

Introducción a Eclipse

1. ¿Qué es Eclipse?

Eclipse es un entorno de desarrollo integrado (IDE) para Java y otros lenguajes de programación. Hoy en día es el entorno de desarrollo para Java más utilizado, con una cuota de mercado de aproximadamente 65%.

Eclipse está desarrollado por una comunidad de código abierto y se utiliza en varias áreas diferentes, por ejemplo, para el desarrollo de aplicaciones Java o Android. La comunidad de código abierto Eclipse cuenta con más de 200 proyectos de software libre que cubren diferentes aspectos del desarrollo de software. El IDE de Eclipse se puede ampliar con diversos componentes de software adicionales llamados *plugins*.

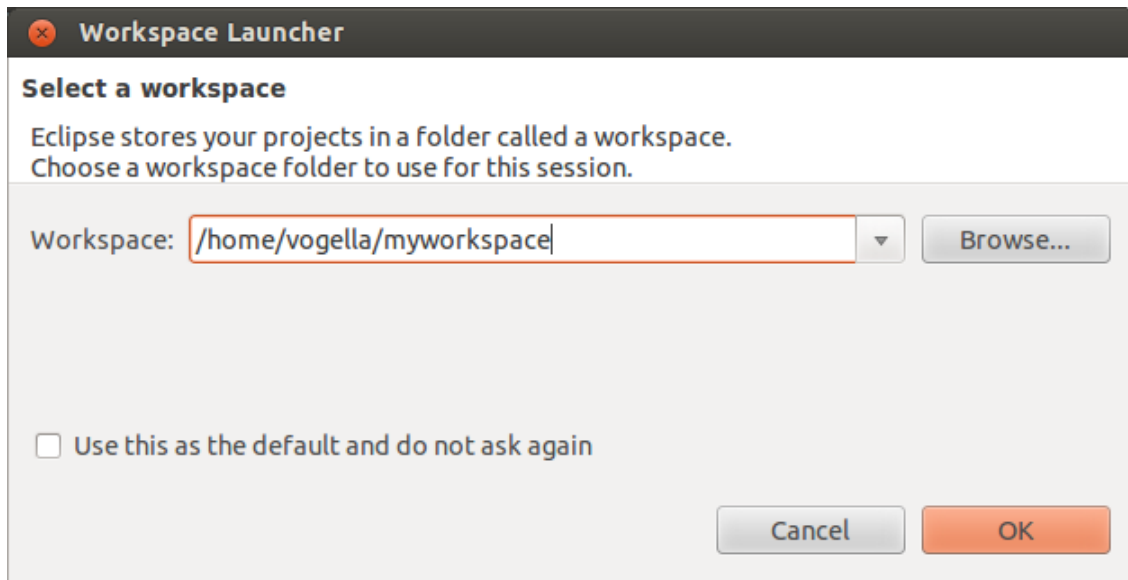
2. Instalación de Eclipse

Antes de instalar Eclipse hay que asegurarse de que tenemos un JRE o JDK instalado en el sistema, ya que Eclipse está desarrollado en Java y necesita de un entorno de ejecución Java para poder ejecutarse.

