TP 1: Cocktail. A la vôtre !

notions : Création de classes et d'objets simples

On souhaite réaliser un programme Java permettant de représenter un bar simpliste, avec des verres et des bouteilles.

1 La classe Verre - première version

Dans un fichier `Verre.java`, écrire la classe `Verre` ayant les deux attributs entiers, `contenance` et `quantité` ainsi que les méthodes :

- `public int getVolumeVide()` qui retourne le volume vide restant ;
- `public int remplir(int q)` qui ajoute q cl de liquide à la quantité contenue dans le verre.
  On prendra garde à ce que le verre ne contienne pas plus que ce qu'il peut contenir...
  On versera donc le *minimum* du volume vide restant et de `q`.
  Cette méthode retournera la quantité effectivement versée.
- `public int boire(int q)` qui fait l'inverse. Cette méthode retourne la quantité effectivement bue, qui peut être plus petite que `q` si le verre n'a pas assez de liquide...
- `public String toString()` qui retourne une représentation de l'état du Verre sous forme de chaîne de caractères (utile pour l'affichage).

Attention : comme d'habitude en programmation, compilez régulièrement votre code dans le Terminal avec la commande `javac Verre.java`.

Remarque : vous pourrez utiliser la méthode `int min(int, int)` de la classe Java `Math`. Exemple d'usage :

```
1 int a=10, res;
2 res = Math.min(15, a); // maintenant, res vaut 10
```

2 Programme de test de la classe Verre

Dans un fichier `TestVerre.java`, écrire une classe `TestVerre` dotée d'une unique méthode de test de signature `public static void main(String [] arg)` qui exécute la suite d'opérations :

- créer un verre
- affecter sa contenance à 20cl,
- remplir le verre avec 10 cl, puis l'afficher
- remplir le verre avec 15 cl supplémentaires
- afficher la quantité contenue dans le verre.
- boire 10 cl du verre, puis afficher le verre
- boire 20 cl du verre, puis afficher le verre
- affecter la quantité contenue dans le verre à 1000 cl, puis afficher le verre.
  Ce verre est un peu bizarre, non ! ?

Compiler, puis tester avec la commande `java Verre`.

3 La classe Verre - seconde bien plus pertinente...

Notre classe `Verre` est assez sympathique, mais elle est perfectible :
— Lorsque le verre est créé en mémoire avec `new`, sa contenance est initialement nulle. Il est donc un peu absurde... Ce n'est qu'ensuite qu'on fixe sa contenance.
— La classe n'est pas sécurisée : elle ne garantit pas les *invariants de classe*. Un petit malin pourrait, par exemple, mettre la quantité contenue à une valeur supérieure à la contenance. Aie !

Dresser la liste des invariants de classe de la classe `Verre`.

Améliorer la classe Verre en procédant aux modifications suivantes :
— rendre les deux attributs privés (mot clé `private`).
— ajouter un contructeur `Verre(int c)` qui construit le verre avec la contenance c. Initialement, le verre est vide.
— ajouter les accesseurs publics `int getContenance()` et `getQuantite()`, qui permettent d'interroger l'état du verre.

Essayer de compiler et comprendre les erreurs de compilation. Modifier le programme de test ; tester.

4 La classe Bouteille

On va maintenant écrire la classe `Bouteille`.

Doter la classe `Bouteille` de 3 attributs (privés, bien sûr...) :
— un attribut `String nom` : le nom du liquide contenu (`"limonade"`, `"whisky"` par exemple)
— un attribut entier `quantite` : la quantité de liquide restant dans la bouteille
— un attribut booléen `boolean ouverte` indiquant si la bouteille est ouverte ou fermée.

