

# TP JAVA-POO: ArrayList

---

<b>I. Introduction.....</b>	<b>2</b>
<b>II. Modélisation et test en console.....</b>	<b>2</b>
2.1. La classe Livre.....	2
2.2. Manipulation des ArrayList.....	2
<b>III. Conception de l'interface graphique.....</b>	<b>3</b>
3.1. Création de la fenêtre Accueil.....	3
3.2. Intégration des données.....	3
3.3. Mise en place des composants.....	4
<b>IV. Implémentation de la logique métier.....</b>	<b>5</b>
4.1. Fonctionnalité "Ajouter".....	5
4.2. Fonctionnalité "Rechercher".....	6
<b>V. Conclusion.....</b>	<b>7</b>

# I. Introduction

L'objectif de ce Travail Pratique (TD2) était de se familiariser avec deux concepts fondamentaux de la programmation Java : l'utilisation des collections dynamiques (`ArrayList`) pour stocker des objets, et la création d'une interface graphique (GUI) simple avec la bibliothèque Swing.

Le projet consiste à développer une mini-application de gestion de livres permettant l'ajout et la recherche d'ouvrages à partir de leur code ISBN, en s'appuyant sur une classe `Livre` préexistante.

# II. Modélisation et test en console

## **2.1. La classe Livre**

Nous sommes partis de la classe `Livre` fournie, qui définit la structure des données. Elle contient les attributs privés suivants : `ISBN`, `titre`, `Auteur` (de type `String`) et `prix` (de type `int`).

Cette classe inclut :

- Un constructeur pour initialiser un livre.
- Des accesseurs (Getters) et mutateurs (Setters) pour chaque attribut.
- Une méthode `Afficher()` pour visualiser les informations du livre dans la console.

## **2.2. Manipulation des ArrayList**

Dans un premier temps, nous avons modifié le programme principal (`main`) pour ne plus gérer les livres via des variables isolées, mais via une collection dynamique.

- Nous avons instancié une `ArrayList<Livre>`.
- Nous y avons ajouté les livres `livre1` et `livre2` créés précédemment, ainsi que deux nouveaux livres.
- Enfin, nous avons parcouru cette liste pour afficher l'ensemble des ouvrages en utilisant la méthode `Afficher()` de chaque objet

```
--- LISTE DES LIVRES (Partie 1) ---
Titre: Harry Potter, Auteur: JK Rowling, Prix: 20€
Titre: Seigneur des Anneaux, Auteur: Tolkien, Prix: 30€
Titre: Les Misérables, Auteur: Victor Hugo, Prix: 15€
Titre: Dune, Auteur: Frank Herbert, Prix: 25€

Process finished with exit code 0
```

## III. Conception de l'interface graphique

### 3.1. Crédation de la fenêtre Accueil

Nous avons créé une nouvelle classe nommée Accueil héritant de JFrame pour constituer notre fenêtre principale, en suivant le modèle "Application Window".



