Interface, factory, classe anonyme

Exercice 1 - Vive l'europe, les plaques changent

Et oui l'europe c'est l'harmonie. Même les plaques d'immatriculation changent. Le nouveau format des plaques d'immatriculation sera :

```
AA 111 AA
```

(2 lettres maximum / 3 chiffres maximum / 2 lettres maximum)

1. Combien de voitures européennes peut-il y avoir ?
2. Dans un premier temps proposez une nouvelle implémentation de la classe `NumberPlate` nommée `EuropeanNumberPlate`.
3. En fait, on souhaite pouvoir gérer la transition, c'est-à-dire avoir un parc de voitures pouvant posséder des plaques à l'ancienne norme ou à la nouvelle.
   Comment doit-on modifier `Car`, `NumberPlate` et `EuropeanNumberPlate` ?
4. Implémenter la solution retenue.
5. Proposez une méthode permettant de calculer le coût de fabrication d'une plaque. On suppose que le coût est proportionnel au nombre de lettres et chiffres utilisés, espace compris et que le coût d'une lettre/chiffre est donné en paramètre.
   Effectuez les changements qui s'imposent sur la hiérarchie de classes.
6. On souhaite pouvoir créer des plaques d'immatriculation en utilisant une méthode possédant un paramètre de type `boolean` permettant de créer des plaques de l'ancien ou du nouveau type.
   Indiquer quel est le profil de cette méthode ?
   Dans quelle classe doit-on mettre cette méthode ?
   Implémenter celle-ci.

Exercice 2 - Manipulation de fichier

Dans cet exercice, on utilisera quelques classes du paquetage `java.io`. Ce paquetage contient une serie de classes qui permettent de manipuler les entrées/sorties et les fichiers ainsi que d'effectuer la serialisation/deserialisation.

1. Ecrire une méthode `printFiles` affichant les fichiers et répertoires d'un répertoire en utilisant la classe `java.io.File`,
   Penser à lire la documentation de `java.io.File`.
2. La méthode `java.io.File.listFiles()` est surchargée, rappelez ce que cela veut dire.
3. Ecrire une méthode `totalSizeOfFiles` calculant le nombre d'octets de l'ensemble des fichiers (pas les répertoires) d'un répertoire.
4. Ecrire une méthode `mostRecentFile` indiquant quel est le fichier ou répertoire le plus récemment utilisé ou `null` si le repertoire ne contient pas de fichier ou répertoire.
5. On souhaite modifier la méthode `printFiles` pour qu'elle n'affiche que les fichiers et répertoires dont le nom commence par une chaine de caractère passée en paramètre.
   Pour filtrer la liste des fichiers concernés on utilisera la méthode `java.io.File.listFile(java.io.FilenameFilter)`.
   1. Dans un premier temps, créer une classe `StartsWithFilter` implémentant l'interface `java.io.FilenameFilter` effectuant le filtrage. Puis changer le code

de la méthode `printFiles` pour quelle utilise le filtre.

2 Dans un second temps, transformer la classe `StartsWithFilter` en classe interne en vous posant la question de la déclarer statique ou pas ?

3 Enfin, transformer la classe `StartsWithFilter` en classe anonyme dans la méthode `printFiles`.

Exercice 3 - Manipulation de fichier (suite)

Dans le but d'ajouter le filtre sur les noms de fichiers aux méthodes `totalSizeOfFiles` et `mostRecentFile`, on souhaite factoriser le code pour que les trois méthodes `listFiles`, `totalSizeOfFiles` et `mostRecentFile` utilisent la méthode ci-dessous :

```
     interface FileAction {
       void perform(File file);
     }
     private static void applyActionOnFiles(File dir,String
prefix,FileAction action) {
       ...
     }
```

1 Ré-écrire la méthode `printFiles` en utilisant la méthode `applyActionOnFiles` et en spécifiant l'action d'affichage sous forme d'une classe anonyme.

2 Expliquer pourquoi la classe anonyme utilisée par `printFiles` peut être stockée dans un champ et l'avantage de faire cela.

3 Ré-écrire `totalSizeOfFiles` en utilisant une inner class. Pourquoi ne peut on pas utiliser une classe anonyme, ici ?

4 Ré-écrire `mostRecentFile` en utilisant une classe interne de méthode plutôt qu'une inner class (juste pour rire).