



Instalación y uso de entornos de desarrollo

Objetivos

- Comprender la utilidad de los entornos de desarrollo integrados.
- Identificar los componentes de un entorno de desarrollo integrado.
- Instalar entornos de desarrollo integrados para crear aplicaciones en Java.
- Saber utilizar entornos de desarrollo integrados para la edición de programas y la generación de archivos ejecutables.
- Instalar y desinstalar módulos (*plug-ins*) en un entorno de desarrollo integrado.
- Conocer los mecanismos de actualización de un entorno de desarrollo integrado.

Contenidos

- 2.1. La utilidad de los entornos de desarrollo integrados
- 2.2. Componentes de un entorno de desarrollo
- 2.3. Instalación de un entorno de desarrollo
- 2.4. Edición de programas y generación de archivos ejecutables
- 2.5. Instalación y desinstalación de módulos
- 2.6. Mecanismos de actualización

Introducción

En la Unidad 1 se explicó el proceso que es necesario llevar a cabo para transformar el código fuente de un programa en código ejecutable. Ello conlleva el empleo de varias herramientas: un editor, un compilador o intérprete, un depurador, etc. Hoy en día, en lugar de emplear estas herramientas de forma aislada, se suelen integrar en una aplicación que recibe el nombre de **entorno de desarrollo** o **entorno de desarrollo integrado** (*integrated development environment*), conocido también por sus siglas en inglés, IDE.

En este tema, se verán cuáles son los componentes de un IDE, se explicará la instalación de dos entornos de desarrollo (Eclipse y Apache NetBeans) y cómo se pueden crear programas mediante ellos. Además, se exemplificará la instalación de módulos adicionales para realizar tareas que los citados entornos no llevan incorporadas, pero que pueden resultar interesantes con el objeto de crear aplicaciones de un determinado tipo o llevar a cabo tareas que quedan fuera del ámbito estricto de la programación.

2.1. La utilidad de los entornos de desarrollo integrados

En la Unidad 1, se explicó el proceso de creación de programas ejecutables, que consta de los siguientes pasos:

1. Creación del código fuente del programa siguiendo las normas sintácticas de un determinado lenguaje de programación de alto nivel.
2. Transformación del código fuente en código objeto mediante el empleo de un traductor (compilador o intérprete). Si el traductor detecta errores de sintaxis, indica el error o errores correspondientes y no se genera el código objeto.
3. Obtención del código ejecutable insertando en el código objeto una serie de rutinas o librerías.

Para la realización de cada uno de estos pasos, se debe emplear una herramienta diferente. Así, para la primera tarea se debe emplear un editor; para la segunda, un compilador o un intérprete; y para la tercera, un enlazador.

El hecho de tener que emplear una herramienta independiente para cada una de estas tareas, cuando estas siempre se tienen que realizar en secuencia para la creación de un programa, es tedioso. Por este motivo, se crearon los llamados **entornos de desarrollo integrados**. Estos se pueden definir como una aplicación informática compuesta por un conjunto de herramientas integradas que facilitan el desarrollo de software. Estas herramientas también se conocen como **entornos de desarrollo**.

Argot técnico

La forma más habitual de referirse a los entornos de desarrollo o entornos de desarrollo integrados es empleando sus siglas en inglés IDE (*Integrated Development Environment*).



Aunque por la definición propuesta, se puede entender que un IDE asiste a la persona usuaria en todas las tareas de desarrollo de software (análisis, diseño, programación, pruebas y mantenimiento), lo cierto es que las actividades que mejor apoyo reciben por la mayoría de los entornos de desarrollo son la programación y las pruebas, por lo que en algunos casos se suele hablar de **entornos de programación**. La mayor parte de los IDE, además de permitir la edición de programas y su ejecución, tras la traducción correspondiente, permiten llevar a cabo las siguientes tareas:

- Ejecutar en modo depuración, para poder corregir los errores encontrados durante las pruebas.
- Analizar la consistencia y calidad del código fuente.
- Ejecución automática de pruebas.
- Control de versiones.
- Generación de documentación.
- Optimización del código (refactorización).

Cabe mencionar aquí que hay IDE creados para su uso con un único lenguaje de programación, pero también existen entornos de desarrollo multilenguaje, es decir, que permiten crear y ejecutar programas escritos en varios lenguajes de programación de alto nivel.

A continuación, se muestra la manera de crear un programa muy sencillo en Java y su ejecución sin el empleo de ningún entorno de desarrollo, en otras palabras, de la manera tradicional.

Para crear y ejecutar programas en Java, independientemente de que se use o no un IDE, es necesario instalar el equipo de desarrollo de Java, conocido de manera abreviada como **JDK (Java Development Kit)**, con el que viene incorporado; asimismo, el entorno en tiempo de ejecución de Java, más conocido como **JRE (Java Runtime Environment)**:

- El JDK es un software que proporciona herramientas para la creación de programas en Java, entre ellas un compilador llamado *javac*.
- El JRE está formado por un conjunto de utilidades que permiten la ejecución de programas escritos en Java. Este entorno está formado por la máquina virtual de Java (JVM), que se trató en la Unidad 1, un conjunto de bibliotecas Java y otros componentes necesarios para que una aplicación escrita en Java pueda ejecutarse.

El JDK y el JRE de Java se instalan conjuntamente. Para ello, se accede a la página web: <https://www.oracle.com/Java/technologies/Javase-downloads.HTML#JavaseJDK>. En esta dirección es donde se accede a la última versión disponible en el momento de escribir este libro, la 16.0.2. En esta página, en la sección correspondiente a la versión Java SE Development Kit 16.0.2, se hace clic en la descarga correspondiente a nuestro sistema operativo. En el caso de que se desee realizar la instalación en Windows, es posible descargar un archivo comprimido, un instalador MSI (por sus siglas en inglés, *Microsoft installer*, conocido ahora como *Windows installer*) o un instalador .exe. En este ejemplo, se elige esta última opción.

Una vez realizada la descarga, se comienza la instalación haciendo doble clic sobre el fichero ejecutable descargado. La instalación no presenta ningún problema y se seleccionan todas las opciones por defecto.

Cuando estén instalados el JDK y el JRE de Java, será posible crear programas en Java sin el empleo de ningún IDE o mediante el empleo de uno como Eclipse.

Para la creación de un programa en Java sin el empleo de ningún IDE, lo primero que se ha de hacer es escribir su código fuente mediante el empleo de un editor de textos, como el bloc de notas. En este, se escribe el siguiente código, por medio del cual se crea un programa que lo único que hace es escribir en la pantalla el texto «¡Hola, mundo!»:

```
1 public class HolaMundo{  
2     public static void main (String[] args){  
3         System.out.println ("¡Hola, mundo!");  
4     }  
5 }
```

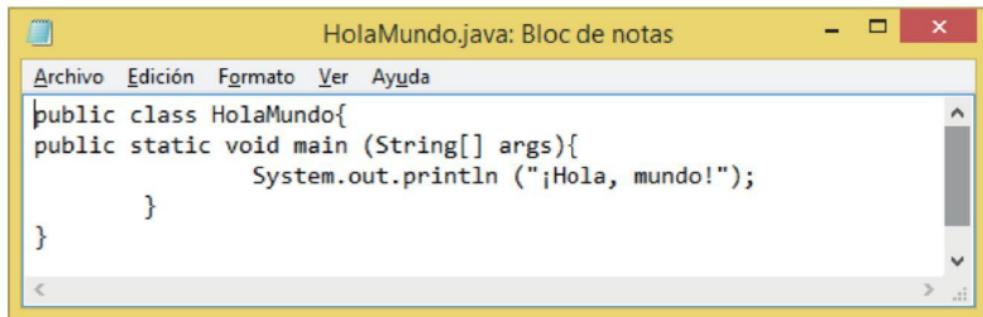


Figura 2.1. Código fuente de un programa que muestra un mensaje en la pantalla y que ha sido creado en el bloc de notas.

Este programa se debe guardar con el nombre de la clase creada y extensión .java, por lo que su nombre será: *HolaMundo.java*. Además, se guarda en la carpeta en la que está instalado el JDK de Java, que es C:\Program Files\Java\jdk-16.0.2\bin. Según la configuración de nuestro sistema y del usuario con el que se haya accedido, puede ser que no sea posible almacenar el archivo en la carpeta indicada. En este caso, se debe ejecutar el editor de textos como administrador.

El siguiente paso es la compilación del programa, para lo que se usa el compilador de Java *javac* ubicado en la carpeta correspondiente al JDK. Con este fin, se abre el **símbolo del sistema** y nos colocaremos en la carpeta del JDK. Una vez en esa carpeta, se escribe *javac* y, a continuación, el nombre del fichero fuente que se desea compilar, como se muestra en la Figura 2.2. Puede ser necesario, también en este caso, ejecutar el símbolo del sistema como administrador, si surge algún problema con los permisos. En la Figura 2.2, se pueden leer las instrucciones ejecutadas desde el símbolo del sistema.



Argot técnico

El **símbolo del sistema**, también conocido por sus siglas en inglés **CMD** (*command prompt*), es un intérprete de línea de comandos que incorpora Windows desde el que se pueden dar órdenes al sistema por escrito empleando el teclado en lugar de utilizando una interfaz gráfica. En una interfaz gráfica de usuario, también conocida como **GUI** (por sus siglas en inglés, *graphical user interface*), se emplean tanto el teclado como el ratón para realizar operaciones sobre elementos gráficos disponibles en la pantalla del ordenador. Así, las interfaces más características de los sistemas operativos actuales son interfaces gráficas.

```
Administrator: Símbolo del sistema
Microsoft Windows [Versión 6.3.9600]
(c) 2013 Microsoft Corporation. Todos los derechos reservados.

C:\Windows\system32>cd C:\Program Files\Java\jdk-16.0.2\bin
C:\Program Files\Java\jdk-16.0.2\bin>javac HolaMundo.java
C:\Program Files\Java\jdk-16.0.2\bin>
```

Figura 2.2. Órdenes que hay que ejecutar en el símbolo del sistema para compilar el archivo con el código fuente.

Si todo ha funcionado correctamente, no se mostrará ningún mensaje y se habrá creado un archivo llamado *HolaMundo.class* en la carpeta donde está el archivo *HolaMundo.java*. En caso de que haya habido algún problema, aparece el correspondiente mensaje de error que se debe corregir. Una vez corregido, se vuelve a ejecutar la orden de compilación del programa.

