Rattrapage JD durée 3h

Seuls documents autorisés :
- Documentation java :
  - Le chemin pour java SE 7 est : /usr/java/DOCS_EXAMEN/docs/api
  - Le chemin pour openjdk-6-doc est : /usr/share/doc/openjdk-6-doc/api
  - S'il y a un bugue à partir du navigateur utiliser /usr/bin/firefox
- le cours et les TP et corrigés de TP

1 Questions de cours

Répondre en deux lignes maximum.

1. Que signifie "this" en Java ? De quel type est-il ?
2. Quelles sont les différences entre une interface et une classe en Java ?
3. Quelle particularité ont les membres déclarés static ?
4. Que signifient les mots clés "final", "protected" et "transient" ?
5. Quelle est la différence entre "redéfinir" et "surcharger" une méthode d'une classe ?

2 Thread et synchronisaiton

Ecrire une classe `Matrice` permettant de représenter une matrice à valeurs entières ayant l lignes et c colonnes. On se propose d'écrire une méthode `public Matrice prodMat(Matrice B) throws Exception` à la classe `Matrice` qui effectue le produite entre deux matrices de manière parallèlle. Pour se faire, après vérification des dimensions des opérandes du produit (et lancement d'une exception en cas d'erreur), on crée m ($m < l$) threads. Chaque thread se charge de calculer quelques lignes de la matrice résultat P :

`Pour j = 0 a l-1`
`P[i,j] =` $\sum_{k=0 \ldots c-1}$ `A[i,k]*B[k,j];`

Ecrire la méthode `prodMat()` en synchronisant les m threads et en garantissant qu'un thread ne calcule une ligne que si celle-ci n'est pas déjà entamée par un autre thread.

3 Distributeur de boissons

L'objectif de cet exercice est de créer une application java qui simule un distributeur de boissons.

1. Créer deux classes principales `ClienLocal` et `DistribLocal` qui vont agir respectivement comme client et comme serveur (en local, donc "communication" par appel direct de méthodes).
2. Créer une classe `Boisson`, représentant une boisson, munie des trois champs `String name`, le nom de la boisson, `int prix`, le prix de la boisson et `int nbStock`, le nombre d'unités restantes en stock.
3. Le stock du distributeur est représenté par un tableau contenant la liste des boissons à vendre par le distributeur. En plus de ce tableau, nommé `Boissons`, la classe `DistribLocal` a quatre autres champs :
   (a) `int solde`, représentant le montant résultant des ventes depuis le dernier chargement.
   (b) `int montant`, représentant le montant courant introduit par le client.
   (c) `String clef`, représentant un mot de passe introduit par un employé afin de vider la caisse du distributeur et remplir ses compartiments.
   (d) `occupe`, indiquant si une boissons est en cours de préparation ou non.
4. Créer une sous-classe `DistException` qui hérite de la classe `Exception` et qui ne définit qu'un seul constructeur (et aucune autre méthode) `DistException(String)` qui appelle le constructeur de la classe mère. Cette exception servira pour des cas tels que "Boisson non disponible" ou "Montant introduit insuffisant".
5. Prévoir les opérations suivantes comme méthodes de `DistribLocal` :

- `public static int DeposerPiece(int somme) throws DistException`.
  L'unique argument de cette méthode représente la valeur de la pièce introduite par le client. La méthode teste si une vente est en cours ou que le mentant cumulé dépasse un seuil prédéfini, auquels cas elle lève une exception de type `DistException` et retourne la somme introduite. Dans le cas contraire, la fonction incrémente le montant introduit pour l'achat en cours.
- `public int select(String name) throws DistException`.
  Si le produit de nom `name` n'est pas disponible (en le cherchant dans `Boissons` et en testant le nombre d'unités restantes) ou si le montant introduit est inférreur à son prix, la méthode génère une exception de type `DistException`. Sinon, elle prépare la boisson demandée (simulation avec un écoulement de temps), met à jour les informations concernant la quantité en stock et le solde du distributeur et renvoie la monnaie restante (s'il y en a).
- `public int annuler(String name)`.
  Elle annule l'achat en cours, s'il y en a, et retourne la monnaie introduite par le client.
- `public int charger(String clef, int qte) throws DistException`.
  Permet à un employé de récupérer le solde du distributeur et de charger tous ses compartiments. Dans le cas où l'argument donné ne correspond pas à la clef du distributeur, la méthode lève une exception de type `DistException`. Nous supposons que le nombre maximum d'unités est le même pour tous les types de boissons du distributeur et vaut `qte`.
- La méthode `main(...)` se trouvera dans `ClientLocal` et proposera à l'utilisateur un menu textuel afin qu'il puisse acheter une boisson. Les entrées seront effectuées avec des appels à `readLine()` sur une référence créée comme suit :
  `BufferedReader stdin = new BufferedReader(new InputStreamReader(System.in));`
  Les sorties seront effectuées avec des `System.out.println(...)`.