MANUAL ECLIPSE

Por: David Espinosa González

**Índice**

1. Personalización del entorno.

**2**

1. Uso del depurador de eclipse.

**5**

1. Shortcuts en eclipse

**7**

1. Java Code Templates

**4.1** Ejemplos de Java code templates

**9**

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

**10**

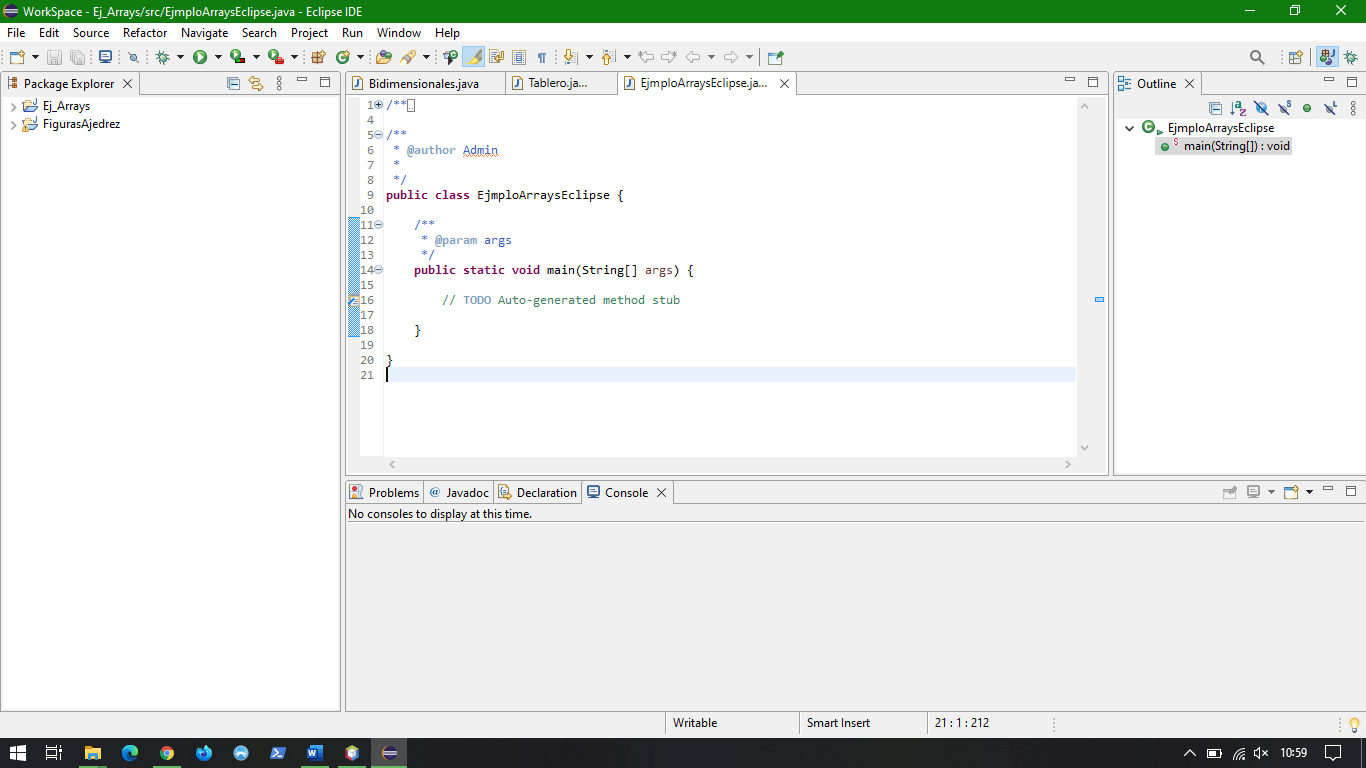
1. Enlazar “javadoc” a un archivo local

**11**

1. **PERSONALIZACION DEL ENTORNO**

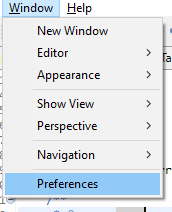
La personalización del entorno es una parte importante porque nos va a ayudar a estar mas cómodos con el programa. En este caso se va a ver como personalizar el entorno de “Eclipse”, y se mostraran ejemplos de la personalización del usuario.

