SLAM 2 - Chapitre 2 : Les bases de la POO en JAVA (TP3)

Exercice 1.

1) Voici les erreurs présentes dans le code de base:

```java
package defaut;
public class Livre {
    // Attributs
    private String titre, auteur;
    private int nbPages;//<--il n'y avait pas de ";"

    // Constructeur
    public Livre(String unAuteur, String unTitre) {//modification de "int unTitre" par "String unTitre" et private en public
        auteur = unAuteur;
        titre = unTitre;
    }

    // Accesseur
    public String getAuteur() {//modification de Char par String
        return auteur;
    }

    // Modificateur
    public void setNbPages(int n) {//ajout de public avant void
        nbPages = n;
    }
}
```

2) La classe pour créer deux livres et renvoyer le nom des auteurs

```java
package defaut;

public class TestLivre {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        //Créer 2 livres
        Livre livre1 = new Livre("Unal Selatana", "secret d'une comptabilitée avantageuse");
        Livre livre2 = new Livre("Pondou Raaaaaaaaaaaaaaaaaaahoul", "Un Roman inspiré par CHAT GPT");

        //Afficher les auteurs
        System.out.println(livre1.getAuteur());
        System.out.println(livre2.getAuteur());
    }

}
```

3) le résultat via la console

```
Unal Selatana
Pondou Raaaaaaaaaaaaaaaaaaahoul
```

Exercice 2.

- Ajouts des accesseurs titre et nbPages:

```java
public String getTitre() {
    return titre;
}
public int getNbPages() {
    return nbPages;
}
```

- Ajouts des modificateurs auteur et titre:

```java
public void setAuteur(String p) {
    auteur = p;
}
public void setTitre(String q) {
    titre = q;
}
```

- Modificateur de setNbPages() pour correspondre à une contrainte où la valeur minimum de page est 1 :

```
public void setNbPages(int n) {
    if (n>0) {
        nbPages = n;
        System.out.println("le nombre appliqué est "+nbPages);
    }
    else {
        System.out.println("valeur inférieure ou égal a zéro");
    }
}
```

Deux résultat via la console (supérieur à 0 et inférieur ou égal à 0):

- avec 47:

```
le nombre appliqué est 47
```

- avec -4:

```
valeur inférieure ou égal a zéro
```

- Dans la méthode main():
- indication du nombre de page à la fin du constructeur pour livre1 et livre2

```
Livre livre1 = new Livre("Unal Selatana", "secret d'une comptabilitée avantageuse",69);
Livre livre2 = new Livre("Pondou Raaaaaaaaaaaaaaaaaaahoul", "Un Roman inspiré par CHAT GPT",199);
```

- commande pour afficher les pages:

```
System.out.println(livre1.getNbPages());
System.out.println(livre2.getNbPages());
```

- console:

```
69
199
```

- calcul du total de page des deux libre:

```
int totalNbPages = livre1.getNbPages() + livre2.getNbPages();
```

- résultat console:

```
268
```

Exercice 3.

1) Après avoir supprimé le constructeur, on peut rentrer les données une par une avec nos différentes fonctions mais elles ne seront pas reconnu comme dans le constructeur vu que les déclarations de variable au début sont faites dans une ligne commençant par private:

```
Livre livreVictorHugo = new Livre();

livreVictorHugo.setAuteur("Victor Hugo");
livreVictorHugo.setTitre("Les Misérables");
livreVictorHugo.setNbPages(1232);

System.out.println("Titre: " + livreVictorHugo.getTitre());
System.out.println("Auteur: " + livreVictorHugo.getAuteur());
```

- Et le résultat sous la console:

```
Titre: Les Misérables
Auteur: Victor Hugo
```

2) Dans ce cas précis, le livre affiche une erreur:

```
Livre livreVictorHugo = new Livre();
```

- La console affiche également une erreur parce que le constructeur ne peut pas être vide s' il existe. Nous pouvons donc en déduire que Eclipse génère un constructeur basique sans spécification qui permet de créer n'importe qu'elle objet.

