TP 4 : Systèmes d'enchères

encapsulation, cycle de vie d'une application orientée objet

Avant de démarrer le TP, vérifiez que vous n'avez pas atteint votre quota d'espace de stockage autorisé :

- placez-vous dans votre $HOME et utilisez les commandes suivantes :
  - `du -sh` pour voir combien d'espace vous avez déjà utilisé
  - `du -sh *` pour voir combien d'espace vous avez déjà utilisé pour chaque fichier (sans fichiers cachés)
  - `du -sch .[!.]* *` pour voir combien d'espace vous avez déjà utilisé pour chaque fichier, y compris les fichiers cachés
- Supprimez les fichiers inutiles.
- Pour éviter des problèmes durant vos TPs d'informatique, vous devriez toujours **garder 300-400 Mo d'espace libre**.

CONSIGNES :

- Sauf indication contraire, **tous** les attributs que vous allez déclarer dans ce TP (et dans les TPs qui suivent) doivent être privés (`private`).
- À priori, la plupart des méthodes devraient être déclarées publiques (`public`). Vous pouvez tout de même déclarer et utiliser des méthodes `private` du moment qu'elles vous sont utiles et votre programme fonctionne correctement.
- Pensez à respecter les conventions de nommage `Java` (vues en cours ou disponibles sur le site d'Oracle)

Le site du TP où vous allez trouver le lien pour forker :
https://github.com/IUTInfoMontp-M2103/TP4

Exercice 1

Vous êtes chargé de développer le système d'enchères **iBaille**. Pour avoir une idée globale du système, voici le principe général de fonctionnement :

— des produits sont mis en vente avec un prix initial (le prix de base) ;
— des utilisateurs peuvent enchérir sur les produits jusqu'à ce que l'enchère soit arrêtée ;
— pour pouvoir participer les utilisateurs doivent payer un coût de participation (différent pour chaque produit) ;
— à chaque enchérissement, le prix de base du produit augmente ;
— à la fin de la vente, l'utilisateur ayant proposé le prix le plus élevé, remporte le produit ;
— pour éviter des enchères inutiles (de 1 centime par exemple), le même pas d'enchère minimal est défini pour tous les produits ;
— lorsqu'un utilisateur *Toto* propose un prix pour un produit, il propose également un prix maximal qu'il est prêt à débourser en cas d'enchère concurrente ; ainsi, si un autre utilisateur *Lolo*, fera une enchère supplémentaire, le prix courant du produit augmentera automatiquement jusqu'à ce qu'un gagnant soit désigné. Par définition, le gagnant est celui dont le prix courant est supérieur aux prix maximaux proposés par les autres utilisateurs.

Remarque : Un utilisateur peut déposer une nouvelle offre d'enchère sur le même produit sur lequel il a déjà déposé une offre d'enchère. Par exemple, il pourra le faire si son offre a été "battue" par un un autre enchérisseur.

1. Implémentez une classe `Produit` avec les attributs suivants : numéro, description (un texte), prix courant, date de début d'enchère (type `LocalDate` du package `java.time`), heure de début d'enchère (type `LocalTime` du package `java.time`), montant du pas d'enchère minimal, coût de participation. Ajoutez un constructeur avec comme paramètres : le numéro, la description, le prix courant et le coût de participation.

2. Comme indiqué précédemment, le pas d'enchère doit être le même pour tous les produits mais modifiable par l'utilisateur. Proposez une solution dans votre programme pour satisfaire cette contrainte.

   Remarque : Ne pas confondre la notion d'*utilisateur du logiciel* (non-informaticien) et l'*utilisateur-programmeur* qui est censé de se servir de votre application pour poursuivre son développement et pour la maintenance, le débuggage etc. Ici l'utilisateur c'est l'informaticien.