Tras compilar correctamente el programa, solo faltará ejecutarlo, para lo que, desde el símbolo del sistema, se escribe *java* y, después, el nombre del fichero sin extensión, en este caso, *HolaMundo*. Como se puede observar en la imagen de la Figura 2.3, el mensaje «¡Hola, mundo!» aparece en la pantalla.

```
Símbolo del sistema
C:\Program Files\Java\jdk-16.0.2\bin>java HolaMundo
¡Hola, mundo!
C:\Program Files\Java\jdk-16.0.2\bin>
```

Figura 2.3. Orden para ejecutar el programa una vez que este ha sido compilado exitosamente. Debajo de dicha orden se muestra el resultado de la ejecución.

En el Apartado 2.4 se verá que, con la ayuda de un IDE, estas tareas se pueden llevar a cabo más rápida y eficientemente mediante herramientas diseñadas a tal efecto.

2.2. Componentes de un entorno de desarrollo

En este apartado, se ofrece una lista de los componentes que vienen incorporados en la mayoría de los entornos de desarrollo, siendo los dos primeros los que forman parte de todos ellos:

- **Editor de textos:** se emplea para la escritura del código fuente del programa. Incluye las funciones propias de cualquier editor de textos (copiar, cortar, pegar, buscar, reemplazar, etc.) e incluye herramientas importantes para la escritura de los programas. Así, marca en ciertos colores determinados elementos del lenguaje, como palabras reservadas y variables, resalta el inicio y el fin de cada bloque, etc. Esto contribuye a la legibilidad de los programas y es un apoyo relevante para las tareas de escritura y revisión del código fuente.
- **Compilador o intérprete:** comprueba la corrección del programa escrito en lenguaje fuente y, en caso de que sintácticamente sea correcto, genera el correspondiente código objeto o directamente el código ejecutable. La diferencia entre los compiladores y los intérpretes, como se indicó en la Unidad 1, es que los compiladores traducen todo el programa fuente y almacenan el resultado, mientras que los intérpretes van traduciendo porciones de código y las van ejecutando, y no almacenan el resultado de la traducción.
- **Depurador (*debugger*):** ayuda a quienes realizan la programación en la tarea de depuración, esto es, en la localización y corrección de errores en el programa. Para ello, permite realizar tareas como la detención de la ejecución del programa en una determinada instrucción mediante el establecimiento de puntos de ruptura, la ejecución del programa instrucción a instrucción, la visualización de los valores de las variables a lo largo de la ejecución, etc. El depurador puede resultar muy útil para detectar y corregir determinados errores que, sin esta herramienta, resultaría complicado.
- **Constructor de interfaz gráfica:** permite crear aplicaciones con una interfaz gráfica de usuario (GUI, por sus siglas en inglés, *graphical user interface*). Una GUI se crea mediante la colocación de distintos controles en la pantalla, como botones, campos de texto, listas o barras de menús, con los cuales interactúa el usuario mediante el ratón o el teclado. La aplicación responde a las acciones (eventos) que realiza el usuario sobre estos controles. La mayoría de los entornos de desarrollo requieren la instalación de módulos adicionales para la creación de este tipo de aplicaciones.
- **Control de versiones:** controla los cambios que se realizan sobre los programas, creando diferentes versiones de estos a medida que se van incorporando dichos cambios.

Recuerda

Todo entorno de desarrollo debe incluir obligatoriamente un editor de textos orientado al lenguaje y un compilador o un intérprete. La mayoría de los entornos de desarrollo incorporan también los otros componentes descritos en este apartado.



Existen, además, como herramientas independientes (no como entornos de desarrollo) editores de textos, que permiten escribir código fuente. Con la mayor parte de estos, es posible escribir en varios lenguajes de programación. Algunos ejemplos de editores de este tipo son Notepad++ y Sublime. La mayoría de estos editores de textos presentan las mismas características que aquellos que van incorporados a los entornos de desarrollo (resaltado de determinados elementos de los programas, señalamiento del comienzo y final de los bloques, etc.), pero no permiten transformar el código fuente escrito en código objeto ni ejecutable.

■ 2.3. Instalación de un entorno de desarrollo

En la actualidad, existen muchos entornos de desarrollo disponibles. En este libro, se explican dos entornos de desarrollo para poder crear programas en Java. En este sentido, el hecho de que Oracle ponga a disposición de quienes desarrollan programas, de forma gratuita, el JDK de Java y otras descargas ha permitido que surjan un buen número de aplicaciones de apoyo, entre ellas, entornos de desarrollo como Eclipse, NetBeans, JBuilder, JDeveloper, etc.

Para explicar cómo se instala un IDE, aquí se han elegido dos de esos entornos, que son de código abierto y gratuitos, concretamente, Eclipse y NetBeans. El software de código abierto es software cuyo código fuente es del dominio público y, por tanto, puede ser utilizado, cambiado e incluso redistribuido. Sin embargo, el software de código cerrado se caracteriza por que el código fuente del producto no está a disposición del público en general.

■ ■ 2.3.1. Instalación de Eclipse

Eclipse presenta una arquitectura abierta y extensible basada en módulos (*plug-ins* en inglés). De esta forma, Eclipse puede ser muy ligero o más pesado, dependiendo de los módulos que se instalen. Otros entornos llevan integrados una serie de herramientas necesarias para desarrollar webs o para crear aplicaciones con interfaces gráficas de usuario, motivo por el cual estos entornos desperdician recursos del ordenador, ya que hacen al IDE más lento durante el tiempo en el que está funcionando. Eclipse se puede instalar en diferentes sistemas operativos, en versiones de 32 y 64 bits.

En primer lugar, aquí se explicará cómo se instala la plataforma Eclipse. Las versiones de Eclipse disponibles se encuentran en la página web <http://www.eclipse.org/downloads/> (Figura 2.4).

Si se hace clic en el botón *Download x86_64*, aparece otra pantalla, en la que, para comenzar la descarga, hay que pulsar sobre el botón *Download* (Figura 2.5).

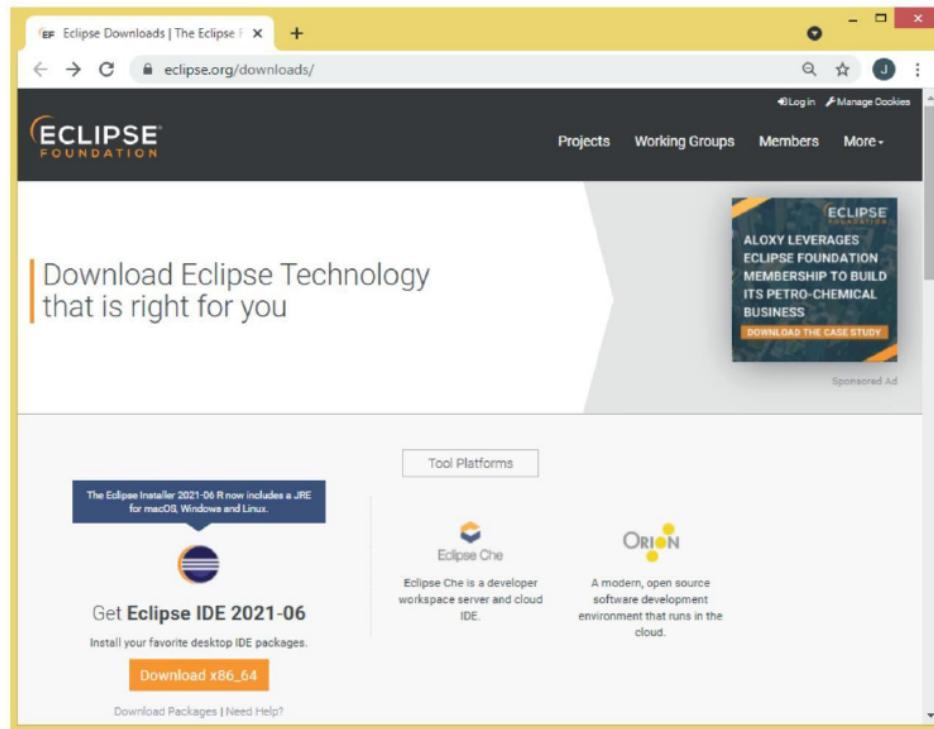


Figura 2.4. Se pulsa en el botón Download x86_64 para descargar Eclipse para sistemas operativos de 32 o de 64 bits.

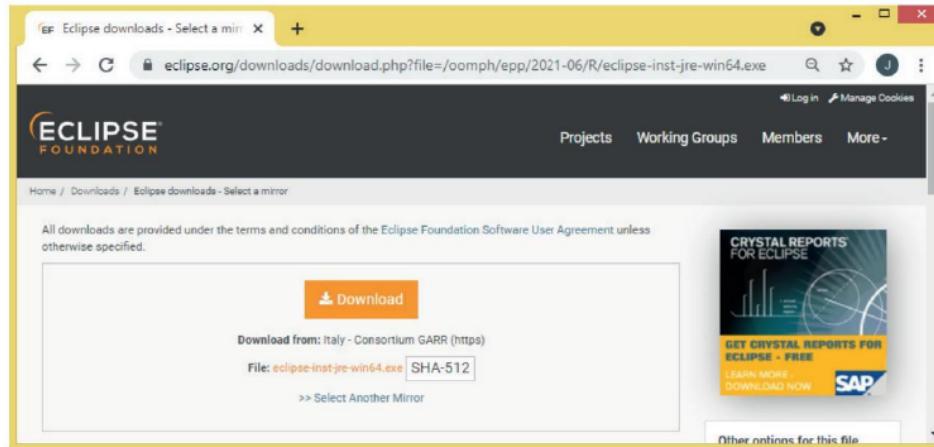


Figura 2.5. Para iniciar la descarga del IDE Eclipse, se hace clic en el botón Download.

Tras hacer doble clic en el archivo ejecutable descargado, se selecciona la primera opción: *Eclipse IDE for Java Developers*. En la ventana emergente, se puede modificar la carpeta en la que se encuentra la máquina virtual de Java (JVM), así como la carpeta en la que se desea realizar la instalación. Lo habitual es que sean válidas las rutas propuestas. Al hacer clic en el botón *INSTALL*, comienza la instalación, al inicio de la cual, se solicita aceptar el acuerdo de licencia. Una vez finalizada la instalación, aparece una ventana como la de la Figura 2.6, en la que se indican las carpetas donde se encuentran instalados JDK y Eclipse, y en la que se puede pulsar en el botón *LAUNCH* para iniciar el IDE por primera vez.

