Práctica 3:

Un entorno de programación en Java

Objetivo de la práctica

Los objetivos de esta tercera práctica de la asignatura son los siguientes:

- Familiarización con el computador y los elementos básicos de su sistema operativo.
- Familiarización con el entorno de desarrollo de programas en Java Eclipse.

1. El entorno de desarrollo Eclipse

Para facilitar la tarea de desarrollo de programas existen entornos que permiten trabajar de un modo cómodo. Podemos citar JBuilder de Borland-Inprise, NetBeans, Eclipse…. Nosotros recomendamos el uso de Eclipse, que es de libre distribución (en el Apéndice 3 se incluye documentación sobre la obtención e instalación del entorno).

Inicia la herramienta Eclipse haciendo doble clic sobre el ejecutable Eclipse.exe que se puede encontrar en el escritorio de los laboratorios de prácticas. Para instalar esta aplicación en otros ordenadores puede consultarse el Apéndice 3. Al iniciarse *Eclipse* aparecerá la pantalla de bienvenida que se muestra en la Figura 1.

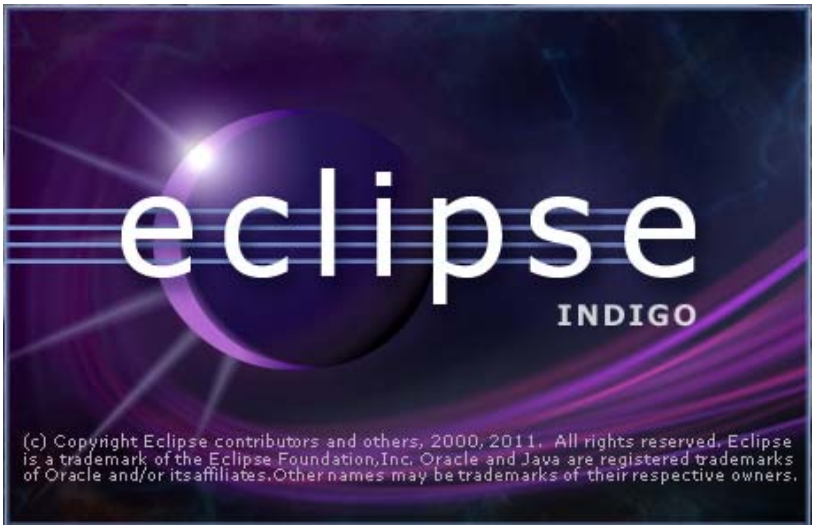


Fig. 1. Pantalla de presentación

Para poder comenzar a trabajar con el entorno de desarrollo será necesario definir un directorio de trabajo (*workspace*), tal y como se observa en la Figura 2. Este *workspace* contendrá todos los ficheros creados con Eclipse. Por ejemplo, se puede establecer el directorio `c:\practicasARQ` como directorio de trabajo.

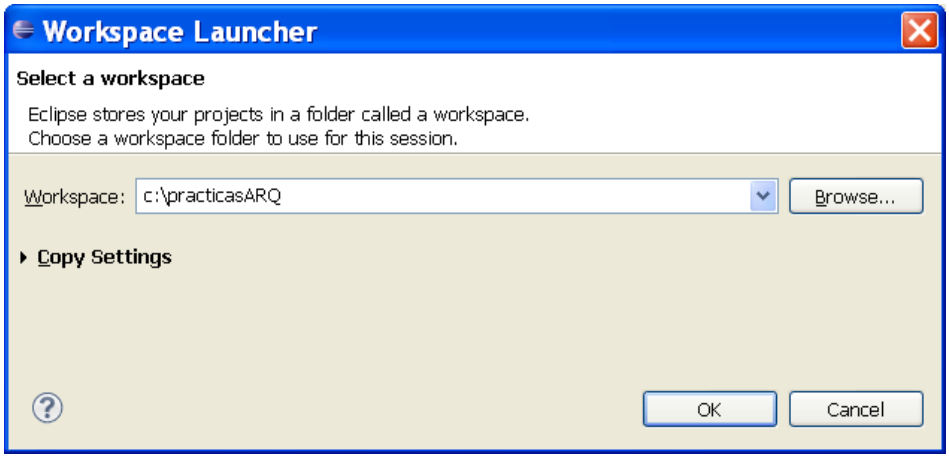


Fig. 2. Configuración del workspace

En los siguientes subapartados aparecen sólo alguna de las características o recomendaciones de uso de la herramienta Eclipse. En caso de que tengas más dudas acerca del uso de la herramienta, deberás consultar la ayuda en línea (opción de menú *Help: Help Contents*), o bien accediendo a la página Web http://help.eclipse.org/, donde se encuentra también toda la documentación del programa.

1.1. Pantalla de bienvenida

La primera vez que se utiliza *Eclipse*, aparecerá una pantalla de bienvenida (ver Figura 3).

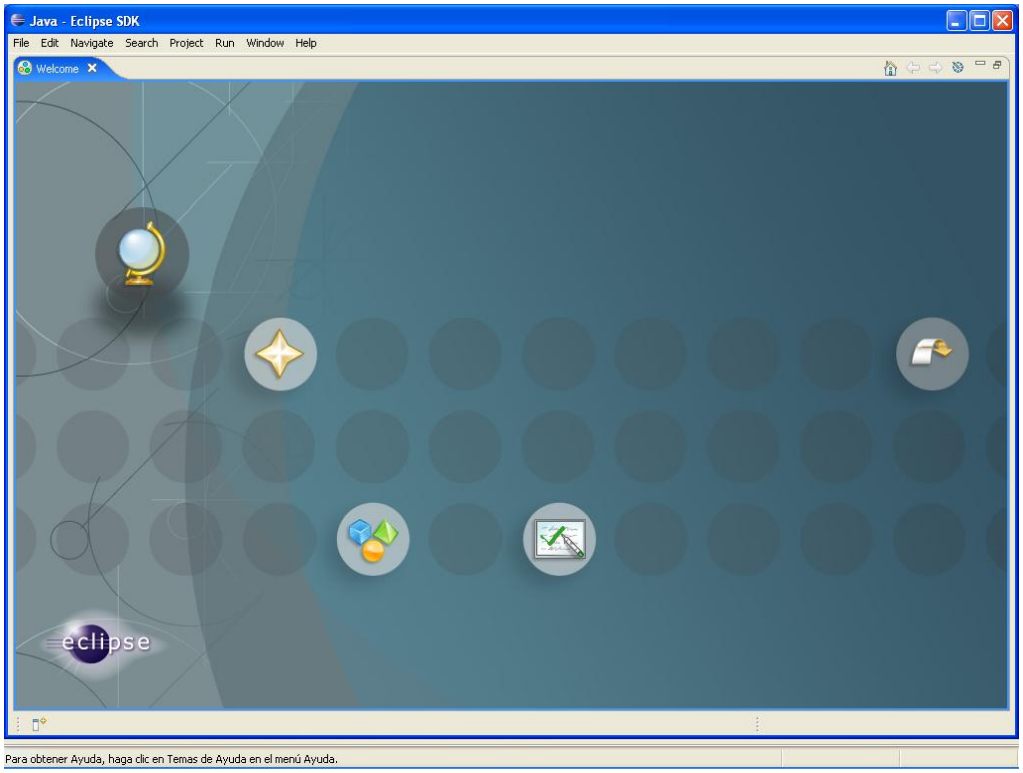
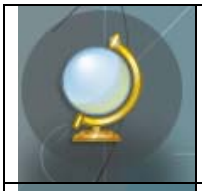
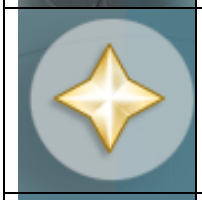
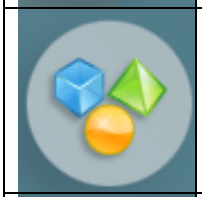
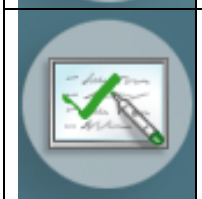
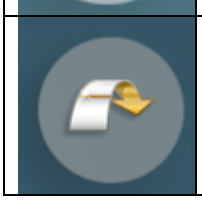


Fig. 3. Pantalla de Bienvenida

En esta pantalla de bienvenida, existen las siguientes opciones a realizar:

	*"Overview"*. Proporciona un vistazo general a las características principales del entorno de desarrollo, sobre todo para familiarizarse con la interfaz de usuario.
	*"What's New"*. Para los que han utilizado con anterioridad la herramienta Eclipse, esta utilidad describe las nuevas características que proporciona la versión 3.2 frente a versiones anteriores.
	*"Samples"*. La mejor manera de conocer cómo funciona la herramienta, sería la ejecución de los ejemplos que Eclipse pone a nuestra disposición.
	*"Tutorials"*. Otra forma de conocer todas las características de *Eclipse* es ejecutando alguno de los tutoriales de manejo del entorno de desarrollo disponibles.
	*"Workbench"*. Usando esta opción se accede directamente al marco de trabajo de Eclipse.

1.2. Entorno de trabajo

El entorno de trabajo o *Workbench* nos proporciona una interfaz de usuario intuitiva, bien estructurada y sencilla de utilizar, ya que tiene bien definidas cada una de las zonas de trabajo. En la Figura 4 se puede observar la interfaz de usuario de *Eclipse*.

