Pierre Georges
300283706

SEG 2105 [Z]
(Assignment 1)
Professor Wassim
El Ahmar

Nalan Kurnaz
300245521

Writing Section of the assignment

Q5

E28 Run a performance analysis in which you compare the performance of Design 5, as you implemented it in the previous exercise, with Design 1. Determine the magnitude of the differences in efficiency, and verify the hypotheses you developed in E26.

The test the program 5 times and these are the results of the 5 attempts.
Attempt 1(Result):
Design 1 Execution Time: 109 milliseconds. Design 5 Execution Time: 81 milliseconds
Design 5 is faster than Design 1. The difference in efficiency is approximately 28 milliseconds. Process finished with exit code 0.
Attempt 2(Result):
Design 1 Execution Time: 127 milliseconds. Design 5 Execution Time: 69 milliseconds. Design 5 is faster than Design 1. The difference in efficiency is approximately 58 milliseconds. Process finished with exit code 0.
Attempt 3(Result):
Design 1 Execution Time: 141 milliseconds. Design 5 Execution Time: 80 milliseconds. Design 5 is faster than Design 1. The difference in efficiency is approximately 61 milliseconds. Process finished with exit code 0.
Attempt 4(Result):
Design 1 Execution Time: 125 milliseconds. Design 5 Execution Time: 64 milliseconds. Design 5 is faster than Design 1. The difference in efficiency is approximately 61 milliseconds. Process finished with exit code 0.
Attempt 5(Result):
Design 1 Execution Time: 119 milliseconds. Design 5 Execution Time: 64 milliseconds. Design 5 is faster than Design 1. The difference in efficiency is approximately 55 milliseconds. Process finished with exit code 0.
The effect causes design 5 to be slower and having the average of 52.6 milliseconds between the design 5 and design 1.

E29 To run a performance analysis, you will have to create a new test class that randomly generates large numbers of instances of PointCP, and performs operations on them, such as retrieving polar and Cartesian coordinates. You should then run this test class with the two versions of PointCP – Design 2 and Design 5.

The test the program 5 times and these are the results of the 5 attempts.
Attempt 1(Result):
Design 2 Execution Time: 101 milliseconds. Design 5 Execution Time: 71 milliseconds. Design 5 is faster than Design 2. The difference in efficiency is approximately 30 milliseconds. Process finished with exit code 0
Attempt 2(Result):

Design 2 Execution Time: 93 milliseconds. Design 5 Execution Time: 63 milliseconds. Design 5 is faster than Design 2. The difference in efficiency is approximately 30 milliseconds. Process finished with exit code 0

Attempt 3(Result):

Design 2 Execution Time: 201 milliseconds. Design 5 Execution Time: 82 milliseconds. Design 5 is faster than Design 2. The difference in efficiency is approximately 119 milliseconds. Process finished with exit code 0

Attempt 4(Result):

Design 2 Execution Time: 105 milliseconds. Design 5 Execution Time: 72 milliseconds. Design 5 is faster than Design 2. The difference in efficiency is approximately 33 milliseconds. Process finished with exit code 0

Attempt 5(Result):

Design 2 Execution Time: 127 milliseconds. Design 5 Execution Time: 97 milliseconds

Design 5 is faster than Design 2. The difference in efficiency is approximately 30 milliseconds. Process finished with exit code 0

We conclude that Design 5 is faster than Design 2.

E30 Summarise your results in a table: the columns of the table would be the two designs; the rows of the table would be the operations. The values reported in the table would be the average computation speed. Make sure you explain your results

	Big polar coordinates (98765432.98765432,98765432.98765432)	Small polar coordinates (11.11,11.11)	Big cartesian coordinates (98765432.98765432,98765432.98765432)	Small cartesian coordinates (11.11,11.11)
Design 1	5485	2408	5391	2879
Design 5	5546	2345	5453	2852

As you could see that Design 5 is slower then Design 1 but since it has a n_0 very far. You could see how this causes it to seem faster.

Q6

How did we do our testing?