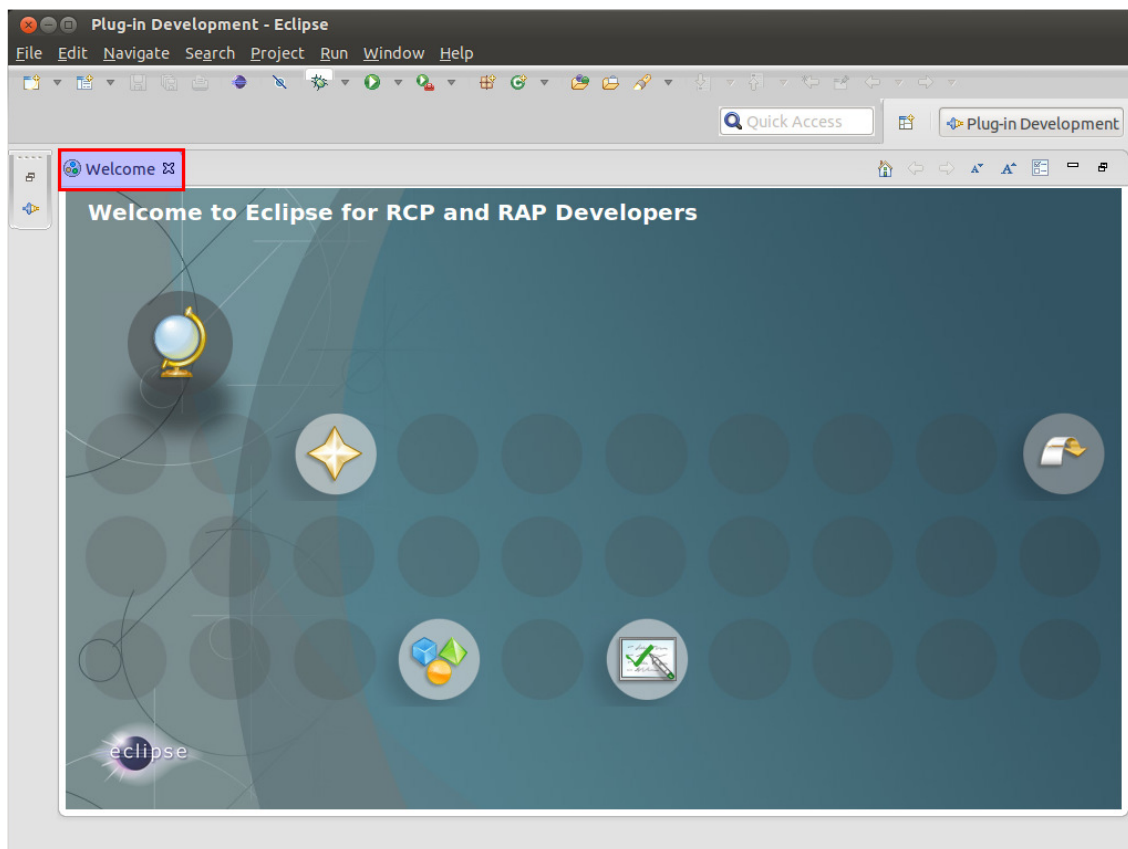
La instalación de Eclipse es muy sencilla. Podemos descargárnoslo de www.eclipse.org en forma de archivo ZIP y sólo tenemos que descomprimirlo en la carpeta donde queramos tenerlo instalado. Para ejecutarlo sólo hay que arrancar el fichero eclipse.exe (Windows) o Eclipse (Linux y Mac), sin necesidad de realizar ninguna instalación.

