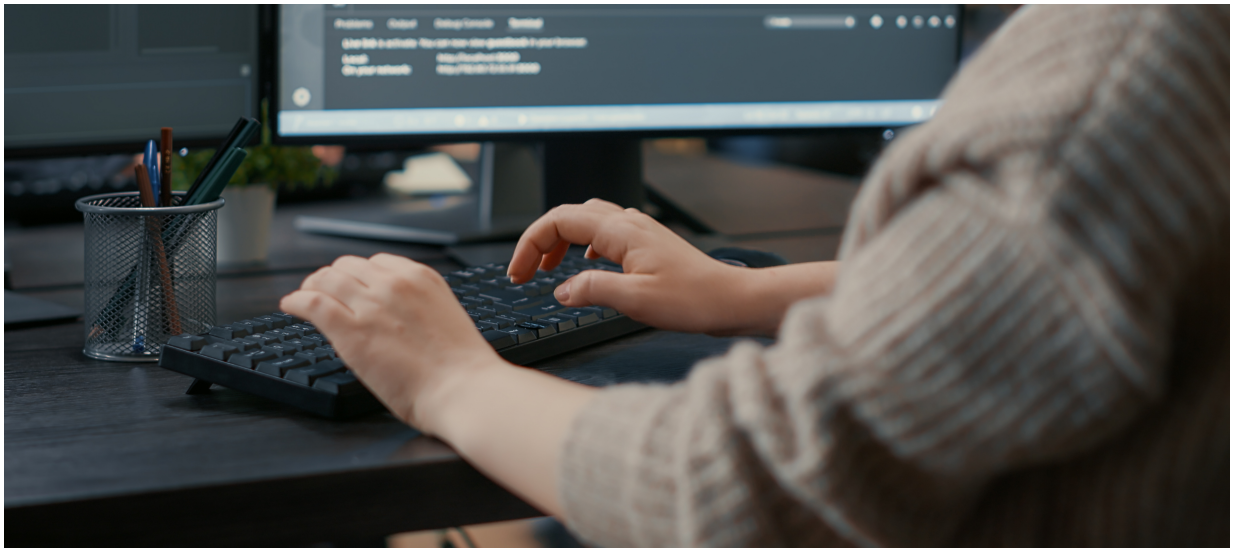
INFORMÁTICA (30717)

# Práctica 2: Un entorno de programación en Java

Grado en Estudios en Arquitectura



Universidad de Zaragoza

Material de prácticas del curso 2025–2026

Universidad Zaragoza

# Tabla de contenidos

# 1. Objetivos de la práctica

Los objetivos de la segunda práctica de la asignatura son los siguientes:

- Familiarizarse con el ordenador y los elementos básicos de su sistema operativo.
- Familiarizarse con el entorno de desarrollo Java Eclipse.

# 2. El entorno de desarrollo Eclipse

Para facilitar el desarrollo de programas, existen entornos que permiten trabajar más cómodamente, tales como NetBeans, Eclipse o IntelliJ. Nosotros recomendamos el uso de ***Eclipse***, un entorno de desarrollo de libre distribución.

Inicia *Eclipse* haciendo doble clic sobre el ejecutable *Eclipse* en la carpeta `software_disponible`, o bien buscando *Eclipse* en la barra de tareas. Para instalar esta aplicación en otros ordenadores puede consultarse el Apéndice. Al iniciar *Eclipse* aparecerá la pantalla de bienvenida que se muestra en la Figura 1.

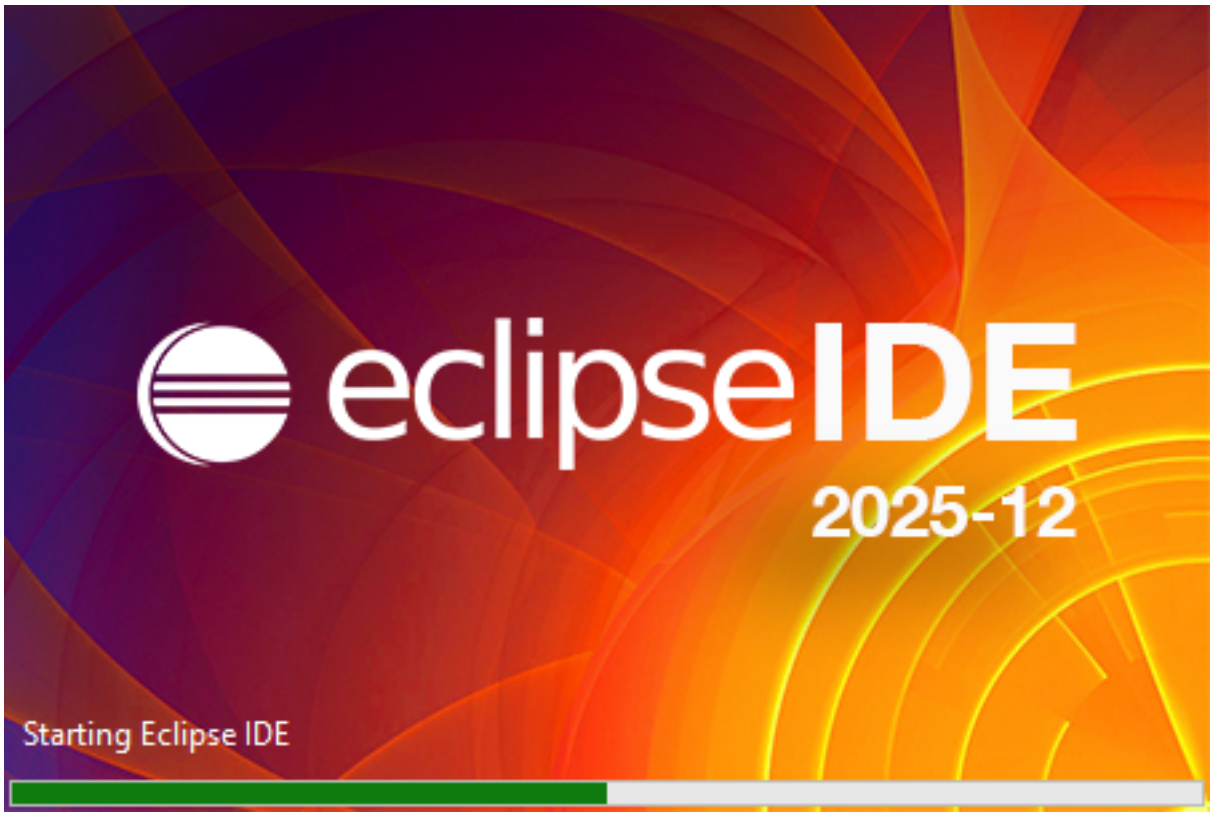



**Figura 1:** Pantalla de bienvenida.

Para poder comenzar a trabajar con el entorno de desarrollo será necesario definir un espacio de trabajo o *workspace*, tal y como se muestra en la Figura 2. Simplemente se trata de una carpeta donde guardaremos nuestros proyectos. Por ejemplo, se puede establecer el directorio `C:/PracticasARQ` como directorio de trabajo.

En los siguientes subapartados aparecen solo algunas de las características de *Eclipse* o recomendaciones acerca de su uso. Si tienes más dudas sobre cómo utilizar *Eclipse*, deberás consultar la ayuda en línea (opción de menú `Help > Help Contents`), o bien accediendo a su página web, donde se encuentra también toda la documentación del programa.

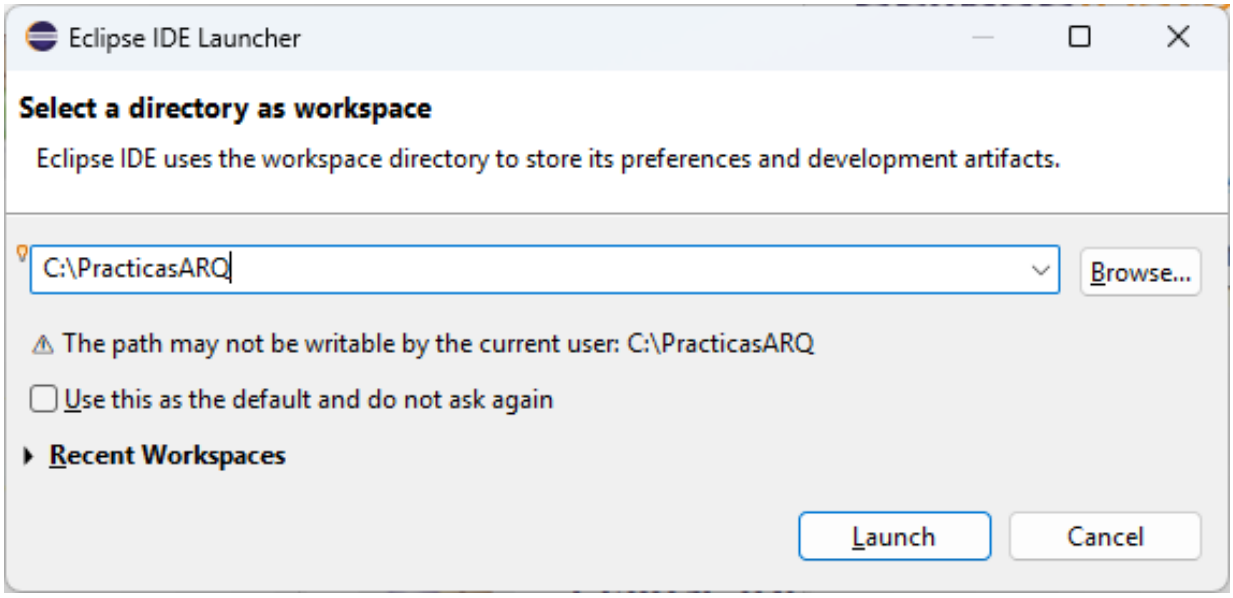



**Figura 2:** Configuración del directorio de trabajo.

## 2.1. Pantalla de bienvenida

Al abrir *Eclipse*, nos aparecerá una ventana de bienvenida (Figura 3) con varias opciones, de las cuales cabe destacar las siguientes:

**Overview**. Proporciona un vistazo general de las características principales del entorno de desarrollo, sobre todo para familiarizarse con la interfaz de usuario.

**Tutorials**. Otra forma de conocer todas las características de Eclipse es ejecutando alguno de los tutoriales de manejo del entorno de desarrollo disponibles.

**Samples**. Una manera de conocer cómo funciona la herramienta es ejecutar algunos de los ejemplos que Eclipse pone a nuestra disposición.

**What's new**. Para los que han utilizado con anterioridad la herramienta *Eclipse*, esta utilidad describe las nuevas características que proporciona la versión instalada frente a versiones anteriores.

**Workbench**. Usando esta opción se accede directamente al espacio de trabajo de Eclipse. Se encuentra arriba a la derecha, junto a un texto que dice *Hide*. Esta es la opción que utilizaremos para acceder al entorno de desarrollo cada vez que iniciemos Eclipse, una vez que ya nos hayamos familiarizado con la herramienta.

Dentro de las opciones que se muestran en la pantalla de bienvendia encontramos *Create a Java project*. Utilizaremos esta opción cada vez que queramos crear un nuevo proyecto, por ejemplo, para cada práctica de programación en Java de este curso. Para esta práctica, podemos crear un nuevo proyecto al que llamaremos `practica2`.

