Exercice 1 - Livre:

1. Déclarer une classe Book contenant les champs privés title et author.

Voir les codes.

2. Puis essayer le code suivant dans une méthode main de la classe Book.

```
Book book = new Book();
System.out.println(book.title + ' ' + book.author);
```

Expliquer: Aucun erreur est affiché, car un attribut déclaré comme “private” est visible par les méthodes dans le scope de la classe (c’est pourquoi les méthodes getter et setter marchent), ainsi que la méthode main.

3- Créer une classe Main (dans un fichier Main.java) et déplacer le main de la classe Book dans la classe Main. Quel est le problème ? Comment peut-on le corriger ?

J’ai cette erreur : The field Book.title is not visible
Et pour corriger ce problème, il y a deux méthodes:

-Changer la visibilité de l’attribut
-Ajouter des “getter” et “setter” pour l’attribut.

4-Rappeler les 4 visibilités possibles en Java.
Pourquoi doit-on toujours déclarer les champs privés sous peine de mort certaine ?

Les quatre visibilité possibles en Java sont : public > protected > par défaut > private.
dont:

-public : accessible partout
-protected : Dans la classe et ses sous-classes et le même package
-par défaut : Accessible depuis le même package
-private : Que dans la classe.

5-Qu'est ce qu'un accesseur ?
Sachant qu'il n'y a aucune raison de changer les champs d'un livre une fois celui-ci créé (au moins pour l'instant), quels sont les accesseurs que l'on doit mettre ici ?

Un accesseur est une méthode publique que l’on crée pour donner le droit de lecture aux autres sur une variable privée.

On peut mettre accesseur getters sans mettre setters vers l’attribut author et title , pour éviter tout changement de ces deux attributs.

6- Comment indiquer à un futur développeur que la valeur du champ title (ou author) ne doit pas être modifiée ? Pourquoi est-ce important ?

On peut utiliser l’attribut “final”.

7- Ajouter un constructeur initialisant (un constructeur initialise, il ne construit pas vraiment) le livre avec un titre (ptitle) et un auteur (pauthor).
Pourquoi le code du main ne fonctionne plus ? Changer celui-ci.

J'ai ajouté le constructeur avec qui prend ptitle et pauthor comme entrée, et j'obtiens cette erreur dans le main: The constructor Book() is undefined
En effet, en Java quand il y a 0 constructeur, le compilateur va générer un constructeur par défaut automatiquement. Mais quand il y a au moins un constructeur, alors il ne va pas le générer.
Dans ce cas , on doit la déclarer explicitement.

8 - Modifier le constructeur précédent pour que les deux paramètres s'appellent title et author. Quel est le problème ? Quelle est la solution ?

Cela peut provoquer une ambiguïté entre les paramètres locaux du constructeur et les variables de la classe.
Pour éviter ce problème, on appelle les variables de la classe avec "this".

9. Écrire un autre constructeur qui prend juste un titre et pas d'auteur.
On initialisera le champ author avec "<no author>" dans ce cas.

Voir les codes.

On appelle les autres constructeurs dans un constructeurs en utilisant le mot clé "this".

Par exemple :

```
public Book(String titre) {
        this(titre, "<no author>");
}
```

10. Comment le compilateur fait-il pour savoir quel constructeur appeler ?
En fonction des paramètres fournis au moment d'instanciation de l'objet.

11. Comment faire maintenant pour que le second constructeur appelle le premier ?
On utilise le mot clé comme illustré dans la question 9.

Exercice 2- Liberté, Égalité et Fraternité

1. Book b1 = new Book("Da Java Code", "Duke Brown");

```
Book b2 = b1;
Book b3 = new Book("Da Java Code", "Duke Brown");
System.out.println(b1 == b2); // True
System.out.println(b1 == b3); // False
```

Qu'affiche le code ci-dessus ? Pourquoi ?
Car "==" compare les référence des objets.

2. Écrire dans la classe Book une méthode (à vous de trouver le nom et la signature exacte de la méthode) qui renvoie true si deux livres ont les mêmes nom et description.
Attention à la comparaison de chaînes de caractères.

C'est la méthode "equals()" qu'on doit écrire, avec la substitution '@Override'

Pour comparer deux chaînes de caractère, on utilise aussi "equals", au lieu de "==".

