M

Manual Intellij

David Espinosa González

IES Palomeras Vallecas

1º DAW

1. Personalización del entorno.

**2**

1. Uso del depurador de Intelij.

**5**

1. Shortcuts en InteliJ

**7**

1. Java Code Templates

**4.1** Ejemplos de Java code templates

**8**

**4.2** Como crear un java code template en intelij

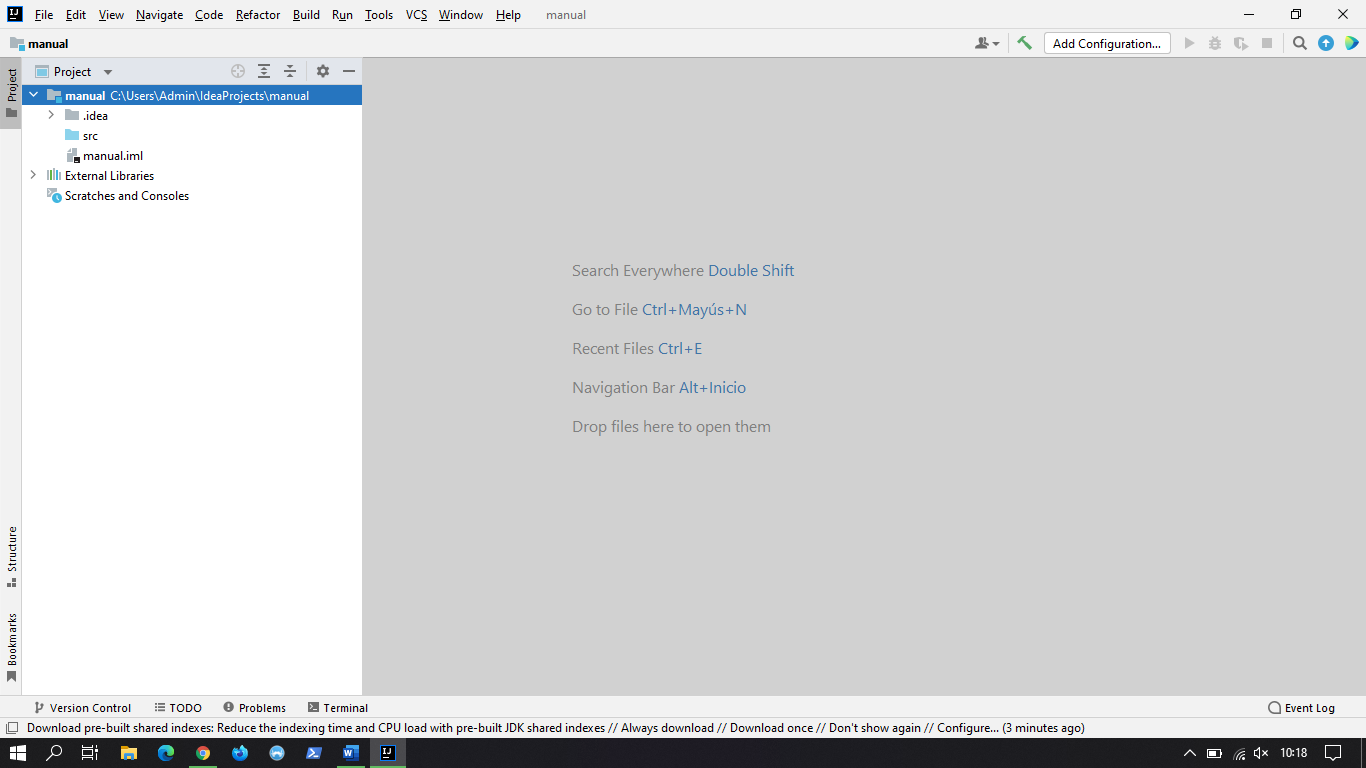
**9**

1. Enlazar “javadoc” a un archivo local

**11**

* + - 1. **Personalización del entorno.**

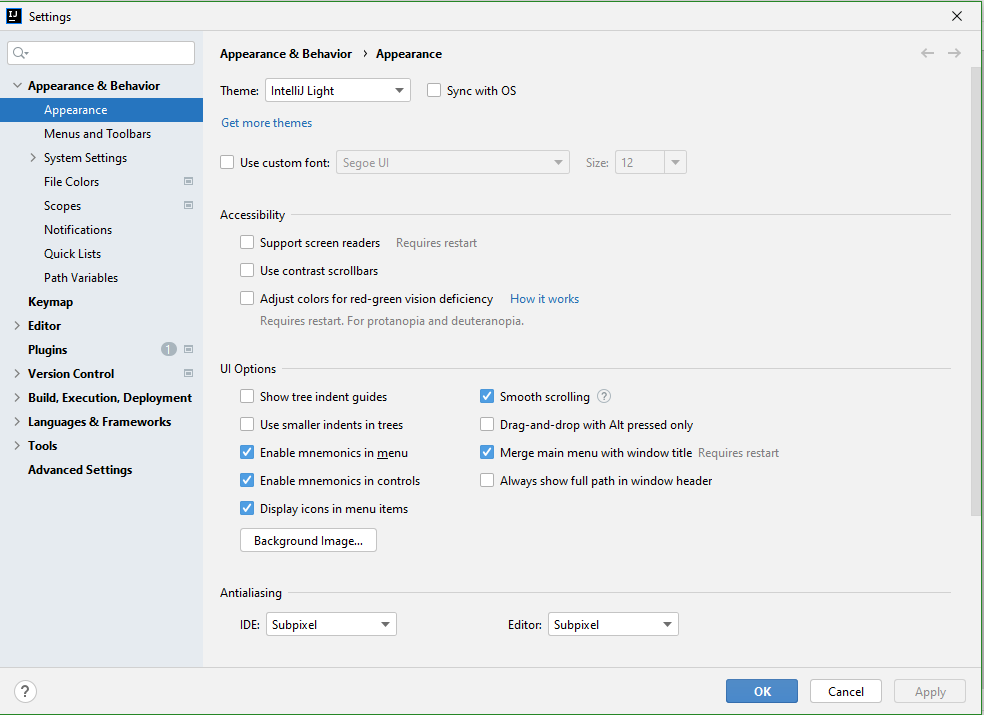
Al igual que la mayoría de entornos, el tema del entorno por defecto es blanco. A continuación, se muestra una imagen del entorno sin haber sido modificado:



He creado un nuevo proyecto llamado manual donde iremos avanzando tanto en la personalización del entorno como en las siguientes partes del manual.

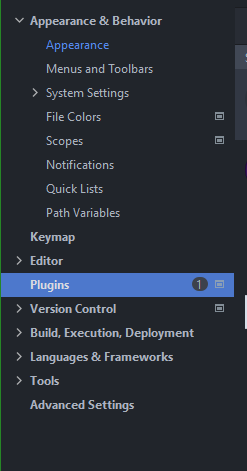
Lo primero que ocurre al abrir el entorno, es la elección del proyecto sobre el que se va a trabajar. Si no quieres abrir un proyecto existente, también puedes crear un de 0. Después, aparecerá una ventana con consejos de como optimizar el entorno. Es una especie de tutorial sin fin, así que lo cerraremos para poder continuar con la personalización.

EL siguiente paso es abrir la pestaña de ajustes, está la podremos encontrar en la primera opción de la barra de herramientas del entorno (File), siendo la quinta opción del desplegable (Settings). Una vez abierta la opción de Settings podremos observar la siguiente pestaña:

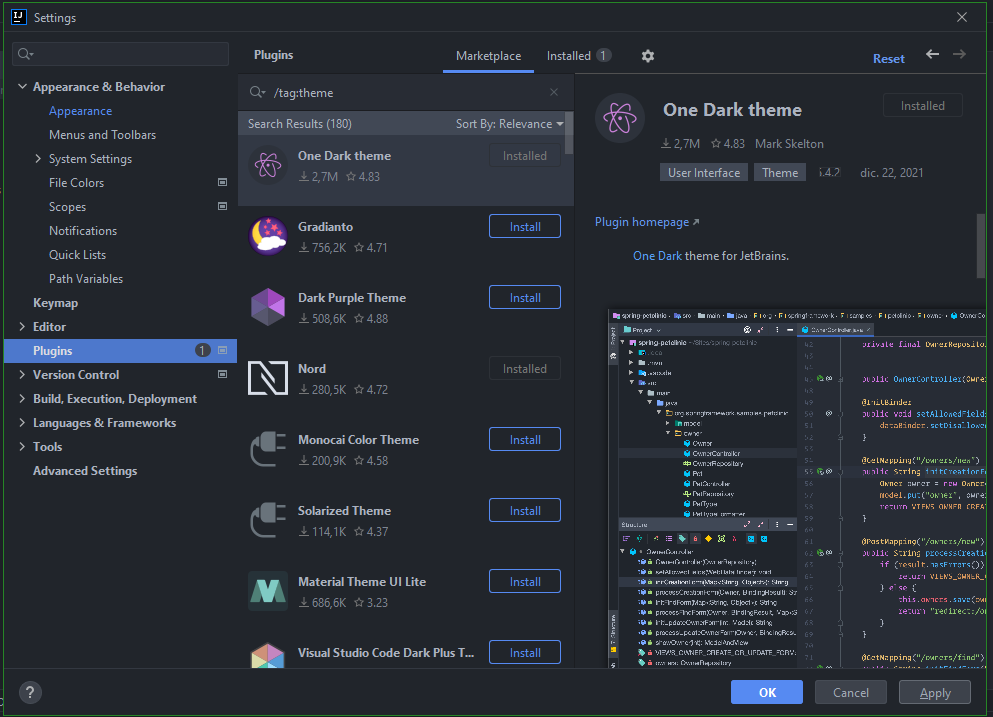
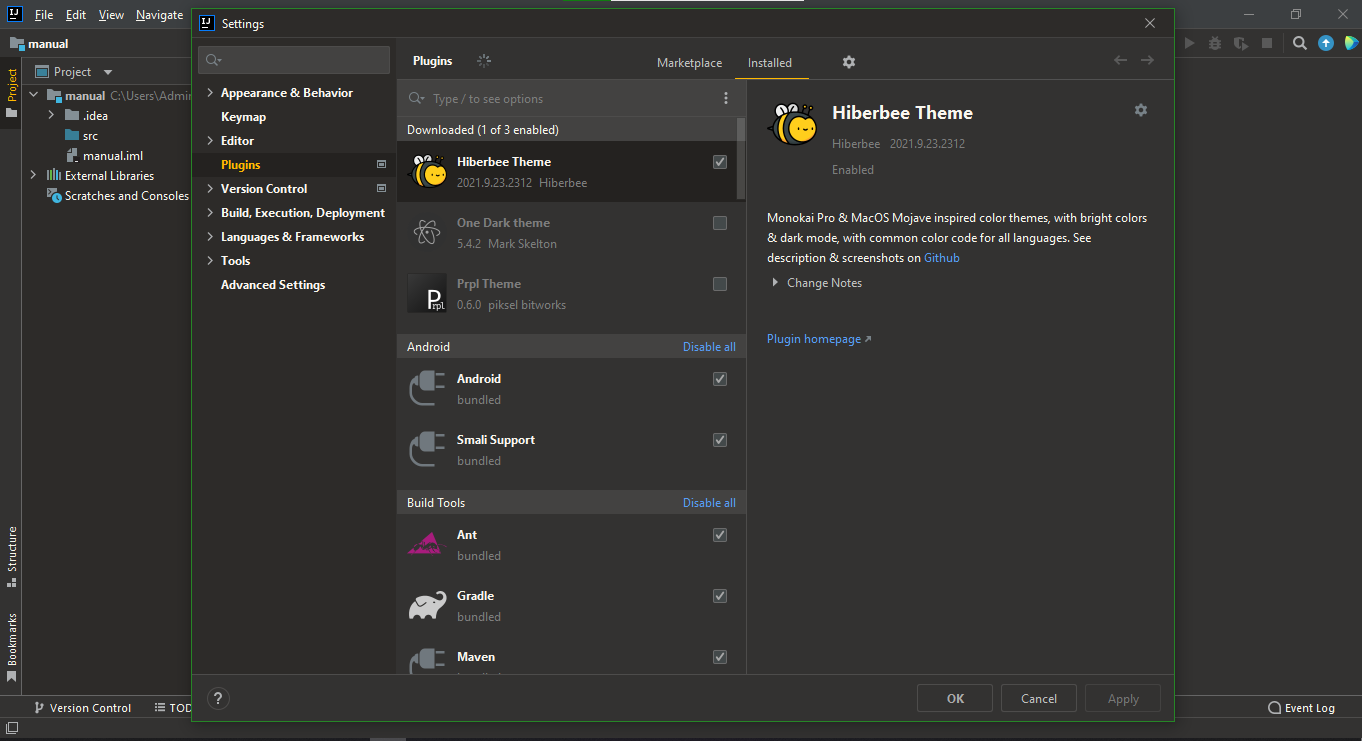


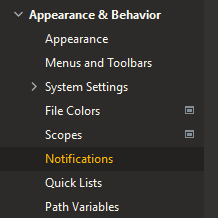
Tal y como se ve en la imagen, nos encontramos en la modificación de la apariencia general del entorno. Existen varios temas generales que podemos usar en vez de ir parte a parte cambiado los colores y demás.

El tema que yo he escogido es “Darcula”. Además, se puede cambiar la fuente predeterminada, esto se puede realizar seleccionando el cuadrado que se encuentra debajo de Themes, donde aparece la opción de: “Use custom Font”. Si no te gustan ninguno de los temas que te ofrece de forma predeterminada, puedes seleccionar la opción que se encuentra debajo de “Themes” que se llama: “Get more themes”. Seleccionando esta opción, la pestaña cambiara a la opción de Pluggins que se encuentra en la barra de la izquierda de la pestaña:



En esta parte podremos observar un desplegable de temas muy variados para poder personalizar nuestro entorno más a nuestro gusto.

Después de revisar los temas en esta parte, cambie mi opción del tema a otro que visualmente me llamo más: “HiberBee Theme”.

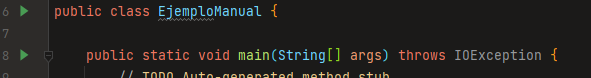


En esta parte de la barra, también podrás modificar el resto de la interfaz del entorno como las barras de los menús, como ver las notificaciones, etc.

* + - 1. **Uso del depurador en IntelliJ**

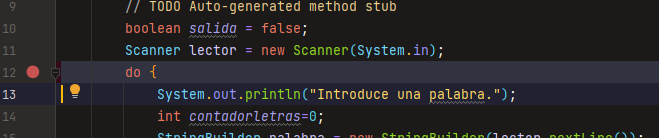
Para poder ver el uso del depurador, se va usar como ejemplo un ejercicio dado en clase anteriormente. El ejercicio en cuestión será el programa que comprueba si una palabra es o no un palíndromo.

Lo que visualmente llama la atención en comparación con los anteriores entornos de desarrollo es que cuando se crea un método aparece al lado del método la opción de lanzarlo.