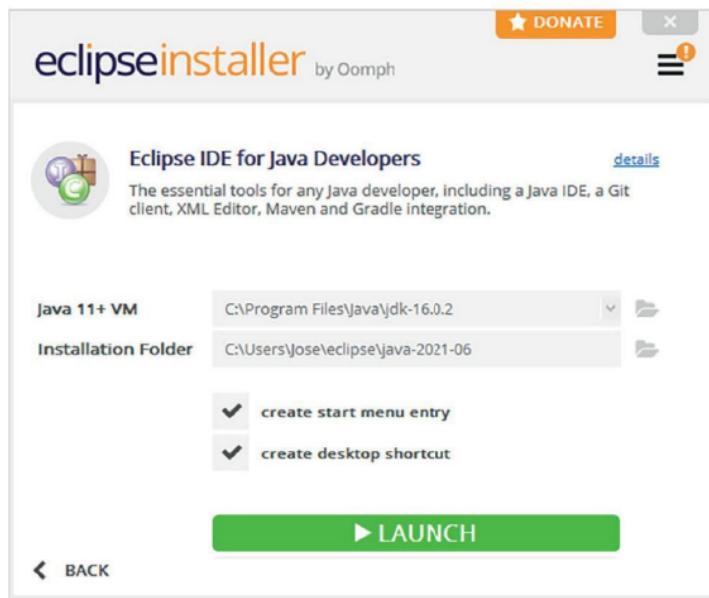


Figura 2.6. Ventana que se muestra tras la instalación exitosa del IDE Eclipse. Para iniciar Eclipse por primera vez, se pincha sobre el botón Launch.

2.3.2. Instalación de NetBeans

Para instalar la última versión de Apache NetBeans, que en el momento de escribir este libro, es la 12.4, se accede a la página web <https://NetBeans.Apache.org/download/index.html>, y allí, se hace clic sobre el botón *Download* (Figura 2.7).

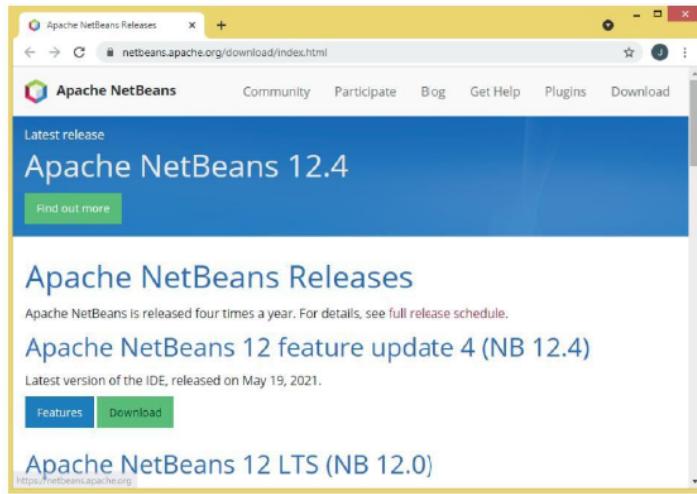


Figura 2.7. La última versión de Apache NetBeans se puede descargar en la página web de descargas de Apache NetBeans, pinchando sobre el botón Download.

Se puede descargar un instalador, disponible para diferentes sistemas operativos, o bien un archivo comprimido. Aquí, se elige esta última opción, pulsando en el enlace correspondiente al archivo NetBeans-12.4-bin.zip (Figura 2.8).

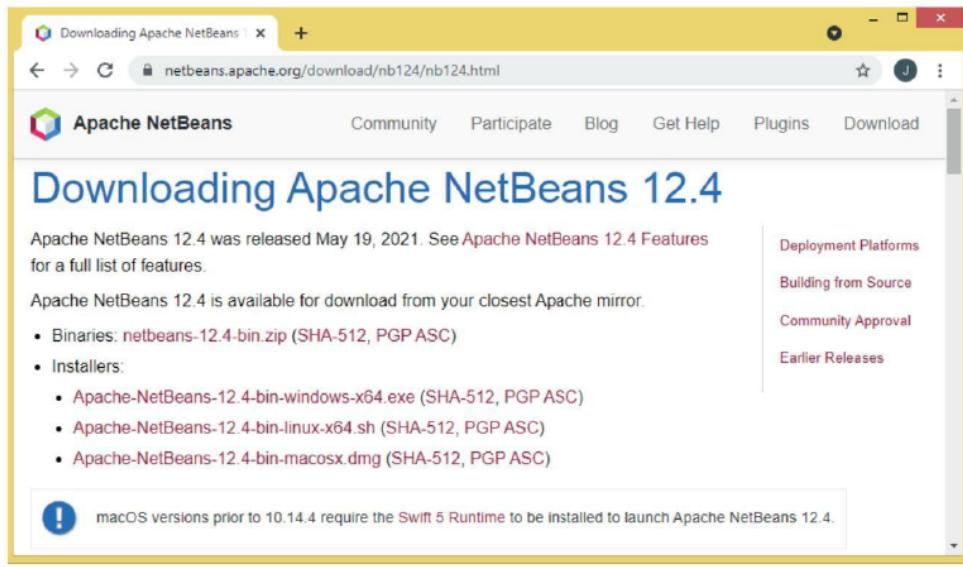


Figura 2.8. Página web en la que se puede descargar un instalador de Apache NetBeans, dependiendo del sistema operativo, o bien un archivo binario (archivo comprimido).

Una vez realizada la descarga del archivo comprimido, hay que descomprimirlo en nuestro equipo. Por ejemplo, se puede descomprimir en la unidad C:\, donde, tras la descompresión, se crea una carpeta con el nombre *NetBeans*. En la carpeta *NetBeans\bin* habrá dos archivos ejecutables para iniciar el IDE NetBeans: *NetBeans.exe* y *NetBeans64.exe*. Es aconsejable crear un acceso directo a uno de estos archivos, dependiendo de si el sistema operativo utilizado es de 32 o 64 bits. Al pinchar sobre este acceso directo, se inicia el IDE NetBeans.

■ ■ ■ 2.4. Edición de programas y generación de archivos ejecutables

Una de las funciones que debe incorporar necesariamente todo entorno de desarrollo es la de permitir la escritura del código fuente de los programas y la generación de archivos ejecutables. En este apartado, se explica cómo se llevan a cabo ambas tareas en Eclipse y NetBeans.

■ ■ ■ 2.4.1. Edición de programas y generación de archivos ejecutables en Eclipse

Al acceder al IDE Eclipse, se solicita que se indique el nombre y ubicación del espacio de trabajo (*workspace*) en el que desea almacenar los proyectos (Figura 2.9). Si la ruta que se propone no nos interesa, se puede modificar escribiéndola directamente o pulsando en el botón *Browse* para navegar por la estructura de carpetas de nuestro equipo. Después, se pincha sobre el botón *Launch*.

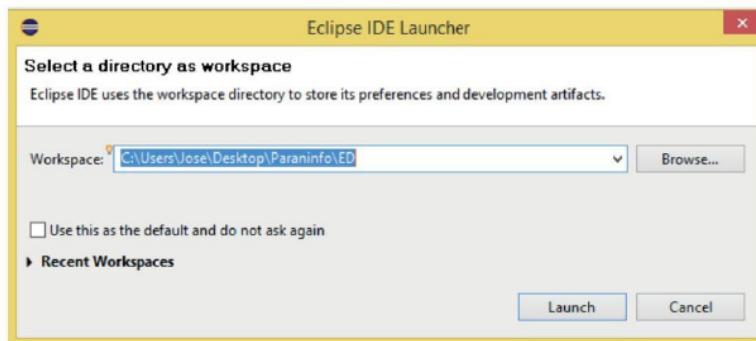


Figura 2.9. Ventana que se muestra la primera vez que se ejecuta Eclipse y las siguientes veces si no se activa la casilla de verificación que se muestra. Se debe elegir la carpeta en la que se desean guardar los proyectos, que se conoce como espacio de trabajo o workspace.

Tras elegir la ubicación del espacio de trabajo, para empezar a trabajar con Eclipse, hay que crear un proyecto seleccionando la opción de menú *File* → *New* → *Java Project*. Seguidamente, se asigna un nombre al proyecto. Por ejemplo, como se muestra en la Figura 2.10, se escribe el nombre *Ejercicios* y se hace clic en el botón *Finish*.

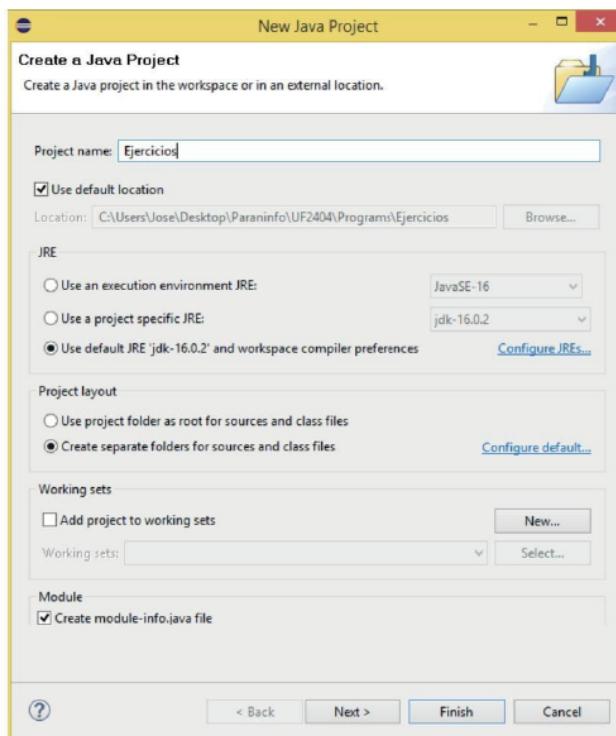


Figura 2.10. Ventana para la creación de un proyecto, dentro del cual se pueden crear varios paquetes o clases.

Una vez creado el proyecto, existen varias opciones. Una de ellas consiste en crear paquetes dentro de nuestro proyecto para englobar diversas clases. Para crear un paquete, se selecciona la opción de menú *File* → *New Package*. Por su parte, si se desea crear clases en el proyecto en el paquete por defecto o en el paquete que se haya creado, se selecciona la opción de menú *File* → *New Class*.

Así pues, siguiendo con el ejemplo, primero se selecciona la opción de menú *File* → *New Package* para crear un paquete llamado *programas*, como se muestra en la Figura 2.11.



Figura 2.11. Ventana para la creación de un paquete dentro de un proyecto. En la opción Source folder, se indica la carpeta en la que se creará y, en la casilla de abajo, se le da un nombre al paquete.

A continuación, para crear una clase, se activa la opción de menú *File* → *New Class*. En la pantalla que aparece después, se indica el paquete en el que se desea ubicar la clase y su nombre. Asimismo, es posible seleccionar diferentes opciones interesantes, como por ejemplo, que se cree un método *main* para la clase que se está creando o que se generen comentarios (Figura 2.12).

