01234567890123456789012345

**Université d'Aix-Marseille III - Licence de math-info 2ème année**

**IN301 : Programmation objet - TP n°5 : héritage**

# 1 Chaise en kit

**1.1** *Ecrire la classe* `Plaque` *qui représente une plaque rectangulaire uniquement caractérisée par sa longueur et sa largeur (en centimètres).* Cette classe contient notamment les méthodes publiques suivantes :

- un constructeur qui prend deux entiers en paramètre et attribue le plus grand entier à la longueur de la plaque et le plus petit entier à sa largeur ;
- `Plaque decoupeSensLongueur(int n)`, qui découpe une largeur de plaque de `n` centimètres dans le sens de la longueur (et c'est donc sa *largeur* qui diminue) et retourne la plaque qui en a été extraite. Ex : si on découpe 30 cm dans le sens de la longueur d'une plaque de dimensions $100 \times 200$ , on extrait une plaque de dimensions $30 \times 200$ et la plaque d'origine n'est plus que de dimensions $70 \times 200$;
- `Plaque decoupeSensLargeur(int n)` qui fait la même chose mais dans le sens de la largeur et, le cas échéant, réoriente la plaque (c'est-à-dire, échange les valeurs de la largeur et de la longueur) si sa longueur est devenue plus petite que sa largeur.

**1.2** On veut maintenant écrire la classe `Barreau` qui représente une plaque dont la largeur est normalement de 3 cm et qu'on ne peut découper que dans le sens de la largeur. On va réutiliser la classe `Plaque` pour écrire la classe `Barreau`. *Expliquer d'abord pourquoi ce serait une mauvaise idée de faire hériter la classe* `Barreau` *de la classe* `Plaque`. *Ensuite, écrire la classe* `Barreau` *en la composant avec la classe* `Plaque`. La classe `Barreau` contient notamment les méthodes publiques suivantes :

- un constructeur qui prend en paramètre une plaque ;
- `Plaque decoupeSensLargeur(int n)` qui fait la même chose que celle de `Plaque` ;
- `boolean conforme()` qui indique si la largeur du barreau est bien de 3 cm.

**1.3** *Ecrire la classe* `TabouretEnKit` *qui représente un tabouret constitué d'une plaque normalement carrée servant d'assise et de quatre barreaux servant de pieds.* Cette classe contient les méthodes publiques suivantes :

- un constructeur prenant en paramètre une plaque et quatre barreaux ;
- `int largeur()` et `int hauteur()` qui retournent respectivement la largeur du tabouret, c'est-à-dire la largeur de son assise, et la hauteur du tabouret, c'est-à-dire la longueur d'un de ses pieds ;
- `boolean conforme()` qui indique si l'assise est carrée et les pieds sont tous conformes et tous de même longueur.

**1.4** *Ecrire la classe* `ChaiseEnKit` *qui hérite de* `TabouretEnKit` *et représente un tabouret muni d'une plaque servant de dossier.* Cette classe possède les mêmes méthodes que `TabouretEnKit` sauf que son contructeur prend une plaque (pour le dossier) comme paramètre supplémentaire, que sa conformité nécessite en plus que la largeur du dossier soit la même que celle du tabouret et que la hauteur de la chaise prenne en compte le dossier.

**1.5** *Ecrire la classe* `TesteChaise` *qui contient uniquement la méthode de classe* `main` *contenant le programme principal qui effectue les opérations suivantes : à partir d'une plaque de 50 cm sur 122 cm (122 = 50 + 60 + 4 × 3), une chaise de largeur 50 cm et de hauteur 110 cm est progressivement créée, puis on affiche sa largeur et sa hauteur et si elle est conforme.*

# 2 Les blagueurs

Les blagueurs aiment se raconter des blagues les uns aux autres. Quand un blagueur apprend une blague d'un autre blagueur, il en rit et raconte cette blague à tous ses amis blagueurs. Au départ, un blagueur connaît une seule blague et ne connaît personne. Puis, il va rencontrer des blagueurs et devenir leur ami et pouvoir écouter des blagues et en raconter.

**2.1** *Ecrire la classe* `Blagueur` *qui contient les méthodes suivantes* :

- `protected void reaction()` qui correspond à la réaction qu'a un blagueur lorsqu'on lui raconte une blague. Cette méthode se contentera d'afficher le nom du blagueur suivi de ``Ha ha''.
- `public void presentation(Blagueur b)`. Pour que deux blagueurs se connaissent, il faut qu'ils se présentent l'un à l'autre. Quand deux blagueurs ne se sont jamais rencontrés, un blagueur peut se présenter à l'autre (en appelant cette méthode sur l'autre avec lui-même en paramètre). L'autre blagueur le considèrera alors comme son ami et se présentera lui-même à lui.
- `public void ecoute(Blagueur b, String blague)` qui correspond à ce que fait un blagueur quand un blagueur `b` lui raconte une `blague`. Si le blagueur ne connaît pas cette blague, il y réagit (appel de la méthode `reaction()`), il mémorise cette blague et il la raconte à tous ses amis. Si le blagueur connaît déjà cette blague, il ne fait rien.

N.B : On pourra utiliser des tableaux de taille 100 pour stocker l'ensemble des amis et l'ensemble des blagues.

**2.2** On veut maintenant représenter un pince-sans-rire, c'est-à-dire un blagueur qui ne rit pas. *Ecrire la classe* `PinceSansRire` *qui hérite de* `Blagueur` *et qui sera en sorte qu'un pince-sans-rire aura la seule particularité de ne pas avoir de réaction (il ne rira pas) quand on lui raconte une blague.*

**2.3** On veut maintenant représenter un idiot, c'est-à-dire un blagueur qui n'a pas de réaction (pas de rire) quand on lui raconte une blague, sauf si on lui donne des explications après. *Ecrire la classe* `Idiot` *qui hérite de* `Blagueur` *et qui comporte notamment la nouvelle méthode* `void ecouteExplications()`*. Lorsqu'un idiot exécute cette méthode, il a une réaction normale de blagueur (le rire).*

**2.4** On veut maintenant représenter un boute-en-train, c'est-à-dire un blagueur qui adore raconter des blagues. Quand un boute-en-train se présente pour la première fois à quelqu'un, il a la particularité de lui raconter immédiatement toutes les blagues qu'il connaît. De plus, quand il raconte une blague à un idiot, il lui donne des explications juste après avoir raconté sa blague. *Ecrire la classe* `BouteEnTrain` *qui hérite de* `Blagueur` *et dont le comportement est décrit ci-dessus.*

Vous testerez vos classes grâce à la séquence d'instructions suivante :

```
Blagueur blagueur = new Blagueur("Riri");
Blagueur bouteEnTrain = new BouteEnTrain("Fifi");
Blagueur pinceSansRire = new PinceSansRire("Loulou");
Blagueur idiot = new Idiot("Dingo");
String histoireToto = "C'est l'histoire de Toto...";
String histoireBlonde = "Comment fait une blonde pour...";

System.out.println("----");
blagueur.presentation(bouteEnTrain);
System.out.println("----");
bouteEnTrain.ecoute(blagueur, histoireToto); // Fifi : "Ha ha"
System.out.println("----");
bouteEnTrain.presentation(pinceSansRire); // Loulou : (rien)
System.out.println("----");
bouteEnTrain.presentation(idiot); // Dingo : (rien)
System.out.println("----");
idiot.presentation(blagueur);
System.out.println("----");
bouteEnTrain.ecoute(blagueur, histoireBlonde);
// Fifi : Ha ha, Loulou : (rien), Dingo : (rien), Dingo : Ha ha
```