# Práctica 2:

# Un entorno de programación en Java

## Objetivo 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.

## 1. El entorno de desarrollo Eclipse

Para facilitar la tarea de desarrollo de programas existen entornos que permiten trabajar de una manera más cómoda. Algunos ejemplos son NetBeans, Eclipse o IntelliJ. Nosotros recomendamos el uso de Eclipse, un entorno de desarrollo de libre distribución (en el Apéndice 3 se incluye documentación sobre la descarga e instalación del entorno).

Inicia la herramienta Eclipse haciendo doble clic sobre el ejecutable Eclipse en el escritorio de los laboratorios de prácticas, o bien buscando Eclipse en la barra de tareas. Para instalar esta aplicación en otros ordenadores puede consultarse el Apéndice. Instalación de Eclipse y JDK. Al iniciar *Eclipse* aparecerá la pantalla de bienvenida que se muestra en la Ilustración 1.

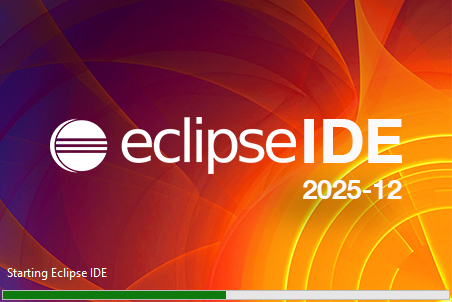


Ilustración 1. Pantalla de presentación.

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

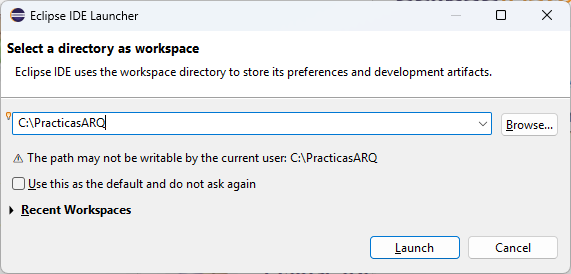


Ilustración 2. Configuración del workspace.

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 (<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 un mensaje preguntándonos si queremos excluir Eclipse del análisis del antivirus de Windows. Como aparece en la Ilustración 3, podemos indicar que se excluya Eclipse de dichos análisis para evitar ralentizar la aplicación.

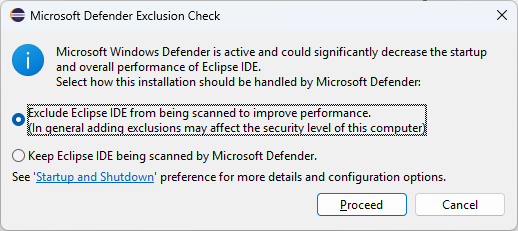


Ilustración 3. Pregunta de exclusión de Eclipse IDE respecto del sistema antivirus de Windows.

Una vez seleccionada una opción, nos aparecerá una ventana de bienvenida (Ilustración 4) 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****.* Otra 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. |



Ilustración 4. Pantalla de bienvenida.

Si cerramos la pestaña de bienvenida, nos conducirá al entorno de trabajo que se explica en la siguiente sección.

En esta pantalla de bienvenida, en la barra izquierda, se nos muestran algunas opciones, entre las que se incluye *Create a Java project*. Utilizaremos esta opción cada vez que queramos crear un nuevo proyecto, por ejemplo, para cada práctica de programación de este curso. Para esta práctica, podemos crear un nuevo proyecto al que llamaremos **practica2**.

### 1.2. 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 5 se puede observar la interfaz de usuario de *Eclipse*.

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

1. En la zona de la izquierda, se halla un menú (*Package Explorer*) que agrupa los componentes de cada uno de los proyectos abiertos.
2. En la zona central de la pantalla se encuentra un editor, que sirve para ver y modificar los ficheros Java con los que trabajamos.
3. 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.).
4. En la parte inferior del editor se encuentran algunas utilidades que nos permiten conocer el estado del proyecto, incluyendo errores (*Problems*) y la salida obtenida en los programas ejecutados (*Console*).

