BIBLIOTHEQUE

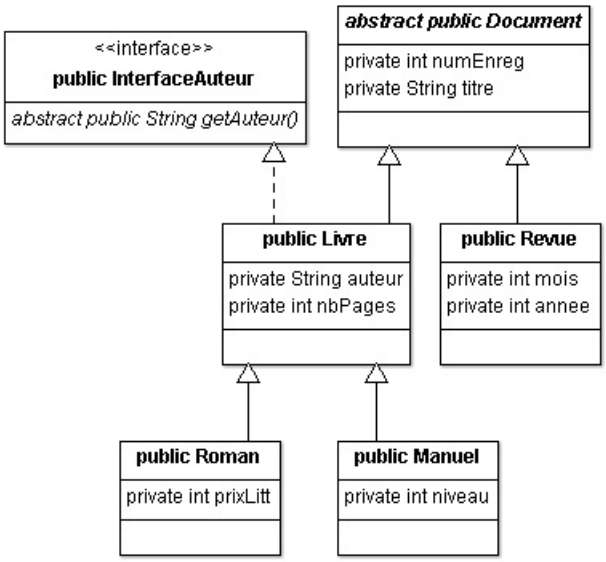
Dans le cadre de la gestion d'une bibliothèque, il s'agit d'écrire une application traitant des documents de nature diverse : des livres, qui peuvent être des romans ou des manuels, des revues, …

Tout document en bibliothèque a un numéro d'enregistrement et un titre. Les livres ont, en plus, un auteur et un nombre de pages. Les romans sont des livres qui ont éventuellement obtenu un prix littéraire : GONCOURT, MEDICIS, INTERALLIE, PULITZER, … . Les manuels scolaires sont des livres caractérisés par leur niveau scolaire. Les revues sont des documents datés : mois, année.

RENDU :

- Ne pas s'autoriser de "warnings" (avertissements de compilation).
- La bonne exécution du code développé ne suffit pas, encore faut-il qu'il soit proprement conçu, implémenté et commenté.
- Pour les commentaires, il convient d'utiliser des annotations afin de générer une Javadoc :
- http://www.ukonline.be/programmation/java/tutoriel/chapitre7/page1.php
- https://openclassrooms.com/courses/presentation-de-la-javadoc (pour une fois!)
- http://www.objis.com/formation-java/tutoriel-java-creation-javadoc-eclipse.html
-

Première partie : BASES DE JAVA



1- Créer un projet Java et importer les sources fournis.

Sous Eclipse, créer un projet Java de nom **Bibliotheque** et importer le fichier archive **bibliotheque.jar**.

Le projet contient maintenant trois paquetages : **model**, **view**, **controller**. Ouvrir et prendre connaissance de chacune des classes les composant.

2- Programmer et tester la classe *Document* et ses sous-classes.

Dans le paquetage *model*, écrire les classes **Document**, **Livre**, **Roman**, **Manuel** et **Revue** pour que chacune réponde aux spécifications suivantes :

- satisfaction de la relation d'héritage définie par le diagramme des classes (cf page précédente)
- les attributs d'instance sont ceux définis dans ce même diagramme des classes
- le constructeur de la classe *Document*, ne prend qu'un paramètre : le titre du document. La valeur de l'attribut numEnreg doit être déterminée automatiquement par incrémentation d'un compteur à chaque création d'un document (ainsi, le premier document créé se verra automatiquement attribuer le n° 1, le deuxième le n°2, …).

  Pour les autres classes, le constructeur prend autant de paramètres qu'il y a d'attributs pour une instance de la classe
- un *getter* et un *setter* pour chaque attribut d'instance

- une méthode toString() qui renvoie une description de l'ouvrage concerné.

Tester (méthode main de la classe **controller.TestDocuments**).

3- Programmer la classe *Bibliotheque*.

Ecrire les méthodes définies dans la classe **model.Bibliotheque**.

Tester (dans la méthode main de la classe *controller.TestDocuments*).

4- Développer des utilitaires d'affichage.

