TP Java n°4 : A l'ouest de Java-Town...

IUP I.I.E.S
23 Fevrier
A. Lemay

Objectifs du TP : Comprendre la notion d'héritage et d'interface. Mots clés `super`, `extends` et `implements`

Exercice 1: Western

On désire réaliser un programme Java permettant d'écrire facilement des histoires de Western. Dans nos histoires, nous auront des brigands, des cowboys, des shérifs, des barmen et des dames en détresses.

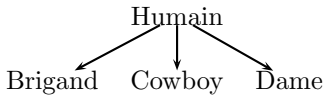
Question 1.1: Tous humains...

Les intervenants de nos histoires sont tous des humains. Un humain est caractérisé par son nom et sa boisson favorite. La boisson favorite d'un humain est, par défaut, de l'eau. Un humain pourra parler. On aura donc une méthode `parle(texte)` qui affiche :
(nom de l'humain) - texte
Un humain pourra également se présenter (il dit bonjour, son nom, et indique sa boisson favorite), et boire (il dira "`Ah ! un bon verre de` *(sa boisson favorite)* `! GLOUPS !`"). Toutes les variables de la classe humain seront privées. Pour connaître le nom d'un humain, on aura besoin d'une méthode `quel_est_ton_nom()` qui renvoie la chaîne contenant le nom de cet humain. De même, on aura besoin d'une fonction pour connaître sa boisson favorite.

Réalisez la classe Humain, dont le constructeur reçoit le nom de l'humain créé en paramètre. Réalisez un programme `Histoire` contenant votre fonction `main`. Dans cet histoire, vous créerez un humain qui se présente et qui boit.

Question 1.2: Brigand, cowboys et dames en détresses

Les dames, les cowboys et les brigands sont tous des humains (si, si !). Par la même, ils ont tous un nom et peuvent tous se présenter. Par contre, il y a certaines différences entre ces trois classes d'individus. Les brigands peuvent kidnapper les dames, les dames peuvent se faire enlever et se faire libérer et les cowboys peuvent les libérer. On a donc le petit schéma suivant :



On dira que la classe `Brigand` est la fille de la classe `Humain` par exemple. On peut indiquer cela en Java. Lorsqu'on crée la classe `Brigand` par exemple, on indique :

```
public class Brigand extends Humain
{
...
}
```

En faisant cela, on dit qu'un brigand est un humain. Par conséquent, il possède un nom, une boisson favorite, une méthode présentation, ... sans avoir besoin de les redéfinir. On dit que la classe `Brigand` *hérite* de la classe `Humain`. On peut par contre ajouter des variables et des méthodes à la classe `Brigand`.

Une dame est caractérisée par la couleur de sa robe (une chaîne de caractère), et par son état (libre ou captive). Elle peut se faire kidnapper (auquel cas elle hurle), se faire libérer par un cowboy (elle remercie alors le héros qui l'a libéré). Elle peut également changer de robe (tout en s'écriant `"Regardez ma nouvelle robe` *`(couleur de la robe)`* `!"`.

Un brigand est un humain qui est caractérisé par un look ("méchant" par défaut), un nombre de dames enlevé, la récompense offerte lorsqu'il est capturé (100 $ par défaut), un booléen indiquant s'il est en prison ou pas. Il peut kidnapper une dame (auquel cas, il s'exclame "`Ah ah !` *`(nom de la dame)`*`, tu es mienne désormais !`". Il peut également se faire emprisonner par un cowboy (il s'écrit alors `"Damned, je suis fait !` *`(nom du cowboy)`*`, tu m'as eu !"` . On dispose également d'une fonction pour connaître la récompense obtenue en cas de capture.

Un cowboy est un humain qui est caractérisé par sa popularité (0 pour commencer, augmente de 1 à chaque dame délivrée) et un adjectif le caractérisant ("vaillant" par défaut). Un cowboy peut tirer sur un brigand (un commentaire indique alors ("`Le` *`(adjectif) (nom)`* `tire sur` *`(nom du méchant)`*`. PAN !`" et le cowboy s'exclame "`Prend ça, rascal !`"). Il peut également libérer une dame (en la flattant).