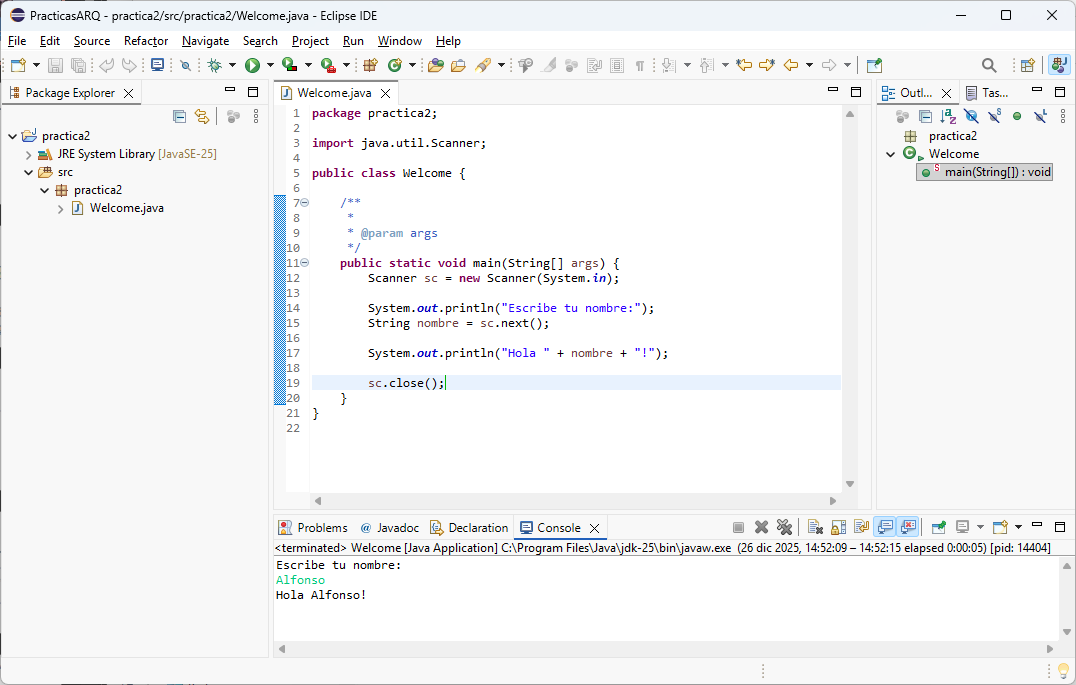


Ilustración 5. Espacio de Trabajo (workbench).

### 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 programas individuales 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 dotarlo de un nombre identificativo (ej. *practica2*). Dicho proyecto corresponderá con un directorio dentro del equipo, y dentro del *workspace* seleccionado al lanzar *Eclipse*. Conviene seleccionar la opción *Create separate source and output folders*. En la Figura 6 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:\practicasARQ\practica2

Observa que en la carpeta practica2 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*).



Ilustración 6. Propiedades que podemos configurar al crear un nuevo proyecto.

### 1.4. 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 Ilustración 7 ilustra este proceso.

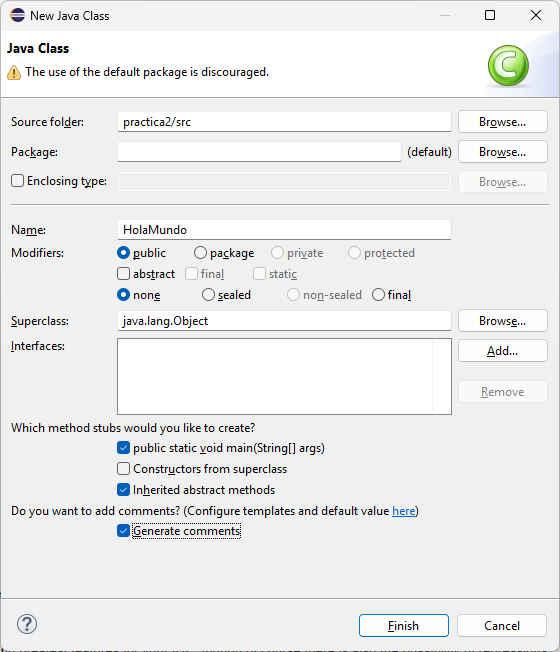


Ilustración 7. Creación de una nueva clase de Java.

Fijaos en que, además de la carpeta, hemos indicado el nombre de un paquete. Dicho nombre no se puede dejar en blanco, por lo que hemos nombrado *practica2*. En futuras sesiones de prácticas hablaremos de qué es un paquete, y cómo crearlo para estructurar las clases que creemos en cada proyecto.

Al finalizar la creación, se creará 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.");

}

}

El código anterior contiene algunos errores introducidos voluntariamente 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 o mayúsculas-tabulador.

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*.

