

TD : En route vers l'héritage ...



Exercice 1 : Association vs Héritage

Association (A-UN) : 

Héritage (EST-UN) : 

Pour chaque énoncé suivant, vous devez :

- ✓ écrire une ou plusieurs phrases qui permet(tent) d'illustrer si la relation entre deux classes (concept) est une **association** ou un **héritage** (en utilisant **A-UN** ou **EST-UN**). Tous les concepts de l'énoncé doivent apparaître dans au moins une phrase.
- ✓ modéliser le **diagramme de classes** correspondant à ces phrases et comportant donc tous les concepts de l'énoncé.

Exemple : Forme, Triangle ?

Réponse attendue :

- une phrase : un Triangle **EST-UNE** Forme (attention écrire bien la phrase dans le bon ordre 😊)

- ET un modèle :



A vous de jouer !

- Félin, Chat ?
- Chirurgien, Docteur ?
- Baignoire, Salle de Bain ?
- Véhicule Terrestre, Voiture, Pneu, Moto ?
- Article, Commande, Ligne de commande ?

Dans ce contexte métier (comptabilité) :



- Livre, Œuvre, Opéra, Film, BD, Roman, Page, Couverture ?

Avant de commencer l'exercice suivant, essayez de répondre à la question suivante :
Quel est à votre avis l'intérêt de l'héritage ?

Exercice 2 : De l'intérêt de l'héritage...

Le service RH (Ressources Humaines) de l'IUT du Limousin vous a sollicité pour que vous lui développiez une application recensant toutes les personnes travaillant ou étudiant actuellement à l'IUT. Votre première activité a été d'**analyser** le problème.

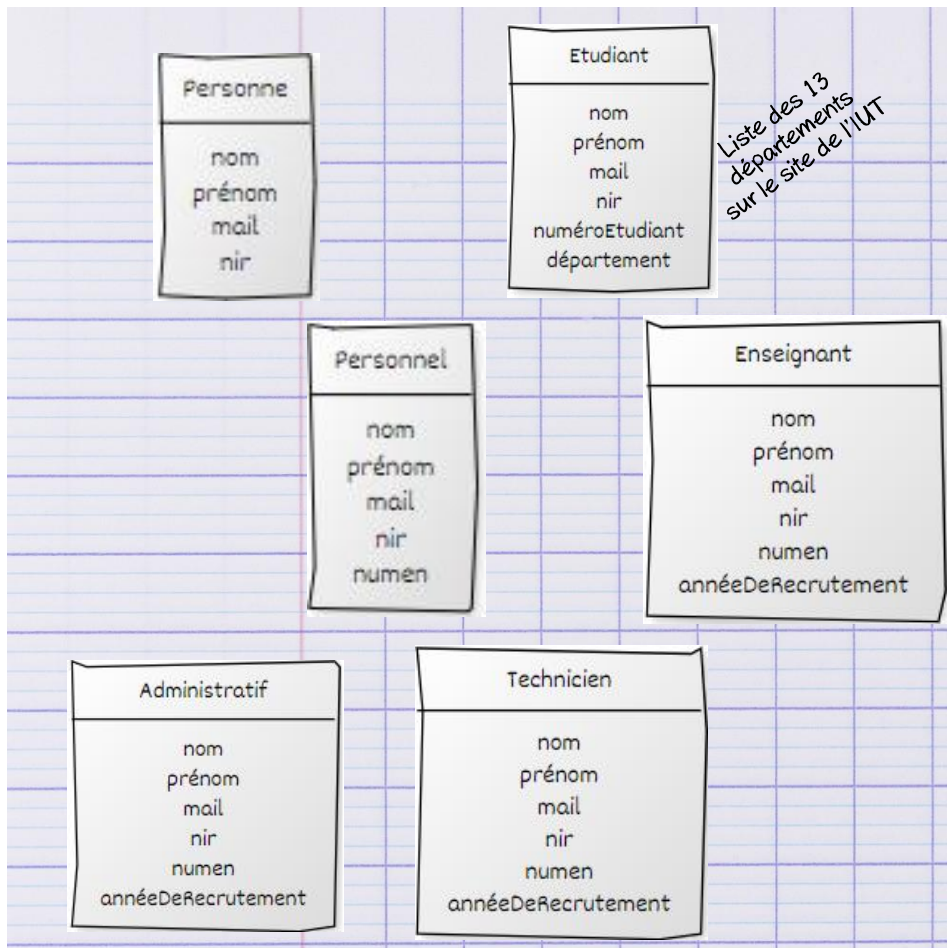
Pour vous avez procédé à un **interview avec les responsables du service RH** dont voici quelques extraits :

