Exercice de classe #1

Travail d'équipe permis (2 personnes)

Date limite de remise : dimanche le 19 juin 2022

Énoncé du problème

Dans cet exercice, vous pratiquez les notions et la syntaxe de base de la programmation orientée objets PHP.

Vous devez implémenter une hiérarchie de classes représentant des formes 3D variées. De plus, votre implémentation permettra la comparaison des objets, même lorsqu'ils sont de formes différentes.

Au lieu de vous donner de longues instructions sur les détails d'implémentation, je profite de cet exercice pour vous présenter le « *diagramme de classes UML* » (fichier PNG joint à cet énoncé).

Vous pouvez vous documenter hors cours sur les *diagrammes de classes* en commençant peut être par la lecture de l'article Wikipédia sur le sujet à l'adresse suivante https://en.wikipedia.org/wiki/Class_diagram (la version française de cet article est malheureusement incomplète, et je ne la recommande pas).

Cependant, pour les besoins de cet exercice, vous n'aurez besoin que des indications suivantes :

- Chaque boîte représente une classe (qui peut être abstraite) ;
- Chaque boîte représentant une classe consiste en 3 cases : le nom de la classe, les champs (ou *attributs*, ou *propriétés*, ou *caractéristiques*), et les méthodes (ou *fonctions*, ou *actions*, etc.) ;
- Les classes (ou méthodes) abstraites ont leurs noms spécifiés en *italique* ;
- Le symbole « - » dénote que le membre a la visibilité privée, le « # » la visibilité protégée, et le « + », la visibilité publique ;
- Les membres *statiques* sont soulignés ;
- La flèche utilisée représente *l'héritage* (on verra d'autres relations dans la suite du cours).

A. Définir la classe `Forme` — 1 point

Produisez le code de la classe abstraite `Forme` tel qu'illustré dans le diagramme de classes.

La fonction de comparaison `comparerAvec()` servirait à *comparer* deux instances de formes quelconques selon la somme de leur superficie et volume (la première étant l'instance sur laquelle cette méthode est appelée, la seconde étant l'instance donnée en argument de la méthode).

Remarque : une fonction de comparaison retourne -1 (ou un nombre négatif), 0, ou 1 (ou un nombre positif), selon que pour les objets comparés le premier soit considéré *plus grand* que le second, les deux soient considérés *égaux*, ou que le premier soit considéré *plus petit* que le second.

B. Instances de formes — 1 point

Toutes les instances de `Forme` contiennent un identifiant unique (l'entier `id`) déterminé dynamiquement au moment de la création de l'instance : l'identifiant de la première instance est 1, celui de la deuxième 2, et ainsi de suite. Pour obtenir cet identifiant on peut utiliser la propriété statique `nbFormes` qui est incrémentée à chaque création d'une nouvelle instance d'une forme 3D, et qui tient compte ainsi du nombre total des objets de forme 3D créés dans le programme.

C. La méthode magique `__toString()` 1 point

Pour chacune des classes de forme 3D, implémentez la méthode `__toString()` pour qu'elle retourne une chaîne de caractères formatée selon le modèle suivant :

```
Cylindre, id = 5, superficie = 248.53, volume = 243.81, hauteur = 15.33, rayon = 2.25.
```

Adaptez cette chaîne pour chacune des classes concrètes `Sphere`, `PaveDroit`, `Cylindre`, et `Cube`, selon les champs particuliers de chaque classe. Toutes les valeurs décimales doivent être affichées avec deux chiffres après le point décimal.

Cette méthode est dite « *magique* » en PHP dans le sens qu'elle sera automatiquement utilisée à chaque fois qu'une instance d'un objet correspondant est utilisée dans le contexte d'une chaîne de caractères (au lieu de causer une erreur de syntaxe due à l'impossibilité de convertir l'objet en `String`). Par exemple si `$monCylindre` est un objet de type `Cylindre`, alors l'instruction « `echo $monCylindre;` » produira la chaîne ci-dessus au lieu de causer une erreur de syntaxe.

D. Compléter l'implémentation des classes concrètes 1 point

Complétez le code de toutes les classes représentées dans le diagramme de classe UML fourni. Voici les formules pour le calcul des superficies et des volumes des différentes formes 3D :

Classe	Superficie	Volume
`Sphere`	$4\pi r^2$	$(4/3)\pi r^3$
`PaveDroit`	$2(Ll + lh + hL)$	Llh
`Cylindre`	$2\pi r(r+h)$	$h\pi r^2$
`Cube`	$6L^2$	L^3

Où `L` = longueur, `l` = largeur, `h` = hauteur, `r` = rayon

E. Intégration 1 point

Produisez un script PHP dans lequel vous implémentez les fonctionnalités suivantes :

- Mettre deux instances de chacune des classes **concrètes** (8 objets en tout) dans un tableau nommé `$formes` ;
- A l'aide d'une boucle et de la méthode `__toString()`, faire afficher à l'écran l'information détaillée de chaque objet instancié ;
- Tester 3 comparaisons de formes différentes et en afficher le résultat à l'écran ;
- **[Optionnel – chocolat 😉]** Trier le tableau contenant les instances de formes en ordre descendant (de la plus « grande » à la plus « petite » forme) et en faire l'affichage à l'écran.

Remise

Faire la remise sur Omnivox dans un fichier zippé (une seule remise par équipe : indiquez les noms des deux personnes dans le nom du fichier de remise le cas échéant).