TP Java
classes-objets-héritage-interface

On souhaite écrire des classes modélisant des articles en vente dans un magasin, ainsi qu'un panier d'articles permettant de les stocker.

Partie 1 - articles

1. Écrire une classe Produit représentant un article ayant deux champs, name pour son nom de type String et prix pour son prix en centimes de type long (on supposera que toutes les manipulations de prix doivent se faire avec des entiers long).
Cette classe devra posséder deux accesseurs permettant d'obtenir le nom de l'article et son prix, ainsi qu'un constructeur qui permet d'en créer des instances.
On veut que le prix d'un article et son nom ne changent pas pour toute la durée de vie de l'objet.

Écrire une classe Test permettant de tester la création d'un article et le fonctionnement des accesseurs de la manière suivante:

```
Produit produit = new Produit("cereales", 500);
System.out.println(produit.getPrix());        // affiche: 500
System.out.println(produit.getNom());         // affiche: cereales
```

2. On souhaite que lorsque l'on affiche un article avec System.out.println(), le nom de l'article s'affiche suivi de deux points (":"), d'un espace et de son prix en euros, où les centimes sont précédés d'un point ('.'), le tout suivi d'un espace et du symbole '€'. Par exemple:

```
Produit produit = new Produit("cereales", 500)
System.out.println(produit);                  // affiche: cereales: 5.00 €
Produit Lait = new Produit("lait",403);
System.out.println(Lait);             // affiche: lait: 4.03 €
```

Indication: utilisez la méthode String.format() qui fonctionne comme printf():
String.format("%d.%02d", 100, 1) retourne la chaîne "100.01".

Partie 2 - panier

1. Écrire une classe Panier modélisant un panier d'articles dans lequel on peut:
- ajouter un article avec ajoutProduit();
- retirer un article avec supprimerProduit(), qui devra renvoyer false si l'produit que l'on essaye de supprimer n'existe pas;
- connaître le nombre d'articles avec nombreProduit();
- calculer le prix total du panier avec prixTotal() (vous écrirez en commentaire au début de la méthode quel est l'ordre de grandeur de la complexité de cette méthode);

Indication: utilisez une collection de java.util pour stocker les articles du panier.

Modifier votre Test pour faire des tests en utilisant le code ci-dessous

```
Produit produit1 = new Produit("cereales", 500);
```

```
Produit produit2 = new Produit("caviar", 50000);
Produit produit3 = new Produit("eau", 101);
Panier pan = new Panier();
pan.ajoutProduit(produit1);
pan.ajoutProduit(produit2);
pan.ajoutProduit(produit3);
System.out.println(pan.nombreProduit());       // affiche: 3
System.out.println(pan.prixTotal());      // affiche: 50601
```

2. Vérifier que le code suivant affiche bien 0.

```
Panier pan = new Panier();
Produit produit = new Produit("cereales", 500);
pan.ajoutProduit(produit);
pan.supprimerProduit(new Produit("cereales", 500));
System.out.println(pan.nombreProduit());       // affiche: 0
```

Sinon, changer votre implémentation en conséquence.

3. Ajouter un poids (poids de type int) exprimé en grammes à la classe Produit et modifiez le constructeur pour qu'il accepte ce paramètre.
Modifier l'implantation de votre panier pour que l'on ne puisse pas ajouter d'article dans le panier si le poids de ce dernier doit dépasser 10 kg.

Attention: le test du poids devra se faire en temps constant (O(1)).
Modifier votre Test pour vérifier que cela fonctionne.
Par exemple, testez:

```
Produit produit1 = new Produit("cereales", 501, 1000);
Produit produit2 = new Produit("caviar", 50000, 500);
Produit produit3 = new Produit("eau", 500, 5000);
Panier pan = new Panier();
pan.ajoutProduit(produit1);
pan.ajoutProduit(produit2);
pan.ajoutProduit(produit3);
// pan.ajoutProduit(produit3);     // produit une erreur
pan.supprimerProduit(new Produit("eau", 500, 5000));
// pan.ajoutProduit(produit3);     // produit une erreur
pan.supprimerProduit(new Produit("eau", 500, 5000));
pan.ajoutProduit(produit3);        // ajout possible!
```

Indication: penser que supprimerProduit() a une valeur de retour.

4. On souhaite que chaque panier d'achat créé puisse disposer automatiquement à sa création d'un numéro de série unique (qui commence à 1 et qui est incrémenté de 1 à chaque nouveau panier créé), et qui soit connu comme l'identifiant (id) de ce panier.
Ajoutez une méthode getId() à la classe Panier qui retourne cet entier de type int, et tout ce dont vous avez besoin pour l'implémenter. Par exemple, vous pouvez tester avec le code suivant.

```
Panier c1 = new Panier();
```

```
System.out.println(c1.getId());        // affiche: 1
Produit produit1 = new Produit("cereales", 501, 1000);
c1.ajoutProduit(produit1);
Produit produit2 = new Produit("caviar", 50000, 500);
c1.ajoutProduit(produit2);
System.out.println(c1.getId());        // affiche: 1
Panier c2 = new Panier();
Panier c3 = new Panier();
Produit produit3 = new Produit("eau", 500, 5000);
c3.ajoutProduit(produit3);
System.out.println(c2.getId());        // affiche: 2
System.out.println(c3.getId());        // affiche: 3
```

5. Ajouter à la classe Panier une méthode toString() qui retourne une représentation du contenu du panier, commençant par l'identifiant unique du panier et le nombre d'articles contenus, puis affichant tous les articles du panier, un article par ligne. Par exemple, avec le code d'exemple de la question précédente, vous devez obtenir:

```
System.out.println(c1); // affiche: panier 1 [2 article(s)]
                        //          cereales: 5.01 €
                        //          caviar: 500.00 €

System.out.println(c2); // affiche: panier 2 [0 article(s)]

System.out.println(c3); // affiche: panier 3 [1 article(s)]
                        //          eau: 5.00 €
```

Partie 3 - produits frais

Certains articles sont particulièrement frais et peuvent nécessiter la prise en compte d'une date limite de consommation. Néanmoins, lorsqu'on les met dans un panier d'achat, ils se comportent comme des articles classiques et ont un nom, un prix et un poids.

