TD2 de Java - L'héritage.

Exercice 1 - Création d'une classe représentant une personne

Nous allons créer une classe qui permet de représenter une personne, qui possède nom, prénom, et âge. Nous créerons ensuite deux classes filles de la classe `Personne` : une classe `Chanteur` et une classe `Auteur`.

1. Créez une classe `Personne` qui aura quatre attributs privés : une chaîne de caractères `prenom`, une chaîne de caractères `nom`, un entier `age`, une profession (chaîne de caractères).
2. Dotez la classe `Personne` de **constructeurs**. Ils pourront admettre trois arguments (on définit l'âge, le nom et le prénom de la personne que l'on instancie, la professions est mise à ""), ou bien deux arguments (l'âge n'est pas précisé et est alors mis à "-1", ce qui signifie non défini, la profession est mise à ""), ou quatre arguments (tout est défini dès l'instanciation).
3. Créez une méthode `sePresenter()` qui permet à une personne de se présenter : elle renvoie donc une chaîne de caractères "Je m'appelle <prenom> <nom>, j'ai <age> ans" ou bien "Je m'appelle <prenom> <nom>" si l'âge n'est pas défini.
4. Créez une classe `Monde` qui contient le `main`, créez deux personnes et faites-les se présenter (affichages à l'écran).
5. Pouvez-vous faire s'afficher le nom, et seulement le nom (sans utiliser `sePresenter()` donc) d'une personne, dans la classe `Monde` ? Créez donc une méthode **publique** `getNom()` qui retourne une chaîne de caractères (le nom) et servez-vous en dans `Monde` pour afficher le nom d'une des personnes que vous avez créées.

Exercice 2 - Une classe fille : Chanteur

1. Créez une classe `Chanteur` qui hérite de la classe Personne : le constructeur (commencez à créer un constructeur à trois arguments, nom, prénom, et âge) doit faire appel au constructeur de la classe mère (`super()`) et définir la profession, qui vaudra la chaîne "chanteur".
2. Définissez dans cette classe une méthode `sePresenter()` qui redéfinit la méthode `sePresenter()` de la classe mère en rajoutant à la fin "et je suis chanteur". Là encore, pour désigner la méthode de même nom de la classe mère vous pourrez utiliser `super` : `super.sePresenter()` désigne la méthode `sePresenter()` de la classe mère.
3. Testez : créer dans `Monde` un chanteur `c`, faites-le se présenter. Maintenant créez une personne p de la sorte :`Personne p=c ;` où `c` désigne le chanteur précédemment créé. Faites se présenter cette personne `p`. Quelle méthode `sePresenter()` est appelée : celle de la classe `Personne` ou celle de la classe `Chanteur` ?.
4. Essayez d'afficher, dans la classe `Monde`, le nom, et seulement le nom du chanteur. Est-il nécessaire de redéfinir la méthode `getNom()` dans la classe chanteur ?
5. Rajoutez un attribut privé à la classe `Chanteur` : maisonEdition (chaîne de caractères) et écrivez une méthode `setEdition()` qui prend en paramètre une chaîne de caractères (le nom de la maison d'édition) qui permet d'attribuer une valeur à cet attribut, et une méthode `getEdition()` qui retourne une chaîne de caractères : le nom de la maison d'édition du chanteur, s'il en a une, une chaîne vide sinon.
6. Dans la classe `Monde`, essayez d'afficher la maison d'édition de la personne `p` précédemment créée. Pouvez-vous le faire ? Il faut considérer `p` comme chanteur et non comme personne, nous allons utiliser le **polymorphisme** de java : au lieu de `p`, écrivez `(Chanteur)p` là où `p` doit avoir la nature d'un chanteur.

Exercice 3 : Classe maison d'édition (ensemble de chanteurs, utilisation d'ArrayList)

1. Créez une classe `MaisonEdition` dont les attributs sont pour l'instant un simple nom et un tableau de chanteurs.
2. Créez des constructeurs de cette classe, et une méthode permettant d'ajouter un chanteur à la maison d'édition et une méthode permettant d'en rajouter un. Vous constaterez que ce n'est pas forcément très pratique avec un simple tableau de chanteur : utilisez alors la classe `ArrayList` pour représenter l'ensemble des chanteurs : `private <Chanteur>ArrayList chanteurs ;`. Et pour créer ce tableau : `chanteurs=new ArrayList() ;`. Une variable de la classe tt ArrayList possède de nombreuses méthodes (ajouter un élément, en retirer un...) il vous faut consulter la documentation pour voir lesquelles sont disponibles :
`http ://java.sun.com/j2se/1.4.2/docs/api/java/util/ArrayList.html`.

Exercice 4 : tableaux, attributs objets

1. **tableaux** : nous allons rajouter deux attributs privés dans la classe chanteur : `titres`, tableau des titres qu'il a créés (c'est donc un tableau de `String`) et `nbTitres`, un entier qui indique le nombre de titres qu'il a créé. Changez les constructeurs de sorte à ce qu'ils initialisent ce tableau (10 titres max) et cet entier, créez une

méthode qui permet d'ajouter un titre à ce tableau, et enfin une méthode qui permet d'afficher la liste de tous les titres créés par un chanteur.

2. Ce tableau est imparfait : il ne contient que les titres, et non les dates de création de ces chansons, le nombre d'exemplaires vendus, etc... De plus, sa taille n'est pas dynamique (il faut l'initialiser à une certaine taille maximale, et utiliser la variable nbTitres pour savoir le nombre de disques présents dans le tableau ; retirer un titre n'est pas non plus facile). Pour faire mieux on pourrait créer une classe Disque et l'utiliser... Si vous avez du temps, explorez cette idée. L'autre solution sera d'utiliser la classe `Vector`.
3. Rajoutez à la classe `Personne` un attribut de type `Personne` : c'est le conjoint éventuel. Rajouter une méthode `marier()` qui permet de se marier à une personne : combien d'arguments d'entrée sont nécessaires ?
4. Dans `Monde`, créez une personne que vous marierez avec votre chanteur `c`. Créez dans la classe `Personne` une méthode qui permet d'afficher si une personne est mariée, et si oui, le nom de son conjoint.