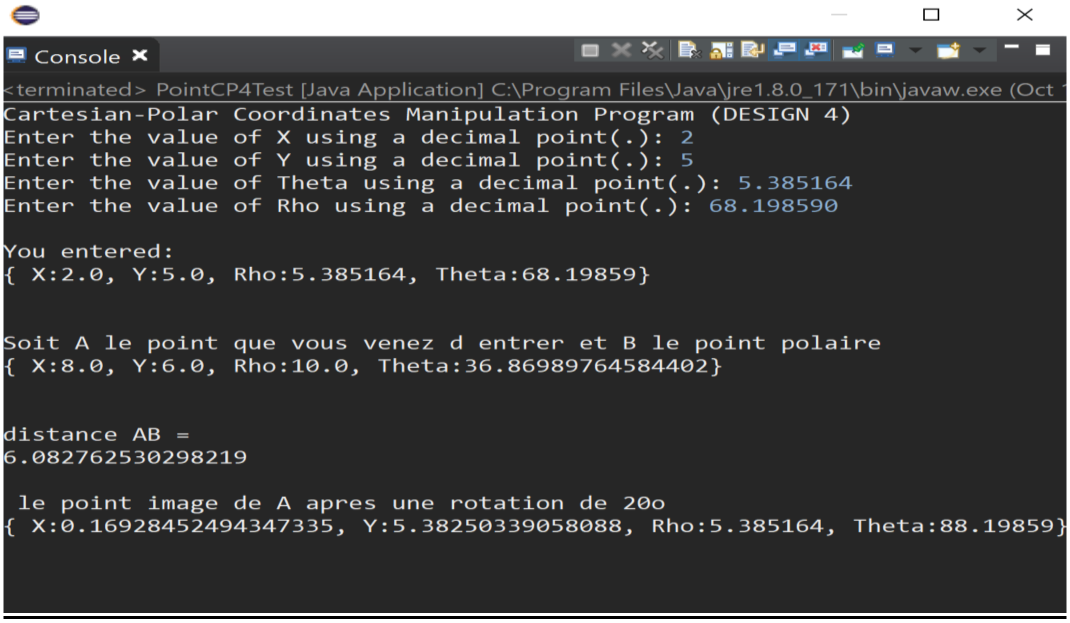
```
<terminated> PointCP2Test (1) [Java Application] C:\Program Files\Java\jre1.8.0_171\bin
POLAR COORDINATES MANIPULATION PROGRAM (DESIGN 2)
Enter the value of Rho using a decimal point(.): 8
Enter the value of Theta using a decimal point(.): 10

You entered:
Polar [8.0,10.0]


Soit A le point que vous venez d entrer et B le point polaire
Polar [10.0,36.86989764584402]


distance AB =
4.6124161302306685

 le point image de A apres une rotation de 20o
Polar [8.0,30.0]
```

