# INSTALACION VISUAL STUDIO CODE

Datos del profe:

Paco Garcia Serrano

Usuario de Github: serraguti

Email de contacto: pacoserranox@gmail.com

Descargamos Visual Studio Code

<https://code.visualstudio.com/>

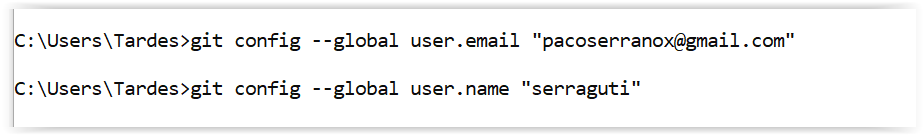
Instalamos Git for Windows

<https://git-scm.com/downloads/win>

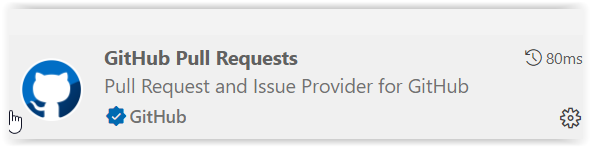
Incluimos, dentro de la línea de comandos (cmd) el usuario y email de Github

git config --global user.email "tuemail@example.com"

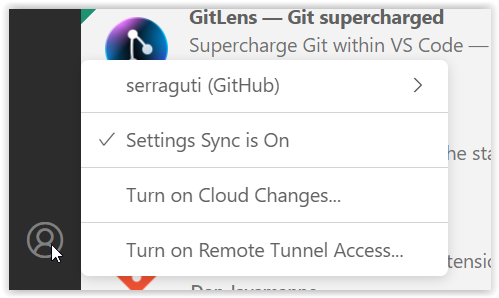
git config --global user.name "usuariogithub"



Dentro de Visual Studio, instalamos la Extensión de GitHub



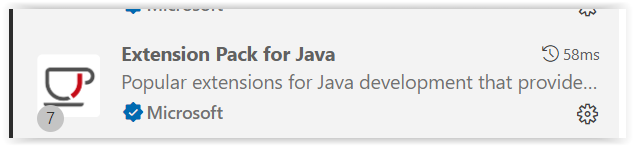
Nos validamos con nuestro usuario de Github dentro de Visual Studio



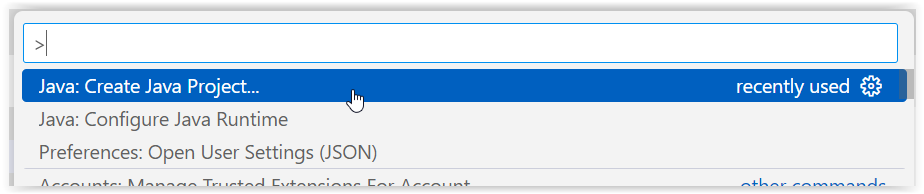
Pulsamos sobre **Backup and Sync**

Tendremos todo listo en casa para trabajar.

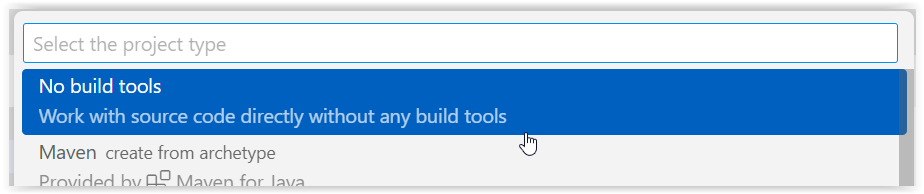
Por si algo falla, instalamos la extensión Java Extension Pack



