Santos Gagbegnon
Graydon Hope

Object Oriented Review - SEG 2105 A1

September 30th, 2018

POINTCP

Comparison Analysis

E26) Advantages and Disadvantages of Design 2, 3 & 6

Design #	Advantages	Disadvantages
2	-Not as many instance variables being stored as in the original pointCP -Fewer lines of code	-Requires a lot of computing to cartesian since both rotate point and distance are calculated using cartesian points -Everytime getX() and getY() are called, the same calculation is being done over and over again. In a case where those methods are called numerous times, the computer has to do the same calculation over and over again. -Sin and Cos (more computationally expensive than square root) is only called every time getX() & getY() are called -Casting from interface to Design 2 will take some more execution time (assuming Design 2 implements design 6)
3	-Not as many instance variables being stored as in the original pointCP	-Everytime getRho() and getTheta() are called, the same

	-Fewer lines of code -Fewer computing situations because methods like rotate point and distance call getX() which is a stored value -Sin and Cos (more computationally expensive than square root) is only called a maximum of once (the constructor)	calculation is being done over and over again. In a case where those methods are called numerous times, the computer has to do the same calculation over and over again. -Casting from interface to Design3 will take some more execution time (assuming Design 3 implements design 6)
6	-Forces any PointCP to contain specific methods, ensuring that all PointCPs have the same basic functionalities -Allows for easy switching between different PointCPs (casting from interface -> D2/D3)	**-**

Testing

Testing 1.1: Validating Design 2 and 6

To ensure design 2 worked properly, runAndPrintTest(10) was called. This method would generate random points and would call every method in design and ensure it worked was expected by printing its results. We also tested Design 6 by using PointCPTestDesign 6 point and casting to to the Design 2 before calling the methods. This test was run 10x to ensure our method worked for all types of values (negative,positive,decimal, cartesian, polar etc…). *Example 1* contains 2/10 test runs.

Example 1: Testing Design 2 and 6

********TEST NUMBER: 3/10 (Polar points in constructor)

Point 1:Polar [78.0,224.0] Point 2:Polar [78.0,224.0]

getX(): -56.10850442641478
getY(): -54.1833528958018
getRho(): 78.0
getTheta(): 224.0
convertStorageToCartesian():Cartesian (-56.10850442641478,-54.1833528958018)
convert
getTheta(): 216.0
convertStorageToCartesian():Cartesian (-15.371322893124004,-11.167919793556987)
convertStorageToPolar():Polar [19.0,-144.00000000000003]
rotatePoint 0: Polar [19.0,-144.00000000000003]
rotatePoint 90: Polar [19.0,-54.00000000000003]
rotatePoint -90: Polar [19.000000000000004,-144.00000000000003]
Distance between point 1: 12.650633568701318

********TEST NUMBER: 9/10 (Cartesian points in constructor)
Point 1:Polar [180.9447429465692,31.66973514666302] Point 2:Polar [180.9447429465692,31.66973514666302]
getX(): 154.0
getY(): 95.0
getRho(): 180.9447429465692
getTheta(): 31.66973514666302
convertStorageToCartesian():Cartesian (154.0,95.0)
convertStorageToPolar():Polar [180.9447429465692,31.66973514666302]
rotatePoint 0: Polar [180.9447429465692,31.66973514666302]
rotatePoint 90: Polar [180.9447429465692,121.66973514666302]
rotatePoint -90: Polar [180.9447429465692,31.66973514666302]
Distance between point 1: 65.11528238439882

Testing 1.2: Validating Design 3 and 6

When testing Design 3, we used the same process as Design 2, except we replaced Design 2 with Design 3 in our code. *Example 2* displays 2/10 tests from the run.

Example 2: Testing Design 3 and 6

********TEST NUMBER: 2/10 (Polar points in constructor)
Point 1:Cartesian (51.655553670626624,116.02027312061031) Point 2:Cartesian (51.655553670626624,116.02027312061031)
getX(): 51.655553670626624
getY(): 116.02027312061031
getRho(): 127.0
getTheta(): 66.0
convertStorageToPolar():Polar [127.0,66.0]
convertStorageToCartesian():Cartesian (51.655553670626624,116.02027312061031)
rotatePoint 0: Cartesian (51.655553670626624,116.02027312061031)
rotatePoint 90: Cartesian (-116.02027312061031,51.65555367062663)
rotatePoint -90: Cartesian (51.655553670626624,116.02027312061031)
Distance between point 1: 344.3101654455537
rotatePoint -90: Cartesian (-76.28566603468914,108.9472218904359)
Distance between point 1: 159.99561666779545

