TD5 : Héritage et Initialisation de classe

POO : Programmation Orientée Objet

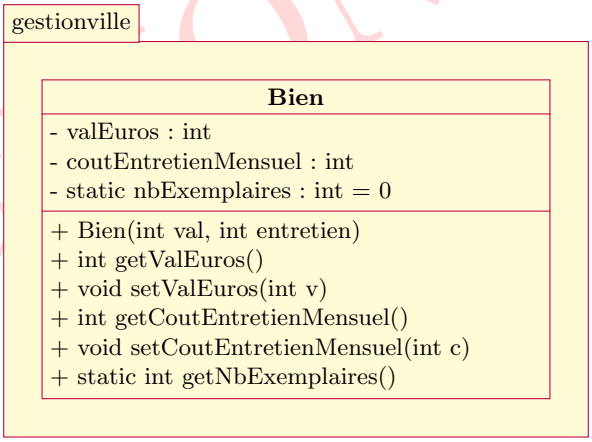
Première année

Dans un souci de transparence envers ses citoyens, une ville décide de concevoir un système informatique lui permettant de représenter l'ensemble des biens (bâtiment, véhicule, ...) qu'elle possède.

▷ **Question 1.** Un bien est défini par sa valeur et son coût d'entretien mensuel. On souhaite également connaître le nombre d'exemplaire de chaque bien. Écrire complètement la classe qui définit un bien.

gestionville.Bien

Responsabilités	Collaborateurs
– A une valeur en euros – A un coût d'entretien en euros – Conserve le nombre total de biens existants	



Réponse

L'objectif de cette question est de rappeler comment se déroule l'instanciation d'une classe et la différence entre les méthodes/attributs d'instance et les méthodes/attributs de classe.
Dans un soucis de gain de place, la correction ne contient pas le code des méthodes de modification des attributs.
Pour le moment, les attributs sont marqués comme `private`.

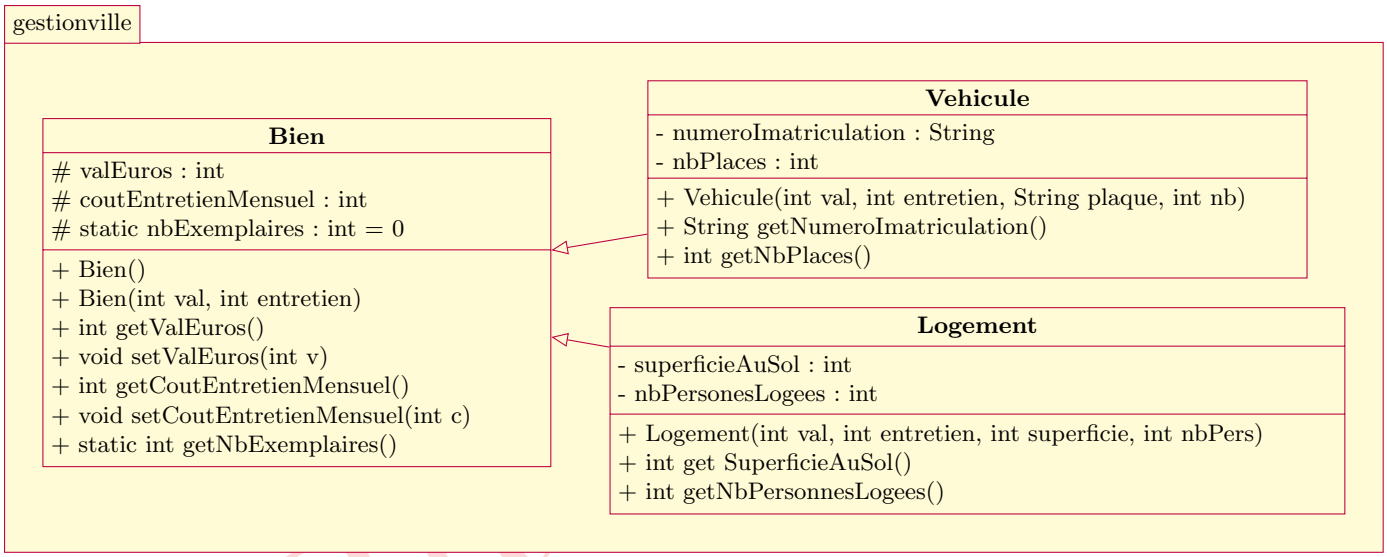
```java
package gestionville;

public class Bien {
    private int valEuros;
    private int coutEntretienMensuel;
    private static int nbExemplaires = 0;

    public Bien(int val, int entretien) {
       Bien.nbExemplaires++;
       this.valEuros = val;
       this.coutEntretienMensuel = entretien;
    }
    public int getValEuros() {
       return this.valEuros;
    }
    public int getCoutEntretienMensuel() {
       return this.coutEntretienMensuel;
    }
    public int setCoutEntretienMensuel(int cout) {
       return this.coutEntretienMensuel = cout;
    }
    public static int getNbExemplaires() {
       return Bien.nbExemplaires;
    }
}
```

Fin réponse

▷ **Question 2.** Chaque bien appartient à une catégorie qui lui permet de posséder des informations supplémentaires. Ainsi, les véhicules possèdent un numéro d'immatriculation et un nombre de places maximum, tandis que les logements sont caractérisés par leur superficie au sol et le nombre de personnes qu'ils peuvent accueillir.
Écrire les classes permettant de modéliser cette application.



