|  |  |  |
| --- | --- | --- |
| Designs | Avantages | Désavantages |
| Design 2 | * Permet de stocker les coordonnées polaires * Permet de retourner les coordonnées polaires | Doit calculer les coordonnées cartésiennes chaque fois pour les retournées |
| Design 3 | Permet de stocker les coordonnées cartésiennes.  Permet de retourner les coordonnées cartésiennes. | Doit calculer les coordonnées polaires chaque fois pour les retournées |
| Design 6 | Permet de retourner les coordonnées polaires et cartésiennes  Permet de stocker les coordonnées polaire et cartésiennes  Très efficace | Doit calculer soit les coordonnées polaire ou cartésiennes pour stocker celle-ci dans une des classes  Utilise 2 classes et une interface  Prend plus de mémoire que design 2 et 3 |

E26

E28

Le design 6 est de loin le design le plus efficace car, il permet de stocker les coordonnées cartésiennes et polaires et les retournées sur la demande de l’utilisateur. Donc le calcul pour le convertir en coordonnées polaire ou cartésiennes se produit seulement une fois. Ce qui rend le design 6 le plus efficace tel notée dans E26.

Quant au design 2 et 3 nous observons que convertir les coordonnées polaires en cartésiennes prend moins de temps que l’opposée. La raison est que design 3 à besoin de l’appel de 5 fonctions mathématiques mais design 2 à besoin que 4 fonctions. Cette petite différence devient très lorsqu’on appelés les fonctions plusieurs fois. Donc, le design 2 est plus efficace que le design 3.

En conclusion, l’efficacité des designs est :

1. Design 6
2. Design 2
3. Design 3.

E29

Design 2: Average execution time in milliseconds: 0.21326130000000001

Design 3: Average execution time in milliseconds: 0.4448906

Design 6: Average execution time in milliseconds: 0.016

E30

Table des temps d’exécutions pour chaque méthode des designs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Designs | getX() | getY() | getRho() | getTheta() |
| Design 2 | 0.1931533 ms | 0.19305809999999998 ms | 0.1665985 ms | 0.1790988 ms |
| Design 3 | 0.1664697 ms | 0.1683425 ms | 0.17754 ms | 0.41491140 ms |
| Design 6 | 0.013 ms | 0.011 ms | 0.004 ms | 0.005 ms |

Observations

* La raison que Design 3 est le moins efficace est le temps d’exécution de sa méthode getTheta()
* Si on ignore la méthode getTheta(), les autres méthodes de design 3 sont plus efficace que celle de design 2
* Nous observons que l’implémentation avec l’interface de design 6 est de loin le plus efficace