SEG2105 - Introduction to Software Engineering

Assignment - 1

Submitted by:

Raphaelle Jean-Baptiste, 300085552

Lucy Amos, 300232230

May 28th, 2023

University of Ottawa

Large-Language Models Disclosure

This document must be attached as the first page of any assignment you submit to this course.

Student Name: Lucy Amos, Raphaelle Jean Baptiste

Student Number: 300232230, 300085552

What large language models did you use to help you complete this assignment? (ChatGPT, Bard, etc.)

ChatGPT

Please provide the 5 prompts you found most useful for solving this assignment:

1. For design 5 (seen in table) and asked ChatGPT to reword the description so that we could understand what it was asking.
2. Asked ChatGPT the best way to implement a performance test for two classes in general.

What limitations did you identify using LLM for this assignment? How did you overcome them?

Without much context, ChatGPT did not understand what we meant for design 5, but it gave a description on concrete subclasses, it however, did not answer our question as we already understood that part, but we didn't necessarily understand what the assignment was looking for. To combat this, we asked our TA through email what they meant for design 5 as they were the ones that were grading it and they understood how design 5 should be properly implemented.

Gave an example of a performance test, but not exactly what we were looking for, but it gave us a hint on where to start. To finish off the assignment we referenced old labs and assignments from ITI1121 as a lot of it did deal with simulation and this sort of performance testing implementation.

Through your experience with using LLM to help you solve this assignment, what have you learned? How did this experience enhance your skills/thought process for future challenges?

We learned that ChatGPT is a context based language that needs a lot of information to be fully effective. We did not want to rely on or trust the software so we avoided using it too much, but we general liked hints that it gave us. This has allowed us to be mindful of the resources online as well as the resources from previous classes. It has also helped us with our problem solving skills as we took what we knew, tried to do research to make it better. It showed us there were better, and more appropriate places to find code that we owned to help us with the answers. In the future, we would like to use LLM to fill in the gaps of our knowledge. We want to enhance our skills, but we also want to optimize the time it takes us to solve these problems.

Part I: PointCP

E26 Create a table describing the various advantages (pros) and disadvantages (cons) of each of the five design alternatives. Some of the factors to consider are: simplicity of code, efficiency when creating instances, efficiency when doing computations that require both coordinate systems, and amount of memory used.

Table I. Alternative Designs and Hypotheses for PointCP Class.

	How Cartesian coordinates are computed	**How polar coordinates are computed**	**Advantages (pros)**	**Disadvantages (cons)**
Design 1: Store one type of coordinates using a single pair of instance variables, with a flag indicating which type is stored	Simply returned if Cartesian is the storage format, otherwise computed	Simply returned if polar is the storage format, otherwise computed	- Code encompassed a lot of different cases for each "type" - Simple straightforward implementation	- At times stores both variables taking up more time and more storage when both coordinated are needed to be stored - Computation of cartesian and polar coordinated less more efficient than other option since one can toggle back and forth from the stored version to mathematically version,loss in accuracy
Design 2: Store polar coordinates only	Computed on demand but not stored	Simply returned	- No need to use large amount of memory to store multiple instances for polar coordinates since variable are just simply returned	- Cannot reference previous polar coordinate used - Converting between cartesian and polar may reduce efficiency because the

			- Simple and straightforward implementation. - No need for conversion for polar coordinates making it more efficient in that aspect	cartesian would have to be computed several time for every class needed.
Design 3: Store Cartesian coordinates only	Simply returned	Computed on demand, but not stored	- Only stores cartesian values and not the polar values meaning that space is reserved. - Simpler as if/else loops are not needed to deduce the nature of the coordinate system. - Does not need to waste time storing values after computation.	- Cannot be reference the results for polar coordinates without recalculation.
Design 5: Abstract superclass with designs 2 and 3 as subclasses	Depends on the concrete class used	Depends on the concrete class used	- Saves time as it depends on the coordinate type for calculation. It does not need to do both, only the one that fits what the user is looking for. - More efficient as it only runs the need function when called upon and not every single time.	- Does not save the coordinates of the other type, therefore it cannot be referenced in the future. - More complex to implement. - Memory used depends on which class being used, not consistent.

			- Stores only the memory that it needs to carry out the task and nothing more. - Code is modulare	- Uses a hierarchy. - Cannot be reference the results for polar coordinates without recalculation.

E28.Run a performance analysis in which you compare the performance of Design 5, Design 2, and Design 3, as you implemented it in the previous exercise, with Design 1. Determine the magnitude of them differences in efficiency, and verify the hypotheses you developed in E26.

