



## ÍNDICE DE CONTENIDO

OBJETIVOS.....	1
TRABAJAR CON JAVA.....	1
Estructura general de un programa Java.....	2
Herramientas para trabajar con Java.....	2
¿QUÉ ES ECLIPSE?.....	3
INSTALACIÓN DE ECLIPSE.....	3
Obtención de Eclipse.....	3
Pasos para la instalación de Eclipse.....	4
Ejecución de Eclipse.....	4
PROGRAMAR EN JAVA CON ECLIPSE.....	4
Creación de proyectos.....	4
Crear un paquete en el proyecto.....	7
Añadir clases o interfaces al proyecto.....	7
Compilación de proyectos.....	8
Ejecución de proyectos.....	9
DEPURACIÓN CON ECLIPSE.....	9
Vista Editor.....	11
Vista Debug.....	11
Vista de Inspección.....	12
Vista Consola.....	12
UTILIDADES A LA HORA DE PROGRAMAR CON ECLIPSE.....	12
Corrector de errores.....	13
Mecanismo para completar código.....	13
Formateo del código.....	13
IMPORTACIÓN DE BIBLIOTECAS .jar A PROYECTO ECLIPSE.....	13
EXPORTACIÓN DE UN PROYECTO COMO UN ARCHIVO .ZIP.....	15
IMPORTACIÓN DE UN PROYECTO DESDE UN ARCHIVO .ZIP.....	17

## OBJETIVOS

- Aprender la instalación y el uso de Eclipse para el desarrollo de proyectos Java.
- Aprender a importar bibliotecas en archivos .jar a un proyecto Eclipse.
- Aprender a exportar un proyecto Java como un archivo .zip.
- Aprender a importar un proyecto Java desde un archivo .zip.

## TRABAJAR CON JAVA

Una de las características más sorprendentes de Java es su portabilidad. Un mismo programa Java puede ser ejecutado en un teléfono móvil con el sistema operativo Android, en un Dual Core con Windows 7 o con Linux, sin necesidad de volverse a compilar. Esto es posible gracias a la arquitectura del software que escogieron los diseñadores de Java. El equipo que diseñó Java definió en primer lugar la arquitectura de un procesador ficticio, neutral, denominado JVM (*Java Virtual Machine*). Dicho procesador tiene su propio conjunto de instrucciones, modos de direccionamiento, etc. Dado un archivo fuente en Java (un archivo de texto con extensión .java), el compilador de Java traduce dicho código a otro, que se conoce como *bytecode*, y que es un código máquina específico para el procesador JVM. En esto Java se diferencia de otros lenguajes como el C, en el cual el código máquina que se genera al compilar es específico para el procesador y el sistema operativo en el que se compila el programa (Dual Core, Sparc...)

El código máquina generado se almacena en un conjunto de ficheros con extensión .class. Los ficheros compilados de Java solamente pueden ser ejecutados en un procesador JVM. Por eso, para ejecutar un programa Java en un procesador Sparc, necesitamos un emulador de JVM para ese procesador, y para ejecutarlo en un Dual Core, necesitamos un emulador JVM para Dual Core. Así, se dice que un programa Java no se puede ejecutar directamente, sino que necesita ser interpretado. La Figura 1 muestra un esquema de esta arquitectura.

La principal ventaja de esta arquitectura es que un programa Java puede ser ejecutado en cualquier plataforma, basta con disponer de la máquina virtual correspondiente. Sin embargo, el tiempo de ejecución es mayor que si el programa se ejecutara directamente sobre el procesador real.

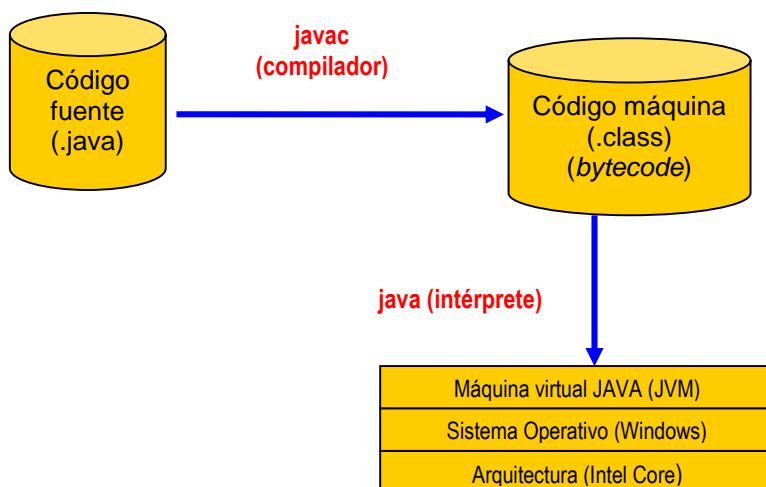


Figura 1. Arquitectura de un programa Java sencillo

## Estructura general de un programa Java

Teniendo en cuenta la arquitectura de la

Figura 1, los programas escritos en Java deberán tener la siguiente estructura:

- En nuestros proyectos al menos debe haber una clase que contenga un método `main()`.
- Habrá una serie de clases de usuario específicas de la aplicación que se esté desarrollando.

Los archivos fuente tendrán extensión `.java`, mientras que los compilados tendrán extensión `.class`. Un archivo `.java` podrá contener más de una clase, pero solamente una de ellas puede ser `public`. El nombre del fichero debe coincidir con el de la clase `public`. Es decir, que si un fichero tiene una clase pública que se llama `PuntoImpl`, su archivo se debe llamar `PuntoImpl.java`. Es importante el uso de mayúsculas y minúsculas (note que el archivo `PuntoImpl.java` comienza por mayúsculas, al igual que la clase).

Por lo tanto, una aplicación Java estará formada por varios archivos `.class` y se ejecutará mediante el nombre de la clase que contiene el método `main()`. Las clases de Java se agrupan en paquetes (*package*). Un paquete es una librería de clases. Si las clases no se definen como pertenecientes a un paquete, se utiliza un paquete por defecto que es el directorio activo.

En caso de que necesite utilizar clases definidas en otros paquetes, deberá importar los mismos. La importación de paquetes se realizará al principio del archivo fuente mediante una sentencia `import`.

## Herramientas para trabajar con Java

Hay muchos programas comerciales y de libre distribución para desarrollar programas con el lenguaje Java. Las únicas herramientas oficiales son las que proporciona la empresa Oracle de forma gratuita bajo el JDK (*Java Development Kit*).

El JDK está compuesto por un conjunto de programas y librerías que permiten desarrollar, compilar y ejecutar programas en Java. Además del JDK, Oracle también distribuye una versión reducida del mismo destinada a ejecutar código (y que, por lo tanto, no permite la compilación) llamada JRE (*Java Runtime Environment*).

Los IDE's (*Integrated Development Environment*) son entornos de desarrollo integrados que permiten escribir, compilar y ejecutar código Java con una sola aplicación. Estos entornos permiten desarrollar aplicaciones de forma más rápida, incorporando, en muchos casos, librerías con componentes ya desarrollados. En los laboratorios, el IDE que se utilizará es Eclipse, con su *plugin* JDT para desarrollo de programas Java. Un *plugin* es un programa de ordenador que se utiliza para expandir la funcionalidad de otro programa de ordenador de forma modular sin afectar a la funcionalidad original que ya proporciona este último.

## ¿QUÉ ES ECLIPSE?

Eclipse es un proyecto de código abierto cuyo principal objetivo es proporcionar una plataforma de desarrollo abierta e independiente de los fabricantes de software además de una serie de *frameworks* para construir software. Aunque inicialmente Eclipse fue un proyecto auspiciado por IBM, en la actualidad hay un consorcio de empresas, conocido como *Consortio Eclipse*, que es independiente de IBM, y al que pertenecen empresas como HP, QNX, Intel, IBM, SAP, Fujitsu, Hitachi, Novell, Oracle, Palm, Ericsson, RedHat y también universidades e institutos tecnológicos.

Se puede considerar que Eclipse es un entorno de desarrollo integrado o IDE (*Integrated Development Environment*) sobre el que se pueden montar mediante *plugins* herramientas de desarrollo para cualquier lenguaje de programación. Esta arquitectura de *plugins* permite, además de poder trabajar con varios lenguajes de programación en el mismo IDE, utilizar otras herramientas que ayuden en el proceso de desarrollo como editores visuales de interfaces, herramientas de desarrollo UML, herramientas de reingeniería,...

