TP1

Exercice 1 :

Nous souhaitons réaliser une application de gestion de location de Voitures.

1. Une voiture est caractérisée par son ***immatriculation***, sa ***marque*** et son ***prix***. Les attributs de la classe doivent être privés.

Créer la classe ***Voiture***. Il faut prévoir pour cette classe, les méthodes suivantes :

- Deux constructeurs : le premier est sans paramètres et le deuxième accepte comme paramètres toutes les informations concernant la voiture,
- Des méthodes getters et setters pour chacune des propriétés,
- La méthode *toString*(),
- La méthode *equals*(Voiture v) qui teste l'égalité entre la voiture courante et la voiture v (Deux voitures sont dites « égales » s'ils ont la même *immatriculation* et la même *marque*).

2. Un client est caractérisé par un *code*, un *nom*, un *prénom* et un tableau de Voitures. A noter qu'un client ne peut louer plus que 5 voitures en même temps.

Créer la classe *Client* qui doit respecter le principe d'encapsulation. Aussi, la classe Client doit avoir un constructeur qui prend en argument le code, le nom et le prénom. Le code du client doit être initialisé uniquement lors de l'instanciation.

Les méthodes usuelles d'un client sont :

- *void louerVoiture(Voiture v)* : permet d'ajouter une voiture au tableau
- *boolean rendreVoiture(Voiture v)* : permet de rechercher la voiture, passée en argument, dans le tableau. Une fois trouvée, la voiture doit être supprimée du tableau. La méthode *rendreVoiture* retourne *true* si la voiture existe dans le tableau sinon false.
- *void afficherVoitures()* : Affiche la liste des voitures louées par un Client
- *String toString ()* : retourne les informations concernant le client.

3. Réalisation d'un scénario de test

Créer la classe `Test` qui implémente la méthode `main` avec le scénario suivant :

- Créer une instance d'un client ayant les caractéristiques suivante :
  c1(1,"Zouari","Amine")
- Créer trois instances de voitures :
  - `v1("120 TN 1234","Peugeot  206",14.500)`
  - `v2("119 TN 4321","Polo 5",16.000)`
  - `v3("121 TN 5678","Fait Punto",13.200)`
- Rattacher ces trois voitures au client `c1`.
- Afficher les voitures du client `c1`.
- Enlever(Rendre) la voiture `v2` du client `c1`.

Exercice 2 :

Soit l'architecture des tâches donnée par les classes suivantes :

- Une interface `Tache`.
- Une classe `TacheElementaire` qui implémente l'interface Tache.
- Une classe `TacheComplexe` qui implémente l'interface Tache.

Soit le code suivant de l'interface Tache :

```
public interface Tache {
   /** Obtenir le nom de la tâche. */
   public String getNom();
   /** Obtenir le coût de la tâche. */
   public int getCout();
 }
```

Une tâche est caractérisée par **un nom** et **un coût**. Une tâche est soit une tâche élémentaire, soit une tâche complexe qui est alors composée d'un ensemble (collections) de tâches élémentaires.

1. Écrire la classe `TacheElementaire` qui contient 2 attributs Nom et Cout et un constructeur avec deux paramètres.
2. On s'intéresse maintenant à la classe `TacheComplexe` :

Une tâche complexe est composée de n tâches élémentaires structurées dans un tableau nommé sousTaches. Il est ainsi possible d'ajouter une sous-tâche à une tâche complexe, ajouter(Tache) ou de supprimer une sous-tâche, supprimer(Tache). Le coût d'une tâche complexe est la somme des coûts des tâches qui la composent.

Écrire la classe `TacheComplexe` en se référant au code suivant de la classe TestTache1 :

```
public class TestTache1 {
  public static void main(String[] args) {
    TacheComplexe tA = new TacheComplexe("A",5);
    tA.ajouter(new TacheElementaire("A1", 10));
    tA.ajouter(new TacheElementaire("A2", 20));
    TacheElementaire TA3 = new TacheElementaire("A3", 50)
    tA.ajouter(TA3);
    System.out.println("Cout de tA = " + tA.getCout());
    tA.supprimer(TA3) ;
    System.out.println("Cout de tA = " + tA.getCout());
   }
}
```