3. La classe java.util.ArrayList correspond à un tableau qui s'agrandit dynamiquement.
A quoi sert la méthode indexOf de ArrayList (RTFM) ?

indexOf recherche et renvoie l'indice du premier élément vérifie la condition dans la liste. En cas d'échec, il renvoie -1.

4. Exécutez le code suivant :

```
public static void main(String[] args){
    Book b1 = new Book("Da Java Code", "Duke Brown");
    Book b2 = b1;
    Book b3 = new Book("Da Java Code", "Duke Brown");
    ArrayList list = new ArrayList();
    list.add(b1);
    System.out.println(list.indexOf(b2));
    System.out.println(list.indexOf(b3));
}
```

Quel est le problème avec les résultats affichés sur la console.
Note : ici, le compilateur génère un warning au niveau du add. Nous verrons dans les prochains TD comment l'éviter.

On voit "0" et "0" , tous les deux objets se trouvent dans la liste alors que b3 n'est pas le même objet que b1.

5. Quelle méthode de Book est appelée par ArrayList.indexOf (RTFM again) ?

ArrayList utilise la méthode "equals" de la classe pour comparer deux objets.

6. Modifier la classe Book pour que indexOf() fait sur l'ArrayList teste si les deux livres ont les mêmes caractéristiques.

Fait.

7. Utiliser l'annotation @Override (java.lang.Override) sur la méthode ajoutée à Book.

Fait

8. A quoi sert l'annotation @Override ?
Pour indiquer explicitement qu'une méthode dans la classe fille est censée redéfinir une méthode de la classe mère.

9. Qu'affiche le code ci-dessous ?

```
Book aBook = new Book("Da Java Code", "Duke Brown");
Book anotherBook = new Book(null, null);
ArrayList list = new ArrayList();
list.add(aBook);
System.out.println(list.indexOf(anotherBook));
```

Où se situe le problème ?
Rappeler pourquoi un code doit arrêter de fonctionner si celui-ci est mal utilisé par un développeur.
Que doit-on faire pour corriger le problème ?

J'ai cet erreur : Cannot invoke "String.equals(Object)" because the return value of "TP4.Book.getTitle()" is null

Pour résoudre ce problème, on peut effectuer une vérification de nullité avant d'appeler la méthode `equals`. Ou ajouter une vérification de nullité dans la méthode `equals` de la classe `Book` pour éviter des problèmes similaires.

10. Rappeler quelle est la règle de bonne pratique concernant l'utilisation de null.
En Java, faut éviter l'utilisation de "null", et utiliser une valeur par défaut.
Et si on doit utiliser, alors faut faire attention à des vérifications de la nullité dans les méthodes.

11. A quoi sert la méthode java.util.Objects.requireNonNull (RTFM) ?
Comment l'utiliser ici pour empêcher de construire un livre avec des champs null ?

Pour vérifier si l'objet est null, s'il est nul alors une exception "NullPointerException" est lancée.
Exemple de code :

```
public Book(String title, String author) {
    this.title = Objects.requireNonNull(title, "Le titre ne peut pas être null");
    this.author = Objects.requireNonNull(author, "L'auteur ne peut pas être null");
}
```

Exercice 3 - Afficher un livre

1. Écrire cette méthode, pour obtenir par exemple l'affichage suivant :
Da Java Code by Dan Duke

Fait

2. Peut-on utiliser l'annotation @Override, ici ?

toString n'est pas déclaré comme abstrait dans la classe objet,
donc il n'y a pas besoin d'utiliser @Override,
Mais par convention, on peut l'utiliser même pour indiquer une substitution ici.

3. Plus difficile, on souhaite maintenant afficher uniquement le titre du livre si le livre est construit avec le constructeur avec un seul argument. Attention, l'affichage de new Book("", "<no author>") devra bien afficher "<no author>". Si vous entrevoyez une solution qui utilise null, pensez plus fort :)

Exercice 4 - Tri à caillou [à la maison]

1. Écrire une méthode swap qui échange les valeurs de deux cases d'un tableau. void swap(int[] array,int index1,int index2)

[ ] : pour déclarer un tableau, et accéder à un index dans un tableau
{} : Pour initialiser un tableau.

Arrays.*toString() :* Permet d'afficher les contenus du string.