« Nous souhaitons pouvoir disposer d'une sorte d'annuaire qui pour chaque personne de l'IUT sera capable de nous communiquer son nom, son prénom, son mail et son NIR (**N**uméro d'**I**nscription au **R**épertoire de l'**I**NSEE) plus couramment appelé le "numéro de sécurité sociale". »

« Alors, il faudra bien faire la différence entre les étudiants et le personnel. Pour les étudiants nous souhaitons connaître leur nom, prénom, mail, NIR, numéro d'étudiants et département d'affectation. A l'IUT du Limousin, il y a actuellement 13 départements, vous retrouverez la liste sur le site de l'IUT »

« Pour chaque personnel, nous devons absolument pouvoir disposer de son nom, prénom, mail, NIR. Ah oui et puis n'oubliez surtout pas le NUMEN qui nous est souvent demandé. Vous connaissez le NUMEN ? C'est le **N**UMéro d'identification de l'**É**ducation **N**ationale qui correspond à un identifiant unique et individuel attribué à chaque personnel de l'Éducation Nationale pour sa gestion administrative et professionnelle. Vous a-t-on dit qu'il y a trois types de personnel : les enseignants, le personnel administratif et les techniciens. L'année de recrutement est également indispensable pour les enseignants, le personnel administratif et les techniciens. Voilà, n'hésitez pas à nous recontacter si vous avez besoin de plus de renseignements et bon courage ... »

Lors de cet entretien, vous avez attentivement écouté vos clients et pris des notes de manière visuelle comme le montre votre feuille :

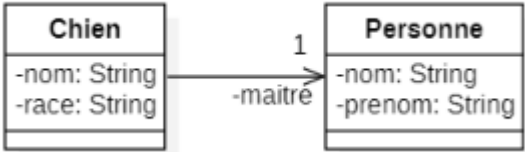


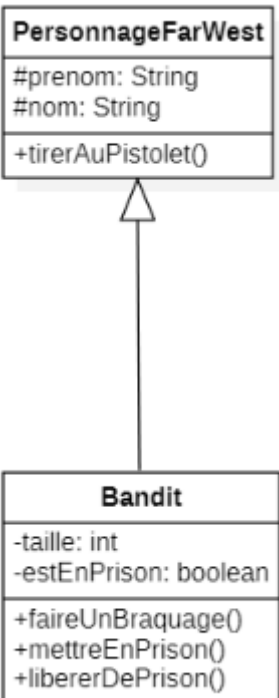
Bien sûr, vous vous êtes ensuite empressé d'aller faire un tour sur le site web de l'IUT pour compléter votre analyse et récupérer la liste des 13 départements (voir annexe)

La première itération de votre phase d'analyse vous ayant permis de mieux comprendre le problème et de récolter les données précédentes, vous décidez maintenant de passer à la **phase de conception**

A partir de vos notes, **proposez un diagramme de classes pour répondre au besoin du service RH. Ce diagramme doit bien sûr être conçu en évitant les redondances.**

Exercice 3 : De la conception à l'implémentation (en Java)...

