Examen
Programmation orientée objet sous Java

3 heures

Rappels et conseils

- Documents non autorisés. Seuls les supports de cours et de TP sont autorisés.
- Il est strictement interdit d'utiliser vos téléphones portables. Éteignez-les et mettez-les loin (les portables ne doivent pas être visibles, ni à portée de main). Toute personne manipulant son portable se verra exclure de l'examen.
- Répondez aux questions directement sur la feuille de code fournie.

Classes et héritage (12 points)

Nous disposons d'une interface java `Article` permettant de décrire des articles à vendre dans un magasin. Chaque article possède deux propriétés : (1) son prix hors taxes et (2) et le taux de TVA qui doit lui être appliqué. Ce dernier dépend de la catégorie de l'article et prend la valeur de l'une des deux constantes déclarées dans l'interface `Article` (constantes `tvaCosmetique` ou `tvaAlimentaire`). Les deux propriétés d'un article peuvent être récupérées en appelant les deux méthodes `double getPrixHT ()` et `double getTauxTVA ()`.

Question 1 *(3 points) Écrire une classe abstraite* `Produit` *implémentant l'interface* `Article` *(voir la page 1 du code fourni). Cette classe doit définir :*

1. *4 champs privés (dont un est statique) :* `denommination` *de type* `String`, `prixHT` *(correspondant au prix du produit hors taxes) de type* `double`, `reference` *de type* `String` *et* `compteur` *(statique) de type* `int`.
2. *Un constructeur* `protected Produit(String denommination, double prixHT, String prefixe)` *permettant d'initialiser les deux champs* `denommination` *et* `prixHT` *avec les valeurs de ses deux premiers paramètres. Le champ* `reference` *sera initialisé en exécutant l'instruction suivante :*
   `this.reference = prefixe + compteur++;`
3. *Un constructeur* `protected Produit(String denommination, double prixHT)` *permettant d'initialiser les deux champs* `denommination` *et* `prixHT` *avec les valeurs de ses deux paramètres. Le champ* `reference` *sera initialisé avec la valeur du champ* `compteur` *(évidemment après l'avoir transformé en* `String`*) qui est incrémenté ensuite. Dans ce second constructeur, le corps ne doit pas contenir de caractère '='.*
4. `double getPrixHT ()` *déclarée dans l'interface implémentée* `Article`*. Cette méthode doit retourner la valeur du champs* `prixHT`*. L'autre méthode* `double getTauxTVA ()` *héritée de l'interface* `Article` *ne sera pas implémentée dans la classe* `Produit`*.*

5. *Deux méthodes publiques d'accès aux champs* `denommination` *et* `prixHT`*.*
6. *Une méthode publique* `double getTVA ()` *permettant de retourner le montant de la TVA associé au produit. Ce montant est égal au prix hors taxes (HT) du produit multiplié par le taux de TVA associé au produit (c'est à dire, égal à* `produit.prixHT` $\times$ `produit.getTauxTVA()`*).*
7. *Une méthode publique* `double getPrixTTC ()` *retournant le prix TTC (toutes taxes comprises) du produit. Le calcul de prix TTC est déterminé par la somme du prix hors taxes et le montant de la TVA correspondant (*`produit.getPrixTTC () = produit.prixHT + produit.getTVA ()`*).*
8. *La méthode toString () (méthode de la classe* `Object`*). Cette méthode doit afficher les informations concernant la* `denommination` *du produit, sa* `reference`*, son* `prixHT` *(prix hors taxes), le taux de TVA qui lui est appliqué, la partie TVA du prix total et le prix TTC. Ci-après un exemple d'affichage concernant un produit* `Oranges` *dont le prix hors taxes est égal à 2 euro et dont le taux de TVA est de 0.05.*
   `Produit Oranges [reference = A0, prixHT = 2.0, prixTTC = 2.1 dont TVA (5%) = 0.1]`

Question 2 *(3 points) Définir deux sous-classes* `ProduitCosmetique` *et* `ProduitAlimentaire` *à la classe abstraite* `Produit` *(remplir les espaces sur la page 2 du code fourni). La première classe* `ProduitCosmetique` *modélisera les produits cosmétiques dont le taux de TVA est égal à la constante* `Article.tvaCosmetique`*. Par conséquent, vous inclurez dans la classe* `ProduitCosmetique`

1. *un constructeur* `public ProduitCosmetique(String denommination, double prixHT)` *permettant d'initialiser les deux champs* `denommination` *et* `prixHT` *à partir des 2 paramètres. Le troisième champ sera initialsé avec la valeur de l'expression suivante :* `''C''+compteur`*. Le champ compteur sera incrémenté à chaque création d'un produit, qu'il soit cosmétique ou alimentaire.*
2. *une méthode* `double getTauxTVA ()` *retournant la constante* `Article.tvaCosmetique`*.*