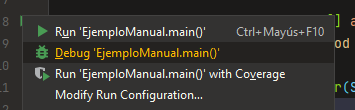
Son las formas de color verde. Al presionar en ellas aparece un desplegable en el que puedes escoger entre lanzar el programa, debugearlo, lanzar el programa de una forma segura o modificar las opciones del lanzador.

Para que haya diferencia entre lanzar el programa y debugear el programa es necesario recurrir al uso de los breakpoints para poder ver paso a paso el recorrido del código. Recuerda que el programa se lanzara de forma normal hasta el breakpoint. Para colocar un breakpoint se hará click sobre la línea de código, justo a la derecha de la numeración de las líneas sin llegar a seleccionar la parte de escritura del código. Cuando el breakpoint este puesto se verá como en la imagen a continuación, con un círculo rojo:

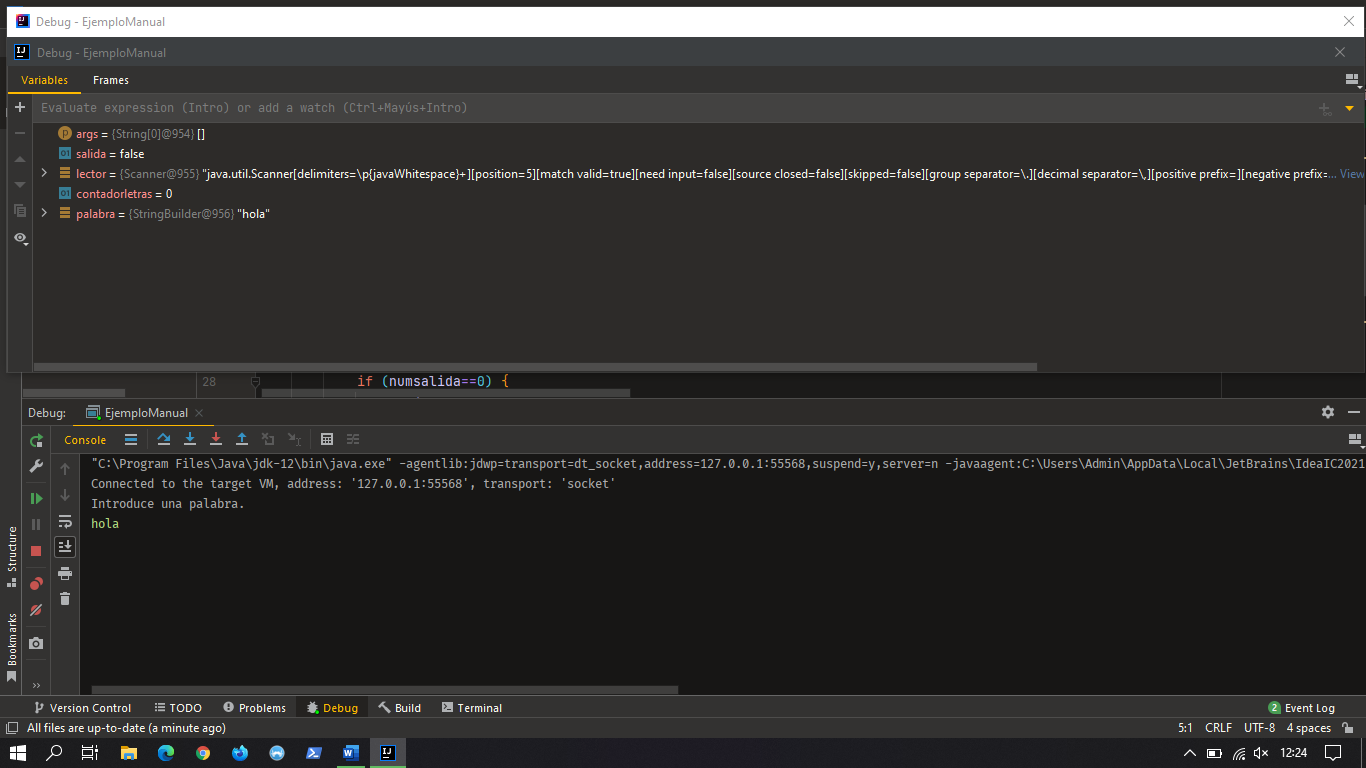


Para quitar el breakpoint será tan sencillo como hacer click sobre ese punto rojo.

Una vez que tenemos un breakpoint para el debugeador veamos que ocurre cuando se lanza el debugeador. Para ello hay que hacer click sobre el triángulo verde del método, y seleccionar la opción del debugeador.



Cuando el código llega al breakpoint seleccionado, se mostrará una nueva pestaña que al igual que en otros entornos muestra las variables. En mi caso, yo saque la ventana en cuestión de la interfaz para poder tenerla visualmente más accesible.



Tal y como se ve en la imagen, en la mitad de la pantalla se puede ver el ejercicio, y debajo de la consola. Las opciones que se encuentran junto a la consola son las que nos irán moviendo por el código.

