TP1 - Introduction à Python et à la programmation orientée objet

26 janvier 2024

Contraintes et remises des TP

- La correction tiendra compte de la briéveté des méthodes que vous écrivez (évitez lesf onctions de plus de 25 lignes) → n'hésitez pas à découper une méthodes en plusieurs sous-méthodes (privées) plus courtes.
- Vos classes et méthodes doivent être documentée, et vos méthodes (hors accesseurs et autres méthodes triviales) doivent avoir des tests unitaires.
- Pour l'UML, vous pouvez utiliser l'outil en ligne http ://www.draw.io/.
- Les noms de classe commencent par une majuscule.
- Les noms de méthodes et d'attributs commencent par une minuscule.
- La convention de nommage des accesseurs est getNomAttribut() et setNomAttribut(...).
- Utilisez un dossier par TP, et un sous dossier par exercice ou série d'exercice dépendant les uns des autres.
- Pour le rendu, placez les schémas UML et autre documents à rendre éventuellement dans un dépôt **git** dont vous mettrez le lien sur Moodle au plus tard une semaine après le TP.

Barème - Tous les exercices : coefficient 1. Propreté, lisibilité et commentaires : coefficient 1.

Exerice 1 - Syntaxe

1. Écrivez un programme qui calcule puis affiche la somme des n premiers entiers (de 0 à n, n inclus). Cet algorithme devra faire partie d'une classe Algèbre.
2. Ajouter une nouvelle fonction factorielle à la classe. La fonction prendra un argument n en entrée.
3. Votre programme doit maintenant être capable d'accepter en entrée des arguments spécifiés en ligne de commande.

Exerice 2 - Vecteur

Dans cet exercice, on souhaite modéliser des vecteurs en trois dimensions à l'aide d'une classe que l'on nommera Vecteur. Ce type de classe est présent dans la plupart des logiciels manipulant des objets en 3D (moteurs de jeu, modeleurs, ray-tracers, ...). Les opération suivante doivent pouvoir être effectuées :

- créer un vecteur avec pour coordonnées [0, 0, 0] ;

— créer un vecteur avec les coordonnées x, y, z fournies en argument au constructeur ;
— additionner le vecteur self avec un vecteur passé en argument puis renvoyer le résultat sous la forme d'un nouveau vecteur — on ne modifie pas self (méthode additionner) ;
— calculer puis renvoyer la norme de self (méthode calculerNorme) ;
— calculer puis renvoyer le produit scalaire de self avec un vecteur passé en argument (méthode calculerProduitScalaire) ;
— calculer puis renvoyer le vecteur correspondant à self après une rotation autour du centre du repère de α radians dans le plan x, y, α étant passé en argument — on ne modifie pas self (méthode tourner) ;
— afficher les coordonnées du vecteur sur la sortie standard (méthode afficher).

1. Dessiner le diagramme de classe UML représentant cette classe.
2. Implémentez et testez la.

Exercice 3 - Triangle

On souhaite maintenant s'appuyer sur la classe Vecteur pour réaliser une classe Triangle. Cette classe doit permettre d'effectuer les opérations suivantes :
— créer un triangle à partir de trois vecteurs (donnant les coordonnées des trois points) ;
— faire tourner le triangle autour du centre du repère en fonction d'un angle α passé en argument de la méthode — on modifie self (méthode tourner) ;
— afficher les coordonnées des vecteurs constituant self sur la sortie standard (méthode afficher).
— faire effectuer une translation du triangle — on modifie self — selon un vecteur passé en argument (méthode deplacer).

1. Ajoutez la classe Triangle au diagramme UML contenant déjà la classe Vecteur.
2. Implémentez puis testez comme précédemment la classe Triangle.

Exercice 4 - Objet 3D

Un objet 3D est constitué :
— d'une liste de triangle (que l'on représentera par une liste) ;
— d'un centre de gravité ;
— d'une couleur, représentée par une classe Couleur (qui correspond à un triplet de réels [r, v, b]).

Les opérations que l'on souhaite pouvoir effectuer sur un Objet3D sont les suivantes :
— créer l'objet à partir de son centre de gravité et de sa couleur ;
— afficher les coordonnées de tous les points de l'objet (méthode afficher) ;
— ajouter des triangles à l'objet (méthode ajouter Triangle)
— effectuer une translation de l'objet en fonction d'un vecteur passé en argument (méthode déplacer) ;

1. Ajoutez les classes Couleur et Objet3D au diagramme UML.
2. Implémentez les classes Objet3D et Couleur

Exercice 5 - 421

On souhaite maintenant réaliser un jeu inspiré du 421. Ce jeu se joue avec 3 dés. Au début de chaque tour, chaque joueur lance 3 dés. Chaque joueur peut alors décider de relancer un ou deux des trois dés (on ne peut pas relancer deux fois le même dé). On observe alors les combinaisons et on compte les points :

- 4, 2, 1 : 10 points ;
- Deux « 1 » et un autre dé : rapporte la valeur du troisième dé (par exemple 5 points pour 5, 1, 1) ;
- Trois chiffres consécutifs (par exemple 4, 5 ,6) : 2 points.

1. Dessinez un diagramme UML de classes pour modéliser ce jeu. Utilisez trois classes : une classe pour représenter un dé, une pour représenter un joueur et une autre pour le jeu.
2. Implémentez et testez votre jeu.