3. Primeros pasos

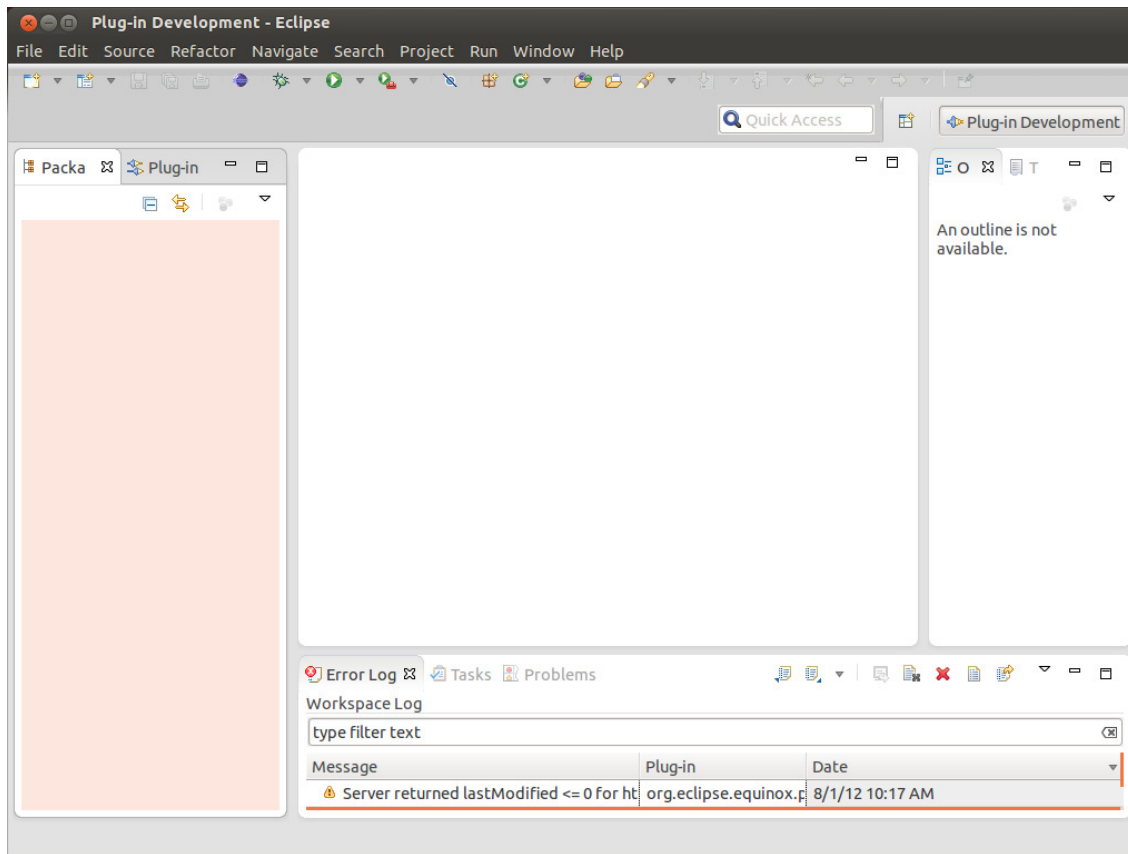
Para iniciar Eclipse, hacer doble clic en el archivo eclipse.exe/eclipse en el directorio donde se ha descomprimido Eclipse. En primer lugar el sistema pedirá un *workspace o área de trabajo* que es el lugar en el que se guardarán todos los proyectos creados en Eclipse. Hay que seleccionar un directorio vacío y pulsar el botón *Aceptar*.



Eclipse se iniciará y mostrará la página de bienvenida.



Cerramos la pantalla de bienvenida y se debería ver una pantalla similar a la siguiente.