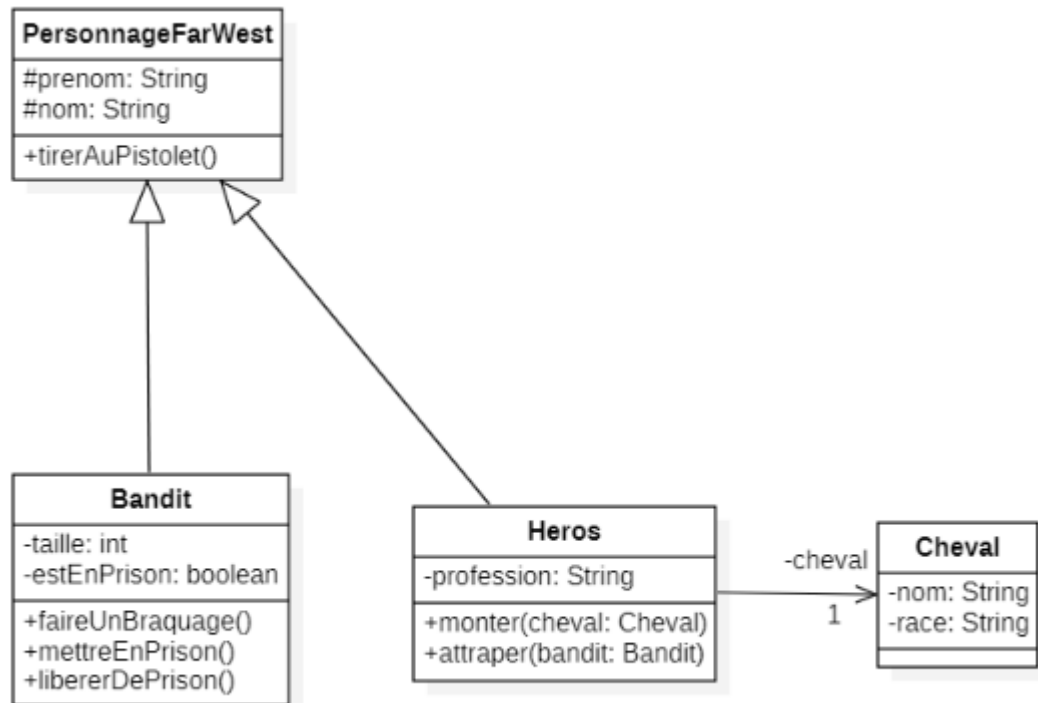
Relation d'association A-UN (Conception)	Déclaration d'un attribut (Implémentation en Java)
 <pre> classDiagram class Chien { -nom: String -race: String } class Personne { -nom: String -prenom: String } Chien "1" --> "1" Personne : -maitre </pre>	<pre> public class Chien { private String nom; private String race; private Personne maitre; // ... constructeur(s), // getteur(s)/setteur(s) au besoin ... } public class Personne { private String nom; private String prenom; // ... constructeur(s), // getteur(s)/setteur(s) au besoin ... } </pre>

Relation de généralisation EST-UN dans le cadre d'un héritage (Conception)	Une classe fille étend une classe mère avec le mot clé extends (Implémentation en Java)
 <pre> classDiagram class PersonnageFarWest { #prenom: String #nom: String +tirerAuPistolet() } class Bandit { -taille: int -estEnPrison: boolean +faireUnBraquage() +mettreEnPrison() +libererDePrison() } PersonnageFarWest < -- Bandit </pre>	<pre> public class PersonnageFarWest { protected String prenom; protected String nom; public void tirerAuPistolet() { // ... } // ... constructeur(s), // getteur(s)/setteur(s) au besoin ... } public class Bandit extends PersonnageFarWest { private int taille; private boolean estEnPrison; public void faireUnBraquage() { // ... } public void mettreEnPrison() { this.estEnPrison = true; } public void libererDePrison() { this.estEnPrison = false; } // ... constructeur(s), // getteur(s)/setteur(s) au besoin ... } </pre>

Restons dans le contexte du Far-West.