Las diferentes partes en las que está dividida la interfaz son:

a) En la zona de la izquierda, se halla un menú ("*Package Explorer*") que agrupa los componentes de cada uno de los proyectos abiertos.

b) En la zona central de la pantalla se encuentra un editor, que sirve para ver y modificar los ficheros Java con los que trabajamos.

c) En la zona de la derecha, hay un resumen (*outline*) que nos permite comprobar la estructura del fichero activo en el editor (clases, métodos, variables, etc.)

d) En la parte inferior del editor se encuentran las aplicaciones que nos permiten conocer el estado de desarrollo del proyecto desarrollado, tanto errores como tareas a completar y la salida producida por los programas ejecutados.

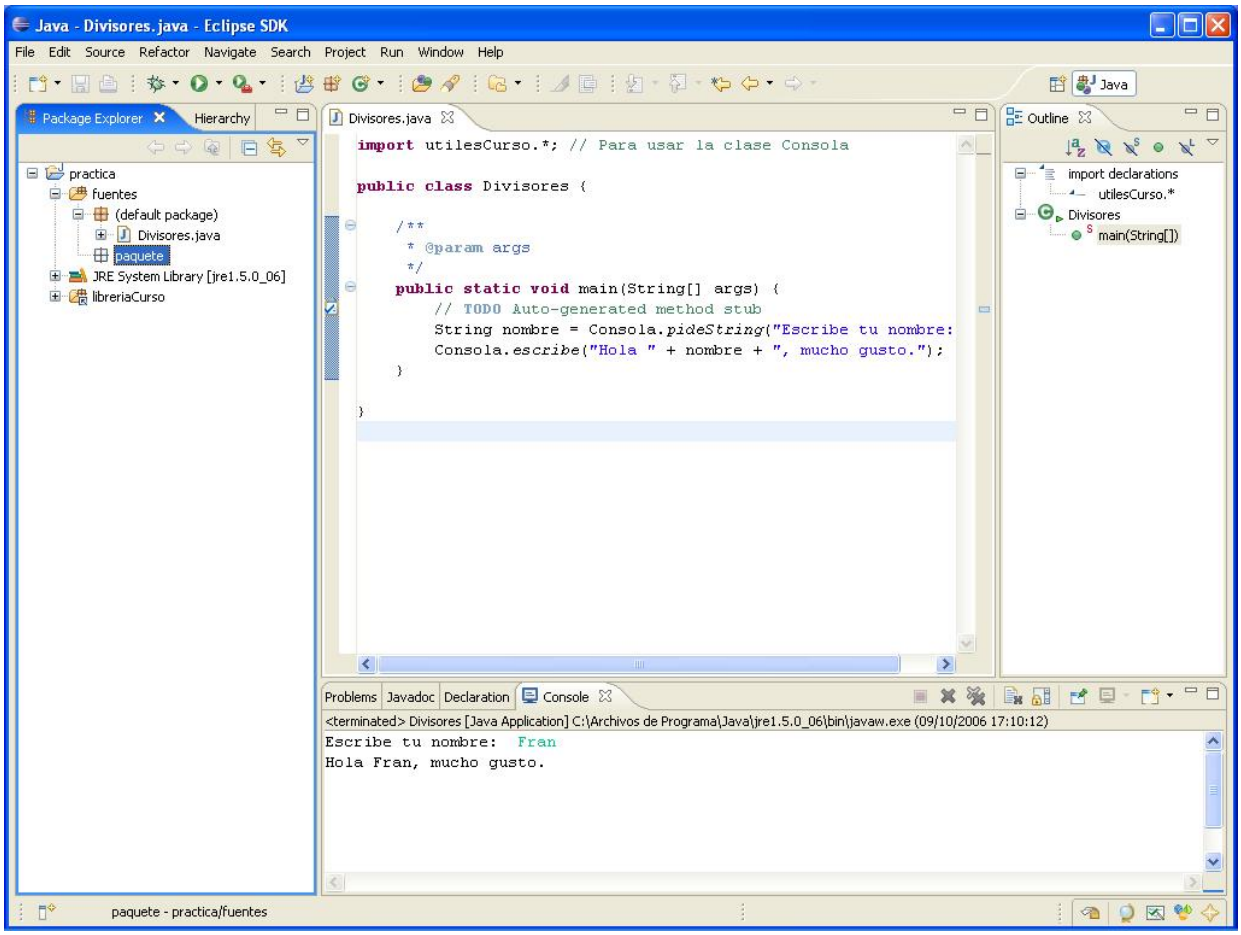


Fig. 4. Entorno de Trabajo (Workbench)

1.3. Crear un proyecto

Para la realización de cada práctica se recomienda crear un proyecto nuevo. De esta forma se pueden agrupar los ficheros correspondientes a una determinada práctica dentro de ese proyecto para compilar y ejecutar los programas que comparten una configuración común.

Los proyectos se crean a través de la opción *"File: New: Java Project".* Para crear el nuevo proyecto hay que dotar a dicho proyecto de un nombre identificativo (ej. *practica3*), que estará representado como un directorio dentro del *workspace* definido cuando se lanzó *Eclipse*. Conviene seleccionar la opción *"Create separate source and output folders"*. En la Figura 5 se pueden observar todas las propiedades que se pueden definir en la creación del proyecto.

El nuevo proyecto que has creado se configurará sobre la carpeta:

```
c:\practica3
```

Observa que en la carpeta `practica3` se han creado dos subcarpetas. En `src` se almacenarán los ficheros con el código (*source*) Java; mientras que en la carpeta `bin` se almacenar los ficheros binarios (*binaries*).

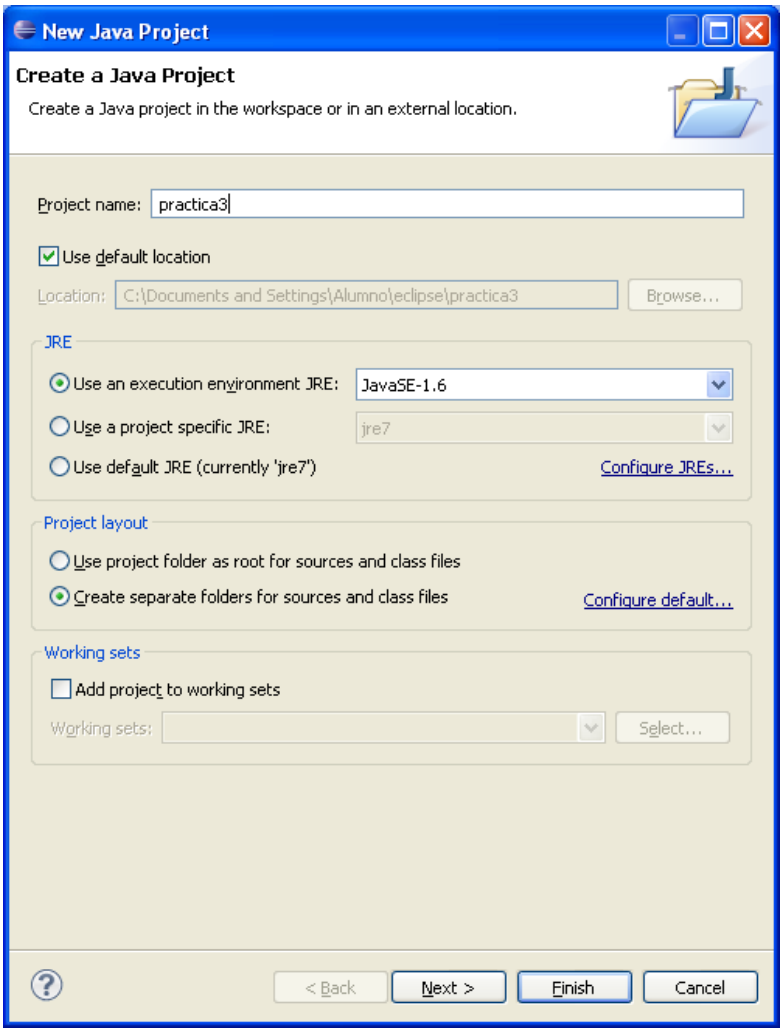


Fig. 5. Propiedades del nuevo proyecto

1.4. Crear una aplicación

Los lenguajes orientados a objetos, como Java, se basan en el concepto de clase. Todas las variables y métodos de Java deben pertenecer a una clase. Una *clase* es una colección de datos (variables) y de métodos (funciones) que operan sobre dichos datos. Cada clase pública debe ir en un fichero con extensión *.java*, que se llame igual que la clase.

Para añadir nuevas clases al proyecto utilizaremos la opción de menú *"File: New: Class"*. Selecciona *src* como subcarpeta e introduce *HolaMundo* como nombre del archivo que contendrá el programa (clase) que deberás escribir a continuación. La opción "*public static void main(String[] args)*" debe marcarse para que la clase sea ejecutable (en ese caso, se ejecutará el método *main* de la clase). La Figura 6 ilustra este proceso.

En la parte superior de la pantalla nos aparece un mensaje advirtiendo que "*el uso del paquete por defecto está desaconsejado*". En futuras sesiones de prácticas hablaremos de cómo crear un paquete donde estructurar las clases que creemos para cada proyecto.

