TP1 JAVA

Exercice1

Un élève sera modélisé par la classe Eleve du paquetage gestionEleves. Cette classe possède trois attributs privés :

- son nom type String,
- un ensemble de notes dans un ArrayList<Integer>
- une moyenne de type **double**. Un élève sans aucune note sera considéré comme ayant une moyenne nulle.

La classe Eleve possède un constructeur permettant uniquement d'initialiser le nom de l'élève et possède:

- **public double** getMoyenne()
- **public void** ajouterNote(**int** note)
- **public** String getNom()
- **public** ArrayList<Integer> getListeNotes()
- **public String** toString() //retourne le nom les notes et la moyenne

Exercice2

Un groupe d'Eleve(s) sera modélisé par la classe ***GroupeEleves*** du paquetage gestionEleves de la façon suivante:

La classe GroupeEleves possède un attribut privé : une collection d'Eleve(s) nommée listeEleves, de type ArrayList<Eleve>.

La classe GroupeEleves ne possède pas de constructeur explicite.

La classe GroupeEleves possède aussi cinq méthodes publiques :

- ***public int nombre()*** //renvoie le nombre d'Eleve(s) contenus dans listeEleves
- ***public ArrayList<Eleve> getListe()*** //renvoie listeEleves.
- ***public void ajouterEleve(Eleve eleve)*** //ajoute l'Eleve reçu en paramètre à listeEleves.
- ***public Eleve chercher(String nom)*** //renvoie l'Eleve dont le nom est indiqué par le paramètre
- ***public void lister()*** //écrit à l'écran la liste des Eleve(s). Elle utilise une ligne par Eleve ; elle utilise la méthode toString de la classe Eleve.

Exercice3:

Complétez la classe Eleve pour faire en sorte d'avoir une classe qui implémente l'interface java.lang.comparable<T>. C'est une interface générique, comme l'indique le <T>.

L'interface ***Comparable<T>*** déclare une seule méthode : ***public int compareTo(T o);*** Quand cette méthode est implémentée, elle doit retourner une valeur strictement négative, nulle ou strictement positive selon que l'objet concerné est plus petit que l'objet o, égal à l'objet o ou

plus grand que l'objet o. (On comparera les élèves selon leur moyenne)

1. Reprendre la classe Eleve écrite précédemment pour la transformer en une classe qui implémente l'interface java.lang.Comparable<Eleve>.
2. Définir la méthode compareTo déclarée par l'interface Comparable. Cette méthode est à nouveau générique ; si la classe implémente Comparable<Eleve>, le paramètre de la méthode doit être de type Eleve : **public int** compareTo(Eleve autreEleve)
3. Modifiez la méthode main de la classe TestEleve pour tester la méthode compareTo.

Exercice4

Il s'agit de modéliser un groupe d'élèves comparables entre eux selon leurs moyennes. On souhaite ajouter à la classe ***GroupeEleves*** deux méthodes :

- une méthode, nommée ***meilleurEleve***, qui retourne l'élève de meilleure moyenne de la liste listeEleves ;
- une méthode, nommée ***trierEleves***, qui trie la liste listeEleves selon l'ordre croissant des moyennes des élèves.

Note:

Utiliser *java.util.Collections.max(listeEleves)*

Et

java.util.Collections.sort(listeEleves)

Exercice5

Créer une classe java ***FileStatistics*** du constructeur ***FileStatistics(String filename)***
Cette classe permet de:

1. Compter le nombre de:
    a. caractères(***getCharCount()***),
    b. mots (***getWordCount()***),
    c. phrase (***getSentanceCount()***)
    d. Paragraphes (***getParagrapheCount()***).
2. Sauvegarder les résultats dans un fichier (***saveCounts(String filename))***
3. Créez une liste de mots distincts avec leurs nombre d'occurrences dans le fichier à l'aide d'un ***HashMap<String, Integer>. (getDictionary())***
4. Sauvegarder le contenue de ce HashMap dans un autre fichier "***dictionnaire.txt***"

NB: utiliser ***line.split("\\s+")*** et ***line.split("[!?.:]+")***

TP2 JAVA

Exercice 1

1. A l'aide de l'interface Runnable Écrire un programme qui crée deux Threads TA et TB, l'un devra compter de 1 à 1000 et l'autre décompter de 1000 à 1.
2. Refaire la même chose en héritant de la class Thread.

Exercice 2

1. Ecrire une classe ***Compteur(String nom, int maxValue)*** qui compte de 1 à MaxValue avec un delay de 100ms. A chaque itération elle affiche son nom et la valeur du compteur.
2. Ecrire une classe TestOrder qui crée et exécute N instances de la classe Compteur.
3. L'affichage sur la sortie standard est il déterministe ou non ?
4. Ajouter dans la fonction run() des délais aléatoires (représentant l'activité à priori inconnue de threads plus complexes)
5. Quel est la principale ressource utilisée concurremment par les threads ?

Exercice 3

1. Ecrire une simple classe ***Valeur(int vleur)*** avec les méthodes public:
    a. Int getVal()
    b. Void add(int i)
    c. String toString()
2. Ecrire une classe ***Ajob(Valeur myVal, int i)*** de type Runnable qui invoque la méthode ***add(i)*** de l'objet ***myVal*** 10^6 fois.
3. Ecrire une classe ***Test*** qui crée un objet **myVal** de type ***Valeur***, crée et exécute deux objet de type ***AJob*** job1(myval, 1) et job2(myVal, -1) puis affiche la valeur stockée dans myVal.
4. Quelle devrait être la valeur stockée par ***myVal*** à la fin du calcul ?
5. Expliquer ce qui arrive, ou se trouve la concurrence ?
6. Comment pourrait-on faire pour que le calcul retourne 0 de façon certaine ?
7. Le programme principal peut il détecter la terminaison des threads ? Afficher le résultat a la fin des deux Thread.
8. Modifier la definition de la methode add() comme suit: ***public synchronized void add(int i)*** verifier le resultat.

Exercice 4

On veut simuler un parking de *N* places. Chaque voiture aura un identifiant (pex 1,2,3,...) et sera simulée par un thread, une voiture se comporte comme suit : elle demande à rentrer dans le parking, quand sa requête est acceptée elle rentre et attend un temps aléatoire puis elle

sort. Le Parking assure qu'il contient moins de *N* voitures il fournit deux méthodes : *enter()* qui retourne *true* si la demande d'entrée est acceptée(et *false* sinon) ainsi que la méthode *quitter()* qui permet à une voiture d'annoncer qu'elle sort.
Écrire le petit simulateur, une voiture demandera à entrer tant que sa demande sera refusée. Ajouter des messages indiquant se que font les voitures. Ajouter au Parking une liste lui permettant d'afficher régulièrement la liste des voitures garées. Quel est le principal problème de cette implémentation ?