### 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 guardar el fichero para que se compile.

Dentro de las herramientas que se muestran en la porción inferior, encontramos la subpestaña *Problems*. En esa subpestaña se nos informará sobre el proceso de compilación indicando los posibles errores (ver Ilustración 8). Puedes seleccionar cada error mediante doble click, y se indicará dentro del código dónde se encuentra 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*

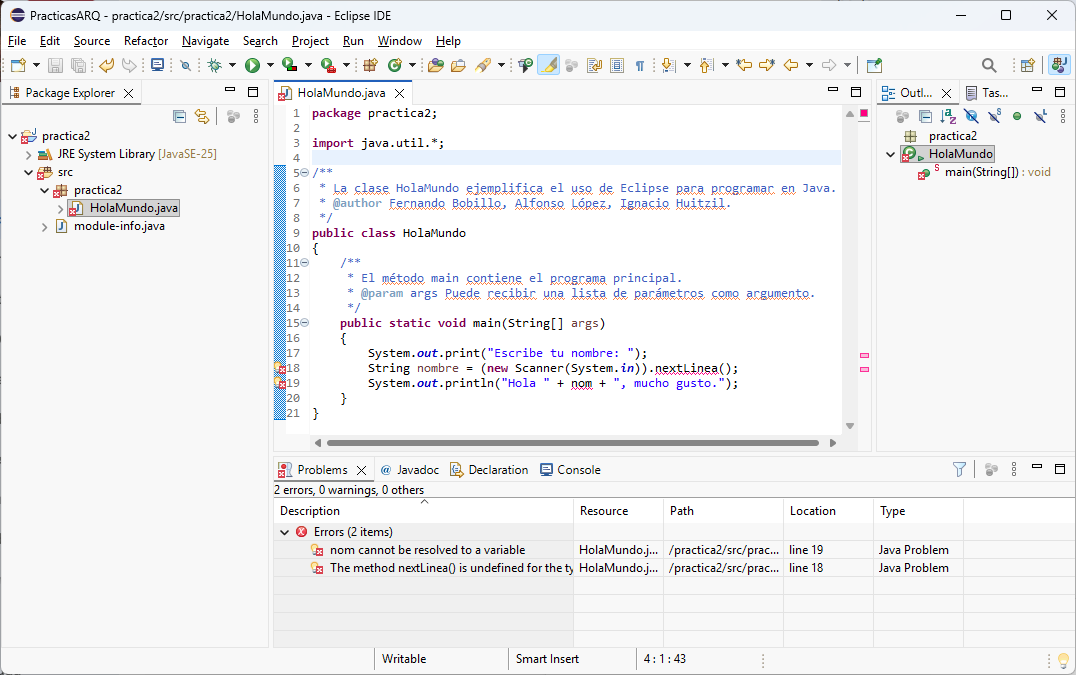


Ilustración 8. Compilación con errores.

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*, o mediante el icono de ejecutar en la barra de herramientas superior .

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, lo ejecutaremos paso a paso para comprender mejor el funcionamiento real del programa, tal y como se ilustra en la Ilustración 9.

