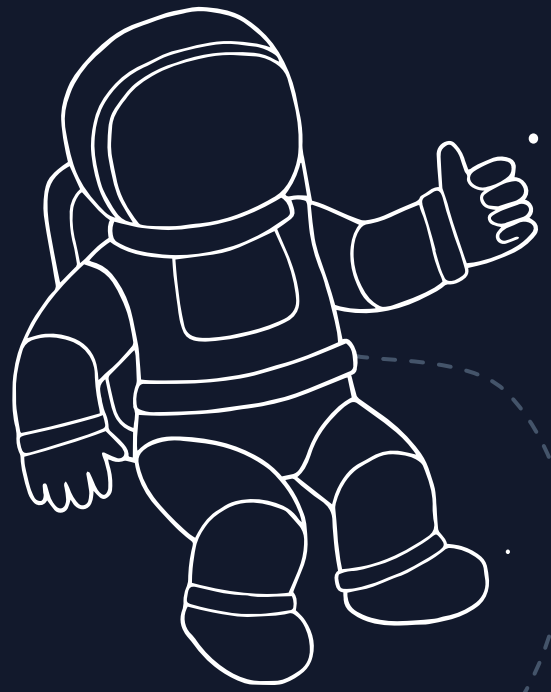
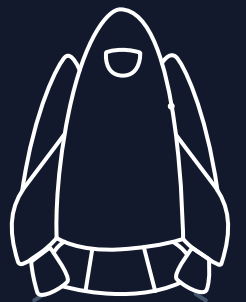


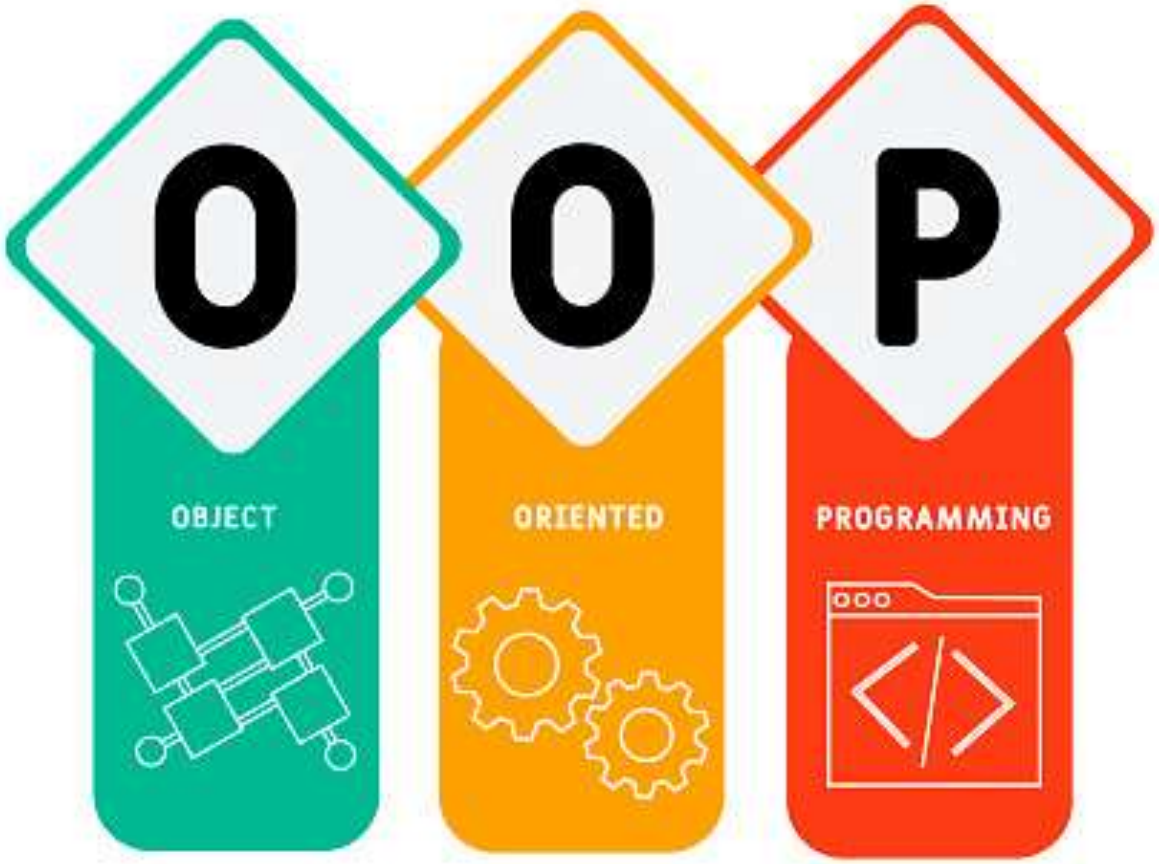
Programa académico CAMPUS



MODULO JAVA
Sesión 5
Encapsulamiento

Trainer Carlos H. Rueda C.

