

# MODIFICADORES DE ACCESO

## MÓDULO 6

### CAPÍTULO 5

"LA ENCAPSULACIÓN ES LA SEPARACIÓN DE LA FUNCIONALIDAD  
DE UNA CLASE DEL ESTADO INTERNO."

"ES SEPARAR QUÉ HACE UNA CLASE DEL CÓMO LO HACE"

# Encapsulación

```
public class RightTriangle_v1 {  
    private double area;  
    private double base;  
    private double height;  
  
    public RightTriangle(double base,  
                        double height) {  
        if (base <= 0 || height <= 0) {  
            this.base = 0;  
            this.height = 0;  
        } else {  
            this.base = base;  
            this.height = height;  
            this.area = base * height / 2;  
        }  
    }  
  
    public double calculateArea() {  
        return this.area;  
    }  
}
```

```
public class RightTriangle_v2 {  
    private double base;  
    private double height;  
  
    public RightTriangle(double base,  
                        double height) {  
        if (base <= 0 || height <= 0) {  
            this.base = 0;  
            this.height = 0;  
        } else {  
            this.base = base;  
            this.height = height;  
        }  
    }  
  
    public double calculateArea() {  
        return this.base * this.height / 2;  
    }  
}
```

# Encapsulación

```
public class EncapsulationDemo {  
  
    public static void main(String[] args) {  
  
        RightTriangle_v1 triangle1 = new RightTriangle_v1(10, 5);  
        RightTriangle_v2 triangle2 = new RightTriangle_v2(10, 5);  
  
        double out1 = triangle1.calculateArea();  
        double out2 = triangle2.calculateArea();  
  
        System.out.println("Class V1: " + out1);  
        System.out.println("Class V2: " + out2);  
    }  
}
```

**El método calculateArea() de ambas clases logra el mismo resultado. En la práctica, podemos utilizar cualquiera indistintamente.**