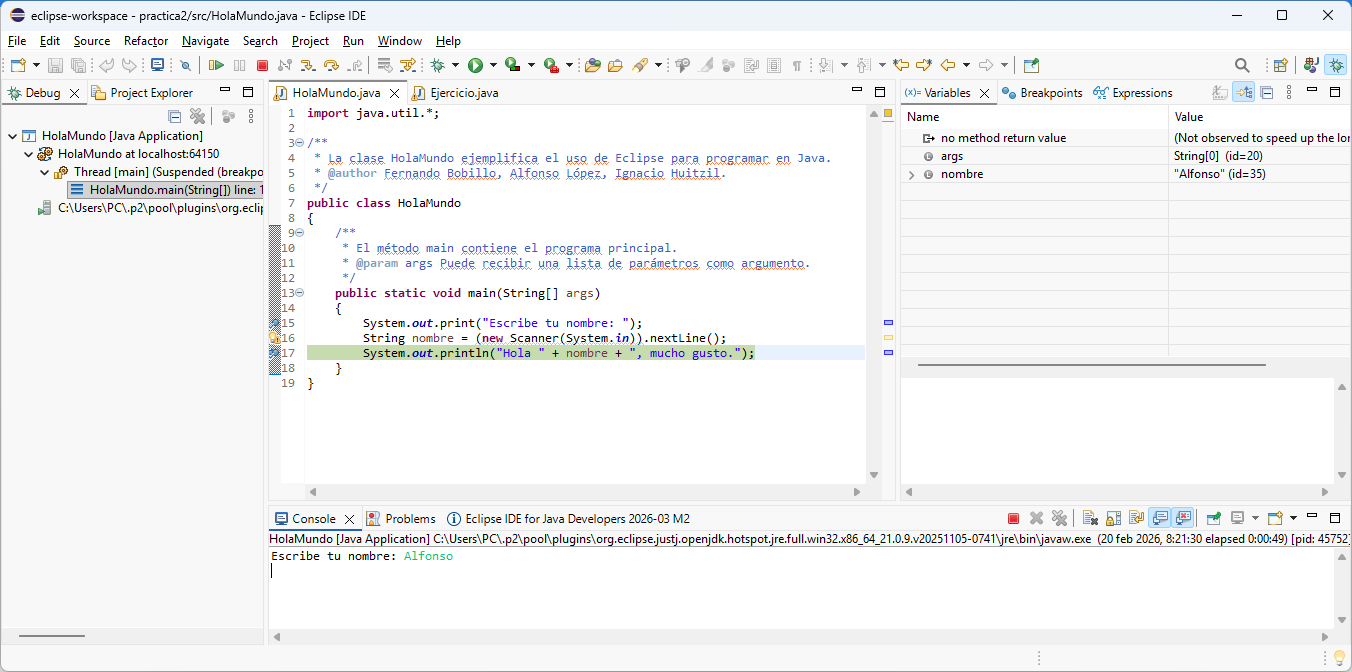


Ilustración 9. Depuración de un programa.

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. 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*, 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:

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);

}

}

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

* Utiliza la opción de menú *File > Export* y elegir *General > File system* (ver Ilustración 10).

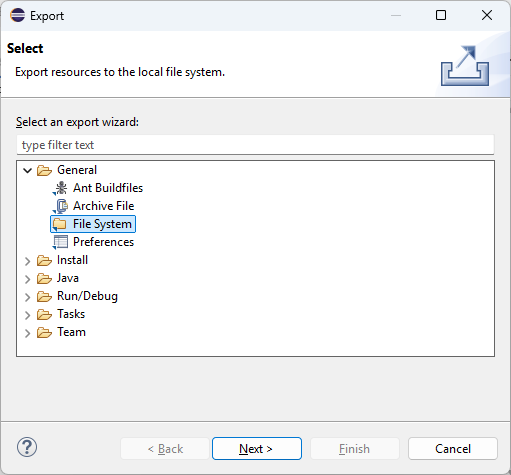


Ilustración 10. Exportar proyectos del workspace.

* 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*.

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 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 Ilustración 11).

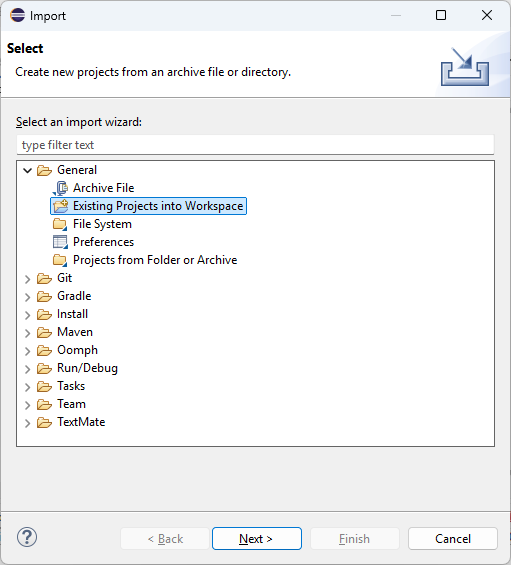


Ilustración 11. Importar proyectos del workspace.

* 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.*

### 1.9. Crear y utilizar bibliotecas .jar

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 (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 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*.

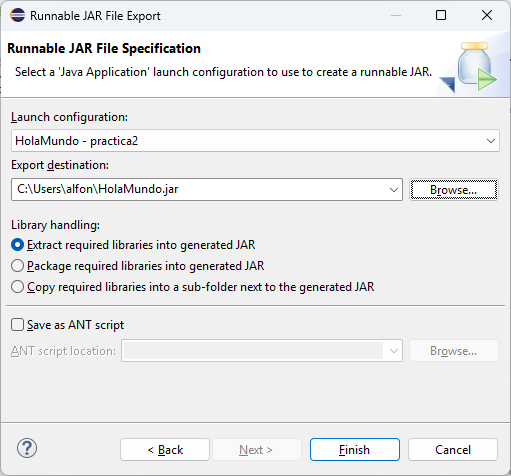


Ilustración 12. Construcción de una biblioteca .jar.

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 /

### 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, hemos añadido en nuestro programa 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 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*.

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 Ilustración 13.

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.