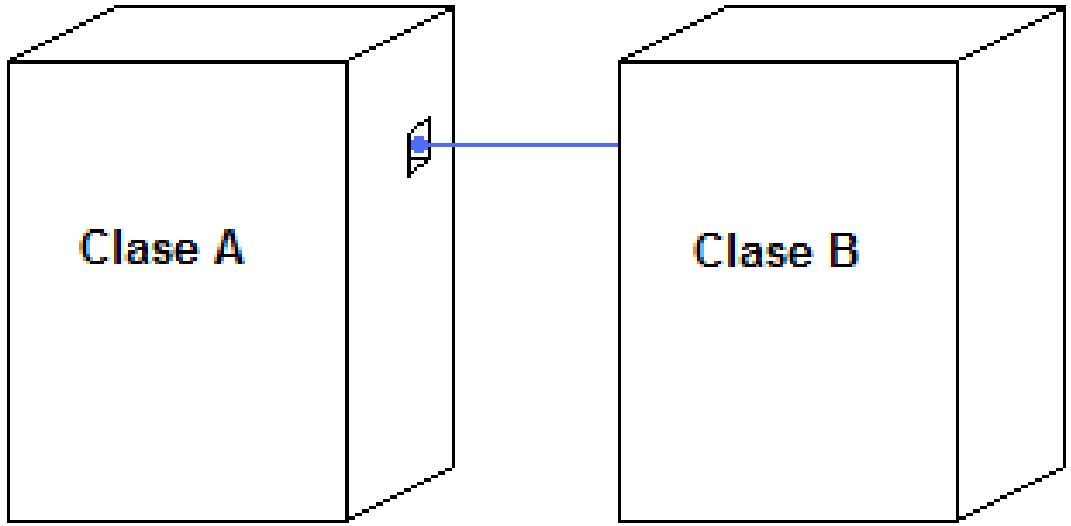

Java™

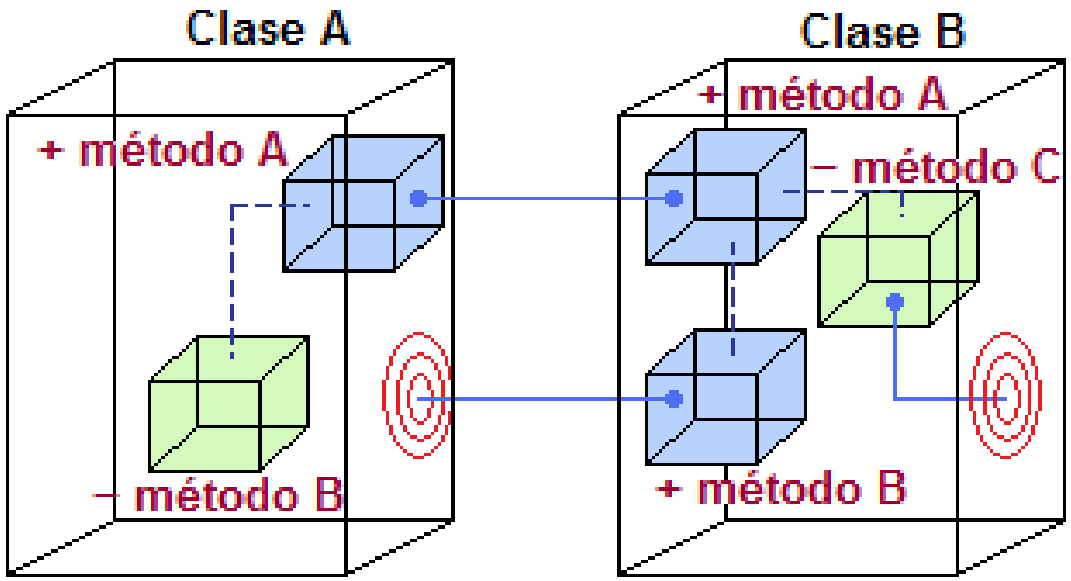

O
O
P
OBJECT
ORIENTED
PROGRAMMING

¿Qué es encapsulamiento?

