Laboratorio L08. **Debugger**

Objetivos

- Poner en práctica conceptos sobre la depuración de programas.
- Depurar programas en Java con el Debug de Eclipse.

Herramientas a utilizar

- *Eclipse*

Entrega

Fichero **.zip** con todo lo solicitado en el laboratorio L08: exportación del proyecto eclipse.

Nomenclatura de los diferentes contenidos:

- Nombre del proyecto Java: *apellidoNombre_L08.*
- La entrega debe ser individual y a través de eGela.

Contexto del Laboratorio

¿Qué es debugear/depurar?

Con el fin de ayudarnos a detectar/encontrar más fácilmente algunos tipos de errores que en algunos determinados casos podrían hasta ser difíciles de reproducir en condiciones normales, la JVM (Java Machine Virtual o Máquina Virtual de Java) nos proporciona una herramienta conocida bajo el término de degugger.

Una herramienta que surge para cubrir las necesidades de encontrar algunos tipos de errores de una forma más sencilla gracias a lo que conocemos bajo el nombre de depurar (o debugear) el código del programa. Esta herramienta tiene como finalidad otorgarnos la capacidad de realizar depuraciones de código que incluso podremos realizar desde una máquina remota. Permitiéndonos llegar a desmenuzar el código hasta el punto de llegar a ver instrucción a instrucción y de forma visible por donde pasa el flujo de nuestro programa. Además también disponemos de varias opciones que nos permitirán añadir una especie de marcas en nuestro código (los breakpoints) e ir saltando entre ellas viendo únicamente determinados puntos del programa, etc.

Algunos de estos errores hay que destacar que sin la herramienta de debugger serían difíciles de reproducir en condiciones normales, y gracias al debugger podremos obtener información muy importante sobre cómo se está ejecutando el código, útil para corregir errores y modificar su funcionamiento.

Todos los entornos de trabajo nos permiten realizar el proceso de debugear de forma muy sencilla y Eclipse, el que estamos usando nosotros, obviamente no será menos. Para ello, primeramente debemos de tener un proyecto con su correspondiente código que debugear. Veremos unos ejemplos un poco más abajo.

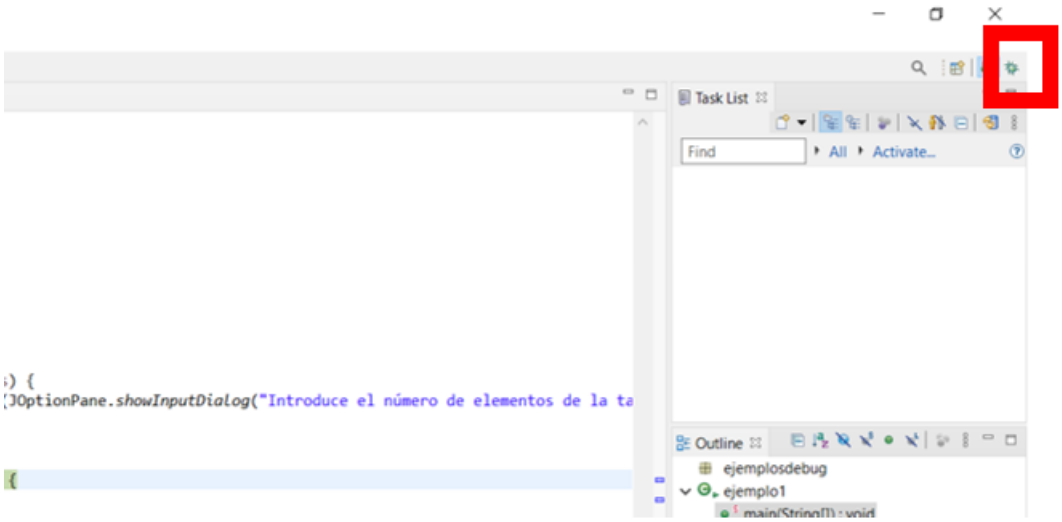
Además, es ideal para ayudarnos a entender mejor el código en casos en los que entender el código se hace sumamente difícil debido a su complejidad, el proceso de depuración, nos ayudará a ejecutar el código de una forma más interactiva observando el código fuente y las variables de ejecución muy fácilmente.

Permitiéndonos con ello el entender mejor su funcionamiento analizando detenidamente, instrucción a instrucción, nuestro código y utilizando la capacidad que nos otorga el modo de depuración (debugger) que disponemos al ejecutar nuestro código en la máquina virtual de Java.

