HAUTE ÉCOLE DE
NAMUR-LIÈGE-LUXEMBOURG

Catégorie technique

Année académique : 2020-2021

TI215 : programmation orientée objet 1
Exercices : série 5

Méthodes et classes abstraites – Interface
Tableaux et listes (ArrayList)

Consignes

- Déclarez **private** les ***variables d'instance*** de toutes les classes. Prévoyez les **getters** et **setters** nécessaires (et uniquement ceux-là).
- Dessinez au fur et à mesure le **schéma des classes** afin d'avoir une vue d'ensemble de l'exercice.
- Chaque fois que vous avez créé une classe, testez-la ; prévoyez les instructions dans la méthode main pour, au minimum :
  - créer un objet de la classe ;
  - afficher sa description (toString( )) ;
  - appeler les méthodes que vous auriez ajoutées.

5.1

Objectif 1 : Manipuler des classes abstraites

① Créez une classe ***abstraite*** intitulée **Forme** qui contient les méthodes suivantes :
la méthode aire( ) dont le type de retour est un double et qui retourne 0.
la méthode volume( ) dont le type de retour est un double et qui retourne 0.
la méthode ***abstraite*** getForme( ) dont le type de retour est un String.

② Créez une classe intitulée **Cercle** qui est une ***sous-classe de la classe Forme*** et qui contient la variable d'instance:
de type int :
- rayon

Prévoyez un *constructeur*.

Appliquez la méthode *aire()* sur un cercle que vous avez créé. Que renvoie-t-elle ?

Redéfinissez ensuite la méthode *aire() pour qu'elle calcule l'aire du cercle*.
Faites appel à la **constante PI** (constante de classe) de la classe **Math** pour retrouver la valeur de π (**Math.PI**).

Implémentez la méthode *getForme ()* qui doit retourner la chaîne de caractères : ***cercle***. Remarquez que java vous oblige à implémenter la méthode getForme() à moins de la redéclarer abstraite auquel cas la classe Cercle serait abstraite également.

③ Créez une classe intitulée **Rectangle** qui est une ***sous-classe de la classe Forme*** et qui contient les variables d'instance:
de type int :
- largeur
- hauteur

Prévoyez un *constructeur*.

Redéfinissez la méthode *aire( )*.

Implémentez la méthode *getForme ( )* qui doit retourner la chaîne de caractères: ***rectangle***.

④ Créez une classe intitulée **Cube** qui est une ***sous-classe de la classe Forme*** et qui contient la variable d'instance:
de type int :
- largeur

Prévoyez un *constructeur*.

Redéfinissez les méthodes *aire() et volume( )*.

Implémentez la méthode *getForme ()* qui doit retourner la chaîne de caractères : ***cube***.

⑤ Créez une classe **Principal** contenant la méthode **main**. Cette méthode doit :
- créer au moins un cercle, un rectangle et un cube ;
- afficher l'aire, le volume et le nom de la forme de chacun de ces objets ;

5.2

Objectif 2 : Manipuler des interfaces

Modifiez ce qui doit l'être pour que ***Forme soit une Interface*** et non plus une classe.

Adaptez les autres classes si nécessaire.

5.3

Objectif 3 : Gérer une collection d'objets via un tableau d'objets

Placez le cercle, le rectangle et le cube créés pour l'exercice 5.1. dans un **tableau**. Ce tableau doit être déclaré comme un tableau **dont les objets sont de type interface Forme**.

Testez le mécanisme du **polymorphisme** en ***bouclant sur tous les éléments du tableau*** et en affichant la description, le nom de la forme, la surface et le volume ***de chaque élément du tableau :***
Pour chaque élément du tableau (boucle ***for*** à prévoir) :

- afficher le ***nom de la forme*** de l'élément **i** du tableau ;
- afficher l'***aire*** de l'élément **i** du tableau ;
- afficher le ***volume*** de l'élément **i** du tableau ;

5.4

Objectif 4 : Gérer une collection d'objets via la classe ArrayList

Même exercice que le 5.3 mais en utilisant une **ArrayList d'objets de type Forme** au lieu d'un tableau.