MOOC Intro POO Java
Exercices semaine 3

Exercice 8 : «EPFLien»

La direction de l'EPFL a besoin que vous l'aidiez à gérer son personnel. Il s'agit donc pour vous d'écrire un programme `Direction.java` répondant aux spécifications décrites plus bas. Votre programme doit être orienté objets et sans duplication de code inutile.

On aimerait stocker, dans un seul tableau, des informations sur 4 catégories d'«EPFLiens» (personnes fréquentant notre noble institution :-)):

1. Les secrétaires. Chaque secrétaire est décrit(e) par son nom, l'année où il/elle est arrivée à l'EPFL, ainsi que par le nom de son laboratoire ou institut.

2. Les enseignants. Chaque enseignant est décrit par son nom, l'année où il est arrivé à l'EPFL, le nom de son laboratoire ou institut, le nom de la section dans laquelle il enseigne (nous supposons que chaque enseignant s'occupe d'une seule section). Chaque enseignant à un salaire.

3. Les étudiants d'échange. Chaque étudiant d'échange est décrit par son nom, l'année où il est arrivé à l'EPFL, le nom de la section dans laquelle il est inscrit, ainsi que le nom de son université d'origine.

4. Les étudiants réguliers, sont ceux qui suivent la totalité de leurs études à l'EPFL. Chaque étudiant régulier est décrit par son nom, l'année où il est arrivé à l'EPFL, le nom de la section dans laquelle il est inscrit et sa note moyenne.

Remplissez votre tableau par autant d'objets (personnes) que vous voulez. Voici quelques exemples:

1. L'étudiant régulier "Gaston Peutimide", inscrit en section de systèmes de communication en 2013. Sa note moyenne est `6.0`.
2. L'étudiant régulier "Yvan Rattrapeur", inscrit en section de systèmes de communication en 2011. Il a obtenu une note moyenne de `2.5`.

3. L'étudiant d'échange "Björn Borgue", inscrit en section d'informatique en 2012. Son université d'origine s'appelle "KTH".

4. L'enseignant "Mathieu Matheu", engagé au Laboratoire des Mathématiques Extrêmement Pures (LMEP) en 1998. Il a un salaire de 10000 francs par mois et il enseigne à la section de physique.

5. La secrétaire "Sophie Scribona", engagée au Laboratoire des Machines à Taper (LMT) en 2005. Elle a un salaire de 5000 francs par mois.

Prévoyez les opérations suivantes dans votre programme:

1. Affichage du nombre d'EPFLiens, dont le nombre d'étudiants.

2. Affichage du nombre d'années moyen pendant lesquelles les personnes enregistrées ont fréquenté l'EPFL.

3. Affichage des informations enregistrées sur chaque personne.

- l'année courante peut être donnée par la tournure suivante `Calendar.getInstance().get(Calendar.YEAR)`. Il faut importer `import java.util.Calendar;`
- vous définirez une méthode `estEtudiant` retournant `true` pour les «EPFLien» étudiants et `false` pour tous les autres;

Exemple d'exécution:

```
Parmi les 5 EPFLiens, 3 sont des etudiants.
Ils sont à l'EPFL depuis en moyenne 8.8 ans
Liste des EPFLiens:
Etudiant regulier:
   Nom : Gaston Peutimide
   Annee : 2013
   Section : SSC
   Moyenne : 6.0
Etudiant regulier:
   Nom : Yvan Rattrapeur
   Annee : 2011
   Section : SSC
   Moyenne : 2.5
Etudiant d'echange:
   Nom : Bjorn Borgue
   Annee : 2012
   Section : Informatique
   Uni d'origine : KTH
Enseignant:
   Nom : Mathieu Matheu
   Annee : 1998
   Laboratoire : LMEP
   Salaire : 10000
   Section d'enseignement : Physique
Secretaire:
   Nom : Sophie Scribona
   Annee : 2005
   Laboratoire : LMT
   Salaire : 5000
```

Exercice 9 : Boîtes aux lettres

Il s'agit dans cet exercice de proposer une conception modélisant une boîtes aux lettres en Java orienté-objet.