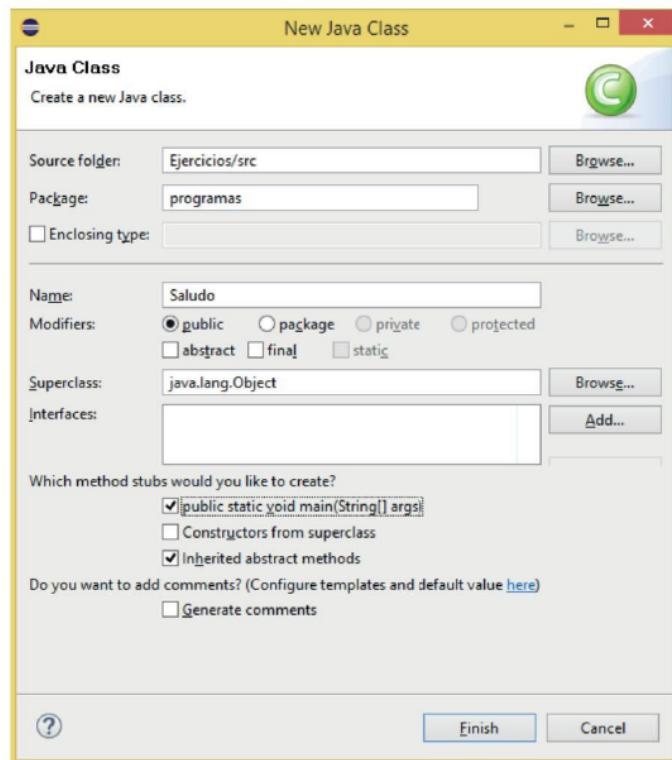


Figura 2.12. Ventana mediante la cual se crea una clase dentro del paquete y del proyecto indicados en las opciones Package y Source folder, respectivamente. Se debe asignar un nombre a la clase.

Tras crear la clase, aparece una pantalla (Figura 2.13), en la que, a modo de ejemplo, se puede escribir una instrucción dentro del método *main* de la clase para mostrar un mensaje, por ejemplo, «¡Hola, mundo!». En esta pantalla, se distinguen las siguientes áreas:

- **Área de proyectos:** en la parte izquierda, *Package Explorer*, se puede navegar por todos los proyectos del espacio de trabajo y por los elementos que los componen.
- **Área de edición:** en la parte central, se puede escribir el código fuente de los programas. En el código que se muestra en la Figura 2.13, el texto aparece resaltado en diferentes colores; así, por ejemplo, las palabras clave del lenguaje aparecen en color morado. Si se escribe una instrucción errónea, el editor la señala subrayando en color rojo lo incorrecto y marcando la línea con una x dentro de un cuadrado con fondo de color rojo.
- **Outline:** en la parte derecha, se muestra el esquema de la clase cuyo código se está editando. Así, se accede más rápidamente a sus métodos y atributos.
- **Consola Java:** en la parte inferior, se muestra el resultado de la ejecución de los programas o los errores de ejecución, en su caso.

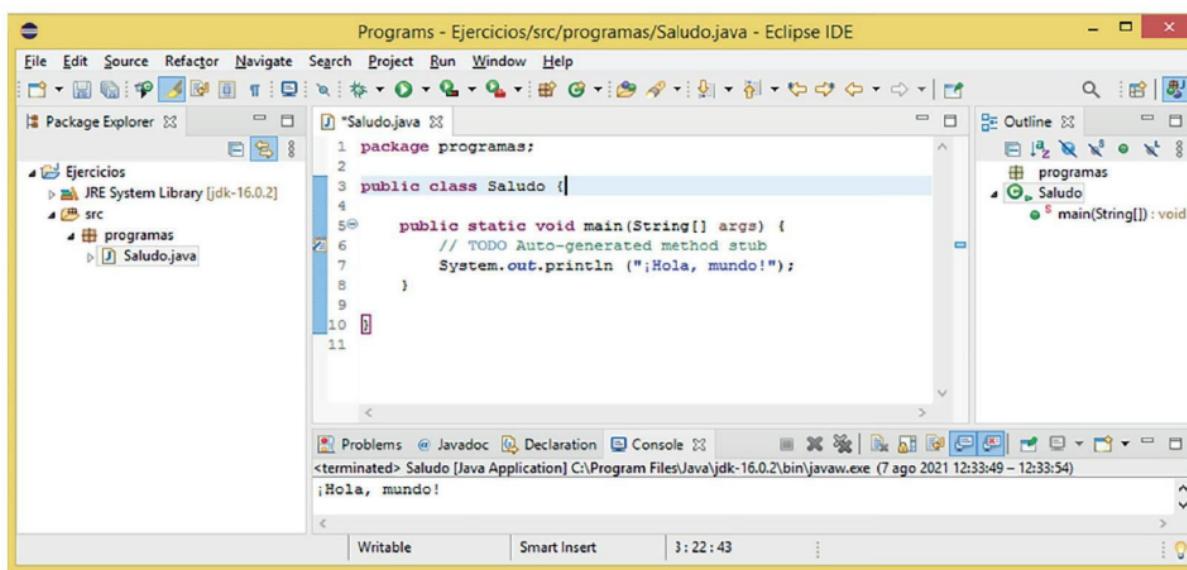


Figura 2.13. Aspecto del IDE Eclipse cuando se está escribiendo el código de un programa, en este caso, uno que tan solo muestra en pantalla el famoso mensaje «¡Hola, mundo!»

Una vez se haya creado un programa en Eclipse con al menos una clase, ya es posible ejecutarlo. Para ello, se debe mostrar en pantalla la clase cuyo código se desea ejecutar y realizar una de las siguientes acciones:

- Seleccionar la clase en cuestión en el explorador del proyecto (parte izquierda de la pantalla) y elegir, en el menú contextual (pulsando el botón derecho del ratón), la opción *Run As → Java Application*.
- Activar la opción de menú *Run → Run*.
- Hacer clic en el icono .

Seguidamente, aparece el resultado de la ejecución en la consola Java (parte inferior de la pantalla).

Por cada proyecto creado en Eclipse, se genera, dentro de la carpeta que se ha asignado al espacio de trabajo (*workspace*), una carpeta con el nombre asignado al proyecto. Es importante saber que, dentro de esa carpeta, habrá a su vez dos carpetas:

- La **carpeta src**, que contiene los ficheros con el código fuente Java correspondiente a cada una de las clases que contiene la aplicación que se está creando. En esta carpeta, habrá una carpeta por cada paquete (si se han creado) y, dentro de estas, un archivo de tipo texto con extensión *.java* por cada clase.
- La **carpeta bin**, en la que se encuentran los ficheros con las clases objeto. En esta carpeta, hay, al igual que en la *src*, una carpeta por cada paquete (si se han creado) y, dentro de estas, un archivo con la extensión *.class* por cada clase.

Actividad resuelta 2.1

Selección de un espacio de trabajo usado recientemente al iniciar Eclipse

¿Es posible, al iniciar Eclipse, seleccionar un espacio de trabajo diferente del que aparece por defecto y que haya sido usado recientemente sin tener que navegar por la estructura de carpetas y ficheros del equipo?

Solución

Sí, es posible. Si se pulsa sobre el enlace *Recent Workspaces* que aparece en la parte inferior izquierda de la ventana que se muestra en la Figura 2.9, aparecerá una lista con los espacios de trabajo empleados recientemente entre los que se puede seleccionar el que se desee, haciendo clic sobre el nombre correspondiente.

Actividad resuelta 2.2

Cambio del espacio de trabajo en Eclipse

Al iniciar Eclipse, aparece una ventana (Figura 2.9) donde se pide el nombre y la ubicación del espacio de trabajo que se quiere usar. Pues bien, ¿es posible cambiar de espacio de trabajo una vez que se ha comenzado a usar Eclipse sin necesidad de reiniciar el IDE?

Solución

La respuesta es afirmativa, pues la opción de menú *File → Switch Workspace* permite seleccionar uno de los espacios de trabajo usados recientemente. Si ninguno de los espacios de trabajo nos interesa, se puede clicar sobre la opción *Other*. En este caso, aparece una ventana como la representada en la Figura 2.9, en la que haciendo clic en el botón *Browse...*, se puede seleccionar, como espacio de trabajo, una carpeta de nuestro equipo.

■ ■ ■ 2.4.2. Edición de programas y generación de archivos ejecutables en NetBeans

Al acceder al IDE NetBeans, para crear una aplicación con este entorno, lo primero que se debe hacer, como con el entorno Eclipse, es crear un proyecto. Con este fin, se activa la opción de menú *File → New Project*. En la ventana que se muestra en la Figura 2.14,

se puede observar que con NetBeans, se ofrece la posibilidad de crear proyectos no solo en Java, sino también en otros lenguajes, como C, C++ o PHP. En este caso, se selecciona la opción de menú *Java with Maven* → *Java Application*. La opción *Java with Maven* → → *Java Frontend Application* permite crear una aplicación Java con GUI, aunque esto también es posible simplemente añadiendo un panel a una aplicación Java creada con la opción *Java Application*.

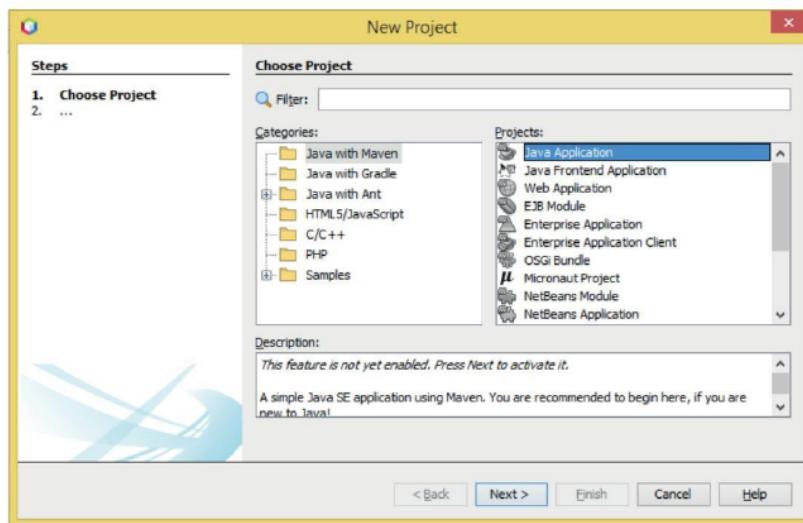


Figura 2.14. Ventana que se muestra al seleccionar la opción de menú File → New Project y que permite crear proyectos en Java, C, C++, PHP, etcétera.

Si se elige la opción *Java with Maven* → *Java Application*, puede ocurrir que se indique a continuación que se debe activar el soporte para Java SE; si este fuera el caso, se pulsa el botón *Activate* (Figura 2.15).