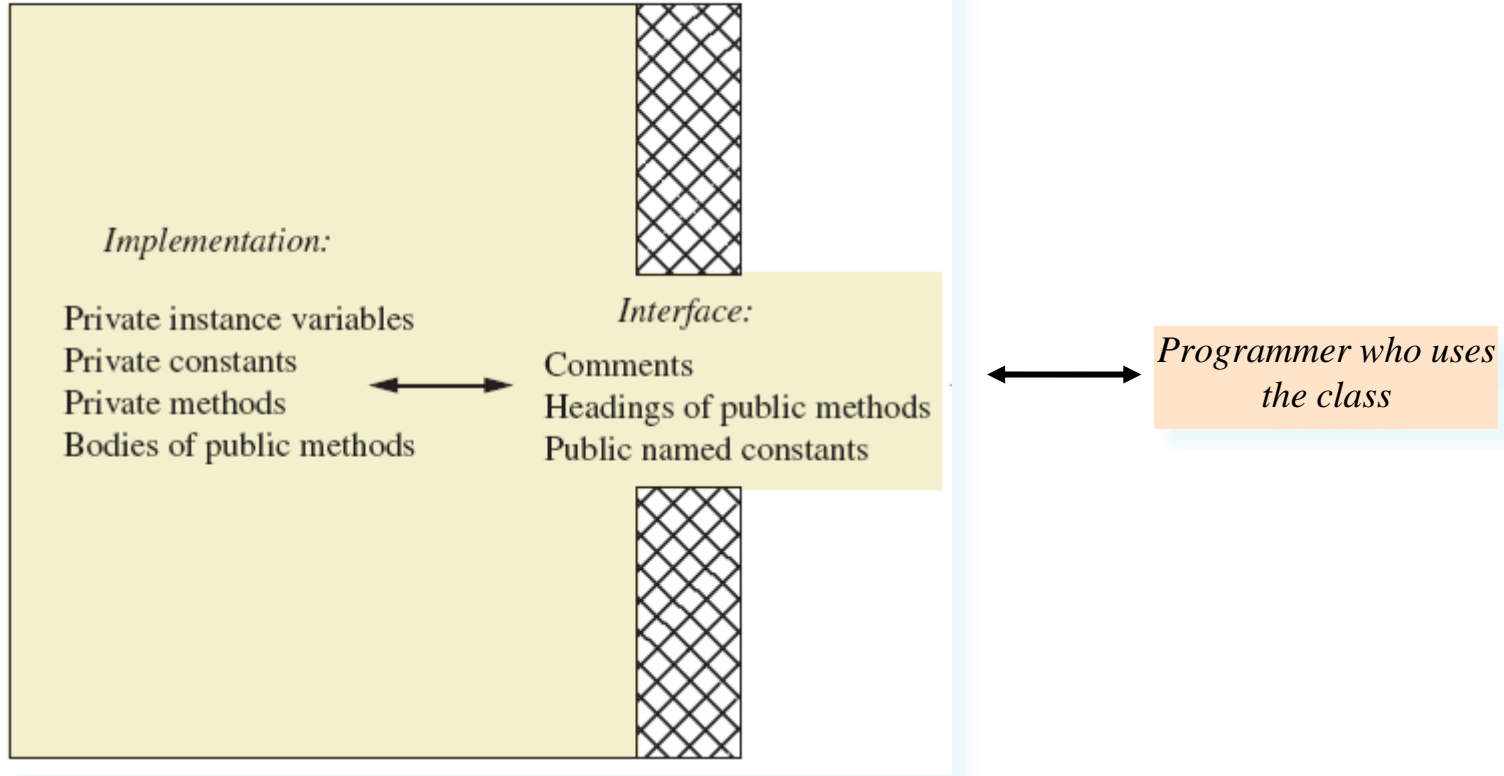


# ¿CÓMO SE LOGRA LA ENCAPSULACIÓN?

1. MODIFICADORES DE ACCESO ADECUADOS

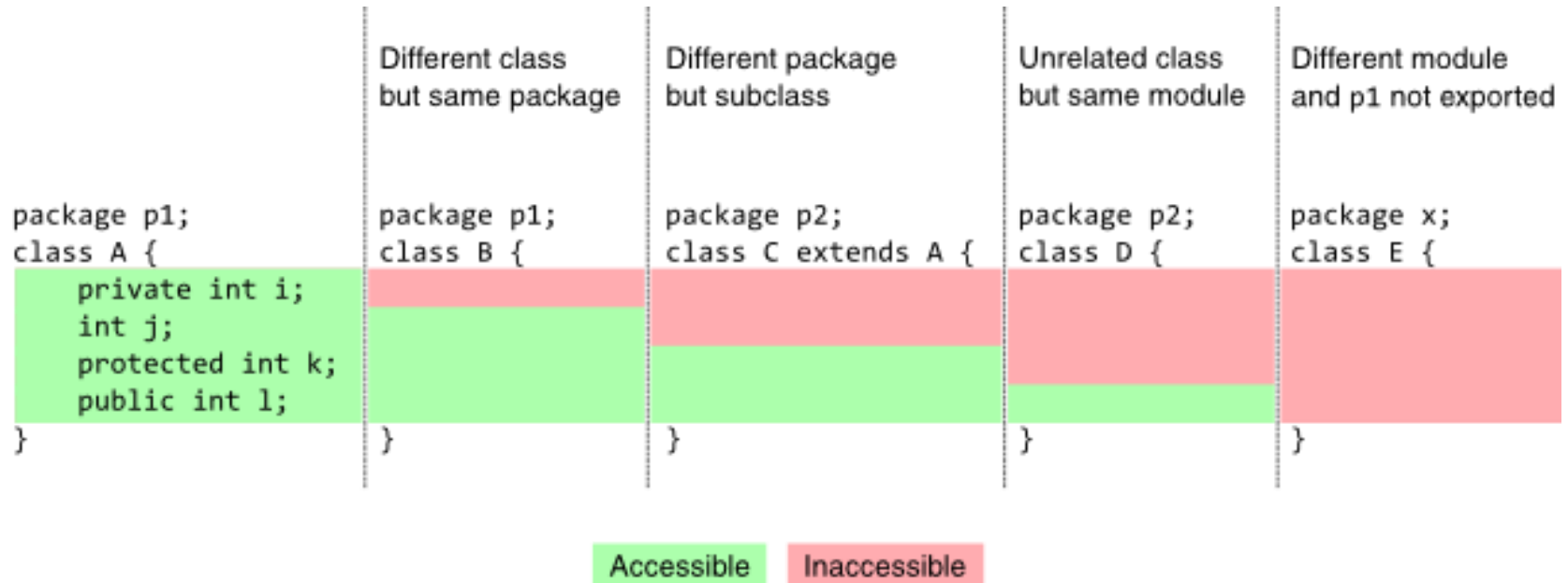
2. MÉTODOS PÚBLICOS PARA INTERACTUAR CON LA CLASE.