Dentro del proyecto Eclipse se engloba tanto el desarrollo del IDE como el de alguno de los *plugins* más importantes, como son los *plugins* JDT (para el lenguaje Java) o CDT (para los lenguajes C y C++).

La *Plataforma Eclipse* está desarrollada completamente en Java y se apoya en la librería SWT para la interfaz gráfica de usuario. Esta librería es nativa, lo que quiere decir que aprovecha los elementos de la interfaz de usuario propios del sistema operativo, lo que hace que tengan el “*Look and Feel*” del sistema y que las interfaces de usuario sean más rápidas y fluidas. Por el contrario, hay que disponer de una librería SWT específica para cada sistema operativo.

## INSTALACIÓN DE ECLIPSE

### Obtención de Eclipse

El IDE de Eclipse se puede obtener en la página oficial del proyecto Eclipse (<http://www.eclipse.org>). Vaya a la sección de descargas (pulsando el botón *Download*) y descárguese el archivo `eclipse-java-mars-R-win32.zip`. En la sección de descargas puede ver distintas versiones de Eclipse. Escoja la versión Eclipse IDE for Java Developers (Figura 2). Trabajaré con Eclipse Mars (la versión que actualmente está en distribución).

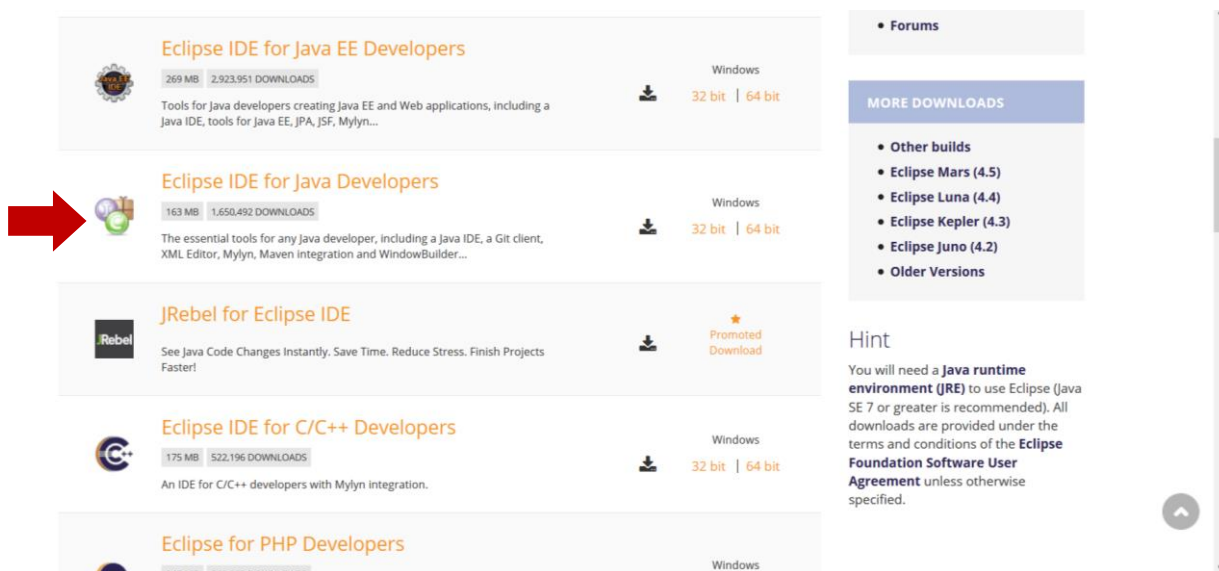


Figura 2. Enlaces de la sección *Downloads* del sitio web [www.eclipse.org](http://www.eclipse.org)

Como Eclipse está escrito en Java, es necesario que en su sistema haya instalado un JRE (*Java Runtime Environment*). Si no tiene ningún JDK instalado en su sistema, vaya a la página de Oracle y descargue e instale uno antes de instalar Eclipse. La versión con la que vamos a trabajar es la 1.8. Para ello acceda a la página de descargas de Oracle (<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>) (Figura 3). La versión que está actualmente en distribución es *Download Java SE Development Kit 8u60*.

Java EE

Java ME

Java SE Support

Java SE Advanced & Suite

Java Embedded

Java DB

Web Tier

Java Card

Java TV

New to Java

Community

Java Magazine

### Java SE Development Kit 8 Downloads

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications, applets, and components using the Java programming language.

The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

See also:

- Java Developer Newsletter (tick the checkbox under Subscription Center > Oracle Technology News)
- Java Developer Day hands-on workshops (free) and other events
- Java Magazine

JDK 8u60 Checksum

#### Java SE Development Kit 8u60

You must accept the [Oracle Binary Code License Agreement for Java SE](#) to download this software.

☐ Accept License Agreement
 ☒ Decline License Agreement

Product / File Description	File Size	Download
Linux ARM v6/v7 Hard Float ABI	77.69 MB	<a href="#">jdk-8u60-linux-arm32-vfp-hflt.tar.gz</a>
Linux ARM v8 Hard Float ABI	74.64 MB	<a href="#">jdk-8u60-linux-arm64-vfp-hflt.tar.gz</a>
Linux x86	154.66 MB	<a href="#">jdk-8u60-linux-i586.rpm</a>
Linux x86	174.83 MB	<a href="#">jdk-8u60-linux-i586.tar.gz</a>
Linux x64	152.67 MB	<a href="#">jdk-8u60-linux-x64.rpm</a>
Linux x64	172.84 MB	<a href="#">jdk-8u60-linux-x64.tar.gz</a>
Mac OS X x64	227.07 MB	<a href="#">jdk-8u60-macosx-x64.dmg</a>
Solaris SPARC 64-bit (SVR4 package)	139.67 MB	<a href="#">jdk-8u60-solaris-sparcv9.tar.Z</a>
Solaris SPARC 64-bit	99.02 MB	<a href="#">jdk-8u60-solaris-sparcv9.tar.gz</a>
Solaris x64 (SVR4 package)	140.18 MB	<a href="#">jdk-8u60-solaris-x64.tar.Z</a>
Solaris x64	96.71 MB	<a href="#">jdk-8u60-solaris-x64.tar.gz</a>
Windows x86	180.82 MB	<a href="#">jdk-8u60-windows-i586.exe</a>
Windows x64	186.16 MB	<a href="#">jdk-8u60-windows-x64.exe</a>

Java SE

Java EE and Glassfish

Java ME

Java Card

NetBeans IDE

Java Mission Control

#### Java Resources

Java APIs

Technical Articles

Demos and Videos

Forums

Java Magazine

Java.net

Developer Training

Tutorials

Java.com

October 25 - 29, 2015  
San Francisco, U.S.

[Register Now](#)

Figura 3. Página de descarga de Java en el sitio web de Oracle

Seleccione la plataforma en la que instalará JDK. Dependiendo del sistema operativo y del procesador del ordenador en el que vaya a instalar el JDK, tendrá que escoger una opción u otra. Por ejemplo, si quiere instalar JDK en un ordenador con sistema operativo Windows y un procesador de la familia i586, tendrá que escoger la opción Windows x86. Pulsando en el enlace asociado puede descargar el archivo `jdk-8u60-windows-i586.exe`. Recuerde que tendrá que aceptar que usted está conforme con la licencia de uso que le proporciona Oracle.

### Pasos para la instalación de Eclipse

- Si no tiene instalado el JDK, ejecute el archivo `jdk-8u60-windows-i586.exe`. Como resultado de la instalación debe tener en el directorio Archivos de programa\Java dos directorios nuevos, uno llamado `jre1.8.0_60` y otro llamado `jdk1.8.0_60`.
- Descomprima el archivo `eclipse-java-mars-R-win32.zip` en `c:\`. Si quiere que le sea más fácil el acceso a Eclipse créese un acceso directo en el escritorio. Para ello muévase al directorio `c:\eclipse` y busque un archivo llamado `eclipse.exe`. Sitúese encima del archivo y pulse el botón derecho del ratón. Le aparecerá un menú desplegable. Escoja la opción *Enviar* → *Escritorio (crear acceso directo)*.