Time testing:

When testing the time when adding a long element coles start in the starting area which is assigned to be the system's current time in millisecond. We also add another one at the end of the code that we need to end. We wrap it up with essay maths of end minis and start to see the time spent between it.

When testing between files to see the time speed per each, find the fastest and the time difference between them. We mostly take the coding program away for the git to be put in a different fill and we do the same step as shown in the top by in these cases the test case is random with data provided immediately pay the test by creating a random number generator and testing the cartizain and the polar coordinates. The time is reserved instead of immediately printed and you do have two starts and two ends

since there are two different timers. For the calculation of the timer. Since we could easily find which one is smaller and which one is bigger. Whe must do bigger minus smaller to showcase the time difference between them.

General testing:

For the general testing part, after we coded designs and understood what each design required, we first started by copying implementations in PointCP.java because it was assumed that it would be much easier to change and add the code rather than writing the whole thing from scratch. Since design2 required storing polar coordinates, depending on the input of the user, the information was converted to polar coordinates and it was stored. The same applied to Design 3; however, Design 5, should be aware of the fact that Design 5 consisted of an abstract class with 2 subclasses and exceptions as well. Testing of design 5 was done in a way that would compute both polar and cartesian coordinates.

Sample outputs from running the tests:

```
C:\Users\16132\OneDrive - University of Ottawa\Desktop\design5>java PointCP2Test
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(.): 9.0
Enter the value of Y using a decimal point(.): 22.5

You entered:
Stored as Polar [24.23324163210527,68.19859051364818]


After asking to store as Cartesian:
Stored as Polar [24.23324163210527,68.19859051364818]
```

Figure 1: Sample 1 of PointCP2Test

```
Enter the type of Coordinates you are inputting ((C)artesian / (P)olar): P
Enter the value of Rho using a decimal point(.): 23.9
Enter the value of Theta using a decimal point(.): 24.6
Origin point: Stored as Polar [24.23324163210527,68.19859051364818]

New point: Stored as Polar [23.9,24.6]

The distance between two point is 17.877265837143263
Enter the angle in degrees for rotation:
45
```

Figure 2: Sample 2 of PointCP2Test

```
C:\Users\16132\OneDrive - University of Ottawa\Desktop\design5>java PointCP2Test
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(.): 899999994
Enter the value of Y using a decimal point(.): 24555678

You entered:
Stored as Polar [9.003349213054216E8,1.5628752886455657]


After asking to store as Cartesian:
Stored as Polar [9.003349213054216E8,1.5628752886455657]
```

Figure 3: Sample 3 of PointCP2Test

```
Enter the type of Coordinates you are inputting ((C)artesian / (P)olar): P
Enter the value of Rho using a decimal point(.): 25.7
Enter the value of Theta using a decimal point(.): 3.56
Origin point: Stored as Cartesian  (9.5,3.5)
New point: Stored as Cartesian  (25.650407256481,1.5958093798029236)
The distance between two point is 16.262275254964806
Enter the angle in degrees for rotation:
45
after rotation, the point is Stored as Cartesian  (9.5,3.5)
```

Figure 4: Sample 1 of PointCP3Test

```
Enter the type of Coordinates you are inputting ((C)artesian / (P)olar): C
Enter the value of X using a decimal point(.): 2.7
Enter the value of Y using a decimal point(.): 4.7
Origin point: Stored as Cartesian  (3.5,3.6)
New point: Stored as Cartesian  (2.7,4.7)
The distance between two point is 1.3601470508735443
Enter the angle in degrees for rotation:
45
after rotation, the point is Stored as Cartesian  (3.5,3.6)
```

Figure 5: Sample 2 of PointCP3Test

```
Enter the type of Coordinates you are inputting ((C)artesian / (P)olar): C
Enter the value of X using a decimal point(.): 5.7
Enter the value of Y using a decimal point(.): 2.9999956
Origin point: Stored as Cartesian  (2.497426424476787,0.11340747032313445)
New point: Stored as Cartesian  (5.7,2.9999956)
The distance between two point is 4.311481014341931
Enter the angle in degrees for rotation:
39
after rotation, the point is Stored as Cartesian  (2.497426424476787,0.11340747032313445)
```