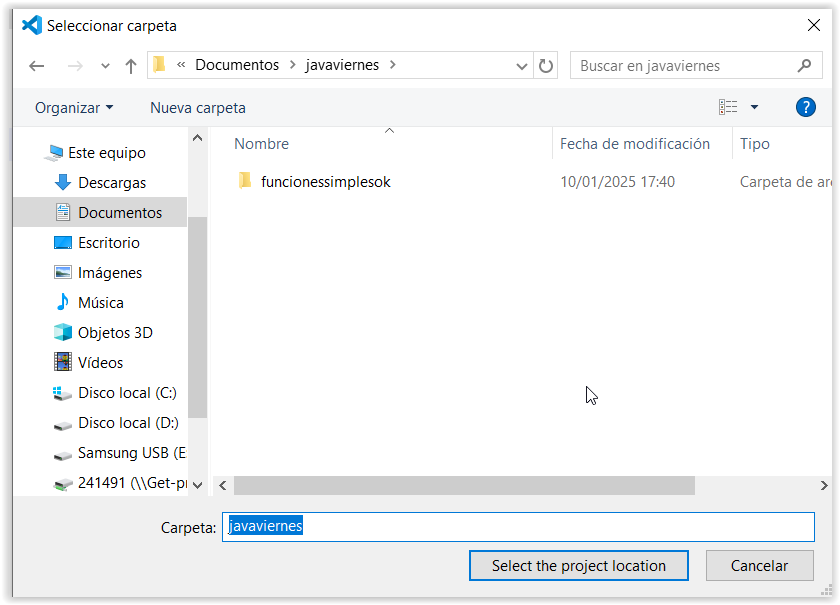
Para crear un nuevo proyecto de Java dentro de VS Code, debemos utilizar la siguiente combinación de Teclas: **CONTROL + SHIFT + P**

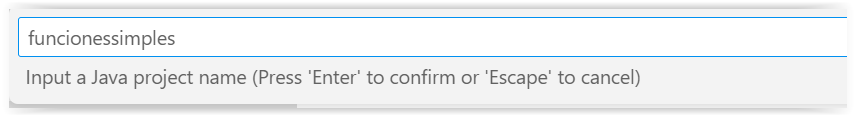


Seleccionamos **No Build Tools**

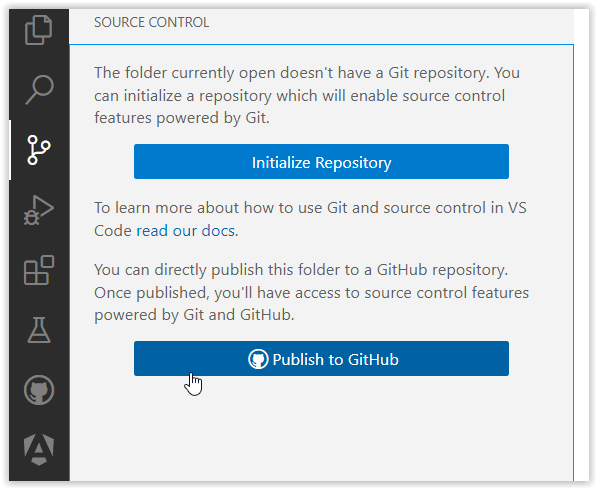


Indicamos la ubicación de nuestro proyecto nuevo.



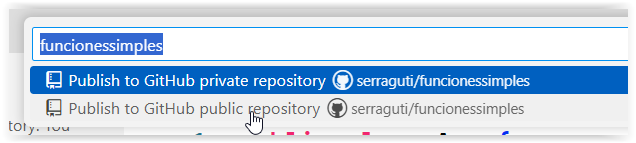


Para sincronizar con Github en Visual Studio, seleccionamos **Source Control**



Pulsamos sobre **Publish to GitHub**

Nos pide nombre del proyecto y visibilidad



Indicamos las carpetas que deseamos sincronizar



Existe un sistema de teclas para poder comentar código de forma automática dentro de Visual Studio Code.

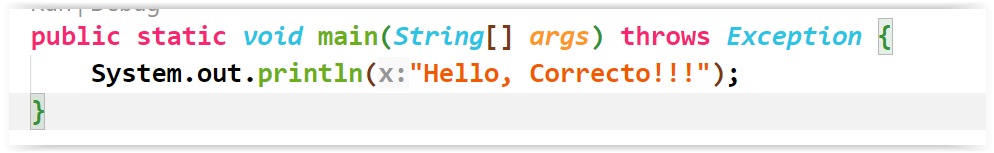
Para comentar código: Control + k + c

Descomentar: Control + k + u

**FUNCIONES DENTRO DE JAVA**

Llevamos utilizando funciones de Java desde el primer día.

Una función es un código que se ejecuta en una clase.



Las funciones nos permiten poder “dividir” códigos en funcionalidades para poder reutilizar el código desde múltiples puntos.

Las funciones no tienen que estar de forma obligatoria en la misma clase, podemos tener múltiples clases con múltiples funciones.

Actualmente, vamos a trabajar con funciones de tipo **static**

Una función **static** es una función que se llama directamente desde la clase.

Existen funciones de objeto y funciones de clase (**static**).

Dentro de Java podemos crear objetos para trabajar con ellos.

Dichos objetos nos sirven para representar elementos personalizados.

Ejemplo: Actualmente estamos en el momento de “utilizar” los objetos.

