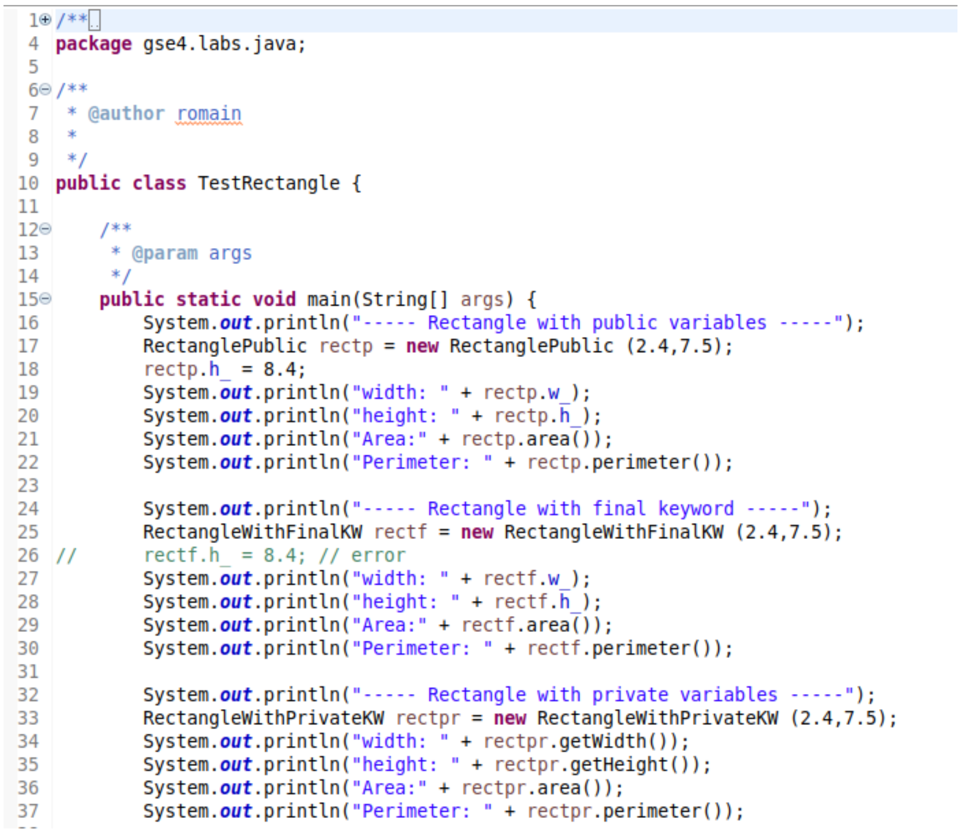
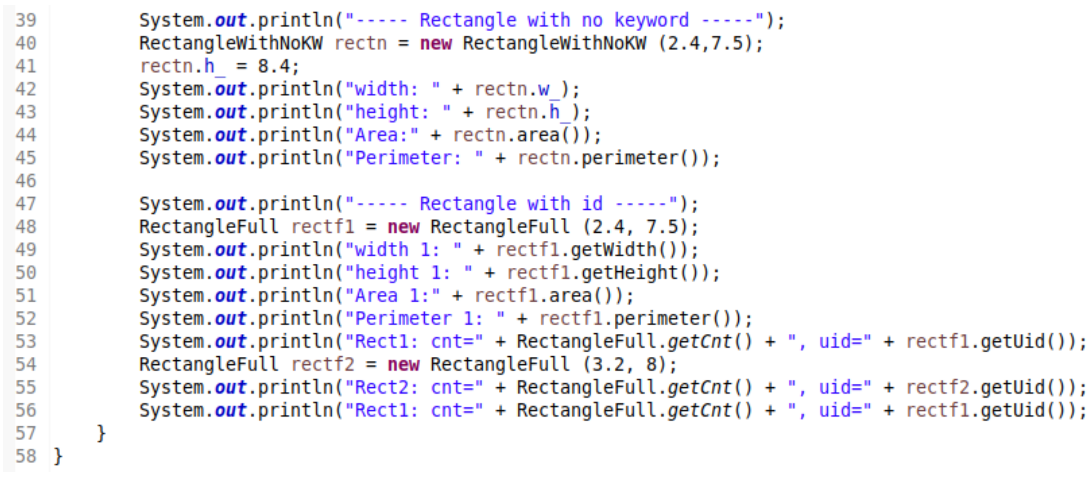
COMPTE RENDU LAB2

TestRectangle

Voici le code de test qui sera utilisé pour obtenir les résultats du rapport.



