


Open · Opened 2 weeks ago by  **Rubén Montero**

Calcular y devolver un valor

Resumen

- Escribiremos **3** métodos distintos en `Greeter`.
 - Dos de ellos devolverán un número entero
 - El tercero *invocará* a los otros dos y sumará esos valores
- Veremos que un método puede invocarse desde *otro método*
- Invocaremos dicho método desde `Main`

Descripción

Hemos visto que:

1. Podemos escribir clases
2. Podemos *instanciar* clases (inicializarlas como variables)
3. Podemos escribir métodos
4. Podemos *invocar* dichos métodos desde `Main`

Pero... ¿podemos invocar los métodos desde *otros sitios*? Por ejemplo, ¿desde *dentro* de la misma clase?

La respuesta es...

...Sí.

Comencemos

En `Greeter.java`, añade un método llamado `firstNumber()`. Devolverá un tipo `int` y **tendrá visibilidad `private`**.

En el cuerpo, sólo *devolverá un valor*. El mismo que se indica a continuación:

```
private int firstNumber() {  
    return 15;  
}
```

Intenta *invocar* dicho método desde `Main.java`. ¿Hay algún problema?

Un segundo método

Elimina cualquier código que genere errores y añade un segundo método a `Greeter.java`, así:

```
private int secondNumber() {  
    return 46;  
}
```

Dos métodos privados

Como estos métodos son `private`, **no** se pueden *invocar* desde *fuera* de la clase.

Entonces borrémoslos, son inútiles

¡Un momento!

Escribamos un **nuevo** método en `Greeter.java`. Se llamará `sumTwoNumbers`

Esta vez, `public`.

También devolverá un tipo `int`:

```
public int sumTwoNumbers() {
```

```
}

```

Los métodos privados tienen sentido

Un método privado sirve para *encapsular* un trozo de código **dentro** de una clase. Piensa que en el futuro escribiremos clases con muchas líneas de código y **no** debe ir todo **junto** como un **churro**.

La idea de la programación estructurada y las *subrutinas* también tienen sentido **dentro de una misma clase**.

¿Cómo se hace?

Basta con escribir esto:

```
public int sumTwoNumbers() {
    return firstNumber() + secondNumber();
}
```

Sería equivalente a:

```
public int sumTwoNumbers() {
    int result = firstNumber() + secondNumber();
    return result;
}
```

```
public int sumTwoNumbers() {
    int first = firstNumber();
    int second = secondNumber();
    return first + second;
}
```

Déjalo como te sea más claro.

Un momento...

La ~~s~~ diferencia ~~s~~

Desde `Main.java`, en ejercicios anteriores, hemos invocado un método de `Greeter` así:

```
Greeter greeter = new Greeter();
greeter.sayHello();
```

pero ahora, desde `Greeter.java`, invocamos sus propios métodos privados así:

```
int first = firstNumber();
int second = secondNumber();
```

Hay **1** diferencia.

En `Main.java` invocamos los métodos *sobre una variable* (antes del punto (`.`)).

En `Greeter.java`, *escribimos los nombres directamente*. La pregunta es: ¿Sobre **qué** variable?

Repasemos... ¿Cuál es el problema?

Se supone que un método es parte de una *variable* (o *instancia*) de una clase en particular.

En `Main.java` queda claro **qué** *variable* ejecuta sus métodos. La hemos creado en la línea:

```
Greeter greeter = new Greeter();
```

Pero... ¿en `Greeter.java` ?

this

Lo entenderemos mejor si vemos que estas dos líneas son **equivalentes**:

```
public int sumTwoNumbers() {
```

```
int first = firstNumber();  
int second = secondNumber();  
return first + second;  
}
```

```
public int sumTwoNumbers() {  
    int first = this.firstNumber(); // Usamos this  
    int second = this.secondNumber(); // Usamos this  
    return first + second;  
}
```

`this` se refiere a **la instancia actual**.

Explicación:

- Si el método `sumTwoNumbers` está siendo *invocado*, **alguien** ha creado una *instancia*
- No sabemos **qué nombre** han elegido para esa *instancia* (pudo ser `greeter`, `myGreeter`, ...), **ni nos interesa**.
- `this` es un *sinónimo* de esa variable, **sea cual sea**. Sólo se puede usar **dentro** de la clase.

Para terminar

Si se ha entendido a la perfección, bien.

Si no, lo dejaremos macerando e iremos comprendiendo más con los siguientes ejercicios.

Ahora, *invoca* el método `sumTwoNumbers` desde `Main.java` e **imprime** el resultado, así:

```
public class Main {  
    public static void main(String[] args) {  
        // ...  
        System.out.println(greeter.sumTwoNumbers());  
    }  
}
```

¿Encaja?

Por último

Una vez verifiques que el *test* funciona correctamente, el ejercicio ha sido completado.

Haz `commit` y `push` para subir los cambios al repositorio.



Rubén Montero @ruben.montero changed milestone to [%Sprint 1](#) 2 weeks ago