09/04/2020

PV SEANCE 1- ENSEIGNEMENT A DISTANCE-POO

QUESTIONS :

1-L'héritage, c'est quoi et comment le définir avec JAVA ?

L'héritage est la définition d'une classe par extension des caractéristiques d'une autre classe, exemple : on a créé une classe *Vehicule*. Ainsi les classes Automobile et Avion ont pu hériter des caractéristiques de *Vehicule*.

L'héritage, est l'un des mécanismes les plus puissants de la programmation orientée objet, permet de reprendre des membres d'une classe (appelée *superclasse* ou *classe mère*) dans une autre classe (appelée *sous-classe*, *classe fille* ou encore *classe dérivée*), qui en hérite. De cette façon, on peut par exemple construire une classe par héritage successif.

En Java, ce mécanisme est mis en œuvre au moyen du mot-clé *extends*

Exemple :

```
public class Vehicule
{
    public int vitesse;
    public int nombre_de_places;
}

public class Automobile extends Vehicule
{
    public Automobile()
    {
        this.vitesse = 90;
        this.nombre_de_places = 5;
    }
}
```

Dans cet exemple, la classe *Automobile* hérite de la classe *Vehicule*, ce qui veut dire que les attributs *vitesse* et *nombre_de_places*, bien qu'étant définis dans la classe *Vehicule*, sont présents dans la classe *Automobile*. Le constructeur défini dans la classe *Automobile* permet d'ailleurs d'initialiser ces attributs.

En Java, le mécanisme d'héritage permet de définir une hiérarchie de classes comprenant toutes les classes. En l'absence d'héritage indiqué explicitement, une classe hérite implicitement de la classe *Object*. Cette classe *Object* est la racine de la hiérarchie de classe.

La classe *Object*

Au moment de l'instanciation, la classe fille reçoit les caractéristiques qui peuvent être héritées de sa super-classe, qui elle-même reçoit les caractéristiques qui peuvent être héritées de sa propre superclasse, et ce récursivement jusqu'à la classe *Object*.

Ce mécanisme permet de définir des classes génériques réutilisables, dont l'utilisateur précise le comportement dans des classes dérivées plus spécifiques.

Il faut préciser que, contrairement à C++, Java ne permet pas l'héritage multiple, ce qui veut dire qu'une classe dérive toujours d'une et d'une seule classe.

2-Combien d'acteurs définir ?

Pour faire simple, on utilisera pour **débuter** un seul acteur qui est le bibliothécaire qui se chargera des différentes opérations , nous pourrons par la suite rajouter l'adhérant qui lui effectuera les opérations qui lui sont autorisées .Dans ce cas il faudra prévoir un menu qui gère l'application sous forme de choix ou de possibilité et relatif a chacun des deux acteurs .

3-L'authentification ?

Vérifier si l'adhérant est inscrit.

4- Peut-on supprimer un document au niveau de la mémoire ?

Java utilise la mémoire gérée. Par conséquent, la seule façon d'allouer de la mémoire consiste à utiliser l'opérateur new. La seule façon de libérer de la mémoire consiste à faire appel au garbage collector.

Vous pouvez également appeler System.gc() pour suggérer que le ramasse-miettes s'exécute immédiatement. Cependant, le Java Runtime prend la décision finale et non votre code.

Selon la documentation Java :

L'appel de la méthode gc suggère que la Java machine virtuelle déploie des efforts considérables pour recycler les objets inutilisés afin de rendre la mémoire actuellement disponible disponible pour une réutilisation rapide. Lorsque le contrôle est renvoyé à partir de l'appel de la méthode, la machine virtuelle Java a fait de son mieux pour récupérer l'espace de tous les objets ignorés.

Je vous conseille tout simplement d'écraser l'objet (considéré que la case est vide)

5-Doit on créer une base de donnée pour les documents

Non

6-Doit utiliser des interface graphiques

Non , pas besoin de AWT- SWING , penser a un menu qui s'affiche de manière régulière après chaque opération .

7-Comment améliorer l'application

Prévoir des exceptions (voir la documentation), prévoir l'affichage d'états : par exemple qui a atteint le maximum d'emprunts , qui doit retourner un livre a une date précise…je laisse votre imagination et votre talent faire le reste .