Suite au nouveau besoin du client, la conception précédente a évolué de la manière suivante : deux classes **Heros** et **Cheval** ont été ajoutées de la manière suivante.

N'oubliez pas de remarquer que dans notre Far-West, pour le moment, seuls les héros peuvent avoir un cheval (c'est une contrainte du client) 😊



1. Notez **A-UN** ou **EST-UN** sur chacune des relations de ce diagramme.
Proposez une implémentation en Java pour les nouvelles classes **Cheval** et **Heros**
Pour l'instant, vous vous contenterez d'un simple `//TODO` comme implémentation des méthodes `monter` et `attraper`
2. Intéressez-vous maintenant à la construction d'objets du Far-West ...

❑ Construction d'objets de type Cheval :

Complétez votre classe **Cheval** en implémentant le constructeur qui, dans un jeu d'essai, offrira la possibilité de créer un cheval de la manière suivante

```
Cheval jollyJumper= new Cheval ("Jolly Jumper", "appaloosa");
```

→ **Pour chaque attribut** de cette classe, posez-vous la question de savoir si sa *valeur de cet attribut évolue* au cours du temps ou s'il s'agit d'un *attribut immuable* afin de faire apparaître ou non le mot clé **final** lors de la déclaration des attributs de cette classe.

❑ Construction d'objets de type **Heros** :

Complétez votre classe **Hero** en implémentant le constructeur qui offre la possibilité de créer un héros de la manière suivante (en considérant bien sûr que la cheval a été correctement créé précédemment)

```
Hero luckyLuke = new Heros("Lucky", "Luke", "cow-boy", jollyJumper);
```

→ **Pour chaque attribut** de cette classe, posez-vous la question de savoir si sa *valeur de cet attribut évolue* au cours du temps ou s'il s'agit d'un *attribut immuable* afin de faire apparaître ou non le mot clé **final** lors de la déclaration des attributs de cette classe.

❑ Construction d'objets de type **Bandit** :

Complétez votre classe **Bandit** en implémentant le constructeur qui offre la possibilité de créer un bandit de la manière suivante. Bien sûr un objet bandit n'est pas en prison à sa naissance (création) , c'est pour cela que l'on ne passe pas cet état en paramètre, mais le constructeur doit se charger de l'initialiser correctement 😊

```
Bandit joeDalton = new Bandit("Joe", "Dalton", 150);
```

→ **Pour chaque attribut** de cette classe, posez-vous la question de savoir si sa *valeur de cet attribut évolue* au cours du temps ou s'il s'agit d'un *attribut immuable* afin de faire apparaître ou non le mot clé **final** lors de la déclaration des attributs de cette classe.

3. Des constructeurs **d'objets du Far-West** de meilleure qualité 😊

🔊 Le constructeur suivant vient d'être implémenté dans la classe mère **PersonnageFarWest** 🔊

```
public class PersonnageFarWest {  
    //...  
  
    public PersonnageFarWest(String nom, String prenom) {  
        this.nom = nom;  
        this.prenom = prenom;  
    }  
  
    //...  
}
```

→ Modifiez l'implémentation des constructeurs de la classe **Bandit** et de la classe **Heros** afin de minimiser la duplication de code en faisant directement appel à ce constructeur 😊

→ **Pour chaque attribut** de **PersonnageFarWest**, posez-vous la question de savoir si sa *valeur de cet attribut évolue* au cours du temps ou s'il s'agit d'un *attribut immuable* afin de faire apparaître ou non le mot clé **final** lors de la déclaration des attributs de cette classe.

