Héritage: calcul d'aires de Polygones

13 février

N. Gast, E. Gaussier

Le but de ce TD est d'aborder l'héritage en Java. On fera particulièrement attention à la bonne utilisation des termes `public`, `private` et `protected`.

Exercice 1. Échauffement

Le package `java.awt` contient une classe `Point` représentant un point dans le plan.

a. Écrire une classe `Polygone` représentant un polygone à n cotés dans le plan. Un polygone sera représenté par un tableau de n points. Cette classe comportera les méthodes suivantes :

- `public Polygone(Point[] sommets)()`
- `public double perimetre()`
- `public double aire()`
- `public String nom()`

La fonction `nom` rendra le type du polygone dont il s'agit ("polygone" dans le cas présent). On ne demande pas d'implémenter les méthodes pour l'instant (elle peuvent renvoyer 0).

b. Écrire une classe de Test qui crée un `Polygone` et appelle les différentes méthodes.

Exercice 2. Exceptions

a. Modifier les méthodes `perimetre` et `aire` pour qu'elles lancent une exception `NoSuchMethodException`.

b. Que doit-on modifier dans la classe Test ?

Exercice 3. Héritage

Rappel : faire attention aux termes "private, public, protected".

a. Écrire une classe `Triangle` qui hérite de la classe `Polygone`. Elle comportera un constructeur `Triangle(Point p1, Point p2, Point p3)`

b. De la même façon, écrire une classe `Quadrilatere` qui hérite de la classe `Polygone`. Puis une classe `Rectangle` qui hérite de `Quadrilatere` et implémenter la fonction `nom` sur ces classes.

c. Implémenter la fonction `aire` sur les classes `Rectangle`, `Triangle` et `Quadrilatere`.

d. Pourquoi un des deux codes suivants est correct et l'autre non ?

```
Rectangle r = new Rectangle(sommets);
System.out.println(r.aire());

Polygone p = new Rectangle(sommets);
System.out.println(p.aire());
```

Exercice 4. Pour aller plus loin

Au choix :

a. Écrire une classe `PolygoneRegulier` qui hérite de la classe `Polygone`. Elle comportera un constructeur `PolygoneRegulier(Point origine, int nombreDeCotes, double angleDeDepart)` et implémenter sa fonction `aire`. Implémenter une nouvelle fonction `perimetre`.

b. Créer une nouvelle Exception `MethodesNonImplementees` que l'on lance lorsqu'une méthode n'est pas encore implémentée et qui lance un message d'erreur compréhensible.