MOOC Intro POO Java
Exercices semaine 2

Exercice 4 : Poème

Dans un fichier `Poeme.java` définissez la classe `Fleur` de sorte que le programme principal suivant :

```
public static void main(String[] args)
  {
     Fleur f1 = new Fleur("Violette", "bleu");
     Fleur f2 = new Fleur(f1);
     System.out.print("dans un cristal ");
     f2.eclore();
     System.out.print("ne laissant plus ");
     System.out.println(f1);
     System.out.println(f2);
  }
```

affiche le texte suivant :

```
Violette fraichement cueillie
Fragile corolle taillée dans un cristal veiné de bleu
ne laissant plus qu'un simple souffle
qu'un simple souffle
```

Exercice 5 : banques

Le programme `Banque1.java` dont le code est fourni ci-dessous, contient un programme bancaire qui est modularisé sous forme de méthodes auxiliaires. Transformez-le en programme orienté objet sous le nom de `Banque2.java` en suivant les étapes suivantes :

- Étudiez le fonctionnement du programme. La banque a 2 clients. Chaque client a un compte privé et un compte d'épargne avec des soldes différents. Le taux d'intérêt d'un compte d'épargne est plus élevé que celui d'un compte privé. Les données de chaque client (nom, ville et soldes) sont affichées avant et après le bouclement des comptes.

- Réfléchissez aux objets que vous aimeriez utiliser dans votre programme et ajoutez les classes correspondantes. Il peut s'agir d'objets de toute nature (client, maison, billet, compte, relation bancaire etc.). N'oubliez pas que la modularisation n'est pas une science exacte. Chaque programmeur décide des classes qu'il trouve utiles et qui lui semblent correspondre au meilleur modèle de la réalité. C'est souvent l'étape la plus difficile d'un projet de programmation.

- Transférez le code concernant les objets dans les classes. Utilisez le mot-clé `private` pour encapsuler les variables et les méthodes d'instance qui ne seront pas utilisées à l'extérieur de la classe. Chaque méthode (auxiliaire ou d'instance) devrait être courte et sans trop d'instructions détaillées. Les identificateurs (noms des variables et des méthodes) devraient être parlants.

Exemple d'exécution du programme:

```
Donnees avant le bouclement des comptes:
   Client Pedro de Geneve
      Compte prive:     1000.0 francs
      Compte d'epargne: 2000.0 francs
   Client Alexandra de Lausanne
      Compte prive:     3000.0 francs
      Compte d'epargne: 4000.0 francs
Donnees apres le bouclement des comptes:
   Client Pedro de Geneve
      Compte prive:     1010.0 francs
      Compte d'epargne: 2040.0 francs
   Client Alexandra de Lausanne
      Compte prive:     3030.0 francs
      Compte d'epargne: 4080.0 francs
```

Code donné:

```
class Banque1 {

    public static void main(String[] args) {
        // Données pour tous les comptes privés (taux d'intérêt):
        double taux1 = 0.01;
        // Données pour tous les comptes d'épargne (taux d'intérêt):
        double taux2 = 0.02;
        // Données pour le premier client (nom et ville):
        String nom1 = "Pedro";
        String ville1 = "Geneve";
        // Données pour le compte privé du premier client (solde):
        double solde1PremierClient = 1000.0;
        // Données pour le compte d'épargne du premier client (solde):
        double solde2PremierClient = 2000.0;
        // Données pour le deuxième client (nom et ville):
        String nom2 = "Alexandra";
        String ville2 = "Lausanne";
        // Données pour le compte privé du deuxième client (solde):
        double solde1DeuxiemeClient = 3000.0;
        // Données pour le compte d'épargne du deuxième client (solde):
        double solde2DeuxiemeClient = 4000.0;

        // Afficher les données du premier client:
        afficherClient(nom1, ville1, solde1PremierClient, solde2PremierClient);
        // Afficher les données du deuxième client:
        afficherClient(nom2, ville2, solde1DeuxiemeClient, solde2DeuxiemeClient);

        // Bouclement du compte privé du premier client:
        solde1PremierClient = bouclerCompte(solde1PremierClient, taux1);
        // Bouclement du compte d'épargne du premier client:
        solde2PremierClient = bouclerCompte(solde2PremierClient, taux2);
        // Bouclement du compte privé du deuxième client:
        solde1DeuxiemeClient = bouclerCompte(solde1DeuxiemeClient, taux1);
```

```
        // Bouclement du compte d'épargne du deuxième client:
        solde2DeuxiemeClient = bouclerCompte(solde2DeuxiemeClient, taux2);

        // Afficher les données du premier client:
        afficherClient(nom1, ville1, solde1PremierClient, solde2PremierClient);
        // Afficher les données du deuxième client:
        afficherClient(nom2, ville2, solde1DeuxiemeClient, solde2DeuxiemeClient);
    }

    static void afficherClient(String nom, String ville,
                double solde1, double solde2) {
        // Cette méthode affiche les données du client
        System.out.println("Client " + nom + " de " + ville);
        System.out.println("   Compte prive:     " + solde1 + " francs");
        System.out.println("   Compte d'epargne: " + solde2 + " francs");
    }

    static double bouclerCompte(double solde, double taux) {
        // Cette méthode ajoute les intérêts au solde
        double interets = taux * solde;
        double nouveauSolde = solde + interets;
        return nouveauSolde;
    }
}
```