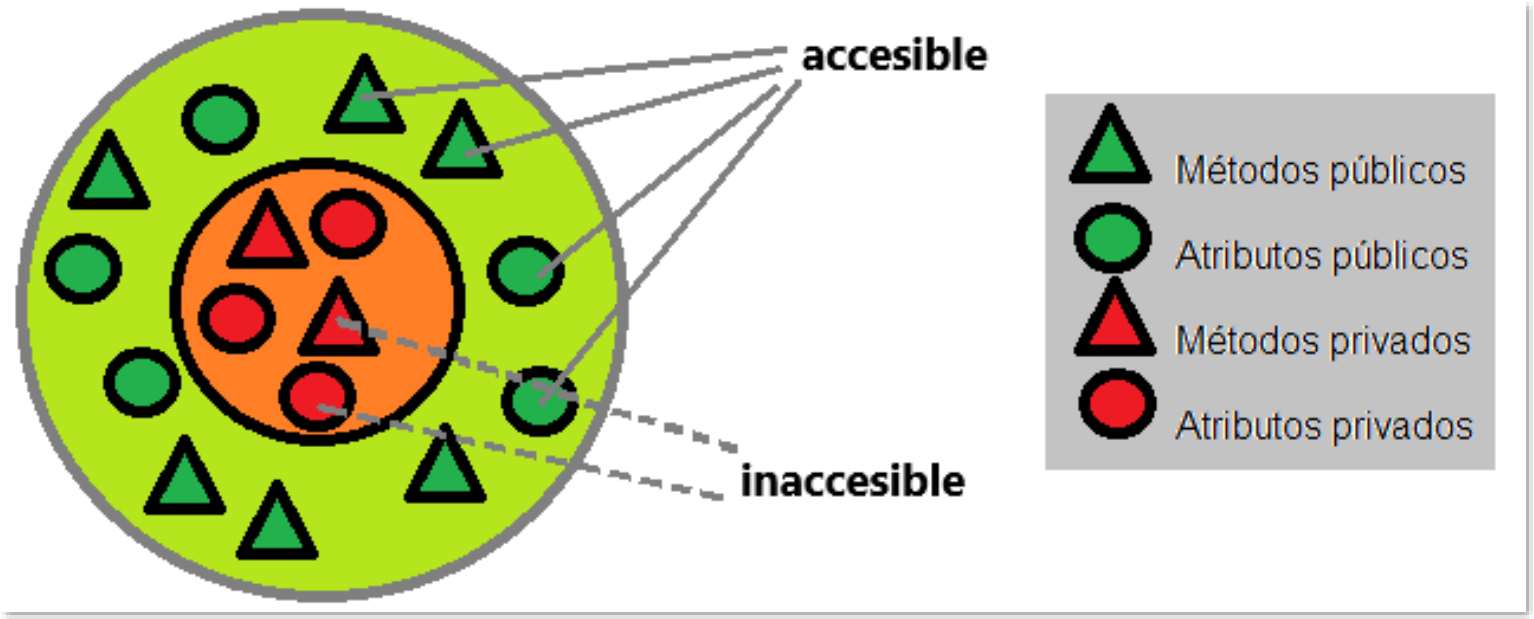
El concepto de encapsulamiento se refiere a la restricción del acceso directo a algunos de los componentes de un objeto, especialmente los atributos, y la prevención de modificaciones no autorizadas. En otras palabras, el encapsulamiento consiste en ocultar los detalles internos de un objeto y exponer solo lo que es necesario. Esta práctica está íntimamente ligada con la noción de visibilidad y los modificadores de acceso.

¿Qué es encapsulamiento?