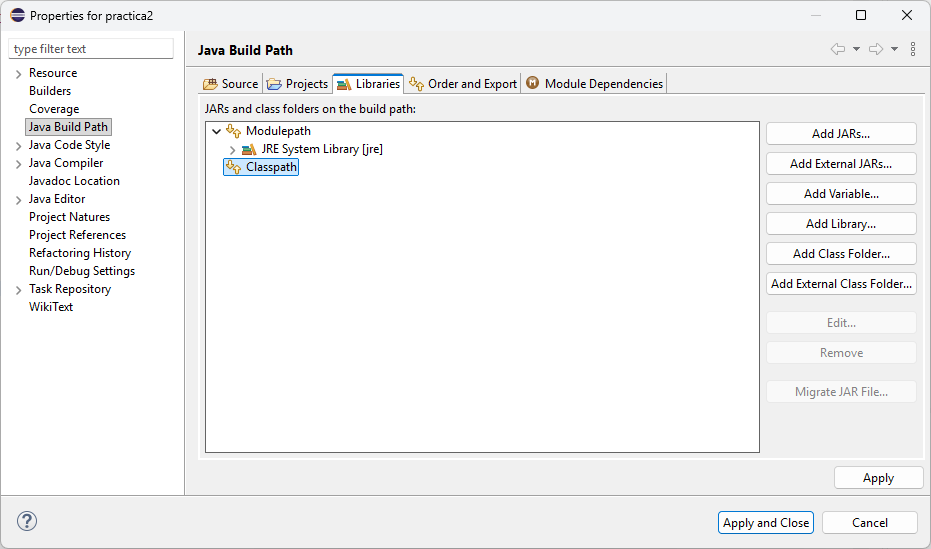


Ilustración 13. Añadiendo una biblioteca .jar a un proyecto.

### 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* (Ilustración 14).

* 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*.

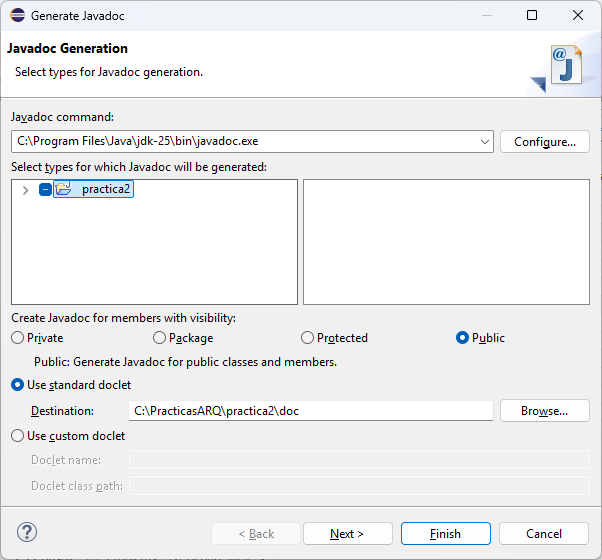


Ilustración 14. Generación de documentación con la herramienta javadoc.

## 1.12. 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 está accesible en la [página oficial de Oracle](https://docs.oracle.com/en/java/javase/25/docs/api/index.html) (Ilustración 15), y lo mismo sucede para la documentación de la API (Ilustración 16). 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 enteros 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*](https://docs.oracle.com/en/java/javase/25/docs/api/java.base/java/lang/Math.html).

