Programmation Orientée Objet
IG2I, L2 – 2015-2016

CTP – 15 Avril 2016

Diego Cattaruzza, Philippe Kubiak, Maxime Ogier

Conditions du déroulement de l'examen

- **durée** : 2h00 ;
- ordinateurs personnels autorisés ;
- documents et pages internet autorisés ;
- accès à Moodle autorisé ;
- les moyens de communication (mails, messageries instantanées, Facebook, Tweeter, téléphones portables, etc.) sont INTERDITS.

Rendu

Vous déposez sur Moodle, dans la zone de dépôt **Dépôt du CTP 2015/2016 - Session 2** une archive compressée de votre répertoire de projet. Cette archive doit être nommée **CTP_Nom_Prenom.zip** avec votre nom et votre prénom (-1 point si ce nommage n'est pas respecté). Vérifiez bien que les fichiers sources que vous avez réalisés lors du CTP y sont présents.

Notation

Cette épreuve de CTP est évaluée sur 20 points. Pour chaque question, la moitié des points est attribuée si le code fonctionne correctement. L'autre moitié des points est attribuée en fonction de la qualité du code (respect des principes de la programmation orientée objet, efficacité des algorithmes) et de la présentation du code (indentation correcte, respects des conventions Java dans les noms de classe, attributs, méthodes et variables, commentaires au format Javadoc).

Consignes

Le sujet est composé de 7 parties liées entre elles, et qu'il faut traiter dans l'ordre. Si vous bloquez sur une question, vous pouvez tout de même essayer de continuer (mais il est probable qu'il soit difficile de tester votre code). Par ailleurs, vous êtes bien sûr autorisés à rajouter les attributs ou méthodes que vous jugez nécessaires même si ce n'est pas dit explicitement dans le sujet.

Sujet

Madame Patente est chargée de la collecte des impôts dans la communauté d'agglomération de Lens. Elle souhaite mettre en place un programme informatique (en Java bien sûr !) afin de pouvoir réaliser les calculs des taxes d'habitation et des taxes foncières à collecter sur l'agglomération. Et elle fait donc tout naturellement appel aux élèves de l'IG2I pour réaliser ce programme (2 heures peuvent suffire pour avoir une première base du modèle objet).

Madame Patente vous explique que chaque contribuable est défini par son nom et son taux d'imposition. Une habitation est définie par une référence unique, une surface (en m^2) et un secteur (qui indique la zone géographique). Chaque habitation possède un propriétaire (on ne tient pas compte de la copropriété).

Une habitation peut être un logement ou un immeuble. Un logement possède un locataire (on ne tient pas compte de la collocation), et peut être un appartement ou une maison. Un immeuble est constitué d'un ensemble d'appartements.

Une maison est caractérisée par son nombre d'étages (on inclut le rez-de-chaussée), et la superficie du terrain. Un appartement est caractérisé par le fait d'être associé ou non à une cave (on considère qu'il n'(est pas possible de posséder plusieurs caves) et un nombre de places de parking. Ces éléments rentrent en compte dans le calcul des taxes.

Le diagramme UML du modèle objet est présenté dans la Figure 1.

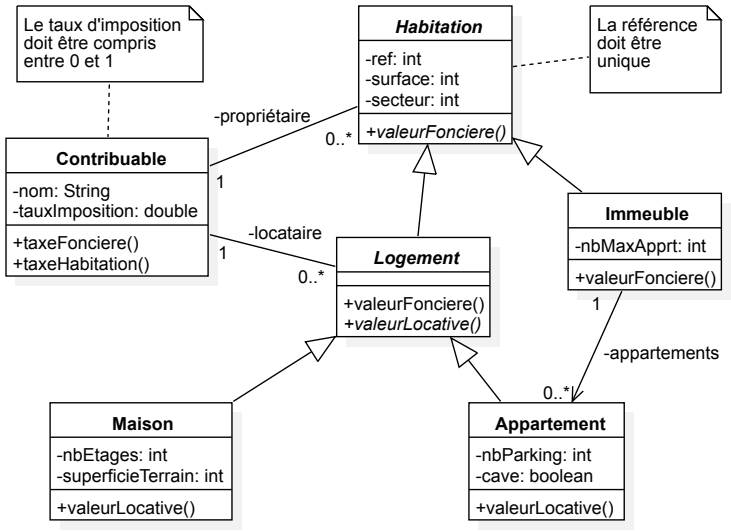

Figure 1: Diagramme UML du modèle objet.