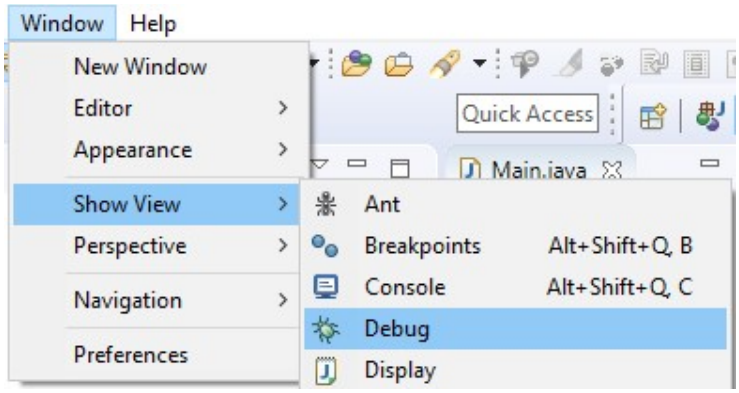
Guión del laboratorio

1. Abre Eclipse importa el proyecto que se suministra y sitúate en la clase ***Ejemplo1***.

2. Pon Eclipse en vista tipo ***Debug***. Para ello picha en el símbolo del *escarabajo verde* que aparece en la esquina superior derecha de la pantalla

