COLLÈGE MONTMORENCY

Atelier #4 : L'héritage (partie 1)

Travail préparatoire pour le TP3.

Vous allez **reprendre le programme de l'atelier 3** et apporter les transformations nécessaires qui permettront de prendre en compte **plusieurs types de produits** comme indiqué dans le diagramme de classe ici-bas.

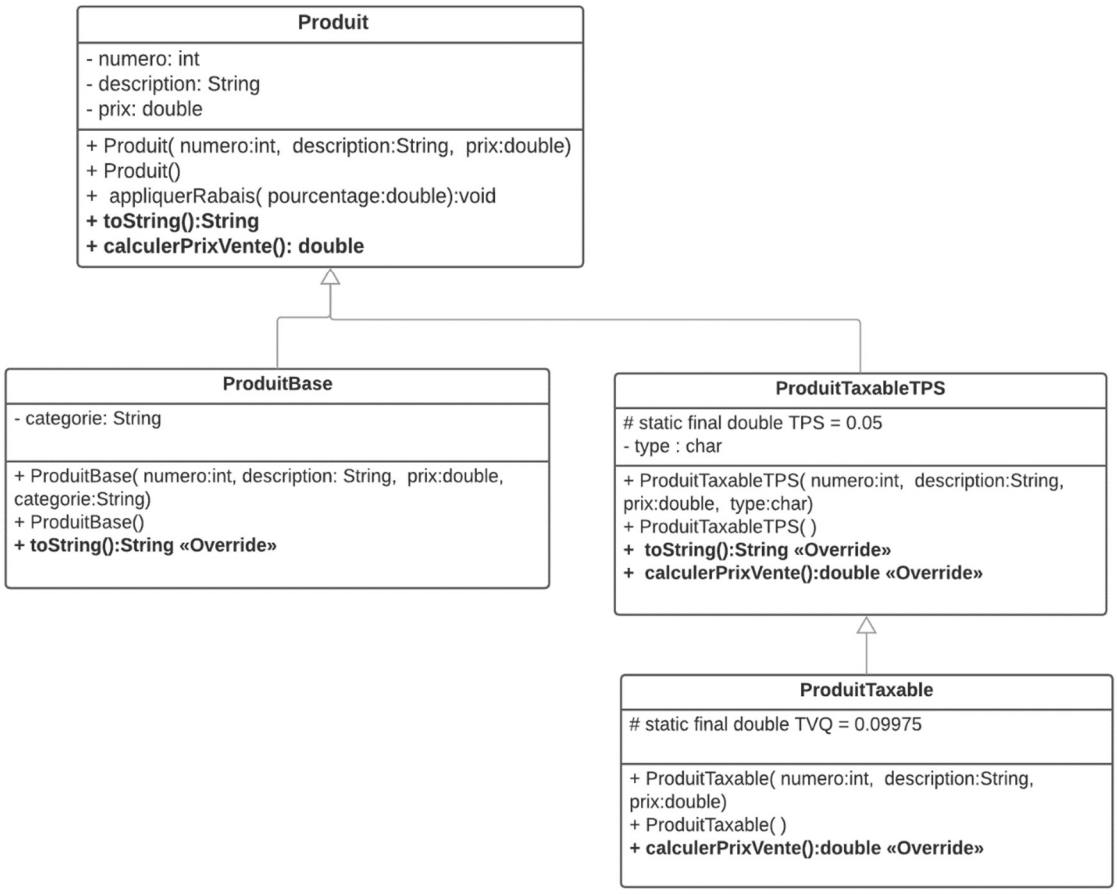

Figure 1: Diagramme de classe

La méthode `calculerPrixVente()` calcule le prix de vente de chaque produit: c'est le prix du produit plus les taxes appliquées. Puisque ce calcul diffère selon le type de produit, elle est redéfinie dans plusieurs classes. Voici comment les taxes sont appliquées sur les différents types de produits.

- Les produits de base ne sont pas taxables. Ces produits sont représentés par la classe `ProduitBase`.

