


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

Devolver un valor

Resumen

- Distinguiremos entre función y procedimiento
- Implementaremos un método `byeWorld` en nuestra clase `Greeter`, que devolverá un `String`
- Invocaremos dicho método desde `Main`

Descripción

Un **procedimiento** (o subrutina) es una *pieza* de código fuente que se ejecuta de forma separada. Al igual que los métodos de las clases, puede *invocarse*.

Una **función** es, igualmente, una *pieza* de código fuente que se ejecuta de forma separada. La diferencia es que **devuelve un valor**.

Hemos usado procedimientos

Nuestras clases declaraban `void` como *tipo de dato devuelto*. Por ejemplo, en `Greeter` tenemos:

```
public void sayHello() {  
    System.out.println("Hello world!");  
}
```

Escribamos funciones

Vamos a estrenar un **nuevo método** en `Greeter` que **devuelva un valor**. Lo escribiremos *debajo* de `sayHello()`.

Llamémoslo `byeWorld`. Devolverá un dato tipo `String`:

```
public void sayHello() {  
    System.out.println("Hello world!");  
}  
  
public String byeWorld() {  
  
}
```

Si te fijas, **aparece un error de compilación** que indica:

```
Missing return statement
```

Claro.

Declaramos que `byeWorld` *devuelve* un `String`, pero... ¿Qué `String`? ¿Qué **valor** exactamente *devuelve*?

Debemos indicarlo con la palabra especial `return`.

Nosotros vamos a devolver el valor: `Bye!`. Así:

```
public String byeWorld() {  
    return "Bye!";  
}
```

¿Para qué sirve devolver algo?

Es el principio básico de funcionamiento de la programación estructurada.

Un programa se divide en partes que procesan datos y producen un resultado. Ese *resultado*, se entrega al resto del programa *devolviéndolo*.

Usémoslo en `Main`

Por ejemplo, en `Main.java` añadamos la siguiente sentencia:

```
public class Main {  
    public static void main(String[] args) {  
        // ...  
        String miString = greeter.byeworld();  
    }  
}
```

Estamos haciendo algo importante:

1. Declarar una nueva variable tipo `String`, llamada `miString`
2. Invocar `greeter.byeworld()`
3. **Asignar** el valor *devuelto* a `miString`

A continuación, desde *esta parte del programa* podremos trabajar con dicho resultado.

Terminando...

Añade un `System.out.println()`¹ así:

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


Ejecuta la aplicación principal. Deberías ver varios *print* que se producen desde *distintas partes del programa*.

¿Comprendes el *flujo de ejecución*?

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.

-
1. Desde IntelliJ IDEA, si tecleas `sout`, verás que te permite autocompletar automáticamente a `System.out.println()`. También se pueden configurar más *shortcuts* 
-



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