



Universidad
Andrés Bello®
Conectar · Innovar · Liderar

FUNDAMENTOS DE PROGRAMACIÓN EN JAVA

PROGRAMACIÓN BÁSICA EN JAVA

El entorno Java para la Programación

Creando aplicaciones de consola en Java

Lo primero que se debe saber es qué tipos de programas se pueden realizar con Java. Estos son fundamentalmente tres:

- Aplicaciones de consola
- Aplicaciones de propósito general
- Applets

Las aplicaciones de consola son aquellos programas en Java que se van a ejecutar en una ventana de comandos o de Shell. Cuando se utiliza un entorno de desarrollo (IDE) como Eclipse o Netbeans, estos incluyen su propia ventana de consola para no tener que ejecutar la que viene con el sistema operativo.

Las aplicaciones de propósito general, son aquellos que pueden construirse para diversos objetivos o para cubrir diferentes necesidades, por ejemplo, un programa hecho en Java sería Netbeans o el mismo Eclipse.

Y los applets son programas creados en Java que se ejecutan dentro de un navegador como si fuera un plugin. En programación, se tiende a enseñar primero las aplicaciones por consola, debido a que no utilizan tantos elementos complejos que puedan confundir a quien se encuentra aprendiendo.

1. Primero que todo, se debe crear un nuevo fichero; esto se puede hacer desde el menú principal File ➤ New File.
2. Seleccionar como tipo de archivo un archivo vacío, en inglés “Empty Java File”.
3. Al fichero se le llamará Estructura.
4. Se ha creado un archivo vacío.
5. Comenzar a escribir código en este archivo.

La declaración `class` permite definir una clase, en la cual su nombre debe coincidir con el nombre del archivo.

La segunda línea es la declaración `main`, y en ella se indica que la parte principal del programa empieza ahí. El método `main` es el punto de entrada de la mayoría de los programas; hay excepciones, como por ejemplo los applets, que son programas que se ejecutan como parte de una página Web, y los servlets que son programas que se ejecutan en un servidor.

Las llaves también forman parte de la estructura del programa, gracias a ellas se pueden crear los bloques.

Depuración de programas usando el IDE

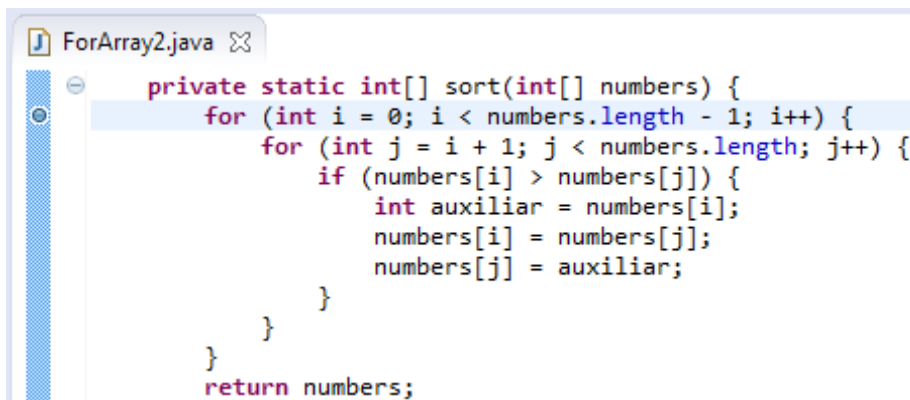
Existen muchos casos en los que se hace difícil entender lo que pasa en un programa. A fin de entender paso a paso los cambios que suceden en un programa a nivel de variables u operaciones, la respuesta está en depurar (debug) el código del programa.

La máquina virtual de Java tiene capacidades de depuración de código, incluso desde una máquina remota. En lo que respecta a Eclipse, este IDE hace que el proceso de depuración sea sencillo.

Depurar un programa es recorrerlo paso a paso, viendo el valor de las variables del programa. Esto permite al desarrollador ver exactamente qué está pasando en un programa cuando se está ejecutando una línea de código específica.

