TP 1 Conception OO avec UML et Java

Environnement de travail

L'environnement de travail dans lequel nous allons travailler est le suivant :
- Java SE 6.0 de Oracle
- Editeur de texte emacs ou votre éditeur préféré
- Un "terminal" pour le travail en dehors de l'éditeur, pour lancer les commandes javac, java,...
- Dans le navigateur Web que vous préférez ajoutez un marque-pages sur la documentation sur les API (http ://download.oracle.com/javase/6/docs/api/index.html)

1 Exercice 1 : Hello World

1. Compilez la classe `HelloWorld` à l'aide de la commande : `javac HelloWorld.java`
2. Exécuter la classe `HelloWorld` à l'aide de la commande : `java HelloWorld`

2 Exercice 2 : Accesseurs et instances

1. Compléter le code Java de la classe `Car` dans le fichier Car.java en donnant les implémentations des méthodes.
2. Développer une application qui crée deux instances de `Car` et affiche une description (le constructeur, le modèle et sa capacité).

3 Exercice 3 : Accesseurs et modificateurs

1. Compléter le code Java de la classe `House` dans le fichier House.java en donnant les implémentations des méthodes.
2. Développer une application qui crée deux instances de `House` et affiche leurs détails (adresse, ville).
3. Ajouter un modificateur et un accesseur pour l'attribut `theNumberOfRoom`.
4. Développer une application qui crée deux instances de `House`, affecte un nombre de chambres aux maisons, affiche le nombre de chambres et calcul et affiche le nombre de chambres total.

4 Exercice 4 : Constructeurs et surcharges de méthode

1. Ajoutez 2 constructeurs pour avoir 3 constructeurs dans la classe `House` :
   - un constructeur qui n'a pas de paramètre (vous utiliserez le mot clé `this`),
   - un qui prend en paramètre l'adresse et la ville,
   - et le dernier qui prend en plus le nombre de chambres.
2. Utilisez les 3 constructeurs (et éventuellement d'autres méthodes) pour créer 3 maisons de 5 chambres dans la méthode main et affichez les descriptions des maisons et le nombre de chambres total.

5 Exercice 5 : Un peu de java.lang

1. En utilisant un tableau, complétez/corrigez la classe `TestMois` (et éventuellement complétez la classe `Mois`, mais sans modifier le code déjà écrit) pour qu'elle affiche le nombre de jours dans un mois donné (vous ne tiendrez pas compte des années bissextiles), par exemple (attention, 11 correspond au mois de novembre) :

```
prompt: java TestMois 11
Le mois de novembre a 30 jours
prompt:
```

6 Exercice 6 : Redéfinition de méthode, la méthode `toString()`

1. Dans la classe `House`, ajoutez une méthode `display()` qui affiche une description de la maison (adresse, ville et nombre de chambres). Utilisez `display()` dans la méthode `main()`.
2. Ajoutez l'instruction `System.out.println(house)` où house désigne une des maisons que vous avez créés. Vous essaierez de comprendre ce qui est affiché après avoir fait les 2 questions suivantes.
3. Ajoutez une méthode `toString()` qui renvoie une chaîne de caractères qui décrit la maison. Donnez à la méthode `toString()` la même signature que la méthode de même nom de la classe `java.lang.Object` (cherchez dans les API du JDK). Exécutez à nouveau l'application. Voyez ce qui est affiché maintenant par l'instruction `System.out.println(house)`. Miracle ! `println()` utilise automatiquement la méthode `toString()` de la classe de l'objet qu'il a à imprimer. Essayez de trouver une explication rationnelle en faisant la question suivante.
4. Il faut savoir chercher dans la documentation de l'API (javadoc). En partant de la classe java.lang.System et en cliquant sur les liens, retrouvez dans la documentation que `System.out.println(objet)` affiche ce que retourne la méthode toString de la classe de objet.
5. Modifiez la méthode `display()` pour utiliser `toString()`.

7 Exercice 7 : Méthode d'instance, méthode de classe

1. Dans la classe `House` écrivez une méthode d'instance `compare` pour comparer 2 maisons sur leur nombre de chambres. `compare` prend une maison en paramètre et elle renvoie 0 si la maison a le même nombre de chambres que l'instance qui reçoit le message, 1 si l'instance courante (`"this"`) a plus de chambres que le paramètre et `-1` sinon.
2. Dans la même classe écrivez une méthode de classe `compare2` pour comparer 2 maisons (passées en paramètres) sur leur nombre de chambres.
3. Développez une application qui crée deux maisons avec des nombres de chambres différents et qui utilise chacune de ces méthodes.

8 Exercice 8 : Héritage et polymorphisme

1. Créez une classe `GuestHouse` qui hérite des attributs et des opérations de la classe `House`, et qui ajoute l'attribut `hostName`. Créez les constructeur, accesseurs, modificateurs qui vont bien.
2. Créez une méthode `display()` qui permet d'afficher l'adresse et la ville de la maison d'hôte plus le nom de l'hôte.
3. Développer une application qui crée une maison et une maison d'hôte et qui affiche leurs informations.
4. Déclarez une maison m1. Créez une maison d'hôte gh1. Faire m1 = gh1. On a le droit ? Déclarez un String hostName. Faire hostName = m1.getHostName(). Affichez hostName. C'est bon ? Non ? Ça y'est ? Comment avez-vous fait ?
5. Maintenant affichez les informations de m1. Que se passe-t-il ? C'est normal ?
6. Comment faire pour afficher les informations d'un maison sans les informations de la maison d'hôte ?