1 La classe Contribuable (2 points)

Un contribuable est définie par son nom et son taux d'imposition. Le taux d'imposition est un nombre réel compris entre 0 et 1. Le taux d'imposition peut être amené à changer en fonction de la situation du contribuable.

Question 1. Créez un nouveau projet nommé **CTP_Nom_Prenom** avec votre nom et votre prénom. Ajoutez un paquetage **population**, dans lequel vous ajoutez une classe **Contribuable**. Ajoutez dans cette classe les attributs nécessaires, un constructeur par données, et les méthodes que vous jugez nécessaires. Redéfinissez la méthode **toString**. Ajoutez une méthode principale afin de vérifier que tout fonctionne correctement avec le code suivant :

```
Contribuable c1 = new Contribuable("Pierre", 0.8); //taux de 0.8
Contribuable c2 = new Contribuable("Paul", 0.7);
Contribuable c3 = new Contribuable("Jack", 0.6);
Contribuable c4 = new Contribuable("Jacques", 1.8);
System.out.println(c1);
System.out.println(c2);
System.out.println(c3);
System.out.println(c4);
```

2 La classe Habitation (3 points)

Selon Madame Patente, une habitation est un concept abstrait au niveau du service des impôts. Une habitation possède une référence qui doit être unique, une surface (en m^2), et un numéro de secteur. Par ailleurs, une habitation possède un propriétaire (de type **Contribuable**), et un contribuable possède donc une collection d'habitations.

Question 2. Ajoutez un paquetage **immobilier**, dans lequel vous ajoutez une classe abstraite **Habitation**. Ajoutez dans cette classe les attributs nécessaires, un constructeur par données, et les méthodes que vous jugez nécessaires. Redéfinissez la méthode **toString** (en particulier, on affichera le propriétaire de l'habitation).

Question 3. Dans la classe **Contribuable**, ajoutez un attribut pour mémoriser toutes les propriétés du contribuable (les habitations dont il est le propriétaire). N'oubliez pas, si besoin de redéfinir les méthodes **equals** et **hashCode** dans la classe **Habitation**.

Question 4. Ajoutez dans la classe **Habitation** une méthode **public boolean setProprietaire(Contribuable proprio)** qui définit le contribuable **proprio** comme propriétaire de l'habitation. Si l'habitation possède déjà un propriétaire, alors **proprio** ne peut pas devenir propriétaire, et la méthode renvoie **false**, sinon elle renvoie **true**.

Question 5. Ajoutez dans la classe **Contribuable** une méthode **public boolean acheter(Habitation h)** qui doit définir que le contribuable est propriétaire de l'habitation (il faut modifier les attributs des deux côtés de la relation). Si l'habitation possède déjà un propriétaire, l'achat ne peut pas se concrétiser, et la méthode renvoie **false**, sinon elle renvoie **true**.

3 La classe Logement (3 points)

Madame Patente vous explique qu'un logement est aussi un concept abstrait au niveau du service des impôts. Il s'agit d'une habitation qui possède un locataire (qui peut être le propriétaire).

Question 6. Dans le paquetage **immobilier**, ajoutez une classe abstraite **Logement** qui hérite de **Habitation**. Ajoutez dans cette classe les attributs nécessaires, un constructeur par données de la surface et du secteur, et les méthodes que vous jugez nécessaires. Redéfinissez la méthode **toString** : vous pouvez faire appel à la méthode **toString** de la classe mère et ajouter l'affichage du locataire.

Question 7. Dans la classe **Contribuable**, ajoutez un attribut pour mémoriser toutes les locations du contribuable (les habitations dont il est le locataire). De la même manière que dans la section précédente, ajoutez dans la classe **Logement** une méthode **public boolean setLocataire(Contribuable locataire)**, et dans la classe **Contribuable** une méthode **public boolean louer(Logement l)** qui doit définir que le contribuable est locataire de l'habitation.