Modificadores de acceso

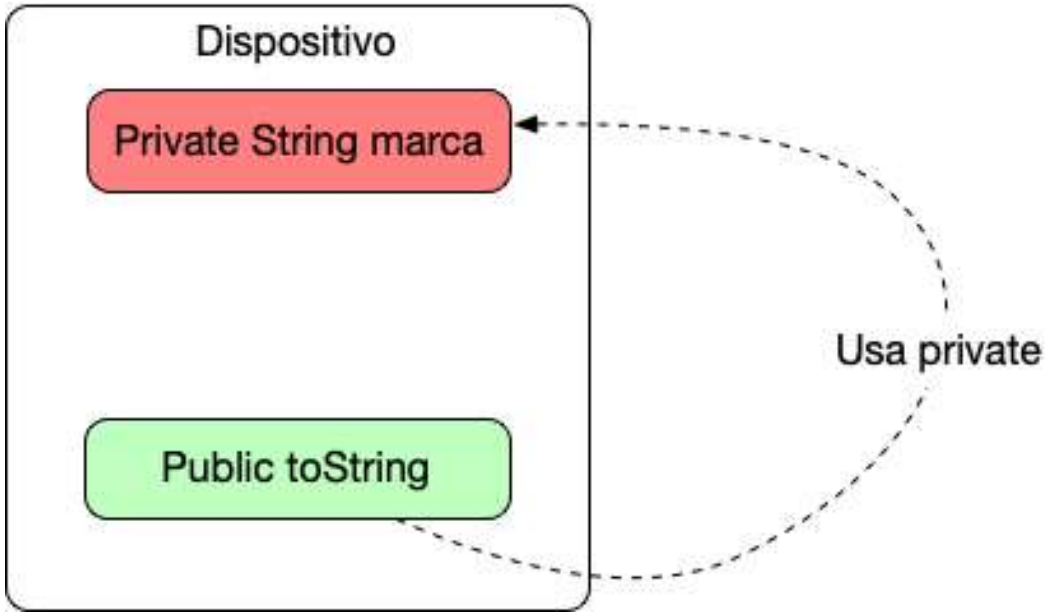


Modificadores de Acceso

- ✓Públicos *(public)*
- ✓Privados *(private)*
- ✓Protegidos *(protected)*
- ✓Defecto *(default)*

Modificador de acceso private

El modificador `private` en Java es el más restrictivo de todos. Por lo tanto, cualquier elemento de una clase que sea declarado como `private` solo puede ser accedido por esa misma clase, y no por ninguna otra clase, sin importar la relación que tengan entre sí. Esto significa que, por ejemplo, si un atributo es declarado como `private`, solo los métodos y constructores de esa misma clase podrán acceder a él, y ninguna otra clase podrá tener acceso a dicho atributo.

