Etagère de livres : Chaînes de caractères et gestion des exceptions.

Soit la classe Livre :

Livre
- isbn : int - titre : String - edition : String
Livre(int isbn,String titre,String edition) + Livre getLivre() + String getTitre() + Livre clone()

La méthode clone permet de cloner un livre.

Soit la classe Etagere qui représente une étagère qui peut contenir un certain nombre de livres. Les livres de l'étagère seront stockés dans un tableau.

- Le constructeur prend en paramètre le nombres de livres que peut contenir l'étagère.
- Méthodes :
  - void afficher() : affiche la liste de tous les livres présents sur l'étagère.
  - void ajouterLivre(Livre livre) ajoute un livre à la fin de l'étagère (impossible d'ajouter un livre dans une étagère pleine).
  - void ajouterLivre(Livre livre,int pos) :ajoute un livre à la position spécifiée (impossible d'ajouter un livre dans une étagère pleine).
  - String lireTitre(String isbn) :affiche le titre du livre dont on donne le numéro isbn.

Remarque :

La gestion de la capacité de l'étagère et de l'existence du livre recherché doit se faire en utilisant le principe de gestion des exceptions java.

Informatisation d'une médiathèque.

Présentation.

Notre médiathèque contient un certain nombre de documents à l'emprunt. Les personnes désirant emprunter des documents doivent être inscrites en tant que client.

Tout client est caractérisé par les attributs suivants :

- *nom* de type String
- *tabEmprunt* : un tableau de type int (un client peut avoir max 4 emprunts en cours)

Le constructeur reçoit en argument le nom du client.

Tout client possède les méthodes

- *void ajoutEmprunt(int numeroDocument)* qui ajoute si c'est possible le numéro du média emprunté, sinon gère l'exception standard *ArrayIndexOutOfBoundsException*. Puisque le média est maintenant emprunté, penser à changer sa disponibilité...
- *String toString( )* qui renvoie sous la forme d'une chaîne de caractères le nom du client et ses emprunts.
- *getNom( )* : accesseur pour le nom

Les documents disponibles sont des DVD vidéo et des livres.

Chaque document est caractérisé par les attributs suivants :

- numéroDocument de type int qui s'incrémente de 1 à chaque nouveau document
- *titre* de type String
- *genre* de type int, il existe 10 genres numérotés de 1 à 10
- *disponible* de type boolean (qui indique si le document est disponible ou pas), à true lors de la création de l'objet.

Chaque document possède les méthodes

- *void rendreIndisponible( )* qui met l'attribut disponible à *false*
- *String toString( )* qui renvoie sous la forme d'une chaîne de caractères toutes les informations du document.
- Un constructeur qui initialise les attributs avec les arguments qui lui sont communiqués.
- Un accesseur pour le genre.

Les livres ont en plus :

- un attribut *nbPages* de type int
- une méthode *toString ( )* qui renvoie toutes les infos du livre

Les DVD ont en plus :

- un attribut *duree* de type int
- une méthode *toString ( )* qui renvoie toutes les infos du DVD

La classe Mediatheque possède :

Les attributs :

- *lesClients* de type ArrayList<Client>
- *static int tabGenre[]* : tableau de 10 compteurs (1 par genre)

Les méthodes :

- *ajoutClient(String nom)* : ajoute le client à la liste des clients (ne faire aucune vérification)
- *void ajoutEmprunt(String nomClient,int numeroDoc)* qui fait appel à la méthode *ajoutEmprunt* de la classe Client pour le client concerné. On considère que le client auquel on ajoute un emprunt est bien inscrit à la médiathèque.
- *afficheStat( )* qui affiche pour chaque genre, le nombre d'emprunts.
- *afficheClient( )* qui affiche les clients et leurs emprunts
- *getTabGenre( )* : accesseur pour le tableau des genres

Tester avec une méthode *main* qui crée quelques clients, ajoute des emprunts pour les clients créés et fait appel aux méthodes *afficheClient( )* et *afficheStat( )* de la classe Mediatheque.