Test design 3



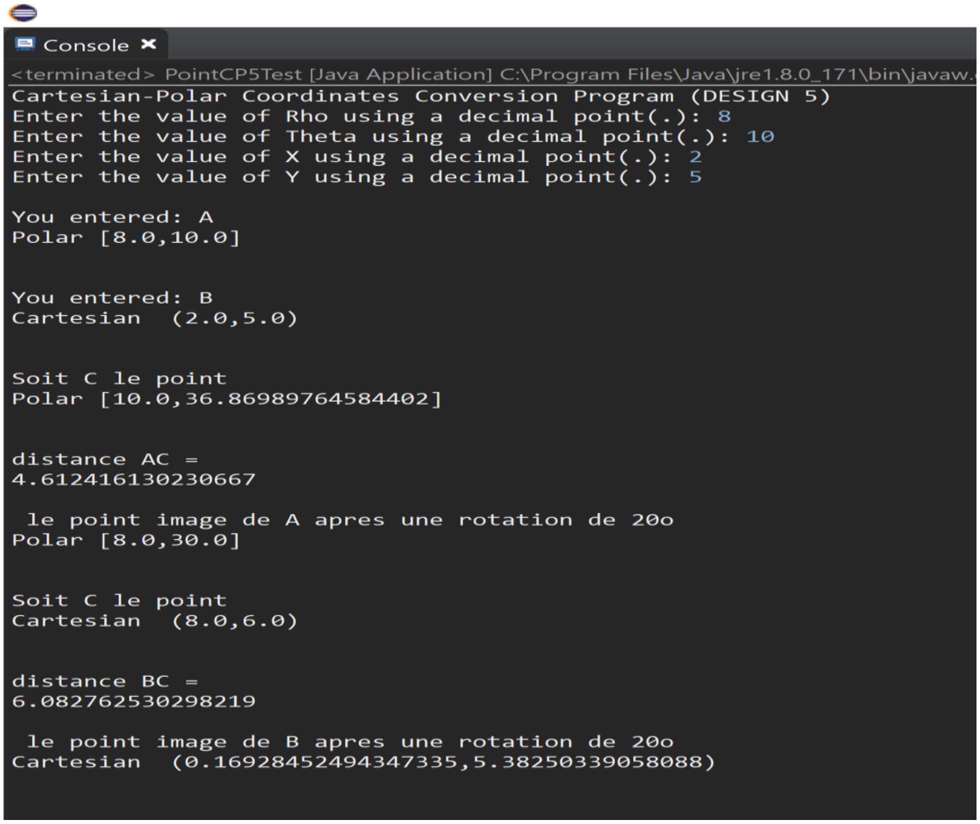
```
<terminated> PointCPTest3 [Java Application] C:\Program Files\Java\jre1.8.0_171\bin\jav
Cartesian Coordinates Manipulation Program (DESIGN 3)
Enter the value of X using a decimal point(.): 2
Enter the value of Y using a decimal point(.): 5

You entered:
Cartesian  (2.0,5.0)


Soit A le point que vous venez d entrer et B le point
Cartesian  (8.0,6.0)


distance AB =
6.082762530298219

 le point image de A apres une rotation de 20o
Cartesian  (0.16928452494347335,5.38250339058088)
```

Test design 4



```
<terminated> PointCP4Test [Java Application] C:\Program Files\Java\jre1.8.0_171\bin\javaw.exe (Oct
Cartesian-Polar Coordinates Manipulation Program (DESIGN 4)
Enter the value of X using a decimal point(.): 2
Enter the value of Y using a decimal point(.): 5
Enter the value of Theta using a decimal point(.): 5.385164
Enter the value of Rho using a decimal point(.): 68.198590

You entered:
{ X:2.0, Y:5.0, Rho:5.385164, Theta:68.19859}


Soit A le point que vous venez d entrer et B le point polaire
{ X:8.0, Y:6.0, Rho:10.0, Theta:36.86989764584402}


distance AB =
6.082762530298219

 le point image de A apres une rotation de 20o
{ X:0.16928452494347335, Y:5.38250339058088, Rho:5.385164, Theta:88.19859}
```

Test design 5



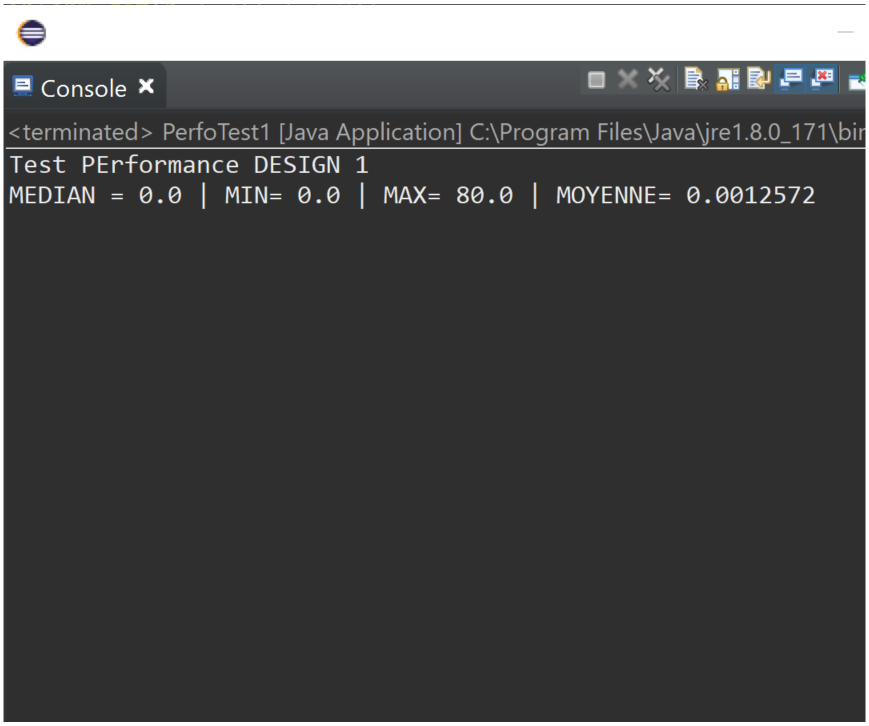
```
<terminated> PointCP5Test [Java Application] C:\Program Files\Java\jre1.8.0_171\bin\javaw.
Cartesian-Polar Coordinates Conversion Program (DESIGN 5)
Enter the value of Rho using a decimal point(.): 8
Enter the value of Theta using a decimal point(.): 10
Enter the value of X using a decimal point(.): 2
Enter the value of Y using a decimal point(.): 5

You entered: A
Polar [8.0,10.0]


You entered: B
Cartesian  (2.0,5.0)


Soit C le point
Polar [10.0,36.86989764584402]


distance AC =
4.612416130230667

 le point image de A apres une rotation de 20o
Polar [8.0,30.0]


Soit C le point
Cartesian  (8.0,6.0)


distance BC =
6.082762530298219

 le point image de B apres une rotation de 20o
Cartesian  (0.16928452494347335,5.38250339058088)
```