Para realizar una depuración de una clase, lo primero que se debe hacer es colocar un punto de interrupción (breakpoint) en una línea de código del programa. Se debe Hacer doble click en el margen izquierdo de la ventana donde está el código, en caso que se desee crear el breakpoint.

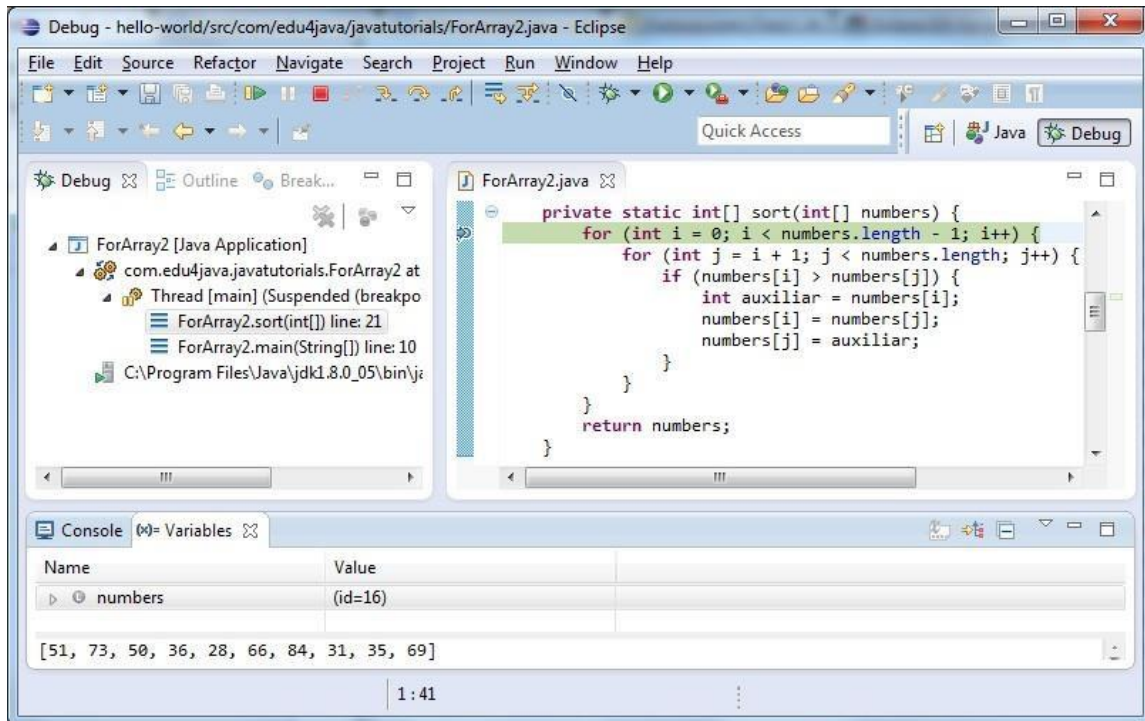
Se usará el siguiente ejemplo, el cual representa a un algoritmo de ordenamiento de un arreglo, en específico el método “Ordenamiento Burbuja” o “Bubble Sort”:



```
private static int[] sort(int[] numbers) {  
    for (int i = 0; i < numbers.length - 1; i++) {  
        for (int j = i + 1; j < numbers.length; j++) {  
            if (numbers[i] > numbers[j]) {  
                int auxiliar = numbers[i];  
                numbers[i] = numbers[j];  
                numbers[j] = auxiliar;  
            }  
        }  
    }  
    return numbers;  
}
```

