Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

Programmation et TP

T1A

Réseau et Sécurité

S16 Membres de classes (mot-clé Static)

Roten Marc

2017/2018

Table des matières

0 Introduction

Avec ce travail, on va s'occupet des membres de classe et aussi introduire ce que définit le mot-clé Static.

1 Exercice 1

1 Indiquer ce qui est affiché lors de l'exécution du programme suivant :

```
01  package s16;
02
03  import s16.gizmo.Gizmo;
04
05  public class Pg1601 {
06    public static void main(String[] args) {
07      Gizmo g = new Gizmo(5);
08      int a = g.a(g);
09      System.out.println(a + "\n" + g.c);
10      a = Gizmo.b(g);
11      System.out.println(a + "\n" + g.c);
12      g = new Gizmo(3);
13      System.out.println(Gizmo.d + g.c);
14      a = Gizmo.d;
15    }
16  }
```

```
17  package s16.gizmo;
18
19  public class Gizmo {
20    private int        a;
21    static  int        b;
22    public  int        c = 1;
23    public  static int d = 2;
24
25    public Gizmo(int g) {
26      a=2; b=1; c *= ++a; d += g;
27    }
28
29    public int a(Gizmo g) {
30      a++; b++; d += g.c;
31      return d;
32    }
33
34    public static int b(Gizmo g) {
35      d += b++; g.c += b;
36      return d;
37    }
38  }
```

Le résultat qui s'affichera à la console est le suivant

10
3
12
6
18

2 Exercice 2 Gizmo

2 Indiquer s'il serait valide (c.-à-d. pas d'erreur de compilation) d'ajouter les instructions suivantes juste *après* la ligne indiquée pour le programme de l'exercice 1 (justifier/expliquer brièvement si nécessaire).

	Ligne	Instruction	OK
1	06	`Gizmo.d++;`	
2	08	`a = Gizmo.b;`	
3	08	`new Gizmo(4).b(g);`	
4	08	`int b = g.b(null);`	
5	08	`a = g.d;`	

	Ligne	Instruction	OK
6	25	`a++;`	
7	25	`this();`	
8	29	`b(Gizmo());`	
9	35	`b = a++;`	
10	35	`g.a++;`	

1 la Variable b de la classe Gizmo est statique, on peut donc l'appeler par le nom be la classe suivi de "."
2 la variable b n'est pas publique, de base non accessible aux autre paquets. il faut lui rajouter Public

3 OK
4 une erreur dpoit être relevée, mettre un try-catch pour gérer cette exception.
5 OK car la variable d de la classe Gizmo est Public static
6 aucune influence car a pointe sur 2 apres le a++ ;
7 besoin d'un paramètre en Int pour le constructeur
8 aucun sens
9 on ne peut pas faire pointer un static sur un private
10 OK

3 Exercice 3 Circle

3 Ajouter les bons modificateurs (`private`, `static`, `final`...) aux déclarations d'attributs et de méthodes de la classe suivante `Circle`, et implémenter les méthodes.

Dans la librairie standard, on trouve ceci pour calculer la racine carrée d'un nombre :

```
package java.lang;
public class Math {
  public static double sqrt(double x);
  ...
}
```

```
public class Circle {
  double  PI = 3.14159265358;
  double  radius;

          Circle(double r);

  double  perimeter(Circle c);
  double  area();
  boolean isSmaller(Circle c);
  Circle  fromArea(double area);

  double  sqrRad() {
    return radius*radius;
  }
}
```

```
public class Circle {
  public static final double PI = 3.14159265358;
  private double radius;
  public Circle(double r);
  public static double perimeter(Circle c);
  public double area();
  public boolean isSmaller(Circle c);
  public static Circle fromArea(double area);
  public double sqrRad() {
    return radius*radius;
  }
}
```

---

4 Exercice 4 Rectangle

4 On souhaite manipuler des objets *Rectangle* avec les fonctionnalités suivantes :

- créer un rectangle en donnant sa hauteur et largeur (nombres à virgules)
- consulter la hauteur, la largeur, ou la surface d'un rectangle
- appliquer un facteur d'échelle sur les dimensions d'un rectangle
- tester si un rectangle est plus grand qu'un autre (surface)

```
Rectangle x;          // (a)
x = new Rectangle(2.1, 3.9);
double a = x.area();
```

```
Rectangle x;          // (b)
x = Rectangle.create(2.1, 3.9);
double a = Rectangle.area(x);
```

Écrire deux versions de cette classe `Rectangle` :

a) une version orientée objet (sans méthodes statiques) (dans le package `s16.vobject`)

b) une version avec 6 méthodes statiques (dans le package `s16.vstatic`)
Dans cette variante b), un constructeur est-il malgré tout nécessaire ? Est-il possible de le supprimer ?