## *Class Definition*



# Modificadores de Acceso

Un modificador de acceso indica el alcance de una variable, método o clase. Hay cuatro distintos modificadores de acceso, donde los más comunes son **private** y **public**.



# Getter

El método getter, también conocido como accessor, es un tipo de método que nos permite leer el contenido de una variable de instancia en una clase.

```
public class RightTriangle_v1 {  
    //...  
    //rest of the variables and methods  
    //...  
    private double height;  
  
    public double getHeight() {  
        return this.height;  
    }  
}
```



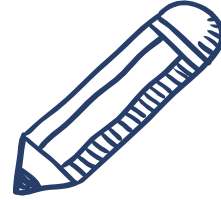
Como la variable **height** es privada, al intentar leerla desde otra clase sería imposible, por lo que hay que utilizar un método público para leer el contenido de la clase.



# Setter

El método setter, también conocido como mutator, nos permite modificar el contenido de una variable de instancia en una clase.

```
public class RightTriangle_v1 {  
    //...  
    //rest of the variables and methods  
    //...  
    private double height;  
    private double area;  
  
    public void setHeight(double height) {  
        if (height > 0) {  
            this.height = height;  
            this.area = this.base * this.height / 2;  
        }  
    }  
}
```



Si modificamos la variable `height`, debemos afectar también la variable `area` para no dejar la información del triángulo inconsistente!

LA MEJOR PRÁCTICA ES DECLARAR TODAS LAS  
VARIABLES DE INSTANCIA COMO PRIVADAS, Y  
DEFINIR MÉTODOS GETTER Y SETTER PARA  
UTILIZARLAS.

```
public class RightTriangle_v1 {

    private double area;
    private double base;
    private double height;

    public RightTriangle_v1(double base,
                           double height) {
        if (base <= 0 || height <= 0) {
            this.setBase(0);
            this.setHeight(0);
        } else {
            this.setBase(base);
            this.setHeight(height);
        }
    }

    private void updateArea() {
        this.area = this.getBase() * this.getHeight() / 2;
    }

    public double getBase() {
        return this.base;
    }
}
```

```
    public void setBase(double base) {
        if (base >= 0) {
            this.base = base;
            this.updateArea();
        }
    }

    public double getHeight() {
        return this.height;
    }

    public void setHeight(double height) {
        if (height >= 0) {
            this.height = height;
            this.updateArea();
        }
    }

    public double getArea() {
        return this.area;
    }
}
```

```
private void updateArea() {  
    this.area = this.getBase() * this.getHeight() / 2;  
}
```

## ¿Por qué el método updateArea( ) es privado?



Cuando leamos la variable de instancia `area`, esperamos que su contenido esté actualizado. La clase debe garantizar este comportamiento, llamando el método `updateArea( )` siempre que alguna de las dimensiones del objeto sean modificados.

## ¿Por qué no definimos un método `setArea()`?

Si el contenido de la variable `area` es modificado, la clase pierde congruencia interna. Debemos garantizar que la siguiente ecuación se cumpla en todo momento:



$$Area = \frac{base \times altura}{2}$$

La actualización de esta variable sólo se debe modificar mediante un cambio en:

- × `base`
- × `altura`