Classes, attributs, et méthodes en UML et Java

Exercice 1 *Classe Étudiant*

La scolarité souhaite modéliser puis développer (en Java) une classe Etudiant commune à ses divers systèmes d'information. On s'intéresse ici à une version préliminaire de cette classe, où devront apparaître les informations suivantes :

- l'âge
- la date de naissance (dans un premier temps, elle ne contiendra que l'année de naissance)
- un attribut `codeIns` déterminant si c'est la première inscription de l'étudiant ou s'il s'agit d'une réinscription
- un attribut `codePays` permettant de distinguer les étudiants français, les étudiants étrangers non francophones et les étudiants étrangers francophones
- 3 notes obtenues à 3 examens
- les accesseurs associés aux attributs précédents
- un constructeur dont les arguments permettent de garnir les attributs précédents
- une méthode permettant le calcul de la moyenne obtenue
- une méthode permettant le calcul de la mention obtenue
- une méthode `ligneResultats` qui retourne une chaîne d'une ligne précisant le nom, la moyenne et la mention, et, seulement s'il est ajourné, les modules obtenus (c'est-à-dire ceux où il a obtenu la moyenne)

Question 1. Proposez une modélisation UML de la classe Etudiant (qui nécessitera peut-être d'introduire de nouvelles classes que vous modéliserez également).

Question 2. Proposez une implémentation en Java de votre modélisation.
On veillera à bien respecter les règles suivantes :

- les attributs sont privés et sont accédés grâce à un couple d'accesseurs
- les méthodes et les attributs sont tous commentés
- le code est bien indenté
- le code est soigneusement testé en développant une classe dédiée au test, et qui manipule autant d'instances d'étudiants que vous le jugerez nécessaire
- les noms des classes commencent par une majuscule, les noms des packages, attributs et méthodes par une minuscule.

Question 3. Toutes les classes disposent implicitement d'une méthode `String toString()` qui retourne une chaîne de caractères dont le rôle est de représenter une instance ou son état sous une forme lisible et affichable. Si on ne définit pas de méthode `toString` dans

une classe, la méthode par défaut est appelée, elle retourne une désignation de l'instance. Il est conseillé de définir une méthode `toString` pour chaque classe.

a- Utilisez la méthode `toString` présente par défaut dans la classe Etudiant. Que renvoie-t-elle ?

b- Écrivez pour chacune des classes que vous avez implémentées une méthode `toString`. Testez ces méthodes.

Question 4. Gestion plus fine de la date de naissance et de l'âge. Vous terminerez par la gestion plus fine de l'âge et de la date de naissance. Pour un attribut ou une variable représentant une date, on utilisera la classe Date présente dans le package java.util. Pour tester la classe Etudiant, vous aurez besoin de créer des dates, correspondant aux dates de naissance des étudiants. Pour cela, on utilisera la classe DateFormat du package java.text. Plus particulièrement, on pourra s'inspirer des lignes suivantes :

```
// on récupère un formatteur de date, qui gère des formats longs
DateFormat df = DateFormat.getDateInstance(DateFormat.LONG);
// on met dans d la date correspondant au 03/07/04.
Date d=df.parse(``03 juillet 2004'');
```

Pour récupérer la date courante, on utilise simplement le constructeur sans paramètre de la classe Date. Pour calculer l'âge, il sera utile de récupérer l'année d'une date, le mois d'une date, et le jour d'une date. On s'inspirera pour cela du code :

```
// création d'un calendrier grégorien
// permettant de récupérer les informations d'une date
Calendar cal=new GregorianCalendar();
cal.setTime(d); // On remplit le calendrier avec la Date d
int annee=cal.get(Calendar.YEAR); // on récupère l'année
int mois=cal.get(Calendar.MONTH); // on récupère le mois
int jour=cal.get(Calendar.DAY_OF_MONTH); // on récupère le jour
```

Exercice 2 *Pour ceux qui ont fini en avance ...*

Implémentez les classes `Rectangle` et `Tortue/Espece` vue en TD, et testez votre implémentation.