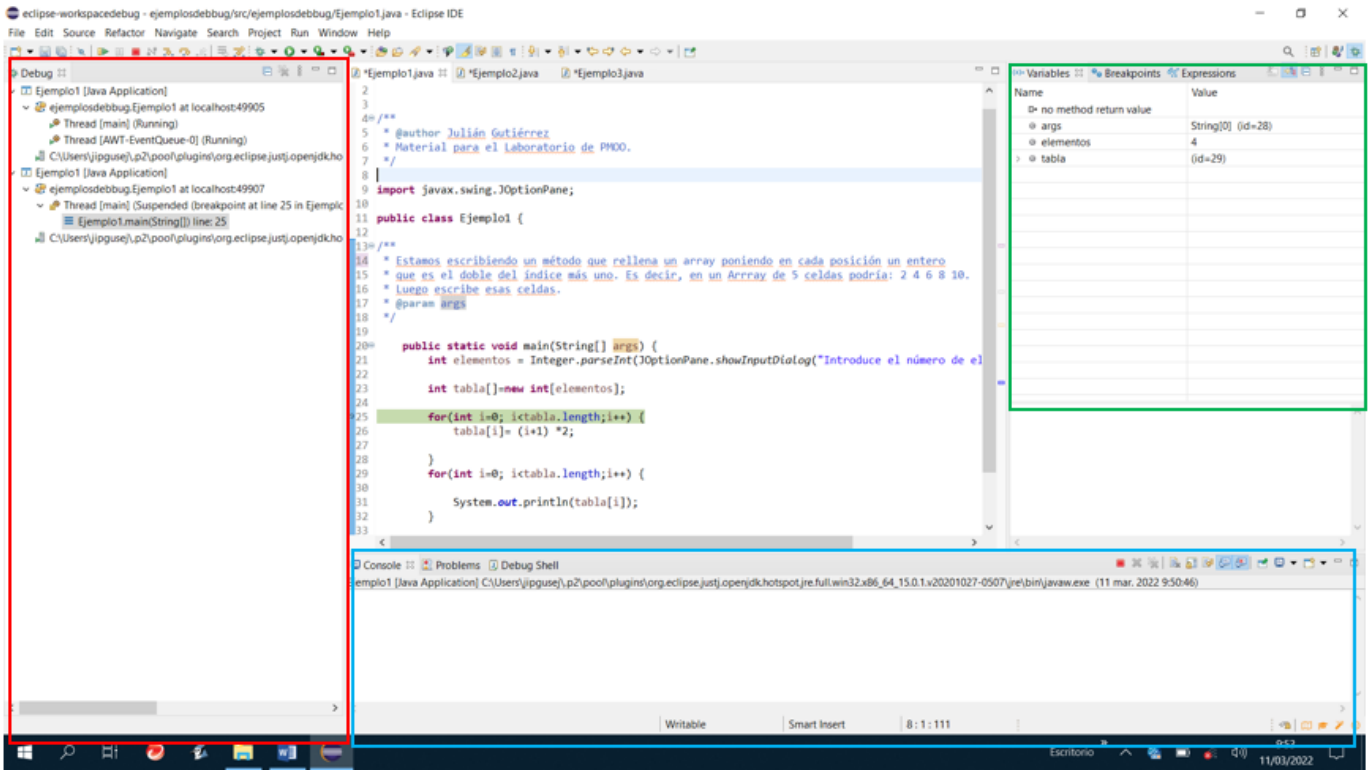


En el hipotético caso que no aparezca la ventana de Debug, podemos abrirla pulsando sobre **Windows > Show View > Debug**



Una vez que se ha abierto la vista de Debug. Aparecen nuevas ventanas: La ventana roja muestra los hilos de ejecución que están en marcha (puede haber varios a la vez), la ventana

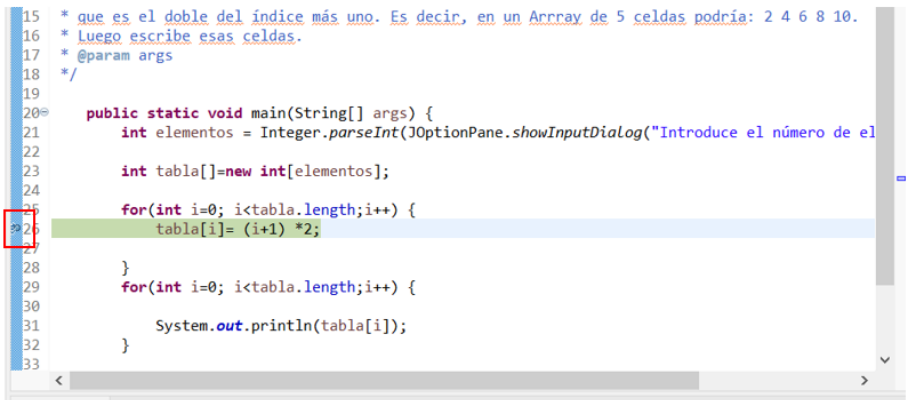
verde muestra el estado de las variables y los métodos en cada momento de la ejecución, y la ventana azul muestra la consola.



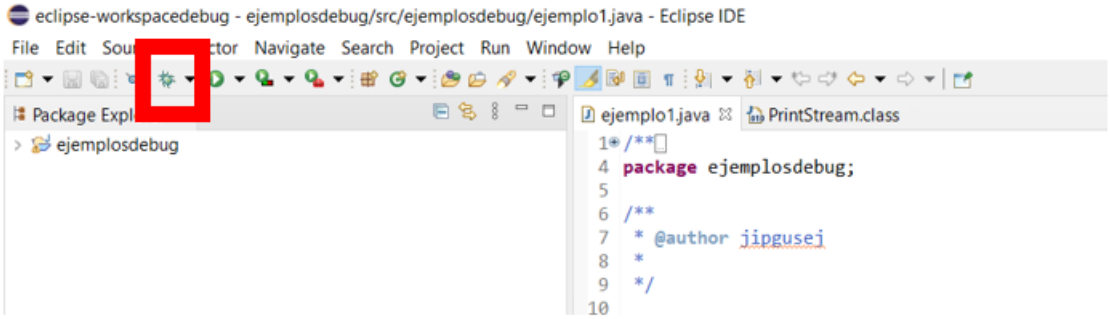
3. Ahora pondremos un ***breakpoint*** (punto de ruptura o punto de parada). Es el mecanismo que nos va a permitir detener el flujo de ejecución de un programa en una instrucción en concreto. Para crear un ***breakpoint***, tenemos varias opciones:

- Situarnos sobre la línea y pulsar la combinación de teclas **CTRL + Shift + B**
- Situarnos en la línea e ir a la opción de la barra de tareas **Run > Toggle BreakPoint**
- Pulsar **botón derecho sobre el margen de la línea** y pulsar sobre la **opción Toggle BreakPoint**
- Hacer **doble click con el botón izquierdo** justo un poco más adelante del inicio de la línea sobre el margen (sería la parte que se ve azul de la imagen inferior)