3) Ajout de deux nouveau constructeur livre:

```
public Livre() {
    auteur = "Titre inconnu";
    titre = "Auteur inconnu";
    nbPages = 0;
}
public Livre(String unAuteur, String unTitre) {
    auteur = unAuteur;
    titre = unTitre;
    nbPages = 0;
}
public Livre(String unAuteur, String unTitre, int unNbPages) {
    auteur = unAuteur;
    titre = unTitre;
    nbPages = unNbPages;
}
```

- modification sur la méthode main pour créer 3 livre de 300 pages:

```
Livre livre1 = new Livre("Unal Selatana", "secret d'une comptabilitée avantageuse",300);
Livre livre2 = new Livre("Pondou Raaaaaaaaaaaaaaaaaaahoul", "Un Roman inspiré par CHAT GPT");
Livre livre3 = new Livre();

livre2.setNbPages(300);
livre3.setAuteur("Nicolas et Micael");
livre3.setTitre("Les Misérables");
livre3.setNbPages(300);
```

- Récupération des informations et résultat sur la console

```
System.out.println(livre1.getTitre());
System.out.println(livre1.getAuteur());
System.out.println(livre1.getNbPages());
System.out.println(livre2.getTitre());
System.out.println(livre2.getAuteur());
System.out.println(livre2.getNbPages());
System.out.println(livre3.getTitre());
System.out.println(livre3.getAuteur());
System.out.println(livre3.getNbPages());
```

```
secret d'une comptabilitée avantageuse
Unal Selatana
300
Un Roman inspiré par CHAT GPT
Pondou Raaaaaaaaaaaaaaaaaaahoul
300
Les Misérables
Nicolas et Micael
300
```

Exercice 4.

1) Création de la méthode affichetoi(), et utilisation dans la classe main

```
public String afficheToi() {
    return "Auteur : " + auteur + ", Titre : " + titre + ", Nombre de pages : " + nbPages;
}
```

```
System.out.println(livre1.afficheToi());
```

```
Auteur : Unal Selatana, Titre : secret d'une comptabilitée avantageuse, Nombre de pages : 300
```

2) Avec cette commande on obtient sur la console

```
defaut.Livre@50040f0c
```

3) Ajout de la méthode toString(): et résultat avec la console depuis la méthode main:

```
public String toString() {
    return getClass().getSimpleName() + "{titre='" + titre + "', auteur='" + auteur + "', nbPages : " + nbPages + "}";
}
```

```
Livre{titre='secret d'une comptabilitée avantageuse', auteur='Unal Selatana', nbPages : 300}
```

4) Lorsque `System.out.println(livre);` est exécuté, la méthode `println(Object x)` de `PrintStream` est appelée, laquelle utilise `String.valueOf(x)`. Cette dernière méthode appelle automatiquement `toString()` sur l'objet passé en argument, sauf s'il est `null`. Comme toutes les classes en Java héritent de `Object`, qui possède une méthode `toString()`, cette dernière est utilisée par défaut pour générer une représentation sous forme de texte de l'objet. Si `toString()` n'est pas redéfinie, elle affiche le nom de la classe suivi d'un identifiant hexadécimal, ce qui n'est pas très lisible. En redéfinissant `toString()` dans une classe, on contrôle directement la représentation textuelle de l'objet, ce qui explique pourquoi `System.out.println(livre);` affiche automatiquement un texte descriptif lorsqu'on a surchargé cette méthode.

5) modification de afficheToi pour correspondre a toString:

```
public String afficheToi() {
    return toString();
}
```

Exercice 5.

1) Ajout et modification en vigueur de la consigne

```
private double prix;
```

```
public double getPrix() {
    return prix;
}
```

```
public void setPrix(double prix) {
    this.prix = prix;
}
```

```
public String toString() {
    String prixAffichage = (prix == -1) ? "Prix pas encore donné" : prix + " €";
    return "Auteur : " + auteur + ", Titre : " + titre + ", Nombre de pages : " + nbPages + ", Prix : " + prixAffichage;
}
```

```
public Livre(String unAuteur, String unTitre, int unNbPages, double prix) {
    auteur = unAuteur;
    titre = unTitre;
    nbPages = unNbPages;
    this.prix = prix;
    prixFixe = true;
}
```

2)

```
private boolean prixFixe;
```

```
public void setPrix(double prix) {
    if (prixFixe) {
        System.out.println("Erreur : Le prix a déjà été fixé et ne peut plus être modifié.");
    } else {
        this.prix = prix;
        prixFixe = true;
    }
    }
public boolean isPrixFixe() {
    return prixFixe;
}
```