Banque avec des clientes (Niveau 1)

Vous avez bien noté que l'affichage du programme `Banque1` ne fait pas de différence entre les clientes et les clients. Ceci est facile à corriger dans la version orientée objets du programme, par exemple en ajoutant une variable d'instance booléenne `masculin` à la classe `Client` (si vous en avez une) et en testant sa valeur dans la méthode d'affichage. Modifiez votre programme, par exemple sous le nom de `Banque3`, pour qu'il affiche "cliente" au lieu de "client" comme suit:

```
Donnees avant le bouclement des comptes:
   Client Pedro de Geneve
      Compte prive:     1000.0 francs
      Compte d'epargne: 2000.0 francs
   Cliente Alexandra de Lausanne
      Compte prive:     3000.0 francs
      Compte d'epargne: 4000.0 francs
Donnees apres le bouclement des comptes:
   Client Pedro de Geneve
      Compte prive:     1010.0 francs
      Compte d'epargne: 2040.0 francs
   Cliente Alexandra de Lausanne
      Compte prive:     3030.0 francs
      Compte d'epargne: 4080.0 francs
```

Exercice 6 : supermarché

Un supermarché souhaite que vous l'aidiez à afficher le total des achats enregistrés par ses caisses. Il s'agit de compléter le programme `Supermarche.java`.

Voici les entités nécessaires pour modéliser le fonctionnement du supermarché :

- les *articles vendus* : caractérisés par leur *nom* (une chaîne de caractères), leur *prix unitaire* (un double) et une information indiquant si l'article est *en solde ou pas* (un booléen).
- les *achats* : un achat est caractérisé par *l'article acheté* et la *quantité achetée* de cet article.
- les *caddies* : caractérisés par l'ensemble des achats qu'ils contiennent .
- les *caisses* : chargées de scanner et enregistrer le contenu des caddies. Une caisse est caractérisée par un *numéro de caisse* (un entier) et le *montant total des achats* qu'elle a scanné (un double).

Le programme principal est fourni dans le fichier `Supermarche.java`. Il a pour but de faire afficher le montant total de chaque caisse au bout d'une journée donnée. **Commencez par l'étudier**.