### Ejecución de Eclipse

Para lanzar Eclipse tiene dos opciones, o bien pulse dos veces con el ratón sobre el archivo `eclipse.exe`, o bien pulse dos veces con el ratón en el acceso directo que se ha creado en el escritorio en el apartado anterior.

## PROGRAMAR EN JAVA CON ECLIPSE

Eclipse es un IDE genérico, no está orientado a ningún lenguaje de programación, por lo que para poder programar en Java necesitamos un *plugin* que lo soporte. En nuestro caso utilizaremos el *plugin* JDT que viene con la versión estándar del entorno Eclipse.

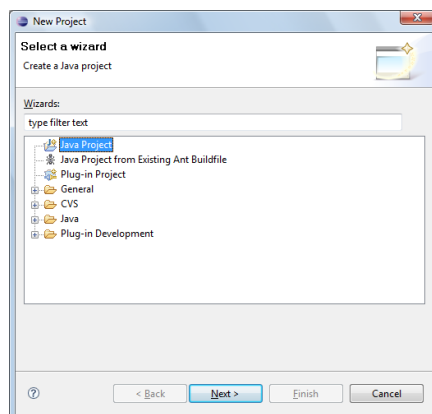
### Creación de proyectos

Para poder crear un programa es necesario crear un proyecto. Un proyecto estará compuesto por un conjunto de recursos relacionados entre sí (código fuente, diagramas de clases, documentación).

Para crear un proyecto nuevo tiene tres opciones:

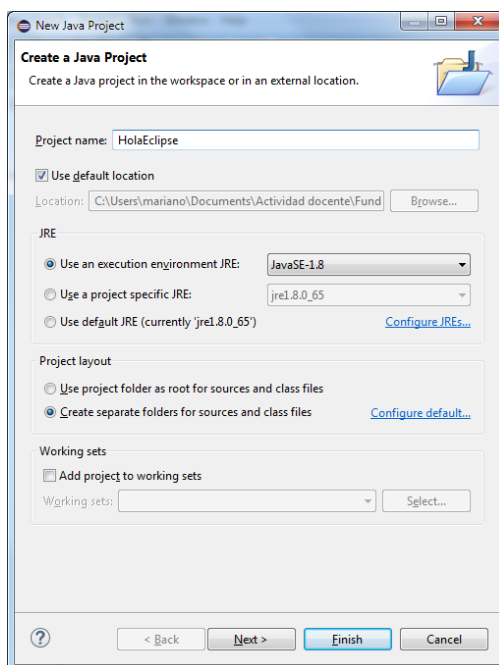
1. Seleccione el ítem *File* del menú principal y luego escoja *New* → *Project*.
2. Pulse el botón nuevo de la barra de herramientas ( )
3. En el explorador de paquetes, pulse con el botón derecho del ratón para mostrar un menú desplegable y seleccione *New* → *Project*.

A continuación le aparecerá la ventana de diálogo de la Figura 4. Seleccione *Java Project* y pulse *Next*.



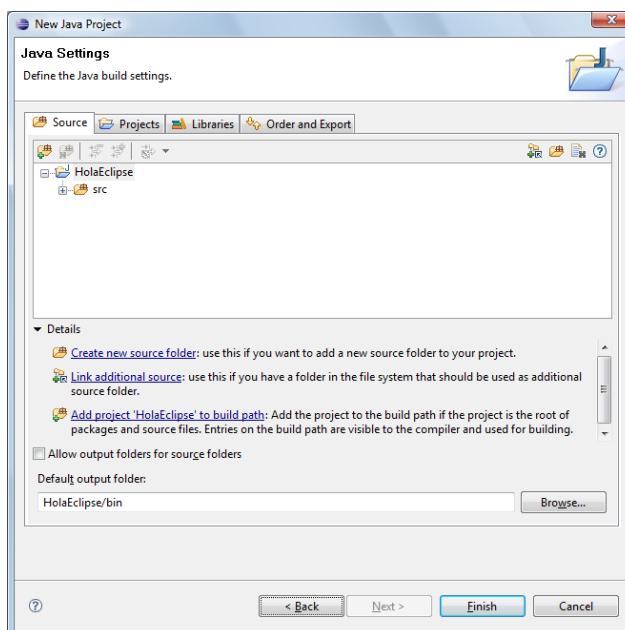
**Figura 4 . Ventana de diálogo *New project***

Le aparecerá otra ventana donde tendrá un cuadro de texto para introducir el nombre del proyecto. Introduzca un nombre, por ejemplo *HolaEclipse*, y mantenga las opciones por defecto. Si tiene instalada una versión de JRE anterior a la 1.8, asegúrese de que escoge JRE 1.8. Para ello en la sección *JRE* seleccione la opción *Use a project specific JRE*, y en la lista desplegable seleccione *jre1.8.0\_65*. En *Project layout* seleccione la opción *Create separate folders for sources and class files* para crear un subdirectorio para el código fuente y otro diferente para las clases compiladas (Figura 5).



**Figura 5. Ventana *New Java Project***

Pulse el botón *Next* y le aparecerá otra ventana con más opciones de configuración (Figura 6). Es recomendable que defina un directorio (llamado por ejemplo *src*) para contener el código y otro (llamado *bin*) para almacenar los *.class* generados.



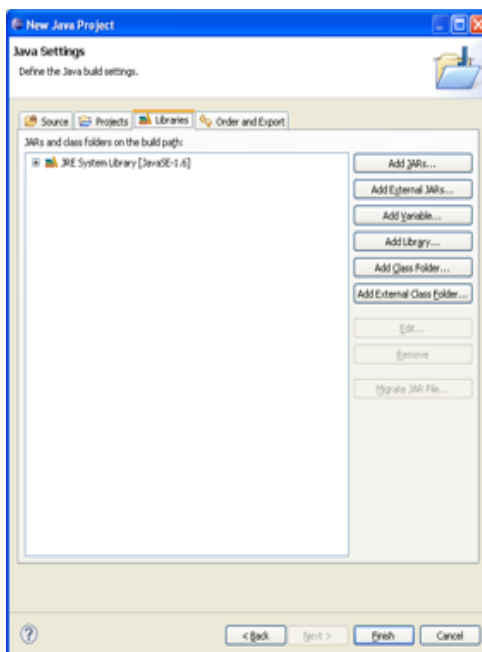
**Figura 6.** Ventana para configurar las opciones Java

En la pestaña *Library* puede añadir todos los archivos `.jar` que sean necesarios (mediante el botón *Add External JARs...*) (Figura 7). Esta configuración se puede modificar en cualquier momento a través del menú desplegable que se muestra en la vista *Package Explorer*. Solo hay que seleccionar el proyecto, pulsar el botón derecho del ratón y en el menú desplegable seleccionar *Properties* → *Java Build Path*.

Al crear el proyecto Java, Eclipse abre de forma automática la *Perspectiva Java*, que no es más que la colección de ventanas o vistas que vienen definidas en el *plugin* JDT para programar con Java. Esta perspectiva está compuesta de las siguientes vistas:

- *Package Explorer*.- Para navegar por los paquetes a los que puede acceder el proyecto.
- *Outline*.- Para mostrar un esquema de la clase cuyo código se está visualizando.

Además, la perspectiva Java hace que se añadan algunos botones a la barra de herramientas para acceder de forma rápida a las funciones más habituales (ejecutar, depurar, crear clases).



**Figura 7.** Pestaña *Libraries* para añadir bibliotecas externas al proyecto

## Crear un paquete en el proyecto

La forma más rápida de crear un paquete es utilizar el *Asistente de creación de paquetes*. Para lanzar el asistente, con la perspectiva Java activa, pulse el botón *New Package* ( ). El *Asistente de creación de paquetes* es un formulario en el que se indicarán las características del paquete que se va a crear (Figura 8). Por convención, los nombres de los paquetes deben comenzar por una letra minúscula.