Si cerramos la pestaña de bienvenida (*Welcome*, arriba a la izquierda), nos conduce al entorno de trabajo que se explica en la siguiente sección.

## 2.2. Entorno de trabajo

El **entorno de trabajo o *workbench*** nos proporciona una interfaz de usuario intuitiva, bien estructurada y sencilla de utilizar, donde están bien definidas las distintas zonas de trabajo. En la Figura 4 se puede observar la interfaz de usuario de *Eclipse*.

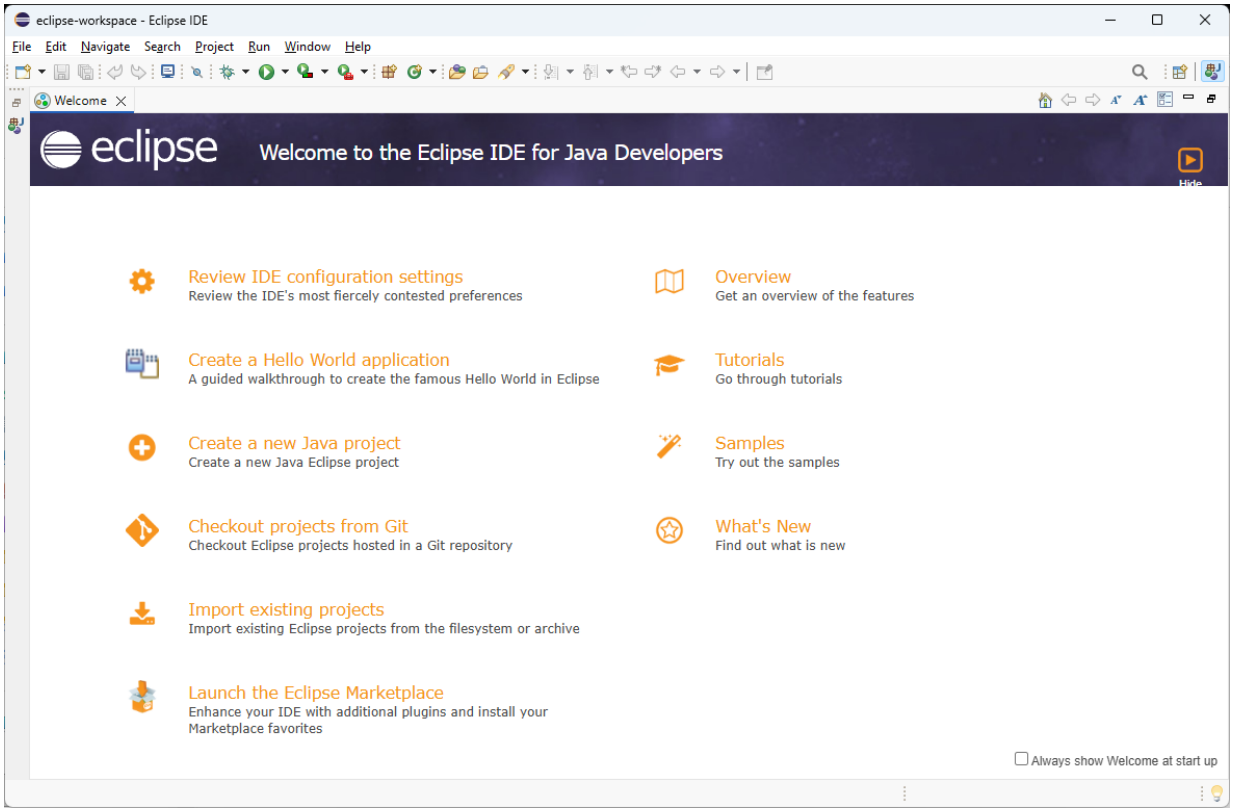



**Figura 3:** Pantalla de bienvenida.

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

- En la **zona izquierda**, se encuentra un menú (***Package Explorer***) que agrupa los componentes de cada uno de los proyectos abiertos.
- En la **zona central** de la pantalla se encuentra un **editor**, que sirve para ver y modificar los ficheros Java con los que trabajamos.
- En la **zona de la derecha**, hay un resumen (*Outline*) que nos indica la **estructura del fichero activo** en el editor (clases, métodos, variables, etc.).
- En la **parte inferior del editor** se encuentran algunas utilidades que nos permiten conocer el estado del proyecto, incluyendo **errores** (pestaña *Problems*) y la **salida** obtenida en los programas ejecutados (pestaña *Console*).

## 2.3. 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 un proyecto para compilar y ejecutar programas individuales con una configuración concreta.

Los proyectos se crean a través de la opción `File > New > Java Project`. Para crear el nuevo proyecto hay que dotarlo de un nombre identificativo (por ejemplo, `practica2`). Dicho proyecto se guarda en un directorio del equipo, dentro del *workspace* seleccionado al lanzar *Eclipse*. Conviene seleccionar la opción ***Create separate source and output folders***. En la Figura 5 se pueden observar todas las propiedades que se deben definir en la creación del proyecto.

Teniendo en cuenta que nuestro entorno de trabajo se encuentra en `C:\practicasARQ`, el nuevo proyecto que has creado se encuentra en la carpeta:

```
C:\practicasARQ\practica2
```

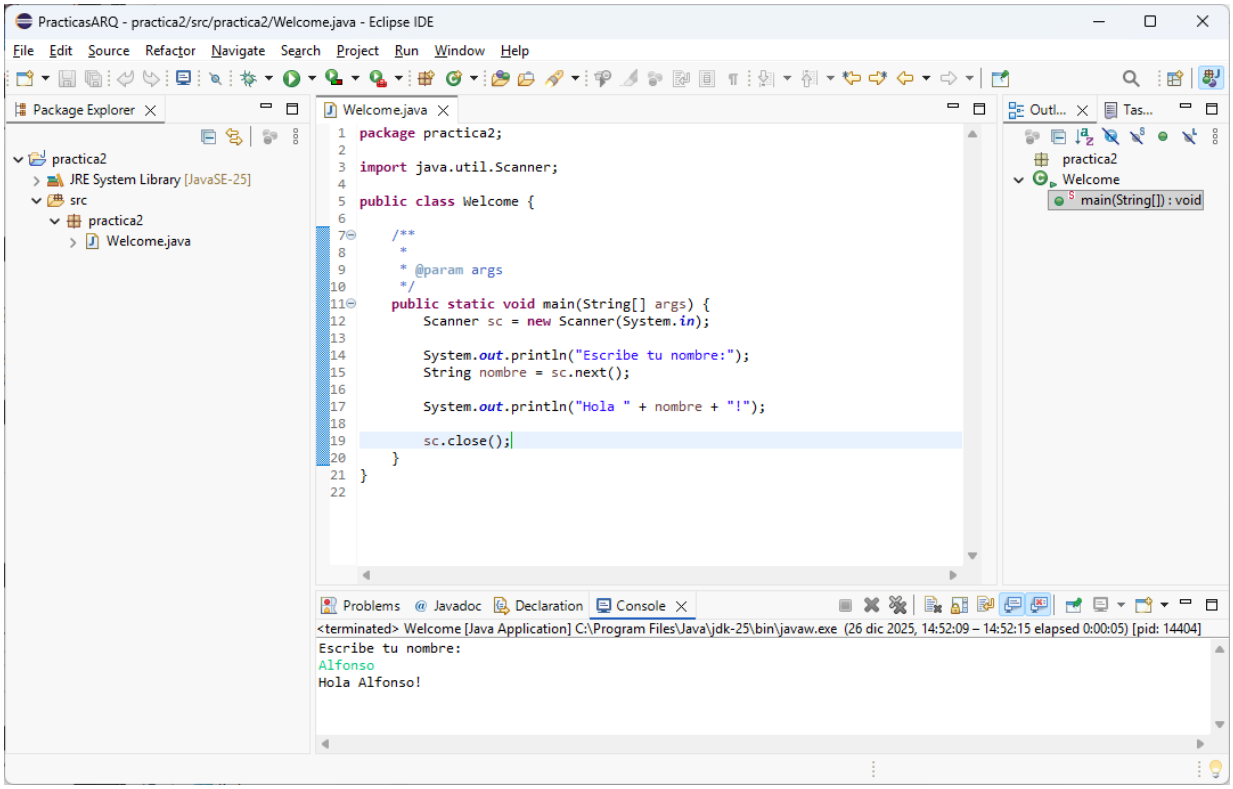



**Figura 4:** Espacio de trabajo de *Eclipse IDE*.

Abre dicha carpeta en el explorador de archivos del sistema operativo y observa que se han creado dos subcarpetas. En `src` se almacenarán los ficheros con el código en lenguaje Java (*source*), mientras que en la carpeta `bin` se almacenan los ficheros binarios (*binaries*).

> Configuración del proyecto
>
> Recuerda **desmarcar la opción *Create module-info.java*** y **marcar la opción *Create separate source and output folders*** para que se cree una carpeta para el código fuente y otra para los ficheros binarios.

## 2.4. Crear una aplicación

Los lenguajes orientados a objetos, como Java, giran en torno al 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`, cuyo nombre es exactamente igual que el de la clase. Por ejemplo, en un fichero `HolaMundo.java`, definiremos una clase *HolaMundo*.

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, ejecutándose por tanto el método *main* de la clase. La Figura 7 muestra la configuración de nuestra nueva clase Java.

Fijaos en que **no hemos indicado el nombre de un paquete, por lo que la clase formará parte un paquete por defecto** (***default***). En futuras sesiones de prácticas hablaremos de qué es un paquete, y cómo crearlo para estructurar las clases que creemos en cada proyecto.