En un principio, el aspecto de la aplicación es el siguiente:



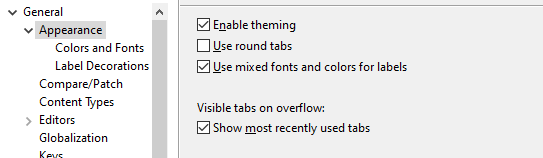
La customización de la aplicación es tanto del fondo, en este caso blanco, como de la fuente de las letras.

Para comenzar la personalización hay que abrir el desplegable de “Window”. Se encuentra como la penúltima opción de la barra de navegación del entorno. En el desplegable nos aparecerá una opción de “Preferences”, que será la que debemos seleccionar para abrir las opciones del entorno.

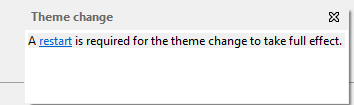


Una vez dentro, debemos desplegar “General”, y en la primera opción nos encontraremos “Appearance”.

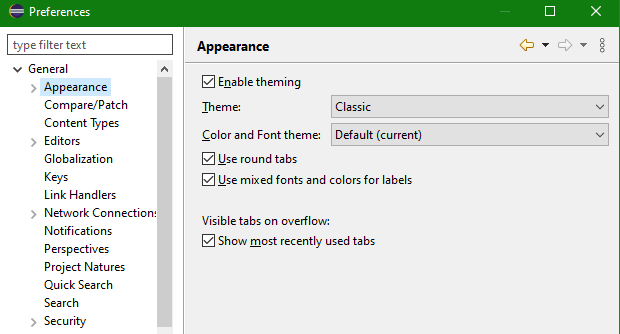
Dentro de “Appearance” hay que, si la opción esta deshabilitada, habilitar la opción de “Enable Therming”.



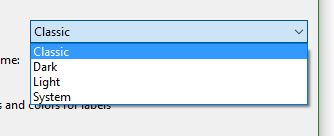
Es importante recalcar que cada vez que se modifique la apariencia, no se realizara el cambio visual completo de forma instantánea, es decir, que eclipse mandara una notificación para que reinicies el programa para realizar el 100% de los cambios.