Pour chaque variante, écrire un petit programme `TestRectangle` testant le fonctionnement.

4.1 Version A

```
public class RectangleVersionA {

  private double heigth,length;
  public RectangleVersionA(double hauteur,double largeur) {
    this.heigth = hauteur;
    this.length = largeur;
  }
  public void displayParams() {
    System.out.println(
        "the length of your rectangle is "+this.heigth+
        "\nthe width of your rectangle is "+this.length
        + "\nthe area of your rectangle is "+this.area());
  }
  public double area() {
    return this.heigth*this.length;
  }
  public void scale(double scale) {
    this.length*=scale;
    this.heigth*=scale;
  }
  public boolean isItBigger(RectangleVersionA yourRectangle) {
    if(yourRectangle.area()>this.area()) {
      return true;
    }else return false;
  }
}
```

classe test utilisée

```
public class TestRectangle {
  public static void main(String[] args) {
    RectangleVersionA rectangle = new RectangleVersionA(2,6);
```

```
    RectangleVersionA rectangle2 = new RectangleVersionA(10,20);
    System.out.println(rectangle.isItBiggerThan(rectangle2));
    rectangle.displayParams();
  }
}
```

résultat à la console

```
<terminated> TestRectangle [Java Application] C:\Program Files\Java\jre1.8.0_151\b
true
the length of your rectangle is 2.0
the width of your rectangle is 6.0
the area of your rectangle is 12.0
```

4.2 Version B

```
public class RectangleVersionB {

  private double heigth,length;
  public RectangleVersionB(double hauteur,double largeur) {
    this.heigth = hauteur;
    this.length = largeur;
  }
   public static void displayParams(RectangleVersionB rect) {
    System.out.println(
        "the length of your rectangle is "+rect.getLength()+
        "\nthe width of your rectangle is "+rect.getHeighth()
        + "\nthe area of your rectangle is "+area(rect)+"\n");
  }
   public static double area(RectangleVersionB rect) {
    return rect.getHeighth()*rect.getLength();
  }
  public double getLength() {
    return this.length;
  }

  public double getHeighth() {
    return this.heigth;
  }

  public void scale(double scale) {
    this.length*=scale;
    this.heigth*=scale;
  }
  public static boolean isThe1stBiggerThan(RectangleVersionB
     rect1,RectangleVersionB rect2) {
    if(area(rect1)>area(rect2)) {
      return true;
    }else return false;
```

```
    }
}
```

classe test utilisée

```
public class TestRectangle {
  public static void main(String[] args) {
    System.out.println("rectangle 1");
    RectangleVersionB rect1 = new RectangleVersionB(2,6);
    RectangleVersionB.displayParams(rect1);
    System.out.println("rectangle 2");
    RectangleVersionB rect2 = new RectangleVersionB(4,7);
    RectangleVersionB.displayParams(rect2);
    System.out.println("1 bigger than 2?");
    System.out.println(RectangleVersionB.isThe1stBiggerThan(rect1,
       rect2));
  }
}
```

résultat à la console

```
<terminated> TestRectangle [Java Application] C:\Program Files\Ja
rectangle 1
the length of your rectangle is 6.0
the width of your rectangle is 2.0
the area of your rectangle is 12.0

rectangle 2
the length of your rectangle is 7.0
the width of your rectangle is 4.0
the area of your rectangle is 28.0

1 bigger than 2?
false
```

On peut enlever les méthodes Getters et Setters et les remplacer par des constantes, comme par exemple dans l'exercice 3 avec public static final.

5 Exercice 5

5 Expliquer dans quelle mesure on pourrait adapter les classes suivantes pour qu'elles modélisent des objets *immuables* :
- la classe `Date` (série S14, exercice. 1), version (a), respectivement version (b)
- la classe `Rectangle` (ex. 4 ci-dessus), version (a), respectivement version (b)

6 Conclusion

Grâce à ce Travail, on a pu mieux comprendre ce qu'étaient les membres de classe et la définition du mot-clé Static