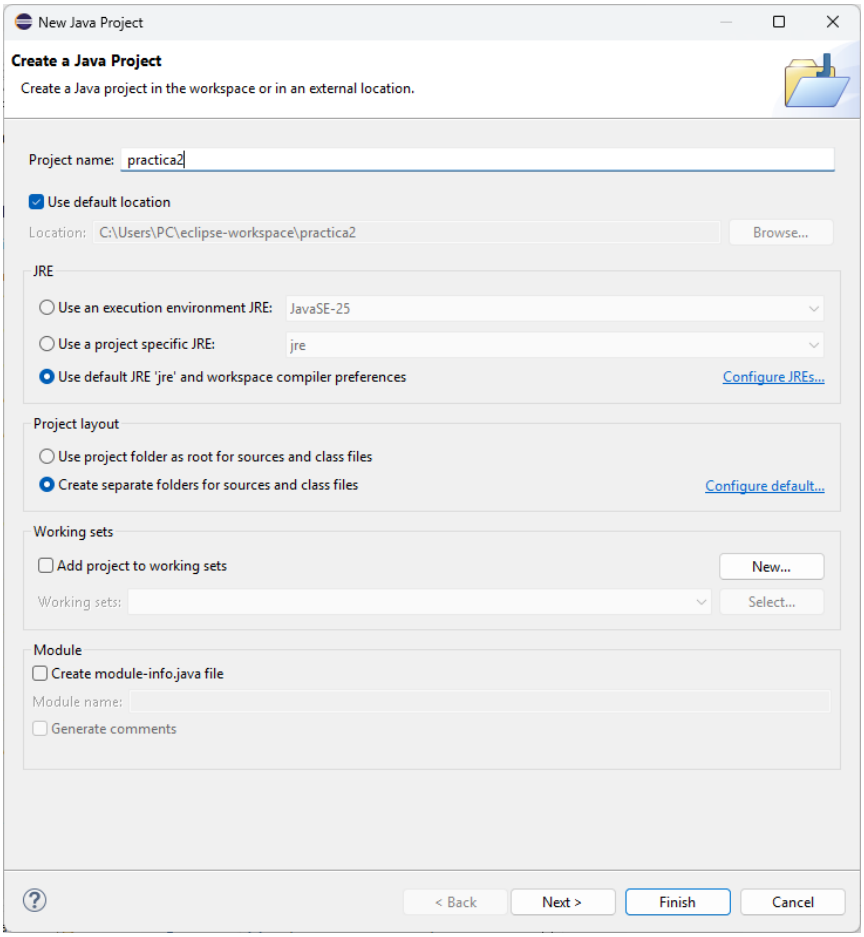



**Figura 5:** Propiedades que debemos configurar al crear un nuevo proyecto.

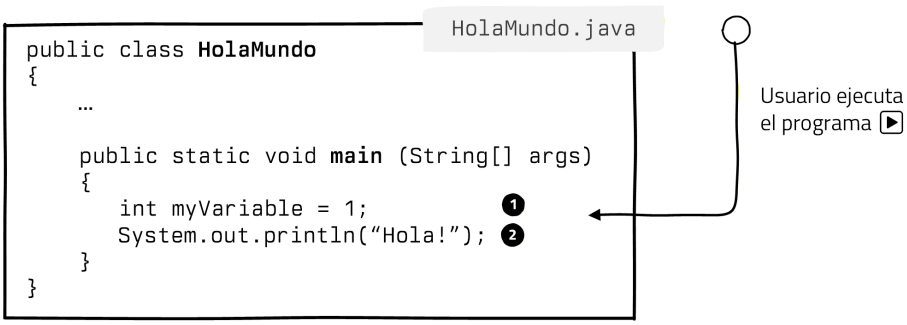



**Figura 6:** Ejemplo de estructura de una clase Java.

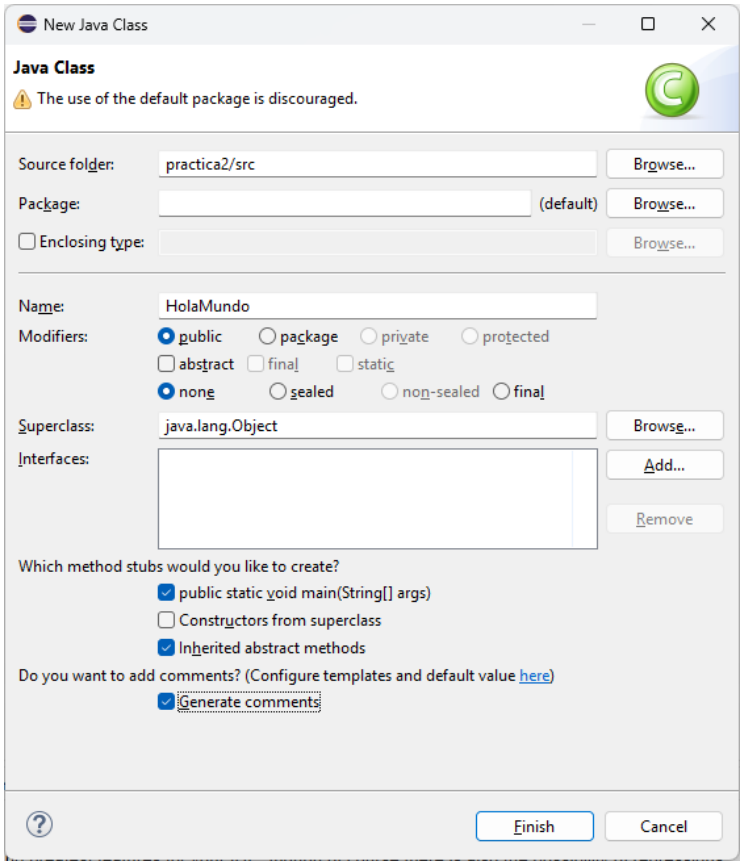



**Figura 7:** Creación de una nueva clase de Java en *Eclipse IDE*.

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

```
import java.util.*;

/**
 * La clase HolaMundo ejemplifica el uso de Eclipse para programar en Java.
 * @author Fernando Bobillo, Alfonso López, Ignacio Huitzil.
 */
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.");
    }
}
```

**Errores de compilación**

El código anterior contiene algunos **errores introducidos intencionadamente** para que se produzcan **errores de compilación**.

El editor de *Eclipse* es un editor típico del entorno Windows: se puede copiar, cortar, pegar, buscar y sustituir. Tiene un sistema de tabulación inteligente que facilita la tarea de escritura. Se puede conseguir aumentar o disminuir la **tabulación de una porción de código seleccionando** dicha porción y pulsando tabulador (`TAB`) o mayúsculas-tabulador (`Shift + TAB`), respectivamente. Además, el editor de *Eclipse* tiene un sistema de **autocompletado** que se activa pulsando `Ctrl + Space` y que facilita la escritura de código, por ejemplo, para completar el nombre de una clase o método a partir de las primeras letras escritas.

**Guarda los cambios** que hemos introducido pulsando los iconos con disquetes que hay en la barra de herramientas superior, pulsando `Ctrl-S` o bien con `File > Save`.

## 2.5. Compilación de aplicaciones

Para compilar un programa, es necesario **traducir el código fuente escrito en lenguaje Java a un código binario** que pueda ser interpretado por la máquina virtual de Java. En el entorno *Eclipse*, la compilación se realiza **automáticamente cada vez que se guarda un fichero**, aunque también es posible compilar manualmente utilizando la opción de menú `Project > Build Project`. Si el programa contiene errores de compilación, *Eclipse* nos lo indicará en la zona inferior del editor, en la subpestaña ***Problems*** (ver Figura 9). En esta subpestaña se nos informará sobre el proceso de compilación indicando los posibles errores. **Puedes seleccionar cada error mediante doble click, y se indicará dentro del código dónde se encuentra el error**.

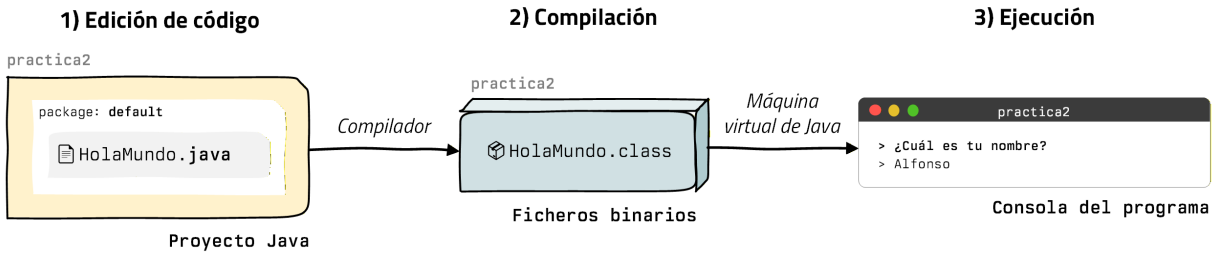



**Figura 8:** Proceso de ejecución de un programa.

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

```
javac HolaMundo.java
```

**Resolución de errores de compilación**

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 guardando el fichero (`Ctrl-S`) o utilizando la opción `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`.

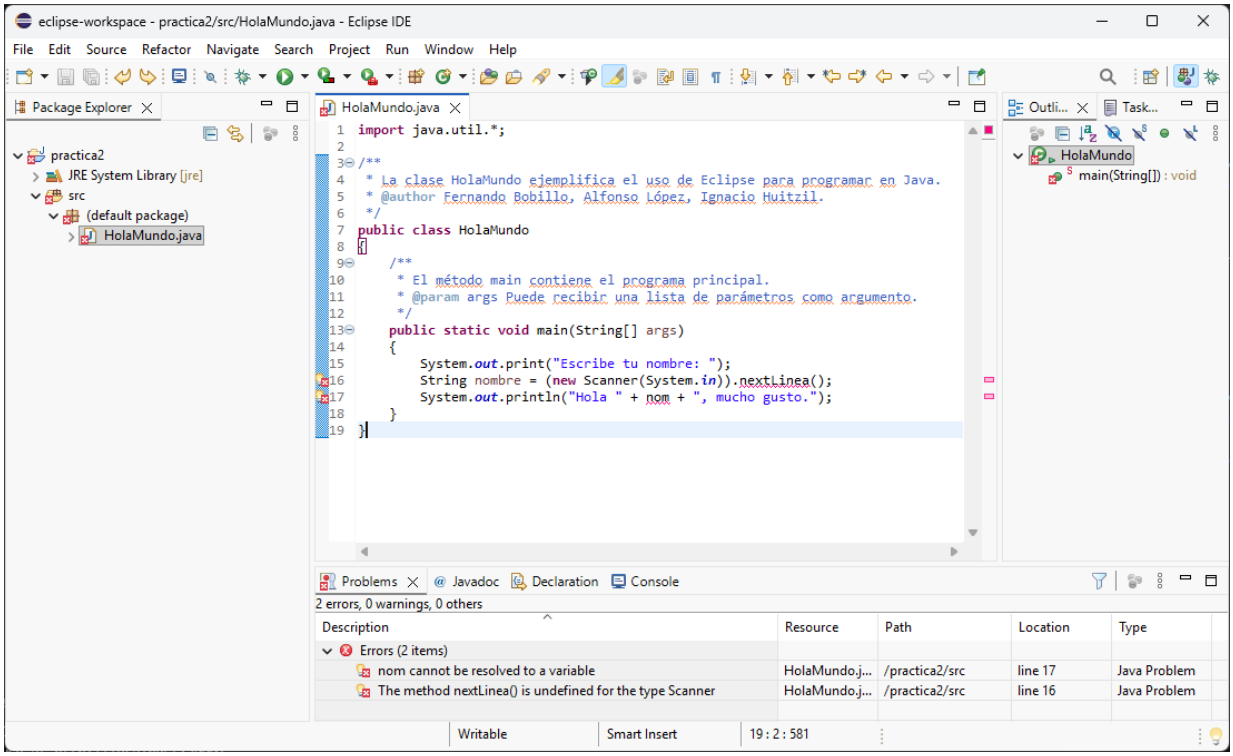



**Figura 9:** Compilación de un programa con errores.

## 2.6. Ejecución de aplicaciones

Una vez que el programa se ha compilado correctamente, es posible ejecutarlo. Para ello, en el entorno Eclipse se utiliza la opción de menú **`Run > Run as: Java Application`** o bien el icono de **ejecutar** en la barra de herramientas superior. Si el programa se ejecuta correctamente, aparecerá una nueva pestaña llamada *Console* en la parte inferior del editor, donde se mostrará la salida del programa.

> **Ejecución sin un entorno de desarrollo**
>
> 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
> ```

## 2.7. Depuración de errores

En el proceso de desarrollo de un programa, es habitual que se produzcan errores de ejecución. Para solucionar estos errores, es necesario **depurar el programa**, es decir, **ejecutar el programa paso a paso** para observar el estado de las variables y comprender mejor el funcionamiento real del programa (ver Figura 10).

### Ejecución paso a paso

La opción `Step into` ejecuta el código *instrucción a instrucción*, mientras que `Step over` *ejecuta las llamadas a métodos como sentencias simples* (sin entrar a ejecutar paso a paso cada instrucción dentro del método).

