**UNIVERSITE D’OTTAWA**

*COURS DE SEG2505[A] – Automne 2023*

*PROF : OUKAIRA AZIZ*

**DEVOIR 1 : REVISION DE LA PROGRAMMATION ORIENTEE OBJET**

**Groupe 7**

A.E : Ngassa Ralph

Etudiants : Ahmed Omar Mohamed 300267489

Gouled Abdillahi 300257802

Hilaire Junior Kalala 300289737

Yvan Kamga 300291315

SOLUTION E26

Devoir1 SEG2505:

E26:

|  |  |  |
| --- | --- | --- |
|  | Avantages | Désavantages |
| Design 1 | * Le code est plutôt simple car soit il renvoie directement le résultat demandé, ou bien il fait de simples calculs avant de renvoyer ce résultat * Le code est aussi efficace en termes de mémoire car chaque point d’un type de coordonnée est associé à une seule et unique variable | * Nous devons tout le temps surveiller le drapeau afin de savoir s’il s’agit d’un cordonnée polaire ou cartésienne * Risque de rencontrer des difficultés à surveiller le drapeau si les calculs des données deviennent complexes. |
| Design 2 | * Le code est plutôt efficace en termes de mémoire étant donné le fait qu’on ne sauvegarde pas les valeurs cartésiennes à chaque fois qu’on doit les calculer. * Le calcul est fait uniquement quand c’est demandé, ce qui fait qu’aucun calcul inutile n’est effectué, et cela nous fait gagner en temps | * Etant donné que les données ne sont pas sauvegardées mais juste calculé quand c’est nécessaire, si nous devons tout le temps avoir accès aux données ou bien si elles changent tout le temps, cela peut être problématique |
| Design 3 | * Le code est plutôt efficace en termes de mémoire étant donné le fait qu’on ne sauvegarde pas les valeurs polaires à chaque fois qu’on doit les calculer. * Le calcul est fait uniquement quand c’est demandé, ce qui fait qu’aucun calcul inutile n’est effectué, et cela nous fait gagner en temps | Etant donné que les données ne sont pas sauvegardées mais juste calculé quand c’est nécessaire, si nous devons tout le temps avoir accès aux données ou bien si elles changent tout le temps, cela peut être problématique |
| Design 4 | * Le code nous fait gagner en temps car aucun calcul complexe n’est nécessaire, nous avons déjà les deux types de cordonnées nécessaires (polaire et cartésienne), nous avons juste à les renvoyer * Le code est simple à écrire et comprendre, en d’autres termes, le code est explicite | * Etant donné que nous avons plusieurs variables, en ce qui concerne l’espace mémoire, le code en occupe une grande quantité |
| Design 5 | * Efficace car à cause du concept de l’héritage, cela peut réduire le nombre d’instances que nous avons à créer car certaines d’entre elles peuvent avoir les mêmes caractéristiques. | * Le code peut ne pas être simple à cause du fait que nous allons devoir employer le concept de l’héritage * Nous pouvons perdre en espace mémoire car le calcul des coordonnées cartésiennes et polaires dépend de la classe utilisée |

Solution E28

Nous remarquons que la Moyenne du temps d’exécution est de 1.8 milliseconde pour le design 5 et il est de 2.5 millisecondes pour le design 1. On en conclut donc que l’utilisation de la classe abstraite dans le design 5 et la possibilité de créer 2 types d’objet en fonction du type des coordonnées est plus efficace.

Solution E29

Nous avons mis au point une petite classe nommée BigTest qui génère 2000 objets et qui appelle chaque méthode d’instance pour chacun de ces objets. En fait, elle génère 2 points (l’un avec des coordonnées cartésiennes et l’autre avec des coordonnées polaires) 1000 fois avec des coordonnées aléatoires à chaque fois, donc 2000 points différents au total. Et à chaque itération, on appelle toutes les méthodes d’instance pour les 2 points actuels.

Solution E30

Nous avons effectué 10 de ce BigTest pour le design 1 et 5

|  |  |  |
| --- | --- | --- |
| Temps d’exécution | Design 1 | Design 5 |
| Minimum | 2 millisecondes | 1 milliseconde |
| Médian | 2.8 millisecondes | 2.5 milliscondes |
| Maximum | 4 millisecondes | 4 millisecondes |