Réponse

Évidemment le concept que l'on aborde dans cette question est la *spécialisation d'une sous-classe*. Dans cette question, on ne fera pas appel au super-constructeur. Autrement dit, on initialise "à la main" toutes les variables d'instances (même celles de la super-classe).
Par contre, il est important de faire remarquer aux élèves qu'il faut absolument un constructeur sans paramètre dans la classe mère `gestionville.Bien`. En effet, la classe mère possède un constructeur avec paramètre, le compilateur ne rajoutera donc pas le constructeur par défaut.
Il faut également passer les variables d'instance et de classe de la classe `gestionville.Bien` en `protected`.

```java
package gestionville;

public class Vehicule extends Bien {
    private String numeroImmatriculation;
    private int nbPlaces;

    public Vehicule(int val, int entretien, String plaque, int nb) {
       this.valEuros = val;
       this.coutEntretienMensuel = entretien;
       Bien.nbExemplaires++;
       this.numeroImmatriculation = plaque;
       this.nbPlaces = nb;
    }
    public String getNumeroImmatriculation() {
       return this.numeroImmatriculation;
    }
    public int getNbPlaces() {
       return this.nbPlaces;
    }
}
```

```java
package gestionville;

public class Logement extends Bien {
    private int superficieAuSol;
    private int nbPersonnesLogees;

    public Logement(int val, int entretien, int superficie, int nbPers) {
       this.valEuros = val;
       this.coutEntretienMensuel = entretien;
       Bien.nbExemplaires++;
       this.superficieAuSol = superficie;
       this.nbPersonnesLogees = nbPers;
    }
    public int getNbPersonnesLogees() {
       return this.nbPersonnesLogees;
    }
    public int getSuperficieAuSol() {
       return this.superficieAuSol;
    }
}
```

Fin réponse

▷ **Question 3.** Ajouter dans chaque classe une méthode `String getInfo()` permettant d'afficher les informations concernant le bien.

Réponse

Le but de cette question est d'aborder la *redéfinition de méthode*. Dans les classes `Vehicule` et `Logement`, on ne fera pas appel à la méthode de la super-classe. Par exemple, la définition de la méthode pour la classe `Bien` peut s'écrire :

```
public String getInfo() {
   return "valeur en Euros: " + this.valEuros
        + "\n cout d'entretien mensuel: " + this.coutEntretienMensuel
        + "\n nombre d'Exemplaires" + Bien.nbExemplaires;
}
```

Fin réponse

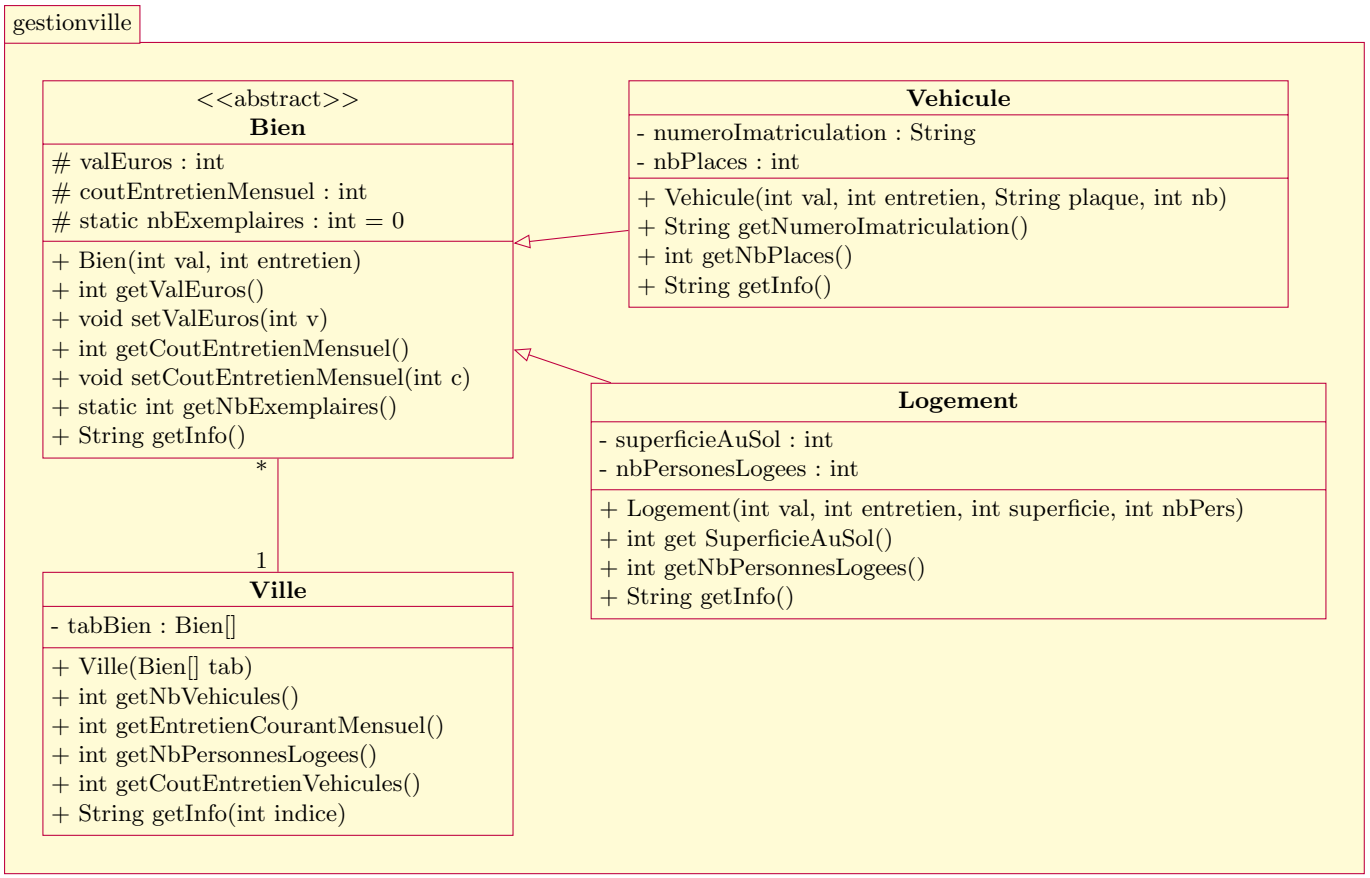
▷ **Question 4.** On souhaite maintenant rendre impossible l'instanciation d'un bien sans qu'il soit précisé si il est de type `gestionville.Logement` ou `gestionville.Vehicule` en déclarant la classe `gestionville.Bien` comme une *classe abstraite*. On souhaite également factoriser le code qui a été dupliqué dans les différents constructeurs et les différentes méthodes des sous-classes.
Effectuer les modifications nécessaires dans les différentes classes.