Exemples de produits de base dont la vente est détaxée : fruits et Légumes, œufs, pains et céréales, poissons, produits laitiers (lait non aromatisé, fromage, beurre, crème, crème sure, yogourt), viande (bœuf, volaille, porc, agneau, viandes préparées, saucisses).

- Les produits détaxés de TVQ sont les produits sur lesquels on paie seulement la taxe TPS. Ces produits sont représentés par la classe `ProduitTaxableTPS`.
  Exemples de produits : livres imprimés et produits pour bébés (les couches pour bébés, les biberons, certains sacs jetables, …).
- Les produits taxables sont les produits sur lesquels on paie la TPS et la TVQ. Ces produits sont représentés par la classe `ProduitTaxable`. Presque tous les produits autres que ceux cités précédemment sont taxables.

Vous devez modifier ou implémenter les classes suivantes:

La classe `Produit`: Adaptez le code de la classe Produit de l'atelier 3.

1. Ajutez le tag `@Override` avant l'en-tête de la `méthode toString()` puisqu'elle est héritée de la classe `Object`.

   ```
   @Override
   public String toString() {
   ```

   Ensuite, modifiez le code de cette méthode de façon à ce qu'elle retourne une description du produit sous la forme:

   ```
   Numero: 1
   Description: Stylo
   Prix: 12,25$
   ```

2. Ajoutez une méthode `calculerPrixVente()`. La signature de la méthode est indiquée dans le diagramme de classe ci-haut. Cette méthode retourne simplement le prix du produit. Elle sera redéfinie dans les classes qui vont hériter de `Produit`.
3. Ajoutez la méthode ci-dessous:

   ```
   Public void appliquerRabais(double pourcentage) {
         setPrix(prix - (pourcentage / 100) * prix);
   }
   ```

La classe ProduitBase: Créez cette nouvelle classe qui hérite de Produit. N'oubliez pas d'ajouter la clause `extends Produit` dans la déclaration de la classe.

1. Ajoutez l'attribut d'instance `categorie`. C'est la catégorie du produit de base (comme fruits et Légumes, œufs, pains…)
2. Ajoutez un accesseur (get) pour l'attribut `categorie`.
3. Ajoutez un mutateur (set) pour l'attribut `categorie`. La catégorie doit avoir au minimum 1 caractère et au maximum 100 caractères. Aucun affichage dans cette méthode. Ajoutez des constantes statiques pour 100 et 1 .

4. Ajoutez le constructeur à 4 paramètres (voir le diagramme de classe). Ce constructeur doit appeler le constructeur à 3 paramètres de la classe `Produit`. Utilisez `super( , , )`.
5. Ajoutez un constructeur par défaut (sans paramètres) qui initialise la description et la catégorie à «**inconnu**» , le numéro à zéro et le prix à zéro.
6. Ajoutez une méthode qui demande de lire une catégorie de produit valide à partir du clavier, puis affecte cette valeur à l'attribut `categorie`.
7. Redéfinissez la méthode `toString`. Cette méthode retournera une description du produit comme ici-bas. **Cette méthode doit faire appel à la méthode `toString()` de la classe mère .** Utilisez `super.toString()`.

```
Numero: 1
Description: Boite de 1 litre de lait en carton Quebon 2%   } super.toString()
Prix: 3,25$
Catégorie: Lait                                             } Ajoutez la catégorie
```

8. Ajoutez une méthode `main()` dans laquelle vous testerez toutes ces méthodes.

La classe `ProduitTaxableTPS`: Créez cette nouvelle classe qui hérite de `Produit`.