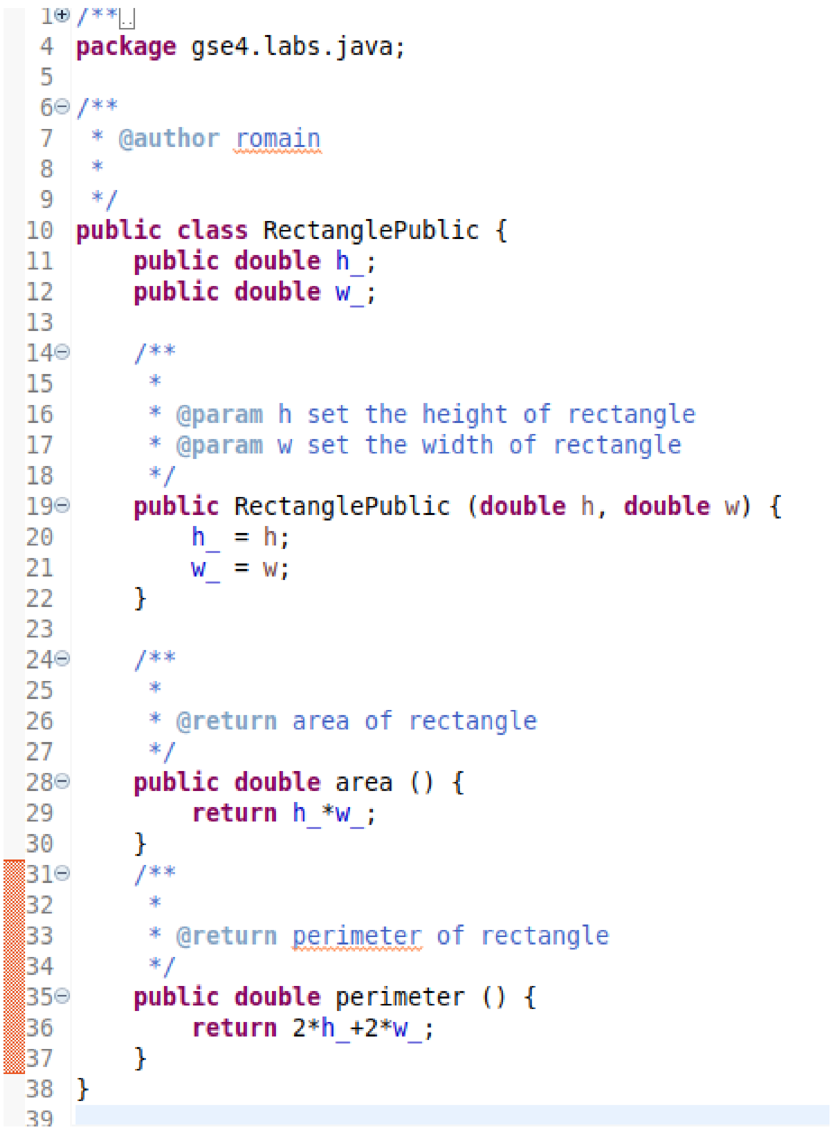
```java
/**
package gse4.labs.java;

/**
 * @author romain
 *
 */
public class TestRectangle {

    /**
     * @param args
     */
    public static void main(String[] args) {
        System.out.println("----- Rectangle with public variables -----");
        RectanglePublic rectp = new RectanglePublic (2.4,7.5);
        rectp.h_ = 8.4;
        System.out.println("width: " + rectp.w_);
        System.out.println("height: " + rectp.h_);
        System.out.println("Area:" + rectp.area());
        System.out.println("Perimeter: " + rectp.perimeter());

        System.out.println("----- Rectangle with final keyword -----");
        RectangleWithFinalKW rectf = new RectangleWithFinalKW (2.4,7.5);
//      rectf.h_ = 8.4; // error
        System.out.println("width: " + rectf.w_);
        System.out.println("height: " + rectf.h_);
        System.out.println("Area:" + rectf.area());
        System.out.println("Perimeter: " + rectf.perimeter());

        System.out.println("----- Rectangle with private variables -----");
        RectangleWithPrivateKW rectpr = new RectangleWithPrivateKW (2.4,7.5);
        System.out.println("width: " + rectpr.getWidth());
        System.out.println("height: " + rectpr.getHeight());
        System.out.println("Area:" + rectpr.area());
        System.out.println("Perimeter: " + rectpr.perimeter());
```



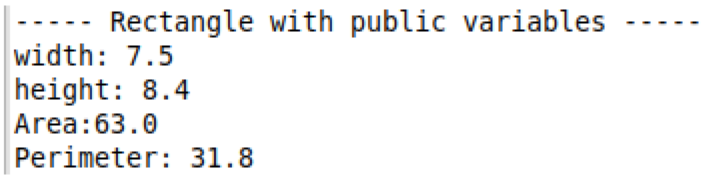
```java
        System.out.println("----- Rectangle with no keyword -----");
        RectangleWithNoKW rectn = new RectangleWithNoKW (2.4,7.5);
        rectn.h_ = 8.4;
        System.out.println("width: " + rectn.w_);
        System.out.println("height: " + rectn.h_);
        System.out.println("Area:" + rectn.area());
        System.out.println("Perimeter: " + rectn.perimeter());

        System.out.println("----- Rectangle with id -----");
        RectangleFull rectf1 = new RectangleFull (2.4, 7.5);
        System.out.println("width 1: " + rectf1.getWidth());
        System.out.println("height 1: " + rectf1.getHeight());
        System.out.println("Area 1:" + rectf1.area());
        System.out.println("Perimeter 1: " + rectf1.perimeter());
        System.out.println("Rect1: cnt=" + RectangleFull.getCnt() + ", uid=" + rectf1.getUid());
        RectangleFull rectf2 = new RectangleFull (3.2, 8);
        System.out.println("Rect2: cnt=" + RectangleFull.getCnt() + ", uid=" + rectf2.getUid());
        System.out.println("Rect1: cnt=" + RectangleFull.getCnt() + ", uid=" + rectf1.getUid());
    }
}
```

