ITC315 - Informatique 2

TP 3 - JAVA : Classes abstraites

Wahabou Abdou

Wahabou.abdou@u-bourgogne.fr

2022 - 2023

Note : *L'objectif de ce TP est de définir et de manipuler une classe abstraite. Une telle classe ne permet pas d'instancier des objets. Elle sert de classe de base pour un héritage. Les méthodes abstraites sont juste déclarées (les signatures desdites méthodes sont fournies) dans les classes abstraites. Leur définition est donnée dans les classes filles.*

Exercice 1 : Cercle et rectangle

On souhaite des cercles et des rectangles. La première figure peut être décrite par une épaisseur des traits et son rayon. La seconde quant à elle peut être définie par une épaisseur des traits et ses dimensions (longueur et largeur). Pour les deux figures on aura besoin de calculer le périmètre et la superficie (méthodes `calculPerimetre` et `calculSuperficie`). Sur la base de cette description, nous observons que ces figures présentent, dans leurs définitions, des attributs et méthodes communs. Ces derniers peuvent donc être définis à un niveau d'abstraction plus élevé (au dessus de ce que l'on peut considérer comme étant un cercle ou un rectangle). Nous pouvons donc prévoir une classe `FormeGeometrique`.

1. Définir une classe abstraite `FormeGeometrique` qui contiendra les attributs communs au rectangle et au carré. Les méthodes communes aux deux formes seront abstraites.

```
public abstract class FormeGeometrique {
  ...
  ...
  public abstract double calculPerimetre();
  ...
}
```

2. Faire en sorte que les attributs de `FormeGeometrique` soient accessibles à partir d'éventuelles classes filles, mais pas à partir d'autres classes qui ne seraient pas dans le même package que `FormeGeometrique`. (Indication : utiliser le mot clé `protected`.)
3. Prévoir un constructeur pour la classe `FormeGeometrique`. Celui-ci prendra en paramètre l'épaisseur des traits fixée.
4. Écrire deux classes `Cercle` et `Rectangle` dérivées de `FormeGeometrique` (elles hériteront de la classe `FormeGeometrique`). Définir judicieusement leurs constructeurs.
5. Définir les méthodes `calculPerimetre` et `calculSuperficie` dans les classes `Cercle` et `Rectangle`.
6. Écrire une classe `Main` contenant une méthode `main` permettant de tester les implémentations précédemment faites. Déclarer `forme1` et `forme2` de type `FormeGeometrique`.
7. Peut-on instancier `forme1` de la manière suivante : `forme1 = new FormeGeometrique(1.0)` ? Justifier la réponse donnée.
8. Créer des objets `Cercle` et `Rectangle`. Calculer le périmètre et la superficie pour ces deux objets.
9. Ajouter une méthode abstraite `toString()` dans la classe `FormeGeometrique`. Quel résultat obtient-on après compilation ? Que peut-on déduire ? Définir cette nouvelle méthode dans les classes filles.

Exercice 2 : Tableau de formes géométriques

1. Créer une classe `TableauFormeGeometrique` qui contiendra un attribut entier `nbFormeGeometrique` qui précisera le nombre de formes à enregistrer dans le deuxième attribut `tabFormeGeometrique` (un tableau à une dimension).
2. Définir un constructeur pour cette classe. Celui-ci prendra en paramètre le nombre de formes qui seront ajoutées (la taille du tableau).
3. Définir une méthode permettant d'ajouter une nouvelle forme (objet de type `FormeGeometrique`).
4. Dans la classe `Main`, créer un objet `TableauFormeGeometrique` et lui ajouter plusieurs formes. Ensuite, afficher toutes les formes enregistrées dans le tableau (cette opération fera implicitement appel aux méthodes `toString` précédemment définies).
5. Écrire une méthode qui permet de trier les formes géométriques stockées dans `tabFormeGeometrique` par ordre croissant de superficie.