4. Terminología importante para trabajar con Eclipse

4.1. Workspace o área de trabajo

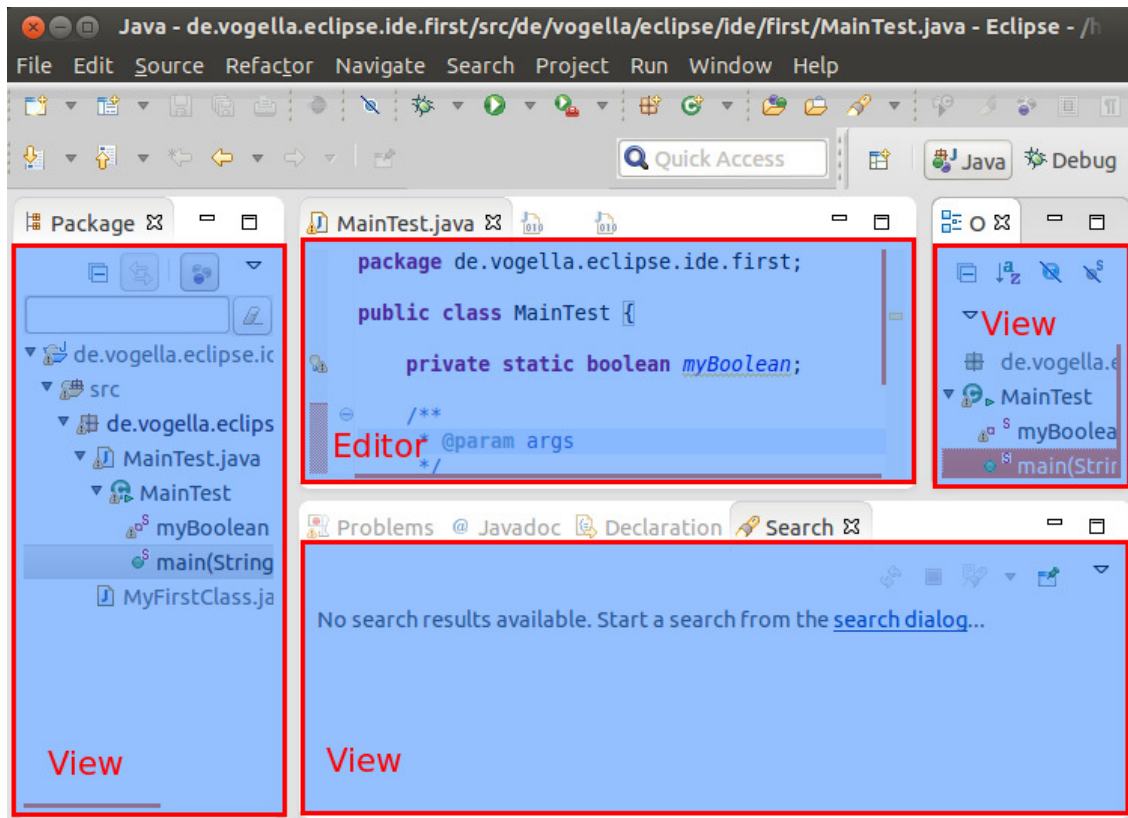
El *workspace* es la ubicación física (ruta) en la que se trabaja. Todos los proyectos, ficheros fuente, imágenes y otros objetos pueden ser almacenados y guardados en el área de trabajo. Se puede elegir el workspace durante el inicio de Eclipse o a través del menú (*File* → *Switch Workspace* → *Other*).

4.2. Proyecto

Un proyecto de Eclipse contiene los archivos fuente, configuración y binarios relacionados con una tarea determinada y los agrupa en componentes reutilizables.

4.3. Vistas y editores

La interfaz de usuario de Eclipse está dividida en *Vistas* y *editores* que permiten navegar a través de los diferentes datos y modificarlos.



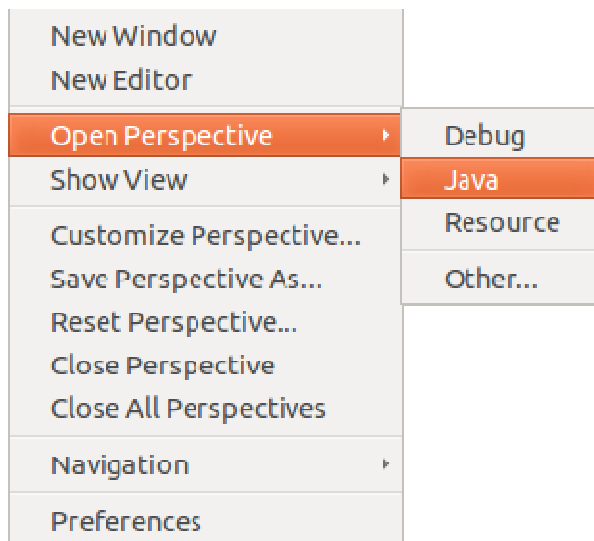
Las *Vistas* son ventanas que nos muestran distinta información sobre el proyecto en el que nos encontramos. Un ejemplo de *Vista* es el *Explorador de paquetes*, que permite navegar por los archivos de los proyectos de Eclipse. Si se cambia algún dato en el Explorador de paquetes, por ejemplo, cambiar el nombre de un archivo, el nombre del archivo se cambia directamente en el sistema de archivos.

Los *editores* se utilizan normalmente para modificar los diferentes archivos. Por ejemplo, el *Editor de Java* se utiliza para modificar los archivos fuente de Java. Los cambios en el archivo de origen se aplican una vez que el usuario selecciona el comando *Guardar*.