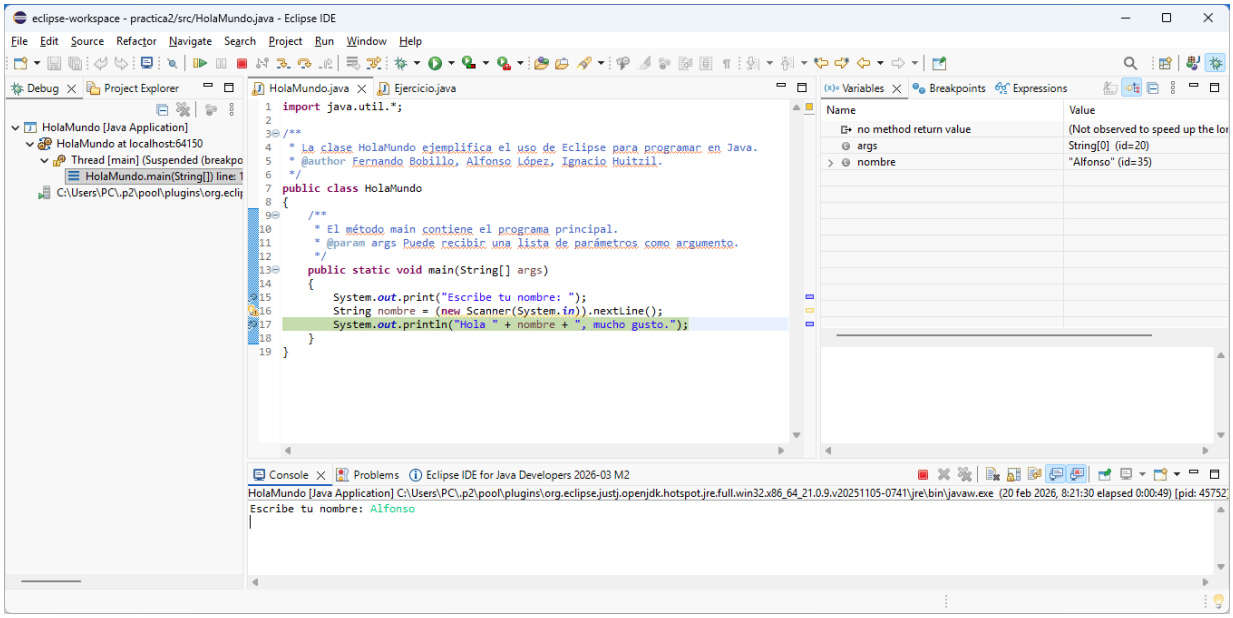



**Figura 10:** Depuración de un programa en Eclipse.

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. 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, justo al lado del número de línea.
- Utiliza la opción `Run > Debug`, o `Debug` en la barra de herramientas superior, 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, haz lo siguiente:

- Crea una nueva clase llamada **Ejercicio** (comenzando con una letra mayúscula).
- 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);
    }
}
```

- Soluciona todos los errores de compilación en el programa anterior.
- Ejecuta el programa paso a paso para observar cuál es el valor que toma la variable `x` justo antes de finalizar el programa.

## 2.8. Exportar 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 una memoria USB y, en otro ordenador, importar la carpeta.

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

- Utiliza la opción de menú `File > Export` y elegir `General > File system` (ver Figura 11).
- Marcaremos el proyecto a exportar y todos sus elementos (carpetas en la parte izquierda, ficheros concretos en la parte derecha). En general, sólo se suele exportar la carpeta `src`, dado que los ficheros binarios (*bin*) se generan a partir de la compilación.
- 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 esta es la más sencilla. Por otro lado, no es estrictamente necesario exportar el proyecto, sino que es suficiente con guardar los ficheros `.java` ubicados en la carpeta `src`.

## 2.9. Importar proyectos

Por otro lado, para importar un proyecto tenemos varias alternativas:

- 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

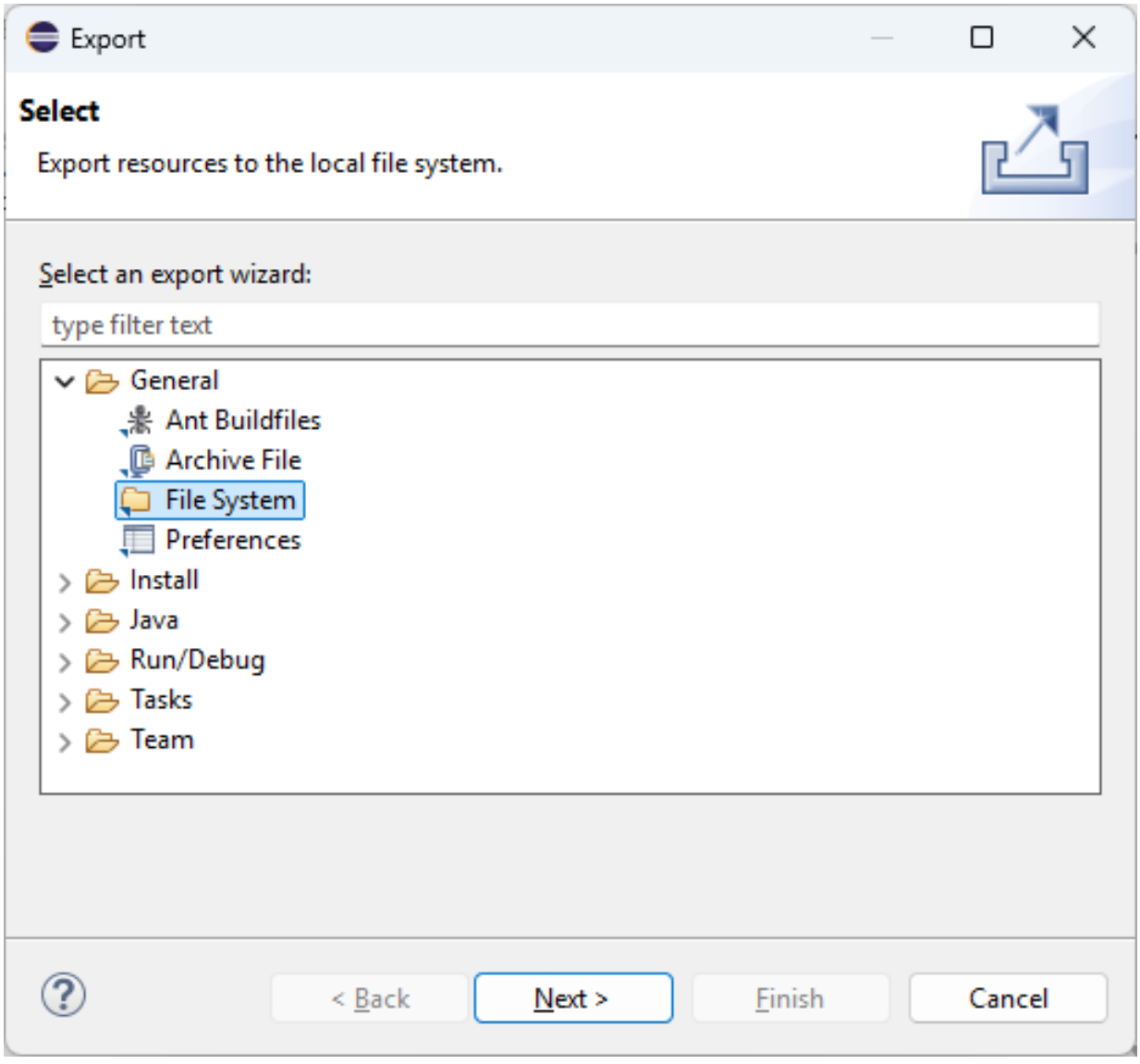



**Figura 11:** Exportación de proyectos a una carpeta de ficheros.

y la importación en el mismo ordenador), deberemos borrar la versión previa para poder importarla correctamente.

- Utilizamos la opción `File > Import` y a continuación seleccionamos `General > Existing projects into workspace` (ver Figura 12). De esta manera, se importará el proyecto completo, incluyendo la configuración de compilación y ejecución. Esta es la opción recomendada para importar proyectos a *Eclipse*.

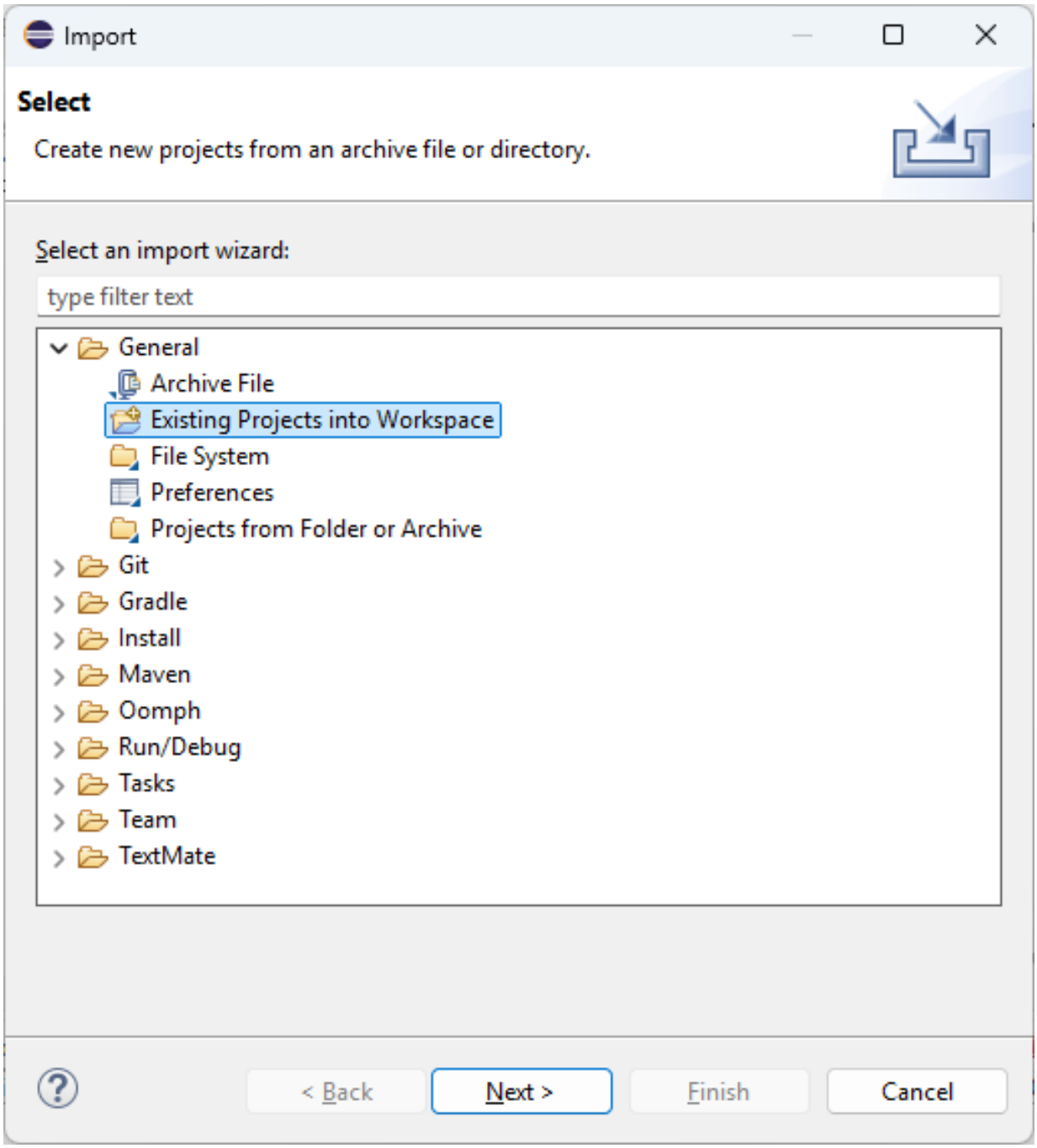



**Figura 12:** Importación de proyectos en un *workspace*.

- 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.

Por ahora recomendamos utilizar la opción **`General > Existing projects into workspace`**.

## 2.10. Crear biblioteca

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

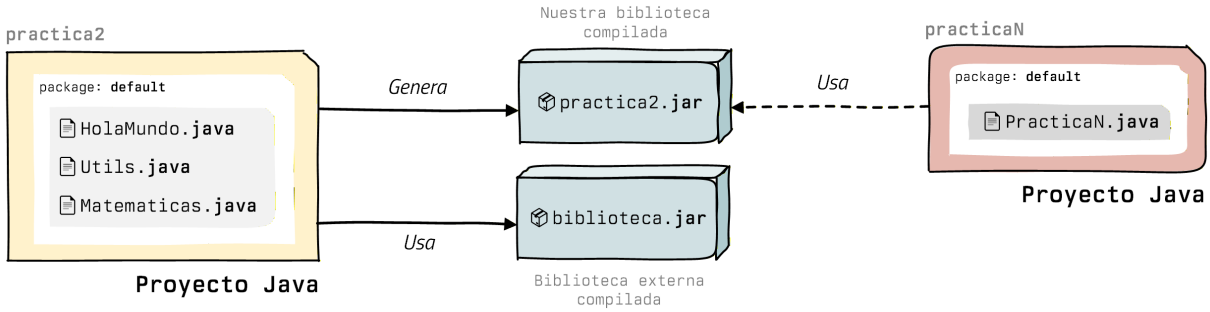



**Figura 13:** Ejemplo de usos de importación y exportación de bibliotecas.

Para crear una biblioteca en el entorno *Eclipse* debemos seleccionar 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 Ilustración 12:

- 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 donde se va a almacenar dicho fichero.
- Pulsa `Finish` para finalizar. Si ha ido todo bien, se habrá generado un fichero `.jar`.

**Ejecución de un programa a partir de un fichero .jar**

Si el programa se llama `HolaMundo.jar`, podrás ejecutarlo escribiendo en una terminal del sistema operativo lo siguiente:

```
java -jar HolaMundo.jar
```

Para abrir una terminal en *Windows*, hay que utilizar la barra de búsqueda e introducir `Símbolo del sistema`. 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 /
```

## 2.11. 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`, hemos añadido en nuestro programa la siguiente instrucción:

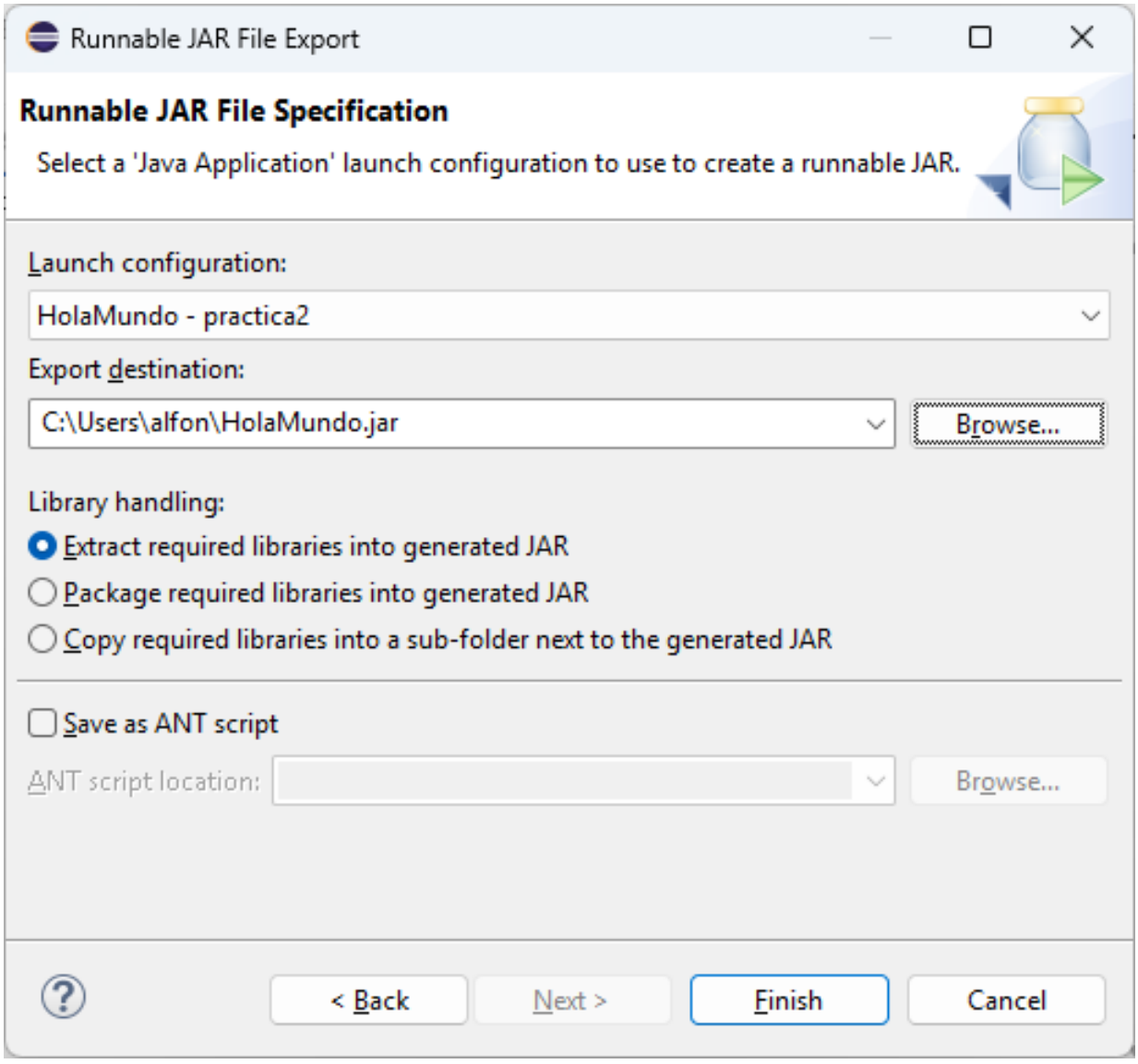



**Figura 14:** Creación de una biblioteca .jar a partir de un proyecto.

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

El proceso de incluir bibliotecas diferentes a las incluidas por defecto en Java puede no ser trivial, dado que hay varias maneras; entre ellas, las siguientes:

- Si tenemos un fichero `.jar` con las clases compiladas, podemos añadirlo directamente.
- 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`.

### Descargar Fichero JAR

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

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

Al compilar, se nos 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 hacer click derecho sobre el nombre del proyecto y seleccionar `Properties > Java Build Path`. Obtendremos una ventana como la de la Figura 15, donde se pueden configurar las bibliotecas que se van a utilizar en el proyecto.

A continuación, hacemos click 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.

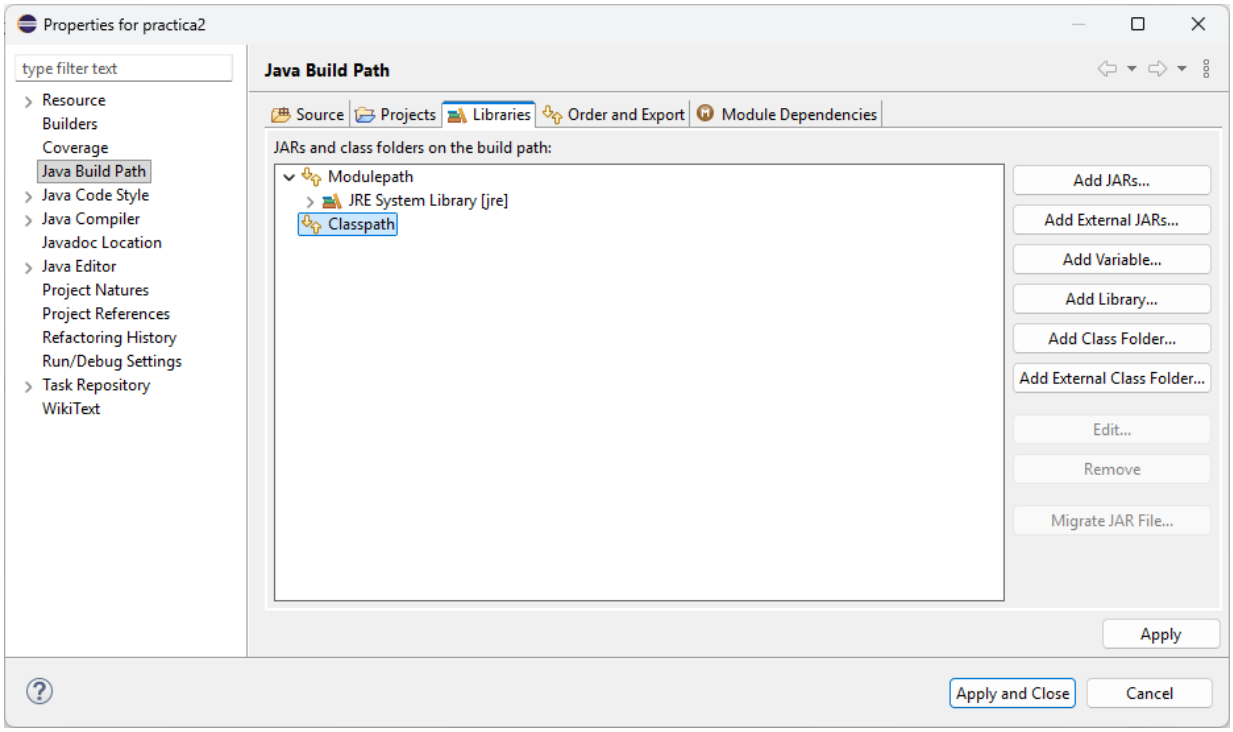



**Figura 15:** Añadir una biblioteca.jar a un proyecto.

## 2.12. 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 16). Al seleccionar esta opción, se muestra una ventana donde se deben configurar algunos parámetros para generar la documentación.

En primer lugar, seleccionamos la ubicación del ejecutable *javadoc* con el botón `Configure`. Normalmente, *Eclipse* detecta automáticamente su ubicación y rellena este campo por nosotros, por lo que no necesitamos 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. Pruébalo para la clase `HolaMundo`.