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

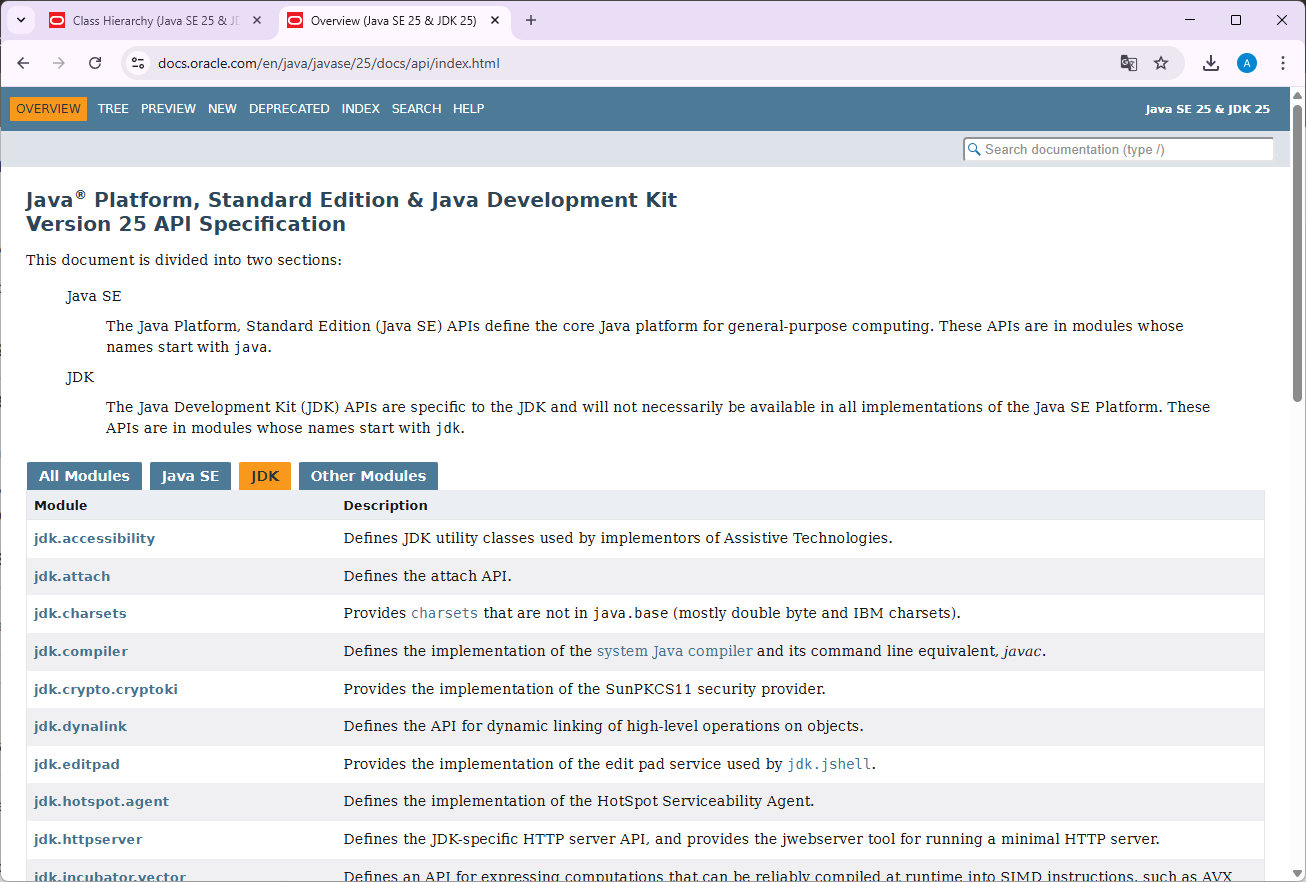


Ilustración 15. Documentación del JDK.