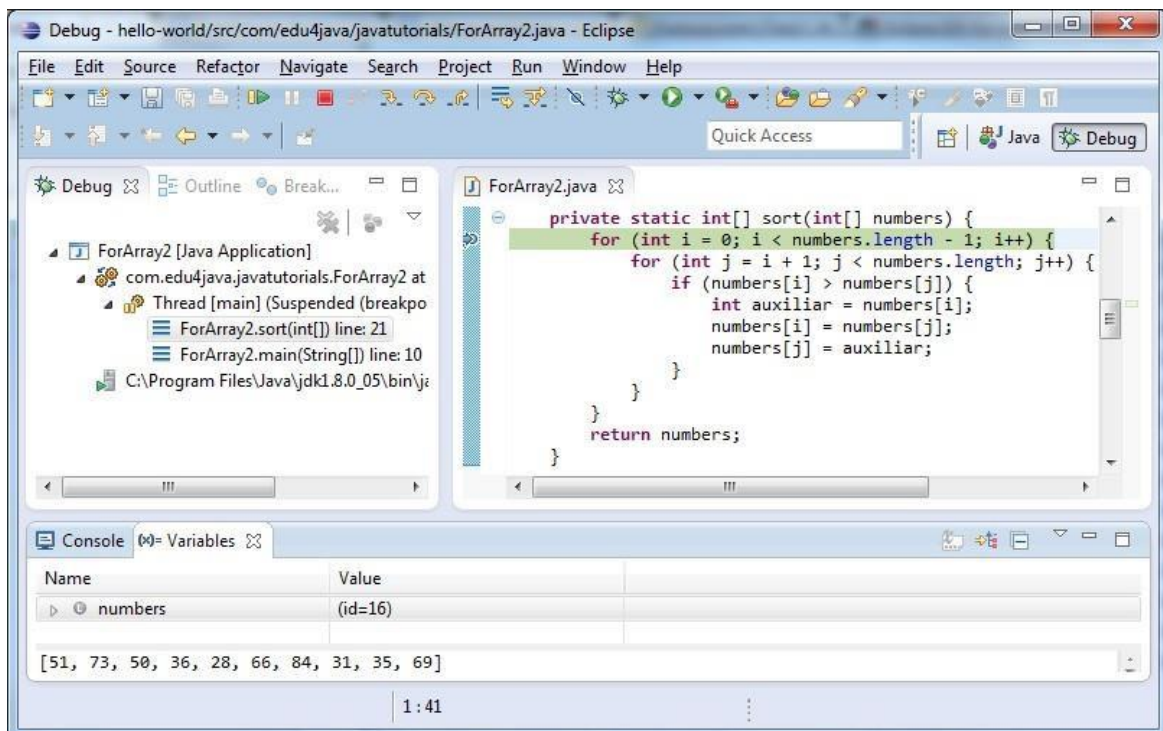
Lo siguiente es iniciar el programa con la opción depurar (debug). El programa se ejecutará normalmente hasta encontrar la línea donde se colocó el punto de interrupción. Eclipse detendrá la ejecución y solicitará permiso para cambiar a la perspectiva Debug. Se debe hacer click en la opción “Remember my decision”, y posteriormente a la opción “yes”. Realizado esto, aparecerá la perspectiva debug con la línea marcada, mostrando información sobre el estado de la ejecución.

Eclipse tiene varias formas de mostrarnos nuestros proyectos. Estas formas son llamadas perspectivas. Aunque no lo hayamos notado, hemos estado trabajando hasta ahora en la perspectiva Java.



Cuando se cambia a la perspectiva Debug, la primera impresión es confusa ya que aparecen nuevas vistas (ventanas) y cambian de posición con respecto a la perspectiva