Dans la classe **view.Affichage**, écrire deux utilitaires d'affichage :

- une méthode void afficherDocuments(**Collection<Document>** docs) qui affiche tous les ouvrages reçus en paramètre.
- une méthode void afficherAuteurs(**Collection<Document>** docs) qui affiche la liste des auteurs, par ordre alphabétique (cf classe Collections), de tous les ouvrages reçus en paramètre qui ont un auteur (indication : utiliser la méthode getClass() ou bien l'opérateur instanceof).

Tester (dans la méthode main de la classe *controller.TestDocuments*) avec la liste des documents de la bibliothèque.

5- Développer des fonctions de recherche, de suppression :

de documents par titre

de roman par type de prix littéraire (GONCOURT ….)

6- Gérer un ensemble conséquent d'un cinquantaine minimum de documents

a) dans un premier temps le tableau de documents de la classe *controller.TestDocuments* est enrichi d'autres documents

b)dans un deuxième temps, les données sont lus dans un fichier d'extension csv (format tableur) à l'aide la classe Scanner. Une classe ReadDocument est à développer.

A tout moment, l'état de la bibliothèque peut être sauvegardé dans un fichier d'extension csv.

Deuxième partie : COMPLEMENTS JAVA

1- Opérer un tri lexicographique des documents de la bibliothèque

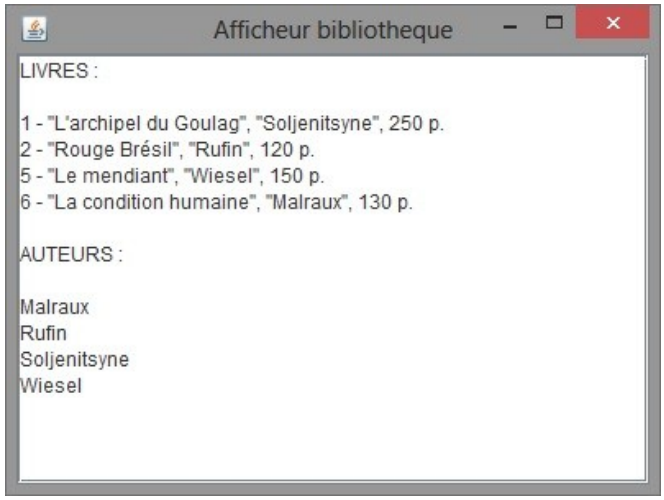
2- Réaliser un clonage profond de votre bibliothèque et réaliser des tests afin de vérifier son bon fonctionnement.

Troisième partie : SWING

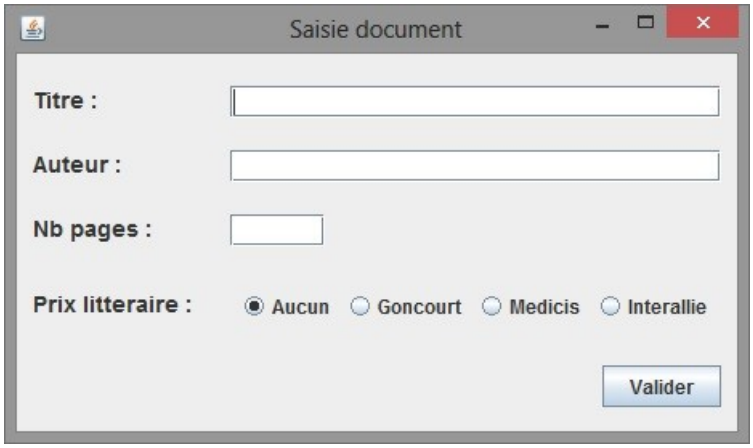
Il s'agit de compléter le programme précédent afin d'offrir des IHM graphiques (GUI). Le développement sera réalisé avec les API Swing.

Créer une fenêtre d'affichage

A l'aide d'un composant **JTextArea** ou tout autre composant de votre choix.



Créer une fenêtre de saisie d'un livre



Le **bouton Valider** permet dans un premier temps d'afficher sur la console les informations saisies et de les mémoriser dans la bibliothèque.

Implémenter les fonctionnalités graphiques correspondant à tous les services développés dans les 2 premières parties.