1. Écrire une classe (ProduitFrais) qui correspond à un article frais ayant, en plus des informations stockées dans la classe Produit, un champ DateLimiteConso de type String correspondant à la date limite de consommation au format YYYY-MM-DD.
L'affichage d'un article frais par System.out.println() doit afficher les informations de l'article dans le même format que pour un Produit, mais précédées de la date limite de consommation. Testez avec:

```
Produit produit1 = new Produit("cereales", 500, 1000);
System.out.println(produit1);     // affiche: cereales: 5.00€
ProduitFrais frais = new ProduitFrais("Saumon", 1450, 800, "01-12-2022");
System.out.println(frais);     // affiche: B:01-12-2022 Saumon: 14.50€
```

2. Vérifier que le code suivant fonctionne, sinon faites les changements qui s'imposent.

```
Produit tin = new Produit("sardine", 500, 500);
ProduitFrais frais = new ProduitFrais("sardine", 500, 500, "01-12-2022");
Panier pan = new Panier();
pan.ajoutProduit(frais);
```

```
pan.supprimerProduit(tin);
System.out.println(cart); // affiche: panier 1 [1 article(s)]
                          //          B: 01-12-2022 sardine: 5.00 €
```

Partie 4 - facture

On souhaite éditer des factures, mais pas seulement d'articles ou d'articles frais, mais plus généralement pour une liste de choses qui peuvent être payées.

1. Pour commencer, on représente par le type Ticket des billets d'évènements ou de spectacles qui peuvent être vendus dans le même magasin que nos articles (Produit) ou nos articles frais (ProduitFrais).
Un billet est représenté par une référence (reference de type String) et par un prix en centimes d'euros (prix de type long).
Écrire une classe Ticket avec les champs nécessaires et un constructeur permettant de créer, par exemple:

```
Ticket ticket = new Ticket("R1 - W-F", 9000);
```

2. On souhaite Testtenant disposer d'un type Payable, qui dispose des méthodes:
- label() qui retourne une String représentant une descritpion textuelle de ce qui doit être payé;
- cout() qui retourne le coût de ce qui doit être payé en centimes (un entier long);
- taxe() qui retourne la proportion du coût qui est de la taxe, exprimée en centièmes de pourcents (en "pour-dix-mille") sous la forme d'un entier long. Par exemple, dans cette unité, 550 représente un taux de taxe de 5,5% et 1960 représente un taux de taxe de 19,6%;

Définir le type Payable et modifier la classe Ticket de sorte que le code suivant fonctionne (on considérera que toutes les instances de la classe Ticket sont par défaut taxées à 25%):

```
Payable payable = new Ticket("R1 - W-F", 9000);
System.out.println(payable.label());          // affiche: R1 - W-F
System.out.println(payable.cout());           // affiche: 9000
System.out.println(payable.taxe());  // affiche: 2500
```

3. On représente Testtenant une facture comme une liste de choses à payer. Créer une classe Facture qui dispose pour l'instant d'un seul constructeur sans argument et d'une unique méthode ajout(Payable p) qui ajoute à la liste de choses à payer l'argument p. Le code suivant doit fonctionner:

```
Facture facture = new Facture();
Payable payable = new Ticket("R2 - W-F", 9000);
Ticket ticket = new Ticket("M2 - R1",12000);
facture.ajout(payable);
facture.ajout(ticket);
```

4. Faites ce qu'il faut pour que la dernière ligne du code suivant fonctionne également (toutes les instances de la classe Produit sont taxées à 10% par défaut).

```
Facture facture = new Facture();
```

```
Payable payable = new Ticket("R2 - W-F", 9000);
Ticket ticket = new Ticket("M2 - R1",12000);
facture.ajout(payable);
facture.ajout(ticket);
Produit produit = new Produit("cereales", 500, 1000);
facture.ajout(produit);
```

5. Pour les articles frais, instances de ProduitFrais, la taxe est réduite de 0,1% par tranche d'un kilo de produit. Par exemple, les sardines en boite sont taxées à 10%, mais les sardines fraiches sont taxées à 10% s'il y en a moins d'1 kg, et à 10% - 0,1% s'il y en a entre 1 et 2 kg... Modifiez ce qu'il faut et vérifiez:

```
Produit tin = new Produit("sardine", 500, 500);
ProduitFrais frais = new ProduitFrais("sardine", 500, 500, "01-12-2022");
ProduitFrais frais2 = new ProduitFrais("sardine x3", 1500, 1500, "01-12-2022");
System.out.println(tin.taxe());     // affiche: 1000
System.out.println(frais.taxe());   // affiche: 1000
System.out.println(frais2.taxe());  // affiche:  990
```

6. On souhaite ajouter dans la classe Facture deux méthodes: montantTotal() qui retourne un long représentant le montant total des choses à payer de cette facture, et taxeTotale() qui retourne un long représentant le montant total des taxes de cette facture.
Que proposez vous pour que le code suivant ne soit pas obligé de parcourir 2 fois l'ensemble des articles de la facture.

```
Facture facture = new Facture();
facture.ajout(tin);
facture.ajout(frais);
facture.ajout(frais2);
System.out.println(facture.montantTotal());  // affiche: 2500
System.out.println(facture.taxeTotale());     // affiche:  248
```