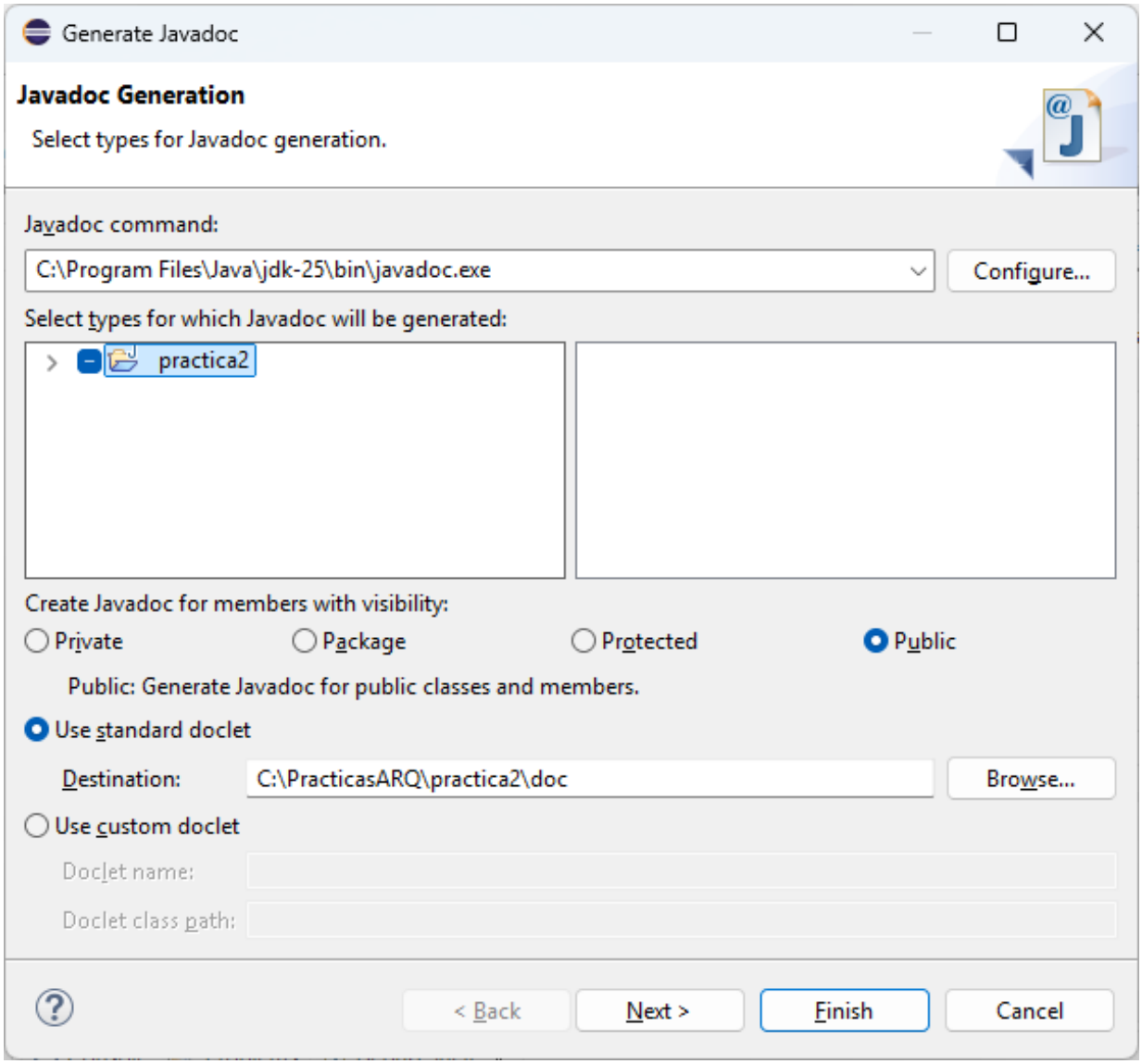



**Figura 16:** Generación de documentación con la herramienta javadoc.

## 2.13. Acceso a la documentación del JDK

Java proporciona una amplia y detallada documentación acerca de las **herramientas que incluye el JDK**, así como la especificación de la API (*Application Programming Interface*), es decir, la documentación de las bibliotecas disponibles por defecto en Java. Esta documentación se encuentra la página oficial de *Oracle* (ver Figura 17), y lo mismo sucede para la documentación de la API (ver Figura 18). 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, modifica la instrucción `x = 1` con métodos de la clase `Math` que permitan obtener un número entero aleatorio: primero se obtendrá un número real aleatorio y luego se redondeará a un número entero. Puede buscarse dicho método en la documentación de la clase `java.lang.Math`.

```
x = (int) Math.???(Math.???());
```

También, modifica la última instrucción del programa para que imprima el valor de `x` con un mensaje descriptivo, por ejemplo:

```
System.out.println("Fibonacci(" + n + ") = " + x);
```

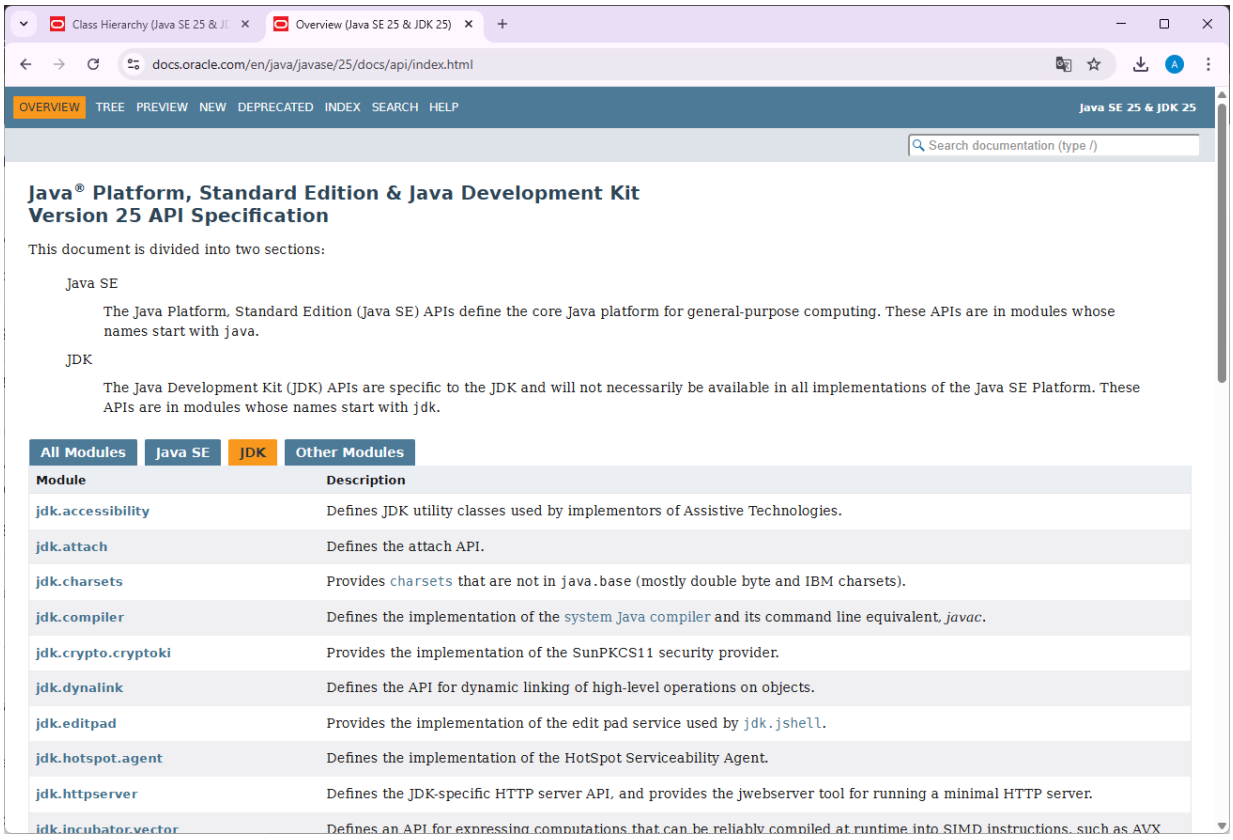



**Figura 17:** Acceso a la documentación del JDK.

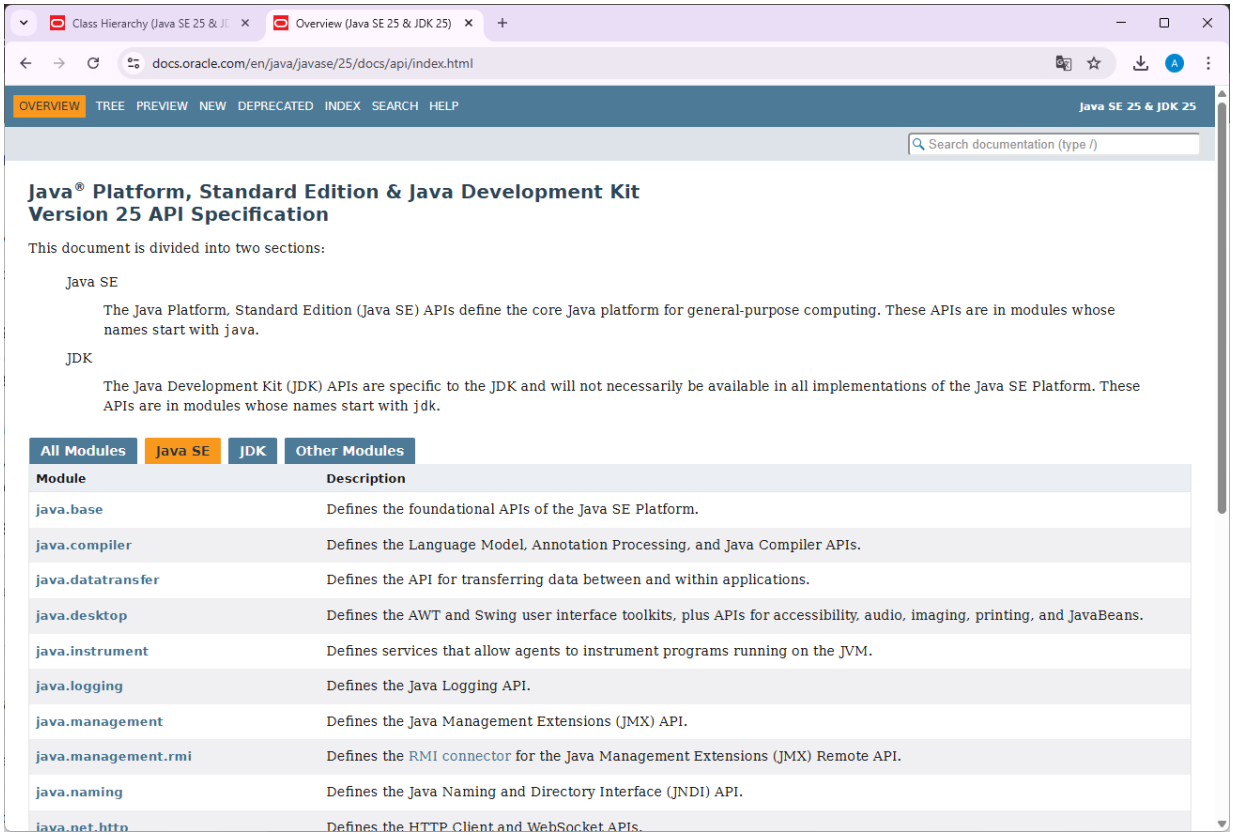



**Figura 18:** Acceso a la documentación de la API de Java.

# 3. Apéndice. Instalación de Eclipse y JDK

En este apéndice explicaremos el proceso de instalación del software necesario para realizar las prácticas de Informática en los ordenadores personales. Existen al menos dos maneras de instalar el software:

## 3.1. Instalación de Eclipse mediante un instalador

- Visita la página web oficial de *Eclipse*.
- Localiza el botón de ***Download***. Nos llevará a la página de descargas.
- Localiza el panel de *Eclipse Installer* (Figura 19). Debemos reconocer nuestro sistema operativo (*Windows*, *Linux* o *macOS*), y pulsar sobre `x86_64`.
- Una vez descargado el instalador, lo ejecutamos.

