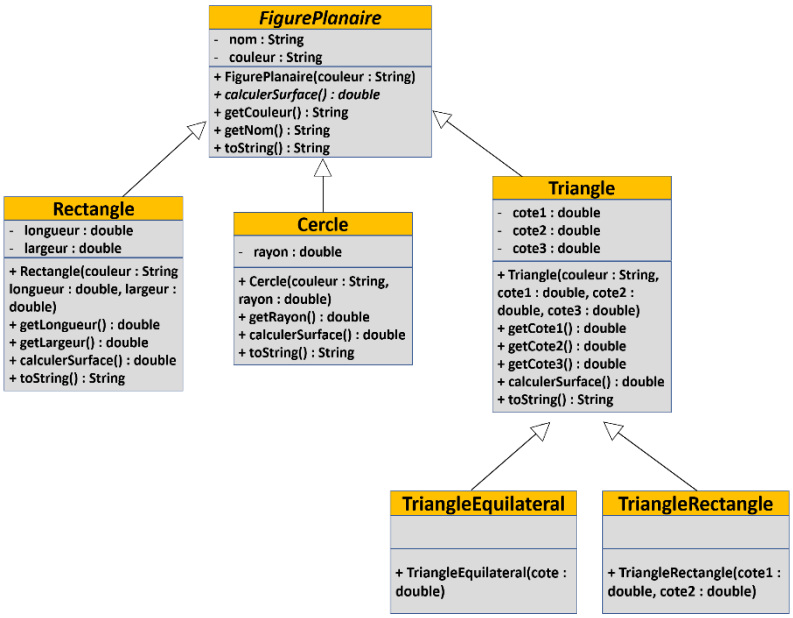
TD 5 Plus sur l'héritage

Dans ce TD nous allons approfondir un peu plus l'idée de l'héritage et nous allons utiliser la notion de classe abstraite. J'inclus à la fin de ce TD un rappel des éléments de géométrie nécessaires (les formules qui donnent les surfaces de différentes figures et la formule pour le volume d'un prisme).

Contexte

Nous allons considérer des diverses figures planaires (par exemple : un cercle, un rectangle, un carré, un triangle, etc.). Nous voulons premièrement calculer la surface de chaque figure, mais en fonction de figure, la surface est calculée selon une formule différente.

Nous partons sur le diagramme de classe suivant :



La classe *FigurePlanaire* sera une classe abstraite (le texte en italique dénote le caractère abstrait d'une méthode ou d'une classe sur les diagrammes de classe) qui aura deux attributs de type String, notamment `nom` et `couleur`. Cette classe aura quelques méthodes concrètes (le constructeur, une méthodes `get` pour les deux attributs et une méthode `toString()` ainsi qu'une méthode abstraite *`surface()`*.

A partir de cette classe nous aurons trois sousclasses concrètes représentant trois figures planaires différentes : un cercle, un rectangle et un triangle. La classe triangle aura elle-même deux sousclasses : `TriangleEquilateral` et `TriangleRectangle`.

Exercice 1

Dans cet exercice le but sera de construire la classe abstraite *FigurePlanaire*.

- Rappel : c'est quoi une méthode abstraite ?
- Rappel : comment écrit-on une méthode abstraite ? (syntaxe)
- Rappel : qu'est qu'une classe abstraite ?
- Vrai ou faux : Une classe abstraite doit obligatoirement contenir au moins une méthode abstraite
- Vrai ou faux : Une classe qui a une méthode abstraite doit impérativement être définie en tant que classe abstraite
- La méthode `toString()` de la classe *FigurePlanaire* retourne : `<nom de la figure>, <couleur>`). Sachant ceci écrivez la classe *FigurePlanaire*.

Exercice 2

Pour ce deuxième exercice, nous allons travailler sur les trois classes de *FigurePlanaire* : `Rectangle, Cercle, Triangle`. Ces trois classes sont concrètes.

