Lab 2

Students: Harsh Gupta, 300042828; Rohan Sood, 8766597
Repository: https://github.com/guptaharsh25/seg2105_lab2

POINTCP

E-26

Design	Advantages	Disadvantages
Design 2: Store polar coordinates only	-No need to compute polar coordinates	-Need to compute cartesian coordinates
Design 3: Store cartesian coordinates only	-No need to compute cartesian coordinates	-Need to compute polar coordinates
Design 6: Interface with designs 2 and 3 implementing it	-Structured contract for writing classes for points, all future point class implementations can implement this interface to know exactly which methods must be implemented -Good software design principles	-code is a bit more complex due to use of interfaces and concrete class

E-28 - E-30

Each numeric value below is the time taken in milliseconds to run each method 300000 times. The minimum, median and maximum value are the statistics of running each test of 300000 iterations 25 times.

Method	**Computation Time Type (milliseconds)**	**Designs**			
		Design 2	**Design 3**	**Design 6**	
				Polar	**Cartesian**
Constructor	Minimum of 25 trials	13	13	13	13
	Median of 25 trials	14	14	14	14
	Maximum of 25 trials	26	26	26	28
getR & getTheta	Minimum of 25 trials	0	27	0	35
	Median of 25 trials	0	31	0	40
	Maximum of 25 trials	6	42	5	48
getX & getY	Minimum of 25 trials	4	0	3	0
	Median of 25 trials	8	0	8	0
	Maximum of 25 trials	10	4	10	4
distance	Minimum of 25 trials	28	13	29	15
	Median of 25 trials	31	14	31	15
	Maximum of 25 trials	41	18	40	19
rotate	Minimum of 25 trials	89	22	87	22
	Median of 25 trials	90	23	88	23
	Maximum of 25 trials	100	29	97	29
polarToString	Minimum of 25 trials	286	218	165	216
	Median of 25 trials	290	224	168	221
	Maximum of 25 trials	509	329	279	329
cartesianToString	Minimum of 25 trials	173	168	183	162
	Median of 25 trials	175	176	186	167
	Maximum of 25 trials	302	217	226	203

How tests were performed: A new testing class was made for each design (for Design 6 there were 2 testing classes). Each class had 7 methods where each method in the class to be tested was being tested. For each test, we calculated the run time for 300000 method calls 25 times and stored it in an array. This array was then sorted and we found the min, median and max run time for the 25 tests and put it in a new array. This new array was returned (and eventually printed).

Sample Output: Following is sample output of Design2

```
--------------------------------------
---------------DESIGN 2---------------
--------------------------------------
Constructor: [13, 14, 26]
getPolar: [0, 0, 6]
getCartesian: [4, 8, 10]
distance: [28, 30, 36]
rotate: [87, 91, 98]
polarString: [178, 190, 303]
cartesianString: [176, 185, 231]
```

Discussion of Results: Overall the results of the tests match our anticipated results. The designs where the polar coordinates were stored were faster at returning polar coordinates and slower at cartesian calculations such as distance. Whereas the classes where cartesian coordinates were stored were faster at returning cartesian coordinates. The addition of the interface in Design 6 did not seem to have an impact on runtime. Although the interface implementation does not impact runtime it makes it much easier to build future classes as we have a contract we must follow if we implement the interface.

ARRAYS

Following are the results of initializing ArrayList, Array and Vector of size 105999999, and summing the values of those respective collections using an iterator or for loop.

Type of calculation		**Time taken (ms)**
Initializing ArrayList	*not declaring initial size*	10464
	declaring initial size	3372
Initializing Array		4623
Initializing Vector	*not declaring initial size*	9939
	declaring initial size	3571
Sum using Iterator in ArrayList		124
Sum using for-loop Array		52
Sum using Iterator in Vector		414

The above results show that for handling large datasets, if the size of the dataset is known then ArrayLists are the most efficient. In this experiment we did not test a dynamic array (a test where the array grows to accommodate bigger data size) so we do not know which collection type is best for when we do not know the size of the dataset. We believe if size is unknown, ArrayList would still be the best option, since instantiating new, bigger arrays is not a very efficient algorithm.

Our recommendation to designers would be to use ArrayLists as it seems to have the best performance for initialization and iterators.