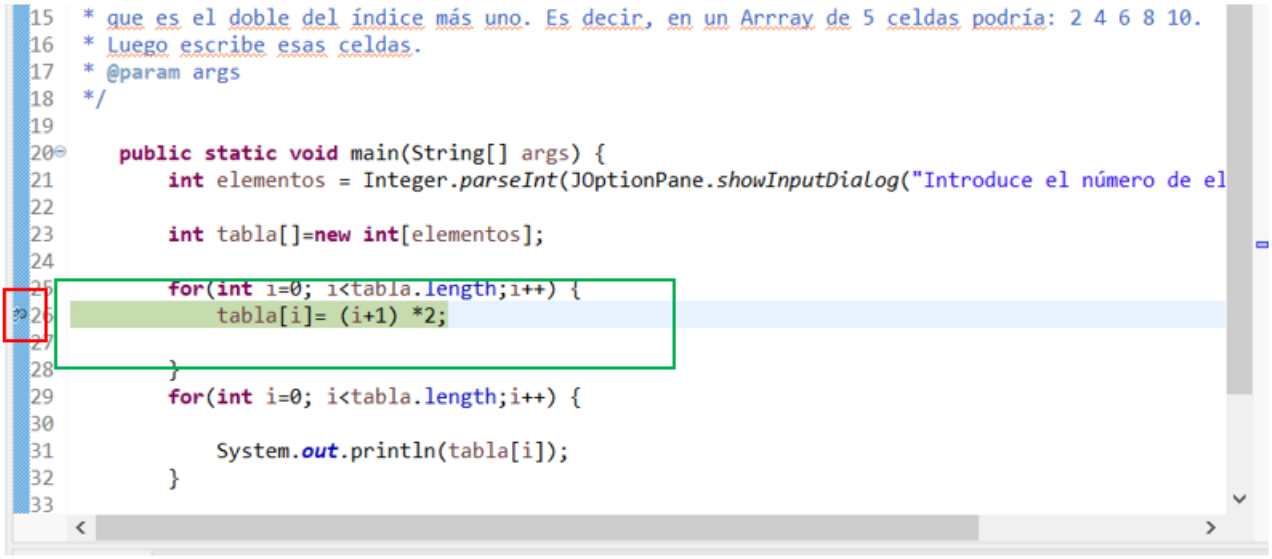
Pon un punto ***breakpoint*** sobre la línea interior de la sentencia ***for*** en el Ejemplo1:



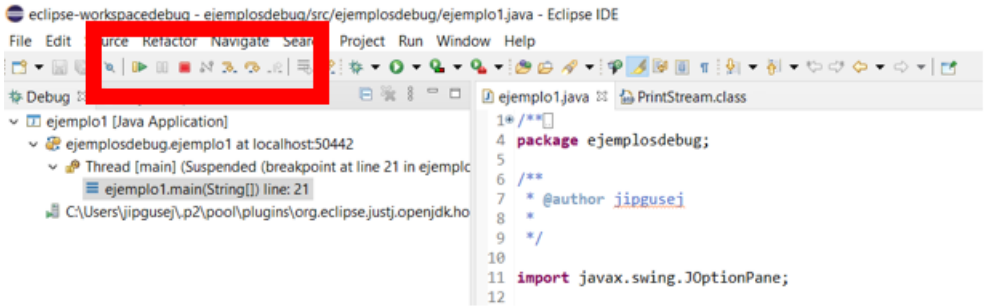
4. Una vez que hemos establecido un ***breakpoint*** es hora de comenzar el Debug. Para ello pinchamos en el botón con el escarabajo verde en la barra de tareas:



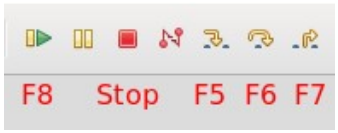
Esto hará que el programa comience a ejecutarse, después de introducir el dato que requiere, se parará en la línea que posee el ***breakpoint***. La marca en verde:



5. Es momento de comenzar a depurar. Para ejecutar la sentencia seleccionada (la que está en verde y con una flecha en el margen), necesitamos continuar avanzando con la ejecución del flujo del programa (aún no se ha ejecutado del todo). Para ello, Eclipse nos brinda un conjunto de comandos de depuración en la barra de tareas:



Los comandos se pueden ejecutar con teclas ***Fx***:



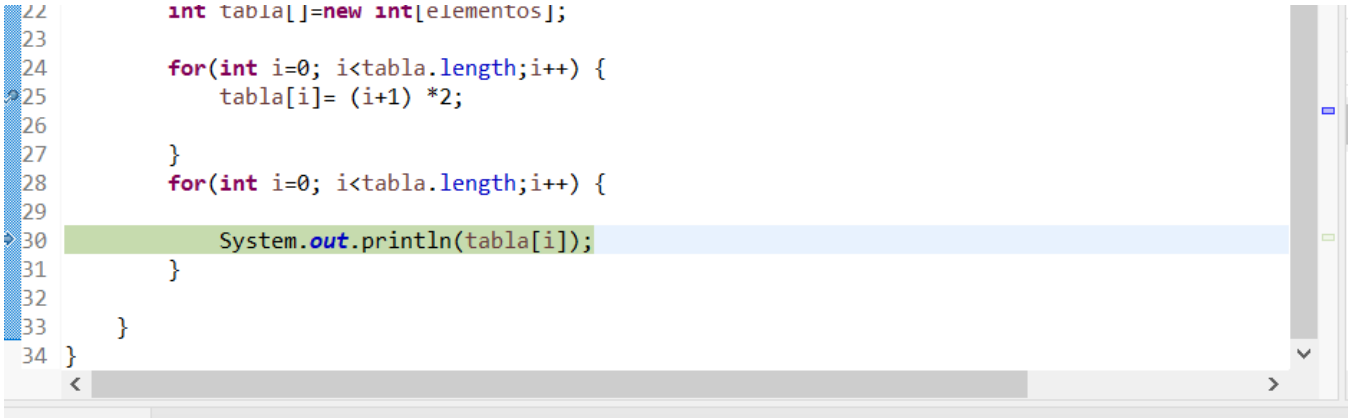
• **F5** Ejecutará la instrucción seleccionada y pasará a la siguiente. Si en la línea hay una llamada a un método saltará a analizar el código de ese método.

• **F6** Ejecutará las instrucciones sin ejecutar los métodos internos. Si pulsamos F6 pasamos a la siguiente instrucción de nuestro programa.

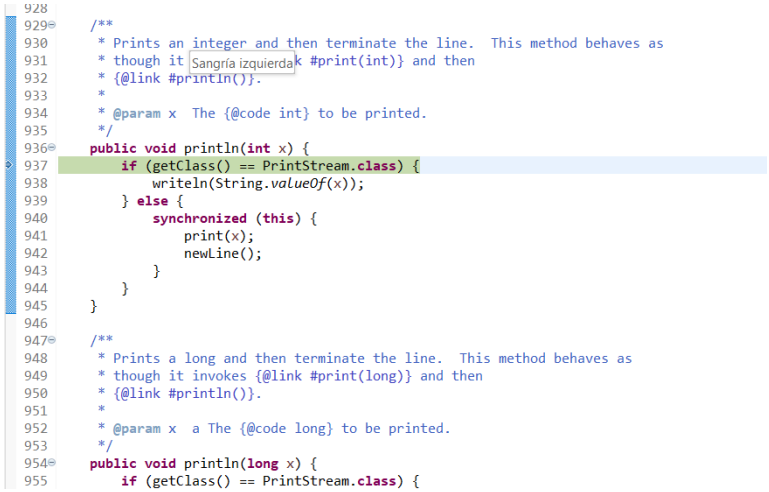
• **F7** Este comando nos permite volver al método que llama a otro método. Lógicamente cuando estamos dentro del método que recibe la llamada.

• **F8** Le indica a la JVM que reanude la ejecución de nuestro código hasta que alcance el siguiente punto de interrupción o punto de observación. Si no hay más *breakpoints*, el sistema sale del modo de depuración y ejecuta el resto del programa normalmente.

Veamos como funciona. Si en este momento pulsamos seguidamente F6, el hilo de ejecución va saltando de línea siguiendo el programa. Pulsar F6 hasta que la línea verde se sitúa sobre la sentencia System.out…..



Si en este momento pulsamos F5, el depurador se meterá en el método que está en la línea println



Si ahora pulsaramos F6 seguiríamos paso a paso este método, con F5 cada vez que encontrase un método entraría en él para depurarlo.