3. Ajoutez à la classe `Produit` une méthode `demarrerEnchere(...)` qui rendra l'objet disponible à l'enchère. Vous ajouterez à cette classe les autres éléments qui vous paraissent nécessaires pour que cette méthode fonctionne. Vous pouvez supposer que la date et l'heure courantes représentent le début. Pour cela vous pouvez utiliser les méthodes `LocalDate.now()` et `LocalTime.now()`, respectivement.

   Ajoutez également la méthode réciproque `arreterEnchere(...)`.

4. Implémentez une classe `Compte` avec les attributs suivants : pseudo, email, adresse, solde de compte. Ajoutez un constructeur approprié.

5. Ajoutez à la classe `Compte` une méthode qui permet de créditer le compte avec une somme donnée.

6. Implémentez une classe `OffreEnchere` qui représentera une enchère proposée par un utilisateur pour un produit donné. Cette classe aura entre autres comme attributs : une date, une heure, un prix "en cours", et un prix maximal autorisé (en cas d'enchère automatique). Ajoutez un constructeur approprié.

   Remarque : nul besoin de passer la date et l'heure en paramètres du constructeur. Vous vous contenterez d'utiliser la date et l'heure courantes.

   Remarque : ici, dans le constructeur, il n'est pas demandé de vérifier que les attributs de l'offre créé sont cohérents avec celles du produit (ce n'est pas la responsabilité de l'objet `OffreEnchere`)

7. Implémentez les méthodes `toString()` pour chacune des 3 classes que vous avez écrites.

8. Ajoutez dans la classe `OffreEnchere`, des méthodes accesseurs (*getters*) pour le prix et le prix maximal.

9. Ajoutez dans la classe `OffreEnchere`, une méthode modifieur (*setter*) pour le prix.

10. Ajoutez à la classe `Compte` le code nécessaire afin que cette classe possède une méthode `creerOffre(...)`, qui : prend en paramètres un produit, un prix courant et un prix maximal ; crée une offre d'enchère et ajoute cette offre à sa liste d'offres d'enchères. Pour vous simplifier la tâche, on vous conseille d'utiliser une structure de données de type liste prédéfinie en Java, comme `java.util.ArrayList` ou `java.util.LinkedList` (mais vous êtes libres d'utiliser d'autres solutions).

11. Ajoutez à la classe `Produit` une méthode `ajouterOffre(...)`, qui prend comme paramètre une offre d'enchère, vérifie si cette offre est **valide** (en vérifiant le pas d'enchère, le fait que la session d'enchère du produit n'est pas arrêtée, etc.) et l'ajoute à la liste d'offres d'enchères de la classe `Produit`.

    Sans modifier les autre classes, pensez à mettre à jour correctement les valeurs de prix des différentes entités de votre application.

12. Simulez votre application dans le programme principal (la classe `IBaille`). Pour cela, vous instancierez des produits (2 au minimum) et plusieurs comptes (3 au minimum). Pour chacun des comptes vous proposerez à l'utilisateur du logiciel (non-informaticien donc) de déposer des enchères pour différents produits en affichant les informations sur le produit et l'offre gagnante en cours. Pensez à testez que les offres d'enchère non-valides ne puissent pas être déposées à un `Produit`. Vous pouvez effectuer cette simulation par des simples affichages sur la console. Pour récupérer les données saisies par l'utilisateur à la console, vous pouvez utiliser la classe `java.util.Scanner` qui permet de "parser" de manière intelligente une chaîne de caractères. Voici un petit exemple de ce que vous pouvez faire avec :

```
import java.util.Scanner;

class IBaille {

   public static void main(String args[]) {

       // on attache un objet Scanner au flux d'entrée associée à la console
       Scanner saisie = new Scanner(System.in);
       System.out.println("Veuillez écrire quelque chose :");

       // récupération de la chaîne de caractères saisie par l'utilisateur
       String réponse = saisie.next();

       // récupération de la chaîne de caractères saisie par l'utilisateur
       // sous forme d'un nombre entier
       int entier = saisie.nextInt();

   }

}
```

Pour plus de détails sur la classe `Scanner`, voir l'API :

https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html