Interrogation n°1 - Objets

La production finale devra être rendue sous la forme du code Le plus simple est de *zipper* le répertoire du code source, de le renommer de votre nom (ex.: `yunes.zip` si votre nom de famille est yunes) et de l'envoyer par mail à `Jean-Baptiste.Yunes@irif.fr` ou de lui demander de passer le déposer sur sa clé USB.

1. Créer une classe `Voiture` ayant pour attributs une marque et un modèle (ex.: Lancia, GT) et des getters correspondants. Une voiture verra ses attributs positionnés définitivement via un constructeur adéquat.

2. Créer une classe `Personne` ayant pour attributs un nom, un prénom et un âge (on créera les getters correspondants). Les valeurs de ses attributs seront déterminées à la construction de l'objet.

3. On cherche à représenter le fait qu'une voiture peut avoir un propriétaire, un seul propriétaire par voiture, mais une personne peut être propriétaire de plusieurs voitures. On va introduire pour cela dans les classes `Voiture` et `Personne` des champs permettant de représenter la relation. D'un côté un champ `Vector<Voiture> mesBiens`, de l'autre `Personne monPropriétaire`. Modifier les constructeurs si nécessaire. Ajouter des méthodes adéquates permettant d'associer à une personne donnée une voiture qui sera sa propriété :

```
Voiture titine...
Personne jojo...
jojo.addBien(titine); // titine devient propriété de jojo et jojo propriétaire de titine
```

4. On cherche à représenter le fait qu'une voiture est assurée (pas nécessairement par son propriétaire). Comment représenter cette relation (modifier les classes, les constructeurs, ajouter des méthodes si nécessaire, etc).

5. Modifier les fonctions de manipulation la relation «assuré» de sorte que seuls des personnes majeures puissent assurer des voitures. Si la majorité n'est pas atteinte la relation n'est tout simplement pas établie et la fonction correspondante renvoie `false` (pour cela modifier légèrement le prototype de la fonction).

6. Vérifier le bon fonctionnement à partir d'une fonction `public static void test()` comme celle-ci :

```
Voiture gt       = new Voiture("Lancia","GT");
Voiture lc       = new Voiture("Lincoln","Continental");
Voiture m66      = new Voiture("Mustang","66");
Voiture l4       = new Voiture("Citroen","SM");
Personne aldo    = new Personne("Capelli","Aldo",28);
Personne john    = new Personne("Hair","John",32);
Personne barbara = new Personne("Santa","Barbara",22);
Personne baby    = new Personne("Baby","Alone",8);
aldo.addBien(gt);
aldo.assure(gt);
john.addBien(lc);
john.addBien(m66);
barbara.assure(lc);
john.assure(m66);
System.out.println(baby.assure(l4)); // devrait être false
```

7. Écrire une fonction `public static void quiAssure(Voiture v)` permettant d'obtenir l'affichage du prénom suivi du nom de la personne qui assure la voiture, tester avec :

```
quiAssure(m66);
```

8. Écrire une fonction permettant de compter le nombre de voitures que possède une personne `public static void int getNombreDeVoiture(Personne p)` et tester avec :

```
System.out.println(aldo.getNom()+" possède "
                   +getNombreDeVoitures(aldo)+" véhicule(s)");
```

9. Fabriquer une collection de toutes les personnes, ainsi qu'une liste de toutes les voitures créées dans les tests et créer une fonction `public boolean test(Vector<Personne> gens, Vector<Voiture> vehicules)` qui renvoie `true` si toutes les voitures sont assurées par une personne de la liste et si tous les véhicules ont un propriétaire dans la liste) et `false` sinon.
   Aide : la classe `Vector` possède une méthode `boolean Contains(Object e)` permettant de déterminer si l'élément `e` est dans le vecteur ou non.