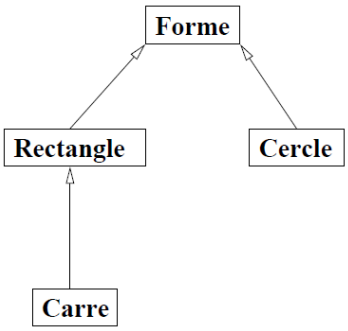
TD04 : Héritage et polymorphisme

Exercice 1 : Une hiérarchie de classes



Considérons les classes `Forme`, `Carre`, `Rectangle` et `Cercle`. L'objectif de cet exercice est d'organiser ces classes de sorte que `Carre` hérite de `Rectangle` qui hérite, ainsi que `Cercle`, d'une classe `Forme`.

Pour le moment, nous considérerons la classe `Forme` comme vide (c'est-à-dire sans aucun attribut ni méthode) et nous nous intéressons plus particulièrement aux classes `Rectangle` et `Carre`.

La classe `Rectangle` héritant d'une classe vide, elle ne peut profiter d'aucun de ses attributs et doit définir toutes ses variables et méthodes.

1. Définir la classe `Rectangle`. Elle doit posséder :
   - Deux attributs privés **longueur** et **largeur** de type `int`.
   - Un constructeur qui prend en argument deux entiers et initialise la longueur et la largeur.
   - Des getters pour la longueur et la largeur.
   - Une méthode `public int surface()` qui renvoie la surface du rectangle.
   - Une méthode publique `public String toString()` qui retourne une description du rectangle (elle doit préciser qu'il s'agit d'un rectangle ainsi que sa longueur et sa largeur).

   La classe `Carre` peut bénéficier de la classe `Rectangle` et ne nécessitera pas la réécriture de ces méthodes si celles-ci conviennent à la sous-classe. Toutes les méthodes et variables de la classe `Rectangle` ne sont néanmoins pas accessibles dans la classe `Carre`. Pour qu'un attribut puisse être utilisé dans une sous-classe, il faut que son type d'accès soit `public` ou `protected`, ou, si les deux classes sont situées dans le même package, qu'il utilise le **type d'accès par défaut**.

2. Définir la classe `Carre` qui hérite de la classe `Rectangle`. Elle doit posséder :
   - Un constructeur qui prend en argument un entier et initialise la longueur et la largeur. Ce constructeur devra faire appel celui de sa classe mère (`Rectangle`) en utilisant la méthode `super()`.

Source : https://perso.telecom-paristech.fr/hudry/coursJava/exercices/objetsCommunicants.html

Remarque :

- L'appel au constructeur d'une classe supérieure doit toujours se situer dans un constructeur et toujours en tant que première instruction ;
- Si aucun appel à un constructeur d'une classe supérieure n'est fait, le constructeur fait appel implicitement à un constructeur vide de la classe supérieure (comme si la ligne `super()` était présente). Si aucun constructeur vide n'est accessible dans la classe supérieure, une erreur se produit lors de la compilation.

- Une redéfinition de la méthode `public String toString()` qui retourne une description du carré (elle doit préciser qu'il s'agit d'un carré ainsi que son côté qui est sa longueur ou sa largeur).

3. Définir une classe `GestionFormes` dans laquelle vous créerez une liste de `Forme`. Vous ajouterez ensuite à cette liste deux instances de la classe `Rectangle` et deux instances de la classe `Carre`.
4. Parcourir la liste et afficher les formes qu'elle contient.
5. Utiliser l'opérateur `instanceof` pour tester l'appartenance d'un objet à une classe comme suit :

```
for(int i=0;i<liste.size();i++) {
      if (liste.get(i) instanceof Forme)
            System.out.println("element " + i + " est une forme");
      if (liste.get(i)  instanceof Rectangle)
            System.out.println("element " + i + " est un rectangle");
      if (liste.get(i) instanceof Carre)
            System.out.println("element " + i + " est un carre");
}
```

6. Compléter votre hiérarchie de classe en définissant la classe `Cercle`. Elle doit posséder :
   - Un attribut privés **rayon** de type `int`.
   - Un constructeur qui prend en argument un entier et initialise le rayon.
   - Un getter pour le rayon.
   - Une méthode `public int surface()` qui renvoie la surface du cercle.
   - Une méthode publique `public String toString()` qui retourne une description du cercle (elle doit préciser qu'il s'agit d'un cercle ainsi que son rayon).
7. Compléter votre méthode `main` en faisant la même chose que pour `Rectangle` et `Carre`.

Source : https://perso.telecom-paristech.fr/hudry/coursJava/exercices/objetsCommunicants.html