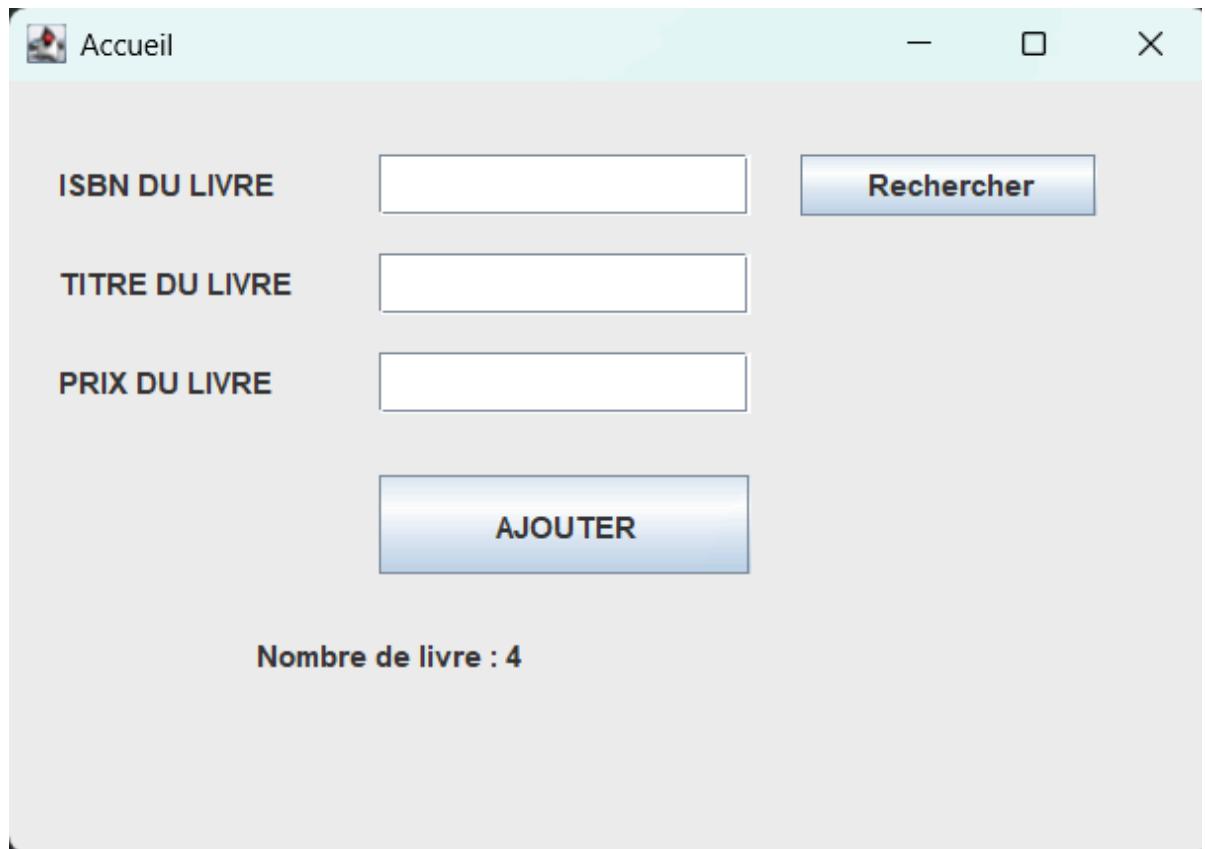
### 3.2. Intégration des données

Au sein de cette classe Accueil, nous avons déclaré un attribut privé `ArrayList<Livre> al` pour stocker les données. Dans le constructeur de la fenêtre, nous avons initialisé cette liste en y ajoutant directement 4 livres par défaut, afin d'avoir des données de test dès le lancement de l'application.

### 3.3. Mise en place des composants

Nous avons reproduit la maquette demandée en positionnant les éléments suivants :

- **Champs de saisie (JTextField)** : Trois champs pour saisir ou afficher l'ISBN, le Titre et le Prix.
- **Libellés (JLabel)** : Pour identifier les champs et afficher le compteur de livres ("Nombre de livre : X").
- **Boutons (JButton)** : Un bouton "AJOUTER" et un bouton "Rechercher".