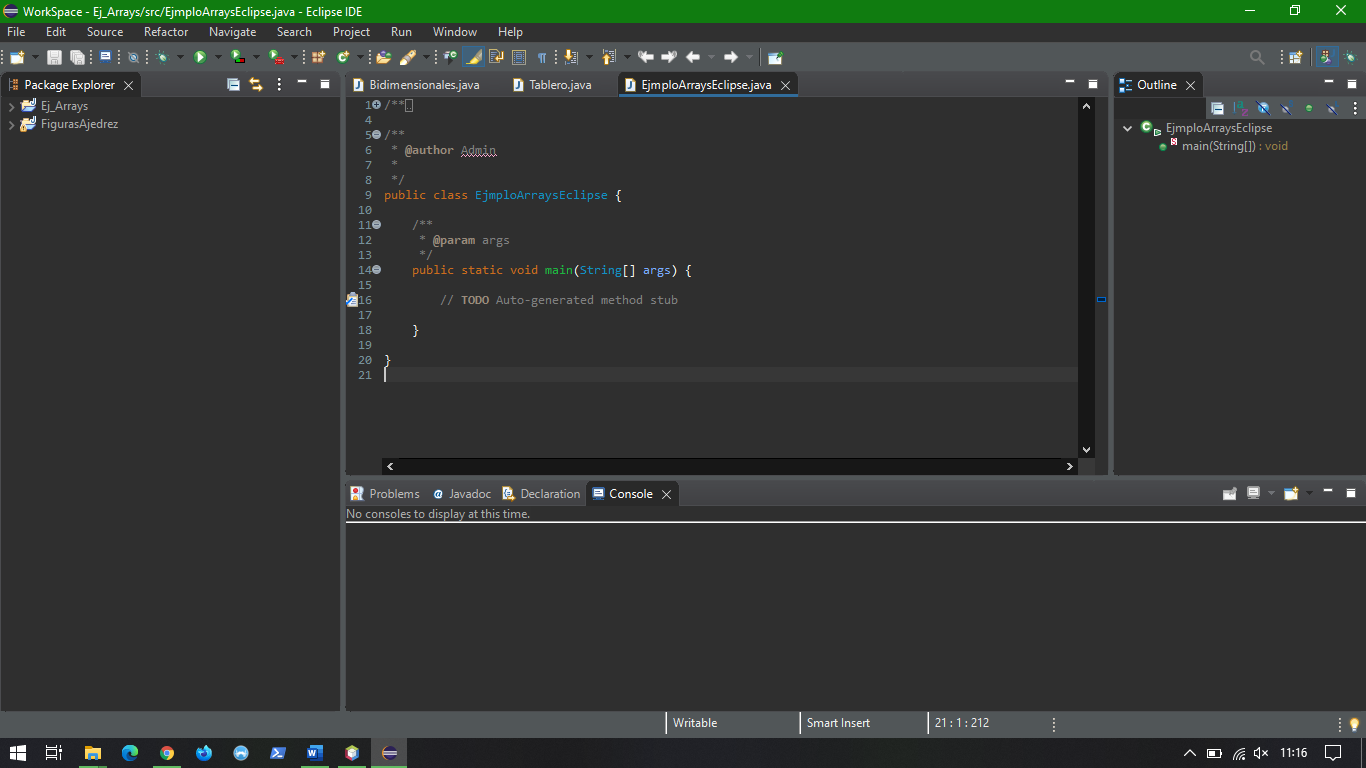
Una vez se seleccione la opción de “Enable Theming”, ese cuadro de opciones se verá modificado al siguiente:



En este momento, podremos cambiar globalmente eclipse poniendo el Theme como classic, dark,….



Una vez seleccionados los cambios no te olvides de reiniciar eclipse para que se den los cambios al completo. Mi personalización final es la siguiente:



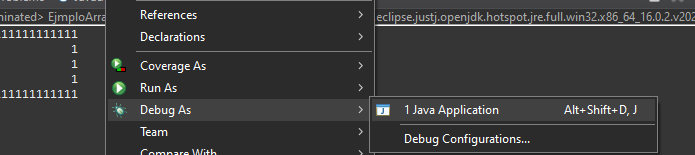
1. **USO DEL DEPURADOR DE ECLIPSE**

El depurador es una herramienta que sirve para ver paso a paso como el programa se ejecuta. Esto nos puede ayudar para poder ver errores en el código, o ver la evolución de las variables según el programa se ejecuta. Para poder usar el depurador bien, hay que utilizar los puntos de ruptura o breakpoints. Estos sirven para parar el código en la línea deseada y que a partir de ese punto las siguientes líneas solo se ejecuten cuando lo ordenemos. Los breakpoints se introducen en el código cuando se pincha encima del número de la línea del código. Es importante resaltar que los breakpoints solo se podrán poner sobre líneas que tengan una línea desarrollada, es decir, que sobre una línea vacía de código no se puede colocar un brealpoint.

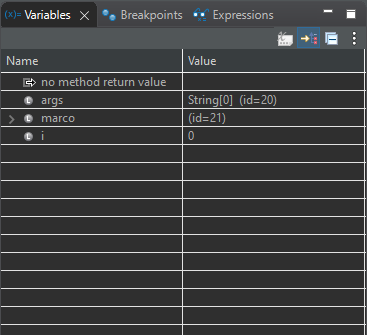


Ese punto al lado del numero indica que en la línea 9 del código hay un punto de ruptura. Para quitar los breakpoints solamente habrá que repetir la misma acción que realizamos para ponerlos. Doble click sobre el numero de la línea.