Al finalizar la creación, se creará automáticamente un pequeño esqueleto con el código inicial común a cada clase, que deberá ser completado para contener el siguiente código:

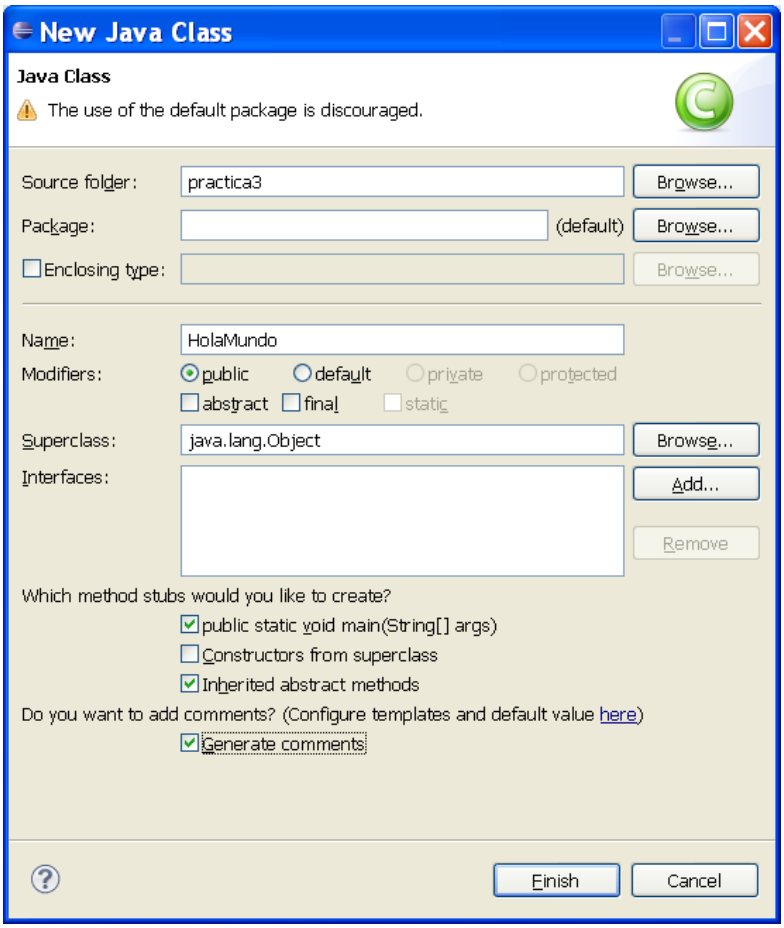


Fig. 6. Creación de una nueva clase Java

```
import java.util.*;

/**
 * La clase HolaMundo ejemplifica el uso de Eclipse para programar en Java.
 * @author Fernando Bobillo.
 */
public class HolaMundo
{

	/**
	 * El método main contiene el programa principal.
	 * @param args Puede recibir una lista de parámetros como argumento.
	 */
	public static void main(String[] args)
	{
		System.out.print("Escribe tu nombre: ");
		String nombre = (new Scanner(System.in)).nextLinea();
		System.out.println("Hola " + nom + ", mucho gusto.");
	}
}
```

El código anterior contiene algunos errores escritos voluntariamente para que luego se produzcan errores de compilación.

El editor de *Eclipse* es un editor típico del entorno Windows, se puede copiar, cortar, pegar, buscar y sustituir etc. Tiene un sistema de sangrado inteligente que facilita la tarea de escritura. Se puede conseguir aumentar el sangrado de una porción de código o disminuirlo seleccionándolo y pulsando tabulador o mayúsculas-tabulador.

Guarda el texto en disco pulsando en el icono del disco que hay en la barra de herramientas o bien con Ctrl-S o bien con *File:Save*.

1.5. Compilación de aplicaciones

Para *compilar* el programa deberás utilizar la opción de menú "*Project: Build Project"*, en el caso de que no esté marcada la compilación automática. En este último caso, será suficiente con salvar el fichero para que se compile.

En la parte de abajo (subpestaña "*Problems"*) se te informará sobre el proceso de compilación indicándote los posibles errores (ver Figura 7). Puedes ir avanzando de error en error, una flecha en el código fuente te señala la línea donde se ha producido el error.

Si trabajásemos sin un entorno de desarrollo, sería equivalente a abrir una ventana de órdenes del sistema operativo y teclear:

```
javac HolaMundo.java
```

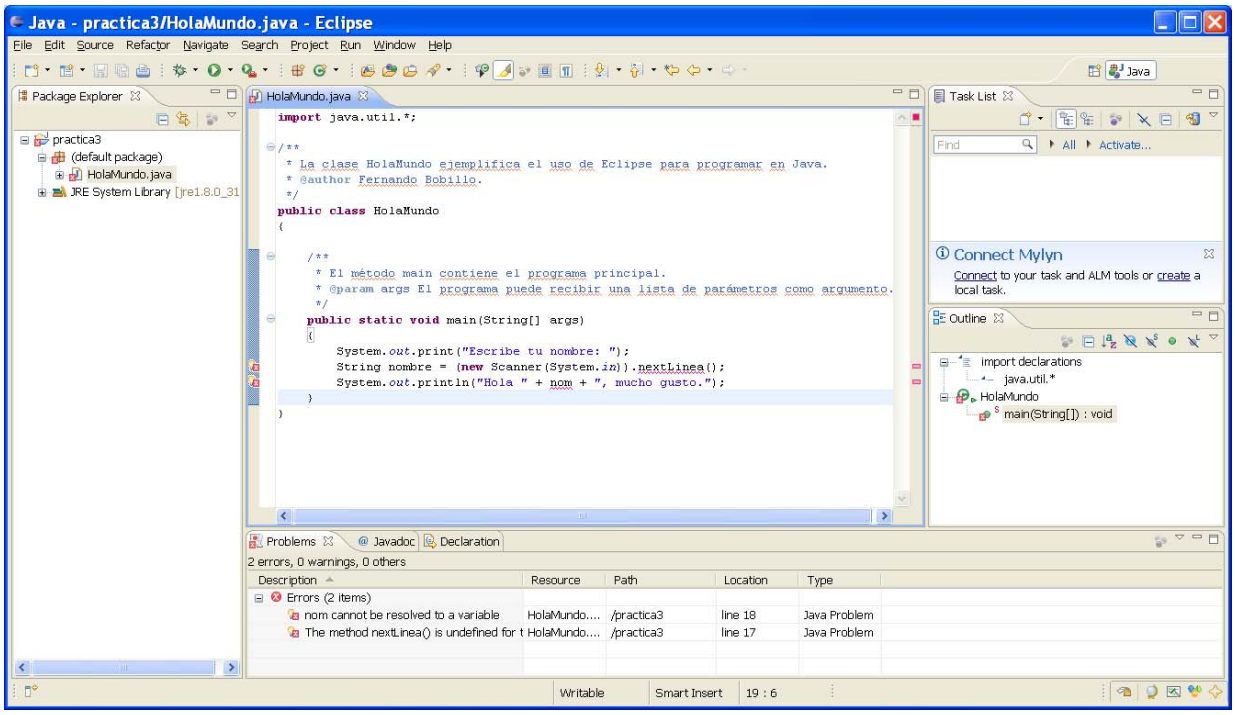


Fig. 7. Aplicación Java con errores

El primer error se produce porque el nombre correcto del método es *nextLine*; el segundo porque se ha escrito mal la variable *nombre*. Corrige ambos errores y recompila con "*Project: Build project"*. Ahora no te deberá mostrar ningún error. El fichero compilado se llama *HolaMundo.class* y se habrá creado en la subcarpeta `bin`.

1.6. Ejecución de aplicaciones

Finalmente, hay que llamar al intérprete de la máquina virtual. En el entorno *Eclipse* esto se hace utilizando la opción de menú "*Run: Run as: Java application"*.

Si trabajásemos directamente con el JDK deberíamos invocar al intérprete (comando java) pasándole como parámetro el fichero compilado sin poner la extensión *.class*. Así pues, en el intérprete de órdenes teclearíamos:

```
java HolaMundo
```

1.7. Depuración de errores

Para depurar los errores de un proyecto, vamos a ejecutarlo paso a paso para comprender mejor el funcionamiento real del programa, tal y como se ilustra en la Figura 8.