RectanglePublic

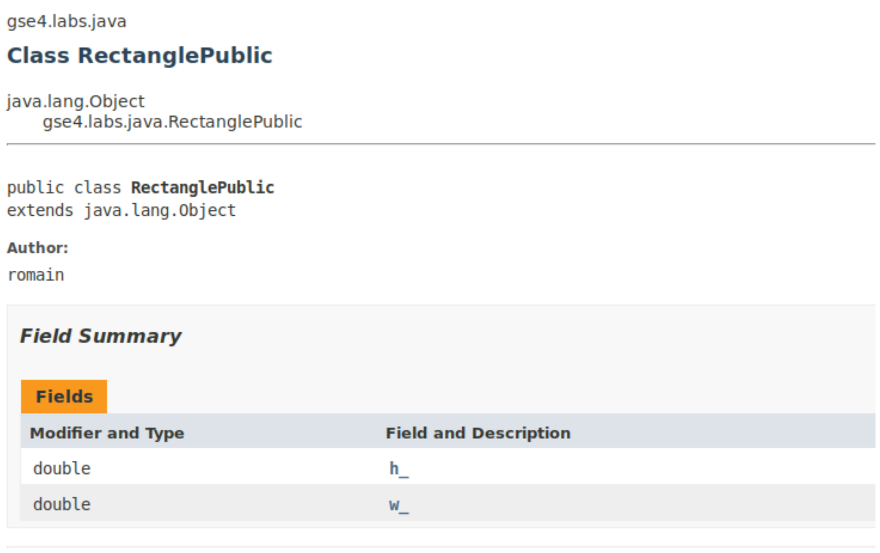
En mettant les variables de classe en public, on les exposent à toute modification par extérieure de la classe.



```
/**
package gse4.labs.java;

/**
 * @author romain
 *
 */
public class RectanglePublic {
    public double h_;
    public double w_;

    /**
     *
     * @param h set the height of rectangle
     * @param w set the width of rectangle
     */
    public RectanglePublic (double h, double w) {
        h_ = h;
        w_ = w;
    }

    /**
     *
     * @return area of rectangle
     */
    public double area () {
        return h_*w_;
    }
    /**
     *
     * @return perimeter of rectangle
     */
    public double perimeter () {
        return 2*h_+2*w_;
    }
}
```



```
----- Rectangle with public variables -----
width: 7.5
height: 8.4
Area:63.0
Perimeter: 31.8
```



gse4.labs.java

Class RectanglePublic

java.lang.Object
    gse4.labs.java.RectanglePublic

```
public class RectanglePublic
extends java.lang.Object
```

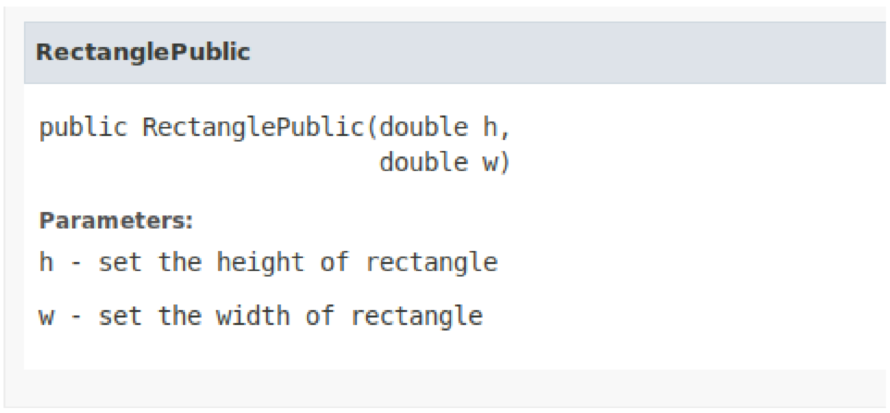
Author:

romain

Field Summary

Fields

Modifier and Type	Field and Description
double	h_
double	w_



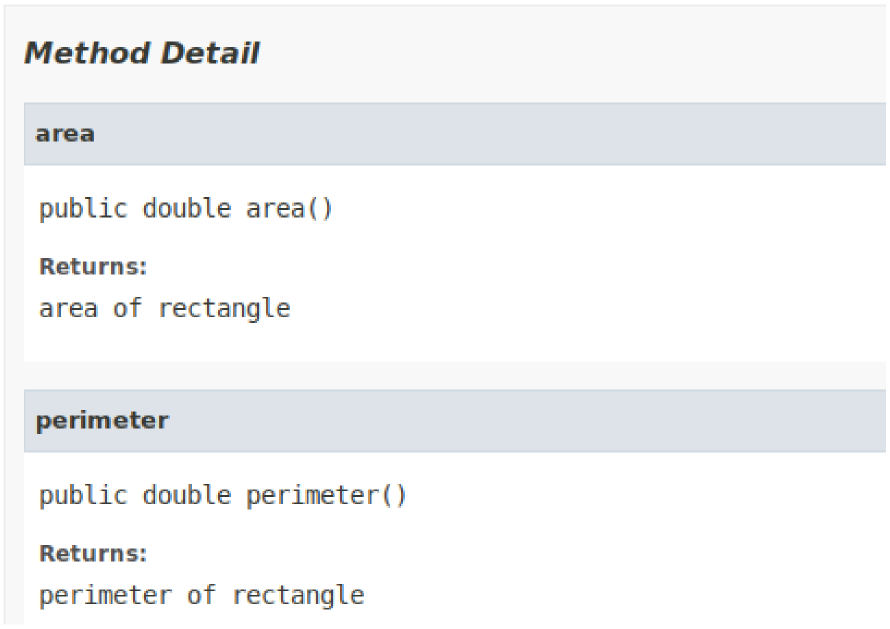
RectanglePublic

```
public RectanglePublic(double h,
                       double w)
```