4 Les classes Maison et Appartement (2 points)

Un logement peut être soit une maison, soit un appartement. S'il s'agit d'une maison, il est utile pour le calcul des taxes de définir le nombre d'étages (on inclut le rez-de-chaussée) et la superficie du terrain qui sont des nombres positifs. S'il s'agit d'un appartement, il est utile pour le calcul des taxes de définir le nombre de places de parking associées à l'appartement. Il faut aussi savoir si une cave est associée ou non (on suppose qu'un appartement ne peut pas avoir plus d'une cave) à l'appartement. Ces caractéristiques peuvent être amenées à changer (ajout d'un étage, réduction de la superficie du terrain, ajout d'une place de parking ou d'une cave).

Question 8. Dans le paquetage **immobilier**, ajoutez une classe **Maison** qui hérite de **Logement**. Ajoutez dans cette classe les attributs nécessaires, un constructeur par données, et les méthodes que vous jugez nécessaires. Redéfinissez la méthode **toString** : vous pouvez faire appel à la méthode **toString** de la classe mère et ajouter ce qui est nécessaire.

Question 9. Dans le paquetage **immobilier**, ajoutez une classe **Appartement** qui hérite de **Logement**. Ajoutez dans cette classe les attributs nécessaires, un

constructeur par données, et les méthodes que vous jugez nécessaires. Redéfinissez la méthode **toString** : vous pouvez faire appel à la méthode **toString** de la classe mère et ajouter ce qui est nécessaire.

Question 10. Dans la classe **Logement**, ajoutez une méthode principale afin de vérifier que tout fonctionne correctement avec le code suivant (et besoin mettez à jour votre code) :

```
Contribuable c1 = new Contribuable("Pierre", 0.8); //taux de 0.8
Contribuable c2 = new Contribuable("Paul", 0.7);
Contribuable c3 = new Contribuable("Jack", 0.6);
Contribuable c4 = new Contribuable("Jacques", 1.8);
// 120 m2 en secteur 1, 3 étages et 100m2 de terrain
Habitation h1 = new Maison(120, 1, 3, 100);
Habitation h2 = new Maison(150, 2, -2, 0);
// 80 m2 en secteur 1, avec 1 parking, sans cave
Habitation h3 = new Appartement(80, 1, 1, false);
Habitation h4 = new Appartement(100, 2, 0, true);
c1.acheter(h1);
c2.acheter(h2);
c1.acheter(h3);
c1.acheter(h4);
System.out.println("c4 achete h4 ? : " + c4.acheter(h4)); // false
c1.louer((Logement) h1);
c2.louer((Logement) h2);
c3.louer((Logement) h3);
c4.louer((Logement) h4);
System.out.println("c2 loue h3 ? : " + c2.louer((Logement) h3)); // false
System.out.println(h1);
System.out.println(h2);
System.out.println(h3);
System.out.println(h4);
```

5 Calcul de la taxe foncière (3 points)

D'après Madame Patente, la valeur foncière d'un logement est calculée de la manière suivante :

- en secteur 1 : 20 euros par m^2 (à multiplier par la surface) ;
- en secteur 2 : 18 euros par m^2 (à multiplier par la surface) ;
- en secteur 3 : 15 euros par m^2 (à multiplier par la surface) ;
- dans les autres secteurs : 12 euros par m^2 (à multiplier par la surface).

Nous verrons par la suite que le calcul de la valeur foncière est différente pour un immeuble.

Question 11. Ajoutez une méthode abstraite **valeurFonciere()** dans la classe **Habitation**. Proposez une implémentation de cette méthode dans le cas d'un logement selon le calcul fourni par Madame Patente.

Pour la calcul de la taxe foncière appliquée à un contribuable, Madame Patente précise qu'il suffit de faire la somme des valeurs foncières de toutes les propriétés de ce contribuable, puis de multiplier cette somme par le taux d'imposition.

Question 12. Dans la classe **Contribuable**, ajoutez une méthode **public double taxeFonciere()** qui renvoie le montant de la taxe foncière de ce contribuable.

Question 13. Dans la classe **Contribuable**, complétez la méthode principale afin de déterminer le montant de la taxe foncière dans les cas suivants :

- le contribuable a un taux d'imposition de 0.2 et n'est propriétaire d'aucun logement ;
- le contribuable a un taux d'imposition de 0.6 et est propriétaire d'une maison de 140 m^2 en secteur 1 ;
- le contribuable a un taux d'imposition de 0.8 et est propriétaire d'un appartement de 60 m^2 en secteur 2 et d'un appartement de 75 m^2 en secteur 4.

