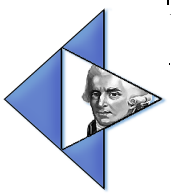
Université de Corse - Pasquale PAOLI	
***Diplôme :* Licence SPI 3ème année**	2022-2023



UE : Ateliers de programmation

Programmation Orientée Objet

Atelier 7 : Personnes, Héritage

- Enseignants : Paul-Antoine BISGAMBIGLIA, Marie-Laure NIVET, Evelyne VITTORI

Nous allons dans cet exercice modéliser le personnel d'une entreprise. Pour cela on met à votre disposition deux codes de classes de base Personne.java et Adresse.java, téléchargeables sur le drive. Créez un nouveau projet Java et dans celui-ci importez les codes copiés.

Question 1 : Amélioration de la classe Personne

Il vous est en premier lieu demandé de faire quelques améliorations et ajouts à la classe Personne pour répondre aux souhaits suivants : On souhaite pouvoir,

1. connaître à tout moment le nombre d'instance de personne ayant été créées.
2. comparer deux personnes suivant leur âge (méthode plusAgee) et retourner vrai si la première personne est plus âgée que la deuxième personne passée en paramètre. Cette méthode doit donc prendre deux paramètres de type Personne
3. comparer l'âge de la personne courante à l'âge d'une autre personne passée en paramètre (méthode plusAgeeQue Qui retourne vrai si la personne courante est plus âgée que la personne passée en paramètre).
4. tester l'égalité entre deux instances de personnes. Cette méthode devra se concentrer sur l'égalité du nom, prénom et date de naissance des personnes testées. Les éventuelles différences au niveau des adresses ne seront pas considérées. (redéfinition de la méthode equals)

Testez ces modifications dans une classe de test depuis un autre package, par exemple créez une classe TestQuestion1 dans le package tp1.test.

Question 2 : Hiérarchie d'héritage

On vous demande à présent de modéliser les faits suivants :

1. Un `Employe` est une Personne (classe fille de la classe `Personne`) possédant un salaire et dont on veut conserver la date d'embauche. De plus, selon le code du travail, une personne ne peut être employée que si elle a plus de 16 ans et moins de 65 ans. Ce dépassement d'âge éventuel doit être empêché. Pour cela, vous pouvez compter sur la rigueur du programmeur, et partir du principe qu'il n'essaiera pas de créer des employés trop jeunes ou trop vieux… Autant dire que c'est un doux rêve !

   La solution que vous allez implémenter consiste à définir les constructeurs en private (ou en protected dans l'optique de l'héritage demandé dans les questions d'après) et à créer une méthode de classe `public createEmploye` qui effectue toutes les vérifications et ne fait appel au constructeur que si tous les paramètres sont valides, dans le cas contraire cette méthode renvoie simplement `null`.

   Remarque : Une solution encore meilleure consiste à utiliser les exceptions, nous l'étudierons dans le prochain TP !

   Par ailleurs pour tous les employés :

   - on souhaite pouvoir `augmenterLeSalaire`, des employés suivant un pourcentage donné (On vérifiera que le pourcentage est positif !)

    - on doit pouvoir faire un `calculAnnuite` pour chacun des employés, c'est-à-dire retourner le nombre d'années que l'employé a passé dans l'entreprise (Toute année commencée sera comptabilisée).
2. Un manager est `Manager` est un Employé dont le salaire est augmenté différemment et qui a à sa disposition une secrétaire.
    - Le calcul du salaire du manager se fait de la façon suivante : on ajoute au pourcentage d'augmentation un bonus de 0,5% pour chaque année d'ancienneté.
    - On doit pouvoir modifier la secrétaire associée à un manager. Attention lorsqu'on assigne à un manager une secrétaire il est bon que la secrétaire en question soit au courant !
3. Une `Secretaire` est un Employé dont le salaire est augmenté différemment, qui a un ou plusieurs chefs (`Manager`) mais jamais plus de 5.
    - Le calcul du salaire de la secrétaire est le suivant : il faut ajouter au pourcentage d'augmentation passé en paramètres un bonus de 0,1% pour chaque manager différent géré par la secrétaire.
    - On doit pouvoir facilement gérer les managers dont dépendent la secrétaire (en supprimer ou en ajouter).

Vous testerez l'ensemble de ces classes en déclarant dans un main de test (depuis un autre package) un tableau d'Employé, pour lesquels vous augmenterez le salaire de 10%. Vous vérifierez ensuite l'exactitude des salaires.
Faîtes également tous les rajouts que vous pensez nécessaires pour améliorer l'utilisation de vos classes.