

## EXERCICES

### Gestion d'une bibliothèque (1/7)

- **Créer une classe Livre contenant 4 attributs:**
  - titre, prixLocation, nbCopies, pourAdultes
- **Prévoir des valeurs par défaut**
- **Créer une classe Bibliotheque contenant une méthode « main »**
  - `public static void main(String[] args){...}`
- **Instancier la classe Livre dans le main**
  - Déclarer deux variables de type Livre
  - Leur attribuer la référence d'une instance de Livre puis d'une autre
- **Analyser le processus de création de variable, de création d'objets, et d'assignation**
- **Rendre les attributs prixLocation et titre publics**
- **Modifier le titre, le prix et le nb de copies du livre dans le main**
- **Développer des méthodes permettant d'ajouter une copie, d'augmenter le prix et de vérifier si le livre est pour adultes**
- **Forcer l'encapsulation des attributs**
- **Permettre aux utilisateurs de connaître le prix d'un livre**
- **Appeler toutes ces méthodes dans le main**

## EXERCICES

### Gestion d'une bibliothèque (2/7)

- **Créer un constructeur qui initialise tous les attributs et qui assure que des données valides sont passées**
  - Nb de copies > 0
  - Prix de location positif
  - Titre réellement mentionné
- **Corriger les appels aux constructeur dans le « main »**
- **Ajouter un attribut qui permette de stocker le nombre maximum de locations autorisées pour chaque livre et adapter le constructeur**

## EXERCICES

### Gestion d'une bibliothèque (3/7)

- **Créer une classe « Etagere »** constituée d'un numéro de rangée et d'un numéro d'aile
- **Créer 3 méthodes + 1 constructeur:**
  - setCoordonnees(int, int): void
  - getRangee(): int
  - getAile(): int
- **Créer une classe « Exempleire »** qui représente une copie d'un livre et est caractérisée par
  - Un numéro unique, un statut, un nombre de locations effectuées, une étagère
- **Créer le constructeur, une méthode pour obtenir le numéro de la copie, et une méthode pour savoir si la copie est actuellement disponible ou non**
- **Créer une méthode pour louer une copie qui attende le nombre maximum d'emprunts autorisés (int).**
  - Si la copie est disponible et si le nombre maximum d'emprunts n'est pas encore atteint:
    - Changer le statut de la copie,
    - Incrémenter le nombre d'emprunts effectués,
    - Afficher à l'écran le nombre d'emprunts effectués jusqu'ici
    - Renvoyer le numéro de l'exemplaire
  - Dans le cas contraire: renvoyer « -1 » pour informer l'utilisateur que la location n'est pas possible
- **Créer une méthode**
  - etagereOccupee(Etagere): boolean**qui indique si l'étagère indiquée en argument correspond à celle utilisée par la copie**

## EXERCICES

### Gestion d'une bibliothèque (4/7)

- **En supposant que la bibliothèque ne peut avoir qu'une copie de chaque livre, adaptez la classe Livre** pour qu'elle puisse conserver la référence de sa copie
- **Adaptez également la méthode « ajouterCopie »** pour qu'elle attende en paramètre le statut de la nouvelle copie, son nombre de locations déjà effectuées et son étagère. Cette méthode incrémentera le nombre de copies de 1 et créera l'instance correspondante de la classe Exempleire en lui fournissant automatiquement son numéro
- **Créer une méthode « louerLeLivre »** dans la classe Livre qui appellera la méthode de location de sa copie et renverra à son expéditeur la réponse renvoyée par la copie elle-même

## EXERCICES

### Gestion d'une bibliothèque (5/7)

- Instancier la classe `Etagere` dans le main
- Créer un objet `Livre` dans le main et appeler sa méthode « `ajouterExemplaire` » à laquelle vous repasserez des valeurs définies dans des variables locales (en ce compris l'objet `étagère` créé plus haut)
- Louer ensuite le livre créé et comparer le nombre de fois que la copie a été utilisée avec la valeur de votre variable locale utilisée pour ajouter l'exemplaire
- Dans la méthode « `ajouterExemplaire` » de la classe `Livre`, appelez la méthode « `setCoordonnees` » de la classe `Etagere` pour modifier les coordonnées de l'étagère reçue en argument. Revenez dans le main et affichez les coordonnées de l'étagère locale. Que constatez-vous? Qu'en déduisez-vous? Corrigez la classe `Etagere` pour éviter ce problème.

## EXERCICES

### Gestion d'une bibliothèque (6/7)

- Modifier la méthode `ajouterExemplaire` de la classe `Livre` pour qu'elle renvoie une référence de l'objet `Exemplaire` qu'elle crée.
- En une seule instruction, créer une variable locale qui indique si une étagère créée pour la circonstance est actuellement occupée par l'exemplaire retourné par la méthode « `ajouterExemplaire` » d'un livre créé pour la circonstance également.
- Surchargez la méthode « `ajouterExemplaire` » de la classe `Livre` pour qu'elle puisse fonctionner avec un seul argument: l'étagère donnée
- Appelez la version surchargée sur un nouveau livre créé dans le main.

## EXERCICES

### Gestion d'une bibliothèque (7/7)

- Créer deux sous classes de Livre: LivreALouer et LivreAVendre en maintenant dans Livre ce qui est commun aux deux sous-classes et en descendant dans LivreALouer tout le reste. Ajouter les attributs et méthodes manquant(e)s dans la classe LivreAVendre.
- La notion d'exemplaire n'étant justifiée que dans le cas des livres loués, adapter les classes en fonction et créer une fonction d'ajout de copie dans la classe LivreAVendre qui attende juste le nombre de copies à ajouter et une fonction de vente de livre qui attende le nombre de copies vendues.
- Quels attributs de la classe Livre devraient être rendus protected pour permettre aux deux sous-classes de fonctionner correctement en l'état?
- Créer une instance de la classe LivreALouer dans le main, ajouter une copie et enregistrer une location.
- Transformer l'attribut pourAdultes de la classe Livre en une chaîne de caractères « lecteursAutorises »:
  - quelles modifications cela entraîne-t-il?
  - comment éviter ces problèmes?