Classe, objet,identité/égalité, scanner

Exercice 1 - Plaque d'immatriculation

On veut modéliser des numéros de plaques d'immatriculation en séparant le code du département des autres informations.
Pour tous les exercices suivants même si ce n'est pas explicitement demandé il vous faudra tester à chaque fois le code que vous écrivez en écrivant un `main` soit dans la classe courante soit dans une autre classe.

1. Écrire une classe `NumberPlate` contenant 2 champs `stateCode` et `serial` représentant respectivement le code du département (penser aux codes 2A et 2B pour la Corse) et la séquence de chiffres et de lettres qui le précède.
2. Écrire le constructeur prenant en paramètres les valeurs des champs précédents, ainsi que les méthodes `getSerial()` et `getStateCode()`.
3. Écrire la méthode `toString()`.
   A quoi sert l'annotation `@Override` ? Doit-on l'utiliser dans ce cas ?
4. Écrire un autre constructeur prenant en paramètres une chaîne de caractères contenant l'immatriculation entière, et qui en extrait le code du département et le numéro de série pour initialiser les champs de l'objet. On supposera que le numéro de département est toujours composé des deux derniers caractères de l'immatriculation entière.
   Pour vous aidez, vous pouvez regarder, dans la documentation, la classe `java.lang.String`.

Exercice 2 - Référence, comparaison d'objet

Éxecutez le code suivant.

```
   public static void main(String[] args){
         NumberPlate a = new NumberPlate("956 XM 94");
         NumberPlate b = new NumberPlate("956 XM 94");
         NumberPlate c = a;

         System.out.println(a==b);
         System.out.println(a==c);

         System.out.println(a.equals(b));
         System.out.println(a.equals(c));
   }
```

Modifier le code de `NumberPlate` pour que le comportement de cette classe soit plus naturel.
Puis executez le code suivant.

```
   public static void main(String[] args){
         java.util.ArrayList<NumberPlate> list = new
java.util.ArrayList<NumberPlate>();
         NumberPlate a = new NumberPlate("956 XM 94");
         NumberPlate b = new NumberPlate("956 XM 94");

         list.add(a);

         System.out.println(list.indexOf(a));
         System.out.println(list.indexOf(b));
```

```
    }
```

Le comportement du code ci-dessus correspond t'il a vos attentes ? Pourquoi ?
Que doit'on modifier ?
Utiliser l'annotation `@Override` pour vérifier.
Expliquer pourquoi il faut de plus redéfinir la méthode `hashCode()`.

Exercice 3 - La classe Voiture

On souhaite maintenant représenter des voitures. Les informations qui nous intéressent sont le numéro d'immatriculation, la marque du véhicule, le nombre de fenêtres de celui-ci, qui nous servira à calculer le montant d'une taxe, et le niveau d'essence (contenu courant du réservoir, en litres).

1. Écrire une classe `Car` contenant un champ `numberPlate`de type `NumberPlate`, un champ `fuelTank` de type primitif double, ainsi que des champs `brand` et `numberOfWindows` respectivement de type `String` et `int`.
2. Écrire les méthodes « getters » de la classe puis une méthode `toString()` affichant les caractéristiques de la voiture : la marque, l'immatriculation et le niveau d'essence.
3. Écrire la méthode `getTax()` qui calcule le montant de la taxe selon la formule `1000*numberOfWindows`.
4. Déclarez dans la classe une constante, `LITERS_FOR_ONE_HUNDRED_KILOMETERS` dont vous choisirez la valeur, et qui représente la consommation de toutes les voitures en litres pour 100 km.
5. Écrire une méthode `drive` qui prend en paramètres le nombre de kilomètres que doit rouler la voiture et qui consomme la quantité d'essence correspondant. Que faire si la quantité d'essence est insuffisante ?

Exercice 4 - Ya quoi en entrée

1. Écrire un programme Java qui demande à l'utilisateur des mots saisis sur l'entrée standard System.in et qui les affiche sur la sortie standard System.out. On utilisera la classe `java.util.Scanner` pour réaliser les saisies ainsi que les méthodes `hasNext()` et `String next()`.
   Pour ceux qui ne se rappellent plus de leur cours de shell, Ctrl-D permet d'arrêter la saisie sous Unix (Ctrl-Z sous Windows).
2. Modifier le programme pour que s'il y ait un nom de fichier sur la ligne de commande alors le programme utilise les mots du fichier sinon le programme utilise l'entrée standard.
3. Si ça n'est pas déjà fait, modifiez votre programme pour que les mots soient affichés une fois qu'ils ont tous été saisis (utilisez `java.util.ArrayList`).
   Quelle est la différence entre un tableau et la classe `java.util.ArrayList`.
4. Le compilateur indique des messages d'alerte (« warning»). Expliquez pourquoi et faites en sorte de le rassurer.
5. Modifier le programme pour qu'il affiche les mots saisis par ordre lexicographique en utilisant la méthode `java.util.Collections.sort()`.