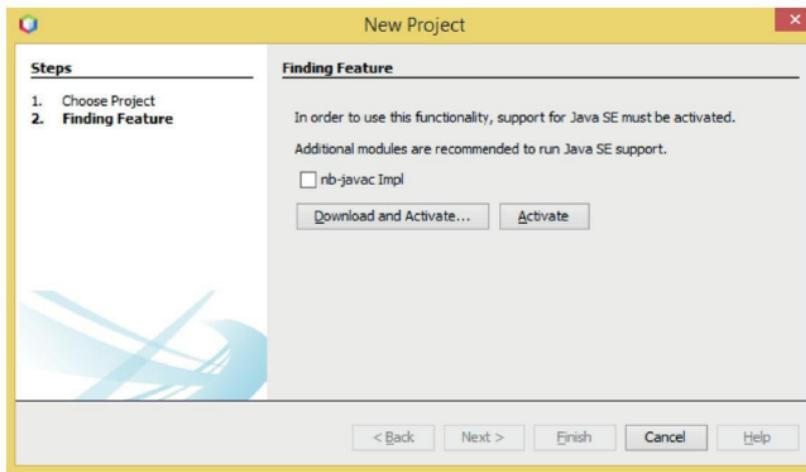


Figura 2.15. Ventana en la que se solicita activar la asistencia para Java SE, para lo que se pulsa sobre el botón *Activate*.

Tras la activación, si esta ha sido necesaria, en la ventana que se muestra en la Figura 2.16, se da nombre al proyecto y, en el campo *Project Location*, se indica su ubicación en nuestro equipo. En los otros campos, se puede dejar la información que aparece por defecto. Al hacer clic en el botón *Finish*, se crea el proyecto.

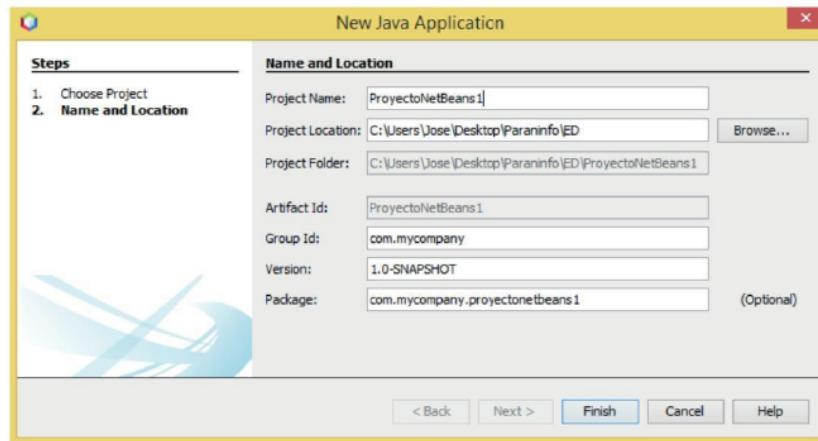


Figura 2.16. Ventana en la que se da un nombre al proyecto y se indica dónde se va a almacenar en el equipo. Si se desea cambiar la ubicación por defecto, se hace clic en Browse para seleccionar una nueva ubicación.

Una vez generado un proyecto, es posible crear en él paquetes y clases, al igual que en Eclipse. En NetBeans, se selecciona el menú contextual del proyecto en cuestión y se activa la opción de menú *New* → *Java Package* o *New* → *Java Class*, respectivamente. Por ejemplo, se puede añadir al proyecto creado anteriormente un paquete llamado *programas*. Este aparecerá inmediatamente en el explorador de proyectos (arriba a la izquierda) dentro de la carpeta *Source Packages*. Luego, se abre el menú contextual de este paquete para crear en él una clase llamada *Saludo* con el fin de mostrar el mensaje «¡Hola, mundo!».

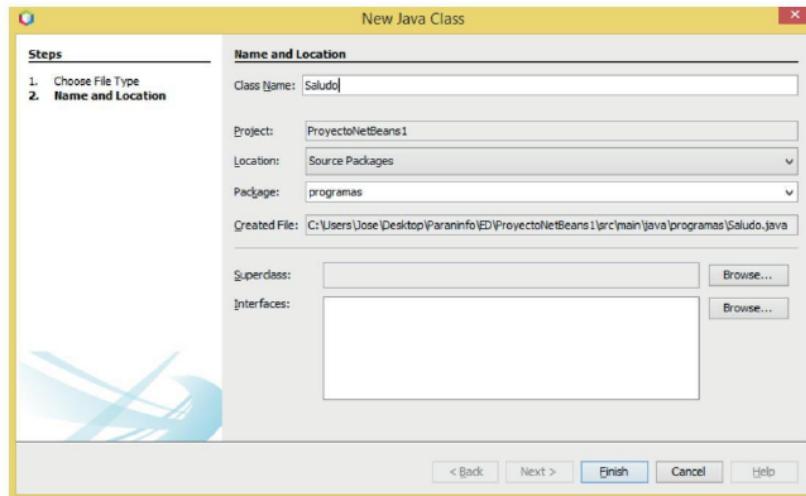


Figura 2.17. Ventana en la que se crea una clase llamada *Saludo* dentro del paquete *programas* del proyecto *ProyectoNetBeans1*.

Al clicar en el botón *Finish* (véase Figura 2.17), se procede a la creación de la clase y se muestra una pantalla como la de la Figura 2.18, en cuya parte derecha, se puede escribir el código fuente de la clase (área de edición). Hay disponible, al igual que en Eclipse, un área reservada al explorador de proyectos (parte superior izquierda) y un área que recibe el nombre de *Navigator*, donde se muestra el esquema de la clase que se está codificando (parte inferior izquierda).

Al igual que se hizo para la clase creada con Eclipse, a modo de ejemplo, se puede incluir, dentro de la clase *Saludo*, un método *main* que muestre el saludo «¡Hola, mundo!».

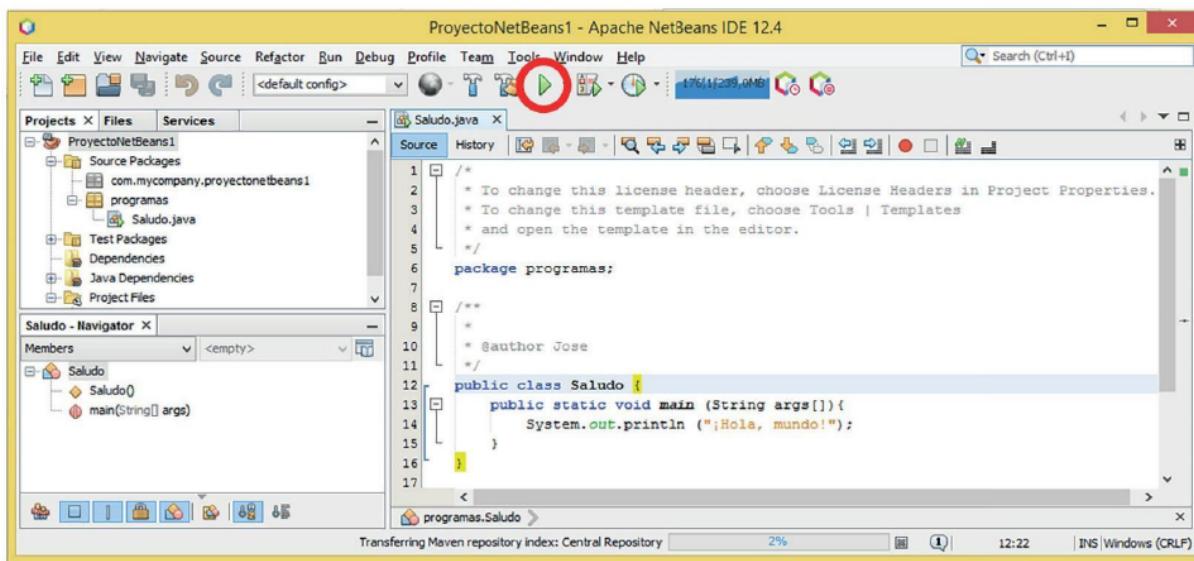


Figura 2.18. Aspecto de la pantalla de IDE NetBeans cuando se está escribiendo el código de un programa. En este caso, se ha creado un programa que tan solo muestra en pantalla el famoso mensaje «¡Hola, mundo!».

Para ejecutar el programa, hay disponibles varias opciones:

- Seleccionar el proyecto en el explorador de proyectos y elegir en el menú contextual la opción *Run*.
- Activar la opción de menú *Run* → *Run Project*.
- Hacer clic en el ícono .

Independientemente de la opción utilizada, emergerá una ventana como la que se muestra en la Figura 2.19, donde se permite elegir una clase del proyecto como clase *main*, es decir, como la clase cuyo método *main* iniciará la ejecución de la aplicación. En este caso, se selecciona la clase *Saludo* dentro del paquete *programas* (*programas.Saludo*). Se puede indicar, además, si se desea que esta elección sea recordada para siempre o solo en la sesión actual.

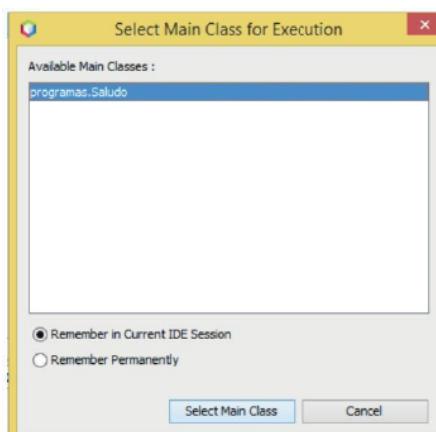


Figura 2.19. Ventana en la que se selecciona la clase cuyo método *main* se quiere ejecutar.

Como en Eclipse, el resultado de la ejecución se muestra debajo del área de edición. Esta área viene etiquetada con la palabra *Output* (salida).

■ 2.5. Instalación y desinstalación de módulos

En los entornos de desarrollo, es frecuente instalar también módulos o *plug-ins* adicionales para la realización de tareas para las que no hay componentes incorporados en la versión instalada.

Argot técnico



El término *plug in* es un verbo inglés que se puede traducir como «enchufar» o «conectar». Se puede interpretar que un *plug-in* es un módulo o componente de software que añade una característica o un servicio adicional a un software existente y que se *enchufa* o conecta a dicho sistema.

