TP n°7 : classes abstraites et polymorphisme, méthodes statiques

Vous allez étudier dans ce TP les classes abstraites. Elles permettent de faciliter la Programmation Orientée Objet en définissant des fonctionnalités générales à vos objets abstraits (vous écrivez alors la méthode normalement, comme vous l'avez vu jusqu'ici…) ou en n'écrivant que le titre de la méthode. C'est alors aux classes filles qui héritent de la classe abstraite de définir complètement la méthode. C'est ce principe qui rend votre classe abstraite : tout n'est pas défini à l'intérieur de la classe.

Ce TP consiste à créer et à tester une classe abstraite Forme. Celle-ci sera utilisée pour définir des classes filles Rectangle et Cercle.

1. Mise en place

- Créez une classe Principale comportant une méthode main
- Créez une classe Forme comportant un attribut de type Point qui correspond aux coordonnées du centre de la forme
    - Créez un constructeur prenant en paramètre un objet de type Point
    - Créez une méthode bougeForme prenant 2 paramètres et permettant de translater le centre de la forme suivant les valeurs de ses paramètres
    - Compilez et testez la création d'un objet de type Forme

2. Passage à l'abstrait : une classe Forme abstraite et une classe fille Rectangle

Lors de la déclaration de la classe Forme, rajoutez le mot clé *abstract* de la façon suivante :

```
public abstract class Forme {
```

- Compilez et testez la création d'un objet de type Forme. Pourquoi cela ne fonctionne-t-il plus ? Mettez la ligne correspondant à la création de votre objet de type Forme dans votre méthode main en commentaire : à partir du moment où votre classe est devenue abstraite, vous ne pouvez plus instancier d'objet de cette classe !

Une classe abstraite est une classe qui contient au moins une méthode abstraite. Définissez une méthode abstraite dans votre classe de la façon suivante :

```
public abstract double getPerimetre();
```

Notez que vous ne devez pas écrire le code de cette méthode, c'est ce qui rend la classe dans laquelle est définie cette méthode ***abstraite***. Cette méthode devra être redéfinie dans les classes filles.

- Créez une classe Rectangle qui hérite de Forme. Compilez immédiatement votre code et lisez le message d'erreur. Cela devrait vous rappeler des souvenirs.
- Définissez un constructeur à votre classe Rectangle. Ce constructeur doit au moins avoir un paramètre de type Point pour coller à la définition du constructeur de Forme. Bien entendu, vous pouvez ajouter des paramètres spécifiques à votre objet Rectangle. Une largeur et une hauteur par exemple…
- Écrivez la méthode getPerimetre() dans la classe rectangle de telle façon qu'elle retroune la valeur du périmètre de votre rectangle.
- Testez la création d'un objet de type Rectangle dans votre main et retrouvez son périmètre.

3. Une nouvelle forme : une nouvelle classe Cercle

De la même façon que pour la classe Rectangle, créez une nouvelle classe Cercle qui hérite de Forme. Testez la création d'un objet de type Cercle dans votre main.

L'intérêt d'avoir d'abord défini une classe abstraite est que l'écriture de cette nouvelle classe Cercle est simplifiée. Vous avez seulement beoin d'écrire ce qui est spécifique à la création et à l'utilisation d'un objet de type Cercle. De plus, vous avez contraint le developpement de cette classe en imposant une façon de definir le constructeur et en forcant l'écriture de la méthode getPerimetre().

4. Une nouvelle méthode abstraite

Ajoutez la méthode abstraite affiche() dans la classe Forme :

```
public abstract void affiche();
```

Faites les ajouts nécessaires dans les classes Rectangle et Cercle pour que votre code compile et que vous puissiez appeler la méthode affiche() dans votre main sur des objets de type Rectangle et Cercle.

Ce principe de classes abstraites est celui utilisé dans beaucoup d'applications. La phase de conception et de réflexion devient alors essentielle. Si vous voulez développer votre propre « Paint », vous utiliserez cette stratégie pour définir un objet géométrique. Dans un jeu type « angry birds » par exemple, on retrouve aussi ce schéma : on a un type abstrait « oiseau » générique. La création de nouveaux types d'oiseaux devient alors beaucoup plus facile et rapide.

En JAVA, une classe abstraite qui ne possède QUE des méthodes abstraites est une … interface.

5. Une méthode *static*

Dans la classe Forme, définir la méthode suivante :

```
public static void afficheFigures(Vector<Forme> vec) { }
```

Écrivez ensuite le code à l'intérieur de cette méthode qui permet de parcourir le vecteur vec et d'appeler la méthode affiche() sur chaque élément du vecteur.

Testez l'appel à cette méthode dans le main. Vous remarquerez que, pour appeler une méthode static, on ne l'appelle pas sur une instance d'un objet dans lequel est défini la méthode. C'est toute la particularité d'une méthode static qui est utilisée dans des cas bien précis (récupération de données systèmes comme la date et l'heure, renvoie d'info non dépendante de la classe…).

Essayez dans cette méthode static de faire afficher la valeur d'un attribut de la classe. Cela ne fonctionne pas : pourquoi ?

Enfin, souvenez-vous de l'utilisation du mot clé *static* devant un attribut (cf TP 1) qui a pour effet de rendre unique l'attribut pour les instances de classe définies (l'attribut devient global à toutes les instances lorsqu'on ajoute le mot clé static devant). Attention à ne pas confondre les 2 utilisations (questions pièges classiques posées en entretien ou en QCM…) :

- *static* devant un attribut rend la valeur de l'attribut global à toutes les instances de classes utilisées dans votre programme ;
- *static* devant une méthode implique que la méthode ne doit pas être appelée sur une instance de la classe (« la méthode se suffit à elle-même »).