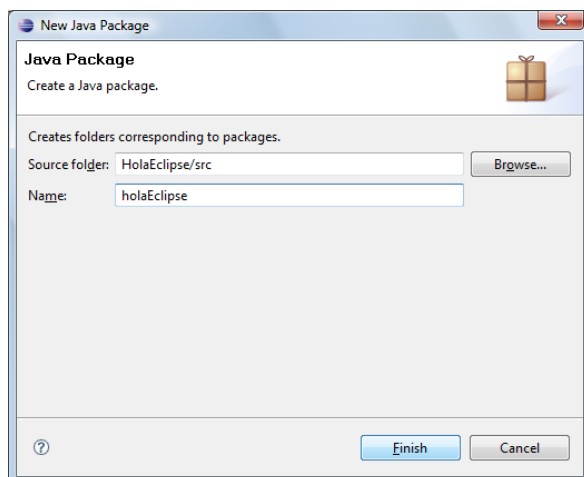


Figura 8. Asistente de creación de paquetes

## Añadir clases o interfaces al proyecto

La forma más rápida de añadir una clase o una interfaz es utilizar el *Asistente de creación de clases o interfaces*. Para lanzar el asistente, con la perspectiva Java activa, pulse el botón *New Class* ( ). Si pulsa en la flecha, le aparecerá una lista desplegable en la que puede escoger añadir otro elemento al proyecto que no sea una clase, como puede ser una *Interfaz*.

Para crear una clase o interfaz dentro de un paquete, lo más cómodo es hacerlo desde la vista del explorador de paquetes (*Package Explorer*) (Figura 9). Para ello sitúese con el ratón en el paquete al que desea añadir la clase y/o interfaz. Pulse el botón derecho y en el menú desplegable seleccione *New*. Luego escoja el tipo de elemento a añadir, bien clase, bien interfaz.

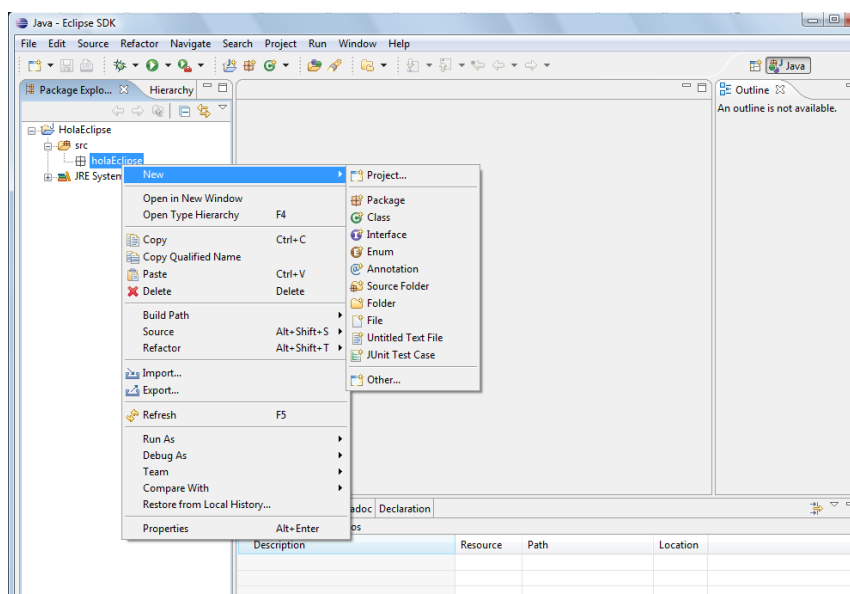


Figura 9. Creación de clases e interfaces en un paquete



El *Asistente de creación de clases o interfaces* es un formulario en el que se indicarán las características de la nueva clase (o interfaz) que se va a crear, tales como nombre, superclase, interfaces que implementa, etc. (Figura 10).

Para el primer ejemplo, seleccione la opción `public static void main (String[] args)`, escoja un nombre para la clase, y pulse el botón *Finish*. Eclipse generará un esqueleto para una clase como el que se muestra en la Figura 11.

Además del asistente, puede crear clases desde cero. Para ello sitúese en el proyecto en la vista *Explorador de paquetes*, y pulse el botón derecho del ratón para mostrar el menú desplegable. Escoja la opción *New* → *File*. Le aparecerá una ventana de diálogo en la que tendrá que introducir el nombre del archivo (en nuestro ejemplo, `HolaEclipse.java`). Note que el archivo debe tener extensión `.java`. En este caso, lo que se crea es un archivo en blanco donde usted deberá teclear todo el código de la clase `HolaEclipse`.

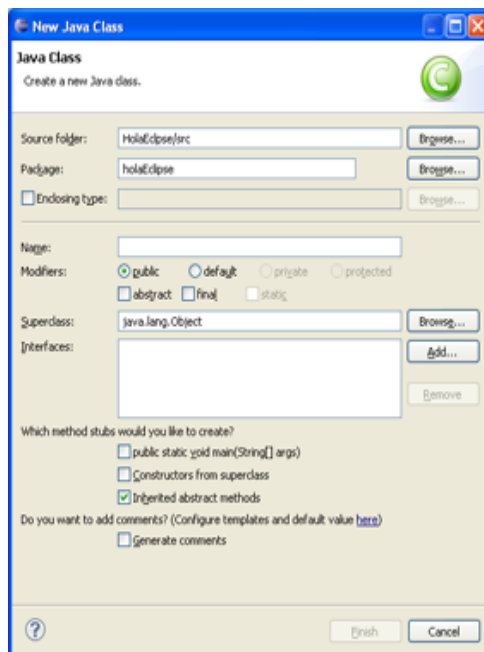


Figura 10. Ventana Nueva Clase

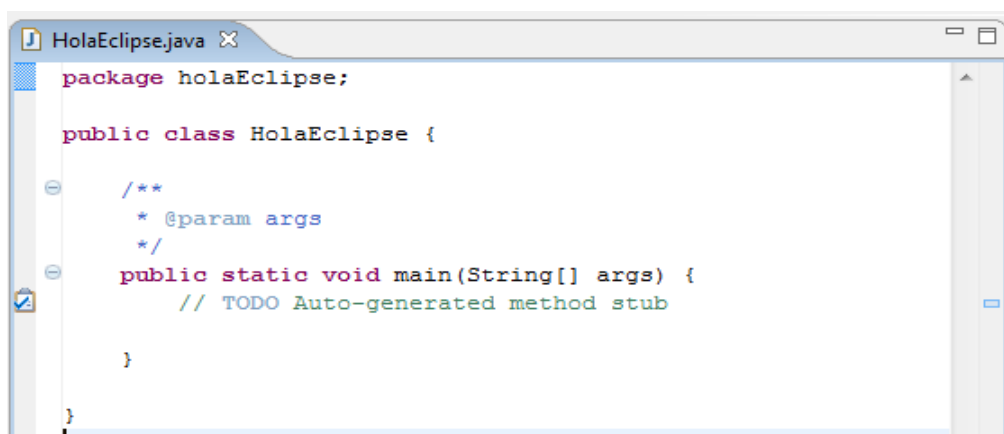



Figura 11. Esqueleto para la clase HolaEclipse

## Compilación de proyectos

Una de las características del IDE Eclipse es que no hay ningún botón para compilar un archivo concreto, ya que la compilación es una tarea que se realiza en segundo plano y de forma automática al guardar los cambios realizados en el código.



## Ejecución de proyectos

Si el proyecto no tiene errores, ejecutarlo es una operación bastante sencilla. Prácticamente todas las opciones de ejecución se pueden gestionar desde el botón *Run* (  ) de la barra de herramientas principal. (Note que para poder ejecutar tiene que tener seleccionada la clase que tenga definido el método `main`.) Este botón puede utilizarse de dos formas:

- Pulsando en el propio botón, con lo que se repetirá la última ejecución realizada.
- Pulsando sobre la flecha, lo que mostrará el menú de ejecución.

El menú de ejecución tiene dos partes, la opción *Run As...*, que permite ejecutar directamente la clase que se está mostrando en la ventana del Editor activo con las opciones de ejecución por defecto, y la opción *Run...* que mostrará una ventana (Figura 12) para definir nuevas configuraciones de ejecución.

En nuestro caso, de momento, nos limitaremos a los valores por defecto, así que utilizaremos la opción *Run As...*, y luego, *Java Application*.