4.4. Perspectiva

Una *perspectiva* es un contenedor visual para un conjunto de *vistas* y *editores*. Cada perspectiva contendrá una barra de herramientas con las tareas de desarrollo correspondientes (que pueden cambiar de una perspectiva a otra). Para cambiar *la perspectiva* actual, hay que seleccionar *Window* → *Open Perspective*.

Las perspectivas principales que se utilizan en Eclipse son la perspectiva *Java* para el desarrollo de Java y la perspectiva *Debug* para depurar nuestros programas.



Se puede cambiar el contenido de una *perspectiva* añadiendo nuevas vistas en el menú *Window* → *Show View*. También se pueden modificar la barra de herramientas y distintas opciones del menú en *Window* → *Customize Perspective*. La perspectiva modificada se puede guardar a través de *Window* → *Save Perspective as*.

5. Primer programa Java

A continuación se describe cómo crear un programa de Java usando Eclipse.

5.1. Crear un proyecto

Seleccionar File-> New-> Java Project y elegir el nombre del proyecto.



El nuevo proyecto se crea y se muestra como una carpeta y su contenido se puede inspeccionar en la vista del explorador de paquetes que aparece a la izquierda.

5.2. Crear una clase Java

Hacer clic con el ratón en el proyecto y seleccionar *New* → *Class*. Elegimos el nombre de la clase (las convenciones de Java sugieren que el nombre de una clase debería comenzar con mayúscula) y si queremos podemos especificar también el paquete que la contendrá (si no especificamos nada, se guardarán dentro de un paquete por defecto).

También existen otros modificadores opcionales que pueden ser añadidos a una clase en el mismo momento de su creación (también podrían ser añadidos

manualmente en otras fases más avanzadas del proceso de desarrollo). Si se pretende que la nueva clase extienda (herede de) otra clase existente, se debería especificar la clase "padre" dentro del campo "Superclass". El botón "Browse..." es de gran utilidad a la hora de encontrar clases que sean posibles candidatas para ser extendidas. Aunque Java sólo soporta herencia única (sólo puede extenderse de una única clase) sí que es posible que una clase implemente más de una interfaz. Una vez más, el botón "Browse..." simplifica la tarea de seleccionar interfaces implementadas. Si se desea que la nueva clase contenga un método "main" (es decir, el punto inicial de ejecución del programa), puede añadirse dicho método automáticamente sólo con marcar la casilla con la opción apropiada. También pueden implementarse de esta manera los constructores de la superclase y todos los métodos abstractos heredados. Esta última opción es muy útil si se desea instanciar la clase puesto que para esto todo método abstracto debería estar implementado.

New Java Class

Create a new Java class.

Source folder:

Package:

☐ Enclosing type:

Name:

Modifiers: ☒ public ☐ default ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:

Which method stubs would you like to create?

☒ `public static void main(String[] args)`

☐ Constructors from superclass

☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments


También se pueden crear nuevos paquetes directamente a través de este diálogo. Si se introduce un nuevo paquete en este cuadro de diálogo, se crea automáticamente.

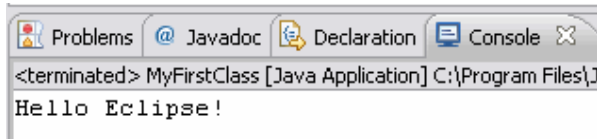
5.3. Compilar y detectar errores

En Eclipse los errores de compilación se muestran en tiempo real subrayando el fragmento de código adecuado con una línea roja. Además el entorno compila automáticamente los archivos salvados. Así pues, no será necesario pasar por el tedioso y lento proceso de compilar - observar los errores - corregir los errores.

Los errores pueden encontrarse fácilmente porque se muestran además como marcas rojas en el margen derecho del editor de código Java. Las advertencias (warnings) se muestran de la misma manera, pero con marcas amarillas.

5.4. Ejecutar el proyecto

Para ejecutar un programa dentro de Eclipse hay que seleccionar "Run > Run..." del menú principal o pulsar el botón . Eclipse ejecutará el programa Java. La salida debe verse en la vista *console*.



