TD 3
Objets et références

Exercice 1. Le bonneteau des références

Le but de cet exercice est de vous faire manipuler les références en Java. Pour cela, rien de mieux que de jouer au bonneteau !
Nous disposons d'une classe `Godet`, présentée ci-dessous. Un Godet peut cacher une bille ou pas (attribut `bille`).

```
1  public class Godet {
2      boolean bille;
3      public Godet(boolean bille) {
4          this.bille = bille;
5      }
6      public String toString() {
7          if (bille)
8              return "O";
9          else
10             return "_";
11     }
12 }
```

La classe `Godet` est utilisée dans la classe `Bonneteau` pour initialiser trois Godets : l'un contient une bille (g1), et pas les autres.

```
1  public class Bonneteau {
2      public static void main(String[] args) {
3          Godet g1 = new Godet(true);
4          Godet g2 = new Godet(false);
5          Godet g3 = new Godet(false);
6          System.out.println(g1.toString()+" "+g2.toString()+"
           ↪  "+g3.toString());
7      }
8  }
```

A l'exécution, le `main` de la classe `Bonneteau` montre la situation initiale, le premier Godet contient une bille et pas les deux autres :

```
O _ _
```

Question 1. Dessinez l'état de la mémoire dans l'état initial, et les références stockées dans `g1`, `g2` et `g3`. Vous représenterez ces références comme des flèches sur les objets correspondants en mémoire.

Question 2. Première partie : En partant de l'état initial, on écrit :

```
g3 = g1;
g1 = g2;
System.out.println(g1.toString()+" "+g2.toString()+" "+g3.toString());
```

Quelle sera la chaîne affichée ? Justifiez votre réponse en dessinant l'évolution des références à chaque étape.

Question 3. Deuxième partie : On repart de l'état initial. Cette fois on écrit :

```
g2 = g1;
g1 = g3;
g3 = g2;
g2 = g1;
g1 = g3;
g3 = g2;
g2 = g1;
g1 = g3;
System.out.println(g1.toString()+" "+g2.toString()+" "+g3.toString());
```

Vous avez tout suivi ? Alors donnez la chaîne affichée. Justifiez votre réponse en dessinant l'évolution des références à chaque étape.

Question 4. Troisième partie - magie ! : On repart de l'état initial. Cette fois on écrit :

```
g2 = g1;
g3 = g1;
System.out.println(g1.toString()+" "+g2.toString()+" "+g3.toString());
```

Quelle est la chaîne affichée ? Justifiez votre réponse (surprenante pour un jeu de bonneteau) en dessinant l'évolution des références à chaque étape.

Question 5. Quatrième partie - triche ! : On repart de l'état initial. Cette fois on écrit :

```
g1 = g3;
System.out.println(g1.toString()+" "+g2.toString()+" "+g3.toString());
```

Quelle est la chaîne affichée ? Justifiez votre réponse (là encore hors des règles du bonneteau standard...) en dessinant l'évolution des références à chaque étape.

Exercice 2. La classe `Intervalle`

Le but de cet exercice est de vous faire créer vous-même une classe `Intervalle`, qui représente des intervalles à extrémités entières. Des spécifications minimales vous sont données ci-dessous.
Votre classe doit :

- Avoir un constructeur qui permet d'initialiser ses extrémités (gauche et droite). Les deux extrémités sont des entiers, et celle de gauche doit être plus petite que celle de droite.
- Avoir des méthodes `get` et `set` permettant de modifier chaque extrémité. Bien sûr la classe doit respecter les principes de l'encapsulation.
- Avoir une méthode `toString` qui renvoie une chaîne de caractères représentant l'intervalle.
- Avoir une méthode `intersecte` qui prend en paramètre un autre `Intervalle`, et renvoie un booléen indiquant si cet autre intervalle intersecte l'intervalle courant ou pas.

Exercice 3. Machine à voter - Examen terminal de décembre 2014

Les États-Unis utilisent beaucoup des machines à voter électroniques, fournies par la marque Diebold.
Le but de l'exercice est d'écrire un objet `Diebold` simulant une machine à voter, avec deux partis : Républicains et Démocrates.

Votre objet doit avoir les méthodes suivantes (à vous de trouver les attributs nécessaires) :

- Constructeur : doit initialiser tous les votes à zéro et ouvrir le vote.
- `public void finDuVote()` : après l'appel de cette méthode, l'objet n'accepte plus d'enregistrer de nouveaux votes.
- `public void voterRepublicain()` : enregistre un vote pour les Républicains si le vote n'est pas fini.
- `public void voterDemocrate()` : enregistre un vote pour les Démocrates si le vote n'est pas fini.
- `public int getVotesTotal()` : renvoie le nombre total de votes enregistrés
- `public int getVotesRepublicains()` : renvoie le nombre de votes enregistrés pour les Républicains.
- `public int getVotesDemocrates()` : renvoie le nombre de votes enregistrés pour les Démocrates.
- `public String toString()` : renvoie une chaîne de caractères contenant les informations suivantes :
  - si le vote est en cours ou terminé,
  - pourcentage de votes pour les différents partis,
  - gagnant actuel des élections.

Attention à la visibilité de vos attributs, rappelez-vous que cet objet décide de l'élection du président d'une grande puissance→ **principe d'encapsulation** !