Design 5 is more efficient to design 1. This is because it does not waste time storing or returning information that is unneeded. For example, design 1 saves and recalculates, an arbitrary function as it could just be simply returned, the value of the inputted whether cartesian or polar. Design 1 also saves the calculated values, which is unnecessary for this class as nothing (aside from recalculation of the other coordinate system which has already been determined as arbitrary) is done with them. Design 5 eliminates the usage of needless calculations and wasted storage. The following show the general outputs of each.

```
Cartesian-Polar Coordinates Conversion Program
Enter the type of Coordinates you are inputting ((C)artesian / (P)olar): P
Enter the value of Rho using a decimal point(.): 6
Enter the value of Theta using a decimal point(.): 6

You entered:
Stored as Polar [6.0,6.0]


After asking to store as Cartesian:
Stored as Cartesian  (5.967131372209639,0.6271707796059207)


After asking to store as Polar:
Stored as Polar [5.999999999999999,6.0]

Elapsed Time 5486
```

Figure 1. Console for PointCPTest

Name	Value
no method return value	
args	String[0] (id=32)
startTime	1685264146201
point	PointCP (id=34)
typeCoord	P
xOrRho	6.0
yOrTheta	6.0

Figure 2. Variable Set Stage #1 for PointCPTest

println() returned	(No explicit return value)
args	String[0] (id=32)
startTime	1685264146201
point	PointCP (id=34)
typeCoord	C
xOrRho	5.967131372209639
yOrTheta	0.6271707796059207

Figure 3. Variable Set Stage #2 for PointCPTest

Name	Value
no method return value	
args	String[0] (id=32)
startTime	1685264146201
point	PointCP (id=34)
typeCoord	P
xOrRho	5.999999999999999
yOrTheta	6.0
finishTime	1685264484229
timeElapsed	338028

Figure 4. Variable Set Stage #3 for PointCPTest

As predicted, there is a loss of accuracy when it comes to the recalculation of the starting coordinates. In this case elapsed time is not considered as it was done in debugging stage one line at a time using “step over”, which increased processing in time.

```
Cartesian-Polar Coordinates Conversion Program
Enter the type of Coordinates you are inputting ((C)artesian / (P)olar): P
Enter the value of Rho using a decimal point(.): 6
Enter the value of Theta using a decimal point(.): 6

You entered:
Polar Stored as [6.0,6.0]


After asking to convert point:
Cartesian Value as (5.966773887365293,0.06531607967535985)

Elapsed Time 4973
```

Figure 5. Console for PointCPTest5

no method return value	(Not observed to speed up the long runni...
args	String[0] (id=16)
startTime	1685266860101
e	ArrayIndexOutOfBoundsException (id=22)
point	PointCPDesign2 (id=27)
Rho	6.0
Theta	6.0
typeCoord	P

Figure 6. Variable Set Stage #1 for PointCPTest5

println() returned	(No explicit return value)
args	String[0] (id=32)
startTime	1685266715235
point	PointCPDesign2 (id=26)
Rho	6.0
Theta	6.0
typeCoord	C
finishTime	1685266749395
timeElapsed	34160

Figure 7. Variable Set Stage #2 for PointCPTest5

In our version of PointCP5, the typeCoord changes and dictates which design it's going run, as well as the final output, however, the actual coordinate points never update. It is clear that this

could be improved upon by getting rid of typeCoord, however, our implementation still adheres to the specifications made above.

It is not always the case that using subclasses will yield a faster result, even while optimized as other factors such as processing speed and optimization conditions also affect the speed. In our case there were outlying times when design 1 yielded a faster time than design 5, however, without a large enough sample size, it is difficult to tell if these are the status quo, or isolated events.

```
Cartesian-Polar Coordinates Conversion Program
Enter the type of Coordinates you are inputting ((C)artesian / (P)olar): P
Enter the value of Rho using a decimal point(.): 6
Enter the value of Theta using a decimal point(.): 6

You entered:
Polar Stored as [6.0,6.0]


After asking to convert to Cartesian:
Cartesian Value as (5.967131372209639,0.6271707796059207)

Elapsed Time 8599
```

Figure 8. Console for PointCPTest2 - Polar Input

Name	Value
no method return value	
args	String[0] (id=16)
startTime	1685316808306
point	PointCP2 (id=17)
Rho	6.0
Theta	6.0
typeCoord	P

Figure 9. Variable Set Stage #1 for PointCPTest2

Name	Value
println() returned	(No explicit return value)
args	String[0] (id=16)
startTime	1685316808306
point	PointCP2 (id=17)
Rho	6.0
Theta	6.0
typeCoord	C

Figure 10. Variable Set Stage #2 for PointCPTest2

```
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
Enter the value of Y using a decimal point(.): 5

You entered:
Cartesian Value as (0.0,0.0)


After asking to convert to Cartesian:
Cartesian Value as (0.0,0.0)

Elapsed Time 8017
```

Figure 11. Console for PointCPTest2 - Cartesian Input

For both designs 2 and 3, which stores polar and cartesian inputs respectively, the variables for each point are only stored when it is the variable the design is allowed to store. This means that it has a specialize function to either take a polar or cartesian input and compute the opposite. As you see in Figure 11, when the wrong coordinate type is imputed, it is unable to calculate anything and the initialized value, 0, is used in its place. In Figure 10, while the typeCoord variable changes, the actual stored numbers do not.