Una vez colocado el breakpoint en el código, al lanzar el depurador, el programa se ejecutará hasta la línea seleccionada, y una vez llegado a ese punto habrá que ir presionando la tecla F8 para que el código avance. Para lanzar el depurador, hay que seleccionar la opción de “debug As” que nos aparecerá cuando presionamos el botón derecho del ratón. Luego hay que seleccionar la opción de “1.Java Application”.

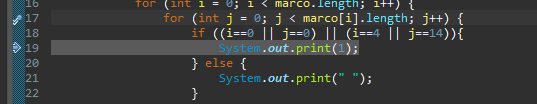


Una vez lanzado el depurador la característica mas llamativa es que se nos aparece una pestaña llamada variables. En esta pestaña se podrá ver la evolución de las variables según se avanza en el código.

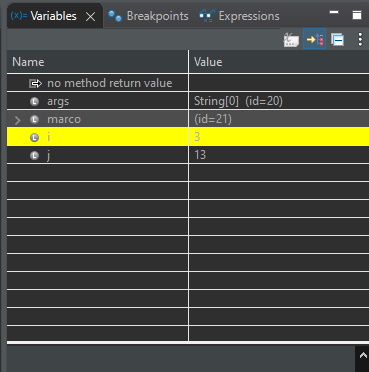
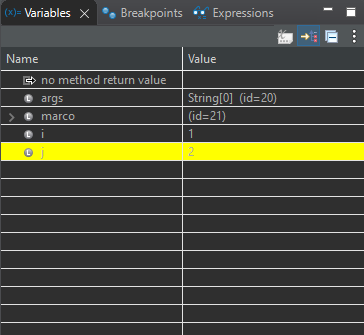


Para avanzar en el código, una vez llegado a punto de ruptura, es necesario presionar el F8. Es posible también ir avanzando con el F5, pero si se avanza con esta opción, cada vez que se llegue a un método del entorno se nos abrirá una pestaña nueva para recorrer el método por dentro. Por eso es mejor usar la opción del F8.

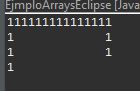
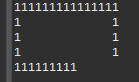
Para saber en qué línea de código nos encontramos, se observa que eclipse nos marca con una flecha la línea, además que nos la resalta por encima:



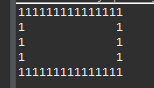
A continuación, se pondrán dos capturas en diferentes momentos de la pestaña de variables para poder observar cómo van evolucionando:



Por último, en la zona inferior la salida ira apareciendo según se ejecuten las líneas:

Para finalizar, cuando el depurador llegue al final, el programa terminara mostrando el resultado del ejercicio.



1. **SHORTCUTS EN ECLIPSE**

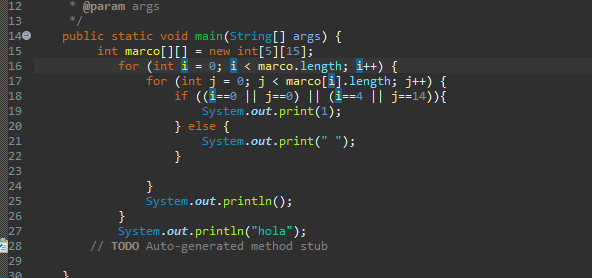
Un shortcut es un atajo de teclado para realizar una acción del entorno. A continuación, se exponen los que me resultan más útiles:

* CRTL + BARRA\_ESPACIADORA: Este shortcut sirve para autocompletar el código

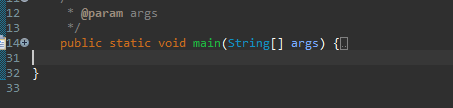
cuando se está introduciendo un JavaCodeTemplate. Si existen diferentes opciones para el JavaCodeTemplate, se abrirá un pequeño desplegable para mostrar las opciones, si no, se autocompletara solo.

