|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Design** | **Simplicity of Code** | **Creating instances Efficiency** | **Computations’ Efficiency** | **Memory used** |
| 2 (PointPolar) | Just as simple as D3 |  | getX, getY, and getDistance will be slower as more conversions must be done.  getRho, and getTheta are faster as they are stored values. |  |
| 3 (PointCoord) | Jut as simple as D3 |  | getX, getY, and getDistance methods will be faster as X and Y coordinates are already stores.  getRho and getTheta are slower as converesions must be done |  |
| 6 (interface +Designs 2,3) | Will require an extra class, the interface to be created |  | Average efficiency for all methods should be in between design 2 and design 3, as it depends on what concrete class was used. |  |

**Assignment 1**

**Part 1: Point classes**

**E.26** Advantages and Disadvantages of each design hypothesis

For the following exercises we created XXXX instances of each class and tested each method XXXX times:

**E.28**

Magnitude of the differences in efficiency of designs:

Comparison to hypotheses from E.26:

**E.29**

Performance analysis on all methods ran using PointTest.java. Results summarized in E.30

**E.30**

Average Computations speeds for the operations on different designs in milliseconds

|  |  |  |  |
| --- | --- | --- | --- |
| **Operations** | **Design 2** | **Design 3** | **Design 6** |
| getX() |  |  |  |
| getY() |  |  |  |
| getRho() |  |  |  |
| getTheta() |  |  |  |
| getDistance() |  |  |  |
| rotatePoint() |  |  |  |
| toString() |  |  |  |

**Part 2: Vector/ArraysLists/Arrays**

|  |  |  |  |
| --- | --- | --- | --- |
| **Time** | **ArrayList** | **Vector** | **Array** |
| **Test 1** | 0.122947988 | 3.260702445 | 0.079015661 |
| **Test 2** | 0.175345685 | 3.750741664 | 0.095158389 |
| **Test 3** | 0.143885242 | 3.377636697 | 0.09134859 |
| **Average** |  |  |  |

The table displays the time taken in seconds for each collection to sum up all the integers stored. We chose to test this collections containing 80 000 000 integers, ranging from 0-9.

From these results, we recommend that developers always use arrays when the size is known as the array was XX times faster than the arraylist. If the size is unknown then we would recommend arrayList over vectors for performance as the vector was XXX slower than the arraylist.