6. La ayuda de contenido y solución rápida

6.1. CTRL + Espacio = Autocompletar

La función de autocompletar permite obtener ayuda en distintas situaciones y resulta muy útil. Se puede invocar pulsando **Ctrl + Espacio**.

Si dentro de una clase queremos crear referencias a otras clases pero no estamos seguros de sus nombres, podemos escribir los primeros caracteres del nombre y pulsar "CTRL + Espacio". Java mostrará las posibles alternativas y podremos seleccionarlas simplemente haciendo clic con el ratón. Una vez creado el objeto, Eclipse nos ayudará mostrándonos todos los métodos y atributos que contiene.

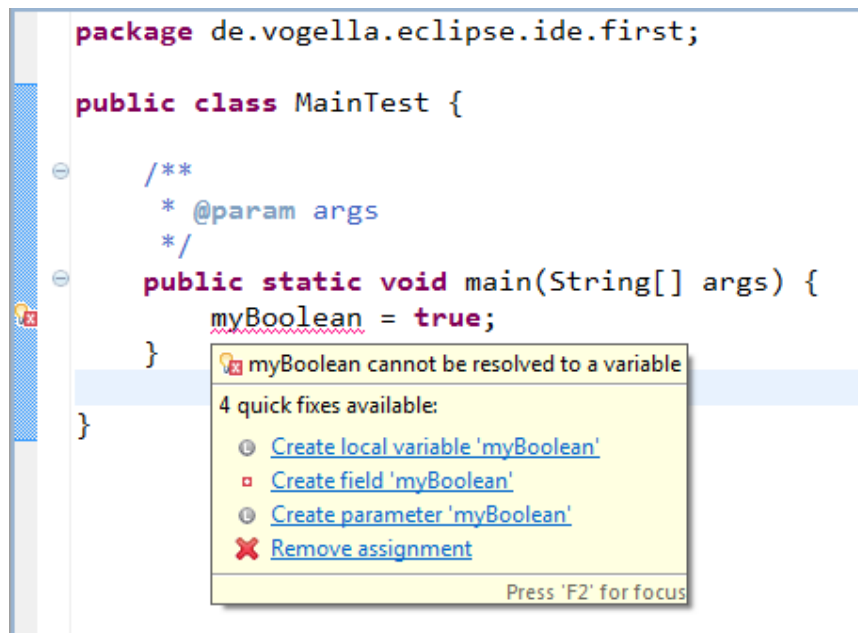
También es posible autocompletar la signatura de los constructores pulsando "CTRL + Espacio" tras escribir (o autocompletar) el nombre de la clase seguido de un signo de apertura de paréntesis, "(".

La función de autocompletar también nos facilita el uso de estructuras while, do y for. Basta con escribir "do", "while" o "for" y pulsar "CTRL + Espacio" para mostrar las posibles opciones. Si el bucle ha sido creado con el propósito de iterar sobre un array de elementos, seleccionar esta opción intentará autocompletar incluso el nombre del array.

Otro ejemplo útil de uso del autocompletado: escribir syso en el editor de un archivo fuente Java y pulsar Ctrl + Espacio . Esto reemplazará syso con System.out.println ("") .