Dans la classe `ProduitAlimentaire`*, vous incluez :*

1. *un constructeur* `public ProduitAlimentaire(String denommination, double prixHT)` *permettant d'initialiser les deux champs* `denommination` *et* `prixHT` *à partir des 2 paramètres. Le troisième champ sera initialsé avec la valeur de l'expression suivante :* `''A''+compteur`*. Le champ compteur sera incrémenté à chaque création d'un produit, qu'il soit alimentaire ou cosmétique.*
2. *une méthode* `double getTauxTVA ()` *retournant* `Article.tvaAlimentaire`*.*

Pour générer les tickets (factures) associés aux passages en caisse (un ticket par passage), une classe `Facture` a été définie (voir la page 3 du code fourni). Cette dernière dispose des champs privés suivants :

1. `produits` de type `Produit []` : tableau stockant les différents produits achetés,
2. `quantitesArticles` de type `int []` : tableau contenant les quantités des différents produits achetés. La quantité du produit `produits[i]` acheté correspond à `quantitesArticles [i]`,
3. `nbDifferentsProduits` : le nombre de différents produits achetés. Celui-ci correspond au nombre de cases occupées dans le tableau `produits`. Il ne dépend pas des quantités sotockées dans le tableau `quantitesArticles`. Par exemple, pour l'achat de `8 oranges` et `3 savons de Marseille`, `nbDifferentsProduits` sera égal à `2` (2 différents produits). `nbDifferentsProduits` doit être strictement inférieur à : `Facture.nbMaxProduit = produits.length = quantitesArticles.length`.

La classe `Facture` dispose aussi des méthodes suivantes :
– un constructeur `public Facture ()` permettant l'allocation de l'espace mémoire pour les tableaux associés aux champs `produits` et `quantitesArticles`.

– `public boolean addProduit (Produit produit, int quantite)` pour ajouter un nouveau produit `produit` au tableau `produits`. Le paramètre `quantité` sera stocké dans le tableau `quantitesArticles`. L'ajout d'un produit peut échouer si le paramètre `produit` est `null` ou si la quantité des produits est supérieure ou égale à la constante `Facture.nbMaxProduit` ou inférieure ou égale à zéro. La méthode retournera `false` si l'ajout échoue, autrement elle retourne `true`.

Question 3 *(2 points) Écrire un méthode* ***`public double totalHT ()`*** *permettant de retourner le montant hors taxes (HT) de tous les produits achetés (produits stockés dans le tableau* ***`produits`****). Cette méthode doit tenir compte des quantités des produits stockées dans le tableau* ***`quantitesArticles`****.*

Question 4 *(2 points) Écrire un méthode* ***`public double totalTTC ()`*** *permettant de retourner le montant toutes taxes comprises (TTC) de tous les produits achetés (produits stockés dans le tableau* ***`produits`****). Cette méthode doit tenir compte des quantités des produits stockées dans le tableau* ***`quantitesArticles`****.*

Question 5 *(2 points) Redéfinir la méthode* ***`public String toString()`*** *pour qu'elle affiche la liste des produits achetés, leurs quantités, le total HT et le total TTC. Ci-après un exemple d'affichage d'une facture :*

```
||Quantite: 3 || Produit Pommes de terre : [reference = A3, prixHT = .99, prixTTC = 1.0395 dont TVA (5%) = 0.0495]
|| Total produit TTC : 3.1185
||Quantite: 2 || Produit Savon de Marseille : [reference = C1, prixHT = 1.5, prixTTC = 1.8 dont TVA (20%) = 0.3]
|| Total produit TTC : 3.6
||Quantite: 1 || Produit Dattes : [reference = A2, prixHT = 3.15, prixTTC = 3.3075 dont TVA (5%) = 0.1575] ||
Total produit TTC : 2.1

Total Hors taxes (HT) = 7.97
Total toutes taxes comprises (TTC) = 8.8185
```

Interface Graphique (6 points)

Le but de cet exercice est de réaliser une interface graphique permettant de générer des tickets (factures) correspondant à un achat de plusieurs produits (une quantité est associée à chaque produit). Pour faciliter votre développement, vous utiliserez les classes définies dans la section précédente (considérez qu'elles sont bien implémentées).

Pour les besoins de notre application, nous avons défini 3 nouvelles classes :
`ListeProduits` (page 4 du code fourni), `Fenetre` (page 5 du code fourni) et `QuestionsEmd2015` (page 5 du code fourni).

