

TD de Compléments en Programmation Orientée Objet n° 2 : Objets, classes, encapsulation

I) Questions de cours

À faire sur le sujet.

À propos des QCM : Dans chaque case, on peut inscrire vrai ou faux ou laisser blanc pour ne pas répondre. En contrôle évalué, de score obtenu est le solde du nombre de bonnes réponses moins le nombre de mauvaises réponses, ramené à zéro si négatif.

Exercice 1 : Généralités

- ☐ Un même code-octet JVM est exécutable sur plusieurs plateformes physiques (x86, PPC, ARM, ...)
- Les entités suivantes peuvent être membres d'une classe :
☐ attribut ☐ méthode ☐ classe ☐ constructeur
- ☐ La JVM interprète du code source Java.
- ☐ Le mot-clé **this** est une expression qui s'évalue comme l'objet sur lequel la méthode courante a été appelée.
- ☐ Toute classe dispose d'un constructeur par défaut, sans paramètre.
- ☐ On peut écrire **public** devant la déclaration d'une variable locale pour qu'elle soit visible depuis les autres classes.
- ☐ Dès lors qu'un objet n'est plus utilisé, il faut penser à demander à Java de libérer la mémoire qu'il occupe.
- Examinez le programme suivant et évaluez les propositions.

<pre>1 class Truc { 2 static int v1 = 0; 3 int v2 = 0; 4 public int getV1() { return v1; } 5 public int getV2() { return v2; } 6 public Truc() { 7 v1++; v2++; 8 } 9 }</pre>	<pre>11 12 13 14 15 16 17</pre>	<pre>public class Main { public static void main(String args[]) { System.out.println(new Truc().getV1()); System.out.println(new Truc().getV2()); System.out.println(new Truc().getV1()); } }</pre>
---	---------------------------------	---

- ☐ La ligne 14 affichera "2". ☐ La ligne 15 affichera "1".
9. ☐ La durée de vie d'un attribut statique est liée à celle d'une instance donnée de la classe où il est défini.

Exercice 2 : Constructeurs des classes imbriquées

Soient les classes :

```

1 public class A{
2     private int a;
3
4     public class AA{
5         private int a;
6         public AA(int y){ this.a = y;}
7     }
8
9     public static class AB{
10        private int b;
11        public AB(int x){ this.b = x;}
12    }
13

```

```

14 public A(int a){
15     this.a = a;
16 }
17
18 public class Main{
19     public static void main(String[] args){
20         A unA = new A(2);
21         // A.AA unAA = new A.AA(3);
22         // A.AB unAB = new A.AB(4);
23         // A.AA autreAA = unA.new AA(3);
24         // A.AB autreAB = unA.new AB(4);
25     }
26 }

```

- Parmi les lignes 21 à 24, quelles sont celles pour qui le programme compile encore quand on les « dé-commente » ?
- Dans la classe `AA`, je peux accéder à la valeur `a` de `A`, en faisant
☐ `A.a` ☐ `A.this.a` ☐ `this.A.a`
- même question dans la classe `AB`

Exercice 3 : Classes imbriquées et visibilité

Soit le programme suivant :

```

1 public class A{
2     private int x;
3     private AA aa;
4     public AA aa2;
5     private AB ab;
6
7     private class AA{
8         private int y;
9         public AA(int y){ this.y = y;}
10    }
11
12    public static class AB{
13        private int z;

```

```

14        private AB(int z){ this.z = z;}
15        public static int getYDUnAA(A a){
16            return a.aa.y;
17        }
18    }
19
20    public A(int x, int n, int z){
21        this.x = x;
22        this.aa = new AA(n);
23        this.aa2 = new AA(n*2);
24        this.ab = new AB(z);
25    }
26 }

```

- Dites s'il y a des erreurs de visibilité.
- Soit la classe `Main` suivante :

```

1 public class Main{
2     public static void main(String[] args){
3         A unA = new A(2, 3, 4);
4
5         System.out.println(A.AB.getYDUnAA(unA));
6     }
7 }

```

- Dessinez l'objet `unA` et son contenu. Qu'est-ce qui est affiché ?

Exercice 4 : Encapsulation

Examinez le programme suivant et dites ce qui se passe ? Est-ce que le comportement est conforme aux commentaires du `main` ? Que faut-il corriger et comment ?

```

1 public class Personne{
2     private String nom;
3     private final int numSecu;
4
5     public Personne(String nom, int numSecu){
6         this.nom = nom;
7         this.numSecu = numSecu;
8     }
9
10    public void changeNom(String nom){
11        this.nom = nom;
12    }
13
14    @Override
15    public String toString(){
16        return nom + " " + numSecu;
17    }
18 }
19
20 public class Cours{
21     private String nom;
22     private HashSet<Personne> inscrits;
23
24     public Cours(String nom, HashSet<Personne>
25         inscrits){
26         this.nom = nom;
27         this.inscrits = inscrits;
28     }
29     //return false si pas dans la liste
30     // des inscrits au départ
31     public boolean exclut(Personne p){
32         return inscrits.remove(p);
33     }
34     //return false si déjà dans liste
35     // des inscrits au départ
36     public boolean inscrit(Personne p){
37         return inscrits.add(p);
38     }
39
40     @Override
41     public String toString(){
42         String s = nom;
43         for(Personne p : inscrits){
44             s += "\n" + p;
45         }
46         return s;
47     }
48 }
49
50 public class Main{
51     public static void main(String[] args){
52         //on crée un certain nombre de personnes:
53         Personne p1 = new Personne ("Adèle",
54             1254);
55         Personne p2 = new Personne ("Brian",
56             1287);
57         Personne p3 = new Personne ("Coralie",
58             2546);
59         Personne p4 = new Personne ("Désiré",
60             2546);
61         //on crée un groupe de copains:
62         HashSet<Personne> copains = new
63             HashSet<Personne>();
64         copains.add(p1);
65         copains.add(p2);
66         copains.add(p3);
67         //pour ce groupe, on crée 2 cours:
68         Cours c1 = new Cours("Couture", copains);
69         Cours c2 = new Cours("Karate", copains);
70         //Coralie est exclue du cours de Couture
71         c1.exclut(p3);
72         //Désiré s'inscrit à celui de Karaté
73         c2.inscrit(p4);
74         //Adèle change de prénom, parce que
75         // l'ancien ne lui plaisait pas
76         p1.changeNom("Adeline");
77         System.out.println(c1);
78         System.out.println();
79         System.out.println(c2);
80     }
81 }

```

Remarque : En général, pour des raisons de persistance des données, ce genre de données a vocation à être géré par une base de données en interfaçage éventuel avec un langage de programmation.

II) Programmer

Finir le TD 1 !