E26

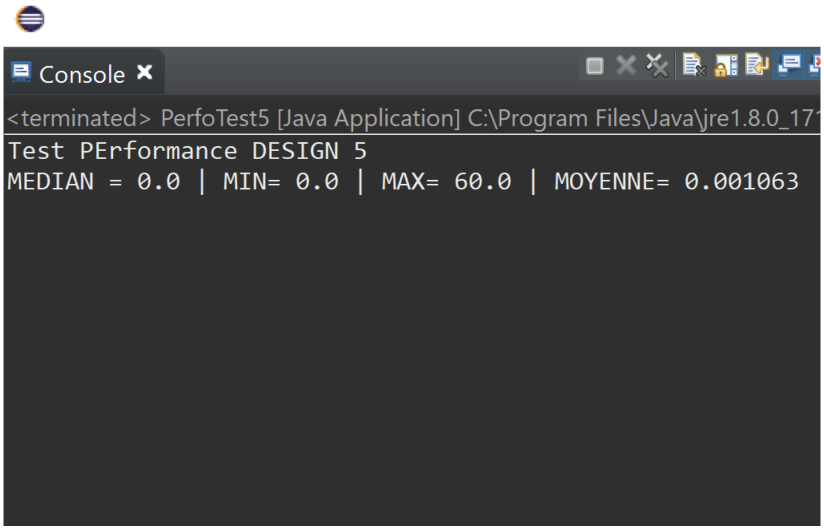
	ADVANTAGES			**DISADVANTAGES**		
	Simplicity of code	**Efficiency when creating instances**	**Efficiency during calculations**	**Simplicity of code**	**Efficiency when creating instances**	**Efficiency during calculations**
Design1	- the code for managing the 2 types of coordinates is done in a single class	- uses less memory space than design 4 for instance creation	- the calculations return correct results according to the type entered	- Long and heavy code. -Too many "if … else"	-uses more space in memory than designs 2,3 and 5 for the creation of instances because it has 3 attributes.	-always does a type check operation before doing the calculations

Design2	-the operation of calculating the image point after a rotation is much simpler with rho and theta	Does not use a lot of memory space for the creation of a new instance because this design has only 2 attributes.	The calculations are optimal for polar type coordinates.	Simple but not optimal code	Nothing to report	No calculations for Cartesian type coordinate points.
Design3	- much simpler point distance calculation operation with the use of X and Y.	Does not use a lot of memory space for the creation of a new instance because this design has only 2 attributes.	The calculations are optimal for coordinates of Cartesian type.	Simple but not optimal code	Nothing to report	No calculations for Polar type coordinate points.
Design4	Extremely simple code as there is no conversion of rho and theta or X and Y as they are already given	I don't see any benefit for this section	The calculations are quite simple given that all the attributes (X, Y, rho and theta) are only returned	The code is simple but not optimal because errors can creep in when entering polar and cartesian type attributes.	Uses a lot of memory space for creating a new instance because this design has 4 attributes	Will not always be correct because the user can enter polar and Cartesian coordinates which are not necessarily equivalent.
Design5	- Simpler and shorter code. -No code duplication -There is a possibility of code reuse -possibility to add other types of point.	Does not use a lot of memory space for creating a new instance because this design has only 2 attributes.	The calculations will always return the exact answers and require less calculation effort because each class calculates according to its attributes	Doesn't have any downsides in my opinion.	To change the type of variable, you must necessarily create a new instance instead of making the change directly in the existing variable	Does not encounter any disadvantages on this plan