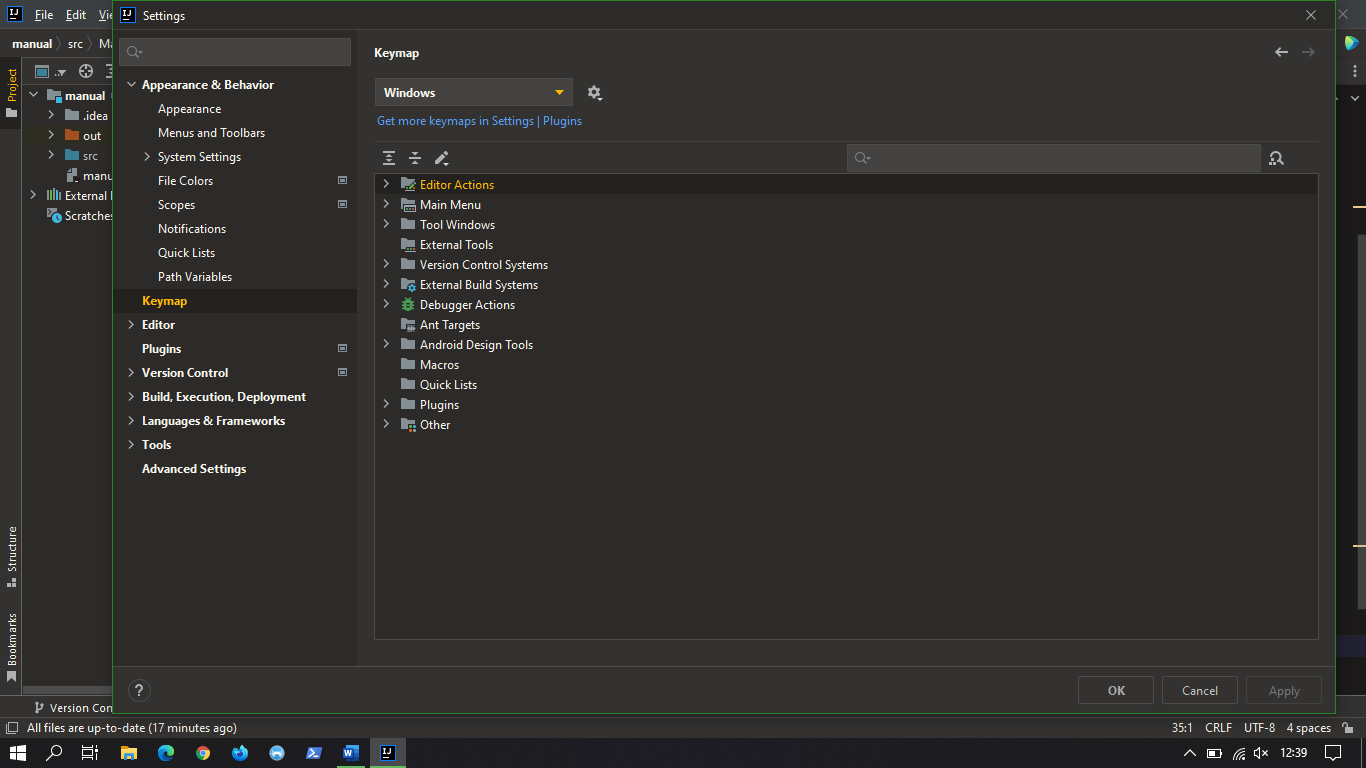


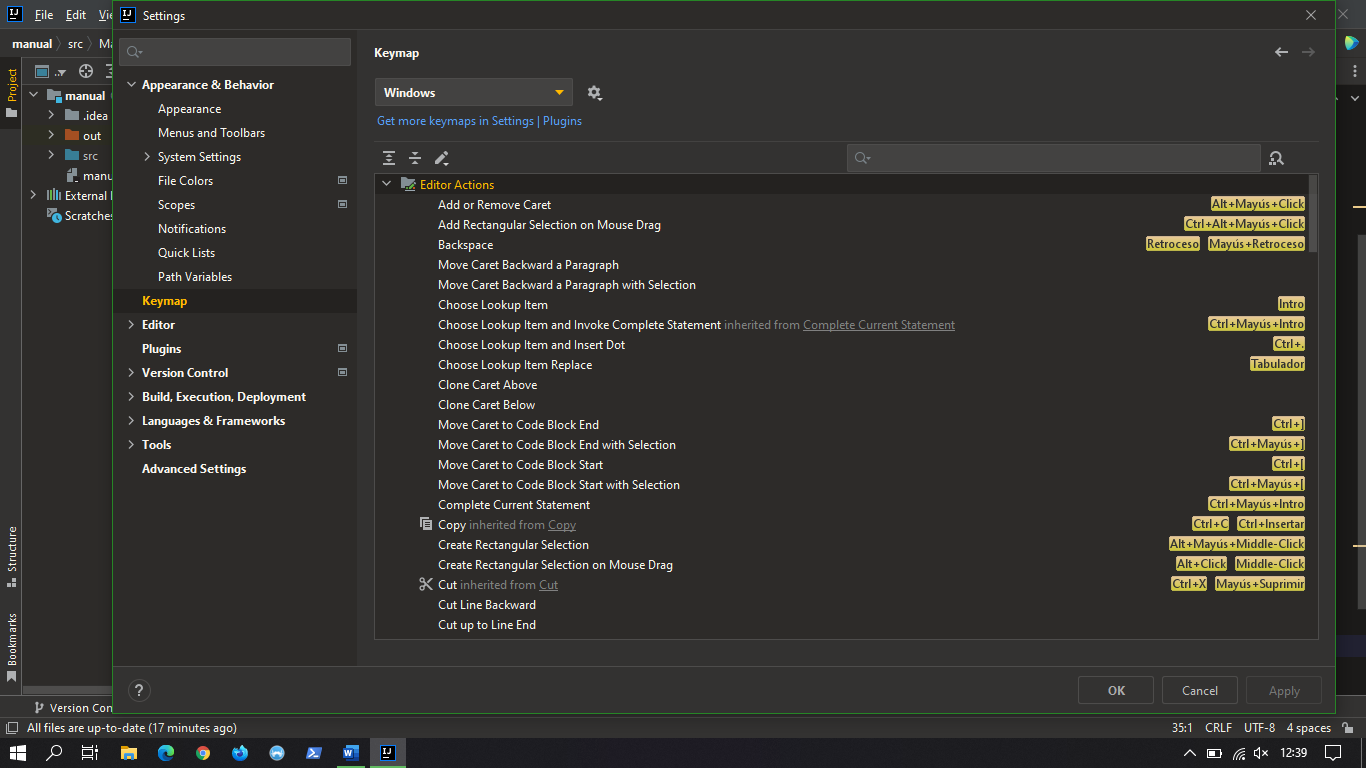
La opción que más usaremos será la primera, la que tiene la flecha doblada que es la opción de step over. Esta opción nos permitirá continuar con la siguiente línea del código sin meternos dentro de métodos externos que usemos (como por ejemplo println). Podemos usar el F8 también como shortcut para seleccionar esa opción.

Al igual que los anteriores entornos, la pestaña de variables se ira modificando según el debugeador avance por el código. También se ira marcando la línea de código en la que nos situamos como referencia de en qué parte se encuentra el debugeador.

* + - 1. **Shortcuts en IntelliJ**

Se pueden consultar todos los shortcuts en la pestaña de “Settings”, en el apartado de “Keymap”. Una vez ahí, se pueden abrir los desplegables para consultar cada uno de los shortcuts.





Observando la lista que ofrece el entorno, los shortcuts que más me han llamado la atención fuero:

* Alt+O: Este shortcut lo que provoca es el código se exporte a un documento de texto.
* Ctrl+Espacio: Este hace que aparezca un desplegable con las opciones de los Java Code Templates.
* Ctrl+Mayus+F5: Esto fuerza una recarga del programa.
* Ctrl+Mayus+F10: Esto lanza el método en el que se esté.