Une classe Media existe (vous ne devez pas écrire son code !) :

Constructor Summary

`Media()`

Method Summary

static `java.util.ArrayList<Document>`	`getLesDocuments()` Méthode qui renvoie tous les documents de notre médiathèque

Aide de la classe ArrayList :

java.util
Class ArrayList<E>

Constructor Summary

`ArrayList()` Constructs an empty list with an initial capacity of ten.
`ArrayList(Collection<? extends E> c)` Constructs a list containing the elements of the specified collection, in the order they are returned by the collection's iterator.

Method Summary

boolean	`add(E e)` Appends the specified element to the end of this list.
E	`get(int index)` Returns the element at the specified position in this list.
int	`size()` Returns the number of elements in this list.

Exercice récapitulatif : Gestion des commandes clients d'un magasin.

Soit la classe Client qui possède

- les attributs privés suivants :
  - numClient : numéro du client qui s'incrémente automatiquement de 1 à chaque nouveau client;
  - nbCommandeCredit : indique le nombre de commandes à crédit du client. A tout moment, un client ne peut avoir que maximum 5 commandes à crédit.
  - montantMaxCreditAutorise : montant maximum autorisé du crédit;
  - montantCredit : montant total des commandes à crédit qui ne peut être supérieur au montantCreditAutorise;
  - listeCommandes : liste de toutes les commandes du client (objet ArrayList).

- les méthodes publiques suivantes :
  - Client(int montantMaxCreditAutorise) : constructeur
  - ajouterCommandeComptant (Commande commande) : ajoute une commande à la liste des commandes et renvoie le numéro de la commande ajoutée.
  - ajouterCommandeCredit (Commande commande) : ajoute une commande à la liste des commandes et renvoie le numéro de la commande ajoutée ou 0 si la commande n'a pas pu être ajoutée, incrémente le montantCredit avec le montant de la commande et nbCommandeCredit de 1 ;
  - payerCommande(int numCommande) : paiement d'une mensualité de la commande. Cette méthode incrémente le nombreVersementRealise (cf. classe Commande). Si toutes les mensualités de la commande sont payées, décrémenter le nbCommandeCredit de 1 et montantCredit du montant de la commande entièrement soldée.
  - toString ( ) : renvoie une chaîne de caractères qui contient toutes les informations du client et les informations concernant toutes ses commandes.

Si des accesseurs sont nécessaires, ajoutez-les.

Soit la classe Commande. Une commande est définie par :

- les attributs privés :
  - un numéro qui va permettre de les identifier (numéro qui va s'incrémenter automatiquement de 1 à chaque nouvelle commande) ;
  - le montant de la commande ;

- pour les commandes à crédit on aura en plus :
  - le nombre de mensualités restantes (nbMensRest) ;
  - le nombre de versements déjà effectués (nbVersementEffectue) ;

- Pour toutes les commandes on aura les méthodes publiques suivantes :
    - la méthode toString( ) qui renvoie toutes les informations concernant la commande ;
    - le constructeur de la classe commande qui doit initialiser les attributs ;

  Si des accesseurs sont nécessaires, ajoutez-les.

Soit la **classe Magasin** qui reprend tous les clients du magasin.

- elle possède donc l'attribut privé :
    - listeClients : liste de tous les clients.
- les méthodes :
    - ajouterClient (Client client) : qui ajoute un client et renvoie le numéro du client ajouté ;
    - ajouterCommandeComptant (int numClient,Commande commande) ;
    - ajouterCommandeCredit (int numClient,Commande commande) ;
    - afficher ( ) : qui affiche tous les clients du magasin et pour chaque client toutes ses commandes ;

  Si des accesseurs sont nécessaires, ajoutez-les.

Ecrire une **classe Test** avec la méthode main qui va ajouter des clients et des commandes aux clients.

Remarques :

- **les problèmes engendrés par l'ajout d'une commande à crédit doivent être gérés par la méthode des exceptions.**
- Exemple pour les listes :

List<Commande>lesCommandes=new ArrayList<Commande>( ) (déclaration de la liste avec un template qui indique que tous les objets de la liste sont de type Commande donc plus de cast par la suite).

Pour parcourir une liste :

```
for (Commande c : lesCommandes)
{
        c.une méthode;
}
```