Conception et programmation orientée objet
Classes, collections et relations

Modélisation, constructeurs, visibilité

Exercice 1 - Livre

On souhaite écrire un programme pour une bibliothèque. Cette dernière nous informe que, pour identifier un livre, elle utilise son auteur, son titre et son nombre de pages.

1. De quels attributs avez-vous besoin pour représenter une telle classe ? Pour chacun, précisez leur type.

Correction

```java
// [XS] I assume all members are constant (final)
// [XS] An author is defined by its name
private final String authorName;

private final String title;
private final int nbPages;
```

2. Par défaut, supposez qu'un livre est composé de 5 pages. Comment faire cela en pratique ?

Correction

```java
private final static int DEFAULT_NB_PAGES = 5;
```

Supposons qu'un tel livre soit représenté par la classe `Book`. On souhaite que, par défaut, un `Book` soit instancié avec un auteur « `UNKNOWN` », un nombre de pages égal à 5 et un titre « `EMPTY` ». En plus de cela, on veut pouvoir instancier un livre en spécifiant explicitement un auteur et un titre. En outre, le nombre de pages des livres que l'on souhaite manipuler ne doit pas être supérieur à 100 pages.

3. En POO, comment instancie-t-on une classe ?

Correction

Utiliser l'opérateur `new <Nom de classe>`

4. Proposez une implantation permettant de répondre aux besoins d'instanciation énoncés.

Correction

```
public class Book {
  public static final String DEFAULT_AUTHOR = "UNKNOWN";
  public static final String DEFAULT_NB_PAGES = "EMPTY";

  private final String author;
  private final String title;
  private final int nbPages;

  // Custom Constructor
  public Book (String p_author, String p_title, String
     p_nbPages) {
    this.author = p_author;
    this.title = p_title;

    if (p_nbPages < 0 || p_nbPages > 100)
    this.nbPages = DEFAULT_NB_PAGES;
    else
    this.nbPages = p_nbPages;
  }
```

5. Proposez une implantation de la classe `Book`. En particulier, il doit être possible de récupérer le titre, l'auteur et le nombre de pages d'un livre, mais une fois un livre créé, il ne peut pas être modifié.

Correction

```
// Add 'final' in member declaration

// {@return the title of the current instance}
public String getTitle() {return title; }

// {@return the author of the current instance}
public String getAuthor() { return author; }

// {@return the current instance's number of pages}
public int getNbPages() { return nbPages; }
```

En `Java`, il est possible de définir une méthode `main` au sein d'une classe, qui a la particularité de servir de point d'entrée pour l'exécution de la classe. Dit autrement, on peut intégrer un programme principal au sein d'une classe et exécuter cette classe comme s'il s'agissait d'un programme, dont le point de départ est la méthode `main`. Par exemple, pour définir cette méthode dans la classe `Book`, on écrit :

```java
public class Book {
  // ...
  public static void main(String[] args) {
    // Le code de notre programme
  }
}
```

En supposant que la classe `Book` soit entièrement définie, écrivez un programme principal dans `Book` dans lequel :

5. vous créez un livre par défaut ;

Correction

```java
// We need another member
public final static DEFAULT_TITLE = "My title";

// Default Constructor
public Book() {
  title = DEFAULT_TITLE;
  author = DEFAULT_AUTHOR;
  nbPages = DEFAULT_NB_PAGES;
}
```

6. vous créez un livre à partir d'un titre et d'un auteur choisis.

Correction

```java
// Custom Constructor 2
public Book(String p_title, String p_author) {
  title = p_title;
  author = p_author;
  nbPages = DEFAULT_NB_PAGES;
}
```

Correction

```java
// Main method
public static void main(String[] args) {

  // Default constructor
  Book book1 = new Book();

  // Custom constructor 2
  Book book2 = new Book("The title", "The author");
}
```

Exercice 2 - Personne

Toujours pour la bibliothèque, elle nous informe que, lorsqu'un livre est emprunté, la personne est identifiée par un nombre unique, ainsi que par son nom et son prénom.

1. De quels attributs avez-vous besoin pour représenter une classe `Person` ? Pour chacun, précisez leur type.

Correction

```
// Single id -> need a static (shared) counter later
private final int nb;

// Last name and first name
private final String lastName;
private final String firstName;
```

2. Comment peut-on garantir que l'identifiant attribué à chaque personne soit unique ?

Correction

```
// 1. Static counter
private static final int COUNTER = 0;

// 2. Custom  constructor
public Person (String p_lastName, String p_firstName) {
  lastName = p_lastName;
  firstName = p_firstName;
  nb = Person.COUNTER; // ou directement nb = COUNTER;
  Person.COUNTER++;
}
```

3. Peut-on instancier une personne à partir de son identifiant, de son nom et de son prénom ? Justifiez.

Correction

[XS] A priori non, car on risque de choisir un identifiant déjà utilisé par la bibliothèque.

La bibliothèque nous informe qu'une personne peut réserver au plus un seul livre. Elle nous indique également que la durée de réservation est gérée par un autre logiciel. Il n'est donc pas nécessaire pour nous de la prendre en compte. De plus, elle ne maintient pas d'historique des personnes ayant emprunté un livre. Ainsi, toute personne existante dans le logiciel a nécessairement emprunté un livre.