La première classe, dont le code est fourni (page 4 du code fourni), stocke tous les produits disponibles dans un magasin. Son constructeur initialise l'objet en y ajoutant tous les produits disponibles à l'achat. Cette classe permet de récupérer un produit en fournissant :
1. son index : `public Produit getProduitByIndex (int i)`
2. sa denommination : `public Produit getProduitByName (String denommination)`.
Il est évidemment possible de récupérer les dénomminations de tous les produits disponibles dans le magasin en appelant la méthode `public String [] getNomsProduits ()`.

La seconde classe `Fenetre` (page 5 du code fourni) correspond à la fenêtre principale de votre application, dont l'affichage est illustré sur la figure (1). Cette classe dispose des champs suivants :
– Un panneau `panneau` destiné à accueillir les 6 combobox (produits et quantités) et les 6 labels.
– Un bouton `factureBouton` situé au sud de la fenêtre.

La dernière classe `QuestionsEmd2015` (page 5 du code fourni) est une classe de test permettant de créer et d'afficher la fenêtre principale de votre application.

Question 6 *(3 points) Complétez le code du constructeur de la classe* ***`Fenetre`*** *afin d'obtenir l'affichage illustré sur la figure (1).*

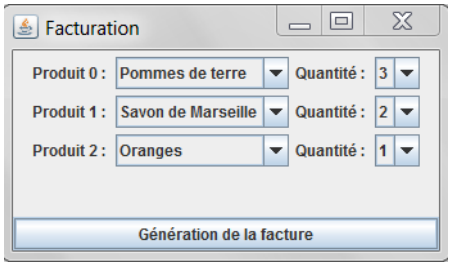

FIGURE 1 – Classes `Fenêtre` et `QuestionsEmd2015`.

Lorsque l'utilisateur (l'acheteur) a choisi les produits à acheter ainsi que leurs quantités, il valide son achat en actionnant le bouton dont l'intitulé est "Génération de la facture". Ceci provoque l'affichage d'une boite de dialogue de type message (icône d'information) dont l'aperçu est illustré sur la figure (2).

Question 7 *(1 points) Complétez le code du constructeur de la classe* `Fenetre` *afin de traiter les événements dont la source correspond au bouton intitulé "Génération de la facture".*

Question 8 *(2 points) Complétez le code de la méthode* `public void actionPerformed(ActionEvent actionEv)` *de la classe* `Fenetre` *afin d'afficher la boite de dialogue sur la figure (2). Évidemment, cette boite de dialogue affiche les propriétés des produits achetés (validés), ainsi que le total HT et TTC.*
Remarque. *Pensez à utiliser la méthode* `toString` *de la question 5.*

Polymorphisme (2 points)

Soient les deux classes suivantes :

```
class A { public void test (long l) { System.out.println(``A.test(long)''); } }
public class B extends A {
    public void test (short s) { System.out.println(``B.test(short)''); }
    public void test (long l) { System.out.println(``B.test(long)''); }
    public static void main(String[] args) {
       A aB = new B ();
       aB.test ((short) 2);
    }
}
```

Question 9 *(2 points) Qu'est ce qui sera affiché sur la console suite à l'exécution de la dernière instruction du programme ci-dessus (la dernière instruction est :* `a.test ((short) 2)`*) ?*

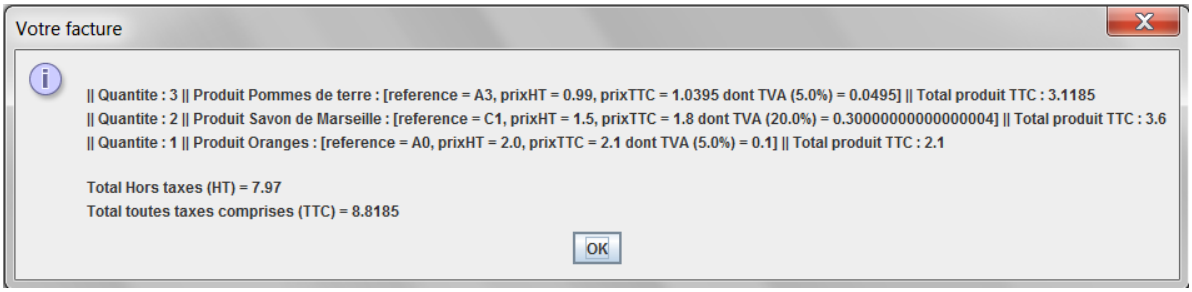

FIGURE 2 – Boite de dialogue des produits achetés.