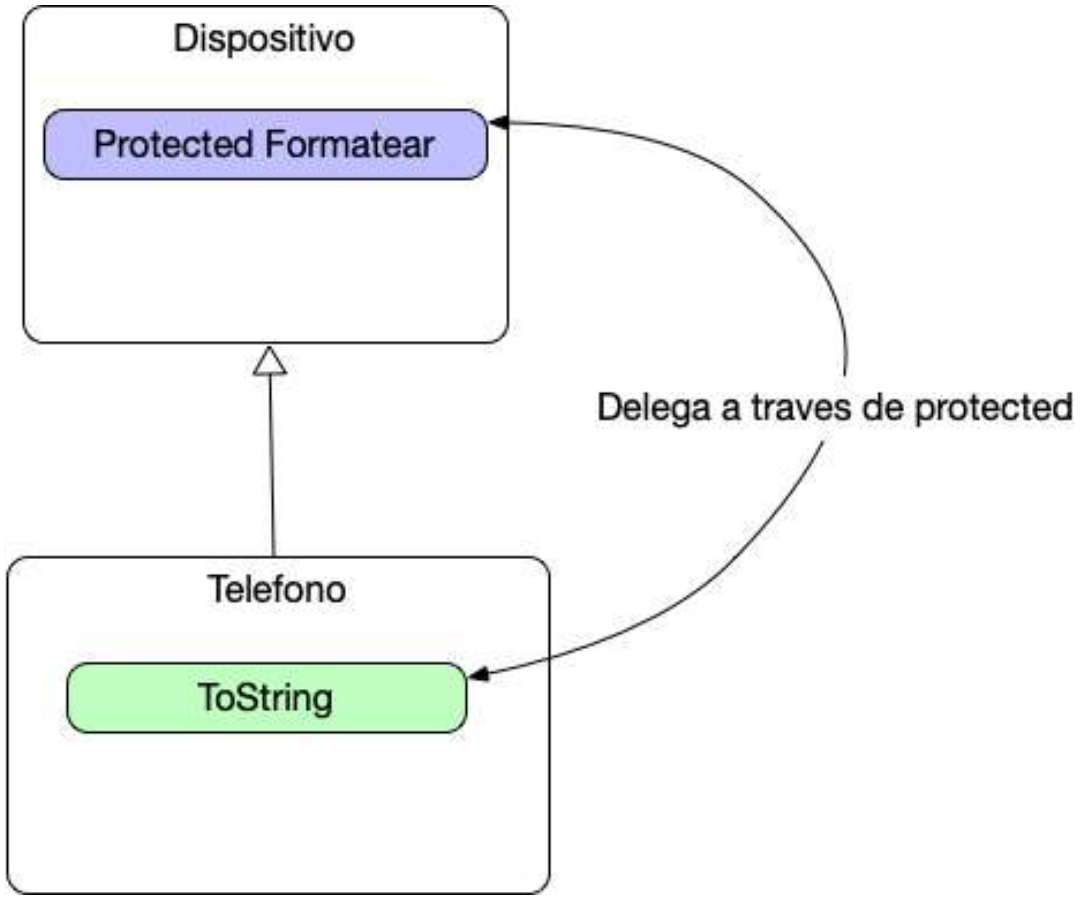


Modificador de acceso por defecto (default)

En Java, existe la opción de no especificar un modificador de acceso, lo que resulta en un nivel de acceso conocido como "acceso por defecto" o "`default`". Esto significa que los elementos sin un modificador de acceso explícito pueden ser accedidos por la propia clase y por otras clases que se encuentren en el mismo paquete. Es importante destacar la importancia de declarar siempre un paquete para nuestras clases, ya que esto afecta el alcance de acceso de los componentes sin un modificador de acceso.

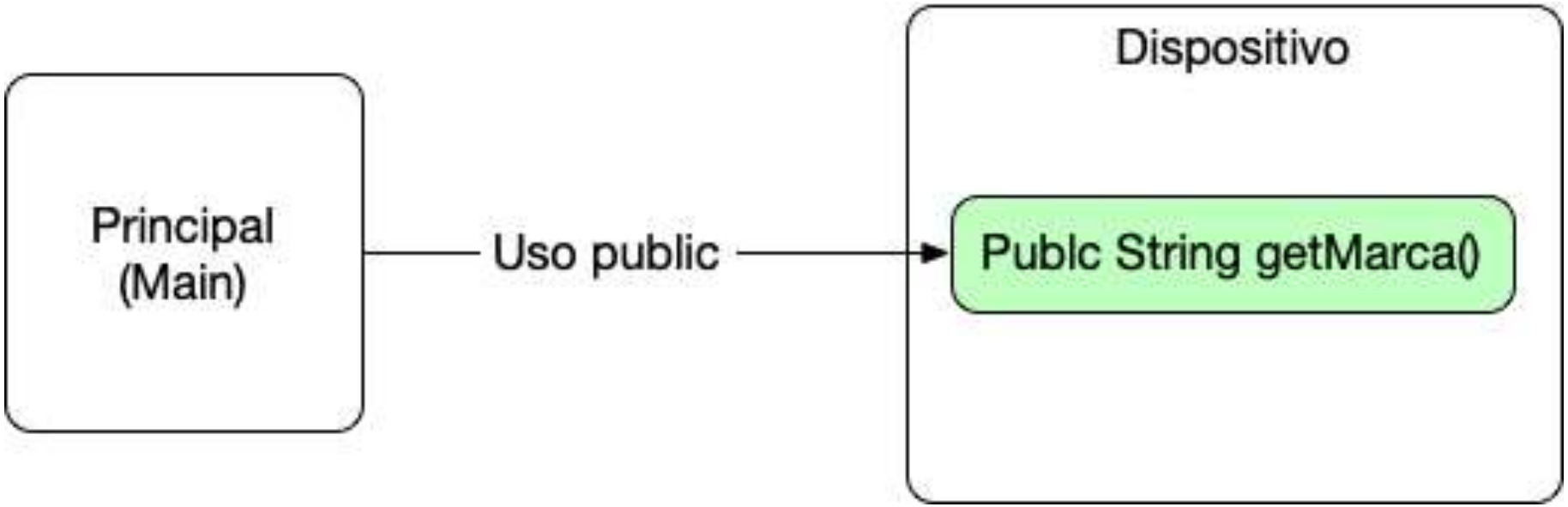
Modificador de acceso protected

El modificador de acceso "`protected`" permite acceder a los componentes con ese modificador desde la misma clase, clases en el mismo paquete y clases que hereden de ella, incluso si se encuentran en diferentes paquetes.