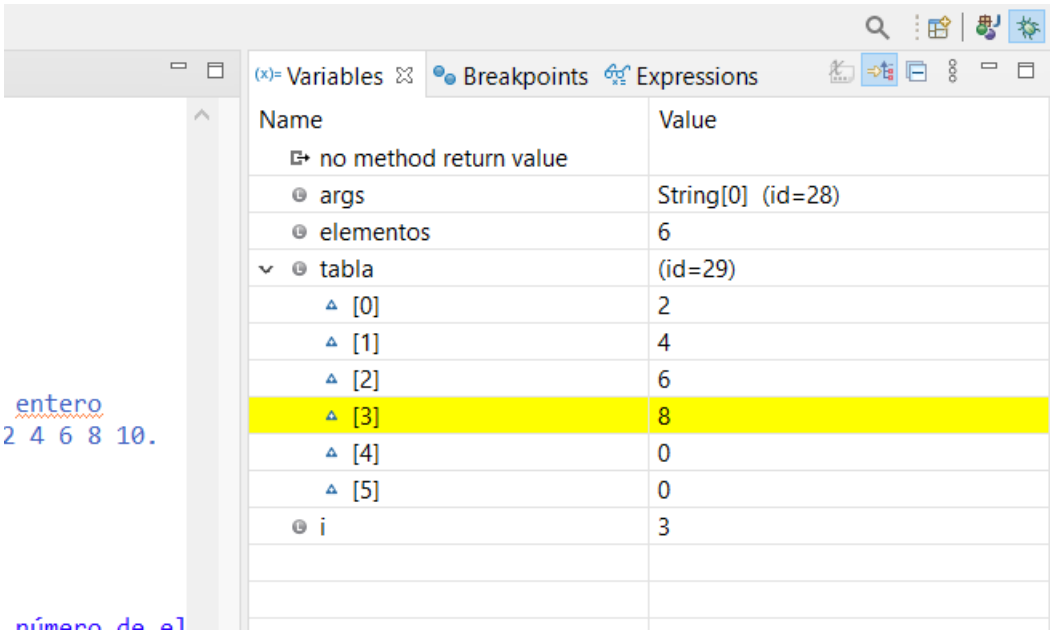
Si pulsamos F7, nos saca del método en el que estamos, en este caso nos sacará del método *println* nos sitúa en el *main* del *Ejemplo1* siguiendo la ejecución:

```
20          int elementos = Integer.parseInt(JOptionPane.showInputDialog("Introduce el número de el
21
22          int tabla[]=new int[elementos];
23
24          for(int i=0; i<tabla.length;i++) {
25              tabla[i]= (i+1) *2;
26
27          }
28          for(int i=0; i<tabla.length;i++) {
29
30              System.out.println(tabla[i]);
31          }
32
33      }
34  }
```

F8 Salta entre breakpoints. Podeís probarlo. Termina esta ejecución y arranca otro hilo de ejecución con el escarabajo verde. Verás que va saltando de ciclo a ciclo de la sentencia *for* (el *breakpoint* está en dicha sentencia).

6. Tal y como hemos visto en el punto 2., en la vista *Debugger* del Eclipse hay una ventana (en nuestro ejemplo ventana verde) en la que se pueden ver los valores que van tomando las variables a lo largo de la ejecución

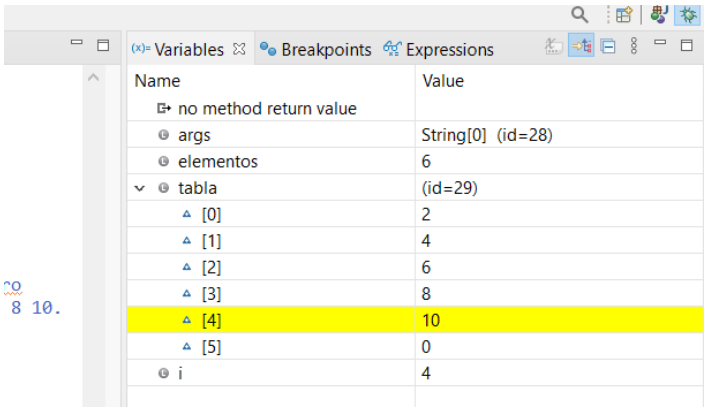
Arranca de nuevo un hilo de ejecución pulsando en el boton del escarabajo verde. Vete consultando como cambian las variables a medida que vamos pasando el programa con F6. Date cuenta que hay una flecha que permite desplegar la variable ***tabla*** (array) para ver todas sus posiciones.