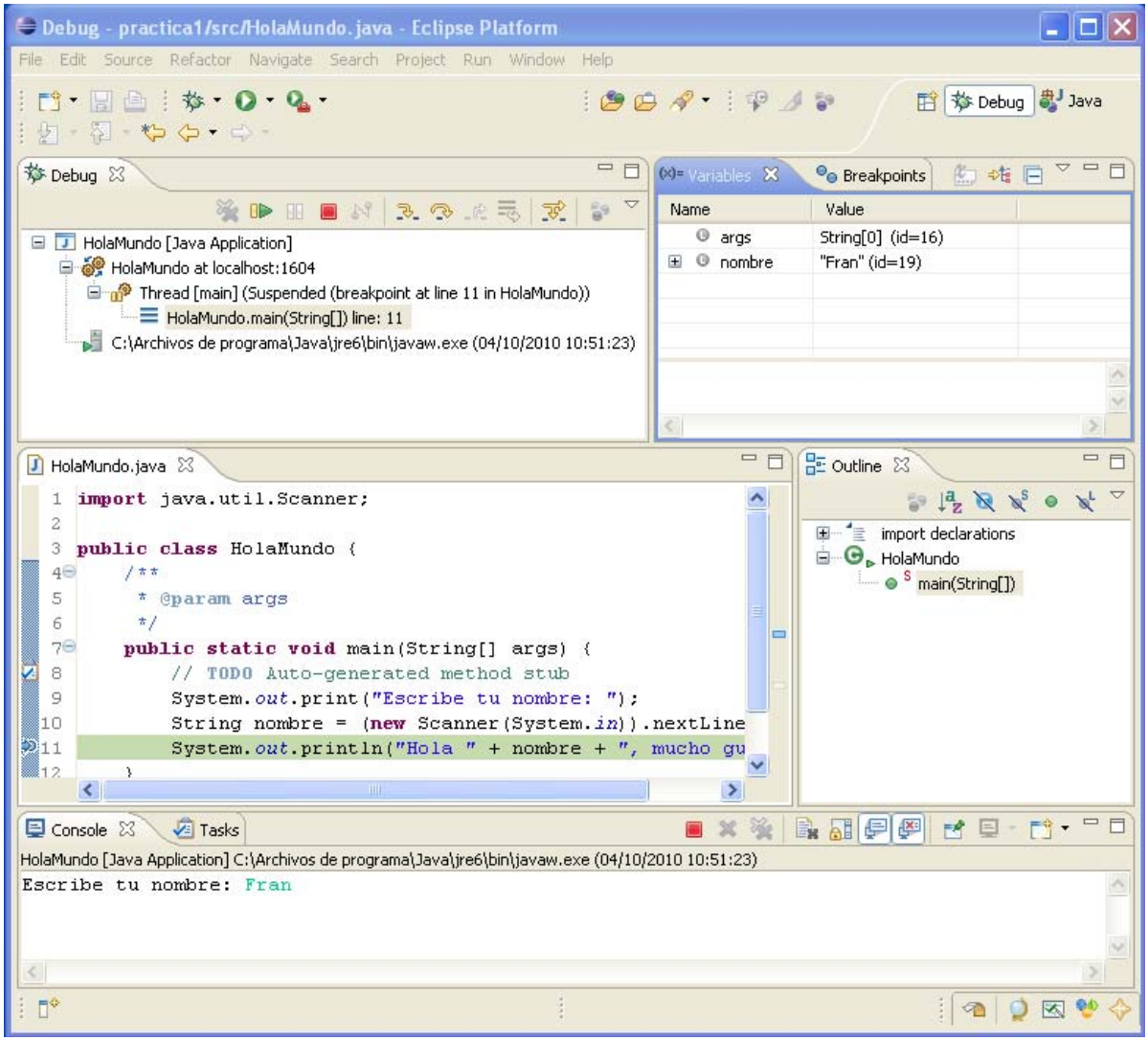


Fig. 8. Depuración de un programa

Para ello, seguimos los siguientes pasos:

- Utiliza la opción de menú "*Run: Toggle line breakpoint"* para establecer (o eliminar) puntos de parada sobre el código que queremos ejecutar paso a paso (por ejemplo, sobre "`n = 10`"). La ejecución paso a paso comenzará en la línea que contenga el punto de ruptura. También se puede utilizar el menú desplegable que aparece al pulsar el botón derecho del ratón sobre el margen izquierdo de la ventana que muestra el código de una clase Java.

- Utiliza la opción "*Run: Debug"* para lanzar la ejecución en modo depuración de errores. Aparecerá una ventana de *debugger* (depurador) con el estado de todas las variables.

- Utiliza *"Run: step into" (F5)* y *"Run: Step over" (F6)* para ejecutar el código instrucción a instrucción. *"Step over"* ejecuta las llamadas a métodos como sentencias simples (sin entrar a ejecutar paso a paso cada instrucción dentro del método).

Para practicar todo lo visto hasta ahora, realiza lo siguiente:

1. Crea una nueva clase llamada *Ejercicio* (comenzando con una letra mayúscula).
2. Pega el siguiente código en la clase creada.

```
public class ejercicio
{
      public static void main(String[] args)
      {
            int n, x, y, z;
            n = 10;
            x = 0;
            y = 1;
            z = 1;

            if (n == 0) || (n == 1)
            {
                  x = 1
            }
            else
            {
                  for (i=2; i<=n; i++)
                  {
                        int s = y + z;
                        x = s;
                        z = y;
                        y = x;
                  }
            }
            System.out.println(Programa terminado);
      }
}
```

3. Soluciona todos los errores de compilación en el programa anterior.
4. Ejecuta paso a paso para observar cual es el valor que toma la variable *x* justo antes de finalizar la ejecución del programa.

1.8. Exportar e importar proyectos

Las opciones de exportar e importar proyectos sirven para transferir proyectos de un ordenador a otro. Por ejemplo, podemos exportar un proyecto Eclipse a una carpeta de ficheros ubicada en una memoria USB y, en otro ordenador, importar la carpeta.

Para **exportar** un proyecto de Eclipse a una carpeta de ficheros, haremos lo siguiente:

- Utilizar la opción de menú "*File: Export"* y elegir *"General: File system"* (ver Figura 9).

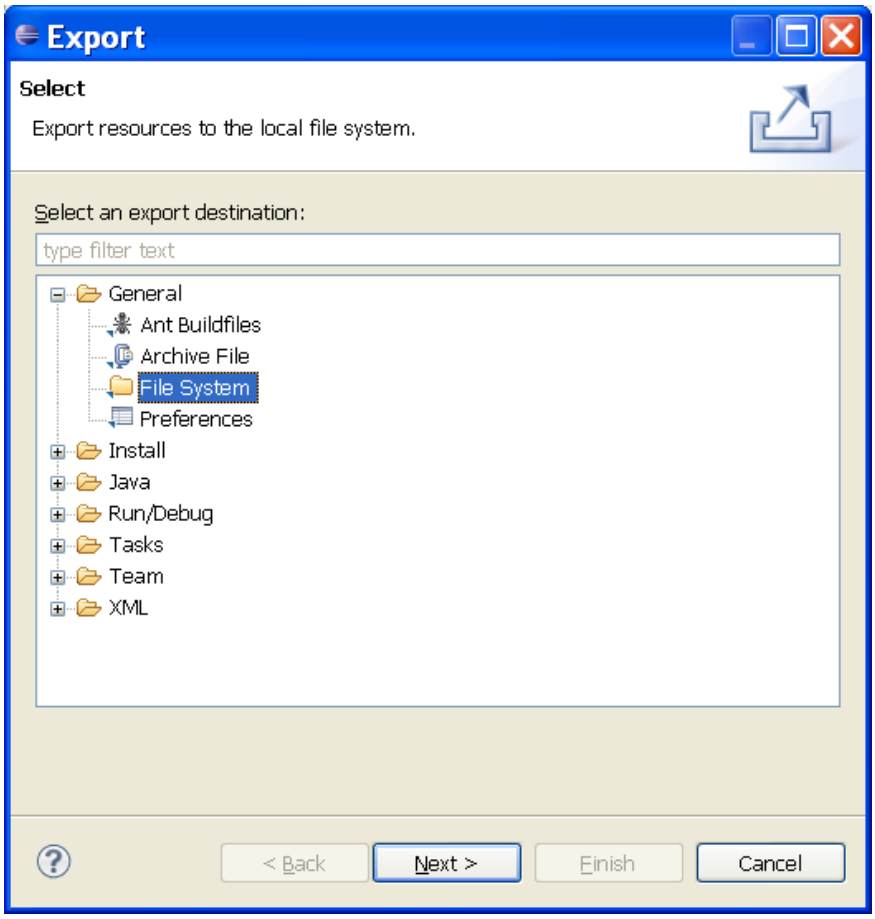


Fig. 9. Exportar proyectos del workspace

- Marcaremos el proyecto a exportar y todos sus elementos (área *"Select the resources to export"*).
- En la zona "*To directory"* se indicará el nombre de la carpeta que vamos a crear.
- Pulsa "*Finish"* para finalizar.

Si ha ido todo bien, habrás generado una carpeta con la estructura de subcarpetas del proyecto actual. Existen otras maneras de exportar proyectos, pero ésta es la más sencilla.

En realidad, no es estrictamente necesario exportar el proyecto, sino que es suficiente con guardar los ficheros *.java* ubicados en la carpeta "*src*".

Para **importar** un proyecto, seguimos los siguientes pasos:

- Copiamos la carpeta de ficheros exportada en el *workspace* del ordenador. Si existe otra carpeta con una versión previa del proyecto (por ejemplo, si estamos practicando la exportación y la importación en el mismo ordenador), deberemos borrarla la versión previa para poder importarla correctamente.

- Utilizamos la opción *"File: Import"* y a continuación seleccionamos *"General: Existing projects into workspace"* (ver Figura 10).