Une boîtes aux lettres recueille des lettres, des colis et des publicités.

Une lettre est caractérisée par :

- son poids (en grammes)
- le mode d'expédition (express ou normal)
- son adresse de destination
- son format ("A3" ou "A4")

Un colis est caractérisé par :

- son poids (en grammes)
- le mode d'expédition (express ou normal)
- son adresse de destination
- son volume (en litres)

Une publicité est caractérisée par :

- son poids (en grammes)
- le mode d'expédition (express ou normal)
- son adresse de destination

Voici les règles utilisées pour affranchir le courrier :

1. en mode d'expédition normal :
   – le montant nécessaire pour affranchir une lettre dépend de son format et de son poids :
   *Formule : montant = tarif de base + 1.0 * poids (kilos), où le tarif de base pour une lettre "A4" est de 2.50, et 3.50 pour une lettre "A3"*
   – le montant nécessaire pour affranchir une publicité dépend de son poids :
   *Formule : montant = 5.0 * poids (kilos)*
   – le montant nécessaire pour affranchir un colis dépend de son poids et de son volume :
   *Formule : montant = 0.25 * volume (litres) + poids (kilos) * 1.0;*
2. en mode d'expédition express : les montants précédents sont doublés, quelque soit le type de courrier;
3. seul le courrier valide est affranchi;
4. un courrier n'est pas valide si l'adresse de destination est vide;
5. un colis n'est pas valide si son adresse de destination est vide ou s'il dépasse un volume de 50 litres.

Les trois méthodes principales liées à la boîte aux lettre sont les suivantes :

1. une méthode `affranchir()` permettant d'associer à chaque courrier de la boîte, le montant nécessaire pour l'affranchir. Cette méthode retournera le montant total d'affranchissement du courrier de la boîte.
2. une méthode `courriersInvalides()` calculant et retournant le nombre de courriers invalides présents

dans la boîte aux lettres.

3. une méthode `afficher()` affichant le contenu de la boîte aux lettre (on indiquera alors quels courriers sont invalides).

Sur papier, commencez par dessiner une hiérarchie de classes permettant de mettre en oeuvre la conception suggérée en tenant compte des contraintes mentionnées. Vous spécifierez dans votre diagramme les classes, les attributs et les entêtes des méthodes (**sans les corps**). Les contraintes suivantes devront être respectées :

1. Votre conception doit être faite de sorte à ce qu'aucune des méthodes requises n'ait besoin de faire de test sur la nature de l'objet auquel elle s'applique.
2. Les classes doivent fournir toutes les méthodes qui leur sont nécessaires.
3. Une classe ne comportera que les méthodes/attributs qui lui sont spécifique.
4. Les modificateurs d'accès devront être clairement spécifiés.
5. Vos classes doivent éviter de dupliquer inutilement des méthodes ou des attributs et elle seront **compatibles avec le programme principal fourni** dans le fichier fourni `Poste.java`

Implémentez ensuite le programme résultant de votre conception dans le fichier `Poste.java`

Avec le programme principal fourni, vous devriez avoir une exécution telle que (le prix indique le coût d'affranchissement):

```
Le montant total d'affranchissement est de 47.4
Lettre
        Poids : 200.0 grammes
        Express : oui
        Destination : Chemin des Acacias 28, 1009 Pully
        Prix : 7.4 CHF
        Format : A3

Lettre
(Courrier  invalide)
        Poids : 800.0 grammes
        Express : non
        Destination :
        Prix : 0.0 CHF
        Format : A4

Publicité
        Poids : 1500.0 grammes
        Express : oui
        Destination : Les Moilles  13A, 1913 Saillon
        Prix : 15.0 CHF

Publicité
(Courrier  invalide)
        Poids : 3000.0 grammes
        Express : non
```