1. Ajoutez la constante de classe (statique) TPS. Cet attribut est protégé (`protected`) pour permettre un accès direct aux classes filles et à celles se trouvant dans le même package.
2. Ajoutez l'attribut d'instance `type` (voir le modèle de classe) . C'est le type de produit.
3. Ajoutez un accesseur (get) pour l'attribut `type`.
4. Ajoutez un mutateur (set) pour l'attribut `type`. Les seules valeurs possibles sont «**l**» et «**b**» ou '\u0000' (caractère null) .
   Note l: pour livre imprimé et b pour Produits pour bébés. Ajoutez 2 constantes statiques pour ces deux valeurs:
   ```
   private static final char LIVRE = 'l';
   ```
   …
5. Ajoutez un constructeur à 4 paramètres (voir le diagramme de classe). Ce constructeur doit appeler le constructeur à 3 paramètres de la superclasse. Utilisez `super( , , )`.
6. Ajoutez un constructeur par défaut (sans paramètres) qui initialise la description à «inconnu», le numéro à zéro, le prix à zéro et le type à `'\u0000'` (caractère `null`). Appelez `this()`
7. Ajoutez une méthode qui demande de lire un type de produit valide à partir du clavier, puis affecte cette valeur à l'attribut `type`.
8. Redéfinissez la méthode `calculerPrixVente()` . Cette méthode retournera le prix du produit plus le montant de la taxe TPS. **Cette méthode doit faire appel à la méthode `calculerPrixVente()` de sa classe mère** . Utilisez `super.calculerPrixVente()`.

9. Redéfinissez la méthode `toString()`. Cette méthode retournera une description du produit sous la forme ci-dessous. **Cette méthode doit faire appel à la méthode `toString()` de sa classe mère** . Utilisez `super.toString()`.

```
Numero: 2
Description: Comment gérer son stress
Prix: 15,75$
Type de produit: 1
Prix incluant les taxes : 16,54$
```

`super.toString()` couvre les 3 premières lignes ; `Appelez calculerPrixVente()` couvre la dernière ligne.

10. Ajoutez une méthode `main()` dans laquelle vous testerez toutes ces méthodes.

La classe `ProduitTaxable`: Créez cette nouvelle classe qui hérite de `ProduitTaxableTPS`.

1. Ajoutez une constante de classe (statique) TVQ. Cet attribut est protégé (`protected`).
2. Ajoutez un constructeur à 3 paramètres. Ce constructeur doit appeler le constructeur à 4 paramètres de la classe mère. Utilisez `super( , , ,)`. L'argument correspondant au type du produit doit être `'\u0000'`.
3. Ajoutez un constructeur par défaut (sans paramètres) qui initialise la description à «inconnu», le numéro et le prix à zéro .
4. Redéfinissez la méthode `calculerPrixVente()` . Cette méthode retournera le prix plus le montant de la taxe TPS plus le montant de la taxe TVQ. **Cette méthode doit faire appel à la méthode `calculerPrixVente()` de la classe mère** . Utilisez `super.calculerPrixVente()`.
5. Modifiez la méthode `toString()` de la classe parent pour ne pas la redéfinir dans cette classe. L'affichage du `type` ne doit se faire que s'il est différent de `'\u0000'`.

```
Numero: 3
Description: Tapis de souris Confort
Prix: 10,00$
Prix incluant les taxes : 11,50$
```

6. Ajoutez une méthode `main()` dans laquelle vous testerez toutes ces méthodes.

La classe ListeProduits : aucun changement ne doit être apporté.

La classe GestionEntrepot: Le tableau `ListeProduits` pourra contenir des produits de différents types (sous-classes de `Produit`):

1. Modifier le code de la méthode `initialiserEntrepot()`. Insérez les produits de différents types ci-dessous dans la liste des produits .
2. Affichez, dans la méthode `main()`, les produits contenus dans le tableau `ListeProduits`. Mettez en commentaires les autres lignes de code.

Produits à insérer :

// produit de type ProduitBase
Numero: 1

Description: Boite de 1 litre de lait en carton Quebon 2%
Prix: 3,25$
Categorie: Lait

// produit de type ProduitTaxableTPS
Numero: 2
Description: Comment gerer son stress
Prix: 15,75$
Type de produit: l
Prix incluant les taxes : 16,54$

// produit de type ProduitTaxable
Numero: 3
Description: Tapis de souris Confort
Prix: 15,75$
Prix incluant les taxes : 18,11$

Partie2: à venir