COLLÈGE MONTMORENCY

Atelier #5.1: les interfaces

Dans cette partie de l'atelier, vous devez **reprendre le programme de l'atelier 4** (voir le package dans Moodle) et apporter les transformations nécessaires pour avoir un meilleur programme. Vous allez ajouter une interface `Taxable`, implémenter l'interface `Comparable` et redéfinir la méthode `equals()` de la classe `Object`. Ces deux dernières modifications permettront de comparer des objets de type **Produit**.

Voici le nouveau diagramme de classe. **Les éléments en rouge ne sont pas à coder**, car ils font partie de Java. **Les éléments en gras devront être ajoutés ou modifiés**.

Notation:

- Une flèche en traits pleins désigne l'héritage (`extends`) ___
- Une flèche en traits pointillés désigne l'implémentation d'une interface (`implements`) ----
- Les attributs et méthodes statiques sont soulignés. Exemple: `- TPS : double=0.05`
- + public.
- - private
- # protected
- Les classes et les méthodes abstraites sont représentées par le stéréotype «`abstract`»
- Les méthodes redéfinies sont représentées par le stéréotype «`Override`»

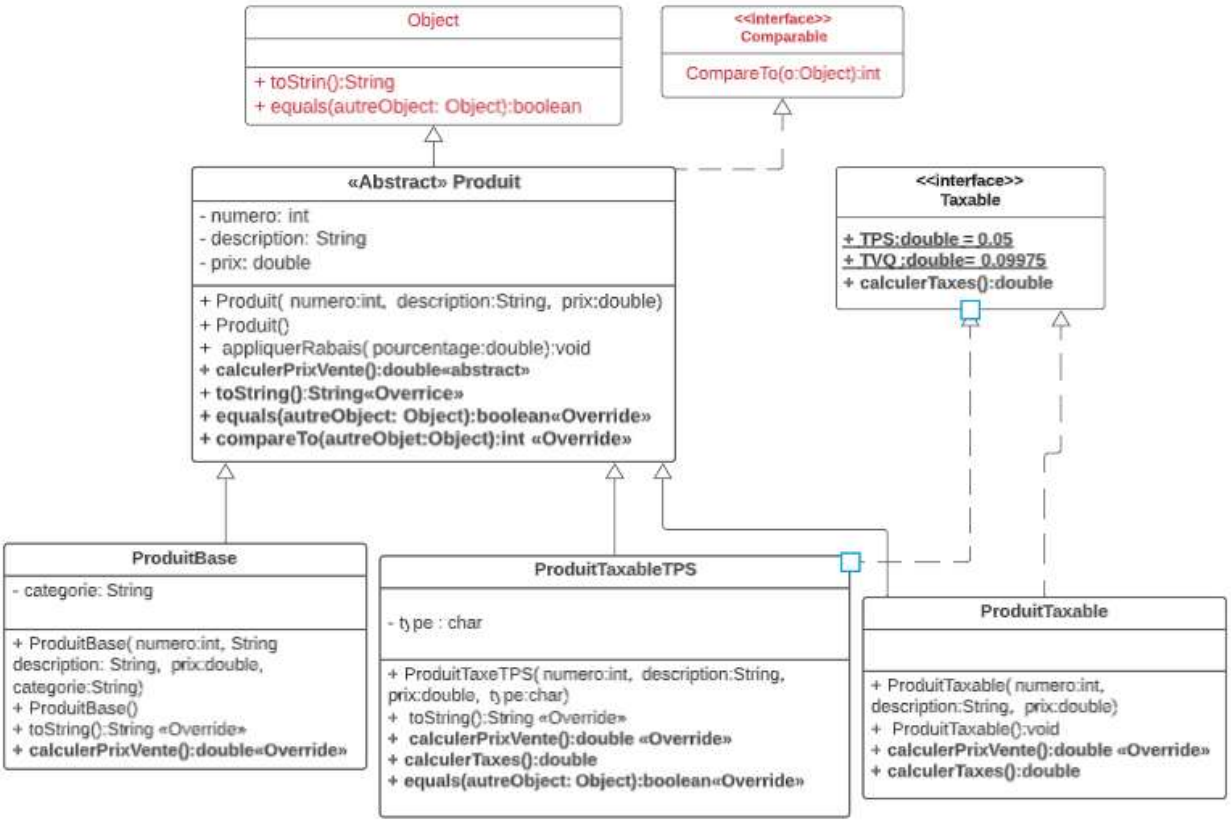

Figure 1: Diagramme de classe

Travail à faire:

Partie1: Héritage

1. La classe `Produit`
    a. Rendez cette classe abstraite(`abstract`) puisque la méthode `calculerPrixVente()` deviendra abstraite (voir le point c).
    b. Ajoutez l'annotation `@Override` à la méthode `toString()` puisque c'est une redéfinition de la méthode `toString()` de la classe `Object`. La classe `Produit` hérite implicitement de la classe `Object`.
    c. Rendez la méthode `calculerPrixVente()` abstraite. Supprimez son code et ajoutez le mot clé `abstract` dans sa déclaration.

    Justification: le code de cette méthode n'est pas encore défini dans cette classe puisqu'il diffère selon que le produit soit un `ProduitBase, ProduitTaxableTPS ou ProduitTaxable`.

2. Créer l'interface `Taxable`
    a. Déplacer les constantes `TPS` se trouvant dans `ProduitTaxeTPS` et `TVQ` se trouvant dans `ProduitTaxable` dans l'interface `Taxable`. Vous devez modifier la visibilité de ces constantes. Elles doivent être publiques.

   b. Ajoutez la méthode `public abstract double calculerTaxes()` dans l'interface `Taxable`

3. La classe `ProduitBase`
   a. Implémentez la méthode `calculerPrixVente()`. Elle retourne le prix puisqu'aucune taxe n'est appliquée pour les produits de base.
4. La classe `ProduitTaxeTPS` (corriger le nom de la classe)
   a. Implémentez l'interface `Taxable`. Implémenter la méthode `calculerTaxes`(). Utilisez la constante `TPS` de l'interface `Taxable`.
   b. Réécrivez la méthode `calculerPrixVente()` pour faire appel à la méthode `calculerTaxes()`.
5. La classe `ProduitTaxable`
   a. Modifiez la déclaration de la classe. Elle doit hériter de la classe `Produit` et non pas de `ProduiTaxeTPS`. Elle doit implémenter l'interface `Taxable`.
   b. Corrigez le constructeur.
   a. Implémenter la méthode `calculerTaxes()`. Utilisez les constantes `TPS` et `TVQ` de l'interface `Taxable`.
   b. Réécrivez la méthode `calculerPrixVente()` pour faire appel à la méthode `calculerTaxes()`.

Partie2: Comparaison des Produits

6. La classe `Produit` doit redéfinir la méthode `equals()`. Deux produits sont égaux s'ils ont le même `numero`. Voir l'exercice sur les interfaces et la classe Object sur Moodle.
7. La classe `ProduitTaxeTPS` doit redéfinir la méthode `equals()`. Deux produits sont égaux s'ils ont le même `numero` et le même `type`.
8. La classe `Produit` doit implémenter l'interface `Comparable`. Implémentez la méthode `compareTo()`. Le critère de comparaison est le prix. Par exemple, un produit de prix 10 est plus petit qu'un produit de prix 23. Voir l'exercice sur les interfaces et la classe Object sur Moodle
9. Testez le bon fonctionnement de ces méthodes. Complétez la classe `TestAtelier5` fournie dans Moodle aux endroits désignés par `----`.