Atelier5 : Polymorphisme

OBJECTIF DU TP

- Appliquer le principe du polymorphisme de la POO

Objectifs Spécifiques de ce TP

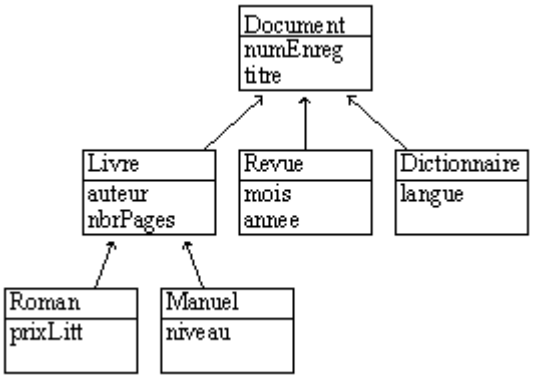
- Utiliser un nouveau EDI (NetBeans)
- Définir des classes avec Héritage
- Redéfinir des méthodes

Travail à faire

Pour la gestion d'une bibliothèque on nous demande d'écrire une application traitant des documents de nature diverse : des livres, des revues, des dictionnaires, etc. Les livres, à leur tour, peuvent être des romans ou des manuels.

Tous les documents ont un numéro d'enregistrement (un entier) et un titre (une chaîne de caractères). Les livres ont, en plus, un auteur (une chaîne) et un nombre de pages (un entier). Les romans ont éventuellement un prix littéraire (un entier conventionnel, parmi : **GONCOURT**, **MEDICIS**, **INTERALLIE**, etc.), tandis que les manuels ont un niveau scolaire (un entier). Les revues ont un mois et une année (des entiers) et les dictionnaires ont une langue (une chaîne de caractères convenue, comme **"anglais"**, **"allemand"**, **"espagnol"**, etc.).

Tous les objets en question ici (livres, revues, dictionnaires, romans, etc.) doivent pouvoir être manipulées en tant que documents.



A. Définir les classes **Document**, **Livre**, **Roman**, **Manuel**, **Revue** et **Dictionnaire**, entre lesquelles existeront les liens d'héritage que la description précédente suggère.

Sachant que tous les attributs de ces classes sont privés, définir :

- le constructeur qui prend autant arguments qu'il y a de variables d'instance et qui se limite à initialiser ces dernières avec les valeurs des arguments,
- une méthode **public String toString()** produisant une description sous forme de chaîne de caractères des objets,
- les variables d'instance sont privés définir également des « accesseurs » get.. permettant de consulter les valeurs de ces variables.

B. Une bibliothèque sera représentée par un tableau de documents. Définir une classe **Bibliotheque**, avec un tel tableau pour variable d'instance et les méthodes :

- **Bibliotheque(int capacité)** - constructeur qui crée une bibliothèque ayant la capacité (nombre maximum de documents) indiquée,
- **void afficherDocuments()** - affiche tous les ouvrages de la bibliothèque,
- **Document document(int i)** - renvoie le $i^{ème}$ document,
- **boolean ajouter(Document doc)** - ajoute le document indiqué et renvoie true (**false** en cas d'échec),
- **boolean supprimer(Document doc)** - supprime le document indiqué et renvoie **true** (**false** en cas d'échec)
- **void afficherAuteurs()** - affiche la liste des auteurs de tous les ouvrages qui ont un auteur (au besoin, utilisez l'opérateur **instanceof**)

C. Ecrire un programme principal permettant de :

- Créer une bibliothèque qui contient les documents suivants :
  - ✓ Num Enreg : 1, titre : java, auteur : Yann hackl et nbre de pages :605 pages
  - ✓ Num Enreg : 2, titre : English, langue :anglais
  - ✓ Num Enreg : 3, titre : Le club des incorrigibles optimistes, auteur : **Jean-Michel Guenassia,** nbre de pages :770 pages et prix littéraire : **GONCOURT**.
  - ✓ Num Enreg : 4, titre : Pour la science, mois : octobre et Année : 2009
- Ajouter le manuel suivant à cette bibliothèque :

  Num Enreg : 5, titre : junior tennis, auteur : mark vale , nbre de pages :60 pages et niveau : 1.
- Afficher tous les auteurs des documents disponibles dans la bibliothèque
- Afficher tous les documents de la bibliothèque.

D. Définir, avec un effort minimal, une classe **Livrotheque** dont les instances ont les mêmes fonctionnalités que les **Bibliotheques** mais sont entièrement constituées de livres. Comment optimiser dans la classe **Livrotheque** la méthode **afficherAuteurs** ?