Modificador de acceso public

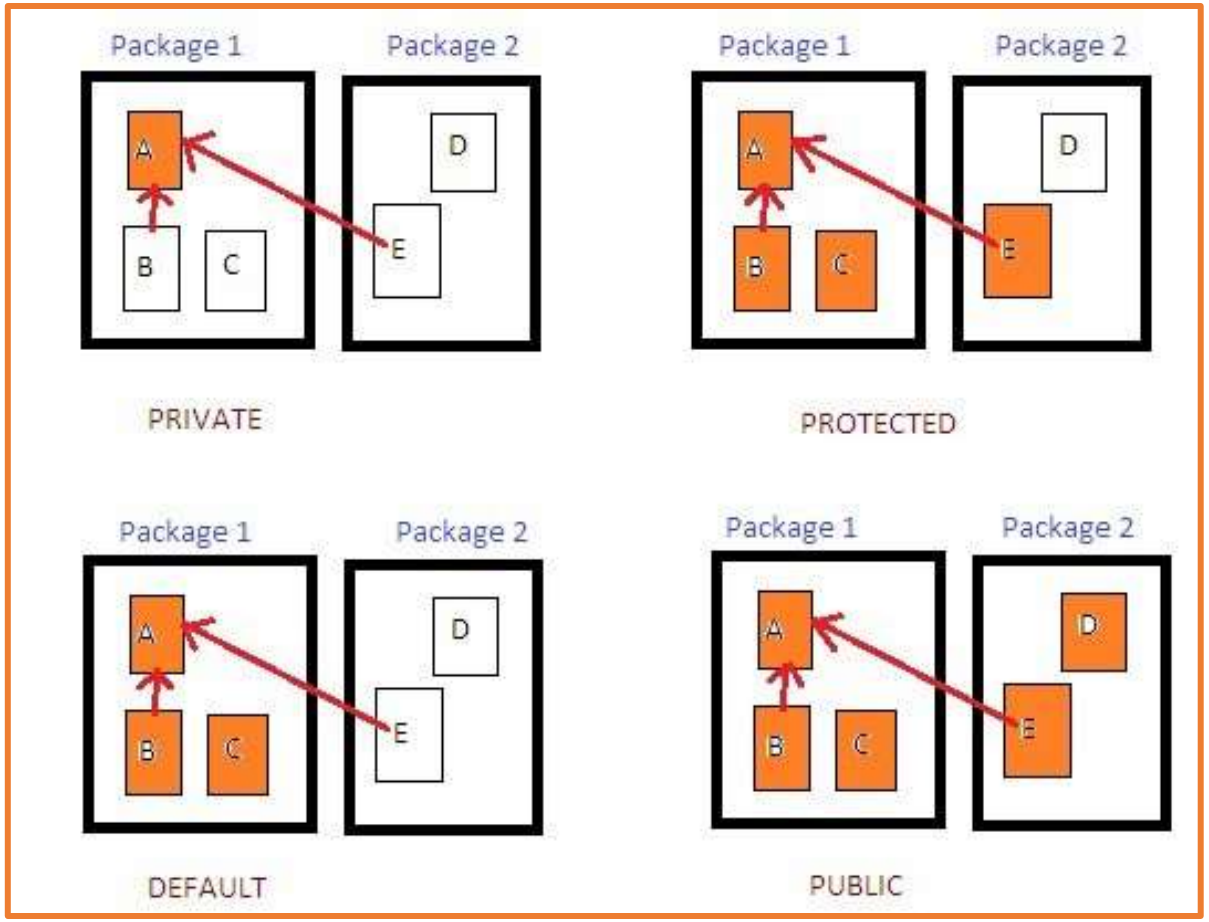
El modificador de acceso "`public`" es el más amplio y permite acceder a los componentes de una clase desde cualquier otra clase o instancia, sin importar el paquete en el que se encuentren. En contraste con el modificador "`private`", el modificador "`public`" permite un acceso más abierto y no impone restricciones en términos de paquetes o procedencia.