Parameters:

h - set the height of rectangle

w - set the width of rectangle



Method Detail

area

```
public double area()
```

Returns:

area of rectangle

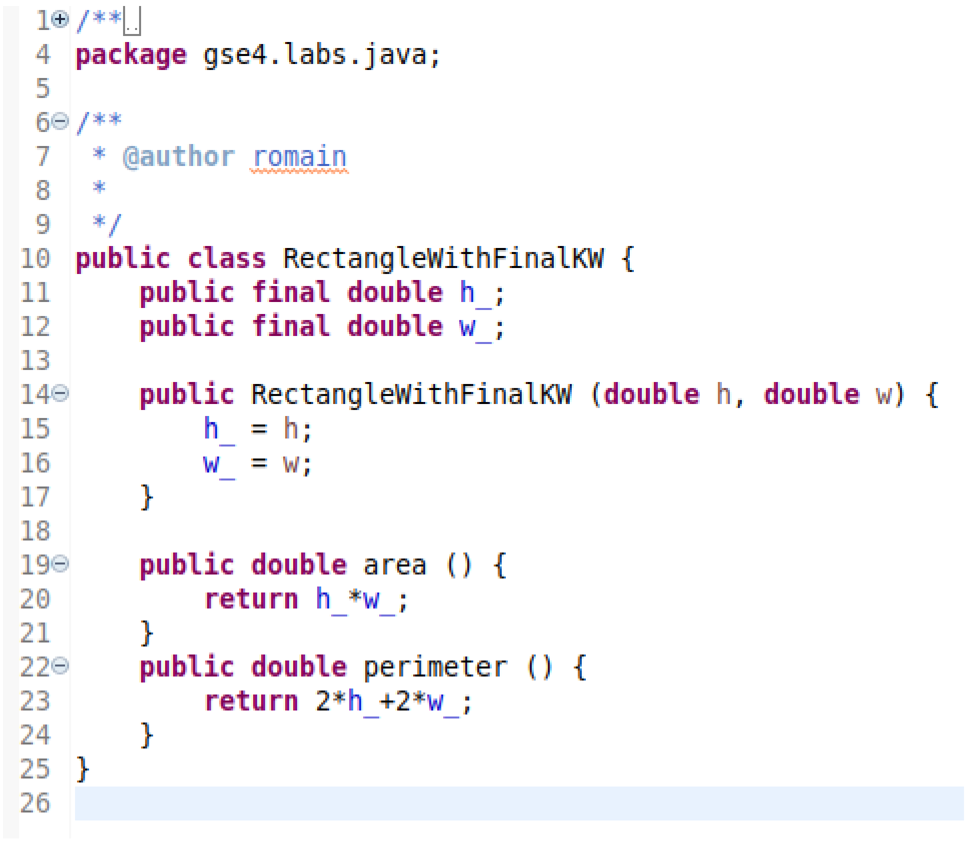
perimeter

```
public double perimeter()
```

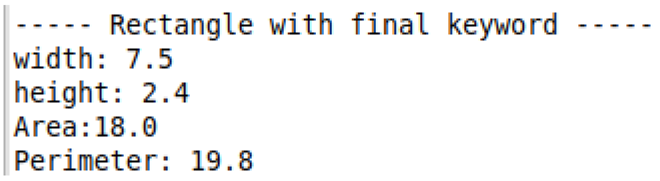
Returns:

perimeter of rectangle

RectangleWithFinalKW



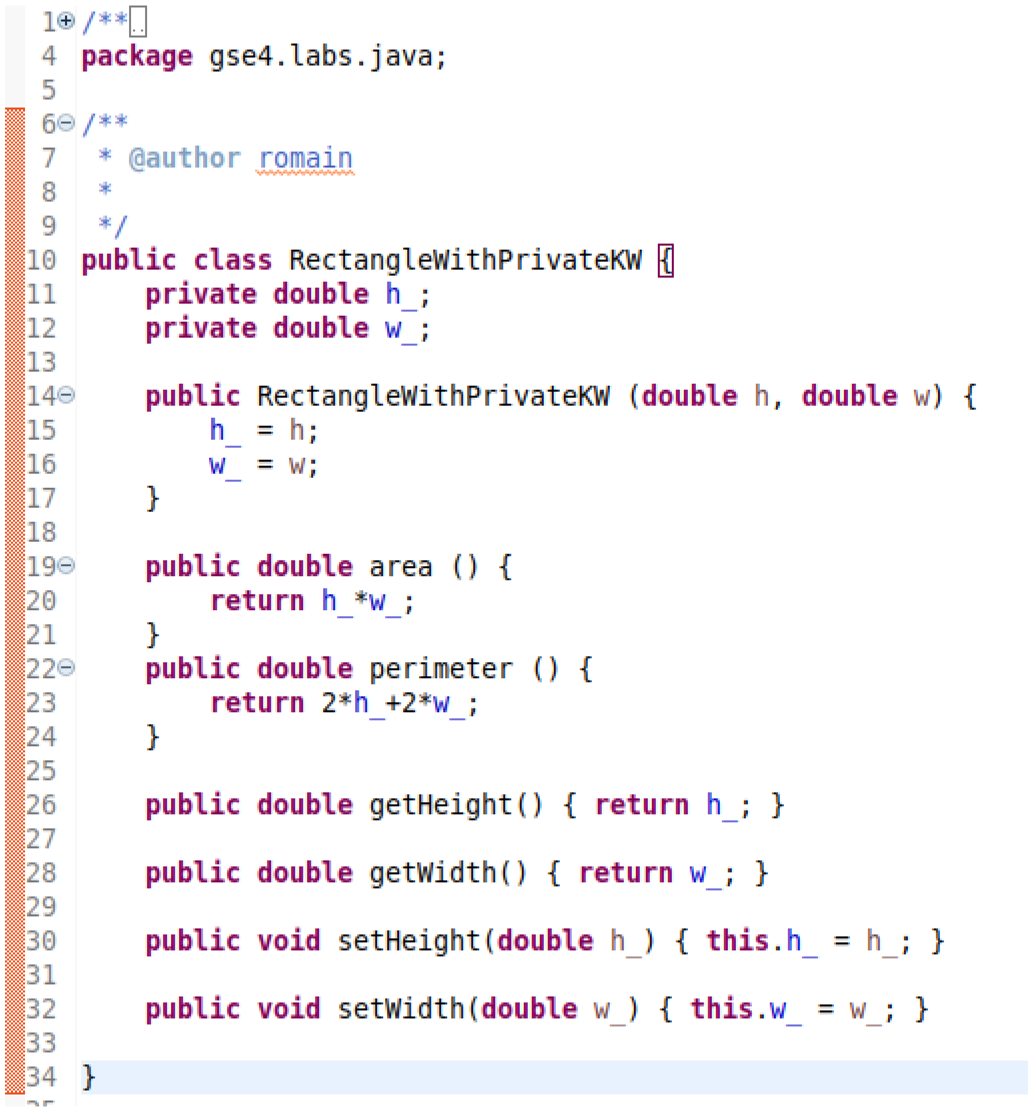
```java
/**
package gse4.labs.java;

/**
 * @author romain
 *
 */
public class RectangleWithFinalKW {
    public final double h_;
    public final double w_;

    public RectangleWithFinalKW (double h, double w) {
        h_ = h;
        w_ = w;
    }

    public double area () {
        return h_*w_;
    }
    public double perimeter () {
        return 2*h_+2*w_;
    }
}

```



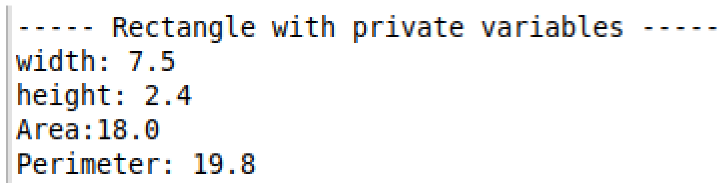
```
----- Rectangle with final keyword -----
width: 7.5
height: 2.4
Area:18.0
Perimeter: 19.8
```

En ajoutant le mot clé *final*, on remarque que les valeurs *w_* et *h_* ne peuvent être modifiées qu’une seule fois. Il s’agit bien là du principe de fonctionnement du mot clé *final*. Après la première modification, les variables sont bloquées et sont donc considérées comme constantes.

RectangleWithPrivateKW



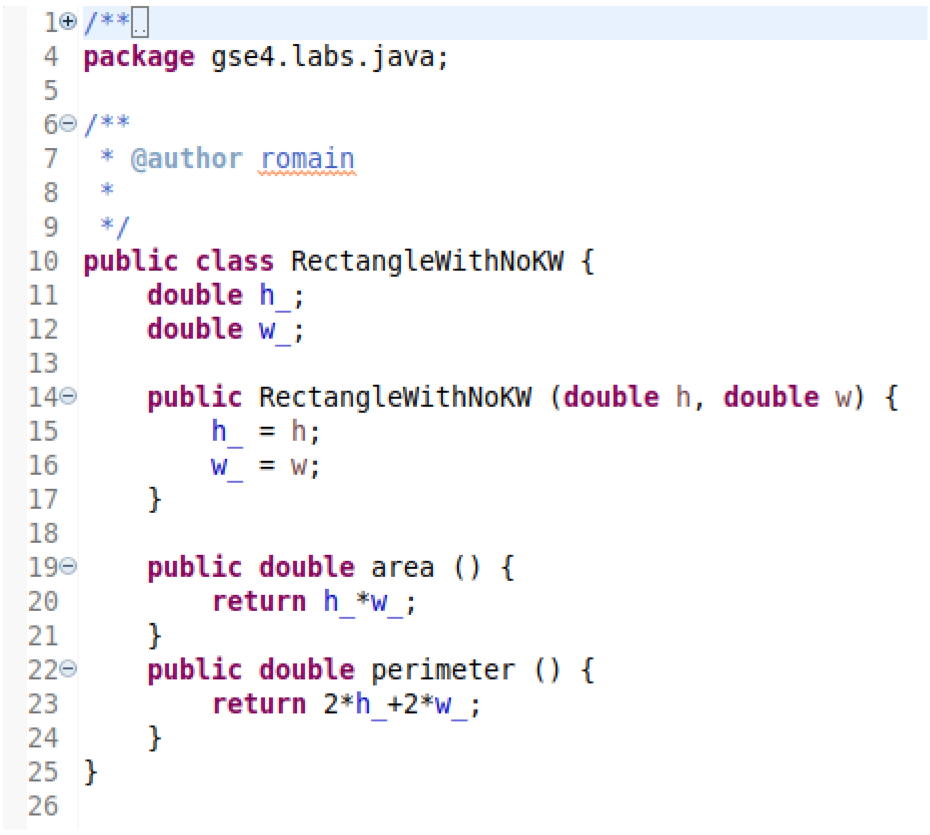
```java
/**
package gse4.labs.java;

/**
 * @author romain
 *
 */
public class RectangleWithPrivateKW {
    private double h_;
    private double w_;

    public RectangleWithPrivateKW (double h, double w) {
        h_ = h;
        w_ = w;
    }

    public double area () {
        return h_*w_;
    }
    public double perimeter () {
        return 2*h_+2*w_;
    }

    public double getHeight() { return h_; }

    public double getWidth() { return w_; }

    public void setHeight(double h_) { this.h_ = h_; }

    public void setWidth(double w_) { this.w_ = w_; }

}
```



