IS2A4 – PPO – T.P. 2a

Résumé

Ce TP aborde l'héritage en java.

N'hésitez pas à consulter la documentation en ligne sur le site d'oracle : `https://docs.oracle.com/javase/8/docs/api/` (regardez la version correspondant la version de java que vous utilisez).

I Préparation du TP (5 min)

I.1. Créez un dossier spécifique pour les TP de PPO qui vous appellerez `TP_PPO` par exemple.

I.2. Dans ce dossier, créez un dossier `TP2a`. C'est dans ce dossier que vous travaillerez aujourd'hui. Durant les prochaines séances de TP, essayez de garder cette pratique.

I.3. Récupérez votre solution du `TP1a` et copié le dans le dossier `TP2a` (vous pouvez aussi récupérer le fichier `TP2a_source.zip` sur moodle et le décompresser dans `TP2a`).

II Livres avec une seul ou plusieurs auteurs (40 min)

Il existe deux grands types de livres, les ouvrages monographiques qui n'a qu'un unique auteur et les ouvrages collectifs qui ont plusieurs auteurs. L'objectif de la suite sera de créer deux classes `OuvrageMono` et `OuvrageColl` qui hériteront d'une classe abstraite `Livre`. Suivez les différentes étapes afin de modifier le code pour créer cet héritage.

II.1. On va commencer par créer la classe `OuvrageMono` en dupliquant la classe `Livre`. Instanciez un objet de la classe `OuvrageMono` dans la classe `Main` pour tester votre nouvelle classe (mettez en commentaire le reste des instances).

Fichier : `Main.java`

```
10        Auteur zola = new Auteur("Zola");
11        OuvrageMono nana = new OuvrageMono(zola, "Nana",19);
12        System.out.println(nana.show());
```

Aide : compréhension des messages d'erreurs :

- ```
  Main.java:11: error: cannot find symbol
          OuvrageMono nana = new OuvrageMono(zola, "Nana",19);
          ^
    symbol:  class OuvrageMono
    location: class Main
  ```

  Vous avez peut être oublié d'importer la classe `OuvrageMono` dans la classe `Main`.

II.2. Créez une nouvelle classe `OuvrageColl` en dupliquant la classe `OuvrageMono` et changez la variable d'instance `auteur` en `auteurs` en la faisant passer du type `Auteur` à `Auteur[]`. Modifiez en conséquence le constructeur pour prendre un tableau d'`Auteur` et changez la méthode `show()` pour voir la liste des auteurs :

Fichier : `OuvrageColl.java`

```
16    public String show() {
17        String out = this.titre + ",␣";
18        for (Auteur aut : this.getAuteurs()) {
19            out += aut.show() + "␣-␣";
20        }
21        out += ",␣" + this.prix;
22        return out;
23    }
```

Ajoutez une instance de la classe `OuvrageColl` dans la classe `Main` pour tester votre code.
```

II.3. En regardant en détail les classes `OuvrageMono` et `OuvrageColl`, on voit qu'une partie du code des deux classes est identique. On va alors uitliser le principe d'héritage pour faire une classe `Livre` parente aux classes `OuvrageMono` et `OuvrageColl`. Pour cela, créez une classe `Livre` en y transférant les choses communes aux deux classes `OuvrageMono` et `OuvrageColl` :

- les variables `titre`, `prix` et `somme`.
- les méthodes `setTitre`, `getPrix` et `getSomme`.

N'oubliez pas d'ajouter un constructeur à la classe `Livre` qui prend un titre et un prix.

II.4. Modifiez les classes `OuvrageMono` et `OuvrageColl` pour qu'elles héritent de la classe `Livre`. Essayez de compilez vos classes `Livre`, `OuvrageMono` et `OuvrageColl`.

Aide : compréhension des messages d'erreurs :

- ```
  media/OuvrageMono.java:6: error: constructor Livre in class Livre cannot be
      applied to given types;
     public OuvrageMono(Auteur auteur, String titre, int prix) {
                                                              ^
    required: String,int
    found: no arguments
    reason: actual and formal argument lists differ in length
  ```

  Vous avez peut être oublié de faire un appel en début de constructeur vers un constructeur de votre classe parent. Si vous ne faîtes aucun appel, JAVA ajoute automatiquement un appel vers le contructeur par défaut (ici `Livre()`) qui n'existe pas dans la classe `Livre` et donc le compilateur râle. Pour régler le problème, faîtes un appel vers le constructeur parent qui prend un titre et un prix (qui lui devrait exister).

- ```
  media/OuvrageMono.java:16: error: titre has private access in Livre
         return this.titre + ", " + this.auteur.show() + ", " + this.prix;
                     ^
  ```

  Vous avez peut être oublié de changer les droits des variables de `Livre` qui seront utilisées dans les classes enfants `OuvrageMono` ou `OuvrageColl`. En effet, une variable `private` ne sera vu que par sa classe mais pas par ses classes enfants. Testez un droit moins restrictif ...

II.5. Testez la classe `Main` en instanciant une bibliothèque et en y ajoutant une instance de `OuvrageMono` ou `OuvrageColl`.

Vous devriez avoir ce genre d'erreur :

```
./mediatheque/Bibliotheque.java:30: error: cannot find symbol
          m += "["+l.show()+"], ";
                   ^
  symbol:  method show()
  location: variable l of type Livre
1 error
```

Ce message correspond au problème dû à la non définition de la méthode `show()` dans la classe `Livre` (même si vous avez donné à `Bibliotheque` un livre (une instance `OuvrageMono` ou `OuvrageColl` est par héritage un livre) et que la méthode `show()` est définie dans vos deux classes).

Une première méthode pour régler le problème de compilation serait alors de définir une méthode `show()` dans la classe `Livre`. Comme nous ne souhaitons pas instancier directement la classe `Livre`, une bonne solution dans notre cas est de rendre la classe `Livre` abstraite et de créer une méthode abstraite `show()` dans cette classe pour forcer toutes les classes héritant de `Livre` a définir sa propre méthode `show()` pour être concrête. Modifiez votre classe `Livre` en suivant la deuxième solution.

Temps total du sujet : 0 heures et 45 minutes