Modificadores

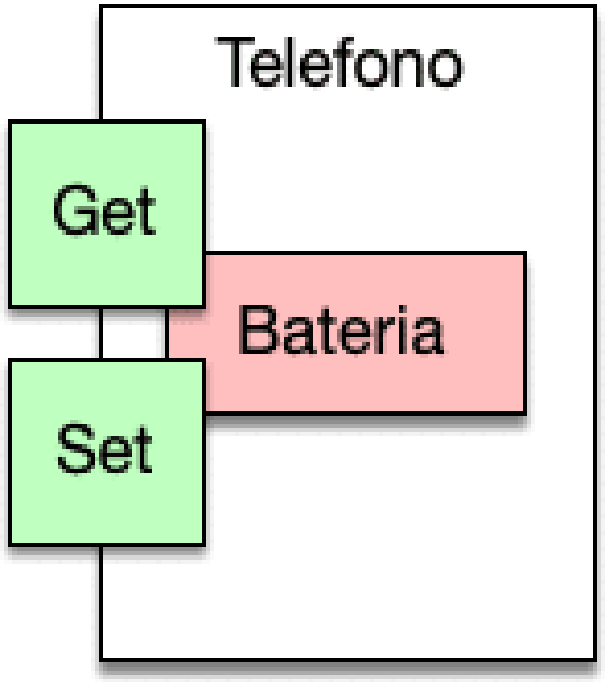
	Clase	Paquete	Subclase	Resto
public	+	+	+	+
protected	+	+	+	-
default	+	+	-	-
private	+	-	-	-

\+ VISIBLE
\- NO VISIBLE



Métodos get y set

Los métodos "`get`" y "`set`" son funciones utilizadas en las clases para obtener (`get`) o modificar (`set`) el valor de un atributo. Estos métodos siguen una convención de nomenclatura en la que el nombre comienza con "`get`" o "`set`", seguido del nombre del atributo. Su modificador de acceso suele ser "`public`", ya que lo que se desea es permitir el acceso y la modificación del atributo desde fuera de la clase. Por ejemplo, "`getNombre`" o "`setNombre`".



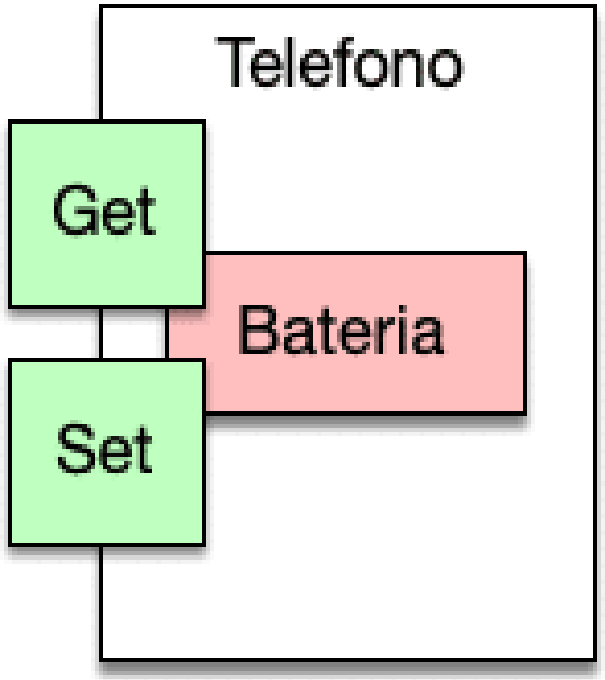
Métodos get y set

```java
public class Persona{  no usages
    private String nombre;  3 usages

    public Persona(String n){  no usages
        nombre = n;
    }

    public String getNombre(){  no usages
        return nombre;
    }

    public void setNombre(String nombre) {  no usages
        this.nombre = nombre;
    }
}
```



Modificadores

```java
public class Dispositivo {  2 usages
    private String marca;  3 usages
    private double precio;  3 usages
    String serie;  3 usages

    public Dispositivo(String marca, double precio) {  1 usage
        this.marca = marca;
        this.precio = precio;
        serie = "";
    }

    public Dispositivo(String marca, double precio, String serie) {  no usages
        this.marca = marca;
        this.precio = precio;
        this.serie = serie;
    }

    public String imprimir() {  no usages
        return "Dispositivo [marca=" + marca + ", precio=" + precio + "]";
    }
}
```

Modificadores

```
class pruebaDispositivo {
    public static void main(String[] args) {
        Dispositivo d= new Dispositivo( marca: "samsung", precio: 200);

        System.out.println(d.getMarca());
        System.out.println(d.getPrecio());

        d.serie = "Galaxy";
        System.out.println(d.getSerie());
    }
}
```

Ejemplo

Cree una clase llamada **Rectangulo** con los atributos longitud y anchura, cada uno con un valor predeterminado de 1. Debe tener métodos para calcular el perímetro y el área del rectángulo. Debe tener métodos *establecer* y *obtener* para *longitud* y *anchura*. Los métodos *establecer* deben verificar que *longitud* y *anchura* sean números de punto flotante mayores de 0.0, y menores de 20.0. Escriba un programa para probar la clase **Rectangulo**.