Figure 6: Sample 3 of PointCP3Test

```
C:\Users\16132\OneDrive - University of Ottawa\Desktop\design5>java PointCP5Test
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(.): 999999.4059
Enter the value of Y using a decimal point(.): 394444.5790

You entered:
Stored as Polar [1074981.5522616378,21.526498123348247]

After asking to store as Cartesian: Stored as Polar [1074981.5522616378,21.526498123348247]

Now enter another point to compute the distance:
```

Figure 7: Sample 1 of PointCP5Test

```
Enter the type of Coordinates you are inputting ((C)artesian / (P)olar): P
Enter the value of Rho using a decimal point(.): 67.99
Enter the value of Theta using a decimal point(.): 45.900077777
Origin point: Stored as Polar [1074981.5522616378,21.526498123348247]

New point: Stored as Polar [67.99,45.900077777]

The distance between two point is 1074919.6223008037
```

Figure 8: Sample 2 of PointCP5Test

```
C:\Users\16132\OneDrive - University of Ottawa\Desktop\design5>java PointCP5Test
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(.): 99999947585
Enter the value of Y using a decimal point(.): 23956483.8847733

You entered:
Stored as Polar [9.999995045456706E10,0.013726061117636058]

After asking to store as Cartesian: Stored as Polar [9.999995045456706E10,0.013726061117636058]
```

Figure 9: Sample 3 of PointCP5Test

the table

Design number	Advantage	Disadvantage
Design 2	Polar coordinates are helpful for analysing how the human body moves. This is due to the pivotal joint motions used by the human body. Polar coordinates make things simpler when considering orbits when radial symmetry is involved or there is a point source, such as the production of water ripples. It helps with object placement and may be used to make an item circle another object. Simple implementation.	issues with counterbalance. restricted avoidance of collisions. In these systems, positional inaccuracy is significant. not very efficient.
Design 3	Calculations of distances between points are trivial. Calculations of areas are relatively easy. Graphic representations are realistic, provided the area covered is not too large. Implementation is easy. efficiency is good as compared to design 2.	These systems are inadequate for handling complicated issues. Memory use rises in comparison to Design 2.

Design 5	Given that both components are integrated into the programme, it is easy to code. coding efficiency increases depending on the nature of the issue. Error finding is simple.	Memory use is higher than for designs 1, 2, and 3.

A discussion of the results

The main purpose of Assignment 1 was to adjust and design codes that produce different outcomes. The expected outcomes differ from each other. Design 1, Design 2 and Design 5 were done through PointCP.java. Design 2 expected to store polar coordinates and compute Cartesian coordinates whereas design 5 was an abstract class implementing both design 2 and design 3 as its subclasses.

Our initial hypothesis was the execution time of design 5 would be slower compared to design 1 and design 2. The reason why we assumed it to be slower is that the code consists of long and complex instructions, and it was also implementing the methods of design 1 and design 2. There were more lines of code to be executed. However, as seen in E28, we experimented that design 5 was far faster than both designs. For example, in Question 5, the execution time between Design 1 and Design 5 differed by 28 milliseconds in the first attempt, and the difference reached 58 milliseconds in the second attempt, meaning that the time that takes Design 5 to execute is smaller than the design 1. We reached the same result in comparing design 2 and design 5. The execution time difference peaked at 119 milliseconds, resulting in a faster performance and execution time than design 2.

This showed that our hypothesis was incorrect. The part that may have been missed out could be the dependability part. In other words, as the code is more dependable to other classes. The execution time gets slower. In design 2 and design 3, we coded so that they inherited from PointCP5. Design 2 and design 3 were depended on PointCP5 file to execute, and this could be the reason why they resulted in slower than design 5.

To conclude, the difference in execution time could be because of dependability of first two designs to last design, or it could be stemmed from an error in our code.