Il s'agit maintenant de coder les structures de données et les méthodes manquantes. **Ces entités doivent pouvoir être testées avec le programme principal fourni** :

```
public class Supermarche {

    public static void main(String[] args) {
        // Les articles vendus dans le supermarché
        Article choufleur = new Article("Chou-fleur extra", 3.50, false);
        Article roman = new Article("Les malheurs de Sophie", 16.50, true);
        Article camembert = new Article("Cremeux 100%MG", 5.80, false);
        Article cdrom = new Article("C++ en trois jours", 48.50, false);
        Article boisson = new Article("Petit-lait", 2.50, true);
        Article petitspois = new Article("Pois surgeles", 4.35, false);
        Article poisson = new Article("Sardines", 6.50, false);
        Article biscuits = new Article("Cookies de grand-mere", 3.20, false);
        Article poires = new Article("Poires Williams", 4.80, false);
        Article cafe = new Article("100% Arabica", 6.90, true);
        Article pain = new Article("Pain d'epautre", 6.90, false);

        // Les caddies du supermarché
        Caddie caddie1 = new Caddie();
        Caddie caddie2 = new Caddie();
        Caddie caddie3 = new Caddie();

        // Les caisses du supermarché
        // le premier argument est le numero de la caisse
        // le second argument est le montant initial de la caisse.
        Caisse caisse1 = new Caisse(1, 0.0);
        Caisse caisse2 = new Caisse(2, 0.0);

        // les clients font leurs achats
        // le second argument de la méthode remplir
        // correspond à une quantité

        // remplissage du 1er caddie
        caddie1.remplir(choufleur, 2);
        caddie1.remplir(cdrom, 1);
        caddie1.remplir(biscuits, 4);
        caddie1.remplir(boisson, 6);
        caddie1.remplir(poisson, 2);

        // remplissage du 2eme caddie
        caddie2.remplir(roman, 1);
        caddie2.remplir(camembert, 1);
        caddie2.remplir(petitspois, 2);
        caddie2.remplir(poires, 2);

        // remplissage du 3eme caddie
        caddie3.remplir(cafe, 2);
        caddie3.remplir(pain, 1);
        caddie3.remplir(camembert, 2);

        // Les clients passent à la caisse
        caisse1.scanner(caddie1);
```

```
        caisse1.scanner(caddie2);
        caisse2.scanner(caddie3);

        caisse1.totalCaisse();
        caisse2.totalCaisse();
    }
}
```

Dans le fichier `Supermarche.java` , déclarez les classes nécessaires à la modélisation du supermarché, telles que suggérées ci-dessus.

Il vous est suggéré d'utiliser un `ArrayList` d'achats pour modéliser le contenu du caddie (Vidéo de la semaine 5 de notre MOOC d'introduction).

Faites bien attention à l'encapsulation (les variables d'instances doivent être privées).

Les méthodes à implémenter dans la classe **concernant les achats** sont :

- `afficher()` affichant les caractéristiques de l'article (son nom, son prix unitaire, la quantité achetée et le prix de l'achat). De plus, si l'article concerné est en solde, il faudra afficher le texte "(1/2 prix)".

  Voici le modèle d'affichage pour afficher() :

  Petit-lait : 2.5 x 6 = 7.5 Frs (1/2 prix)

  où `Petit-lait` est le nom de l'article, 2.5 son prix unitaire, 6 la quantité achetée, 7.5 le prix de l'achat et (1/2 prix) une indication que l'article est en solde (et donc à demi-prix). Cette indication ne doit évidemment apparaitre que si l'article est en solde.

- toute autre méthode vous semblant nécessaire.

Pour les caddies :

- `remplir(..)` conforme au programme principal fourni.
  Réfléchissez à comment stocker le contenu du caddie (qui sera scanné par la suite).
- toute autre méthode vous semblant nécessaire.

Pour les caisses :

- `totalCaisse()` qui affiche son numéro et la valeur de son champ montant total selon la forme de l'exemple suivant :

  ```
  La caisse 1 a encaisse 121.15 Frs aujourd'hui.
  ```

  où 1 est le numéro de la caisse et 121.15 le montant total. Vous supposerez que ce montant total est stocké comme attribut (et qu'il est mis à jour par la méthode `scanner(..)`, ci-dessous).

- `scanner(...)` : cette méthode, qui doit être conforme au programme principal fourni, permet à la caisse d'afficher le ticket de caisse correspondant au contenu du caddie. Cette méthode doit aussi **mettre à jour le montant total** de la caisse en y ajoutant le montant des achats du caddie.