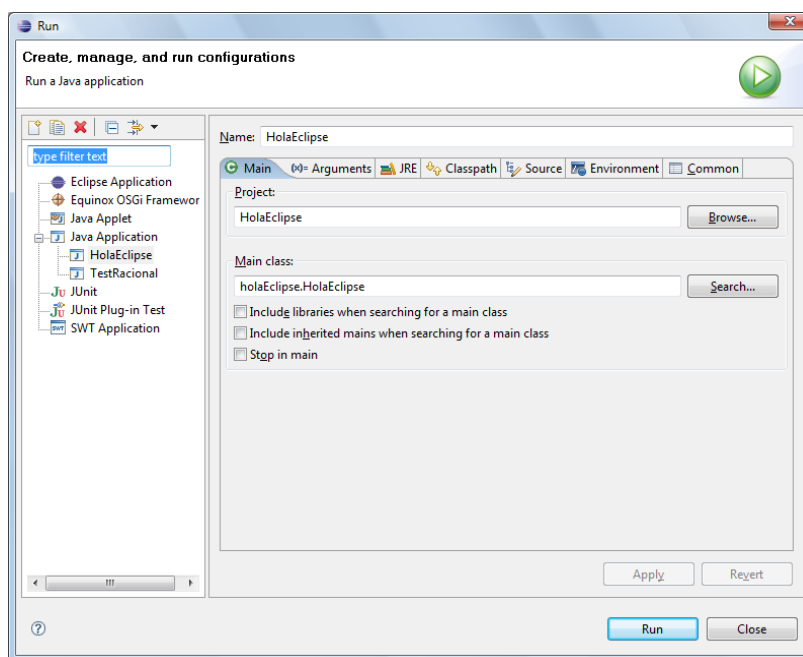



Figura 12. Ventana de configuraciones de ejecución

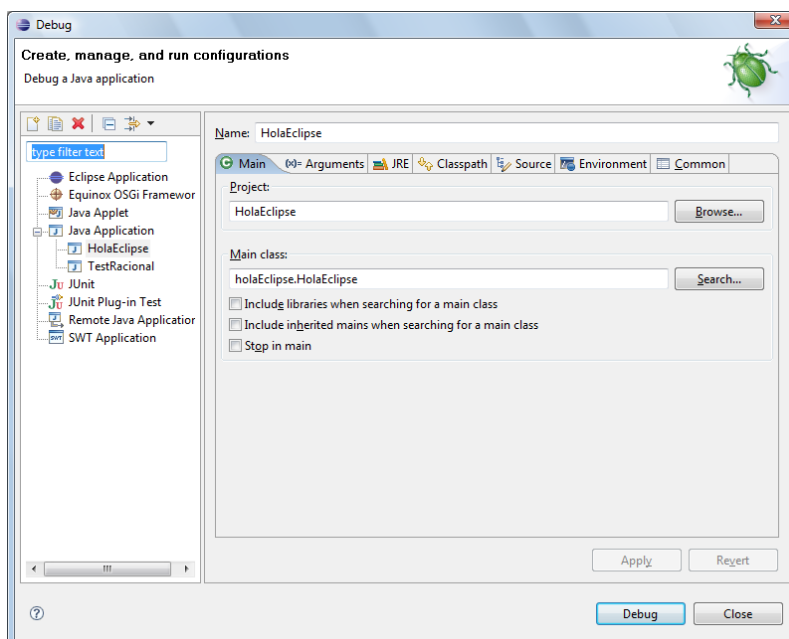
## DEPURACIÓN CON ECLIPSE

Muchas veces su programa no funcionará tal y como espera. Por ello, le serán de utilidad herramientas de depuración más potentes que un simple `println()` y que le permitan ejecutar operaciones tales como realizar una ejecución paso a paso, modificar el valor de algunos campos, parámetros de métodos y variables locales.

Para abrir una sesión de depuración tiene varias opciones:

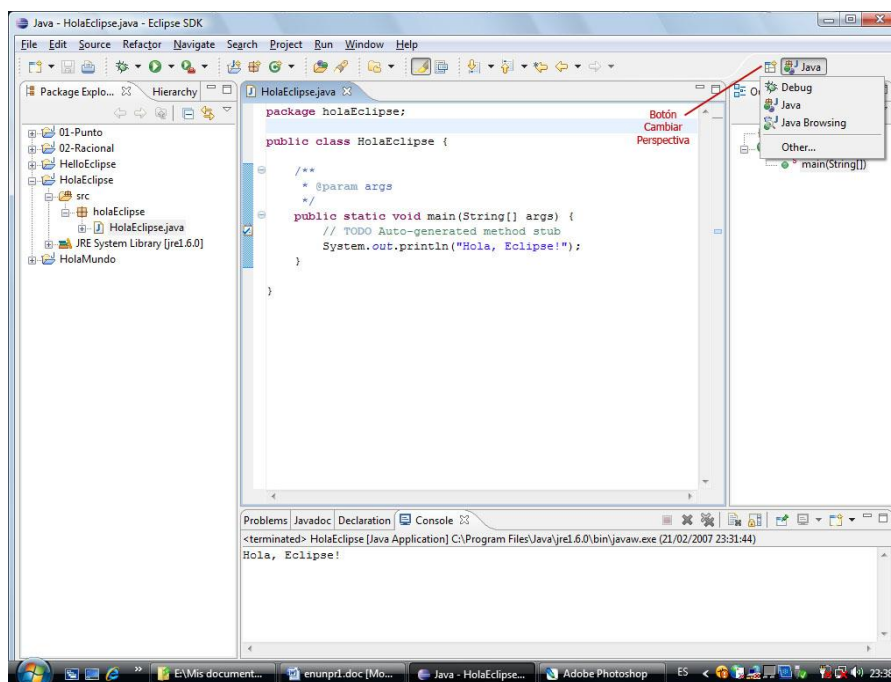
- En la vista Explorador de paquetes sitúese en la clase que va a depurar (en nuestro ejemplo, `HolaEclipse.java`) y seleccione *Debug As* → *Java Application*.
- En la ventana de edición, seleccione *Debug* → *Debug...* del menú contextual que aparece al pulsar con el botón derecho del ratón.
- En la barra de herramientas pulse el botón de depuración (  ).

Con la segunda y tercera opción aparecerá una ventana de configuración de la sesión de depuración (Figura 13). En proyectos más complejos, esta ventana puede dar mucho juego. En nuestro caso, la única opción que nos puede ser de utilidad es la opción *Stop in main* si quiere que el programa se detenga en la función `main()`.



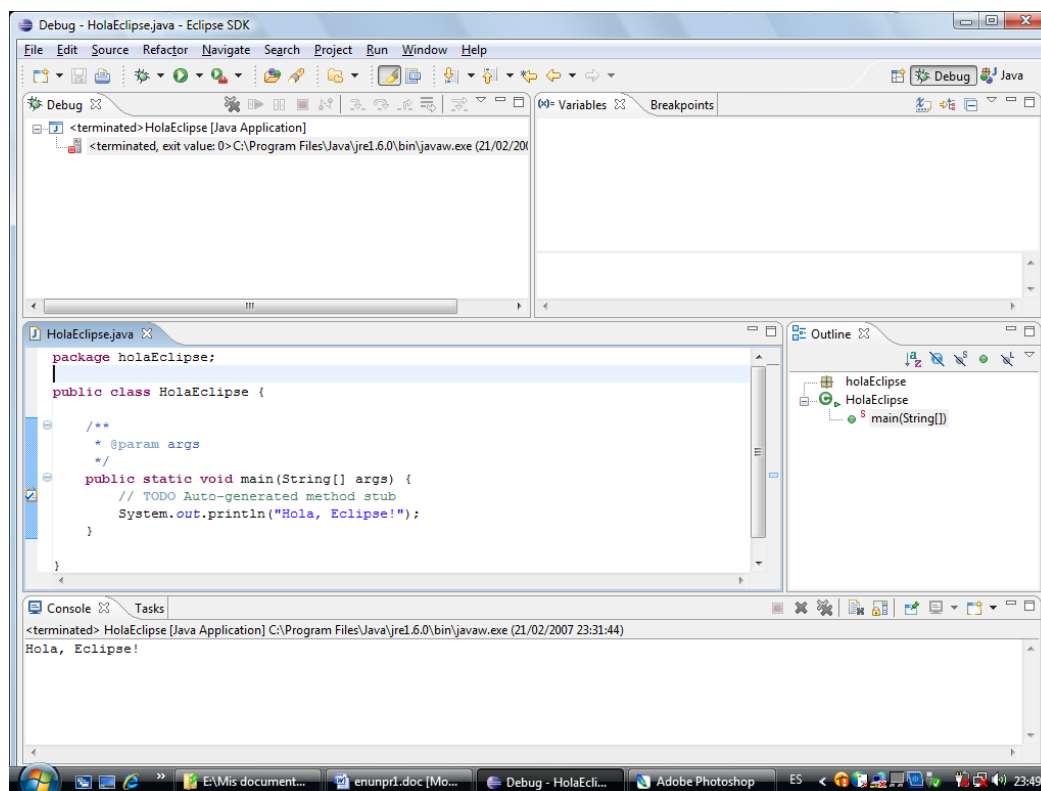
**Figura 13. Ventana de configuración de la sesión de depuración**

Al pulsar el botón de depuración, Eclipse, de forma automática, abre la *Perspectiva de Depuración* (Figura 15), donde se muestra toda la información relativa al programa que se está depurando. Usted también puede cambiar a esta perspectiva mediante el botón de perspectivas situado en la parte superior derecha de la ventana principal (Figura 14).