■ 2.5.1. Instalación y desinstalación de módulos en Eclipse

Se pueden consultar cuáles son los módulos instalados y los que hay disponibles para Eclipse, seleccionando en el menú la opción *Help → Eclipse Marketplace*. Se muestra entonces una pantalla, como la de la Figura 2.20. Si ya tiene algún módulo instalado, puede ocurrir que Eclipse haya detectado actualizaciones para alguno de ellos, en cuyo caso se indicará, como se puede observar en la figura. Si se desea actualizar los módulos, que es lo más deseable, se hace clic en el botón *Update now*.



Figura 2.20. Ventana que se muestra al seleccionar la opción de menú *Help → Eclipse Marketplace*, útil para instalar o desinstalar módulos en Eclipse. Si hay actualizaciones disponibles para los módulos instalados, Eclipse informa sobre este hecho, como es el caso.

En la siguiente ventana, se solicita confirmar las actualizaciones y, posteriormente, se aceptan los acuerdos de licencia correspondientes, tras lo cual comienza la actualización de los módulos afectados, que se lleva a cabo en segundo plano. Una vez que la actualización haya finalizado, se indica que es necesario reiniciar Eclipse para que esta tenga efecto.

■■■ Instalación de WindowBuilder en Eclipse

Dado que Eclipse no tiene incorporada de serie la opción de crear aplicaciones con interfaces gráficas de usuario (GUI, *graphical user interface*), es preciso instalar un módulo que permita crear este tipo de programas. Swing es la librería que permite crear aplicaciones con interfaces gráficas de usuario en Java.

Para instalar el módulo que incorpora Swing a Eclipse, se selecciona de nuevo la opción de menú *Help → Eclipse Marketplace* y, en la ventana que aparece en pantalla, se escribe *WindowBuilder* en el cuadro de texto que hay a la derecha de la palabra *Find*. WindowBuilder es el módulo que permite incorporar Swing a nuestra instalación de Eclipse. Una vez escrito, se pulsa el botón *Go* a la derecha y aparece más abajo, entre otros, el módulo *WindowBuilder 1.9.5*. Seguidamente, se clica sobre el botón *Install* correspondiente a este módulo (véase Figura 2.21).

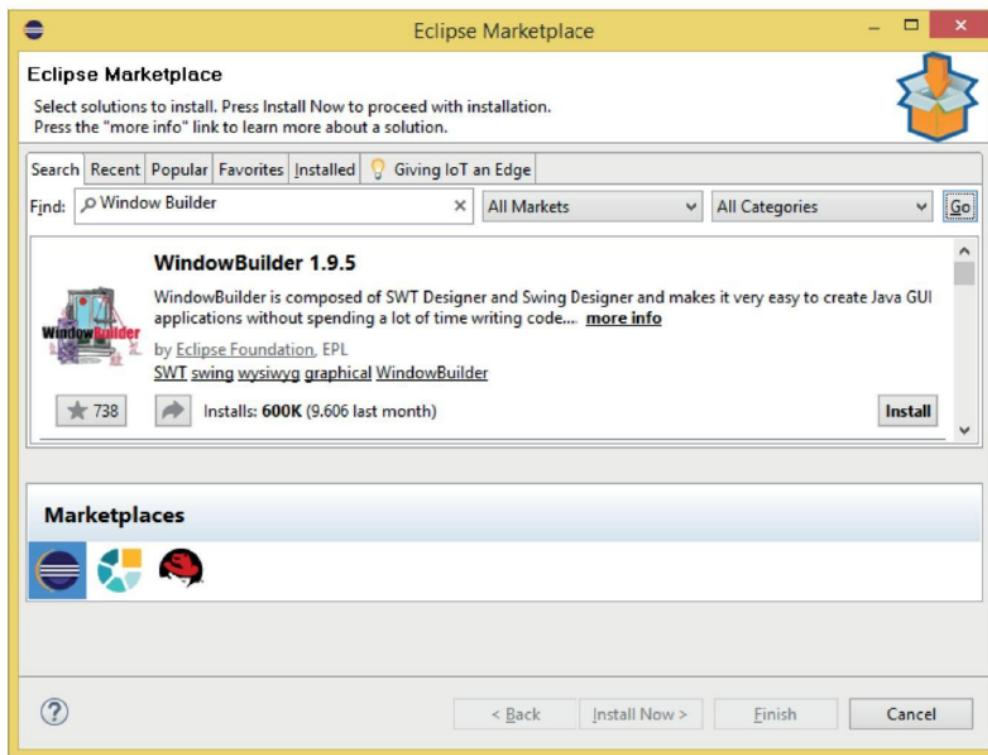


Figura 2.21. En Eclipse Marketplace, se muestra el módulo WindowBuilder preparado para su instalación.

A continuación, aparece una nueva ventana con los elementos del módulo que se va a instalar (Figura 2.22), de entre los cuales se puede seleccionar solo algunos o todos ellos.

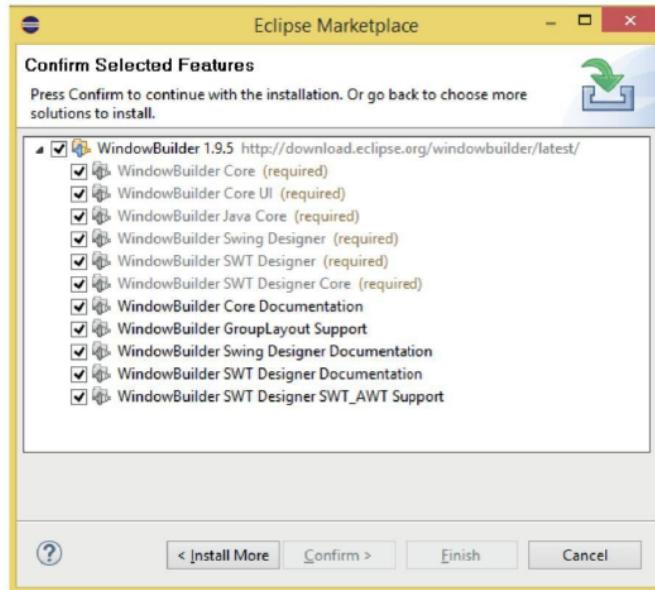


Figura 2.22. Elementos del módulo WindowBuilder disponibles para su instalación. Se marca aquellos que se desea instalar.

Una vez elegidos los componentes que se desean instalar, se pulsa el botón *Confirm* y se acepta el acuerdo de licencia en la siguiente ventana, tras lo cual, comienza la instalación en segundo plano. Al finalizar la instalación, debe reiniciarse Eclipse para que tenga efecto la actualización del software.

El módulo WindowBuilder sirve para la creación de aplicaciones con una interfaz gráfica de usuario, en las que se presentan diferentes tipos de controles, como los que se suelen mostrar en los formularios. Algunos de ellos son los siguientes:

- **Etiquetas (*JLabel*):** contienen texto que no se puede modificar.
- **Campos de texto (*JTextField*):** sirven para escribir texto en su interior.
- **Cuadros de contraseña (*JPasswordField*):** en ellos, se escribe texto que se desea que no sea visible, de manera que el texto introducido queda oculto, generalmente, por medio de puntos.
- **Áreas de texto (*JTextArea*):** se utilizan para introducir texto que ocupa varias líneas.
- **Botones (*JButton*):** sirven para dar una orden.
- **Casillas de activación (*JCheckBox*):** mediante estas, la persona usuaria puede elegir una opción o varias, de manera que pueden estar activadas o no. Si hay varias casillas marcadas, pueden estar activadas de manera independiente.
- **Botones de opción (*JRadioButton*):** son similares a las casillas de activación, pero los botones de opción son excluyentes, mientras que aquellas no lo son.
- **Listas (*JList*):** permiten escoger un valor de entre varios, habiendo dos posibilidades: la selección de una sola opción o de varias (con la tecla Ctrl pulsada).
- **Cuadros combinados (*JComboBox*):** son listas desplegables que pueden ser utilizadas también como cuadros de texto, en los que se puede escribir.

En la Figura 2.23, se muestran algunos de los controles descritos en la página anterior.



Figura 2.23. Ventana creada con Swing gracias a la instalación del módulo WindowBuilder en Eclipse y en la que se muestran distintos tipos de controles.

Para crear una aplicación visual, se activa la opción del menú *New → Other → WindowBuilder → Swing Designer → Application Window*.

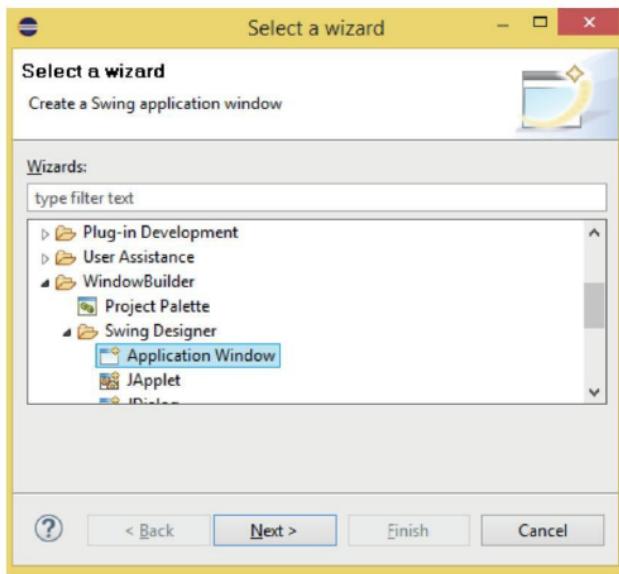


Figura 2.24. Ventana para crear una aplicación visual con Eclipse.

Una vez hecho esto, tras indicar el paquete en el que desea ubicar la aplicación visual, se da un nombre a esta. Entonces, aparece una pantalla, como la se muestra en la Figura 2.25, en la que aparece el código de la aplicación, que ha sido generado de manera automática.