Si yo quiero guardar un texto: String texto = “soy un texto”;

Si yo quiero guardar un número: int numero = 77;

Si yo quiero guardar una Persona?? (Nombre, Apellidos y Edad??)

Cuando tenemos un objeto, podemos incluirle propiedades y funciones.

Una propiedad es una característica de un objeto

Una función es una acción de un objeto.

Persona persona;

persona.nombre = “Alumno”;

persona.edad = 25;

persona.**comprobarEdad()**; //este método apunta al objeto Persona.

Quiero poder validar cualquier DNI de cualquier persona.

Elementos necesarios. Un código para validar un DNI

Dicho código debe estar en algún sitio (Clase)

Validaciones.**validarDni(DNI)**;

Validaciones.**validarMatricula(matricula)**;

Una función es un conjunto de instrucciones.

Dentro de una función tenemos dos opciones de trabajar con funciones:

1. **void:** Estas funciones son un conjunto de instrucciones

Sintaxis:

public static void nombreFuncion()

{

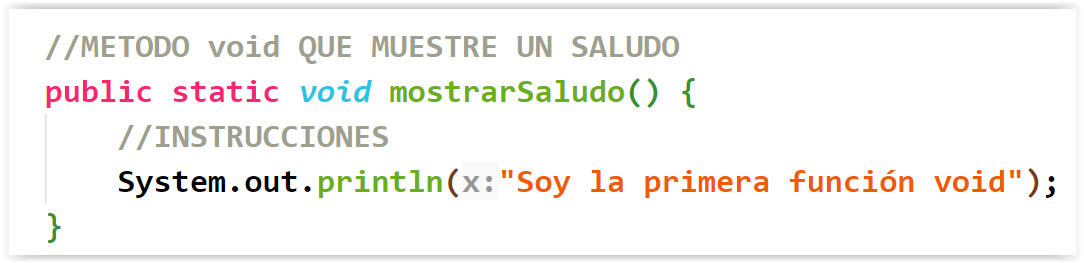
//INSTRUCCIONES

}

Debemos escribir siempre en la clase **App**

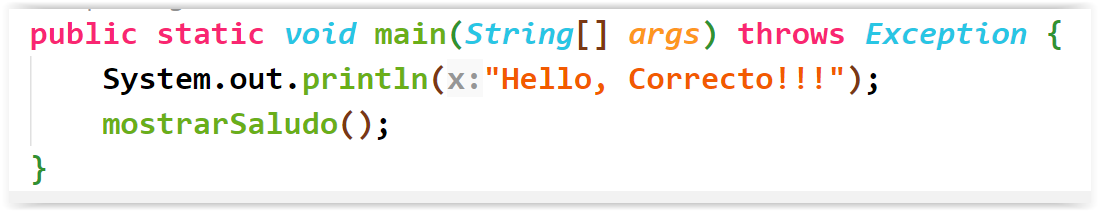
Esto es la declaración de una función **void**

Una función NO sirve de nada si no la llamamos.



Para llamar a una función **void** debemos hacerlo desde otra función (**main**)

La llamada a una función **static** DENTRO de la misma clase, simplemente se escribe el nombre de la función para la llamada con los paréntesis.



1. **return:** Sonfunciones con un conjunto de instrucciones y que devuelven un valor cuando se han ejecutado.

public static TIPODATO nombreFuncion()

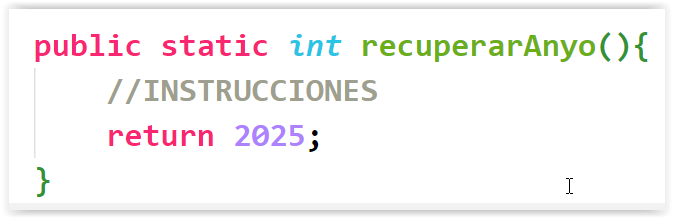
{

//INSTRUCCIONES

return VALOR\_DEL\_TIPO\_DATO;

}

Vamos a realizar una función que nos devolverá un número (2025).



La llamada se realiza igual pero se pierde en el olvido…

En la llamada a una función **return** debemos almacenar el valor que devuelve en la petición a la función.



**PARAMETROS EN LAS FUNCIONES/METODOS**

Una función puede recibir parámetro o parámetros para su ejecución.

Los parámetros son enviados entre los paréntesis de la declaración de la función y estarán separados por comas.

Los parámetros NO son opcionales, es decir, si declaramos una función con parámetros **debemos** enviar dichos parámetros en la llamada.