  L'affichage tu ticket de caisse doit se faire selon le modèle ci-dessous et doit utiliser la méthode `afficher` précédemment codée :

  ```
  ========================================
  14/10/11
  Caisse numéro 2

  100% Arabica : 6.9 x 2 = 6.9 Frs (1/2 prix)
  Pain d'epautre : 6.9 x 1 = 6.9 Frs
  Cremeux 100%MG : 5.8 x 2 = 11.6 Frs

  Montant à payer : 25.4 Frs
  ========================================
  ```

  Pour afficher la date courante vous pouvez utiliser les instructions suivantes

  ```
  Date dateCourante = new Date();
  SimpleDateFormat formatDate = new SimpleDateFormat("dd/MM/yy");
  System.out.println(formatDate.format(dateCourante));
  ```

  Il faudra au préalable avoir fait les importations suivantes en début de fichier:

```
import java.util.Date;
import java.text.SimpleDateFormat;
```

- toute autre méthode vous semblant nécessaire.

Une fois le programme complété, l'exécution du programme principal devrait ressembler à ceci:

```
========================================
14/10/11
Caisse numero 1

Chou-fleur extra : 3.5 x 2 = 7.0 Frs
C++ en trois jours : 48.5 x 1 = 48.5 Frs
Cookies de grand-mere : 3.2 x 4 = 12.8 Frs
Petit-lait : 2.5 x 6 = 7.5 Frs (1/2 prix)
Sardines : 6.5 x 2 = 13.0 Frs

Montant à payer : 88.8 Frs
========================================
========================================
14/10/11
Caisse numero 1

Les malheurs de Sophie : 16.5 x 1 = 8.25 Frs (1/2 prix)
Cremeux 100%MG : 5.8 x 1 = 5.8 Frs
Pois surgeles : 4.35 x 2 = 8.7 Frs
Poires Williams : 4.8 x 2 = 9.6 Frs

Montant à payer : 32.35 Frs
========================================
========================================
14/10/11
Caisse numero 2

100% Arabica : 6.9 x 2 = 6.9 Frs (1/2 prix)
Pain d'epautre : 6.9 x 1 = 6.9 Frs
Cremeux 100%MG : 5.8 x 2 = 11.6 Frs

Montant à payer : 25.4 Frs
========================================
La caisse  numero  a encaisse 121.15 Frs aujourd'hui
La caisse  numero  a encaisse 25.40 Frs aujourd'hui
```

Exercice 7 : Segmentation de mots

Dans cette exercice nous allons créer une classe `TokenizableString` permettant d'extraire et afficher les mots d'une phrase.

Un objet `TokenizableString` est défini par :

- son contenu (`String`);
- la position de début d'une sous-séquence (`from`, un entier);
- la taille de cette sous-sequence (`len`, un entier).

La phrase entrée par l'utilisateur doit être passée en paramètre du constructeur.

Créez ensuite une méthode `boolean nextToken()` qui s'occupera de déterminer l'index de début et la longueur du mot suivant. On considérera que les séparateurs de mots sont les séquences d'au moins un caractère *espace* (i.e. `' '`).

La méthode positionnera les attributs `from` et `len` de sorte qu'ils déterminent l'index du premier caractère du mot suivant et sa longueur, pour autant qu'un tel mot existe. Dans ce cas la fonction devraretourner `true`.

Dans le cas contraire, les valeurs retournées dans `from` et `len` ne sont pas significatives, et le résultat retourné par la fonction doit être `false`.

Créez ensuite la méthode `void tokenize()` qui utilisera la méthode précédemment créée pour séparer et afficher l'ensemble des mots de la chaîne entrée, à raison de un mot par ligne, placés entre apostrophes.

Vous pouvez tester en écrivant une méthode `main` avec le corps suivant :

```
String phrase;
System.out.println("Entrez une chaine :");
phrase = scanner.nextLine();
TokenizableString toToken = new TokenizableString(phrase);
toToken.tokenize();
```

Utilisez l'exemple de fonctionnement ci-après pour vérifier la validité de votre programme ; faites en particulier attention à ce que les apostrophes entourent les mots **sans qu'il y ait d'espace entre les deux**.

Vérifiez également que le programme se comporte correctementmême lorsque la chaîne entrée se termine par une suite d'espaces.

```
Entrez une chaîne :  heuu bonjour, voici ma chaîne !
Les mots de " heuu bonjour, voici ma chaîne !  " sont:
'heuu'
'bonjour,'
'voici'
'ma'
'chaîne'
'!'
```