El instalador permite obtener múltiples herramientas de Eclipse, aunque nosotros sólo necesitamos el entorno de desarrollo para Java. La instalación es bastante automática, y simplemente tendremos que seleccionar Eclipse IDE for Java Developers (Figura 20) y pulsar el botón `Install` para iniciar la instalación.

Una vez seleccionada dicha opción, se nos dirige a una pantalla como la que se muestra en la Figura 21. El objetivo es configurar dos rutas: la localización de las herramientas de desarrollo para Java y la ruta de instalación de Eclipse. Podéis personalizar ambas rutas, pero no es recomendable hacerlo. En particular, la primera ruta apunta a un *Java Development Kit* (JDK), que, si no queremos descargar junto con Eclipse, deberá instalarse por separado. Esta última opción, aunque es menos sencilla, se describe en el siguiente punto y, entre otras ventajas, permite probar los comandos java y javac en la terminal del ordenador.

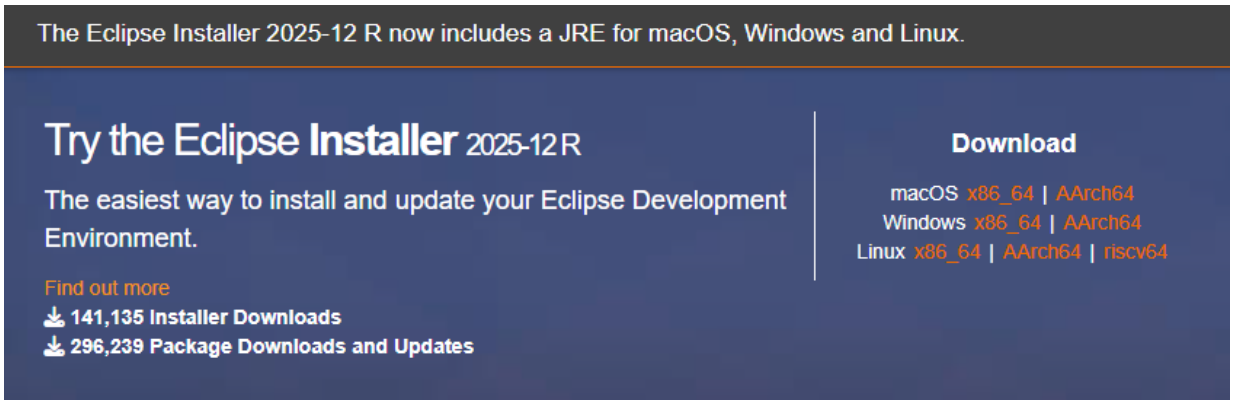



**Figura 19:** Panel de descarga del instalador de Eclipse.

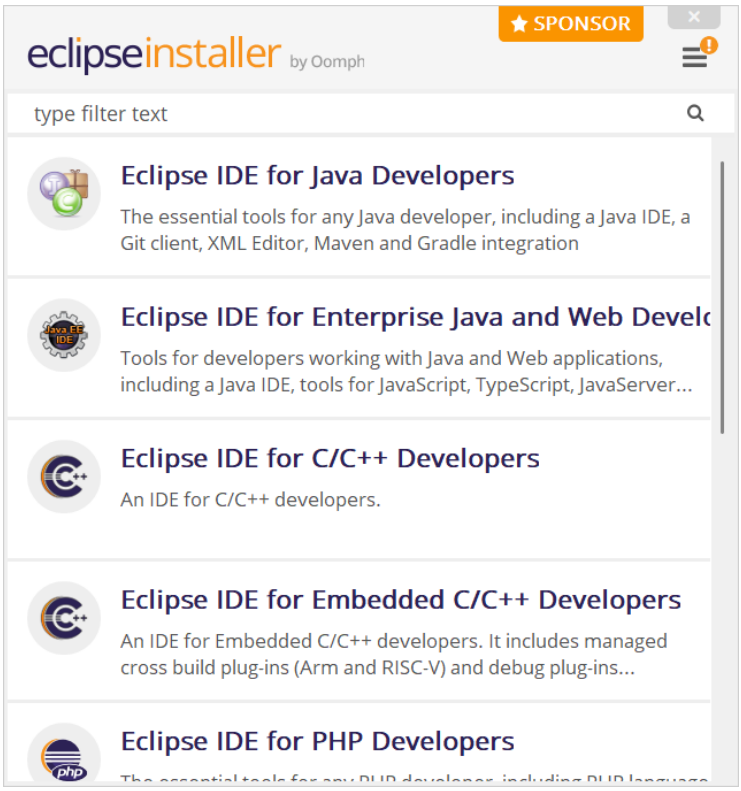



**Figura 20:** Herramientas disponibles en la suite de Eclipse.

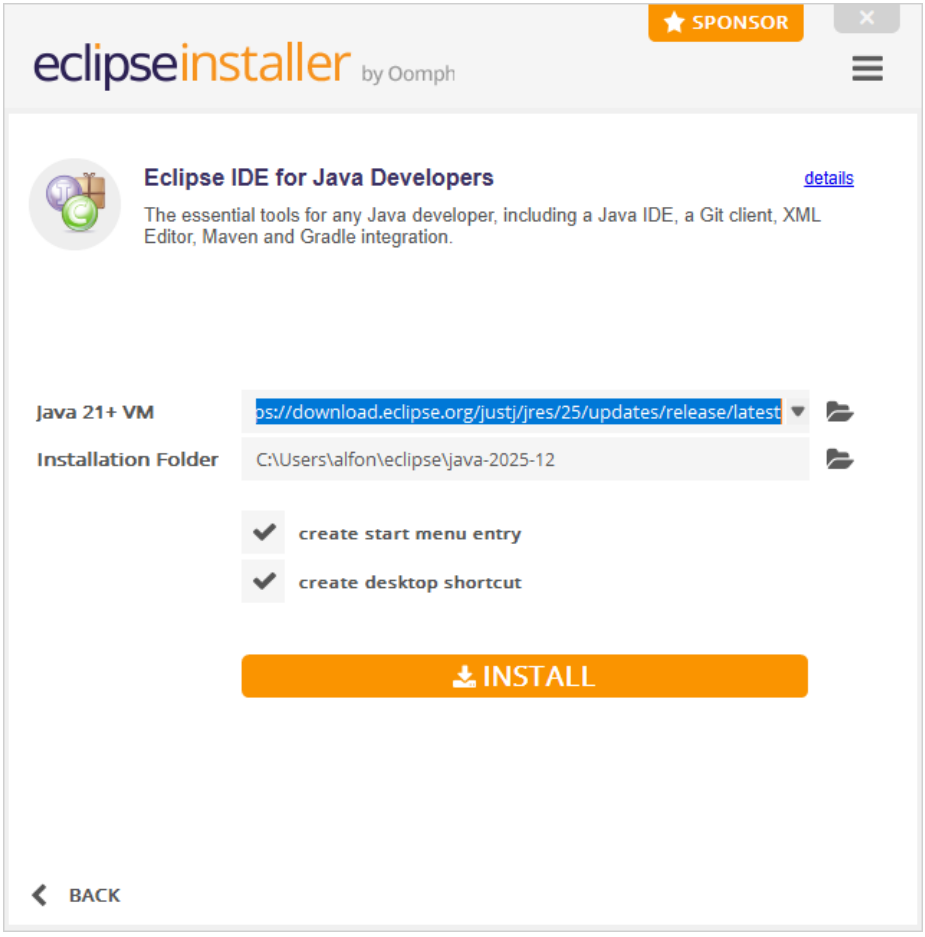



**Figura 21:** Instalación de Eclipse para Java, indicando que queremos descargar un JRE.

Una opción alternativa consiste en indicar la ruta de un *Java Runtime Environment* (JRE) (versión 21 o superior) previamente instalado (ver Figura 22). Para la descarga e instalación del JDK de Java, por favor, dirígete a la sección B.1 del Apéndice. Por ejemplo, en la siguiente imagen, se ha detectado una instalación en la ruta `C:/Program Files/Java/jdk-25`, que es la ruta de instalación por defecto de un JDK.

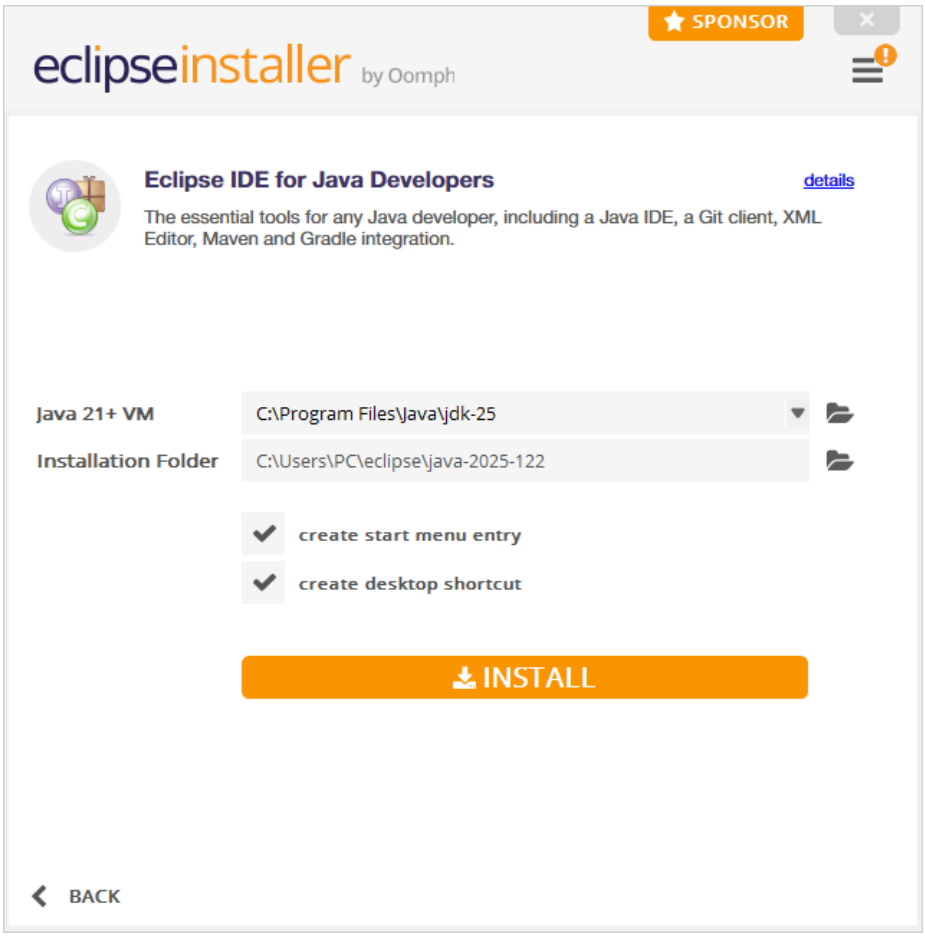



**Figura 22:** Instalación de Eclipse para Java, indicando que queremos utilizar un JDK previamente instalado.

## 3.2. Instalación alternativa de Eclipse sin instalador

### 3.2.1. Instalación del Java Software Development Kit (JDK)

Para comenzar, debemos descargar la última versión del JDK de la página web de Oracle:

- Descargamos el JDK de Java de la página oficial de Oracle. En este momento, la última versión disponible es la **25**.
- Debemos seleccionar el sistema operativo de nuestro ordenador personal (*Linux*, *macOS* o *Windows*) y descargar un instalador. En *Windows*, podemos descargar tanto `x64 Installer` como `x64 MSI Installer`.
- Una vez descargado, abrimos el archivo para comenzar el proceso de instalación. El proceso es completamente automático y no habrá que hacer nada, salvo utilizar el botón `Next` para avanzar al siguiente paso. La ruta de instalación por defecto en Windows es `C:/Program Files/Java/jdk-25`. **Se recomienda no modificar esta ruta, salvo que no tengamos espacio en `C:`.**

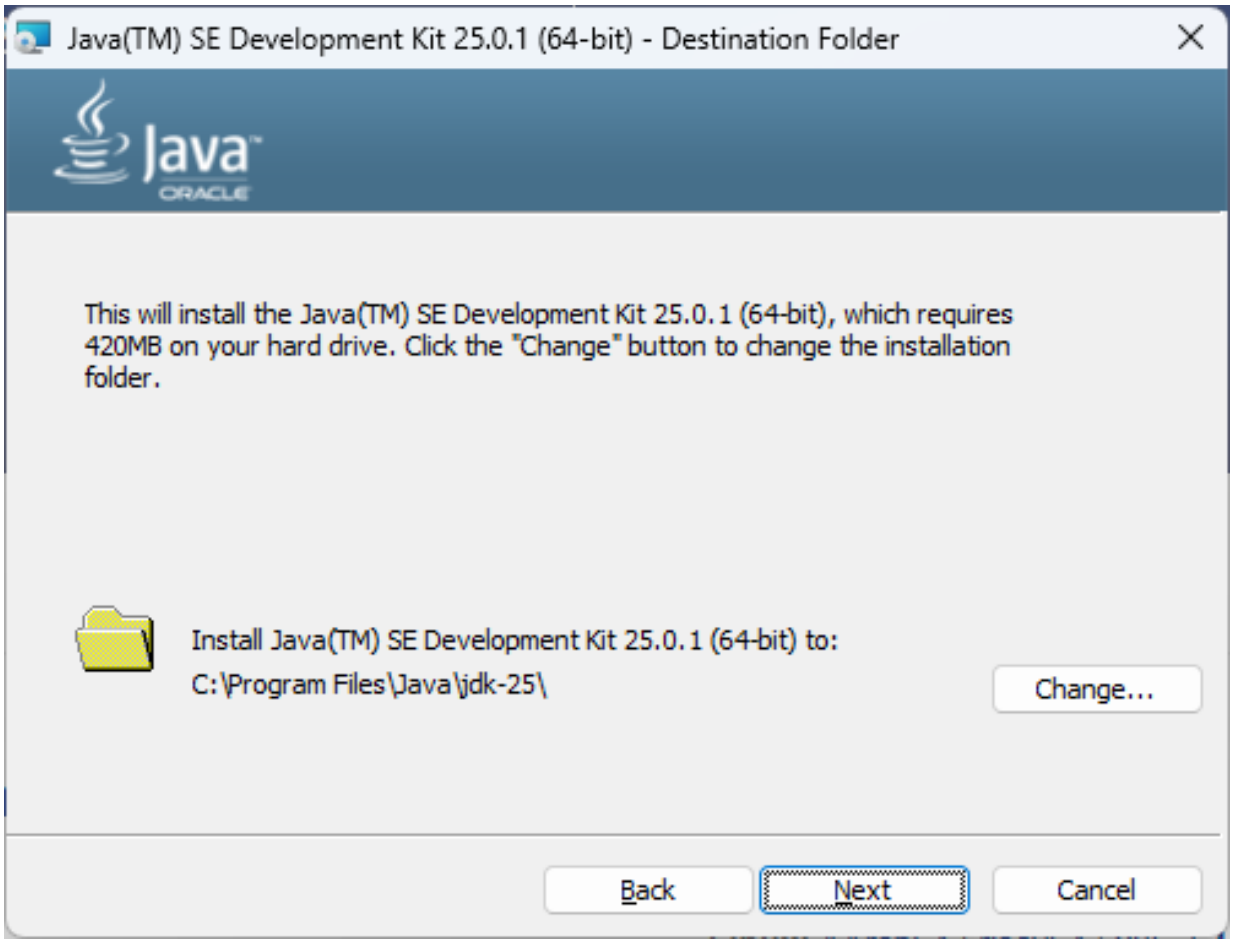



**Figura 23:** Paso intermedio de la instalación del JDK. Se muestra la ruta de instalación por defecto, la cual no modificaremos.

### 3.2.2. Instalación de Eclipse

- Visita la página web oficial de *Eclipse*.
- Localiza el botón de `Download`; nos llevará a la página de descargas.


**Figura 24:** Panel de descarga de Eclipse sin instalador.

- Partiendo de un panel similar al de la Figura 24, reconocemos nuestro sistema operativo y pulsamos sobre `x86_64`. Se descargará un `.zip` con una carpeta dentro llamada `eclipse` (ver Figura 25). Podemos situar dicha carpeta en cualquier directorio de nuestra elección, por ejemplo, en la unidad `C:`. Ten en cuenta que esta opción no requiere instalación, y por tanto, accederemos siempre a dicha carpeta cuando queramos iniciar *Eclipse* haciendo click sobre `eclipse.exe`. En la Figura 25 se muestra el contenido del fichero `.zip` descomprimido.

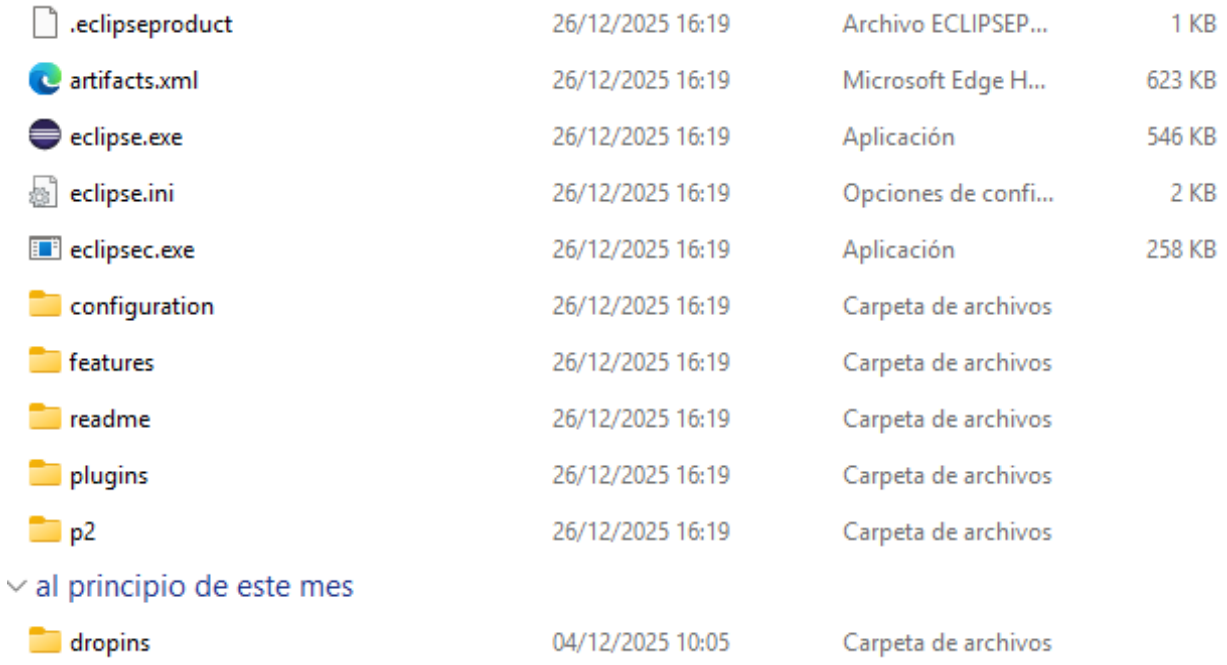



**Figura 25:** Ficheros contenidos en el directorio eclipse, tras descomprimir el fichero .zip descargado de la página oficial de Eclipse.

## 3.3. Pantalla de bienvenida

La primera vez que se utiliza *Eclipse*, aparecerá un mensaje preguntándonos si queremos excluir *Eclipse* del análisis del antivirus de Windows. Como aparece en la Figura 26, podemos indicar que se excluya *Eclipse* de dichos análisis para evitar ralentizar la aplicación.

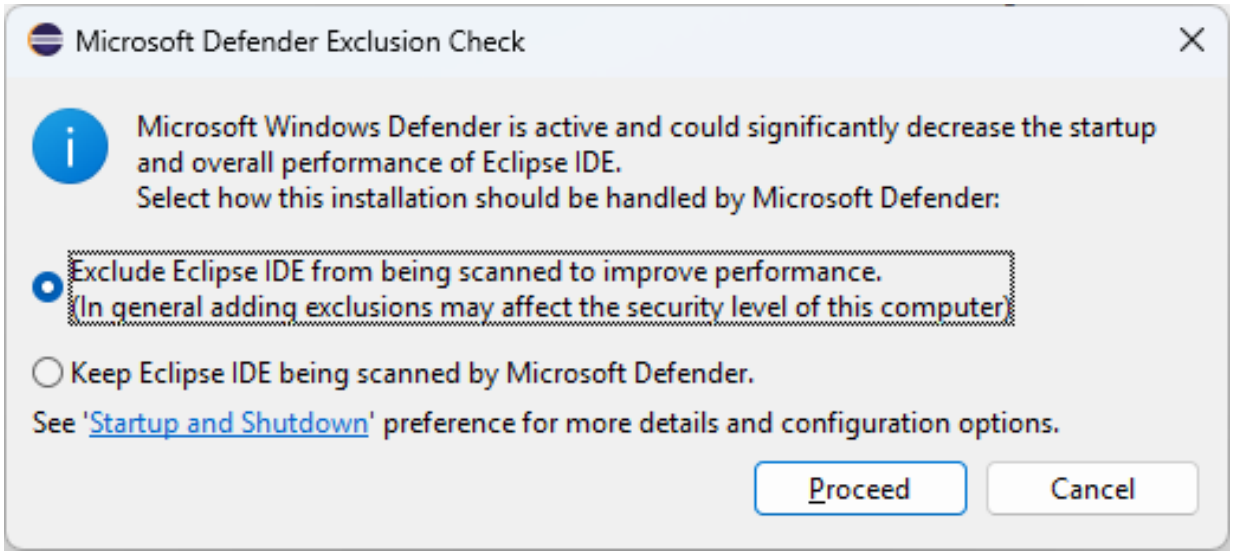



**Figura 26:** Pregunta de exclusión de Eclipse IDE respecto del sistema antivirus de Windows.

## 4. Entrega de la solución

**Sólo debe realizar la entrega un miembro del grupo**. Sube a Moodle el fichero `Ejercicio.java` con el código de la clase `Ejercicio`. No es necesario subir el proyecto completo ni ningún otro fichero.