Es la misma sintaxis para todas las funciones (void, return)

public static void metodoParametros

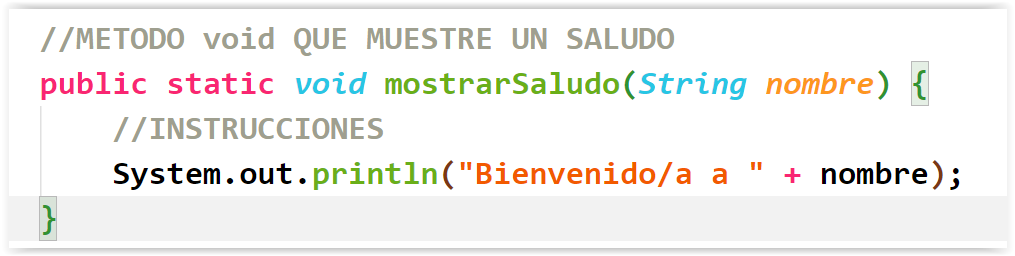
(**tipoDato nombreParam1, tipoDato nombreParam2**)

{

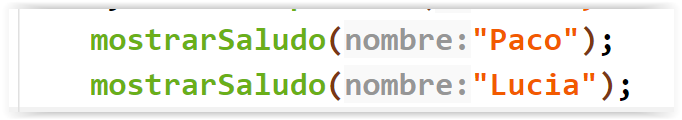
//INSTRUCCIONES

}

Por ejemplo, vamos a modificar el método **mostrarSaludo()** recibiendo un nombre para saludar.

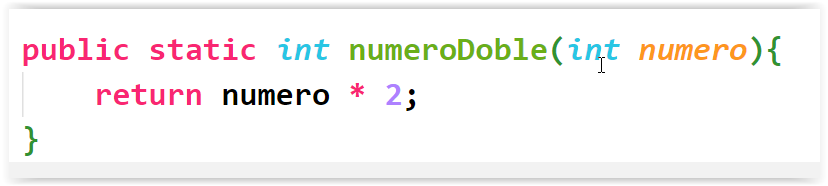


En la llamada debemos enviar el valor para los parámetros

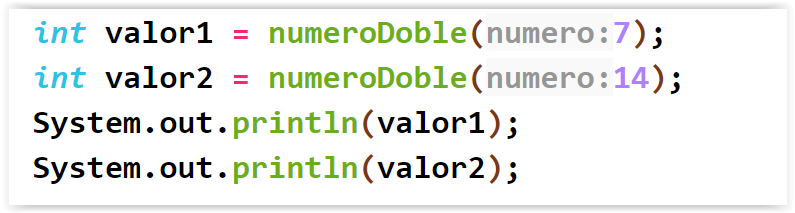


Lo mismo sucede si hacemos un método/función **return**

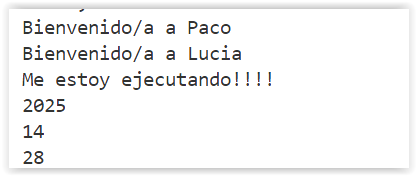
Vamos a realizar un método que devuelva el doble de un número que recibamos.



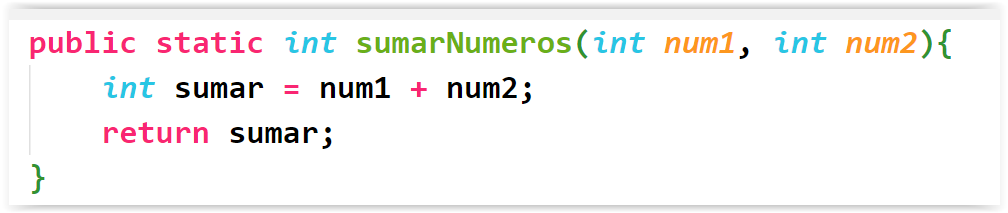
Las llamadas ya son dinámicas dependiendo del número



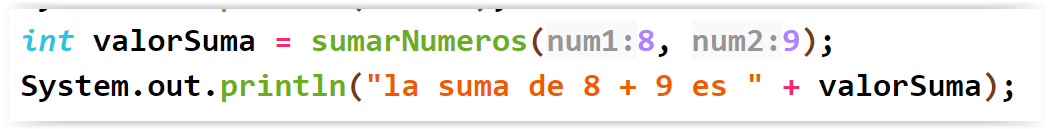
Salida del programa



Por ejemplo, vamos a realizar una función para devolver la suma de dos números.