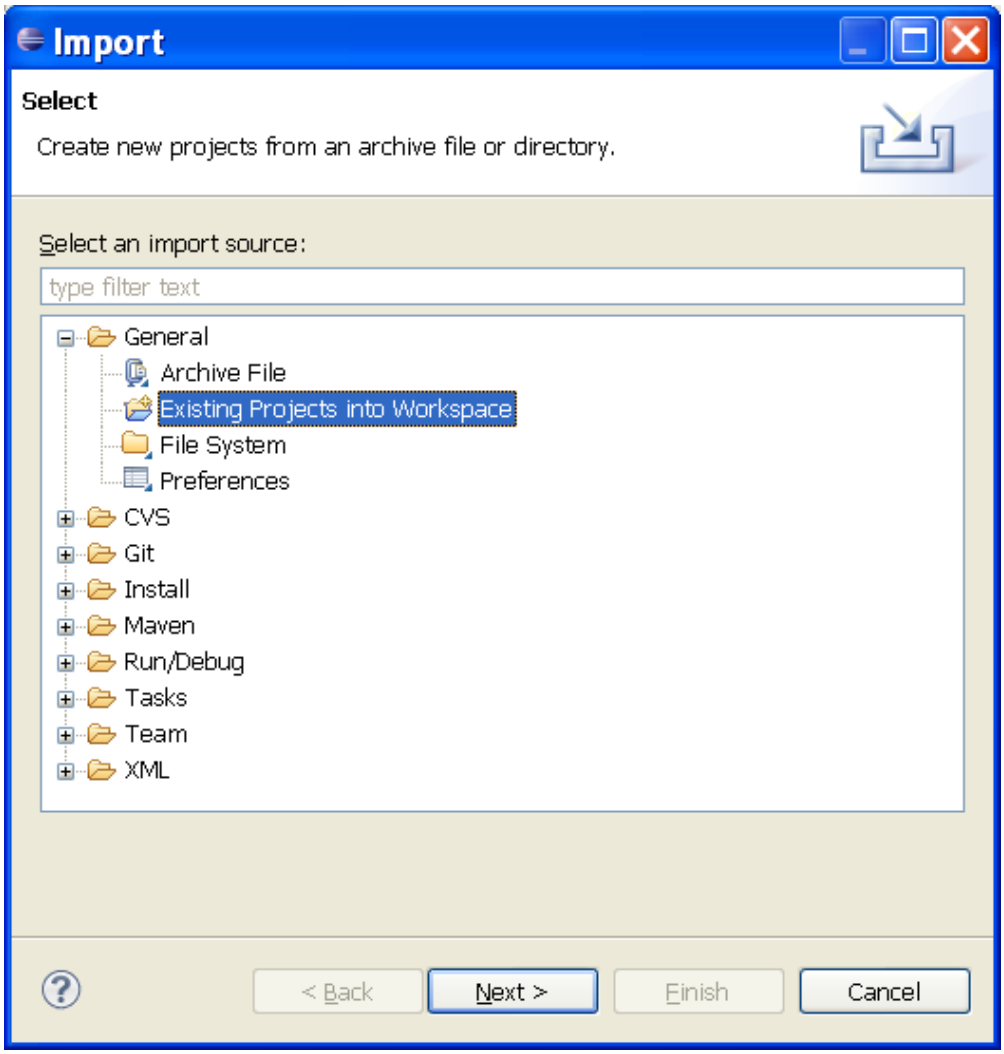


Fig. 10. Importar proyectos del workspace

Otra opción alternativa para importar proyectos es crear un nuevo proyecto a partir de clases Java existentes utilizando la opción *"Import : General : File system"*. De esta manera, podríamos importar únicamente los ficheros *.java* sin necesidad de considerar el resto del proyecto. Esta opción la utilizaremos en las prácticas 8 y 9, por el momento recomendamos utilizar la opción *"General: Existing projects into workspace".*

1.9. Crear y utilizar bibliotecas .jar

Cuando se quiere distribuir una aplicación desarrollada en Java que incluya un número considerable de clases y paquetes, conviene generar una biblioteca. Las bibliotecas en Java son archivos con extensión .jar que comprimen (al estilo de los ficheros *.zip)* un conjunto de clases compiladas (*.class)* organizadas en paquetes. Se puede explorar el contenido de un fichero *.jar* con programas de compresión de ficheros como WinZip.

Para crear una biblioteca en el entorno *Eclipse* debemos sellecionar el proyecto que queremos exportar, utilizar la opción de menú "*File: Export",* seleccionar *"Java: Runnable JAR File"* y seguir los siguientes pasos, ilustrados en la Figura 11:

- En "*Launch configuration*" se elige la clase principal, es decir, la que tiene el método *main* que se ejecutará una vez que ejecutemos el fichero .jar.
- En "*Export destination"* se indica el nombre del fichero que vamos a crear y la ruta en que se va a almacenar dicho fichero.
- Pulsa "*Finish"* para finalizar. Si ha ido todo bien, se habrá generado un fichero .jar.

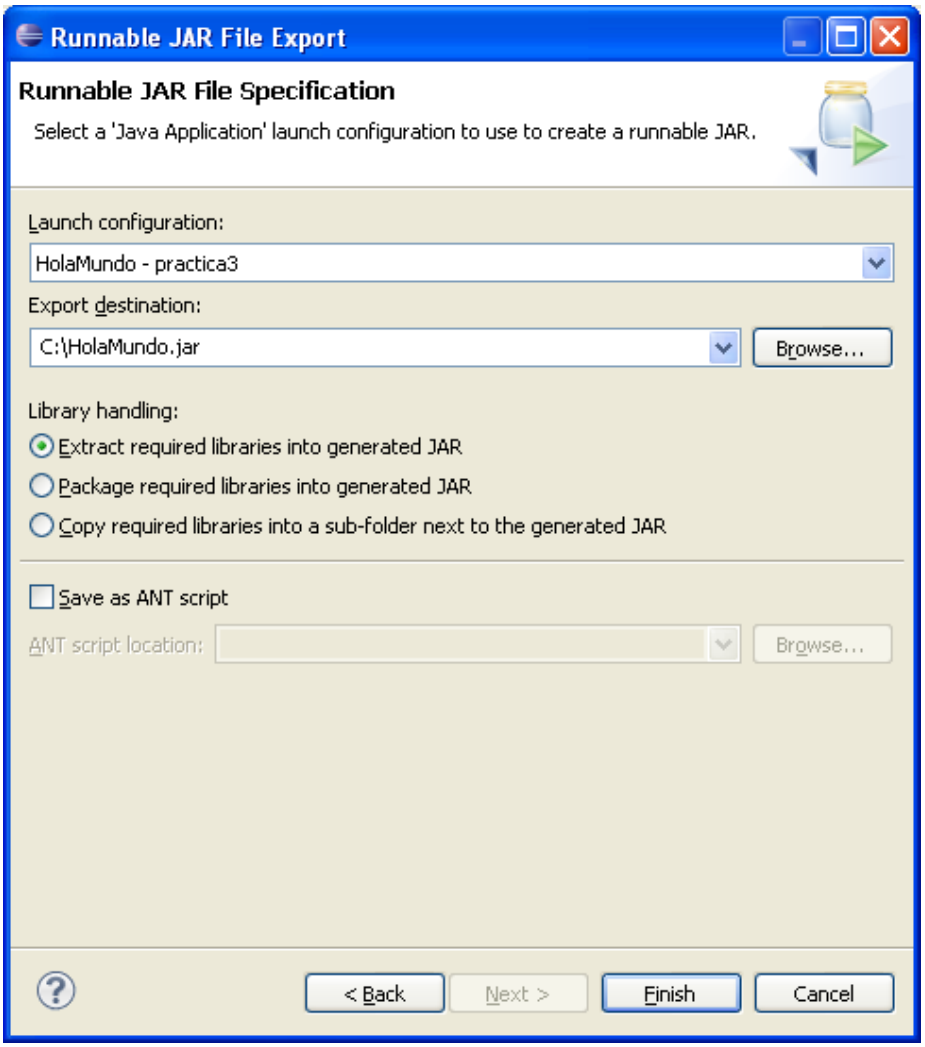


Fig. 11. Construcción de una biblioteca .jar

Si el programa se llama `HolaMundo.jar`, podrás ejecutarlo escribiendo en una ventana de órdenes del sistema operativo:

```
java –jar HolaMundo.jar
```

Para abrir una ventana de órdenes en Windows XP, hay que seleccionar `Inicio - Ejecutar` escribir `cmd` y pulsar Return. Antes de poder ejecutar el .jar con la instrucción anterior, debemos posicionarnos en la carpeta que lo contenga. Para simplificar, guarda el fichero .jar en la carpeta raíz del disco duro `C:` y posteriormente escribe en la ventana de órdenes la siguiente instrucción:

```
cd \
```

1.10. Añadir bibliotecas a un proyecto

Si queremos utilizar alguna de las clases ya definidas en Java, simplemente es necesario añadir una sentencia en nuestro programa para importar la biblioteca correspondiente. Por ejemplo, para utilizar las clases definidas en el paquete `java.util`, nuestro programa ha realizado la sentencia

```
import java.util.*;
```

El proceso de incluir bibliotecas diferentes a las incluidas por defecto en Java puede no ser sencillo, ya que hay varias opciones para hacerlo de una manera correcta. Entre las distintas opciones que tenemos para incluir bibliotecas están:

- Si tenemos un fichero *.jar* con las clases compiladas, nos interesará añadirlo.
- Si tenemos las clases compiladas pero no un fichero *.jar*, nos puede interesar incluir el directorio con clases compiladas.
- Si vamos a utilizar una biblioteca común, es recomendable crear una *variable*, e incluirla como biblioteca.

En este apartado vamos a describir el primer escenario. Para ello, vamos a importar a modo de ejemplo la biblioteca .jar incluida en Moodle, `biblioteca.jar`.

Antes de ello, vamos a escribir en nuestro programa la siguiente instrucción:

```
Practicas.escribeMensaje();
```

Observemos que el compilador informa de un error: la clase `Practicas` no ha sido definida todavía. Para solucionarlo, vamos a importar una biblioteca donde se define dicha clase. Debemos pinchar el botón derecho del ratón sobre el nombre del proyecto y seleccionar *"Properties: Java build path*". Obtendremos una ventana como la de la Figura 12.

A continuación, pinchamos sobre "*Add External JARs*" y seleccionamos el fichero a importar. Una vez hecho esto, podemos comprobar que es posible ejecutar nuestro programa sin ningún tipo de problemas, pues el código necesario se obtiene en el fichero importado.