* CRTL + SHIFT + C: Este shortcut sirve para comentar o des comentar código. En el caso que estén seleccionadas varias líneas de código, comentara todas ellas con las dos barras. (//…..)
* CTRL + I: Este shortcut se utiliza para indentar el código. La indentación del código sirve para organizar el código de forma visual y poder observar por ejemplo que partes están entre los for, los if, los while, …Es muy útil para poder ver si se están colocando las llaves de cierre bien y para ver donde se ejecuta cada línea del código.
* CTRL + S : Este shortcut general sirve para realizar un guardado del código. Personalmente lo uso cada vez que modifico algo del código, cuando ya lo terminé de escribir, para que de esta manera tenga las modificaciones guardadas.
* CRTL + +: Este shortcut sirve para abrir llaves que estén comprimidas en el código. Imagen explicativa después del último shortcut.
* CRTL + -: Este shortcut sirve para cerrar llaves que no estén comprimidas. Es lo contrario al shortcut anterior.

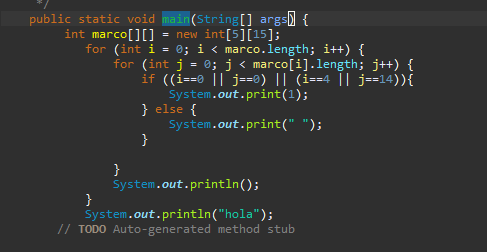
(imagen normal)



(código comprimido con CRTL+-)



(código descomprimido con CRTL++)



1. **JAVACODETEMPLATES**

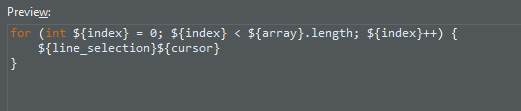
**4.1 Ejemplos de JavaCodeTemplates.**

Un JavaCodeTemplate es una abreviación para poder escribir un método del entorno. A continuación, expongo los 5 que personalmente me parece que son básicos:

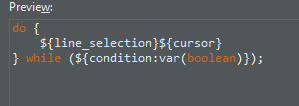
* Sysout: Este JavaCodeTemplate, nos devuelve el método de impresión siguiente



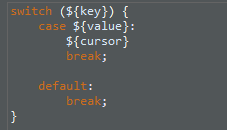
* For: Este JavaCodeTemplate, nos devuelve el método de for . Al presionar CRTL+BARRA\_ESPACIADORA, este JavaCodeTemplate nos propone una lista de posibles for. He usado como ejemplo el for correspondiente a un array, ya que este manual uso los arrays como ejemplo en el depurador. La impresión será la siguiente:



* Do: Este JavaCodeTemplate nos devuelve el método DoWhile. La impresión será la siguiente:



* Switch: Este JavaCodeTemplate, nos devuelve el método del SwitchCase. La impresión será la siguiente:



* Var: Este JavaCodeTemplate, nos devuelve el método para crear una nueva variable. La impresión será la siguiente:



* 1. **Creación de un JavaCodeTemplate.**

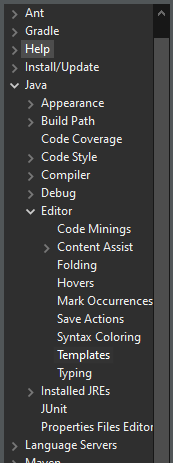
Tal y como se ha visto en el apartado anterior, los JavaCodeTemplates nos pueden ahorrar mucho tiempo a la hora de programar. Por este motivo es importante también saber como crear JavaCodeTemplates para poder acelerar nuestra escritura de código.