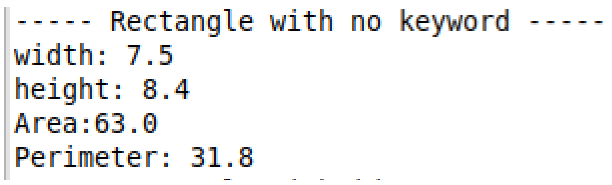
```
----- Rectangle with private variables -----
width: 7.5
height: 2.4
Area:18.0
Perimeter: 19.8
```

En mettant les variables en privé, on les rend inaccessibles depuis l'extérieur de la classe. Il faut donc créer des accesseurs et modificateurs pour pouvoir manipuler ces variables.

RectangleWithNoKW



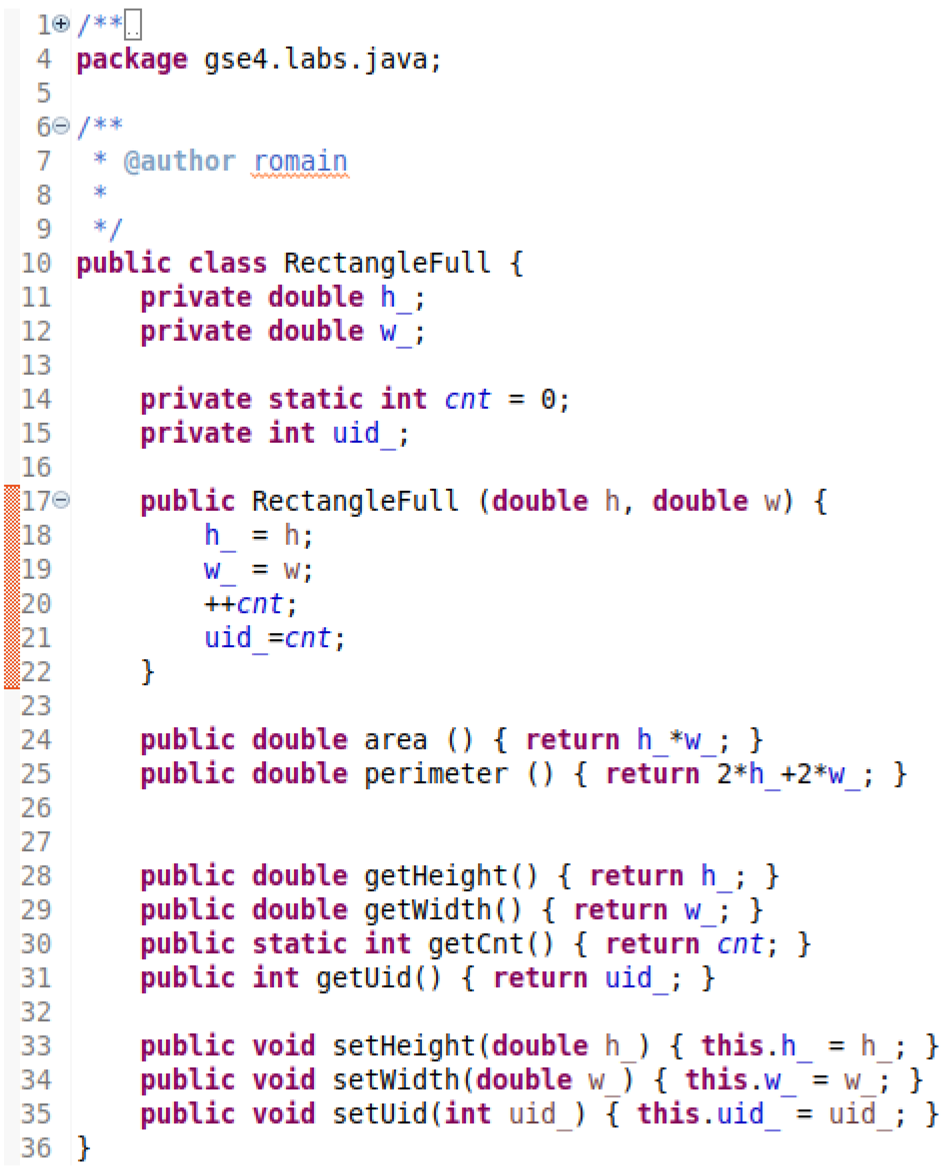
```java
/**
package gse4.labs.java;

/**
 * @author romain
 *
 */
public class RectangleWithNoKW {
    double h_;
    double w_;

    public RectangleWithNoKW (double h, double w) {
        h_ = h;
        w_ = w;
    }

    public double area () {
        return h_*w_;
    }
    public double perimeter () {
        return 2*h_+2*w_;
    }
}