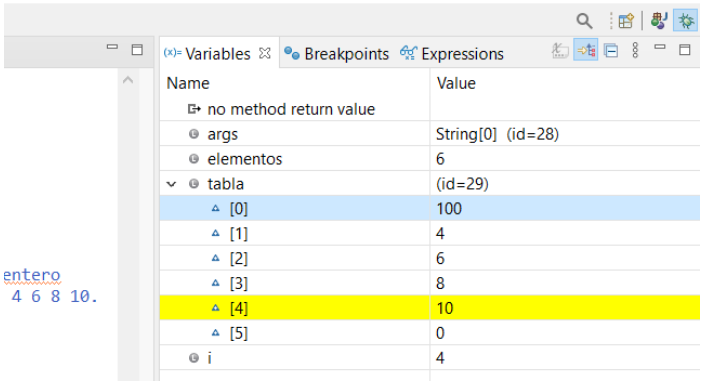
Esta opcion de consultar variables es muy interesante para poder ver, sin tener que poner trazas de impresión (System.out. print…), si nuestro programa está funcionando bien. Es muy interesante también para ver en que estado queda un *arraylist* después de practicar un **remove**, o cuando se añade un nuevo elemento, etc

7. Siguiendo con la ventana de las variables, el debugg nos permite cambiar en cualquier momento el valor de una variable para probar cosas que nos interesen.

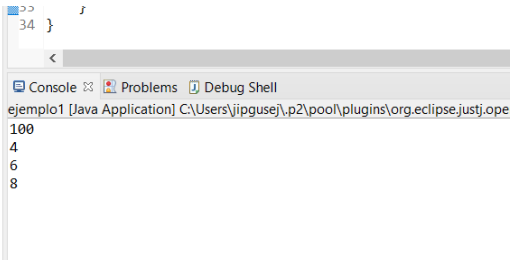
Arranca de nuevo un hilo de ejecución y haz que se den valores a varias posiones de la variable ***tabla***,



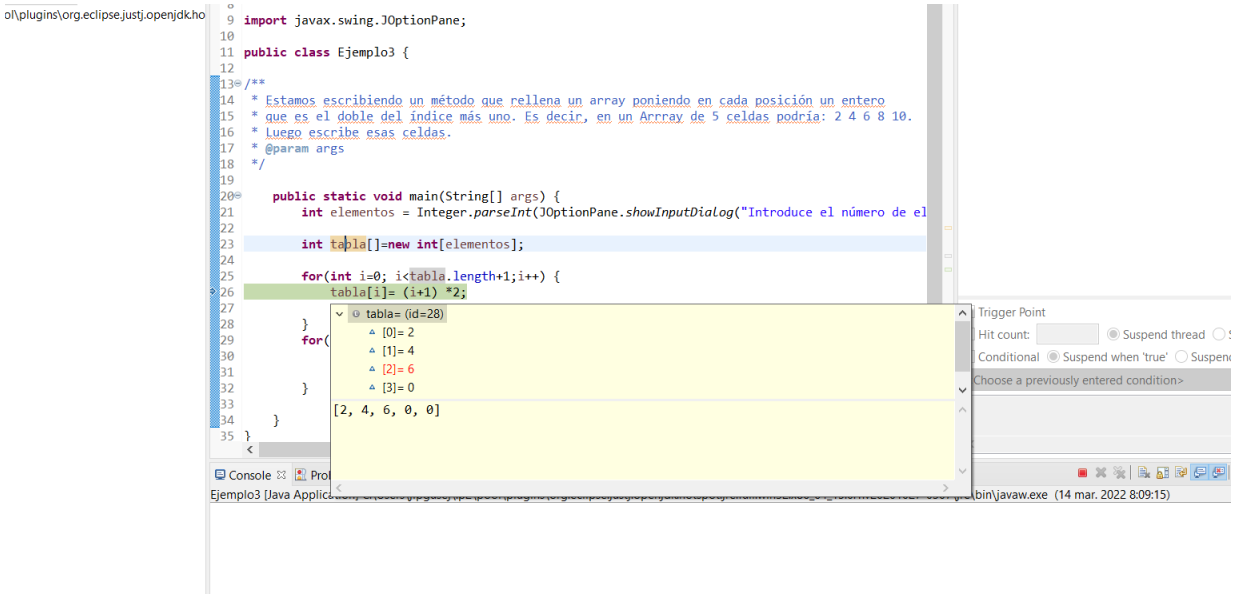
después cambia el valor de la primera posición antes de que lleguemos a la sentencia que los escribe por pantalla. Para cambiarlo ponte en la celda de la posición cero, pon el valor 100 por ejemplo y pulsa INTRO (esto es importante sino no cambia el valor).



