E26

|  |  |  |
| --- | --- | --- |
| Design | Advantages | Disadvantages |
| Design 1 | Allows to store cartesian or polar values and will be faster returning that type of coordinate that is stored | Will be slower to return the type of coordinate that is not store because it calculates the value when returning it |
| Design 2 | Fast to store and return polar coordinates | Slower to store or return cartesian coordinates because it has to convert it to or from polar |
| Design 3 | Fast to store and return cartesian coordinates | Slower to store or return polar coordinates because it has to convert it to or from cartesian |
| Design 4 | Fast to return both types of coordinates | Will always have to convert values to store the other type of coordinates when storing because it stores both types at the same time |
| Design 5 | Allows to use the best type of subclass depending on if we’ll use cartesian or polar more | Better for using only one type of coordinates |

E28 & E29 see `pointcp/test.java` program output:

Average time for getting X from 5000 PointCP1 is: 1.52E-4 milliseconds

Average time for getting Y from 5000 PointCP1 is: 1.64E-4 milliseconds

Average time for getting Rho from 5000 PointCP1 is: 9.9E-5 milliseconds

Average time for getting Theta from 5000 PointCP1 is: 0.002135 milliseconds

Average time for getting Distance from 5000 PointCP1 is: 1.76E-4 milliseconds

Average time for getting Rotate from 5000 PointCP1 is: 2.31E-4 milliseconds

Average time for getting X from 5000 PointCP5 is: 7.8E-5 milliseconds

Average time for getting Y from 5000 PointCP5 is: 6.7E-5 milliseconds

Average time for getting Rho from 5000 PointCP5 is: 8.1E-5 milliseconds

Average time for getting Theta from 5000 PointCP5 is: 1.68E-4 milliseconds

Average time for getting Distance from 5000 PointCP5 is: 1.6E-4 milliseconds

Average time for getting Rotate from 5000 PointCP5 is: 2.16E-4 milliseconds

E30

**Average Computation Speed in 1E-5 Milliseconds from 5000 Iterations**

|  |  |  |
| --- | --- | --- |
|  | **PointCP1** | **PointCP5** |
| Average time for getting X | 15.2 | 7.8 |
| Average time for getting Y | 16.4 | 6.7 |
| Average time for getting Rho | 9.9 | 8.1 |
| Average time for getting Theta | 213.5 | 16.8 |
| Average time for getting Distance | 17.6 | 16 |
| Average time for getting Rotate | 23.1 | 21.6 |

PointCP5 was consistently faster than PointCP1, since we use the best type of subclass depending on which coordinates are used. This creates a large discrepancy in “Average time for getting Theta” since PointCP1 has to do a complicated calculation every time, leading to a much higher time than any other category;