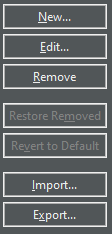
A continuación, se muestran los pasos para crear un JavaCodeTemplate para el método del Scanner. Se pasará de introducir:

Scanner teclado = new Scanner (System.in);

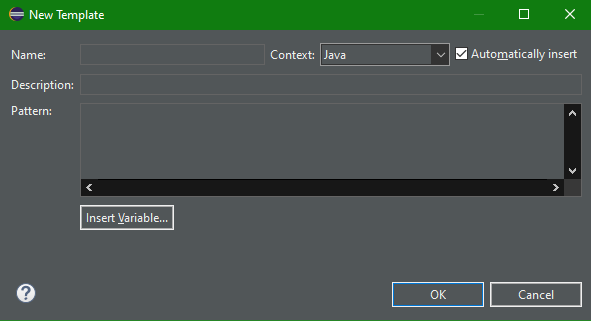
a:

scann

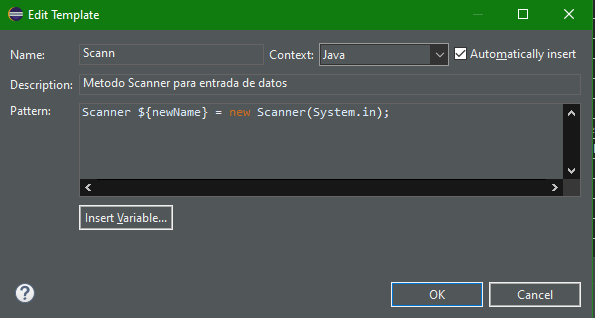
Al igual que en la personalización, hay que dirigirse a la ventana de “Preferences”. Una vez allí, se buscará el apartado “Java” en la lista de la izquierda. Dentro de la opción “Java”, hay que dirigirse a “Editor”, y dentro de editor al apartado de “Templates. Dentro de este apartado se pueden ver todas los JavaCodeTemplates que tiene el entorno. Estos se pueden modificar y también se puede habilitar/deshabilitar. A la derecha de ellos se puede ver varias opciones entre las que están: New, Edit, Remove, Resore Removed, Revert to Default, Import, y, Export. La opción que debemos de escoger para crear un JavaCodeTemplate es la de New.



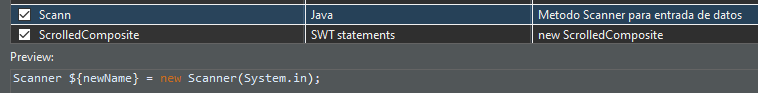
Una vez seleccionada la opción de “New”, se nos abrirá un cuadro como el siguiente:



En esta nueva ventana es donde introduciremos el nuevo JavaCodeTemplate junto con el método que queramos que se ejecute.