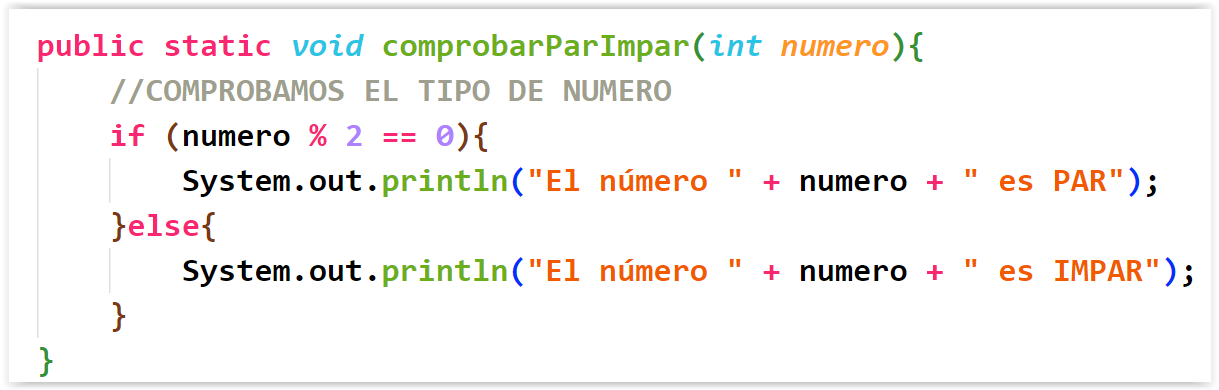


Llamada al método:

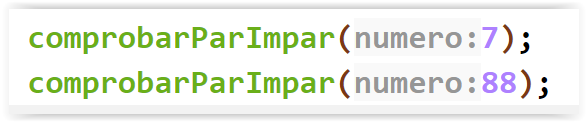


Vamos a realizar una función/método que recibirá un número y **mostrará** si el número es par o impar.

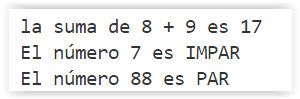
El símbolo para el resto en Java es **%**



Realizamos la llamada:



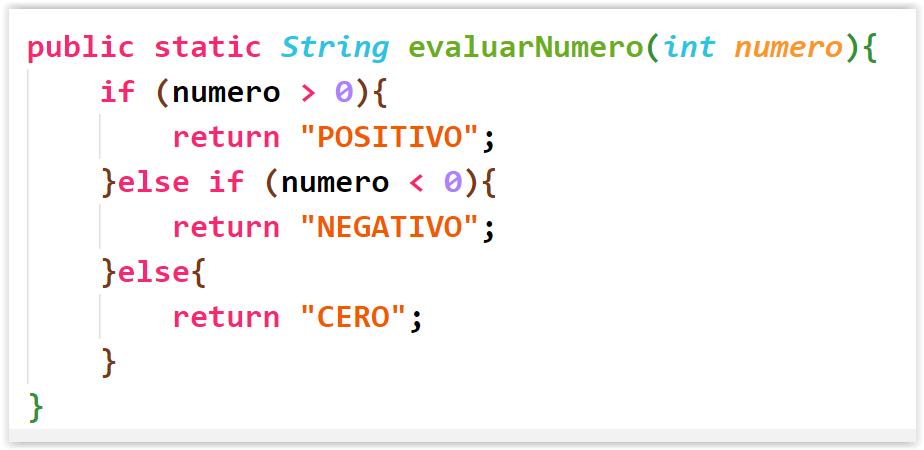
Y veremos el resultado en pantalla



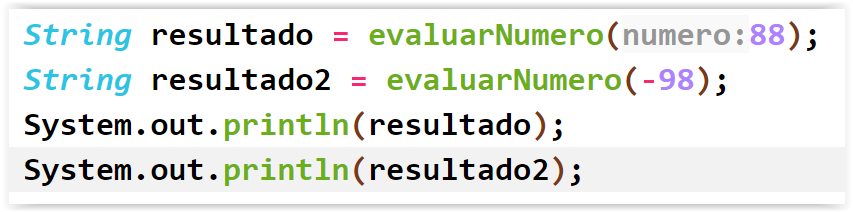
Ejemplo:

Creamos una nueva función (en la misma clase) que recibirá un número.

Debemos **devolver** si el número recibido es POSITIVO, NEGATIVO o CERO



Realizamos la llamada



Resultado de la ejecución