**Figura 14. Cambio de perspectiva**

Una sesión de depuración se puede empezar desde cualquier perspectiva. Sin embargo, la perspectiva de depuración (Figura 15) es la que está diseñada para ello.



**Figura 15. Perspectiva de Depuración**

La perspectiva de depuración está formada por varias vistas, que se explican con un poco más de detalle a continuación.

### Vista Editor

Es la ventana situada en el centro de la pantalla. En ella se muestra el programa que se está depurando. Sobre el programa se va marcando la traza con una flecha azul situada en el margen izquierdo. La línea a la que señala la flecha será la que se ejecuta a continuación.

### Vista Debug

Es la ventana etiquetada como *Debug* y situada en la parte superior izquierda. Aquí se controla la ejecución del programa que se está depurando. Tiene asociada una barra de botones, que es desde donde se realizan las funciones habituales de ejecución paso a paso, detener la depuración, ejecutar hasta el final.

Algunas de estas funciones también se pueden realizar a través del teclado. Así, pulsando F5 puede ir ejecutando paso a paso el programa y pulsando F6 puede ejecutar una función sin tener que entrar en su código y ejecutarlo paso a paso.

Además de acceder a estas funciones a través de la barra de botones, puede hacerlo utilizando la opción *Run* del menú principal, una vez que el programa está suspendido. Así, en la Figura 16 puede observar que con la opción *Resume* o bien pulsando F8, se reanuda la ejecución del programa, y con la opción *Terminate* termina la ejecución del programa. Pulsando F5 o la opción *Step Into* da un paso en la ejecución del programa, y si ese paso es la llamada a una función o a un método, entrará en su código para ejecutarlo paso a paso. Finalmente, con F6 (opción *Step Over*), como se comentó anteriormente, ejecuta la función sin entrar en su código.

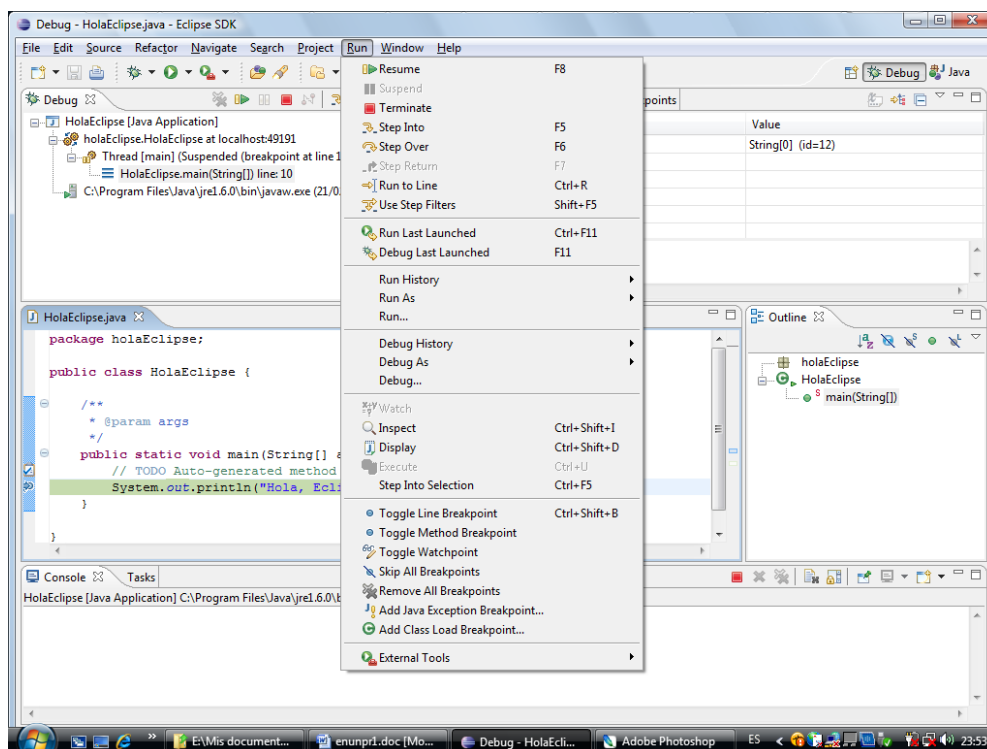


Figura 16. Menú *Run* con las opciones de depuración

## Vista de Inspección

En esta ventana, situada a la derecha de la ventana *Debug*, se pueden ver los valores de las variables, puntos de ruptura o expresiones que intervienen en el programa. Podrá observar que en esta ventana aparecen dos pestañas, una etiquetada como *variables*, donde podrá ver los valores de todas las *Variables* (atributos, etc) que se pueden alcanzar en el ámbito de la línea que se está ejecutando en ese momento, y otra etiquetada como *Breakpoints* que presenta la lista de inspección de puntos de ruptura.

Para establecer un punto de ruptura o *breakpoint* basta con hacer doble clic con el ratón en el margen izquierdo del editor de código a la altura de la línea en la que se quiere establecer dicho punto de ruptura. El *breakpoint* creado se identificará por un punto azul situado sobre la línea.

En la pestaña de *Breakpoints* puede ver todos los puntos de ruptura definidos y también configurar sus propiedades. Con el menú contextual puede activar o desactivar un punto de ruptura, eliminarlo, configurarlo para que se detenga la ejecución cuando pase por él un número determinado de veces, etc.

## Vista Consola

Es la ventana situada en la parte inferior de la perspectiva *Debug*, aunque también aparece en la perspectiva *Java*. En esta ventana se encuentran redireccionadas la entrada y la salida estándar durante la ejecución del programa.

## UTILIDADES A LA HORA DE PROGRAMAR CON ECLIPSE

El *plugin* JDT de Eclipse viene con unas cuantas utilidades que le ayudarán en las tareas de programación. Una de ellas es lo que se conoce como *reconocimiento sintáctico* de las palabras reservadas del lenguaje. Así, las palabras reservadas de Java aparecerán en negrita y de color morado, los comentarios aparecerán en verde y los comentarios de documentación en azul.

## Corrector de errores

Otra de las utilidades que incorpora JDT es el marcado sobre el código del programa de aquellos sitios en los que se puede producir un error de compilación. Esta característica funciona de forma parecida a como lo hacen los correctores ortográficos de los procesadores de texto.

Cuando JDT detecta un error de compilación, la sentencia errónea se marca subrayándola con una línea ondulada de color rojo. Si en lugar de un error de compilación, lo que se produce es un aviso o *warning*, la línea en lugar de roja es amarilla.

Además de señalar la sentencia concreta, la línea en la que se produce el error o *warning* también se marca con un icono de error que aparece en la barra de desplazamiento izquierda del editor. Si pulsa una vez con el ratón sobre esta marca, Eclipse mostrará un menú desplegable con posibles soluciones para los errores detectados. Si selecciona alguna de las sugerencias, Eclipse realizará los cambios de forma automática.

## Mecanismo para completar código

El mecanismo utilizado por Eclipse para completar código es similar al que utilizan otros IDE's, es decir, cuando deja de escribir durante unos segundos, se muestran, si los hay, todos los términos (palabras reservadas, nombres de funciones, de variables, de campos, etc.) que empiecen por los caracteres que ha tecleado.