Annexe : Extrait de <https://www.iut.unilim.fr/les-formations/but/>

Pôle Tertiaire

GEA

Gestion des Entreprises et
des Administrations - LIMOGES

📍 Limoges

>> Téléchargez la plaquette

>> Accédez à la fiche web détaillée

TC

Techniques de
Commercialisation

📍 Limoges

>> Téléchargez la plaquette

>> Accédez à la fiche web détaillée

GEA

Gestion des Entreprises et
des Administrations - BRIVE

📍 Brive

>> Téléchargez la plaquette

>> Accédez à la fiche web détaillée

CS

Carrières Sociales

📍 Guéret

>> Téléchargez la plaquette

>> Accédez à la fiche web détaillée

Pôle Numérique

MMI

Métiers du Multimédia et
de l'Internet

📍 Limoges

>> Téléchargez la plaquette

>> Accédez à la fiche web détaillée

INFO

Informatique

📍 Limoges

>> Téléchargez la plaquette

>> Accédez à la fiche web détaillée

Pôle Industriel

GB

Génie Biologique

📍 Limoges

>> Téléchargez la plaquette

>> Accédez à la fiche web détaillée

GMP

Génie Mécanique et
Productique

📍 Limoges

>> Téléchargez la plaquette

>> Accédez à la fiche web détaillée

MP

Mesures Physiques

📍 Limoges

>> Téléchargez la plaquette

>> Accédez à la fiche web détaillée

HSE

Hygiène, Sécurité,
Environnement

📍 Tulle

>> Téléchargez la plaquette

>> Accédez à la fiche web détaillée

GIM

Génie Industriel et
Maintenance

📍 Tulle

>> Téléchargez la plaquette

>> Accédez à la fiche web détaillée

GEII

Génie Électrique et
Informatique Industrielle

📍 Brive

>> Téléchargez la plaquette

>> Accédez à la fiche web détaillée


GCCD


Génie Civil
Construction Durable

📍 Éggletons

>> Téléchargez la plaquette

>> Accédez à la fiche web détaillée

	<p align="center">A revoir ... (Notions mises en pratique dans ce TD)</p>
	<p>→ Dans la rubrique Structure d'une classe Accès direct à la rubrique via : https://unil.im/butoo2 (URL : https://www.youtube.com/playlist?list=PLzzeuFUy_Cnh_jAwFXkYMjRd9wYhObqAL)</p> <p align="center">❑ 03. Modificateur de visibilité : protected</p>
	<p>→ Dans la rubrique Constructeur d'objet Accès direct à la rubrique via : https://unil.im/butoo4 (URL complète : https://www.youtube.com/playlist?list=PLzzeuFUy_Cni3_xF9bI5oNDvrc757-4ih)</p> <p align="center">❑ 03. Construction d'objet qui héritent d'autres objet, constructeur vide par défaut</p>

	<p align="center">A (re)découvrir (pour se préparer au prochain TD et TP)</p>
	<p>→ Dans la rubrique Programmation Objet : Encapsulation, Héritage et Polymorphisme en Programmation Objet Accès direct à la rubrique via : https://unil.im/butoo5 (URL complète : https://www.youtube.com/playlist?list=PLzzeuFUy_CniZpKCGfQ9HpJFxfkqGeGFO0)</p> <p align="center">❑ 03. Héritage de type entre classes et interfaces (8 :15) <i>(pour l'instant faites abstraction de la partie implements Comparable que nous détaillerons plus tard. Tant que vous n'avez pas vu la notion d'interface, pour simplifier la compréhension de cette vidéo, considérez qu'un implements joue à peu près le même rôle qu'un extends 😊)</i></p> <p align="center">❑ 04. Héritage de comportement : utilisation des méthodes d'une super-classe</p> <p align="center">❑ 05. Résolution d'un appel de méthode polymorphe</p>
	<p>→ dans la rubrique Classes, classes abstraites et interfaces Accès direct à la rubrique via : https://unil.im/butoo6 (URL complète : https://www.youtube.com/playlist?list=PLzzeuFUy_CniTo0Pm8Tdh7MVVYhF32fdx)</p> <p align="center">❑ 01. Ecriture et utilisation d'une classe abstraite Operation</p> <p align="center">❑ 02. Extension concrète d'Operation et utilisation</p>