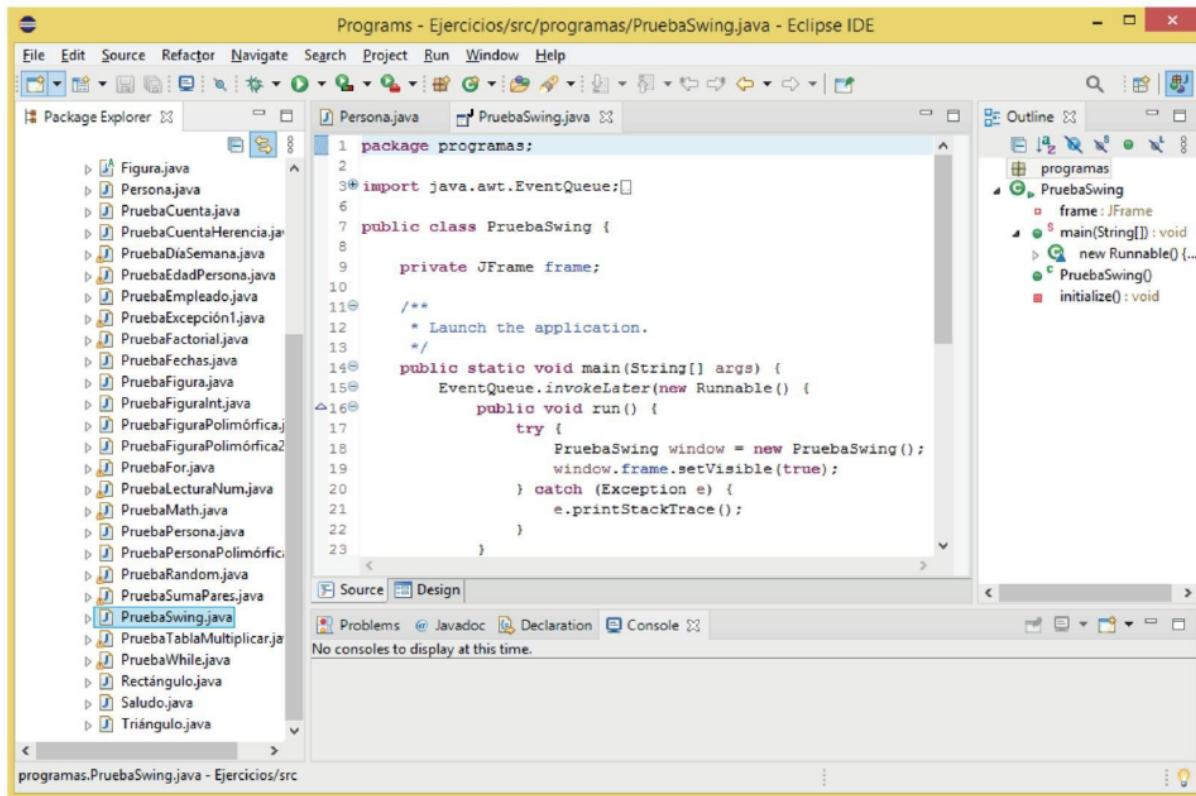


Figura 2.25. Pantalla que aparece inmediatamente después del momento en que se crea una aplicación visual. En el área central se muestra el código fuente de la aplicación porque está seleccionada por defecto la pestaña Source, pero se puede visualizar su diseño seleccionando la pestaña Design.

Si se clica sobre la pestaña *Design*, que está situada en la parte inferior del área central, aparece el área de diseño de la ventana de la aplicación (Figura 2.26). A la izquierda de dicha área, se muestra una paleta con diversos elementos. Los más importantes son los componentes o controles que se pueden añadir a la ventana que se está creando (etiquetas, campos de texto, botones, casillas de activación, etc.).

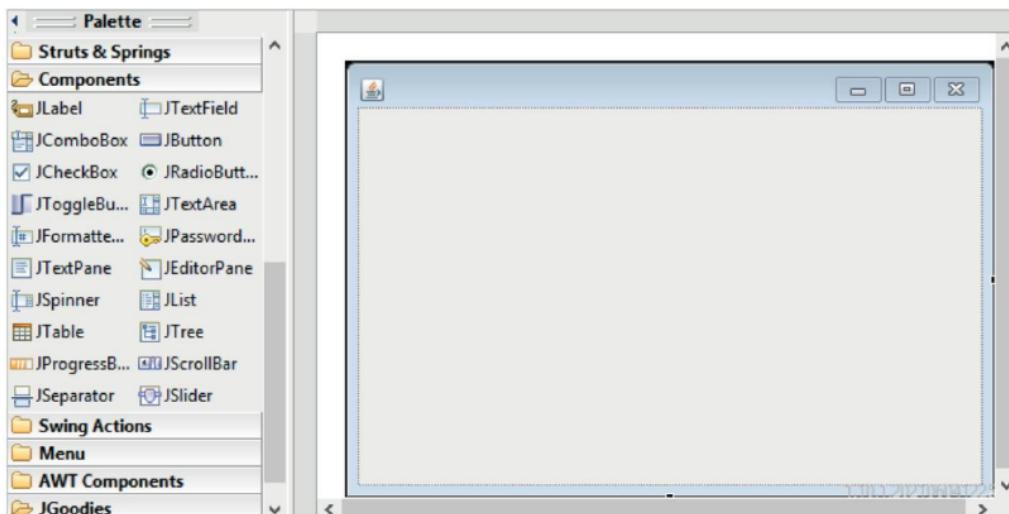


Figura 2.26. Área de diseño de la ventana que se está creando y que inicialmente está en blanco, pero en la que se pueden colocar los controles o componentes que aparecen en la paleta de la izquierda.

■■■ Instalación de Papyrus SysML en Eclipse

Otro módulo que resulta interesante instalar es el que permite crear diagramas UML (por sus siglas en inglés, *Unified Modeling Language*) en Eclipse. Para ello, se activa la opción del menú *Help → Eclipse Marketplace* y se escribe *Papyrus* en el cuadro de texto que hay a la derecha de la palabra *Find* (Figura 2.27). Papyrus es un software que permite crear, editar y visualizar diagramas UML 2.5.

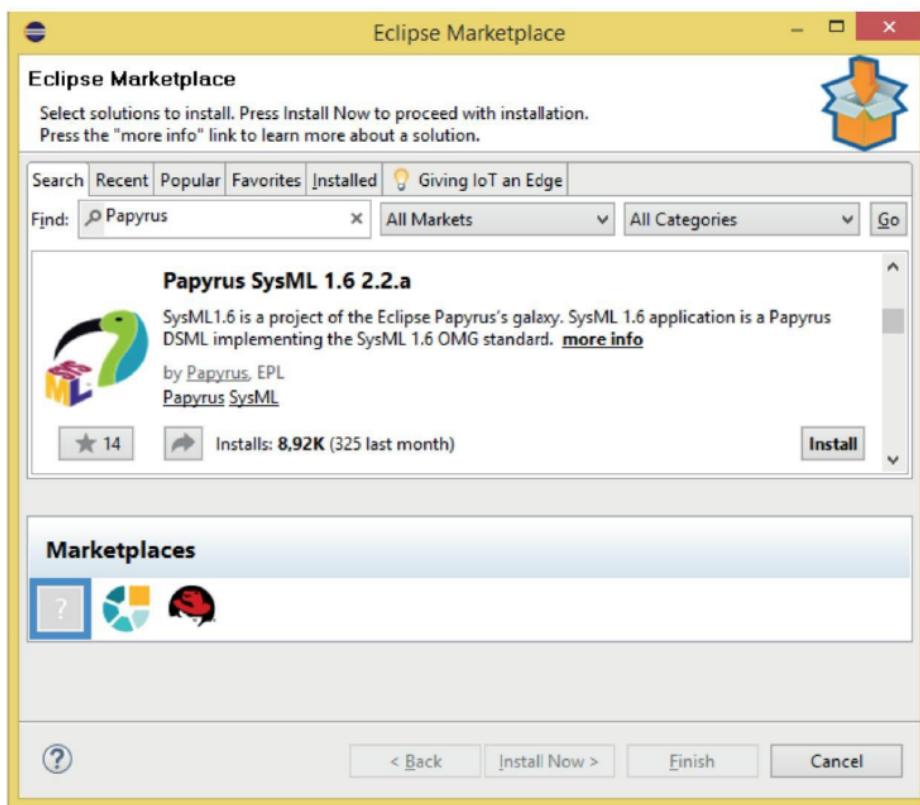


Figura 2.27. Ventana de Eclipse Marketplace en el que se muestra el módulo Papyrus SysML preparado para su instalación.

Para instalar el módulo Papyrus SysML, se pulsa sobre el botón *Install*, se seleccionan todos sus componentes y, finalmente, se hace clic en el botón *Confirm*. Luego, se debe reiniciar el IDE Eclipse para que la instalación tenga efecto.

Actividad propuesta 2.1

Instalación de un módulo en Eclipse

Instala en Eclipse un módulo que permita crear programas en Python y ejecutarlos.

■■■ Modificación y desinstalación de módulos en Eclipse

Tras la instalación de un módulo en Eclipse, este aparece en la pestaña *Installed* de Eclipse Marketplace (véase Figura 2.28).

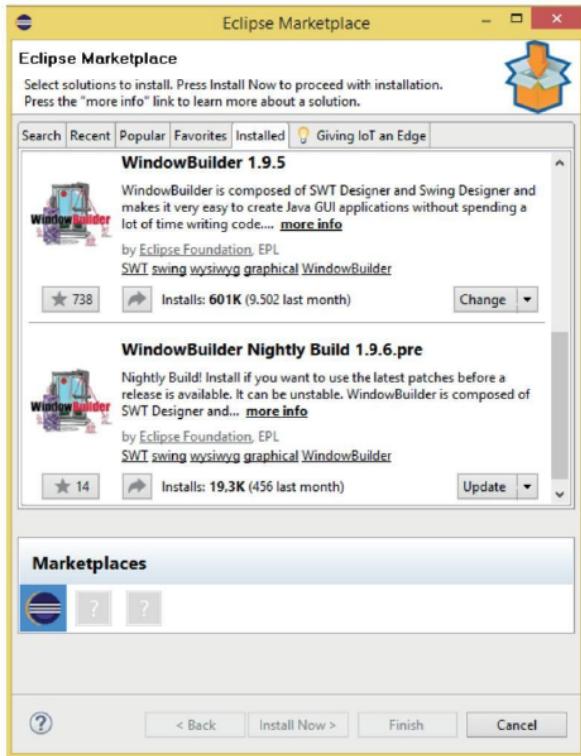


Figura 2.28. Para saber cuáles son los módulos instalados en Eclipse Marketplace, se pincha sobre la pestaña Installed. Estos módulos se pueden actualizar (si hay alguna actualización disponible), cambiar o desinstalar.

Por cada módulo instalado, en la parte inferior derecha, aparece un botón en el que puede leer *Update* o *Change*. Si aparece el texto *Update*, quiere decir que hay alguna actualización disponible para ese módulo, en cuyo caso, se puede activar la opción *Update* que aparece al desplegarla para llevar a cabo la actualización. También aparece la opción *Change*, que sirve para modificar elementos del módulo instalado. Así, si se pincha esta opción para el módulo WindowBuilder 1.9.5, aparece la ventana que se muestra en la Figura 2.29.