Cuando se escriben algunos caracteres, como por ejemplo el punto, se lanza el mecanismo para completar código de forma automática, sin necesidad de esperar unos segundos.

Si el mecanismo para completar código tarda en lanzarse, puede hacerlo usted manualmente pulsando la combinación de teclas `Ctrl+Espacio`.

La utilidad para completar código también le puede ayudar cuando escriba una llamada a una función, ya que a la hora de escribir los parámetros que se le pasan a un método se va mostrando en una caja de texto el tipo de dato que estos pueden tener.

## Formateo del código

A la hora de programar es importante tener un código que sea legible, ordenado y claro. Eclipse dispone de una herramienta para darle formato al código de forma automática, según unos criterios preestablecidos.

Para invocar a esta herramienta solamente tiene que pulsar con el botón derecho del ratón sobre la ventana de edición del archivo fuente. Le aparecerá un menú desplegable. Seleccione la opción *Source* → *Format*.

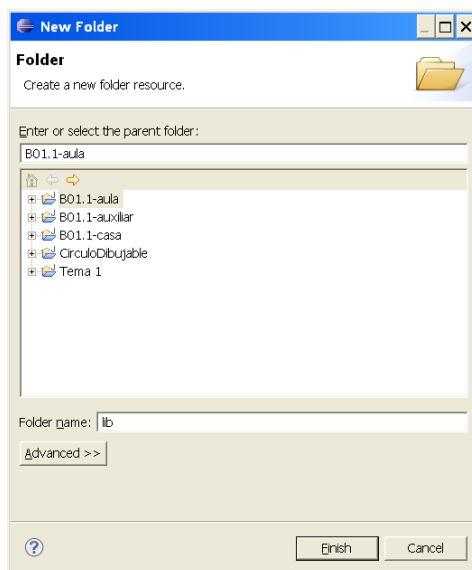
## IMPORTACIÓN DE BIBLIOTECAS .jar A PROYECTO ECLIPSE

---

Para añadir una biblioteca .jar a un proyecto Eclipse es aconsejable tenerla bien localizada. Existen dos opciones: tener un directorio común para todos los proyectos en el que se guarden todas las bibliotecas, y así ahorramos espacio en disco; o bien, hacer que cada proyecto tenga un directorio, al que se suele llamar `lib`, en el que estén todas las bibliotecas usadas en el proyecto. La ventaja de esta última opción es que cuando exportemos el proyecto, no se nos olvidará incluir las bibliotecas.

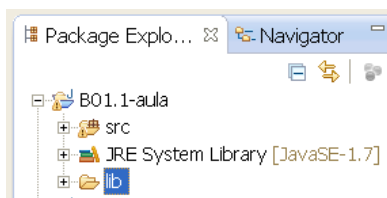
En el caso de nuestra asignatura, y debido a que las bibliotecas que vamos a usar son pequeñas, optaremos por guardar en cada proyecto sus propias bibliotecas. Para hacer esto daremos los siguientes pasos en Eclipse:

1. Crear en el proyecto una carpeta nueva llamada `lib`. Para ello, colocamos el ratón encima del proyecto, y tras pulsar el botón derecho escogemos los ítems *New* > *Folder*. Aparecerá un asistente (Figura 17), en el que tendremos que escribir el nombre de la carpeta en el cuadro de texto *Folder name*, y asegurarnos de que en el árbol de directorios está seleccionado el proyecto en el queremos añadir la carpeta.



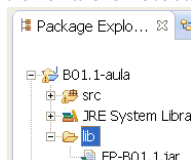
**Figura 17. Asistente de creación de carpetas o directorios**

Tras pulsar *Finish*, veremos la carpeta creada en la vista *Package Explorer* (Figura 18).



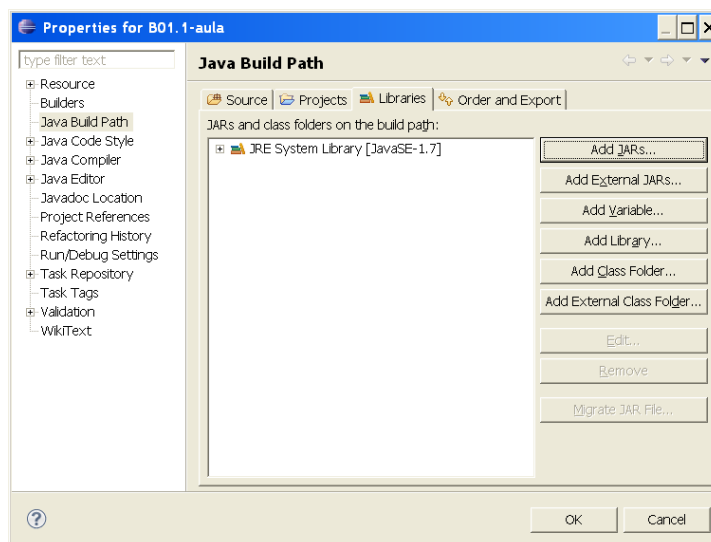
**Figura 18. Vista *Package Explorer* que refleja la carpeta creada**

2. Copiar en la carpeta `lib` el archivo *jar* que contiene la biblioteca. Para copiar puede usar *Copy&Paste* (Figura 19)



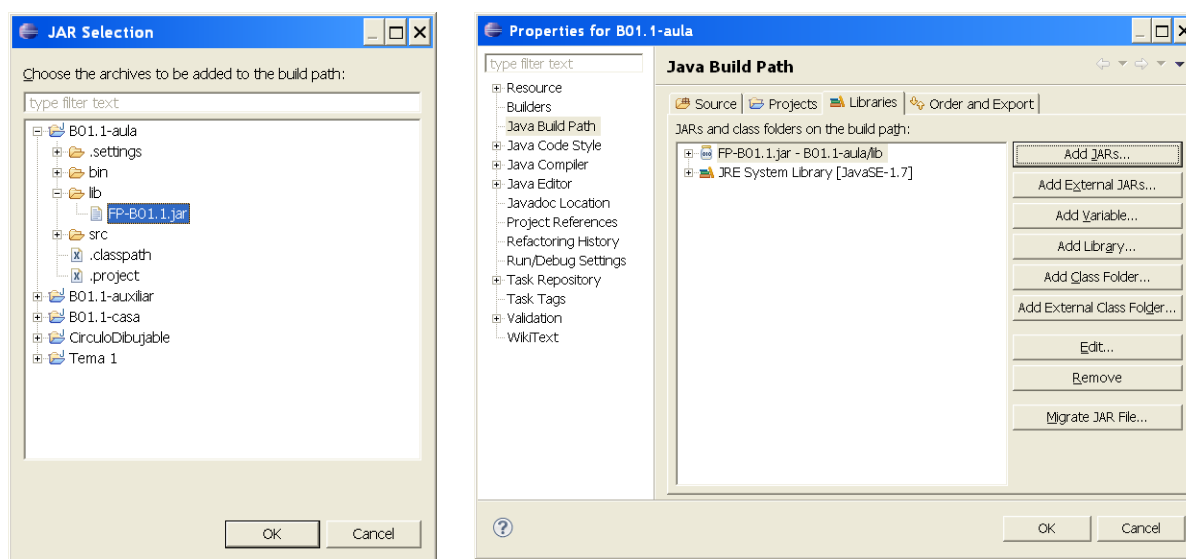
**Figura 19. Carpeta `lib` con la biblioteca `FP-B01.1.jar`**

3. Configurar el proyecto para indicar dónde está el archivo de la biblioteca. Para ello pulsando con el botón derecho en el proyecto y escogemos las opciones *Build Path > Configure Build Path*. Aparecerá un asistente (Figura 20), en el que seleccionamos la pestaña *Libraries* y pulsamos el botón *Add JARs*.