```



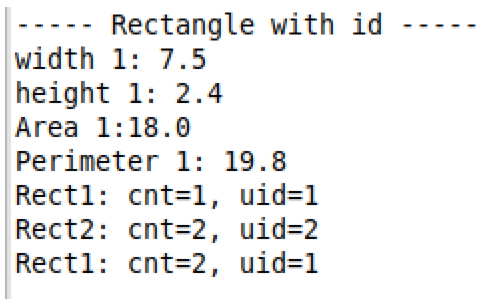
```
----- Rectangle with no keyword -----
width: 7.5
height: 8.4
Area:63.0
Perimeter: 31.8
```

Si aucun mot clé n’est spécifié, le scope par défaut est public. Le programme se comporte donc comme si les variables étaient précédées du mot clé *public*.

RectangleFull

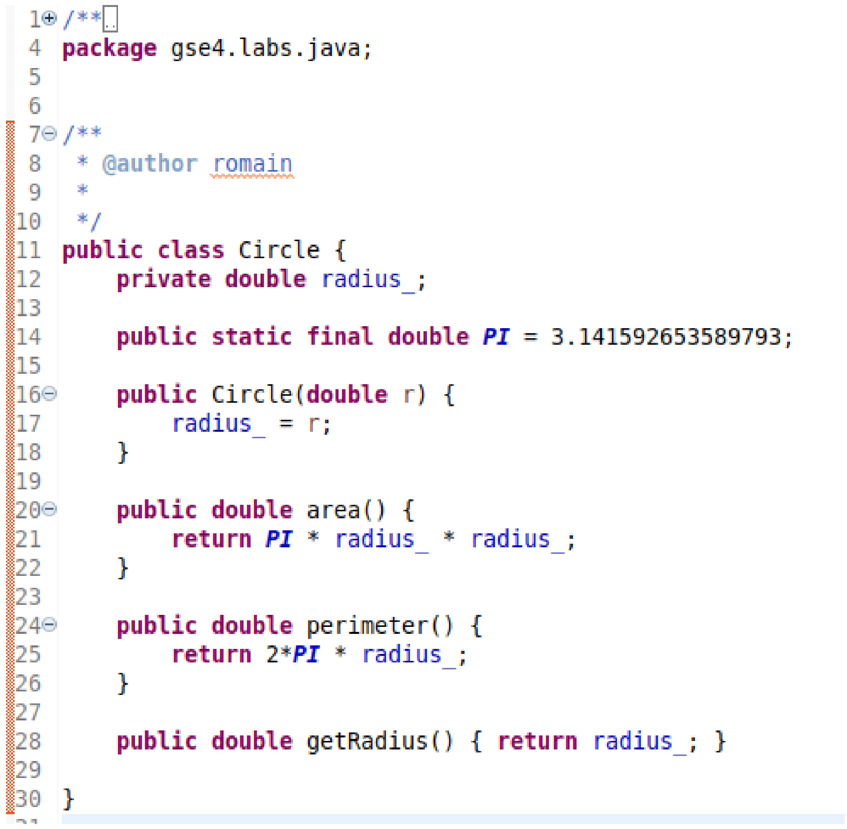


```java
/**
package gse4.labs.java;

/**
 * @author romain
 *
 */
public class RectangleFull {
    private double h_;
    private double w_;

    private static int cnt = 0;
    private int uid_;

    public RectangleFull (double h, double w) {
        h_ = h;
        w_ = w;
        ++cnt;
        uid_=cnt;
    }

    public double area () { return h_*w_; }
    public double perimeter () { return 2*h_+2*w_; }


    public double getHeight() { return h_; }
    public double getWidth() { return w_; }
    public static int getCnt() { return cnt; }
    public int getUid() { return uid_; }

    public void setHeight(double h_) { this.h_ = h_; }
    public void setWidth(double w_) { this.w_ = w_; }
    public void setUid(int uid_) { this.uid_ = uid_; }
}
```



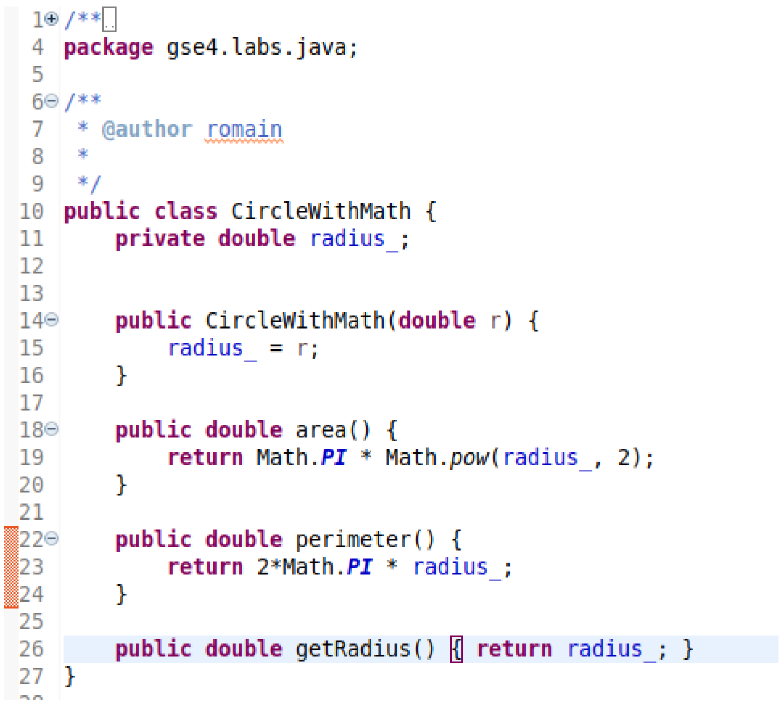
```
----- Rectangle with id -----
width 1: 7.5
height 1: 2.4
Area 1:18.0
Perimeter 1: 19.8
Rect1: cnt=1, uid=1
Rect2: cnt=2, uid=2
Rect1: cnt=2, uid=1
```