6 Calcul de la taxe d'habitation (3 points)

D'après Madame Patente, la valeur locative d'une maison est calculée de la manière suivante :

- en secteur 1 ou 2 : 25 euros par m^2 (à multiplier par la surface) plus 100 euros par étage plus 2.50 euros par m^2 de terrain ;
- dans les autres secteurs : 18 euros par m^2 (à multiplier par la surface) plus 85 euros par étage plus 3.25 euros par m^2 de terrain.

La valeur locative d'un appartement est calculée de la manière suivante : 20 euros par m^2 (à multiplier par la surface), plus 57.50 euros par place de parking, plus 38.25 s'il y a une cave.

Question 14. Ajoutez une méthode abstraite **valeurLocative()** dans la classe **Logement**. Proposez une implémentation de cette méthode dans les cas d'une maison et d'un appartement selon le calcul fourni par Madame Patente.

Pour la calcul de la taxe d'habitation appliquée à un contribuable, Madame Patente précise qu'il suffit de faire la somme des valeurs locatives de toutes

les locations de ce contribuable, puis de multiplier cette somme par le taux d'imposition.

Question 15. Dans la classe **Contribuable**, ajoutez une méthode **public double taxeHabitation()** qui renvoie le montant de la taxe d'habitation de ce contribuable.

Question 16. Dans la classe **Contribuable**, complétez la méthode principale afin de déterminer le montant de la taxe d'habitation dans les cas suivants :

- le contribuable a un taux d'imposition de 0.3 et est locataire d'un appartement de 85 m^2 sans parking et avec une cave ;
- le contribuable a un taux d'imposition de 0.6 et est locataire d'une maison de 140 m^2 en secteur 1 avec 3 étages et 500 m^2 de terrain ;
- le contribuable a un taux d'imposition de 1 et est locataire d'un appartement de 60 m^2 avec un parking et une cave et d'une maison de 175 m^2 en secteur 3 avec 2 étages et 1000 m^2 de terrain.

7 La classe Immeuble (4 points)

Un immeuble est une sorte d'habitation qui est caractérisée par un nombre maximal d'appartements. On associe aussi à un immeuble une collection d'appartements (dont la taille doit être inférieure au nombre maximal d'appartements).

La valeur foncière d'un immeuble est calculée selon la formule suivante : 125 euros par m^2 plus 25% de la somme des valeurs foncières des appartements dans cet immeuble.

Question 17. Dans le paquetage **immobilier**, ajoutez une classe **Immeuble** qui hérite de **Habitation**. Ajoutez dans cette classe les attributs nécessaires, un constructeur par données de la surface, du secteur et du nombre maximal d'appartements ; et les méthodes que vous jugez nécessaires. Redéfinissez la méthode **toString** : vous pouvez faire appel à la méthode **toString** de la classe mère et ajouter l'affichage nécessaire.

Question 18. Ajoutez dans la classe **Immeuble** une méthode **public boolean ajouterAppart(Appartement a)**. Cette méthode ajoute l'appartement **a** à la collection d'appartements de l'immeuble. Si l'ajout n'est pas possible (le nombre maximal est atteint ou l'appartement est déjà dans l'immeuble) la méthode renvoie **false**, et la méthode renvoie **true** si l'ajout a bien été effectué.

Question 19. Fournissez dans la classe **Immeuble** une implémentation de la méthode **valeurLocative()**. Enfin, ajoutez une méthode principale dans la classe **Immeuble** afin de tester votre code. Vous pouvez vous baser sur le code suivant :

```
Immeuble imm = new Immeuble(600, 1, 4);
Appartement app1 = new Appartement(120, 1, 3, true);
Appartement app2 = new Appartement(180, 1, 2, false);
Appartement app3 = new Appartement(200, 1, 2, true);
Appartement app4 = new Appartement(100, 1, 0, true);
Appartement app5 = new Appartement(120, 1, 2, false);
imm.ajouterAppart(app1);
imm.ajouterAppart(app2);
imm.ajouterAppart(app3);
imm.ajouterAppart(app4);
System.out.println("Ajout app5 ? : "+imm.ajouterAppart(app5)); // false
Contribuable proprio = new Contribuable("Jean", 0.75);
proprio.acheter(imm); // mais il n'achète pas les appartements
System.out.println("Taxe focnière : "+proprio.taxeFonciere());
```