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 four versions of PointCP – Design 1, Design 2, Design 3 and Design 5.

****See PointCPPerformanceTest.java****

E30 Summarize 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

Table II. Comparison of Average Time To Complete Operations in Nanoseconds

Operations	Design 1	Design 2	Design 3	Design 5
Retrieving Polar coordinate	362	103	165	152
Retrieving Cartesian coordinate	164	174	104	157
Convert Cartesian to Polar	133	84	78	Does not have this function
Convert Polar to Cartesian	176	68	68	Does not have this function
Convert to Opposite	Does not have this function	Does not have this function	Does not have this function	81
getX()	58	106	59	97
getY()	72	96	71	95
getRho	77	64	79	70
getTheta	129	62	126	112
Convert to String	1296	884	793	1225

An observation made is that the more instances made, the shorter the time. Initially, the computation would have been in milliseconds, but because of the intense speed, milliseconds were too small to be viable answers and therefore the will be computed in nanoseconds. Also due to the extremely small numbers, it would be more beneficial to display the average time it takes to process a task, rather than how many tasks could be processed over a second.

After 50000 iterations, it would appear that the first design shows more efficient speeds than the fifth design. This may be because committing information to memory allows it to access it quicker without having to recalculate it everytime. These leads to the hypothesis that storing to memory takes less time

than recalculation, but only by a very small margin. Designs with specialized functions show efficient speeds further observation where it be comes clear that they are incapable of processing more. This becomes evident with Designs 2 and 3 respectively as retrieving the polar coordinate and the cartesian coordinate take less time than their opposing counter part. This is because for design 2, the polar coordinate is stored and design 3 the cartesian coordinate is stored, therefore it could just pull them from memory without having to recalculate each time, thus yielding a faster result.

Converting to string takes the most time. It is believed that the method used to convert to string is inefficient, and the use of reference data types have not been optimzed in our version. In design 2 and 3, the coordinate type can only be one or the other, and if it isn't then the program does not compute properly as it does not store coordinates from the other system. Because of this, the to string function is always consistent and does not call for a recalculation, therefore it saves more time.

To conclude, our hypotheses for each designs were partially correct. We have correctly predicted how variables are stored in the system of each class. We have also correctly stated how the memory storage is used for each design, and what it means for efficiency. What we did not predict correctly was time efficiency. We initially predicted that design 5 would be more efficient than design 1 most of the time, however, that was not the case as in most instances, design 1 is more efficient. Margins of error such as device, access time, and memory storage speed and retrieval must be taken into account as well as the optimization of the code. We have concluded that in general the minimum execution time is about 5000 ms and the maximum execution time is about 10000 ms for each design.

TESTS:

The first thing we did was to make a *PointCPTest.java* for each design called *PointCPTest2.java, PointCPTest3.java, PointCPTest5.java*. We then ran both cartesian and polar coordinates and added time elapse (in Milliseconds) to each one to see how much time it took to complete each one. We observed how the variables were (or were not stored) and how that impacted the final output. The figures 1-11 show samples of the console output as well as the variable storage information. We compared the results with our hypotheses to come to the conclusion that we were correct about outputs and memory storage.

The second test was the performance test. We made a new test file called *PointCPPerformanceTest.java*. There is a method that generates random points (random bool for polar or cartesian, and random doubles for x/rho or y/theta) within each design class. This test class times (in Nanoseconds) how long it takes for each design to complete an operation (see: *Table II*). It counts the total time and divides it by the number of instances to show the average time it takes for the for an operation to be completed. This was done because computational speed was too small to be significant. There are 50000 instances that were done and the final totals are displayed.

Part II: Object Oriented Review

****All files are located in code-part2 directory and the implementation could be tested using the Test.java file.****

OUTPUT:

Running Address.java and Employee.java Test...

---------- Employee DataBase ----------

Random employee Id generated ...

Employee Name generated ...

| First address information----------

Employee street name generated...

Postal Code generated ...

| Secondary address information----------

Employee street name generated...

Postal Code generated ...

| Your employee Information
---------- Employee information ----------
Employee ID : 0146
Employee Name : John Smith
Employee Hours : 40
Employee Rate : 15.5

1 || Employee Adresss ----------
Street Name : mainland drive
Number : 48

Postal Code : K9O P5T

2 || Employee Adresss ----------
Street Name : okaland 3rd avenue
Number : 800
Postal Code : K9O P5T