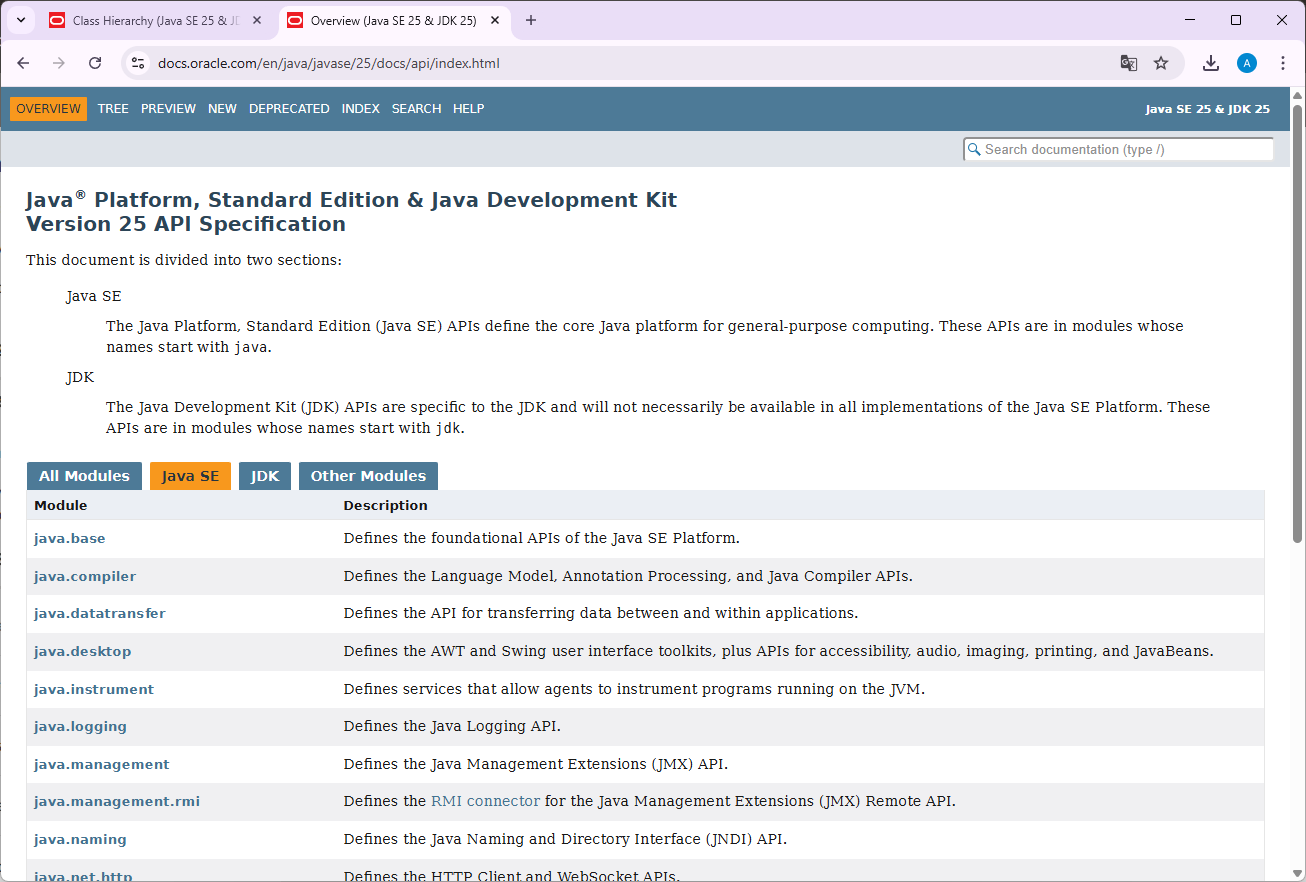


Ilustración 16. Documentación de la API de Java.

## 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:

### Instalación de Eclipse mediante un instalador

* Visita la [página web oficial](https://eclipseide.org/) de Eclipse.
* Localiza el botón de *Download*. Nos llevará a la [página de descargas](https://www.eclipse.org/downloads/packages/).
* Localiza el panel de *Eclipse Installer* (Ilustración 17). 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* (Ilustración 18).

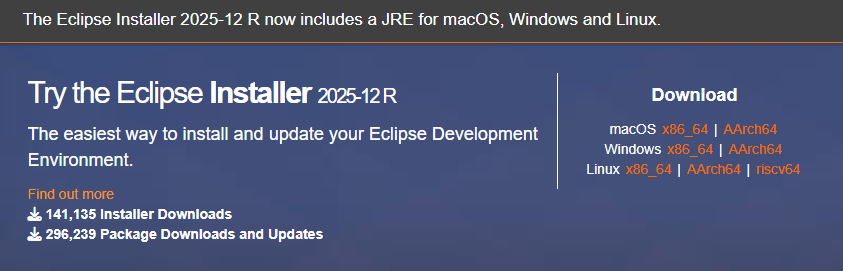


Ilustración 17. Instalador de la suite de Eclipse.

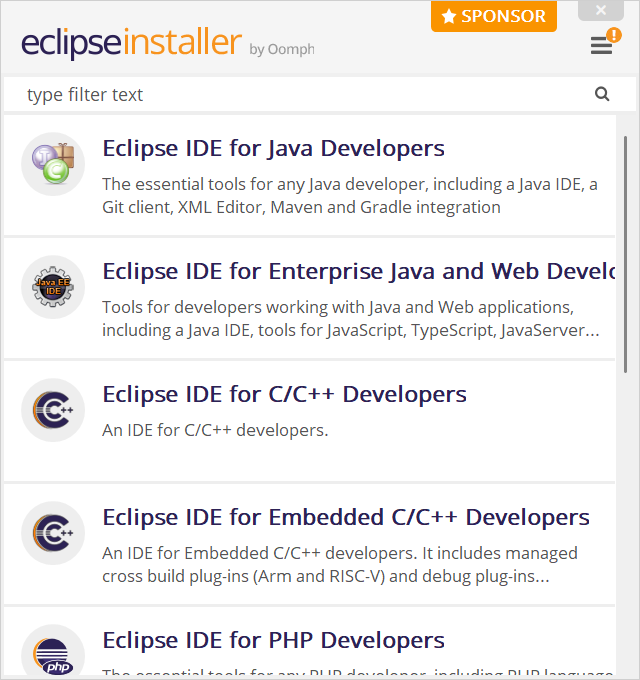


Ilustración 18. Herramientas disponibles en la suite de Eclipse.

* Una vez seleccionada dicha opción, se nos dirige a una pantalla como la que se muestra en la Ilustración 19. 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.