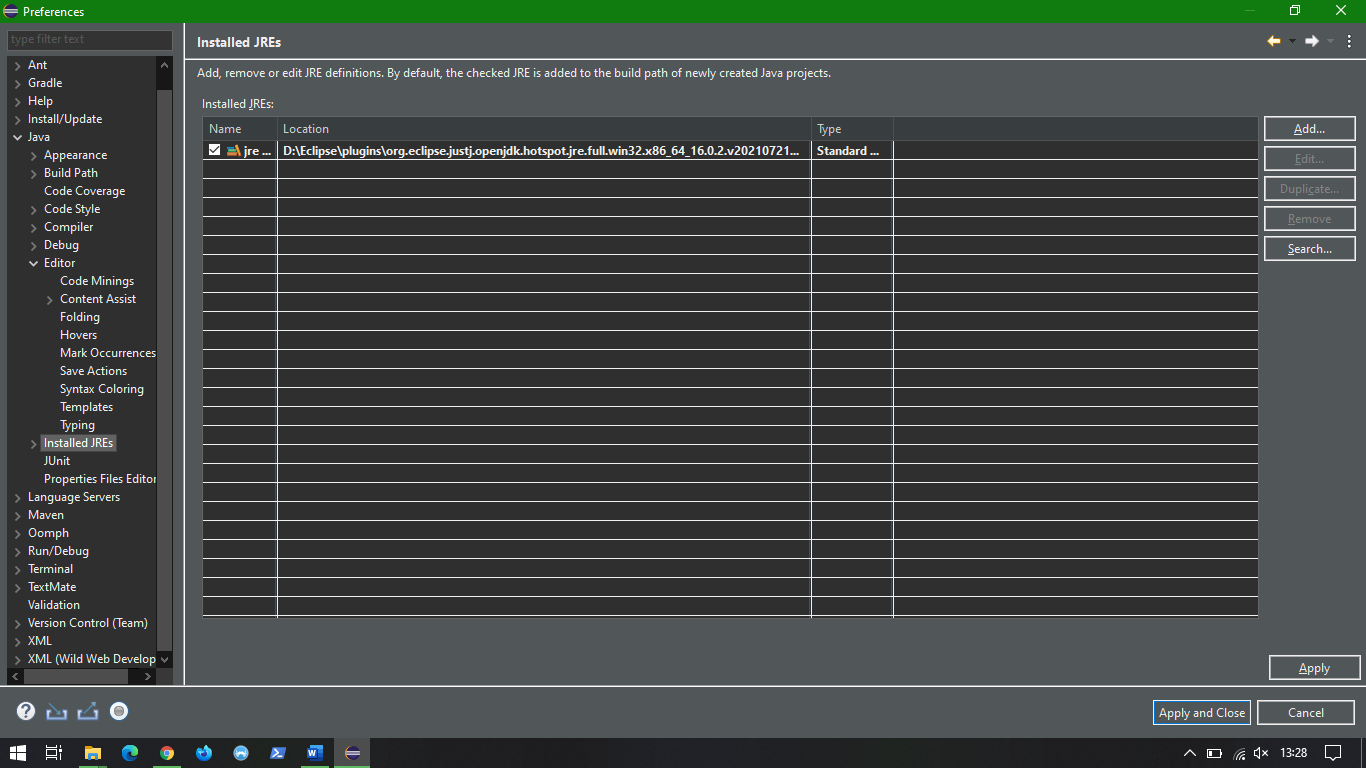


Y de esta forma se crearía el JavaCodeTemplate del Scanner.

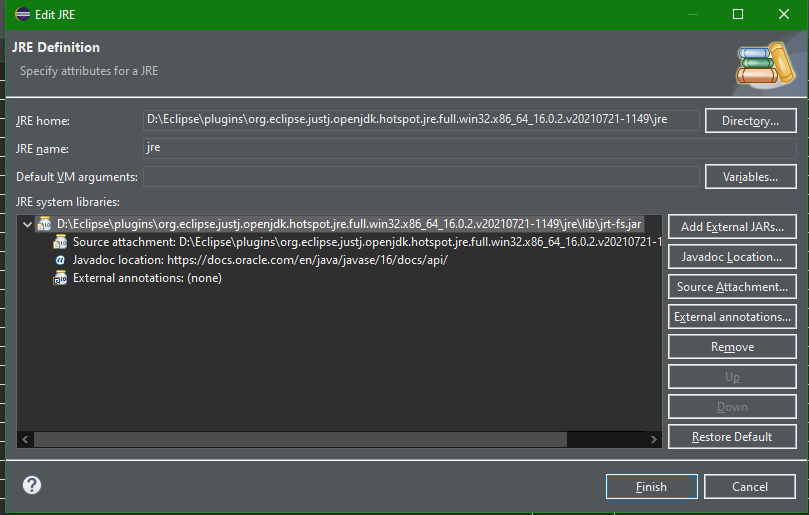


1. **ENLAZAR “JAVADOC” A UN ARCHIVO LOCAL**

Al igual que en apartados anteriores, debemos situarnos en la ventana de “Preferences”. Una ves situados en “Preferences”, debemos ir a “Java”. Dentro de “Java”, nos moveremos a “Installed JREs”.



Una vez estemos en “Installed JREs”, seleccionaremos el archivo que nos aparece y le daremos a editar. Al hacer esto, eclipse nos mostrara en pantalla la información detallada del archivo JRE.



En este punto, seleccionaremos la tercera opción del desplegable “External annotatios”.

En la ventana nueva que aparece debemos seleccionar la opción de “External location” y en el “Path” seleccionaremos el archivo.ZIP que contiene el javadoc.