        Destination :
        Prix : 0.0 CHF

Colis
        Poids : 5000.0 grammes
        Express : oui
        Destination : Grand rue 18, 1950 Sion
        Prix : 25.0 CHF
        Volume : 30.0 litres

Colis
(Courrier invalide)
        Poids : 3000.0 grammes
        Express : oui
        Destination : Chemin des fleurs 48, 2800 Delemont
        Prix : 0.0 CHF
        Volume : 70.0 litres

La boite contient 3 courriers invalides

Exercice 10 : Puissance 4

Dans le jeu de puissance 4, il s'agit d'aligner quatre pions de même couleur horizontalement, verticalement ou en diagonale sur une **grille carrée**.

Note : une version vidéo du développement pas à pas du jeu du Puissance4 est disponible dans le MOOC d'introduction (semaine 7). Essayer de le faire par soi-même en partant d'une donnée demeure un bon exercice. La spécification de cet exercice (méthode à produire etc.) est un peu différente de celle de la version MOOC.

1. Briques de base

Pour programmer ce jeu, on distinguera 3 types d'objets : des `Joueurs`, un `Jeu` et une `Partie`.

Un `Joueur` sera caractérisé par une couleur (`int`) (1 pour bleu ou 2 pour rouge par exemple) et par un nom (`String`).

Un `Jeu` sera une table carrée dont chaque case peut contenir un pion (c'est-à-dire une couleur), ou être vide.

Le jeu devra avoir une taille fournie lors de son initialisation (valeur par défaut : 8).

On fournira également une méthode `joueCoup` prenant un numéro de colonne et une couleur, et ajoutant le pion de cette couleur dans la colonne correspondante si c'est possible, c'est-à-dire si la colonne n'est pas pleine. Le pion doit être placé directement après le dernier pion placé sur la même colonne.

Cette méthode retournera un booléen pour indiquer si le pion a été placé ou non.

On ajoutera de plus une méthode `cherche4()` à la classe `Jeu`, qui retourne true s'il y a un gagnant.

La classe `Partie` sera constituée d'un joueurs ordinateur, d'un joueur humain (voir point 2) et d'un jeu.

2. Les joueurs

Pour pouvoir effectivement jouer, il nous faut maintenant implémenter des joueurs.

Nous introduirons alors deux types de joueurs :

- Les joueurs humains, pour lesquels il existe une méthode `joue` qui consiste à afficher le `Jeu` (faire le nécessaire), demander à l'écran le numéro de colonne à jouer jusqu'à ce que le coup soit valide, puis le jouer effectivement sur le Jeu.
- L'ordinateur, pour lequel dans cet exercice il existe une méthode `joue` très simple : jouer en la première position possible rencontrée.
  On informera l'utilisateur du coup joué par un message à l'écran.

  Vous pouvez, bien entendu, si vous le souhaitez, implémenter des stratégies un peu plus

évoluées.

Les joueurs ordinateurs et humains auront évidemment de nombreux points communs. Il est donc utile d'avoir une super-classe `Joueur` dont ces deux classes dériveront.

3. Une partie

Testez votre programme en créant une partie dans la méthode `main`, avec un Joueur humain et un ordinateur (qui commence).

Pour cela la classe `Partie` devra avoir une méthode `joue` qui fait jouer les joueurs tour à tour et vérifie à chaque fois s'il y a un gagnant ou si le jeu est plein et affiche le résultat en conséquence.

4. Exemple

Voici un exemple de déroulement:

Dans l'exemple ci-dessous `12345678` indiquent des numéros de colonnes.

```
Entrez votre nom:
Le
--
Le programme a joué en 1
```

```
B
--------
12345678
Joueur Le, entrez un numéro de colonne  (entre 1 et 8) :
1
Le programme a joué en 1
```

```
B
R
B
--------
12345678
Joueur Le, entrez un numéro de colonne  (entre 1 et 8) :
2
```

```
Le programme a joué en 1



B
B
R
BR
--------
12345678
...
```
