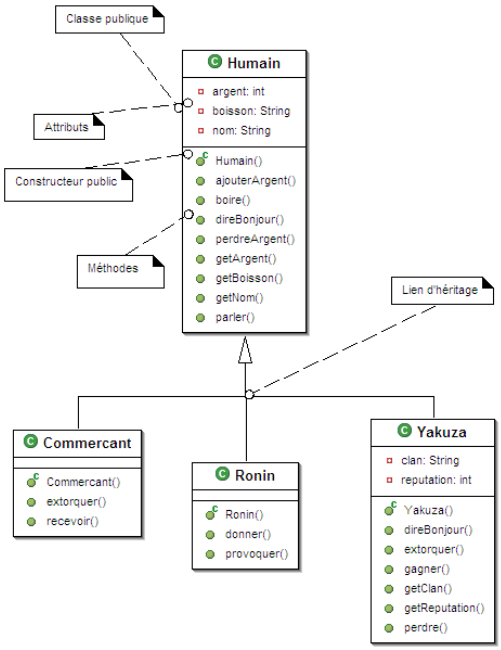
La pierre et le sabre - Java faire mal !

Objectifs *: manipuler les concepts d'héritage, d'interface, de surcharge et de redéfinition*

On désire réaliser un programme Java permettant d'écrire facilement des histoires de Samouraïs dans lesquelles apparaissent des commerçants, des ronins, et des yakuzas.

Ci-contre, le diagramme de classes décrivant les objets et les relations d'héritage que vous allez devoir coder.



➳ Commencez par créer une classe `monHistoire`, qui contient une méthode `main`, et qui vous servira de support pour tester tous les objets et les méthodes que vous allez écrire. Vous trouverez en dernière page du sujet un exemple de programme de test et des sorties attendues.

1 Tous humains !

Tous les intervenants sont des humains. Un humain est caractérisé par son nom, sa boisson favorite, et la quantité d'argent qu'il possède. Tous les attributs de la classe `Humain` sont privés.

Un humain peut :

- parler, via une méthode `parler(texte)` qui affiche : `(nom de l'humain) - texte` ;
- dire bonjour (il dit : `Bonjour ! Je m'appelle ... et j'aime boire du ...`) ;
- boire (il dit : `Ahhh, un bon verre de .... ! GLOUPS !`). Ces méthodes font appel à la méthode `parler`.
- pour manipuler les attributs, un humain propose des accesseurs en lecture (méthodes `get` pour chaque attribut privé ;
- enfin, il a deux méthodes permettant de gagner ou de perdre de l'argent.

➳ Écrivez la classe `Humain` avec ses champs et ses méthodes selon la description précédente. Cette classe possède un constructeur à trois paramètres : le nom de l'humain, sa boisson préférée, et la quantité d'argent qu'il possède.

2 Commerçants, Ronins, et Yakuzas

Les commerçants, les ronins et les yakuzas sont tous des humains, mais avec des particularités supplémentaires. On parle de *spécialisation*, ou d'*héritage* de la classe `Humain`. Pour indiquer cela en Java on utilise le mot clef `extends`, par exemple pour la classe `Ronin` on écrira :

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

Cela signifie qu'un ronin possède tous les éléments humains. Par conséquent, il possède un nom, une boisson favorite, une méthode de présentation... sans avoir besoin de l'écrire à nouveau. On peut par contre ajouter des attributs et des méthodes à cette nouvelle classe.

2.1 Commerçant

Un commerçant est un humain dont la boisson préférée est le thé. Il peut se faire extorquer son argent : il perd tout son argent et il parle pour dire que le monde est vraiment trop injuste. Il peut aussi recevoir de l'argent d'un ronin qu'il remercie alors avec déférence.

➳ Écrivez la classe `Commercant` : codez les méthodes `int seFaireExtorquer()` et `void recevoir(int a)`. Eclipse vous signale une erreur, qu'il vous propose de corriger. Le mot-clé `super` permet d'appeler une méthode (ou le constructeur dans notre cas) de l'objet parent.

2.2 Yakuza

Un yakuza est un humain appartenant à un clan (chaîne de caractères), et caractérisé par sa réputation (un entier par défaut égal à 0). Il peut extorquer un commerçant : il prend tout l'argent du commerçant, gagne un point de réputation et annonce ce qu'il vient de faire. S'il perd dans un duel l'opposant à un ronin, il perd tout son argent, perd un point de réputation, et annonce sa défaite. Dans le cas contraire, il gagne en réputation et crie sa victoire.

➳ Le constructeur de cette classe à les mêmes paramètres que ceux de sa classe mère plus une chaîne de caractères représentant le clan du yakuza. Ajoutez les accesseurs en lecture des deux nouveaux attributs ainsi que les méthodes `void extorquer(Commercant c)`, `void gagner()`, et `int perdre()`.

2.3 Ronin

Le ronin est un humain, il a un attribut d'honneur initialisé à 1. On lui ajoute deux méthodes, la première lui permet de donner de l'argent à un commerçant. La seconde lui permet de provoquer un yakuza en duel. Si le double de l'honneur du ronin est plus grand que la réputation du yakuza, le ronin gagne : il prend l'argent du yakuza, gagne en honneur et annonce sa victoire. S'il perd, on décrémente son compteur honneur, et il râle à cause de sa défaite. En cas de victoire on fait appel à la méthode perdre du yakuza et inversement en cas de défaite on fait appel à la méthode gagner du yakuza.

➳ Écrivez la classe `Ronin`.