Réponse

On fait appel au super-constructeur avec paramètre `super(...)` ainsi qu'aux méthodes de la super-classe (`super.getInfo()`) en factorisant le code de la méthode `getInfo()`).

Fin réponse

▷ **Question 5.** Pour le mini système d'information que nous sommes en train de développer, une ville peut être considérée comme une classe d'objets qui référence l'ensemble des biens qu'elle possède. Cette classe doit offrir les services suivant :
– Consulter les informations d'un bien particulier,
– Consulter le nombre total de véhicules,
– Calculer le coût total mensuel d'entretien des biens,
– Consulter le nombre total de personnes logées,
– Calculer le coût total mensuel d'entretien pour l'ensemble des véhicules.
Écrire la classe `gestion.Ville` en s'aidant du diagramme de classes suivant :



Réponse

Les concepts mis en oeuvre dans la solution proposée sont les suivants :
- méthode `String getInfo(int i)` : liaison dynamique,
- méthode `int getNbVehicules()` : appel méthode de classe,
- méthode `int getEntretienCourantMensuel()` : parcours de tableau et liaison dynamique,
- méthode `getCoutEntretienVehicules()` : `instanceof`,
- méthode `int getNbPersonnesLogees()` : `instanceof`, trans-typage forcé (*cast*) et parcours de tableau.

```
1  package gestionville;
2
3  public class Ville {
4      private Bien[] tabBiens;
5
6      public Ville(Bien[] tab){
7          this.tabBiens = tab;
8      }
9      public String getInfo(int i){
10         return this.tabBiens[i].getInfo();
11     }
12     public int getNbVehicules(){
13         return Vehicule.getNbExemplaires();
14     }
15     public int getEntretienCourantMensuel(){
16         int result = 0;
17         for (int i = 0; i < this.tabBiens.length; i++) {
18             result +=  this.tabBiens[i].getCoutEntretienMensuel();
19         }
20         return result;
21     }
22     public int getCoutEntretienVehicules(){
23         int result = 0;
24         for (int i = 0; i < this.tabBiens.length; i++){
25             if (this.tabBiens[i] instanceof Vehicule){
26                 result += this.tabBiens[i].getCoutEntretienMensuel();
27             }
28         }
29         return result;
30     }
```

```
31      public int getNbPersonnesLogees(){
32          int result = 0;
33          for (int i = 0; i < this.tabBiens.length; i++){
34              if (this.tabBiens[i] instanceof Logement){
35                  result += ((Logement) this.tabBiens[i]).getNbPersonnesLogees();
36              }
37          }
38          return result;
39      }
40  }
```

Fin réponse

▷ **Question 6.** On souhaite maintenant distinguer les dépenses selon qu'elles sont relatives aux logements de fonction ou aux logement sociaux.
Proposer différentes solutions et comparer les.

Réponse

Les deux alternatives sont :
- deux sous-classes de `Logement`,
- un nouvel attribut dans `Logement`.
  Il faut juste discuter des évolutions futures

Désolé pour cette correction un peu rapide...

Fin réponse