- Vrai ou faux : les classes `Rectangle, Cercle` et `Triangle` peuvent, mais ne doivent pas impérativement implémenter la méthode surface.
- Rappel : qu'est-ce que les objets de la classe `Rectangle` héritent de la superclasse *FigurePlanaire* ?
- Vrai ou faux : est-ce qu'on peut avoir le code suivant dans une méthode principale (`main`) ?

```
FigurePlanaire figure = new FigurePlanaire("triangle", "bleu");
```

- En plus des attributs hérités de la classe *FigurePlanaire*, les objets de type `Rectangle` ont deux attributs de type double, notamment largeur et longueur. Le constructeur de la classe `Rectangle` mettra le nom de la figure à `"Rectangle"` et les autres attributs de l'objet aux valeurs mis en paramètre du constructeur.

  Ecrivez ce constructeur.

- Les objets de type Cercle héritent de leur superclasse un attribut `nom` (mis par défaut à `"Cercle"`), une couleur de type `String` et ils ont également un attribut `rayon` de type `double`. La méthode toString() de la classe Cercle doit retourner le String suivant : `"Cercle, <couleur>, rayon = <rayon>"`. Ecrivez cette méthode.
- Les objets de type Triangle héritent de leur superclasse un attribut `nom` (mis par défaut à `"Triangle"`), une couleur de type `String` et ils ont également trois attributs de type double, correspondant à la taille de leurs trois côtés. En utilisant les formules ci-dessous écrivez la

méthode `surface` de la classe `Triangle`. Pour calculer la racine carrée d'un double `x`, on utilise la méthode statique `Math.sqrt(x)`.

- Ecrivez également les méthodes surface des classes Rectangle et Cercle, en sachant que la constante π est retrouvable en utilisant l'instruction Math.PI.

Exercice 3

Dans cet exercice nous allons créer des objets des types mentionnés ci-dessus et nous allons calculer leur surface. Le code dans cet exercice se trouve dans la méthode public static void main(String[] args) d'une classe TD5.

- Ecrivez du code pour créer les figures suivantes :
  - Un cercle rouge de rayon 10
  - Un rectangle bleu 5 par 10
  - Un carré noir de côté 10
  - Un triangle vert avec côtés 3, 5 et 7
  - Un triangle équilatéral marron de côté 3
  - Un triangle rectangle jaune de côtés 5 et 10.

- Nous voulons calculer la surface de chaque figure. À ce but, nous voulons mettre les figures créées dans un tableau qui s'appelle `figures`.
  - Quel sera le type du tableau ?
  - Ecrivez du code qui calcule la surface de chaque forme.

- Dans la méthode main on a cette ligne de code :

```
final FigurePlanaire figure = new FigurePlanaire("Triangle", "marron");
```

  Est-ce que ce code compile ? Pourquoi (ou pourquoi pas) ?
- Même question pour chacune de ces deux lignes de code :

```
System.out.println(cercle.getRayon());
System.out.println(figures[0].getRayon());
```

Exercice 4

Nous allons maintenant étendre notre programme. Nous voulons calculer les modules des prismes avec la base ayant la forme d'une figure planaire donnée. La classe `Prisme` aura deux attributs :

Un attribut `base` de type *`FigurePlanaire`*
Un attribut `hauteur` de type `double`

Elle aura les méthodes suivantes :

Un constructeur avec la signature `Prisme(FigurePlanaire, double)`
Des méthodes `FigurePlanaire getBase()` et `double getHauteur()`
Une méthode `double calculerVolume()` qui calcule le volume d'une prisme
Une méthode `String toString()` qui retourne "`Prisme, hauteur = <hauteur>, base : <base>`".

- Ecrivez le constructeur et les méthodes `calculerVolume()` et `toString()` de la classe `Prisme`
- Dans la classe TD5, affichez pour chacune des figures déjà créées, le volume d'un prisme avec ces figures en tant que base, et une hauteur de 10.

Rappel Math :

Cercle : surface $\pi \cdot r^2$

Rectangle : surface $longueur \cdot largeur$

Triangle : calculez $s = \frac{cote1 + cote2 + cote3}{2}$, et la surface $\sqrt{s(s - cote1)(s - cote2)(s - cote3)}$

Volume prisme : $surface_{base} \cdot hauteur$