E28 & E29



```
<terminated> PerfoTest1 [Java Application] C:\Program Files\Java\jre1.8.0_171\bir
Test PErformance DESIGN 1
MEDIAN = 0.0 | MIN= 0.0 | MAX= 80.0 | MOYENNE= 0.0012572
```



```
<terminated> PerfoTest5 [Java Application] C:\Program Files\Java\jre1.8.0_17
Test PErformance DESIGN 5
MEDIAN = 0.0 | MIN= 0.0 | MAX= 60.0 | MOYENNE= 0.001063
```

Answer:

For a performance test that runs 5,000,000 times. We can see **that design 5 takes 60 milliseconds maximum and 1.063x10-3 ms on average and design 1 takes 80 milliseconds maximum and 1.2572063x10-3 ms on average.** We can therefore conclude with this information that Design 5 is faster in terms of execution than Design 1 and this confirms the assumptions made in question E26. But I realized that the results could often vary and be biased depending on the Garbage collector.

Explanation on the creation of my performance test..

Step 1: creation of the algorithm which will create the variables with random data and then test all the functions of the classes. This was done in a function called ***algo().***

Step 2: creation of the function ***test(int X)***: which will execute the algo function X times (in the case of our tests X = 5,000,000), calculate the average execution time, find the median, the minimum and the maximum.

Step 3: call the function ***test(int X)*** in the main.

E30

	Design 1	**Design 5**
Création Variable	MEDIAN = 0.0 | MIN= 0.0 | MAX= 73.0 | MOYENNE= 1.166E-4	MEDIAN = 0.0 | MIN= 0.0 | MAX= 26.0 | MOYENNE= 1.217E-4
getX()	MEDIAN = 0.0 | MIN= 0.0 | MAX= 1.0 | MOYENNE= 5.1E-6	MEDIAN = 0.0 | MIN= 0.0 | MAX= 1.0 | MOYENNE= 5.4E-6
getY()	MEDIAN = 0.0 | MIN= 0.0 | MAX= 1.0 | MOYENNE= 6.3E-6	MEDIAN = 0.0 | MIN= 0.0 | MAX= 1.0 | MOYENNE= 4.8E-6
getRho()	MEDIAN = 0.0 | MIN= 0.0 | MAX= 21.0 | MOYENNE= 5.5E-6	MEDIAN = 0.0 | MIN= 0.0 | MAX= 3.0 | MOYENNE= 6.4E-6
getTheta()	MEDIAN = 0.0 | MIN= 0.0 | MAX= 1.0 | MOYENNE= 3.33E-5	MEDIAN = 0.0 | MIN= 0.0 | MAX= 3.0 | MOYENNE= 3.78E-5
convertStorageToCartesian ()	MEDIAN = 0.0 | MIN= 0.0 | MAX= 2.0 | MOYENNE= 3.46E-5	MEDIAN = 0.0 | MIN= 0.0 | MAX= 3.0 | MOYENNE= 4.15E-5
convertStorageToPolar ()	MEDIAN = 0.0 | MIN= 0.0 | MAX= 3.0 | MOYENNE= 5.77E-5	MEDIAN = 0.0 | MIN= 0.0 | MAX= 2.0 | MOYENNE= 3.84E-5
getDistance(PointCP)	MEDIAN = 0.0 | MIN= 0.0 | MAX= 40.0 | MOYENNE= 1.382E-4	MEDIAN = 0.0 | MIN= 0.0 | MAX= 2.0 | MOYENNE= 3.67E-5
rotatePoint(double)	MEDIAN = 0.0 | MIN= 0.0 | MAX= 77.0 | MOYENNE= 2.038E-4	MEDIAN = 0.0 | MIN= 0.0 | MAX= 2.0 | MOYENNE= 1.28E-4