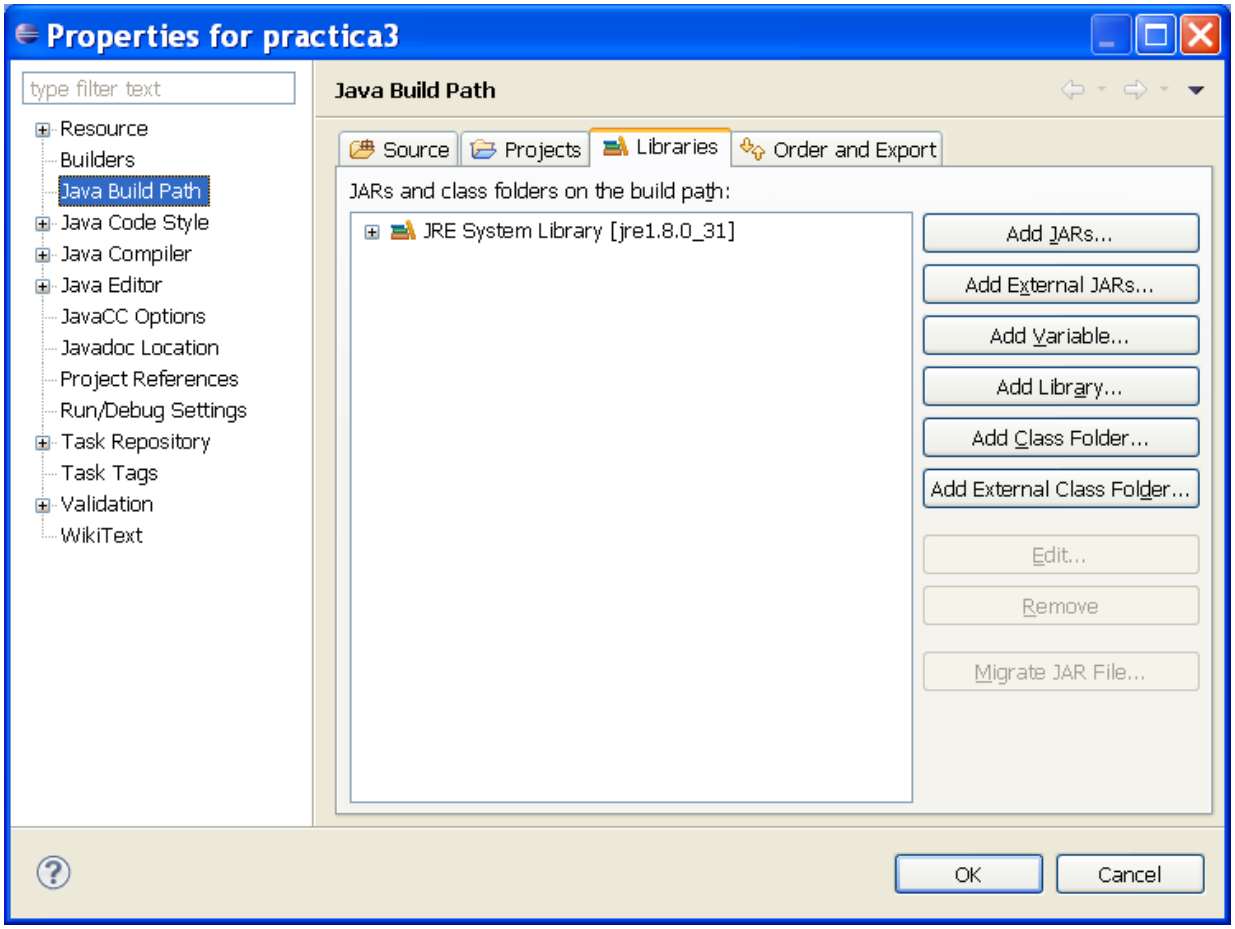


Fig. 12. Añadiendo una biblioteca .jar a un proyecto

1.11. Generar documentación

Para generar la documentación de las clases y paquetes desarrollados se utiliza la herramienta javadoc, incluida en la JDK**.** Desde el entorno de *Eclipse,* se facilita la utilización de esta herramienta a través de la opción de menú *"Project: Generate Javadoc"* (Figura 13).

- En primer lugar, seleccionar la ubicación del ejecutable javadoc con el botón "*Configure*". Normalmente, Eclipse detectará automáticamente su ubicación y rellenará este campo por nosotros, por lo que no necesitaremos hacer nada.

- A continuación se debe configurar el directorio de salida de la documentación utilizando la caja de texto "*Destination"* y el botón "*Browse"* asociado.

- Por último, se pulsará el botón "*Finish"* para finalizar y generar la documentación. La documentación podrá abrirse con cualquier navegador HTML.

Probadlo para la clase `HolaMundo`.

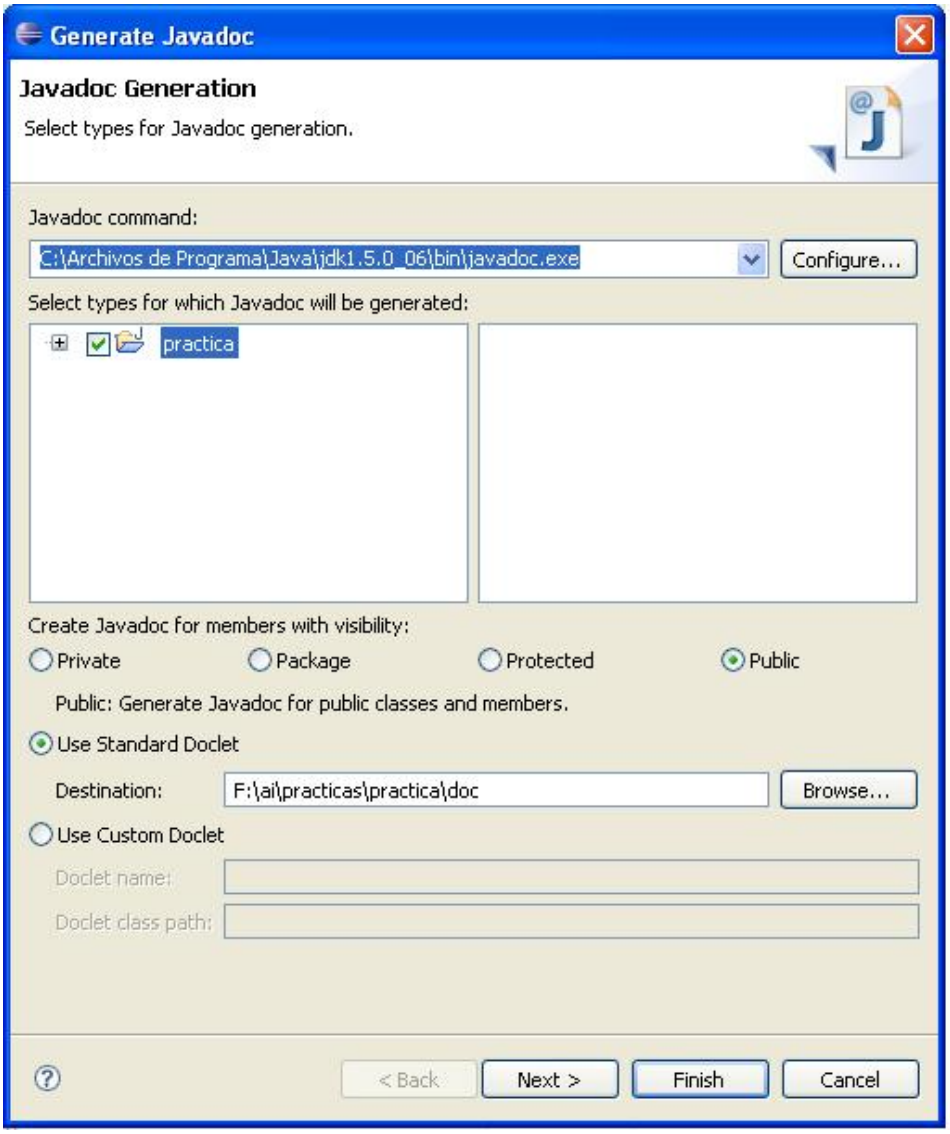


Fig. 13. Generación de documentación con la herramienta javadoc

1.12. Acceder a la documentación del JDK

Java proporciona una amplia y detallada documentación acerca de las herramientas que incluye la JDK así como la especificación del API (*Application Programming Interface*), es decir, la documentación de las bibliotecas disponibles por defecto en Java. Esta documentación está accesible en Internet, en la página http://docs.oracle.com/javase/7/docs (Figura 14). La documentación del API está en http://docs.oracle.com/javase/7/docs/api (Figura 15). Esta documentación puede descargarse para poder consultarla sin necesidad de acceder a Internet; el material descargado puede visualizarse con cualquier navegador web.

