TP 3 : Classe et objet

Classe, objet, attribut, méthode, constructeur, composition, test unitaire automatisé avec JUnit.

Indications préliminaires

Vous définirez la classe Personne dans un paquet personne et la classe exécutable dans un paquet main.
Vous respecterez les conventions de programmation en Java recommandées par Sun.

Exercice – Généalogie

1. Ecrivez une classe Personne permettant de décrire complètement une personne, sachant que l'on souhaite avoir autant d'informations que dans la chaîne suivante : "M. Harry Cover né en 1976, célibataire.", telle qu'à chaque création de personne doit apparaître un message de ce type : «Naissance de la 23ème personne.». Vous écrivez en particulier pour la classe Personne :

   - un constructeur ;
   - une méthode toString qui retourne une chaîne de caractères similaire à la phrase donnée ci-dessus ;
   - une méthode afficheAge qui affiche une phrase donnant l'âge de la personne en fonction d'une année donnée en paramètre. La méthode gère le temps de conjugaison du verbe avoir (en se basant sur l'année courante) et le nombre du mot an (singulier ou pluriel) ; par exemple, selon l'année donnée en paramètre, la méthode retournera une des phrases de la forme suivante (lorsque l'année courante est 2017) :

     M. Harry Cover avait 1 an en 1977.
     M. Harry Cover avait 26 ans en 2002.
     M. Harry Cover aura 44 ans en 2020.
     M. Harry Cover a 41 ans en 2017.
     M. Harry Cover est né en 1976.
     M. Harry Cover n'était pas né en 1900.
   - une méthode afficheAge sans paramètre qui affiche une phrase donnant l'âge de la personne par rapport à l'année courante, exemple lorsque l'année courante est 2017 : M. Harry Cover a 41 ans en 2017.
   - une méthode getAge qui retourne l'âge de la personne par rapport à l'année courante.

   Comment connaître le nombre de personnes créées, sans avoir sous la main une instance de Personne ?

   Ecrivez un programme de test pour tester votre classe.

   Corrigez votre classe Personne de sorte qu'elle passe les tests JUnit fournis dans PersonneTest.java. **Vous ne devez pas modifier la classe PersonneTest**, mais vous pouvez ajouter dans la classe Personne les méthodes minimales (dont le corps est réduit au minimum) nécessaires pour que les tests passent. Remarquez l'annotation @Ignore utilisée dans PersonneTest pour désactiver temporairement certains tests.

   Etudiez la classe de test et particulièrement la façon dont est défini le contexte commun des tests dans la méthode annotée par @Before.

2. Ajoutez à la classe Personne un attribut conjoint et examinez les conséquences que cela peut avoir sur l'ensemble du code. Attention, il est impossible d'avoir plusieurs conjoints.

   Ajoutez une méthode marier(Personne p) qui permet de marier une personne à une autre. Modifiez la méthode toString() de sorte que le nom devienne : "[les deux noms des conjoints par ordre alphabétique] né(e) [nom de naissance]". Par exemple, si Mlle Homalie se marie avec M. Térieur, son nom deviendra Mme "Homalie-Térieur née Homalie". Et le nom de M. Térieur deviendra M. "Homalie-Térieur née Térieur".

   Activez la méthode de test testMarier et corriger votre classe Personne de sorte qu'elle passe les tests JUnit de la méthode testMarier().

3. Ecrivez une méthode d'instance int compareTo(Personne p) permettant de comparer deux personnes par leur âge, qui renvoie 0 si la personne a le même âge que la personne spécifiée, une valeur négative si elle a un âge inférieur à celui de la personne spécifiée, une valeur positive sinon. Ecrivez une méthode de classe int compareTo(Personne p1, Personne p2) pour comparer l'âge de deux personnes. Testez les méthodes en utilsant les tests Junit testCompareToPersonne() et testCompareToPersonnePersonne().

4. On veut maintenant être en mesure de déterminer si une personne est l'ancêtre d'une personne donnée. Pour cela, ajoutez à la classe Personne deux attributs pere et mere. Ecrivez le code du constructeur permettant de créer un objet de type personne en lui passant son père et sa mère lors de l'initialisation (ou null si l'on ne connaît pas le parent).

   Vous devrez avoir défini au moins deux constructeurs de Personne. Dans votre implantation, l'un des constructeurs appelle-t-il l'autre ? Modifiez vos constructeurs pour que ce soit le cas.

   Dans la classe PersonneTest, remplacez l'instruction initAvantQuestion4(); par initAPartirDeQuestion4(); dans la méthode annotée par Before et enlevez les commentaires devant la méthode initAPartirDeQuestion4(). Certaines personnes sont maintenant créées avec des ancêtres, de la façon suivante :

   - M. Alex Térieur, époux de Anne Hommalie, de père inconnu et de mère inconnue
   - Mlle Anne Homalie, épouse de Alex Térieur, de père inconnu et de mère inconnue
   - M. Alain Térieur, époux de Félicie Tassion, de père Alex Térieur et de mère Anne Homalie
   - Mlle Félicie Tassion, épouse de Alain Térieur, de père inconnu et de mère inconnue
   - Mlle Sarah Tatouille, de père inconnu et de mère inconnue
   - M. Aéropos Térieur, de père Alain Térieur et de mère Félicie Tassion
   - Mlle Lara Tatouille, de père inconnu et de mère Sarah Tatouille
   - M. Enguerrand Térieur, de père Aéropos Térieur et de mère Lara Tatouille

   Ecrivez une méthode estAncetre(Personne p) qui retourne vrai si la personne courante est ancêtre de la personne p. Remarque : p peut provenir de la branche paternelle ou de la branche maternelle à chaque génération.

   Testez avec le test Junit de la méthode testEstAncetre().

5. On désire afficher l'arbre généalogique d'une personne, ainsi, pour Enguerrand Térieur :

```
Enguerrand Térieur
|___de père Aéropos Térieur
        |___de père Alain Térieur époux de  Félicie Tassion
                |___de père Alex Térieur époux de Anne Homalie
                |___de mère Anne Homalie épouse de Alex Térieur
        |___de mère Félicie Tassion épouse de Alain Térieur
|___de mère Lara Tatouille
        |___de mère Sarah Tatouille
```

   Donnez le code de la méthode void afficherAieux() de la classe Personne qui affiche l'arbre généalogique d'une personne avec une bonne indentation.

   Testez avec le test Junit de la méthode testAfficheAieux().

6. Définissez une méthode equals(Object o) pour vérifier l'égalité de deux personnes. Testez avec le test Junit de la méthode testEquals(). Vérifiez que tous les tests de PersonneTest. sont actifs (sans annotation @Ignore), et votre classe Personne doit maintenant passer tous les tests.

7. Générer la javadoc de la classe PersonneTest et observez le résultat obtenu.