4. Comment peut-on modifier notre classe `Book` pour ajouter la possibilité qu'un livre soit emprunté ?

Correction

```
// Add a (non final) boolean
// private boolean borrowed;

// Need of both getter and setter
public boolean isBorrowed() {return borrowed};
public void setBorrowed(boolean p_isBorrowed) { borrowed =
   p_isBorrowed; }
```

5. Comment peut-on représenter le fait qu'une personne ait emprunté un livre en particulier ?

Correction

```
// Add a [Book] member in class [Person]
// private Book book;

// And tag 'book' as borrowed in the constructor
```

6. De quels constructeurs a-t-on besoin dans la classe `Person` ?

Correction

```
// A [Person] is identified with its name (at least)
// Since a [Person] should have a [Book] as soos as it
   exists, we do it in the constructor.
public Person(String p_lastName, String p_firstName, Book
   p_book) {
  // Appel du premier constructeur
  this(p_lastName, p_firstName);

  book = p_book;
  book.isBorrowed(true);
}
```

[XS] Le paragraphe ci-dessous n'appelle pas forcément une réponse de suite, on peut exposer le problème aux étudiants (ils mettront peut-être eux-mêmes le doigt dessus) et discuter des réponse possibles à l'oral.
Dans le code précédent, rien ne garantit que le Book fourni en argument du constructeur n'ait pas déjà été emprunté. On pourrait tester son état "borrowed" pour le vérifier et estimer que ce livre ne soit pas empruntable tant qu'il n'a pas été rendu. => il faudrait donc empêcher la création de la Personne si le livre fourni en argument n'est pas déjà emprunté. ==> mais cela peut être surprenant de bloquer une telle création : une Personne n'a pas besoin de livre pour exister en tant que telle. Mais si on considère que toute personne dans le logiciel a forcément un livre, alors il faut utiliser un autre livre et passer par une classe extérieure (finalement, c'est le rôle de la bibliothèque qui
1. Vérifie si le livre à affecter à la personne n'est pas déjà emprunté. => dans ce cas-là, appel du constructeur avec le Book en argument (comme avant)
2. Si ce n'est pas le cas, appeler le constructeur avec un livre tiré au hasard dans la bibliothèque et appeler le même constructeur.
On peut remarquer que la solution d'utiliser un livre par défaut pour traiter le deuxième cas ne semble pas pertinente, car chaque livre (y compris ce défaut) est unique. Ou bien on imagine un livre par défaut "vide", qui signifie finalement que la personne n'a pas de livre...

7. Est-il nécessaire de pouvoir modifier le livre emprunté par une personne ? Justifiez.

Correction

[XS] Oui, pour empêcher de le passer à une autre personne.

8. Proposez une implantation de la classe `Person`.

Correction

Voir code plus haut

Exercice 3 - Des livres et des personnes

L'objectif de cet exercice est de comprendre la notion de visibilité des variables et des méthodes, et plus spécifiquement les visibilités `public` et `private`. Pour cela, on souhaite écrire un programme principal (c'est-à-dire une méthode *main*) dans la classe `Personne`. Dans ce programme :

1. instanciez deux livres *book1* et *book2* ;

Correction

```
// Constructor (we assume 'borrowed' is false)
Book book1 = new Book("Title 1", "Author 1");
Book book2 = new Book("Title 2", "Author 2");
```

2. instanciez deux personnes *person1* et *person2*.

Correction

```
Person person1 = new Person("Last name 1", "First name 1",
    book1);
Person person2 = new Person("Last name 2", "First name 2",
    book2);
```

Désormais, on suppose que la classe `Book` dispose des éléments suivants :

```
public class Book {
  private int reference;
  public int prix;
  // ...
  public void afficherLivre() {
    // Affiche un livre
  }
  private void lireLivre() {
    // Lit un livre
  }
}
```

4. Peut-on appeler la méthode `afficherLivre` sur *book1* et *book2* ? Justifiez.

Correction

Oui car la méthode est publique.

5. Peut-on appeler la méthode `lireLivre` sur *book1* et *book2* ? Justifiez.

Correction

Oui, uniquement depuis la classe [Book]. Mais pas depuis une classe extérieure car la méthode est privée.

6. Peut-on affecter un `prix` négatif à un livre ? Si oui, que peut-on changer pour garantir qu'un prix soit modifiable ET qu'il soit toujours positif ?

Correction

Oui, car le type `int` est défini sur un intervalle incluant des nombres négatifs.
Et pour garantir qu'il soit modifiable et toujours positif, il faut : 1) qu'il soit `private` au lieu de `public`, pour éviter une modification directe sans contrôle par la classe `Book`.
2) définir un modificateur (setter) qui vérifie que l'argument associé au prix à affecter soit positif.

7. Peut-on directement modifier la `reference` d'un livre ? Justifiez.

Correction

Oui, uniquement depuis la classe `Book`. Mais pas depuis une classe extérieure, car cet attribut est `private`.

Pour la suite de cet exercice, on souhaite désormais écrire un programme principal dans la classe `Book`. Dans ce programme :

Correction

```
// Write the code inside 'main'
public static void main(String[] args) { ... }
```

8. Instanciez deux livres `book1` et `book2` ;

Correction

```
Book book1 = new Book("Title 1", "Author 1");
Book book2 = new Book("Title 2", "Author 2");
```

9. Instanciez deux personnes `person1` et `person2` ;

Correction

```
Person person1 = new Person("Last name 1", "First name 1",
    book1);
Person person2 = new Person("Last name 2", "First name 2",
    book2);
```

10. Peut-on appeler la méthode `afficherLivre` sur `book1` et `book2` ? Justifiez.

Correction

Oui (voir raison plus haut)

11. Peut-on appeler la méthode `lireLivre` sur *book1* et *book2* ? Justifiez.

Correction

Non (voir raison plus haut)

12. Peut-on modifier directement la `reference` d'un livre ? Justifiez.

Correction

Non car cet attribut est privé. Un modificateur est nécessaire.