Modifica la clase `Ejercicio` para que cada vez que se ejecute escriba por pantalla un número diferente. Para ello, añadir en la línea 25 la siguiente instrucción, reemplazando el texto en rojo (???) por el método de la clase `Math` que permite calcular números aleatorios; puede buscarse en http://docs.oracle.com/javase/7/docs/api/index.html?java/lang/Math.html

```
System.out.println( Math.???() );
```

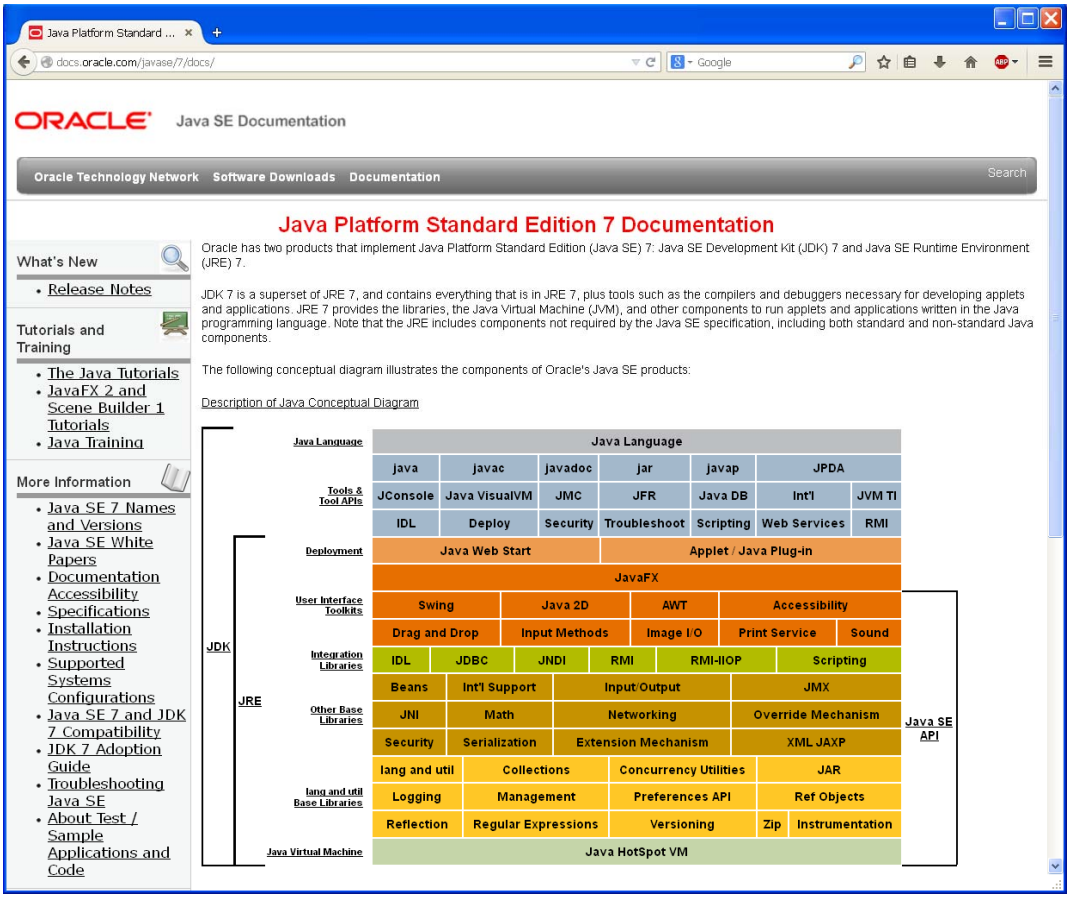


Fig. 14. Documentación del JDK

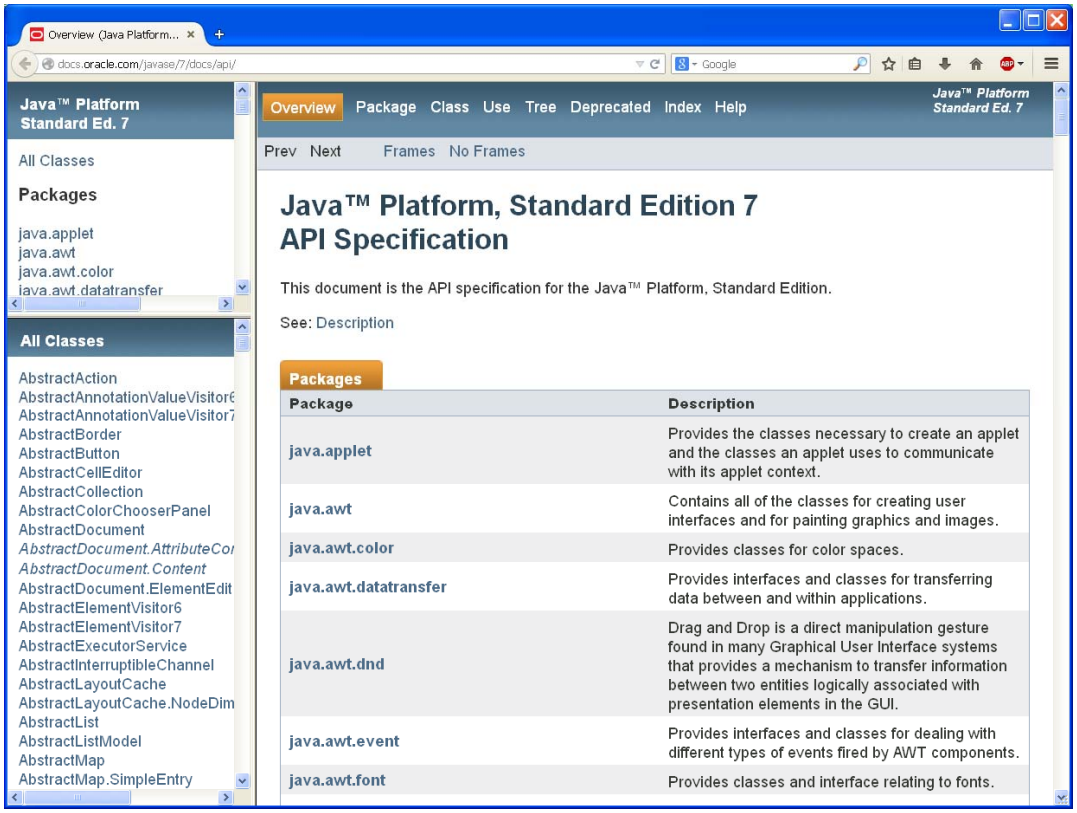


Fig. 15. Documentación de la API de Java

Apéndice 2. Argumentos en la línea de órdenes

Es posible pasar argumentos a una aplicación Java en el momento de ejecución, a través de parámetros pasados por línea de órdenes. Para simular esta funcionalidad en Eclipse, se debe proceder como se explica a continuación. Primero, seleccionar la clase que queremos ejecutar en el panel "*Package explorer*", después la opción del menú desplegable *"Run as: Run configurations"* y, finalmente, la pestaña "*Arguments"* e introducir los argumentos de la línea de órdenes en el área de texto *"Program arguments"* (ver Figura 16).

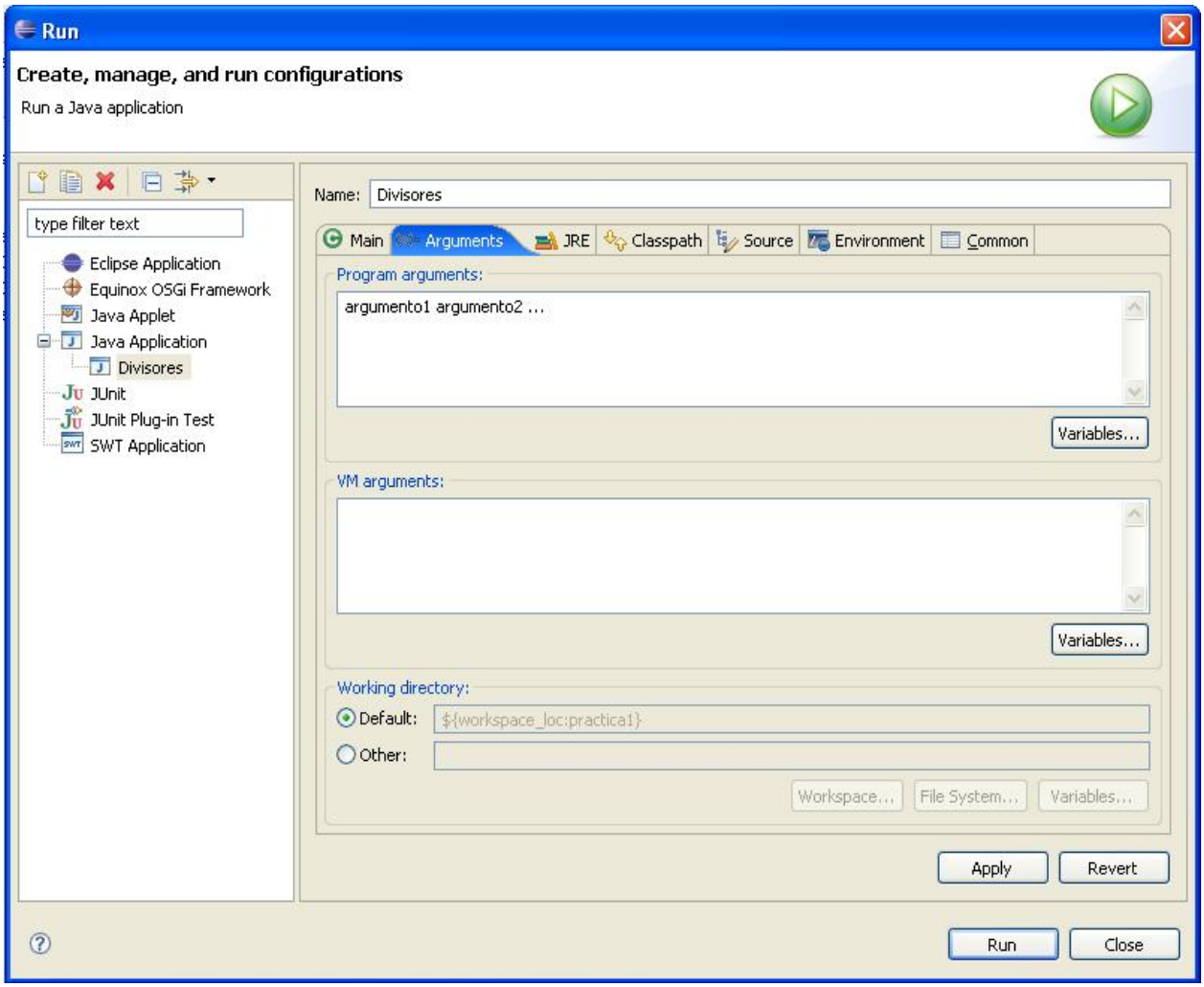


Fig. 16. Añadir argumentos en la línea de comandos

Apéndice 3. Instalación de Eclipse y JDK

Finalmente, explicaremos el proceso a seguir para instalar el software necesario para realizar las prácticas de Informática en los ordenadores personales. Esto requiere dos pasos:

A) Instalación del Java Software Development Kit (JDK)

Para comenzar, descargar la última versión de JDK 7 (no se recomienda utilizar la versión más reciente, JDK 8) de la página web de Oracle. Para ello, seguir los siguientes pasos:

- Acceder a la página web http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html y buscar un apartado con un nombre que comienza por "Java SE Development Kit" (ver Figura 17). En el momento de escribir el guión, el nombre exacto es "Java SE Development Kit 7u75", pero en el momento de descargar el enlace podría existir una versión más reciente. Si este enlace no funciona, acceder a la web http://www.oracle.com/technetwork/java/javase/downloads/index.html y seleccionar alguna de las versiones de "Java Platform (JDK)".