Réalisez les trois classes `Brigand`, `Cowboy` et `Dame`. Modifiez votre histoire pour tester ces classes.

Question 1.3: surcharge

Au sein d'une classe fille, vous avez appris jusque là à ajouter des attributs et des méthodes. Il est également possible de changer les attributs et méthodes de la classe mère. Pour cela, il suffit simplement de les redéfinir dans la classe fille. On parle alors de surcharge.

Quand on demande son nom à un dame, elle répond `Miss` *`(son nom)`* et un brigand dira *`(son nom)`* `le` *`(son look)`* (par exemple `Bob le méchant`). Un cowboy dira simplement son nom (ce sont des gens simples). Surchargez la méthode `quel_est_ton_nom()` dans les classes `Brigand` et `Dame`.

Testez. Les méthodes que vous avez écrite "remplaceront" (on parle de surcharge) la méthode `quel_est_ton_nom()` de la classe Humain.

Question 1.4: super

On désire aussi changer le mode de présentation des brigands, dames et cowboys. Un brigand parlera aussi de son look et du nombre de dames qu'il a enlevé et de la récompense offerte pour sa capture. Une dame ne pourra s'empêcher de parler de la couleur de sa robe, alors qu'un cowboy diront ce que les autres disent de lui (son adjectif) et parlera de sa popularité.

Si on récrit la méthode qui permet à une personne de se présenter, la méthode de la classe `Humain` sera remplacé. On désire quand même appeler cette méthode. Le brigand Bob se présentera par exemple de la manière suivante :

```
(Bob) - Bonjour, je suis Bob le méchant et j'aime le Tord-Boyaux.
(la méthode de présentation de la classe Humain)
(Bob) - J'ai l'air méchant et j'ai déjà kidnappé 5 dames !
(Bob) - Ma tête est mise à prix 100 $ !
```

Au sein d'une sous-classe d'`Humain`, on peut surcharger la méthode `presentation()`, et appeler la méthode de présentation de la classe mère par la syntaxe `super.presentation()`.

De même, on veut donner une boisson par défaut à chaque sous-classe d'humain (lait pour la dame, tord-boyaux pour le brigand et whisky pour le cowboy). Pour cela, on créé un constructeur pour chacune de ces sous-classes dans lequel on appelle le constructeur de la classe mère (c'est à dire `super.Humain(...)`) avant d'attribuer la boisson par défaut.

Modifiez vos classes en ce sens. Testez.

Question 1.5: Un verre, patron !

Un barman est un humain dont la boisson favorite est le Vin. Il peut servir n'importe quel humain, en lui donnant un verre de sa boisson favorite (les barmen ont un sixième sens pour cela).

Il est caractérisé par le nom de son bar. Par défaut, il s'agira du bar `Chez` *`(nom du barman)`*. La classe barman aura deux constructeurs. Soit on crée un barman en indiquant uniquement son nom, soit on indique également le nom de son bar. Quand un barman se présente, il n'oublie pas de mentionner le nom de son bar. Quand un barman parle, il termine toutes ses phrases par `"Coco."`.

Réalisez la classe Barman. Testez.

Question 1.6: I'm the law

On ajoute à notre histoire des shérifs. Un shérif est un cowboy qui peut coffrer des brigands (en criant `"Au nom de la loi, je vous arrête !"`. Il peut également rechercher un brigand. Un commentaire indique alors qu'il placarde une affiche dans toute la ville, et il dit, par exemple, `"OYEZ OYEZ BRAVE GENS !! 200 $ à qui arretera Bob le brigand mort ou vif !!"`. Tout le monde s'accorde pour dire que les shérifs sont honnêtes. Il est caractérisé par le nombre de brigands qu'il a coffré qu'il ne manquera pas de préciser lorsqu'il se présente. Il refuse de se faire appeler autrement que par `Shérif` *`son nom`*.