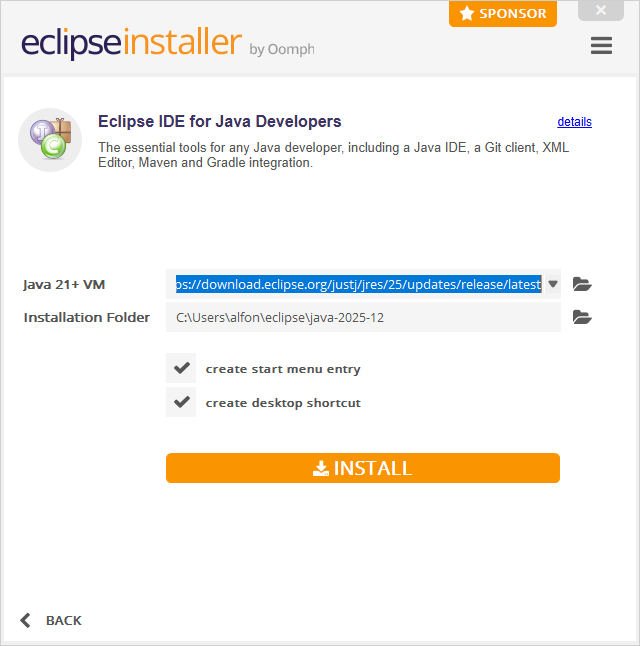


Ilustración 19. Instalación de Eclipse para Java, indicando que queremos descargar un JRE.

* Una opción alternativa consiste en indicar la ruta de un *Java Runtime Environment* (JRE)(versión 21 o superior) previamente instalado (ver Ilustración 20). 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.

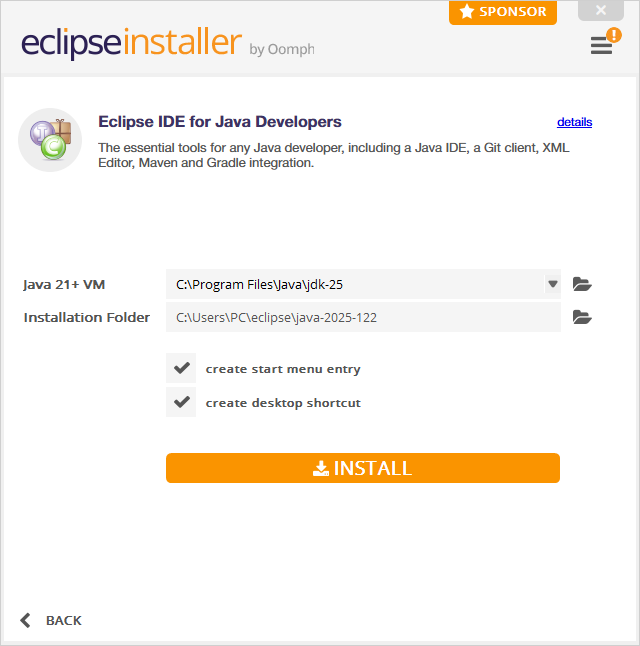


Ilustración 20. Instalación de Eclipse para Java, indicando la ruta de un JDK previamente instalado.

### Instalación alternativa de Eclipse sin instalador

#### 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](https://www.oracle.com/java/technologies/downloads/#jdk25-windows). 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 entre pasos. 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:*.

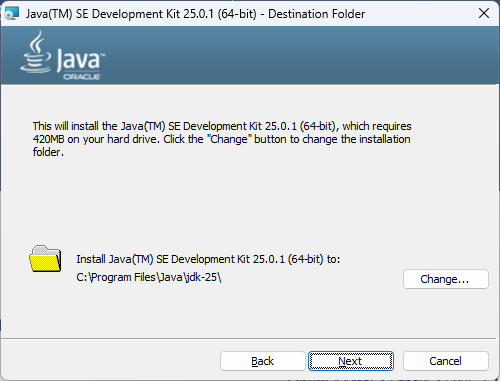


Ilustración 21. Paso intermedio de la instalación del JDK. Se muestra la ruta de instalación por defecto, la cual no modificaremos.

#### Instalación de Eclipse

* Visita la [página web oficial](https://eclipseide.org/) de Eclipse.
* Localiza el botón de *Download*. Nos llevará a la [página de descargas](https://www.eclipse.org/downloads/packages/).
* Partiendo de un panel similar al de la Ilustración 22, reconocemos nuestro sistema operativo y pulsamos sobre *x86\_64*. Se descargará un *.zip* con una carpeta dentro llamada *eclipse* (ver Ilustración 23). 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 Ilustración 23 se muestra el contenido del fichero .zip descomprimido.



Ilustración 22. Panel de descarga de Eclipse sin instalador.



Ilustración 23. Ficheros contenidos en el directorio eclipse, tras descomprimir el fichero .zip descargado de la página oficial de Eclipse.