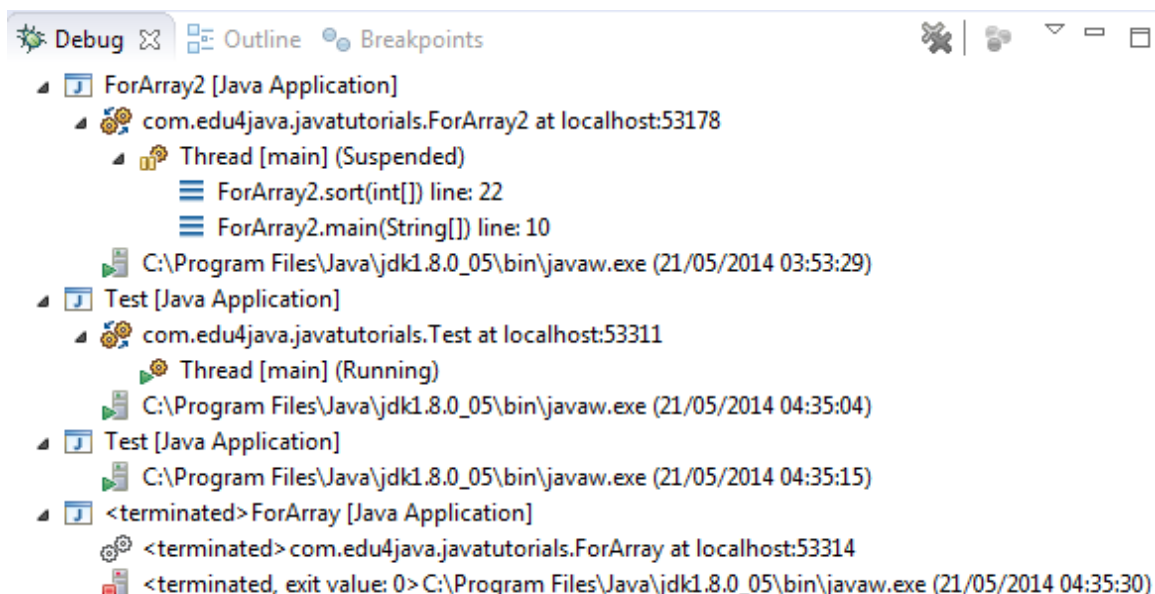
Java. Dado que se usará mucho la perspectiva Debug, se aconseja reordenar las pestañas. Arrastrando y soltando las pestañas, se podrá modificar su posición.



En la esquina superior derecha, se puede ver la barra de perspectivas, la que incluirá la perspectiva Debug seleccionada. Si se desea volver a la perspectiva Java, basta con hacer click en la opción “Java”.

Las nuevas vistas de la perspectiva debug son Debug, Variables y Breakpoint. Además, la vista de código, muestra marcada con verde, la línea donde está detenida la ejecución. El detalle de las nuevas vistas es el siguiente:

- **Vista Variables:** En esta vista se puede apreciar una línea por cada variable, campo o parámetro. Si se hace click en cada uno, se verá su contenido.
- **Vista Breakpoints:** Aquí se puede ver la lista de todos los puntos de interrupción que se tienen al momento. Es posible habilitar o deshabilitar un breakpoint con el check a la izquierda de cada uno.
- **Vista Debug:** La vista debug muestra la lista de aplicaciones que se están ejecutando, y al mismo tiempo algunas que terminaron su ejecución.



- En la imagen anterior, se puede ver que los tres primeros programas están ejecutándose y el último terminado. Los dos primeros se ejecutan en modo debug, y se puede ver el hilo de ejecución (thread). El tercero no está en debug por lo que no se ve el hilo.
- El primero tiene la ejecución suspendida en la línea 22, del método sort, de la clase ForArray2. También vemos que está detenido en el método main, línea 10 de la misma clase. Esto es lo que se conoce como “pila de ejecución”.

- La pila de ejecución se forma con las subsecuente llamadas de un método dentro de otro método. En la base está el primer método “main” desde donde se llamó al método sort.

Botones para controlar la depuración

En la barra de botones principal, al lado de imprimir, se ubican los botones que permiten manipular la ejecución de la depuración de código.



Las funciones de estos botones, por orden, son:

1. **Resume (F8):** continúa con la ejecución (hasta el próximo breakpoint).
2. **Suspend:** se puede detener la ejecución aunque no se alcance un breakpoint (muy útil cuando se ingresa a un ciclo infinito).
3. **Stop:** detiene la depuración.
4. **Step Into (F5):** se detiene en la primera línea del código del método que se está ejecutando. Si no hay método, hace lo mismo que Step Over.
5. **Step Over (F6):** pasa a la siguiente línea que se ve en la vista de código.
6. **Step Return (F7):** vuelve a la línea siguiente del método que llamó al método que se está depurando actualmente. O lo que es lo mismo, sube un nivel en la pila de ejecución, que se aprecia en la vista Debug.

Documentando el código con Javadoc

Documentar un proyecto es algo fundamental de cara a su futuro mantenimiento. Cuando se programa una clase, se debe generar documentación lo suficientemente detallada sobre ella como para que otros programadores sean capaces de usarla sólo con su interfaz. No debe existir necesidad de leer o estudiar su implementación, lo mismo que sucede cuando se desea usar una clase del API Java, momento en el que regularmente no se lee la información ni se estudia su código fuente.