********TEST NUMBER: 7/10 (Cartesian points in constructor)
Point 1:Cartesian (44.0,145.0) Point 2:Cartesian (44.0,145.0)
getX(): 44.0
getY(): 145.0
getRho(): 151.52887513606112
getTheta(): 73.11966983447614
convertStorageToPolar():Polar [151.52887513606112,73.11966983447614]
convertStorageToCartesian():Cartesian (44.00000000000001,145.0)
rotatePoint 0: Cartesian (44.00000000000001,145.0)
rotatePoint 90: Cartesian (-145.0,44.000000000000014)
rotatePoint -90: Cartesian (44.00000000000001,145.0)
Distance between point 1: 248.2438317461282

Testing 1.3: Tracking Run Time of Methods

When running our tests to track run time we called each method individually. We ran each method 125,000,000 times and ensured once the test started we did not open any tabs or run any other programs to ensure equal testing. We also rotated every single point by 90 degrees each time the method was called. Each method was run 10 times to obtain a good amount of data and was averaged out to get the median, worst and best time.

		Methods									
		getX	**getY**	**getRho**	**getTheta**	**convertStorageToP**	**convertStorageToC**	**rotateP**	**getDistance**	**Constr.**	**toString**
Design 2 w/ interface	**Average (ms)**	10,183	10,231	9,944	9,908	9902	11,141	27,086	15,700	4,935	18,780
	Worst Time (ms)	10,269	10,252	10,031	9,922	9960	11,161	27,139	15,817	4,966	19,069
	Best Time (ms)	10,269	10,217	9,904	9,890	9878	11,122	27,028	15,668	4,916	18,662
Design 3 w/ interface	**Average (ms)**	9901	9,897	9,851	18,016	18,046	9,829	13,736	9,858	9,308	90,134
	Worst Time (ms)	9916	9,914	9,894	18,155	18,144	9,856	13,816	9,930	9,695	90,809
	Best Time (ms)	9887	9,887	9,836	17,951	18,010	9,810	13,685	9,813	9.097	89,445
Design 6	**Average (ms)**	18,195	18,142	18,143	27,065	27.106	18,148	23,830	18,311	n/a	n/a
	Worst Time (ms)	18,376	18,186	18,217	27,593	27,244	18,194	24,676	18,604	n/a	n/a

	Best Time (ms)	18,143	18,120	18,110	26,959	27,051	18,126	23,615	18,183		
Design 2 No interface	**Average (ms)**	10,827	10,261	9,877	9,872	9,879	11,157	27,368	15,691	4,949	18,885
	Worst Time (ms)	10,925	10,335	9,905	9,924	10,015	11,277	29,892	15,788	5,034	19,123
	Best Time (ms)	10,282	10,218	9,848	9,853	9,845	11,107	26,894	15,642	4,899	18,792
Design 3 No interface	**Average (ms)**	18,244	18,254	18,238	27,281	27,324	18,258	23,788	18,253	9,127	81,680
	Worst Time (ms)	18,325	18,337	18,330	27,690	27,441	18,341	23,920	18,290	9,193	82,297
	Best Time (ms)	18,198	18,199	18,206	27,179	27,238	18,194	23,723	18,219	9,084	81,441

Note:
convertStorageToP() = convertStorageToPolar()
convertStorageToC() = convertStorageToCartesian()
rotateP() = rotatePoint()
Constr. = constructor

Arrays

Testing Array:

Array Time (ns)	Adding to array	Summing elements in array (loop)
Average - 3 runs	7327662254	44750890.33333

Best	7135833068	43241604
Worst	7450003720	45653713

Testing ArrayList:

ArrayList Time (ns)	Adding to arraylist	Summing elements in arraylist (iterator)
Average - 3 runs	10728345412.66667	103809358.33333
Best	10014486578	102218733
Worst	11702450310	106608772

Testing Vector:

Vector Time (ns)	Adding to vector	Summing elements in vector (iterator)
Average - 3 runs	10295771304	410897855.66667
Best	10059395821	404758788
Worst	10414830596	414919782

Note: Each test was run 123,000,000 times.

Conclusion

In conclusion, a basic array does better than an ArrayList and Vector when adding elements to the back. It also faster when summing up all the elements. Because of this, a recommendation to software engineers would be to use a basic array when adding many elements to a list/vector or summing up the elements in the list. If using an array is not possible, using a vector to sum all elements present is more efficient than an ArrayList. All in all, one should avoid using ArrayLists when possible.