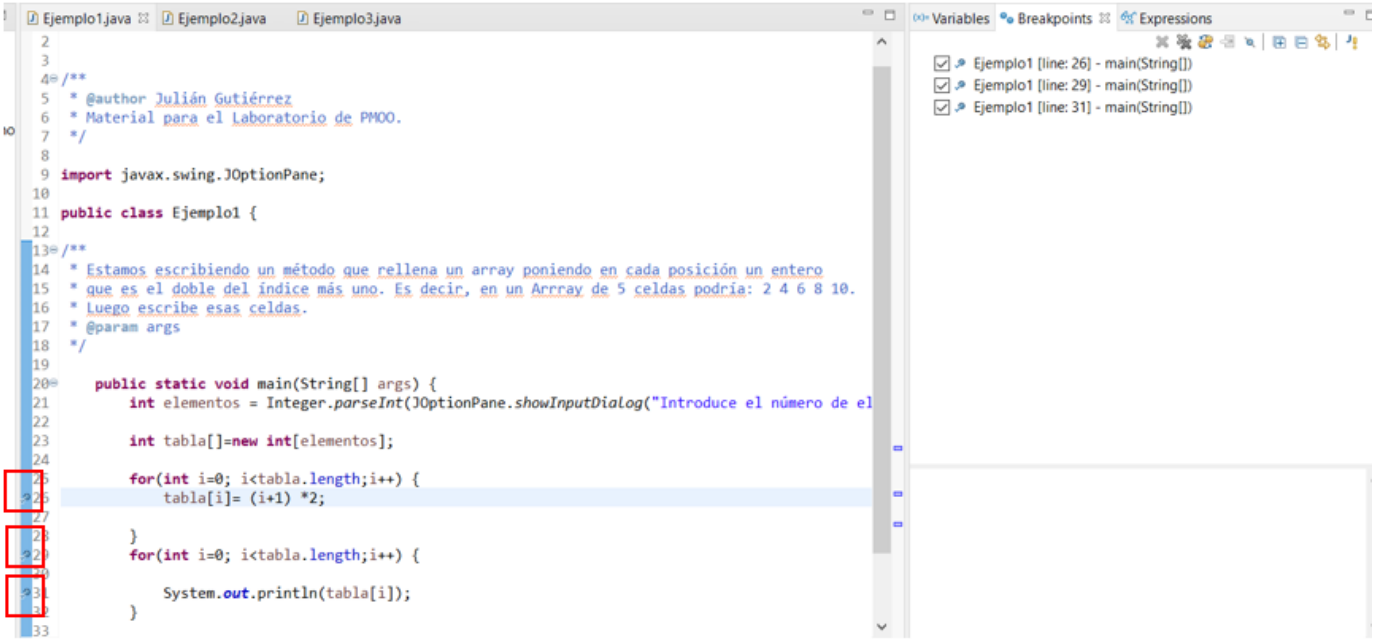
Si sigues con la ejecución, el resultado por consola será que el primer número que se imprime es 100.



8. Otra funcionalidad importante con relación las variables es la siguiente. Cuando se situa el ratón encima de una variable se muestra el valor de esa variable en el momento concreto del proceso de Debug en el que se está. Utilizar esta funcionalidad ayuda a descubrir muy rapidamente los errores en mucho programas.



9. En la misma ventana de las variables, existe una pestaña de breakpoints. Esta sirve para saber cuantos tenemos en nuestro programa, y podemos habilitarlos o desabilitarlos a nuestra necesidad. Para verlo, sitúa varios breakpoints en la clase ***Ejemplo1*** y comprueba la pestaña.



10. Ahora para poner en practica todo lo que has aprendido, ponte en la clase ***Ejemplo2***. Hay un error de programación. Debería hacer lo mismo que la clase ***Ejemplo1***. Usa el Debugger y descubre que pasa. CORRIGE LA CLASE Y ESCRIBE UN COMENTARIO SEÑALANDO EL ERROR.

11. Sigue aprendiendo. Descubre el error que presenta el ***Ejemplo3***. CORRIGE LA CLASE Y ESCRIBE UN COMENTARIO SEÑALANDO EL ERROR.

12. Exporta el proyecto y realiza la entrega.