Travail Pratique 6 : Système de Gestion de Bibliothèque

Objectif

- Utiliser l'héritage et le polymorphisme pour créer un système de gestion de bibliothèque.
- Appliquer les concepts de surcharge, redéfinition, getters et setters.
- Créer un diagramme UML pour illustrer la structure du programme.

Contexte

Vous allez concevoir un système pour gérer les livres et les abonnés d'une bibliothèque. Ce système doit inclure différentes catégories de ressources et types d'abonnés avec des méthodes spécifiques pour chaque type.

Durée

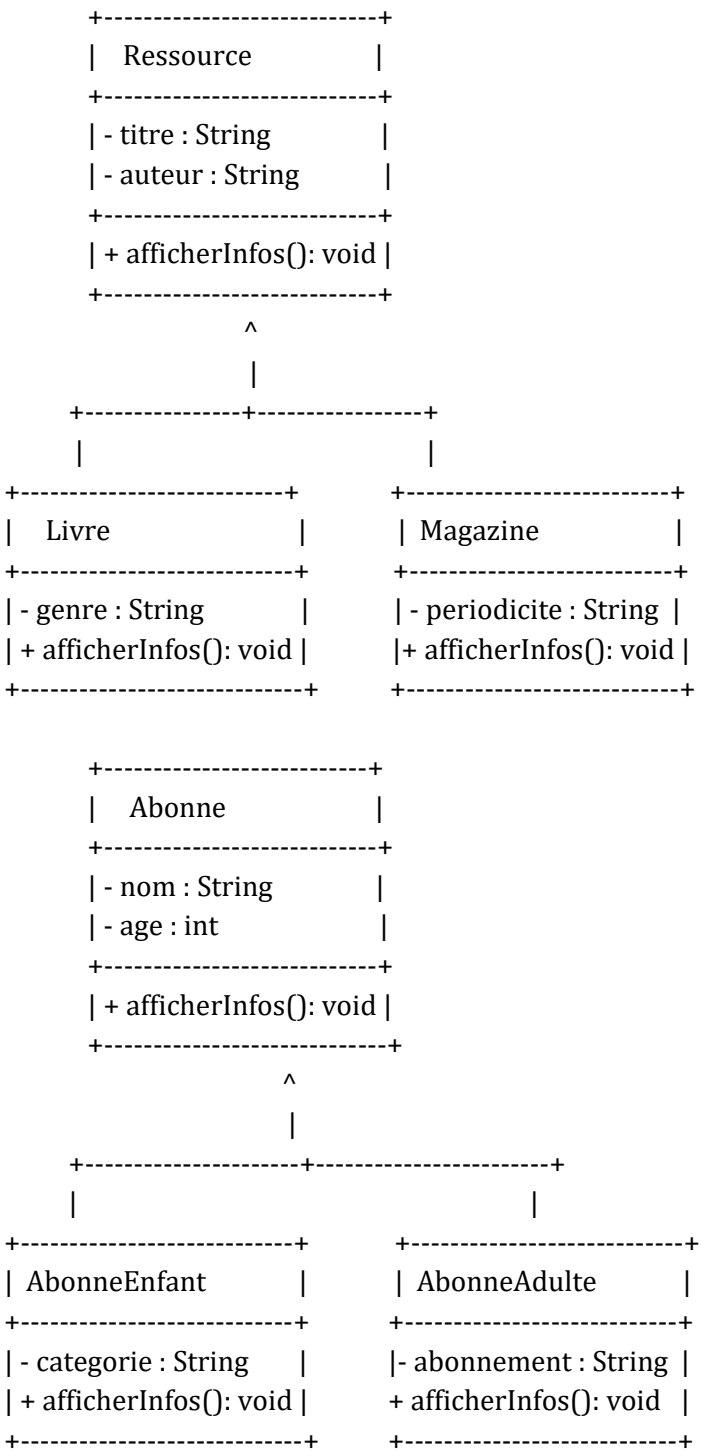
1h30

Diagramme UML

Avant de commencer le codage, réalisez un diagramme UML illustrant les relations suivantes :

- Une classe `Ressource` qui sert de classe parent pour différents types de ressources.
- Des classes enfants `Livre` et `Magazine` qui héritent de `Ressource`.
- Une classe `Abonne` qui représente un abonné de la bibliothèque.
- Des classes enfants `AbonneEnfant` et `AbonneAdulte` qui héritent de `Abonne`.

Diagramme UML suggéré :


Ressource
- titre : String
- auteur : String
+ afficherInfos(): void
Livre
- genre : String
+ afficherInfos(): void
Magazine
- periodicite : String
+ afficherInfos(): void
Abonne
- nom : String
- age : int
+ afficherInfos(): void
AbonneEnfant
- categorie : String
+ afficherInfos(): void
AbonneAdulte
- abonnement : String
+ afficherInfos(): void

Instructions de Codage

Créez la classe `Ressource`

- Attributs privés : `String titre` et `String auteur`
- Constructeur qui initialise `titre` et `auteur`
- Getters et Setters pour `titre` et `auteur`
- Méthode `afficherInfos()` pour afficher le titre et l'auteur

Créez la classe `Livre` qui hérite de `Ressource`

- Attribut : `String genre`
- Constructeur qui initialise `titre`, `auteur` et `genre`
- Getters et Setters pour `genre`
- Redéfinir `afficherInfos()` pour inclure le genre

Créez la classe `Magazine` qui hérite de `Ressource`

- Attribut : `String periodicite`
- Constructeur qui initialise `titre`, `auteur` et `periodicite`
- Getters et Setters pour `periodicite`
- Redéfinir `afficherInfos()` pour inclure la périodicité

Créez la classe `Abonne`

- Attributs : `String nom` et `int age`
- Constructeur pour `nom` et `age`
- Getters et Setters pour `nom` et `age`
- Méthode `afficherInfos()` pour afficher le nom et l'âge

Créez la classe `AbonneEnfant` qui hérite de `Abonne`

- Attribut : `String categorie`
- Constructeur pour `nom`, `age` et `categorie`
- Getters et Setters pour `categorie`
- Redéfinir `afficherInfos()` pour afficher les infos spécifiques

Créez la classe `AbonneAdulte` qui hérite de `Abonne`

- Attribut : `String abonnement`
- Constructeur pour `nom`, `age` et `abonnement`
- Getters et Setters pour `abonnement`
- Redéfinir `afficherInfos()` pour afficher les infos spécifiques

Créez une classe `Main` pour tester le polymorphisme

- Instanciez des objets de type `Livre`, `Magazine`, `AbonneEnfant`, et `AbonneAdulte`
- Utilisez un tableau (ou liste) et appelez `afficherInfos()` sur chaque objet

Exigences supplémentaires

- Utilisez `super` pour appeler le constructeur de la classe parente.
- Assurez-vous que chaque méthode `afficherInfos()` redéfinie contient les informations propres à chaque classe.

Exemple de sortie attendue

Livre : "Le Petit Prince", Auteur : Antoine de Saint-Exupéry, Genre : Fiction
Magazine : "Science & Vie", Auteur : Rédaction, Périodicité : Mensuel
Abonne Enfant : Alice, Âge : 12, Catégorie : Élève
Abonne Adulte : M. Dupont, Âge : 45, Abonnement : Annuel

Questions pour aller plus loin

1. Pourquoi est-il utile de redéfinir `afficherInfos()` dans chaque classe ?
2. Comment le polymorphisme fonctionne-t-il dans ce TP ?
3. Expliquez le rôle de chaque attribut et méthode dans le diagramme UML.