6.2. Quick Fix

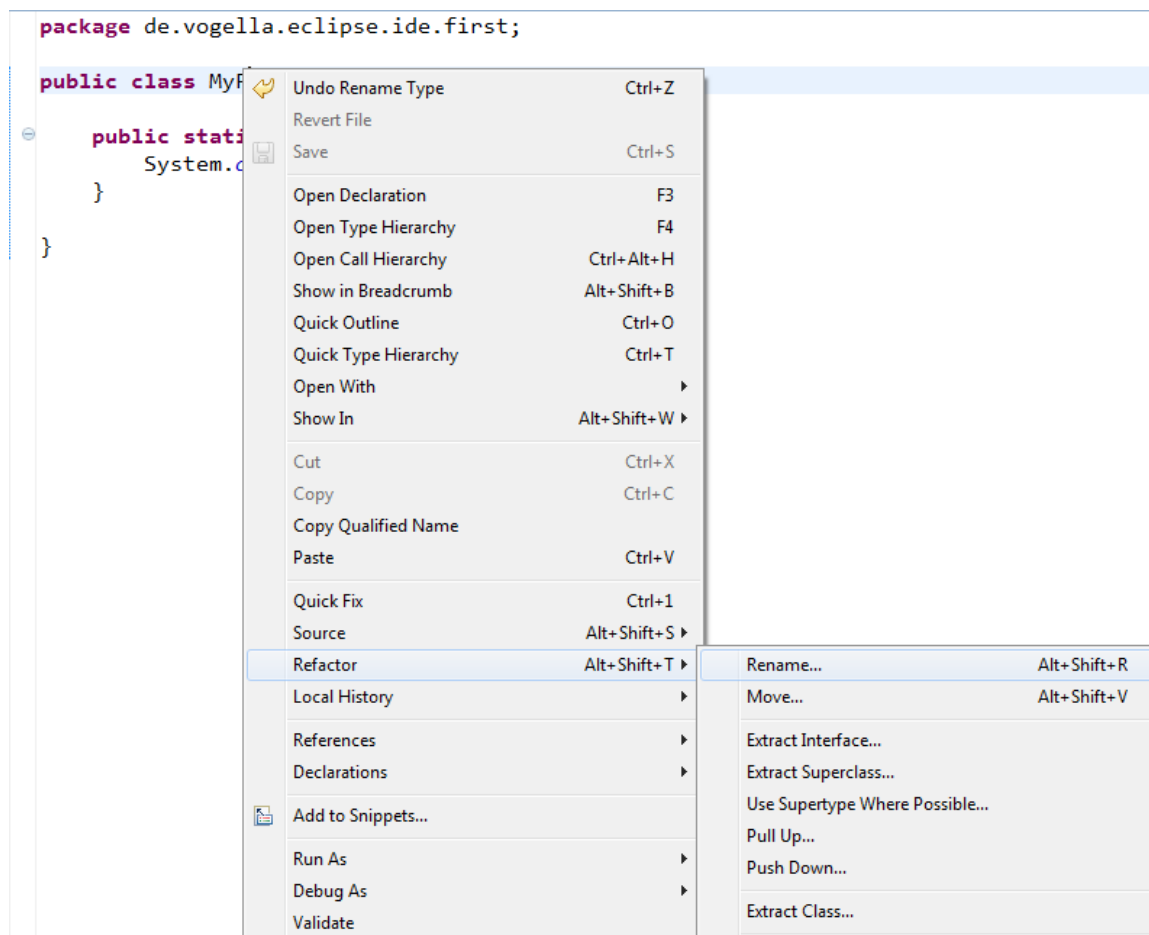
Siempre que Eclipse detecta un problema, podemos seleccionar el texto subrayado y presionar Ctrl + 1 para ver las propuestas de cómo resolver este problema.



7. Menú refactor

Refactorizar consiste en reestructurar el código sin cambiar su comportamiento. Por ejemplo cambiar el nombre de una clase Java o un método es una actividad de refactorización.

Eclipse permite varias actividades de refactorización, por ejemplo el cambio de nombre o mover. Para cambiar el nombre de una clase, seleccionamos la clase, hacemos clic con el botón derecho y seleccionamos *Refactor* → *Rename*. Eclipse se asegurará de que en todas las llamadas en el área de trabajo también se cambiará el nombre de la clase.



Referencias:

Eclipse IDE Tutorial.

<http://www.vogella.com/articles/Eclipse/article.html>

Otro Tutorial de Eclipse.

[http://dis.um.es/~bmoros/privado/bibliografia/tutorial%20eclipse%20para%20novatos%20java%20\(Pollino\).pdf](http://dis.um.es/~bmoros/privado/bibliografia/tutorial%20eclipse%20para%20novatos%20java%20(Pollino).pdf)

Java Debugging with Eclipse Tutorial.

<http://www.vogella.com/articles/EclipseDebugging/article.html>