Lic.Info 2ème année Université d'Orléans

SOL3IF02 – POO

TP n°3 – Héritage

Exercice 1. Reprendre l'exercice 5 de la feuille de TP 2 et modifier la classe `Maillon` de manière à permettre de gérer des piles de tout type d'objets.

Quelle méthode doit-on utiliser dans la classe `Pile` pour retourner le contenu de la pile sous la forme d'une chaîne de caractères ? Faites les modifications nécessaires.

Exercice 2. Une entreprise souhaite gérer ses employés.

Dans un nouveau projet `POOTP3`, créer un package nommé `exo2`. Ajouter une classe `Main` contenant la méthode `main` dans laquelle vous pourrez créer des objets de type `Employe`, `Responsable` et `Commercial` au fur et à mesure de la création des classes correspondantes comme indiqué ci-dessous. Toutes les classes seront créées dans le package `exo2`.

Les employés

Un employé est caractérisé par un nom de type `String`, un matricule (identifiant) de type `int`, un salaire de base mensuel de type `double` et une date d'embauche de type `LocalDate` (voir note en annexe). Le salaire brut mensuel sera calculé en ajoutant au salaire de base une prime d'ancienneté dès 3 années de présence dans l'entreprise.

La convention collective de l'entreprise prévoit le versement d'une **prime d'ancienneté progressive** :

* 4% du salaire de base mensuel à partir de 3 ans d'ancienneté
* 7% du salaire de base mensuel à partir de 6 ans d'ancienneté
* 10% du salaire de base mensuel à partir de 9 ans d'ancienneté

Créer la classe `Employe`. Cette classe fournit un constructeur ayant trois paramètres, le nom, le salaire de base mensuel et la date d'embauche. Le matricule sera généré automatiquement en partant de 1.
Redéfinir la méthode `toString()` retournant l'état d'un employé sous la forme d'une chaîne de caractères.
Ajouter une méthode `public double calculerSalaireBrutMensuel()`.

Les responsables

Certains employés ont des responsabilités hiérarchiques. Ils ont sous leurs ordres d'autres employés.

Créer une classe `Responsable` sachant qu'un responsable **est un employé** caractérisé par un titre de type `String`, un attribut `pourcentagePrime` de type `int` permettant de calculer sa prime de responsabilité et un attribut `lesSubordonnes` de type un tableau de type `Employe`.

Cette classe fournira un constructeur ayant les paramètres suivants : `(String nom, double salaireBase, LocalDate dateEmbauche, String titre, int pourcentagePrime, Employe... lesSubordonnes)`.

Le salaire brut mensuel d'un responsable se calcule à partir du salaire de base auquel on ajoute la prime d'ancienneté et la prime de responsabilité. Cette prime s'obtient par la formule suivante : `salaire de base * pourcentagePrime /100`.
Redéfinir la méthode `toString()` retournant l'état d'un responsable sous la forme d'une chaîne de caractères indiquant qu'il s'agit d'un responsable, fournissant son état en tant qu'employé ainsi que son titre et les noms des employés qui sont sous sa responsabilité directe.

Les commerciaux

Les commerciaux sont des employés dont le salaire brut mensuel se calcule en ajoutant au salaire de base la prime d'ancienneté ainsi qu'un intéressement proportionnel à leurs ventes fixé à 5 %.
Créer une classe `Commercial` sachant qu'un commercial **est un employé** caractérisé par un attribut `ventesDernierMois` de type `double`.
Cette classe fournit un constructeur ayant trois paramètres, le nom, le salaire de base mensuel et la date d'embauche.
Ajouter la méthode `public void setVentesDernierMois (double ventes)` qui met à jour le champ `ventesDernierMois`.
Redéfinir la méthode `toString()` retournant l'état d'un commercial sous la forme d'une chaîne de caractères indiquant qu'il s'agit d'un commercial, fournissant son état en tant qu'employé ainsi que ses ventes du dernier mois.

Le personnel

Créer une classe `Personnel` caractérisée par un tableau d'employés et un nombre d'employés. La capacité du tableau sera fixée à 1000 au départ.

Cette classe permettra la recherche d'un employé connaissant son matricule, l'ajout d'un employé (sans doublons), le calcul du montant des salaires mensuels à verser. On souhaite pouvoir consulter, étant donné un employé, son nom suivi des noms des employés qui sont sous sa responsabilité directe, s'ils existent.

La classe fournira donc les méthodes suivantes :

- `public Employe rechercherEmploye (int matricule)` qui retourne l'employé correspondant au matricule ou `null` si le matricule ne correspond à aucun employé.
- `public boolean ajouterEmploye (Employe e)` ajoute l'employé passé en paramètre s'il n'est pas déjà présent et retourne vrai sinon elle retourne faux. Si on atteint la capacité du tableau, on la double avant d'ajouter.
- `public double montantSalairesBrutsMensuels()` qui retourne le total des salaires bruts pour le mois en cours.

Redéfinir la méthode `toString()` retournant l'état de chacun des employés de l'entreprise sous la forme d'une chaîne de caractères. On passera à la ligne après chaque employé.

Gestion du personnel

Dans le `main`, créer un objet de type `Personnel`, ajouter les employés créés précédemment puis appeler les différentes méthodes. On pourra afficher le personnel de l'entreprise ainsi que le montant des salaires bruts mensuels versés.

Annexe

Note : la classe **`java.time.LocalDate`** permet de gérer des objets non modifiables qui représentent une date au format année-mois-jour.

- La date du jour s'obtiendra en appelant la méthode : `public static LocalDate now()`.
- La méthode `public String toString()` retourne une chaîne de caractères représentant la date du type 2018-10-25.
- La méthode `public static LocalDate of (int year, int month, int dayOfMonth)` permet d'obtenir un objet de type `LocalDate` ayant les caractéristiques fournies en paramètre.

Pour calculer le nombre d'années entre deux dates, vous pouvez utiliser `java.time.temporal.ChronoUnit.YEARS.between (LocalDate d1, LocalDate d2).` qui retourne une entier de type `long`.