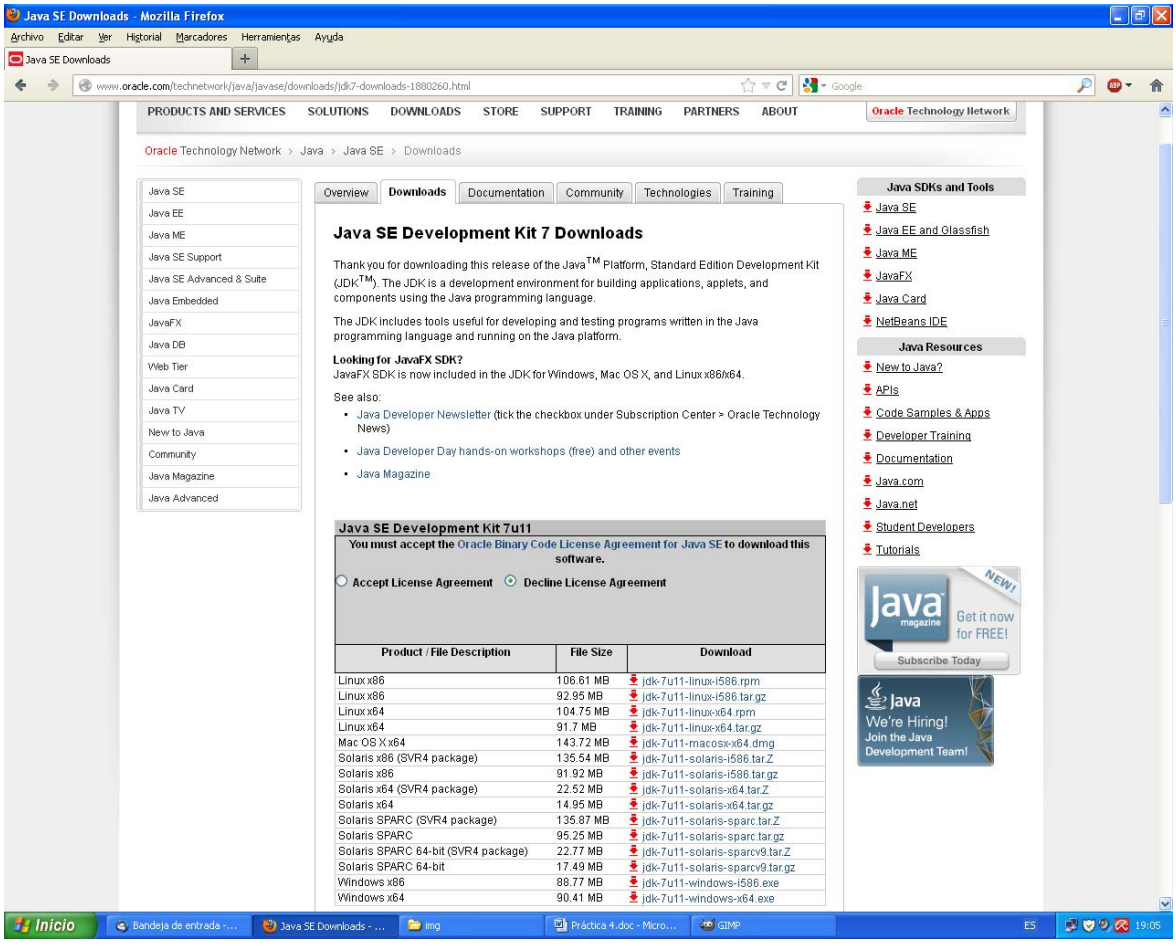


Fig. 17. Página web de descarga del JDK

- Seleccionar la opción "*Accept License Agreement*" y pinchar sobre el enlace correspondiente a nuestro sistema operativo. Por ejemplo, para sistemas operativos Windows de 64 bits, seleccionar el fichero jdk-7u75-windows-x64.exe.

- Una vez descargado el fichero, ejecutarlo. Los pasos del instalador no tienen mayor dificultad; en cada una de las pantallas puede seleccionarse la opción de continuar (*"Next"*) sin modificar las opciones por defecto.

B) Instalación del entorno Eclipse

- Visitar la página web http://www.eclipse.org/downloads.
- Localizar el primer enlace de la lista, *"Eclipse IDE for Java Developers (99 MB)"*. A la derecha, aparecerá uno o varios enlaces dependiendo de nuestro fichero operativo. Seleccionaremos el correspondiente.
- Aparecerá una pantalla como la de la Figura 18. Pinchamos en la flecha verde y se descargará un fichero comprimido en formato *.zip*.
- Alternativamente, podemos acceder a versiones más antiguas mediante el enlace "*Older Versions*" en la página inicial de descargas de Eclipse. La versión instalada en las aulas de prácticas es *"Eclipse Helios SR2 Packages (v 3.6.2)"*, a la que se puede acceder directamente desde el enlace http://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/helios/SR2/eclipse-java-helios-SR2-win32.zip.
- Para instalar Eclipse, basta con descomprimir el fichero comprimido utilizando cualquier programa de compresión de ficheros, como por ejemplo *WinZip*. Una vez descomprimido, Eclipse ya está listo para utilizarse.

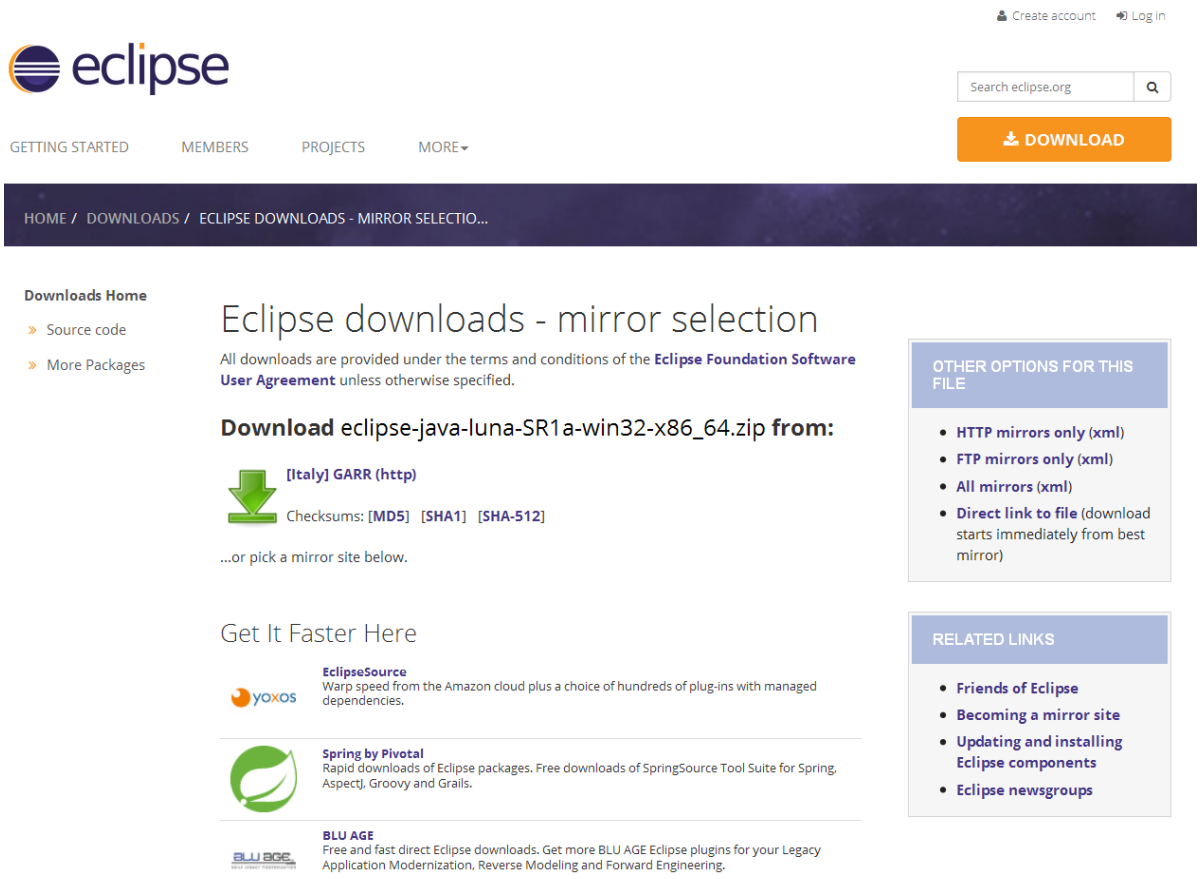


Fig. 18. Página web de descarga de Eclipse