* Ctrl+F2: Esto para la ejecución del código que este lanzándose o en el debugeador.

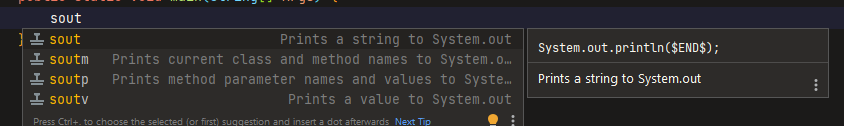
1. **Java code templates**

4.1 Ejemplos java code templates

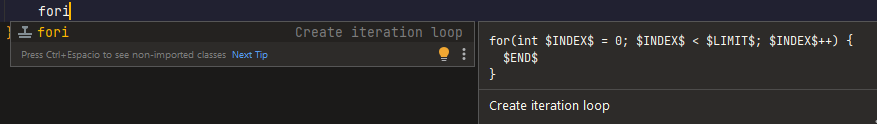
Un java code template es una abreviación para escribir directamente un método o formula. Podría decir que es un shortcut, pero más complejo. La gran diferencia entre un shortcut y un java code template es que el funcionamiento de los shortcuts no depende de la conexión a internet. Sin embargo, los java code templates si necesitan esa conexión a internet o en su defecto tener almacenada esa información en un fichero externo. Este fichero se llama comúnmente JavaDoc. En el ultimo apartado del manual se mostrará como enlazar el archivo JavaDoc a el entorno para no depender de la conexión para poder usar los Java code templates.

A continuación, se muestra una lista de 5 Java code templates que más comúnmente se utilizan:

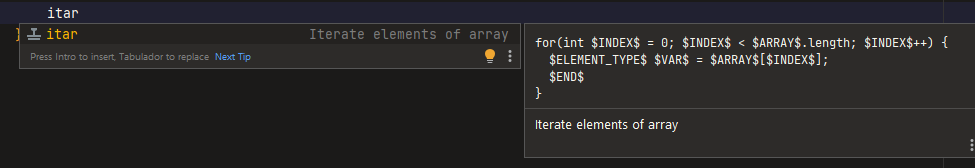
* Sout: Este Code template inyecta directamente lo siguiente -> Sistem.out.println();



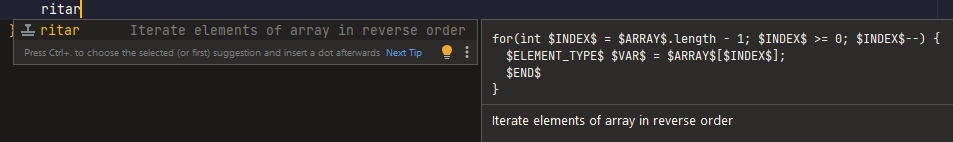
* Fori: Este code template inyecta directamente la forma con los corchetes y las condiciones de un for con variable i.