Réalisez la classe Shérif. Testez.

Question 1.7: Les shérif sont des cowboys dans l'âme

Comme un shérif est un cowboy, on peut créer un cowboy en faisant :

```
Cowboy Clint =  new Sherif("Clint"),
```

Testez. Quelles sont les fonctions qui sont appelées (quand le cowboy se présente par exemple). Peut-on demander à ce cowboy de coffrer un brigand ?

Exercice 2: Interfaces

On désire faire des histoires de western plus intéressantes, en faisant intervenir des Ripoux. Un ripoux est un shérif qui est brigand. Comment feriez vous ?

On aimerait donc faire en sorte qu'un ripou hérite de deux classes (Shérif et Brigand). Cela n'est pas possible en Java, qui n'autorise que l'héritage "simple" (un seul père). Il est par contre possible de s'en sortir avec la notion d'interface.

Une interface est une espèce de contrat que doit remplir une classe. Elle liste les méthodes que doit posséder une classe qui suit cette interface (qui l'implémente). On crée ainsi une interface Hors_la_loi. Cette interface indiquera qu'une classe qui désigne un hors la loi doit être munie d'une méthode pour kidnapper les dames, d'une autre pour se faire emprisonner par un cowboy, une pour donner son nom, et une autre pour donner la hauteur de sa récompense. On mettra par exemple :

```
public interface Hors_la_loi
{
    public void emprisonne(Cowboy c);

    public void kidnappe(Dame dame);

    public int get_mise_a_prix();

    public String quel_est_ton_nom();
}
```

(à remplacer éventuellement par les nom des méthodes que vous avez utilisées). On indique le *prototype* de chaque méthode, sans mettre de code. On saura que toute classe qui implémente l'interface `Hors_la_loi` possède ces quatre méthodes.

Pour dire qu'une classe suit une interface, on utilise le mot clef `implements`. Par exemple, pour dire qu'un brigand est un hors-la-loi, on mettra :

```
public class brigand extends Humain implements Hors_la_loi
{

}
```

Dans la classe Brigand, il faut être bien sur que toutes les méthodes décrite dans l'interface Hors_la_loi existent !

Question 2.1: Brigand

Écrivez l'interface Hors_la_loi et dites qu'un brigand est un hors-la-loi.

Question 2.2: Les cowboys chassent les hors-la-loi

On peut également utiliser l'interface hors-la-loi dans nos fonctions. Ainsi, un cowboy n'arrête pas seulement les brigands, mais tous les hors-la-loi. De même une dame peut se faire kidnapper par n'importe quelle sorte de hors-la-loi. Modifiez les fonctions concernées pour tenir compte de ceci (vous devriez normalement uniquement remplacer Brigand par Hors_la_loi). Testez.

Question 2.3: Tous pourris

Créez maintenant la classe Ripoux en utilisant l'interface hors-la-loi. Créez un ripou dans votre histoire et testez.

Question 2.4: Calamity Jane

Une femme brigand est une dame qui a décidé de passer de passer du côté des hors-la-loi. Créez la classe Femme_Brigand et testez.

Question 2.5: Ugh !

Un indien est un humain qui est caractérisé par son nombre de plumes (qu'il mentionne quand il se présente) et son totem (Coyote par défaut). Il termine toutes ses phrases par `Ugh !`. Sa boisson favorite est le jus de racine. Un indien peut scalper un visage pale (il peut alors s'ajouter une plume).

Par conséquent, un visage pale est un humain qu'on pourra scalper. Un visage pale disposera donc d'une méthode `scalp()` (il s'écriera alors, par exemple, `Aïe ma tête !`. Réalisez la classe Indien et l'interface VisagePale. En particulier, les brigands, les cowboys et les dames sont des visages pales. Testez.