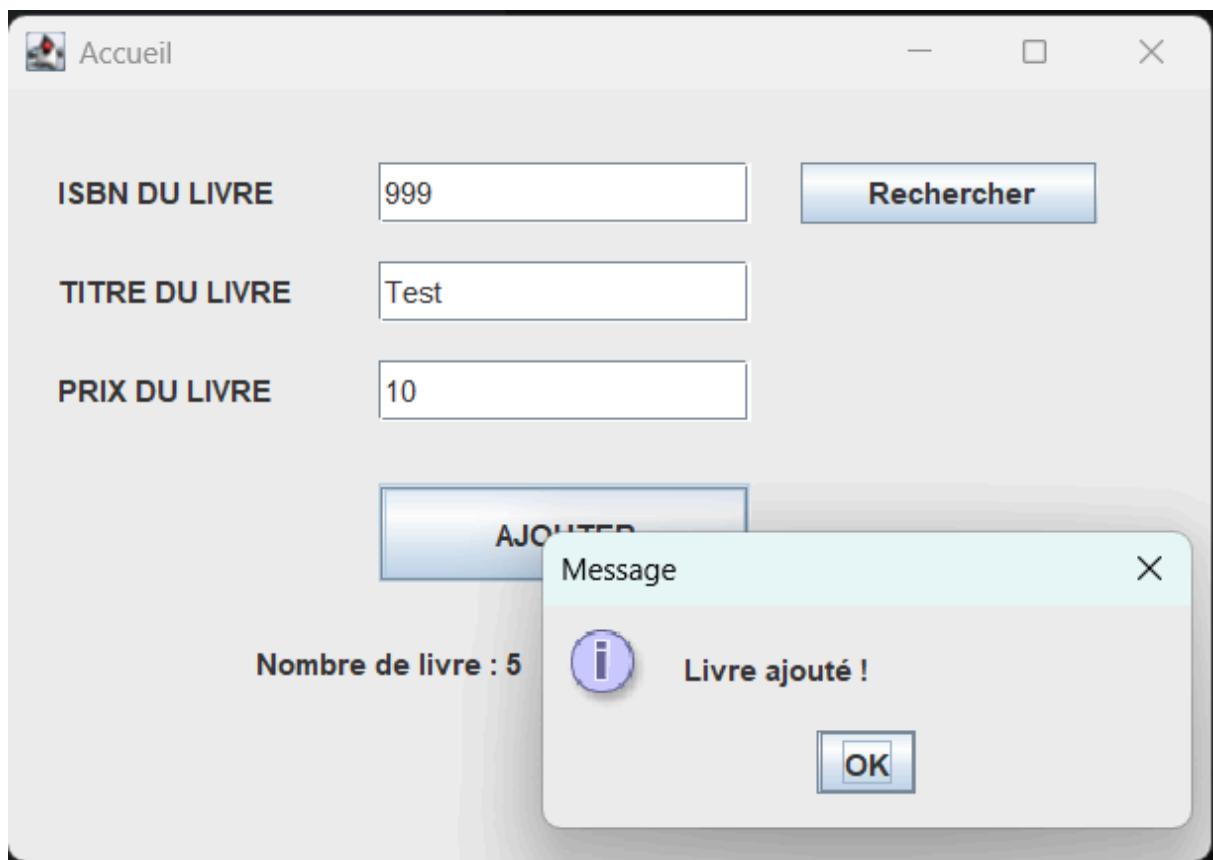


## IV. Implémentation de la logique métier

### 4.1. Fonctionnalité "Ajouter"

Le bouton "AJOUTER" permet de créer un nouvel objet Livre à partir des informations saisies.

- Nous avons récupéré les textes des champs ISBN et Titre.
- Pour le prix, nous avons converti la saisie (String) en entier (int) via `Integer.parseInt()`.
- Le nouveau livre a été ajouté à l'ArrayList `al`.
- Enfin, nous avons mis à jour le label du compteur pour refléter la nouvelle taille de la liste.



## 4.2. Fonctionnalité "Rechercher"

Le bouton "Rechercher" parcourt l'ArrayList pour trouver un livre correspondant à l'ISBN saisi.

**Cas nominal (Livre trouvé) :** Les champs "TITRE" et "PRIX" sont remplis automatiquement avec les données du livre trouvé. Nous avons utilisé `String.valueOf()` pour afficher le prix dans le champ texte.

The figure consists of three vertically stacked screenshots of a Java Swing application window titled "Accueil".

**Screenshot 1:** ISBN DU LIVRE: 1111111 | Rechercher | TITRE DU LIVRE | PRIX DU LIVRE | AJOUTER | Nombre de livre : 4

**Screenshot 2:** ISBN DU LIVRE: 1111111 | Rechercher | TITRE DU LIVRE: Harry Potter | PRIX DU LIVRE: 20 | AJOUTER | Nombre de livre : 4

**Screenshot 3:** ISBN DU LIVRE: 00000 | Rechercher | TITRE DU LIVRE: ERREUR | PRIX DU LIVRE: ERREUR | AJOUTER | Nombre de livre : 4

## V. Conclusion

Ce TP nous a permis de comprendre l'intérêt des `ArrayList` pour gérer un nombre indéfini d'objets, contrairement aux tableaux classiques de taille fixe. Nous avons également appris à lier cette structure de données à une interface graphique Swing, en gérant les interactions utilisateur (clics boutons) et les conversions de types nécessaires. L'application finale est fonctionnelle et permet une gestion basique mais efficace d'une petite bibliothèque.