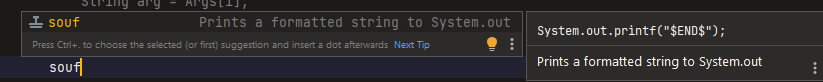
* Itar: esta opción itera los componentes de un array dentro de un string.



* Ritar: esta opción es la misma que la anterior, pero en esta el array se itera a la inversa.



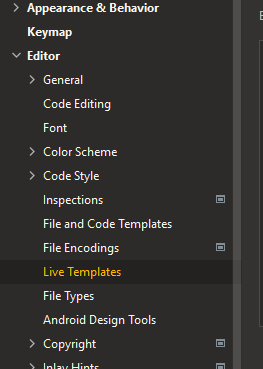
* Souf: Es otra versión diferente al sout. En esta no se realiza un println, si no un prinf.

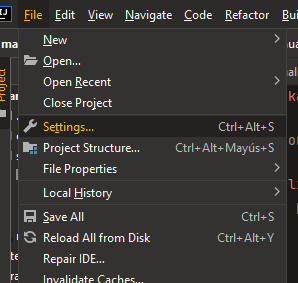


* 1. Creación de un Java code template.

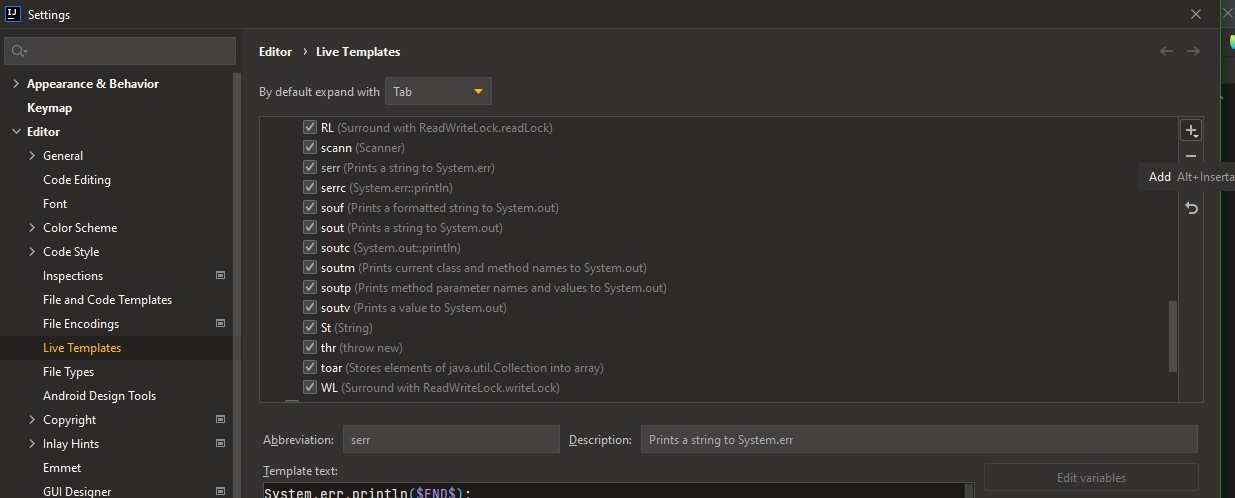
En este apartado crearemos nuestra propia abreviación para introducir un método en el entorno. El método escogido es la introducción de la clase scanner.

La abreviación para este code template será Scann, y el resultado será:

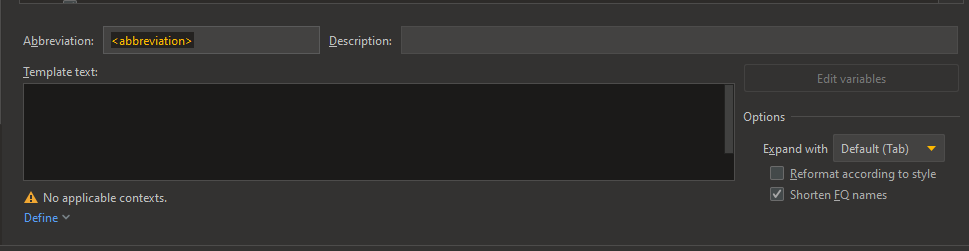
Scanner {x} = new Scanner(System.in);

Lo primero que haremos será dirigirnos a Settings\Editor\LiveTemplates.