3 Redéfinition et surcharge de méthodes

3.1 Le yakuza est fier de son clan

Lorsque le Yakuza dit bonjour on veut qu'il annonce son clan en plus de sa présentation normale. On utilise `super.direBonjour()` afin d'appeler la méthode `direBonjour()` de la classe `Humain`.

➳ Ajoutez la méthode `direBonjour()` dans la classe `Yakuza`, pour redéfinir celle de la classe `Humain`.

3.2 Les samouraïs

Les samouraïs sont des ronins liés à un seigneur (représenté par une chaîne de caractères, contenant le nom). Le nom de son seigneur est donné au constructeur. Lorsqu'un samouraï se présente, il annonce quel seigneur il sert.

Alors que tous les autres humains ne peuvent boire que leur boisson favorite, le samouraï a une super capacité spéciale : il peut boire n'importe quelle boisson. C'est le seul à pouvoir boire du thé en journée, prendre des bières à l'apéro et accompagner son repas de saké. Codez la méthode `boire(String boisson)` qui surcharge celle de `Humain`.

Comme un samouraï est un ronin on peut créer un ronin en faisant :

```
Ronin musaichi = new Samourai("Samourai", "the", 20, "Miyamoto"};
```

➳ Écrivez la classe `Samourai`. Ajoutez une classe de test contenant une méthode `main`. Vérifiez quelles méthodes peuvent être appelées (`boire(String boisson)`, `boire()`, `direBonjour()`...) et quel est le résultat de chaque invocation de méthode.

4 Une belle histoire pour tester...

➽ Ci-dessous vous avez un exemple de code de test. Vérifiez le bon fonctionnement de vos classes, n'hésitez pas à tester toutes les méthodes en améliorant cette histoire pour être sûr que tout fonctionne.

```
public static void main(String[] args) {
  Humain h=new Humain("Prof", 10, "Porto");
  h.direBonjour();
  h.boire();

  Commercant c=new Commercant("Marchant", 35);
  c.direBonjour();

  Yakusa y=new Yakusa("Yaku le noir", 42,
                      "biere", "WarSong");
  y.extorquer(c);

  Ronin r=new Ronin("Roro", 61, "sake");
  r.donner(10,c);
  r.provoquer(y);
  r.direBonjour();
```

```
(Prof) - Bonjour ! Je m'appelle Prof, j'aime boire du Porto et j'ai
10 sous en poche
(Prof) - Mmmm ! Un bon verre de Porto !
(Marchant) - Bonjour ! Je m'appelle Marchant, j'aime boire du thé et
j'ai 35 sous en poche
(Marchant) - J'ai tout perdu ! Le monde est trop injuste...
(Yaku le noir) - J'ai piqué le fric de Marchant
(Roro) - Tiens Marchant voilà 10 sous.
(Marchant) - Je te remercie généreux donateur !
(Yaku le noir) - J'ai perdu mon duel et mes 77 sous, snif...
(Roro) - Je t'ai eu petit yakusa !
(Roro) - Bonjour ! Je m'appelle Roro, j'aime boire du sake et j'ai
138 sous en poche
```

➽ Déssinez un diagramme de séquences représentant les différentes actions de votre histoire.

5 Quelques personnages secondaires

5.1 Les traîtres

Un traître est un samouraï qui peut rançonner des commerçants. Il possède un attribut représentant son niveau de traîtrise, celui-ci étant nul lors de la création d'un traître. Chaque fois qu'il extorque un commerçant sont niveau de traîtrise augmente de 1. Il ne peut plus extorquer si son niveau atteint 3 (vous afficherez des messages en faisant parler le traître). Lorsqu'il dit bonjour, il affiche son niveau actuel de traîtrise. Enfin, il peut faire croire qu'il est gentil (méthode `faireLeGentil()`) en faisant des dons à n'importe quel humain d'une somme d'argent (il dira qu'il fait ami-ami). Son niveau de traîtrise diminue alors du dixième de ce montant (sans jamais descendre en dessous de 0).

➽ Ajoutez la classe `Traitre`

5.2 Les grand-mères

Une grand-mère est un humain assez particulier. Elle a une mémoire de 30 humains, qu'elle peut enrichir en faisant connaissance avec des humains de nos histoires. Ensuite elle passe ses journées à ragoter, en se demandant si untel ou untel est un commerçant, un Ronin, un Samouraï... Par contre elle détecte tout de suite avec son flair incroyable les traîtres. Enfin, une grand-mère ne boit que de la tisane.

Pour représenter sa mémoire, vous pouvez utilisez un tableau d'humains, initialisé avec 30 cases :

```
private Humain[] memoire=new Humain[30];
```

Comme elle ne sait pas vraiment dire qui est qui, nous allons lui fournir une méthode privée `String humainHasard()` qui renvoie aléatoirement une chaîne de caractère contenant "Commerçant" ou "Ronin" ou etc. Pour cela vous utiliserez un objet `Random` (voir la documentation en ligne de Java, dans le paquetage java.lang) et la construction syntaxique `switch`.

Pour détecter si un objet est d'un type particulier, il existe un mot-clé en java `instanceof` qui permet de tester l'appartenance à une classe en particulier :

```
if (monObjet instanceOf maClasse) faireQuelquechose();
```

Utilisez-le pour détecter les traîtres !

➽ Ajoutez la classe `GrandMere`, qui contient deux méthodes publiques : `void faireConnaissanceAvec(Humain h)` et `void ragoter()`. Cette dernière affiche pour tous les humains qu'elle connaît un message du type "je crois que XXX est un Ronin" ou "je sais que YYY est un traître !".

➽ Enrichissez votre programme de test avec de nouveaux personnages et faites contrôler votre travail par votre encadrant de TP.