**Figura 20. Asistente de configuración de rutas**

En el asistente de selección de JARs escogemos el archivo JAR de la carpeta `lib` de nuestro proyecto. El archivo JAR queda añadido a la pestaña *Libraries* (Figura 21)



**Figura 21. Navegador de proyectos y archivo JAR añadido al asistente**

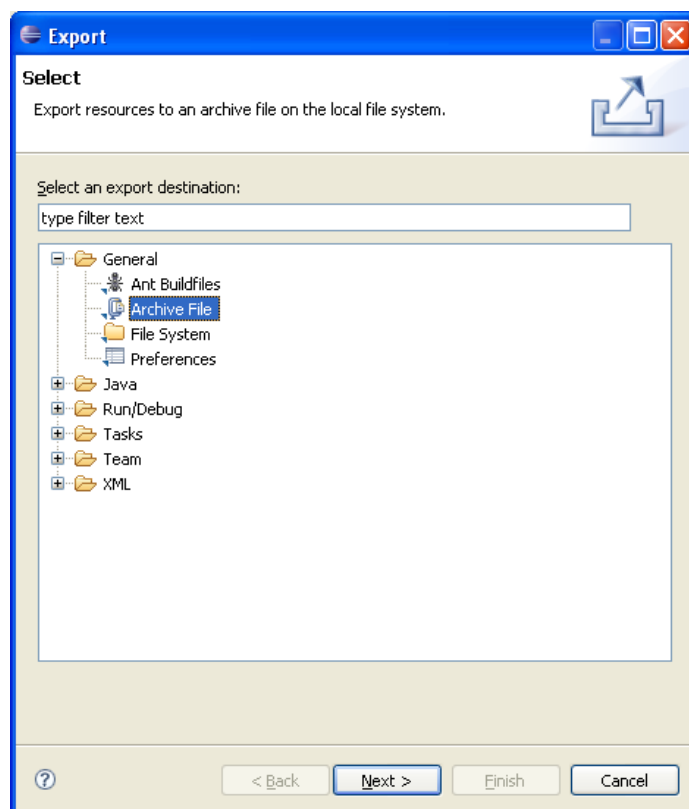
Al pulsar el botón *OK*, se cerrará el asistente y comprobaremos que en la vista *Package Explorer* ha aparecido un nuevo elemento asociado a nuestro proyecto llamado *Referenced Libraries*, que incluye a nuestro archivo *.jar*.

## EXPORTACIÓN DE UN PROYECTO COMO UN ARCHIVO .ZIP

La acción de exportar un proyecto nos va a permitir llevarnos nuestro trabajo de un ordenador a otro. Si estamos trabajando en un laboratorio de la Escuela y queremos llevarnos el trabajo a nuestro ordenador de casa, tendremos que realizar esta operación por cada uno de los proyectos. El resultado será que tendremos un archivo *.zip* por cada uno de los proyectos en los que estamos trabajando. Los pasos que hay que dar para exportar un proyecto son:

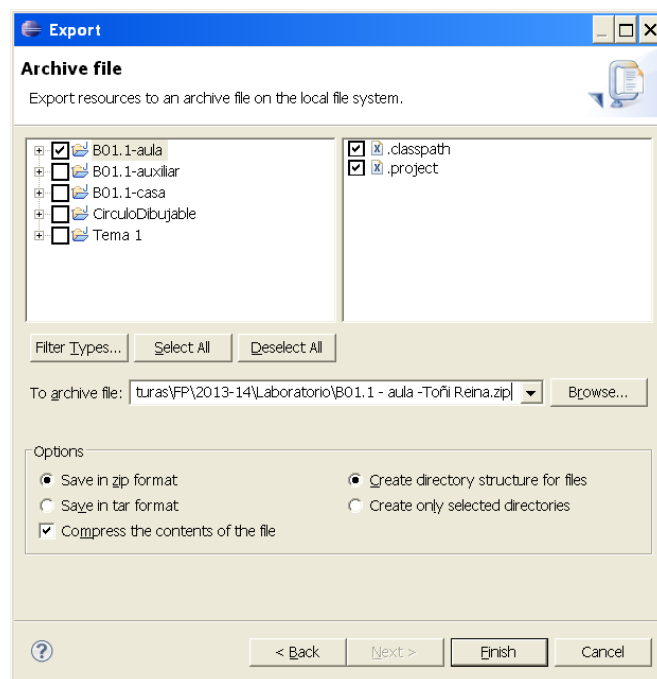
1. Seleccionar en el menú *File>Export*. Aparecerá un asistente en el que desplegaremos la carpeta *General* y escogeremos la opción *Archive File* (Ver Figura 22).





**Figura 22. Asistente para exportar un proyecto**

2. En la siguiente ventana del diálogo con el asistente, debe asegurarse de que están marcados todos los directorios que intervienen en el proyecto (bin , src y lib si lo hubiera). Fíjese también que debe marcar la opción de guardar como archivo .zip (Figura 23). Fíjese bien en el directorio o carpeta en la que va a guardar su archivo .zip. Y escoja un nombre significativo, por ejemplo, B01.1-aula-Toñi Reina Quintero.zip.



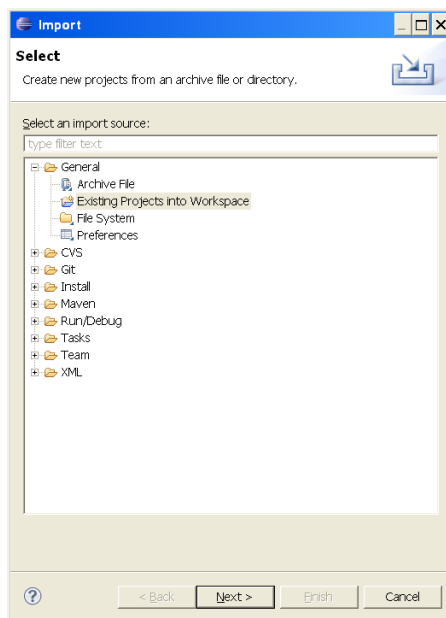
**Figura 23. Configuración de exportación del proyecto**

Al pulsar *Finish*, Eclipse creará el archivo .zip, que podrá copiar para llevarlo a otro equipo informático.

## IMPORTACIÓN DE UN PROYECTO DESDE UN ARCHIVO .ZIP

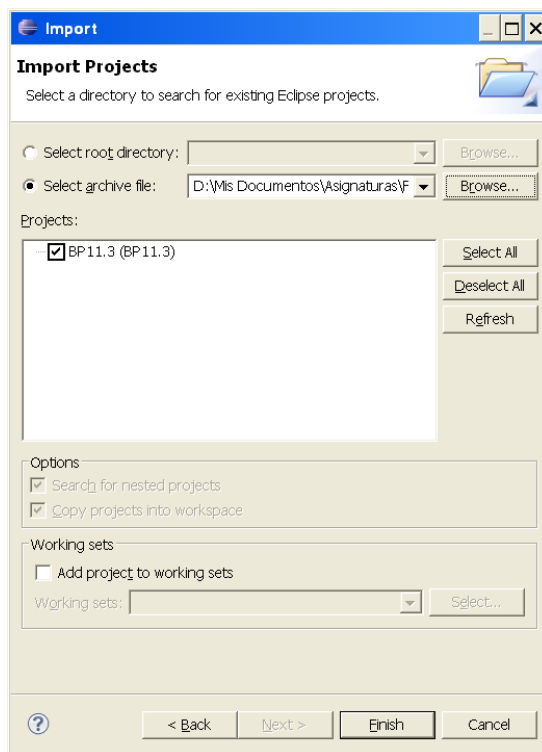
La acción de importar un proyecto nos va a permitir copiar en nuestro espacio de trabajo (*workspace*) un proyecto creado en otro ordenador. Si estamos trabajando en un laboratorio de la Escuela y queremos copiar el trabajo que hemos realizado en casa (previa importación en un .zip), tendremos que hacer lo siguiente:

1. Seleccionar en el menú *File>Import*. Aparecerá un asistente en el que desplegaremos la carpeta *General* y escogeremos la opción *Existing Projects into Workspace* (Ver Figura 24).



**Figura 24. Asistente de importación de proyectos**

2. Pulsamos *Next* y en el siguiente paso seleccionamos la opción *Select archive file* y pulsamos el botón *Browse*. Aparecerá una nueva ventana con un navegador que nos permitirá bucear por el sistema de archivos y escoger el archivo .zip en el que tenemos el proyecto (Figura 25). Al pulsar *Finish*, Eclipse añadirá el proyecto al *Workspace*.



**Figura 25. Selección de archivo .zip en el asistente de importación**