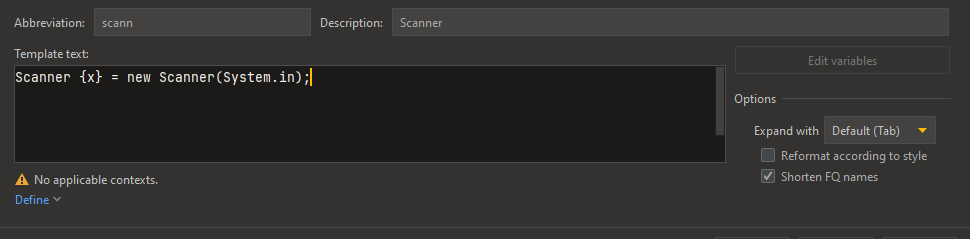
Una vez estemos en este apartado, tendremos que irnos a la parte de la derecha del todo, donde se encuentra un símbolo de +.

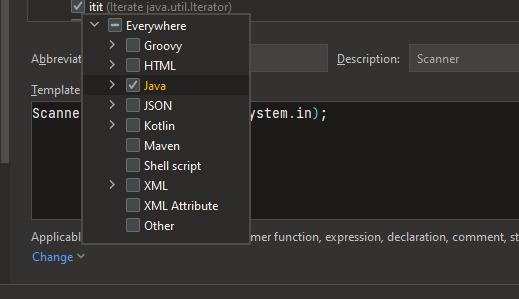


Deberemos de seleccionar ese símbolo de + para poder introducir nuestro Code template. Deberemos seleccionar la opción de live template para poder continuar, ya que la opción de template group lo que hará será crear una nueva carpeta para mas code templates, y ya que las carpetas están divididas por los lenguajes de programación, yo creare el code template dentro de la carpeta de Java. Una vez este seleccionada la opción de live template, se nos habilitara una opción para modificar/crear el template a nuestro gusto.



La abreviación que usare, será Scann. En la descripción solamente dejare el nombre del método que es Scanner. Y, por último, en el template text, introduciremos el código que queremos tener al final. Al final quedara tal que así:

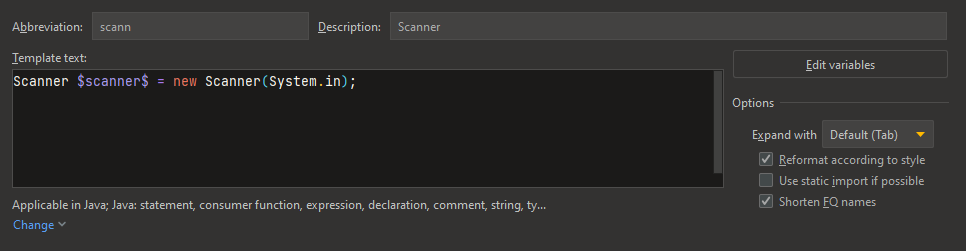




Una vez tengamos esto, procederemos a definir el template. Para ello, seleccionaremos la opción de define que se encuentra en la parte inferior izquierda de la imagen, y seleccionaremos la opción de Java.

Una vez seleccionemos java, solo habrá que aplicar y aceptar esa ventana y tendremos nuestro code template.

Para hacer un code template que sea mas versátil, cambie la parte “{x}” por “$scanner$”, y de esta forma al introducir el code template, nos permite modificar y poner el nombre al scanner de una forma más rápida.

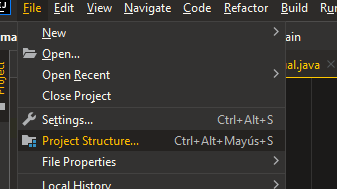




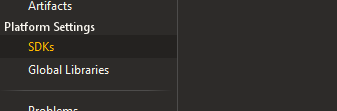
1. **Enlazar javadoc al entorno**

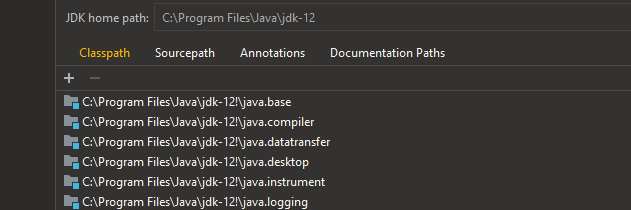
Para enlazar el archivo JavaDoc a este entrono, lo primero que hay que hacer es colocar el archivo .zip del javadoc dentro de la carpeta del entorno sin descomprimir.

Una vez se ha realizado el paso anterior, nos iremos a Files\ProyectStructure:

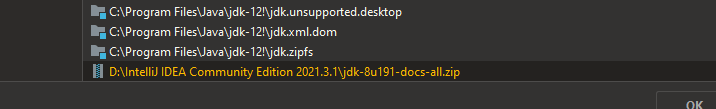


Dentro de la nueva pestaña que se nos abre, iremos a seleccionar la opción de SDKs para poder ver las librerías que el proyecto esta utilizando. Para enlazar el JavaDoc lo que haremos será agregar el zip como un ClassPath a las librerías existentes.





Para agregar el archivo seleccionaremos el símbolo de + que se encuentra debajo de ClassPath y seccionaremos el archivo .zip dentro de las opciones del desplegable. Una vez seleccionado el archivo .zip aparecerá en la última opción de la lista.



De esta manera ya tendremos nuestro entorno preparado para trabajar sin conexión.