Exercice : Gestion d'une bibliothèque en Java

Contexte

Vous êtes chargé·e de modéliser une petite bibliothèque en Java. La bibliothèque contient des livres et des DVD, qui peuvent être empruntés par des adhérents. Votre mission est de créer les classes nécessaires pour représenter ces éléments et leurs interactions, en utilisant les concepts d'héritage et d'encapsulation.

Partie 1 : Modélisation des ressources de la bibliothèque

1. Classe `Ressource`

Créez une classe `Ressource` qui représente une ressource générique de la bibliothèque. Cette classe doit contenir les attributs et méthodes suivants :

Attributs :

- `titre` (String) : le titre de la ressource.
- `anneePublication` (int) : l'année de publication de la ressource.
- `disponible` (boolean) : indique si la ressource est disponible à l'emprunt.

Méthodes :

- Un constructeur qui initialise les trois attributs.
- Les accesseurs (getters) et mutateurs (setters) pour chaque attribut.
- Une méthode `afficherDetails()` qui affiche les détails de la ressource (titre, année de publication, disponibilité).

2. Classe `Livre` (héritage de `Ressource`)

Créez une classe `Livre` qui hérite de `Ressource`. Ajoutez les attributs et méthodes spécifiques aux livres :

Attributs supplémentaires :

- `auteur` (String) : le nom de l'auteur du livre.
- `nombrePages` (int) : le nombre de pages du livre.

Méthodes :

- Un constructeur qui initialise les attributs de la classe parente et ceux de `Livre`.
- Redéfinissez la méthode `afficherDetails()` pour inclure les informations spécifiques au livre (auteur, nombre de pages).

3. Classe `DVD` (héritage de `Ressource`)

Créez une classe `DVD` qui hérite de `Ressource`. Ajoutez les attributs et méthodes spécifiques aux DVD :

Attributs supplémentaires :

- `realisateur` (String) : le nom du réalisateur du DVD.
- `duree` (int) : la durée du DVD en minutes.

Méthodes :

- Un constructeur qui initialise les attributs de la classe parente et ceux de `DVD`.
- Redéfinissez la méthode `afficherDetails()` pour inclure les informations spécifiques au DVD (réalisateur, durée).

Partie 2 : Modélisation des adhérents

4. Classe `Adherent`

Créez une classe `Adherent` qui représente un adhérent de la bibliothèque.

Attributs :

- `nom` (String) : le nom de l'adhérent.
- `numeroAdherent` (String) : le numéro unique de l'adhérent.
- `ressourcesEmpruntees` (ArrayList de ressources) : une liste des ressources actuellement empruntées par l'adhérent.

Méthodes :

- Un constructeur qui initialise le nom et le numéro de l'adhérent, et initialise la liste des ressources empruntées.
- Une méthode `emprunter(Ressource ressource)` : ajoute une ressource à la liste des ressources empruntées et met à jour sa disponibilité.
- Une méthode `rendre(Ressource ressource)` : retire une ressource de la liste des ressources empruntées et met à jour sa disponibilité.
- Une méthode `afficherRessourcesEmpruntees()` : affiche la liste des ressources actuellement empruntées par l'adhérent.
  - Example de retour:
  - ```
    Détails de l'adhérent Jean Dupont (N°A123) :
        Ressources empruntées :
          - Le Petit Prince (Livre)
    ```
    unkown language

Partie 3 : Test et interaction

5. Classe `Bibliothèque`

Créez une classe `Bibliothèque` avec une méthode `main()` pour tester votre implémentation.

Dans la méthode `main()` :

- Créez au moins deux livres et deux DVD avec des informations variées.
- Créez un adhérent.
- Faites emprunter une ressource à l'adhérent, puis affichez les détails de l'adhérent et des ressources.
- Faites rendre une ressource par l'adhérent, puis affichez à nouveau les détails.

Exemple d'exécution attendu

Voici un exemple de sortie attendue après l'exécution de votre programme :

Détails du livre :

Titre : Le Petit Prince

Auteur : Antoine de Saint-Exupéry

Année : 1943

Pages : 96

Disponible : false

Détails de l'adhérent Jean Dupont (N°A123) : Ressources empruntées :
    ```

- Le Petit Prince (Livre)

Consignes supplémentaires

- Respectez les principes d'encapsulation (attributs privés, accesseurs/mutateurs publics).
- Utilisez le mot-clé `super` dans les constructeurs des classes filles pour appeler le constructeur de la classe parente.
- Gérez les cas où une ressource n'est pas disponible lors d'un emprunt.

Points à vérifier

- L'héritage est correctement utilisé pour éviter la duplication de code entre `Livre` et `DVD`.
- Les méthodes `afficherDetails()` sont bien redéfinies dans les classes filles.
- La liste des ressources empruntées est correctement mise à jour lors des emprunts et des retours.