Javadoc es una utilidad de Oracle para la generación de documentación de APIs en formato HTML a partir de código fuente Java. Javadoc es el estándar para documentar clases de Java. La mayoría de los IDEs utilizan javadoc para generar de forma automática documentación de clases. BlueJ también utiliza Javadoc y permite la generación automática de documentación, y visionarla bien de forma completa para todas las clases de un proyecto, o bien de forma particular para una de las clases de un proyecto.

Se debe analizar en primer lugar qué se debe incluir al documentar una clase:

- Nombre de la clase, descripción general, número de versión, nombre de autores.
- Documentación de cada constructor o método (especialmente los públicos) incluyendo: nombre del constructor o método, tipo de retorno, nombres y tipos de parámetros si los hay, descripción general, descripción de parámetros (si los hay), descripción del valor que devuelve.

Las variables de instancia o de clase no se suelen documentar a nivel de Javadoc.

Para generar la documentación de un proyecto automáticamente se debe seguir normas a la hora de realizar los comentarios dentro del mismo. Si se han respetado estas normas, en Eclipse se dispone de la opción del menú principal Project --> Generate Javadoc ..., que abre la documentación del proyecto en un navegador web. Para ver la documentación de una clase específica, se debe seleccionar el nombre de la clase en cuestión desde el menú que se desplegará.

Además de documentar las clases, todo proyecto debería tener un archivo Léeme o Readme. En Eclipse se puede acceder al Readme.txt de proyecto haciendo doble click en el nombre del archivo en el menú de proyectos. En este archivo sería adecuado incluir al menos: título del proyecto, descripción, versión, cómo arrancar el proyecto, autores e instrucciones para los usuarios.

La documentación para Javadoc debe incluirse entre símbolos de comentario, que han de empezar con una barra y doble asterisco, y terminar con un asterisco y barra simple.

```
/** * Esto es un comentario para javadoc */
```

La ubicación le indica a Javadoc qué representa el comentario: si está incluido justo antes de la declaración de clase se considerará un comentario de clase, y si está incluido justo antes de la signature de un constructor o método se considerará un comentario de ese constructor o método.

Para alimentar Javadoc se usan ciertas palabras reservadas (tags) precedidas por el carácter "@", dentro de los símbolos de comentario Javadoc. Si no existe al menos una línea que comience con @, no se reconocerá el comentario para la documentación de la clase.

En la tabla siguiente se muestran algunas de las palabras reservadas (tags), aunque hay más de las que aquí se mencionan.

Tag	Descripción	Comprende
@author	Nombre del desarrollador	Nombre autor o autores
@deprecated	Indica que el método o clase está obsoleto (propio de versiones anteriores) y que no se recomienda su uso	Descripción
@param	Definición de un parámetro de un método, es requerido para todos los parámetros del método	Nombre de parámetro y descripción
@return	Informa de lo que devuelve el método, no se aplica en constructores o métodos void.	Descripción del valor de retorno
@see	Asocia con otro método o clase	Referencia cruzada
@version	Versión del método o clase	Versión

Las etiquetas @author y @version se usan para documentar clases e interfaces. Por tanto no son válidas en cabecera de constructores ni métodos. La etiqueta @param se usa para documentar constructores y métodos. La etiqueta @return se usa solo en métodos de tipo función.

Dentro de los comentarios se admiten etiquetas HTML, por ejemplo con @see se puede referenciar una página web como link para recomendar su visita de cara a ampliar información.

Anexo: Referencias

1.- Lenguaje Java y Entorno de Desarrollo

Referencia: <http://www.itech.ua.es/j2ee/2006-2007/doc/sesion01-apuntes.pdf>

2.- Cómo escribir comentarios con la herramienta Javadoc

Referencia: <https://www.oracle.com/technical-resources/articles/java/javadoc-tool.html>

3.- Depurar un programa Java con Eclipse

Referencia: <http://www.edu4java.com/es/java/depurar-debug-programa-java-eclipse.html>