TP 3 (première partie : l'héritage)

Exercice 1/

Attention : l'encapsulation devra être systématique (utilisation du mot-clé private).

Question 1/ Créez une classe `Sport` avec 2 variables d'instance *code* et *libellé*.
Ecrivez un constructeur qui reçoit les valeurs initiales en paramètres.
Ecrivez les accesseurs, une méthode *public String toString()*, et une méthode pour l'affichage (utilisant *toString()*).

Question 2/ Un sport collectif a **en plus** un nombre de joueurs**.**
Créez une classe `SportCo` qui hérite de la classe `Sport` et qui possède une variable d'instance *nbJoueurs.*
Ecrivez les accesseurs et une méthode *public String toString()*.

Question 3/ Créez une classe `LesSports` ayant en variable d'instance une instance de la classe `ArrayList` de `Sport,` et des méthodes d'instances permettant :
- d'ajouter un sport dans cette `ArrayList`
- de lister les sports enregistrés dans cette `ArrayList`.

Question 4/ Dans une classe `ProgSports`, écrivez un *main()*, qui crée une instance de `LesSports` de MAX sports, puis ajoute cinq sports, dont trois sports collectifs, et les affiche.

Exercice 2/

Attention : l'encapsulation devra être systématique (utilisation du mot-clé private).

On s'intéresse à une course de chevaux.

Un cheval est caractérisé par un numéro, un nom, un sexe, une race.
Un cheval de course possède en plus un entraîneur, un jockey et un montant des gains.

L'entraîneur est une personne qui dispose d'un numéro de licence d'entraîneur (String).
Un jockey est une personne qui a un poids et un salaire.

Une personne est décrite par son nom, son prénom et son adresse.

Une course est caractérisée par un nom, une dotation (en Euros) et un nombre et une liste de partants.

Question 1/ Ecrire une classe `Personne` ayant comme variables d'instances le nom, le prénom et l'adresse de la personne. Prévoir le constructeur (avec paramètres), les accesseurs et la méthode *toString()* .

Ecrire les sous-classes `Entraineur` et `Jockey`.

Ecrire un *main()* testant ces classes.

Question 2/ Ecrire une classe `Cheval` ayant comme variables d'instance un numéro d'inscription au service d'identification des équidés, qui devra être géré automatiquement (utiliser une variable de classe et l'affecter à la variable d'instance *numero*), un nom, un sexe, une race.
Prévoir le constructeur, les accesseurs et la méthode *toString()*.

Ecrire la sous-classe `ChevalDeCourse`.

Prévoir le constructeur permettant de créer une instance de `ChevalDeCourse` possédant un numéro, un nom, un sexe, une race, un montant des gains et un entraineur , mais pas encore de jockey , et les méthodes
affiche() qui affiche les renseignements sur un cheval de course,
attribue_jockey (Jockey j) qui attribue le jockey *j* au cheval de course.

Tester ces classes.

Question 3/ Ecrire une classe `Course` ayant en variables d'instance le nom de l'épreuve, sa dotation, une instance d'`ArrayList` de `ChevalDeCourse` .

Prévoir le constructeur qui crée une course sans chevaux initialement, et les méthodes :

- affiche() qui affiche les renseignements sur les chevaux de la course (avec le nom et la dotation de celle-ci),

- chevalPresent(String nom) qui retourne vrai si le cheval de nom *nom* est présent dans la course, faux sinon (utilisation de la méthode equals de la classe `String`).
Le programme de test de cette méthode doit afficher la valeur attendue et la valeur obtenue, pour chacun des 4 cas suivants : un cas absent, un cas moyen, les extrêmes.

- enregistre(`ChevalDeCourse` c) qui enregistre le cheval de course *c* dans la course (on utilisera la méthode précédente pour qu'il n'y ait pas d'homonymes).

Tester cette classe.

Question 4/ Dans la classe `Course`, ajouter une méthode recherche_cheval (`String` nom) qui affiche les renseignements sur le cheval de nom *nom*, ou un message *« absent »* sinon.

Tester-la.
Le programme de test doit afficher la valeur attendue et la valeur obtenue, pour chacun des 4 cas suivants : un cas absent, un cas moyen, les extrêmes.