Circle



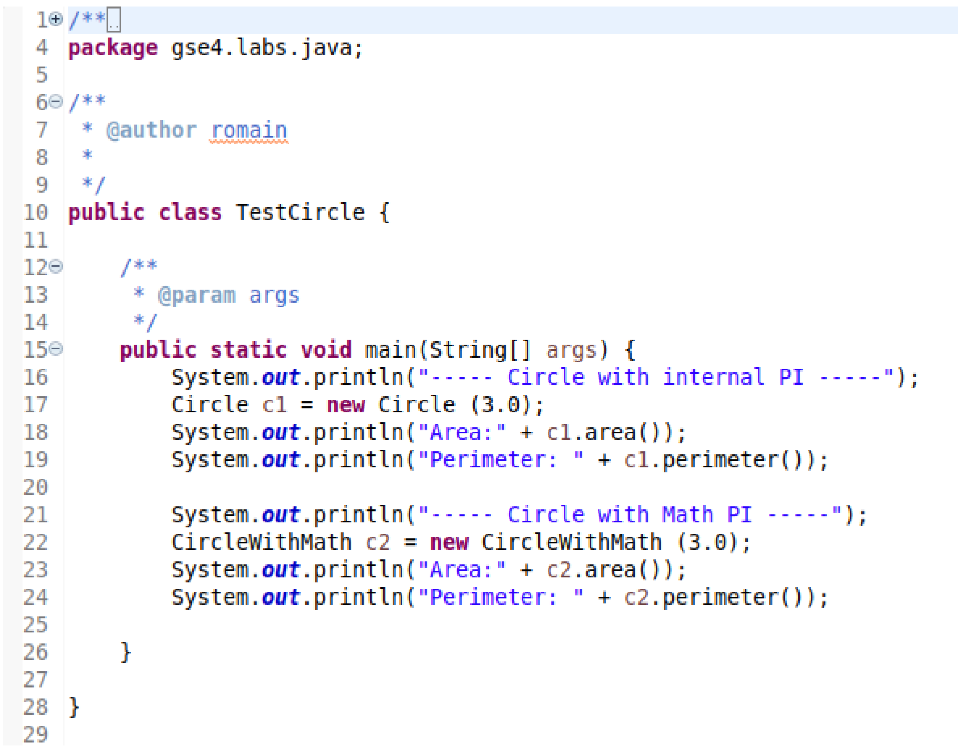
```java
/**
package gse4.labs.java;


/**
 * @author romain
 *
 */
public class Circle {
    private double radius_;

    public static final double PI = 3.141592653589793;

    public Circle(double r) {
        radius_ = r;
    }

    public double area() {
        return PI * radius_ * radius_;
    }

    public double perimeter() {
        return 2*PI * radius_;
    }

    public double getRadius() { return radius_; }

}
```

CircleWithMath

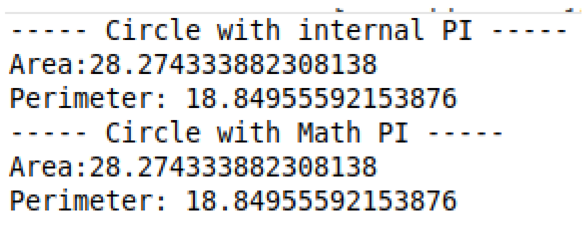


```java
/**
package gse4.labs.java;

/**
 * @author romain
 *
 */
public class CircleWithMath {
    private double radius_;


    public CircleWithMath(double r) {
        radius_ = r;
    }

    public double area() {
        return Math.PI * Math.pow(radius_, 2);
    }

    public double perimeter() {
        return 2*Math.PI * radius_;
    }

    public double getRadius() { return radius_; }
}
```

TestCircle



```java
/**
package gse4.labs.java;

/**
 * @author romain
 *
 */
public class TestCircle {

    /**
     * @param args
     */
    public static void main(String[] args) {
        System.out.println("----- Circle with internal PI -----");
        Circle c1 = new Circle (3.0);
        System.out.println("Area:" + c1.area());
        System.out.println("Perimeter: " + c1.perimeter());

        System.out.println("----- Circle with Math PI -----");
        CircleWithMath c2 = new CircleWithMath (3.0);
        System.out.println("Area:" + c2.area());
        System.out.println("Perimeter: " + c2.perimeter());

    }

}
```



```
----- Circle with internal PI -----
Area:28.274333882308138
Perimeter: 18.84955592153876
----- Circle with Math PI -----
Area:28.274333882308138
Perimeter: 18.84955592153876
```

Ici on remarque que dans le cas de l’utilisation de PI et *pow* de la librairie *Math*, nous n’avons pas besoin d’inclure celle-ci dans le programme. En effet, *Math* est une librairie standard de java et est automatiquement rajouté par Eclipse.