Doter la classe `Bouteille` des méthodes suivantes :
— Contructeur `Bouteille(String nom, int q, boolean ouverte)` qui construit une bouteille de nom `"nom"` avec la quantité de liquide q et pouvant être initialement ouverte ou fermée.
— Contructeur `Bouteille(String nom, int q)` qui construit une bouteille de nom `"nom"` avec la quantité de liquide q. La bouteille est initialement fermée.
— l'accesseur `String getNom()` qui retourne le nom de la bouteille.
— `void ouvrir()`
— `void fermer()`
— `int verser(int q)` qui verse "par terre" la quantité q de liquide de la bouteille. Bien sûr, on ne peut verser plus plus que ce que la bouteille contient. Cette méthode retourne la quantité versée. Attention : la bouteille doit être ouverte. Si elle ne l'est pas, il s'agit d'une erreur : on pourra *lever une exception* pour arrêter la méthode et signaler l'erreur.
— `void verser(Verre v, int q)` qui verse la quantité q de liquide de la bouteille dans le verre v. Cette méthode retourne la quantité effectivement versée. Comme on ne veut pas perdre de ce précieux liquide, on ne versera pas plus que la quantité que peut contenir le verre. Attention, la bouteille doit être ouverte.
— `public String toString()` qui retourne une représentation de la Bouteille sous forme de chaîne de caractères (utile pour l'affichage).

5 Un programme

Ecrire une classe `Tp1` contenant la méthode `public static void main(String [] arg)` qui exécute par exemple la suite d'opérations :
— création d'une bouteille `btl1` de coca de 100 cl, initialement fermée.
— création d'une bouteille `btl2` de whisky de 75 cl, initialement ouverte.

— création d'un verre de contenance 20cl,
— versement de 5cl de whisky dans le verre
— affichage des bouteilles et du verre
— versement de 15cl de coca dans le verre (... que se passe-t-il si la bouteille n'est pas préalablement ouverte ?)
— affichage des bouteilles et du verre
— tentative de versement de 10cl de coca dans le verre
— ferme les bouteilles
— boire le verre en 2 fois
— affichage des bouteilles et du verre

6 Une classe `Bar`

Dans notre petit exercice, une bar sera définie par son nom et une série de n emplacements pouvant chacun contenir une bouteille. Les emplacements seront numérotés de 0 à $n-1$ et seront représentés par un tableau de bouteilles.

Ecrire la classe `Bar` correspondant aux spécifications suivantes.

Attributs :
— `String nom` : le nom du bar
— un tableau `tab` de bouteilles : l'élément d'indice `i` de ce tableau représente le i-ième emplacement du bar. Si `tab[i]` vaut `null`, cela signifie que l'emplacement est vide. Sinon, c'est qu'il contient une bouteille.

Méthodes :
— Le contructeur public `Bar(String nom, int nbEmplacements)` qui intialise l'objet `bar`.
Il doit créer (instancier, avec `new`, donc !) le tableau de bouteille, de taille `nbEmplacements`.
Le bar est initialement vide (pas de bouteille).
— Les accesseurs publics `String getNom()` et `int getNbEmplacements()`.
— la méthode `void ranger(Bouteille b, int numEmplacement)` qui range la bouteille `b` dans l'emplacement numéro `numEmplacement`.
Si l'emplacement contient déjà une bouteille, il s'agit d'une erreur...
— la méthode publique `Bouteille prendre(int numEmplacement)` qui retourne la bouteille de l'emplacement `numEmplacement` et vide l'emplacement.
Si l'emplacement ne contient pas de bouteille, la méthode retourne `null`.
— la méthode publique `Bouteille prendre(String nom)` qui recherche et "prend" dans le bar une bouteille nommée `nom` (une bouteille de "coca" par exemple).
La méthode retourne null si le bar ne contient pas de bouteille du `nom` voulu.
— `public String toString()` qui retourne une représentation textuelle du bar, en listant tous les emplacements et le détail des bouteilles.

7 Le programme de test, la suite

Modifier le programme de test pour :
— créer le bar nommé "OMaitre" qui a 5 emplacements (oui, le bar OMaitre est un petit bar)
— ranger la bouteille de coca et la bouteille de whisky dans des emplacements
— prendre les bouteilles dans leurs emplacements avant de remplir le verre, puis les ranger à nouveau dans le bar.
— afficher le bar à différentes étapes.