Ejercicio clase

Cree una clase llamada ***CuentaDeAhorros***. Use una variable `static` llamada `tasaInteresAnual` para almacenar la tasa de interés anual para todos los cuentahabientes. Cada objeto de la clase debe contener una variable de instancia private llamada `saldoAhorros`, que indique la cantidad que el ahorrador tiene actualmente en depósito.

Proporcione el método `calcularInteresMensual` para calcular el interés mensual, multiplicando el `saldoAhorros` por la tasaInteresAnual dividida entre 12; este interés debe sumarse al `saldoAhorros`. Proporcione un método `static` llamado `modificarTasaInteres` para establecer la `tasaInteresAnual` en un nuevo valor. Escriba un programa para probar la clase `CuentaDeAhorros`. Cree dos instancias de objetos `CuentaDeAhorros`, `ahorrador1` y `ahorrador2`, con saldos de $2000.00 y $3000.00 respectivamente. Establezca la `tasaInteresAnual` en 4%, después calcule el interés mensual para cada uno de los 12 meses e imprima los nuevos saldos para ambos ahorradores. Luego establezca la `tasaInteresAnual` en 5%, calcule el interés del siguiente mes e imprima los nuevos saldos para ambos ahorradores.

Ejercicio clase

Cree una clase llamada *Complejo* para realizar operaciones aritméticas con números complejos. Estos números tienen la forma: $parteReal + parteImaginaria * i$

en donde *i* es: $\sqrt{-1}$

Escriba un programa para probar su clase. Use variables de punto flotante para representar los datos private de la clase. Proporcione un constructor que permita que un objeto de esta clase se inicialice al declararse. Proporcione un constructor sin argumentos con valores predeterminados, en caso de que no se proporcionen inicializadores. Ofrezca métodos *public* que realicen las siguientes operaciones:

a) Sumar dos números *Complejo*: las partes reales se suman entre sí y las partes imaginarias también.

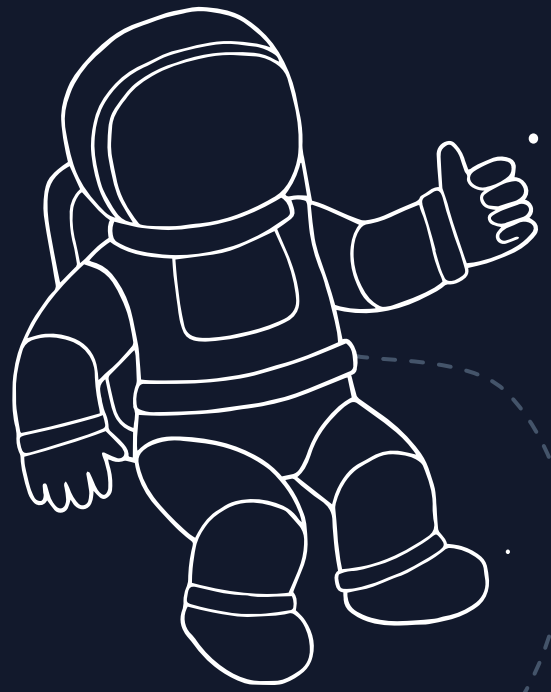
b) Restar dos números *Complejo*: la parte real del operando derecho se resta de la parte real del operando izquierdo, y la parte imaginaria del operando derecho se resta de la parte imaginaria del operando izquierdo.

c) Imprimir números Complejo en la forma:
(*parteReal*, *parteImaginaria*).

$$(3+2i)+(4+i)=(3+4)+(2+1)i=7+3i$$

$$(2-2i)+(2+3i)=(2+2)+(-2+3)i=4+i$$

$$(4+2i)-(2-3i)=(4-2)+(2-(-3))i=2+5i$$



Programa académico CAMPUS

Módulo JAVA