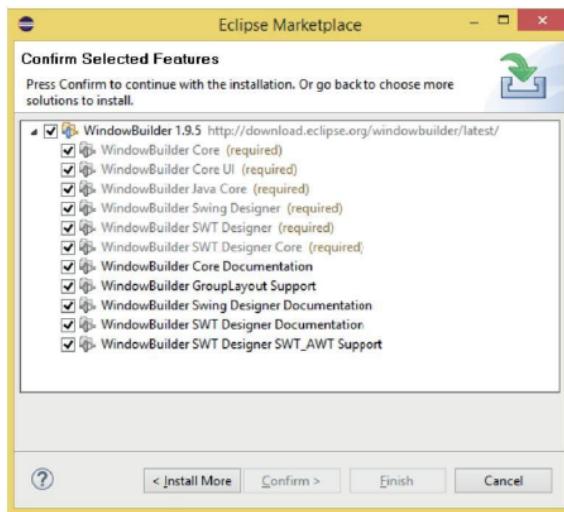


Figura 2.29. Ventana que muestra los elementos del módulo PowerBuilder 1.9.5 instalados por si se desea eliminar alguno o añadir otro que no esté incorporado.

En esta ventana, se puede desmarcar alguno de los elementos del módulo, en cuyo caso se activa el botón *Confirm* y se procede a su desinstalación. En caso de haber seleccionado un módulo que no incluya todos los elementos disponibles para este, también se podrá instalar algún elemento adicional.

Para todos los módulos instalados, también aparece la opción *Uninstall*, que habrá que seleccionar si se desea desinstalarlo, para lo cual se pedirá confirmación.

Actividad propuesta 2.2

Modificación de elementos de un módulo instalado en Eclipse

Desinstala alguno de los elementos opcionales que componen el módulo para programar en Python instalado en la Actividad propuesta 2.1.

Actividad propuesta 2.3

Desinstalación de un módulo en Eclipse

Desinstala en Eclipse el módulo instalado en la Actividad propuesta 2.1.

2.5.2. Instalación y desinstalación de módulos en NetBeans

Al igual que en Eclipse, en NetBeans, se pueden instalar módulos o *plug-ins* que permiten realizar tareas para las que no hay complementos incorporados en la instalación del IDE. En este entorno, se emplea la opción de menú *Tools → Plugins*. Si se activa esta, aparece una ventana con varias pestañas.

Si hay actualizaciones disponibles para el IDE, estas aparecerán en la pestaña *Updates*. También se puede clicar en el botón *Check for Updates* para que en ese momento se busquen actualizaciones disponibles.

En la pestaña *Available Plugins*, es posible visualizar los módulos disponibles para su instalación en NetBeans, como se muestra en la Figura 2.30. Al marcar cada módulo, aparece su descripción en la parte derecha de la ventana.

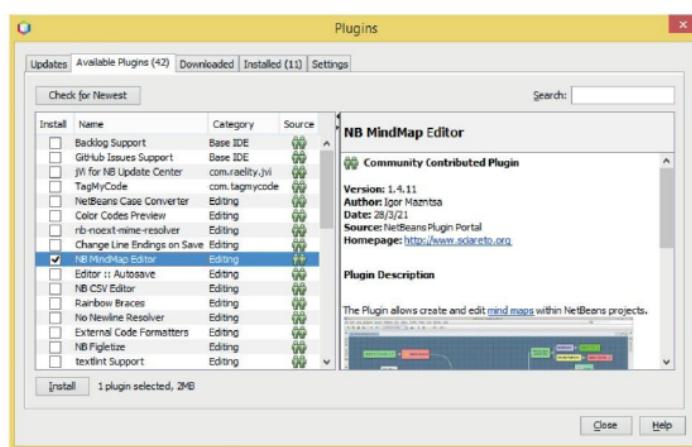


Figura 2.30. Módulos o plug-ins disponibles para su instalación en Apache NetBeans.

A modo de ilustración, aquí se explica cómo instalar el módulo NB MindMap Editor, que permite crear y editar mapas mentales dentro de los proyectos de NetBeans. A tal fin, se selecciona dicho módulo y se hace clic en el botón *Install*. Tras aceptar el acuerdo de licencia, comienza su instalación, que se puede indicar que tenga lugar en segundo plano (casilla de verificación *Run in Background*). Cuando ya está instalado, es necesario reiniciar el IDE para lo que habrá que indicar si se desea reiniciar el entorno en ese momento o más tarde.

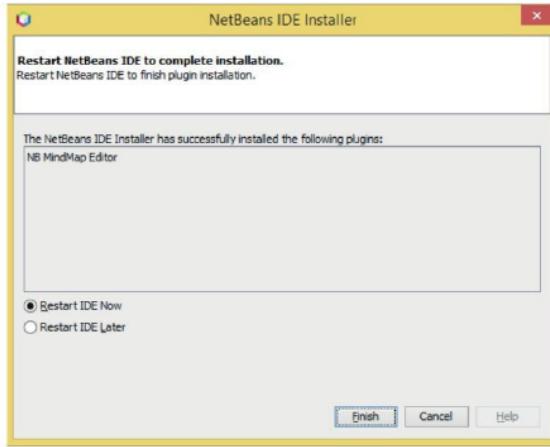


Figura 2.31. Ventana que se muestra al finalizar la instalación de un módulo en NetBeans. Para que tenga efecto, hay que reiniciar el IDE, y abajo se pregunta si se desea hacerlo en ese momento o más tarde.

Una vez instalado un módulo, aparece este en la pestaña *Installed*. Como se puede observar en la Figura 2.32, en esta pestaña, se indica para cada módulo instalado si está activo o no en la columna *Active*.

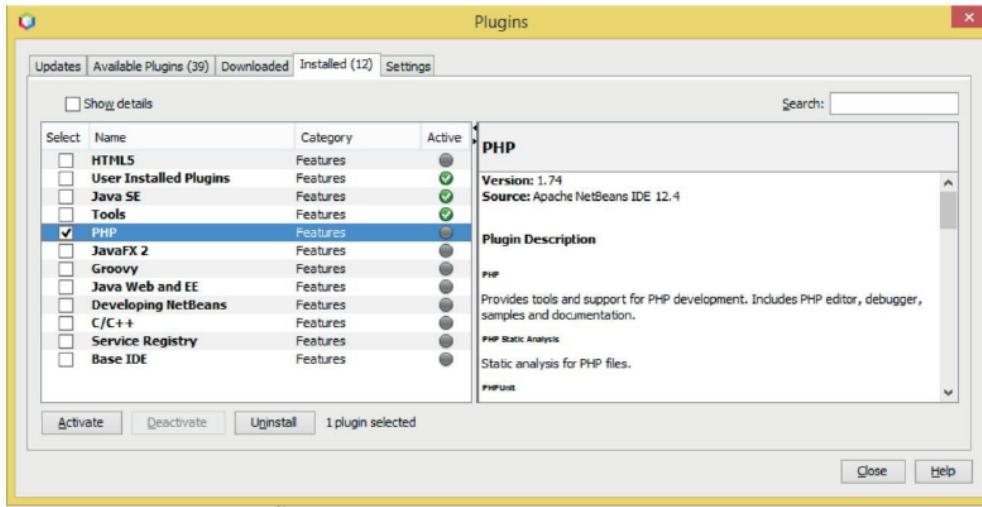


Figura 2.32. Visualización de los módulos o plug-ins instalados en Apache NetBeans. Es posible activar y desactivar cada módulo instalado y también desinstalarlo, tras seleccionarlo en la columna Select.

Para activar cualquier módulo, tan solo es necesario marcarlo y hacer clic en el botón *Activate*; en caso de que se quiera desinstalar, se pulsa sobre el botón *Uninstall*.

2.6. Mecanismos de actualización

Es importante mantener actualizado el entorno de desarrollo que se utilice, así como los módulos o *plug-ins* que se han instalado.

En el Apartado 2.5.1, se indicó que, en el caso de Eclipse, al seleccionar la opción de menú *Help → Eclipse Marketplace*, si hay actualizaciones disponibles para algún módulo, estas se indican al entrar en *Eclipse Marketplace*, pudiendo proceder desde ahí a su actualización. No obstante, existe también la posibilidad de buscar actualizaciones no solo para los módulos, sino también para el IDE, activando la opción de menú *Help → Check for Updates*, tras lo cual, en la ventana emergente, aparecen todas las actualizaciones disponibles, como se ve en la Figura 2.33.

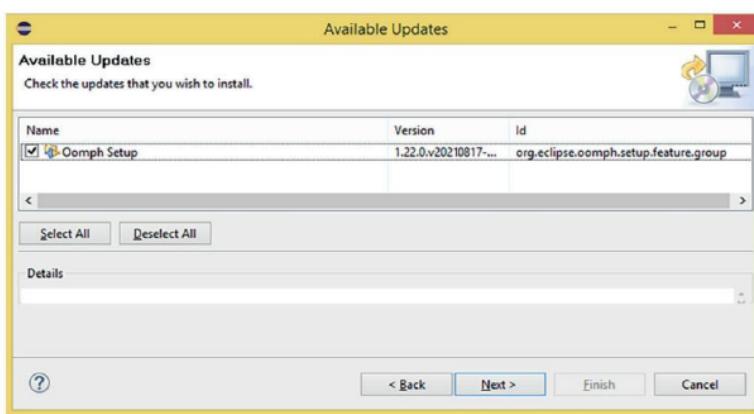


Figura 2.33. Actualizaciones disponibles para Eclipse. Se puede elegir las que se desea instalar.

En esta ventana, es posible seleccionar las que deseen instalar y, una vez seleccionadas, se pulsa en el botón *Next*. Después de esto, se confirma y se inicia el proceso de instalación tras aceptar el correspondiente acuerdo de licencia.

En el caso de NetBeans, para actualizar el IDE, existe la misma opción de menú *Help → Check for Updates*. Una vez activado, aparecen en una ventana las actualizaciones disponibles y se puede proceder entonces a instalarlas pinchando sobre el botón *Next*.

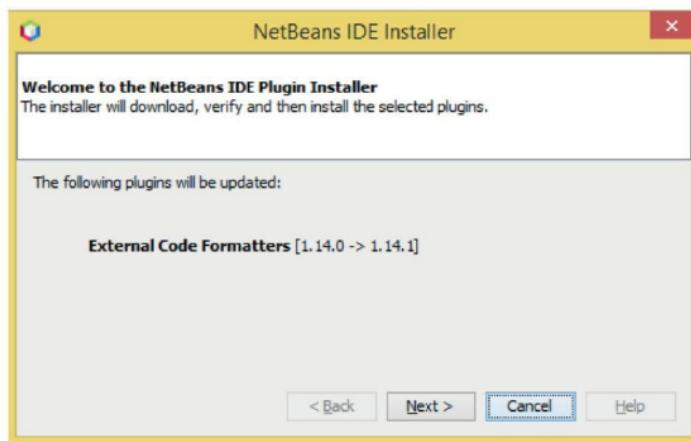
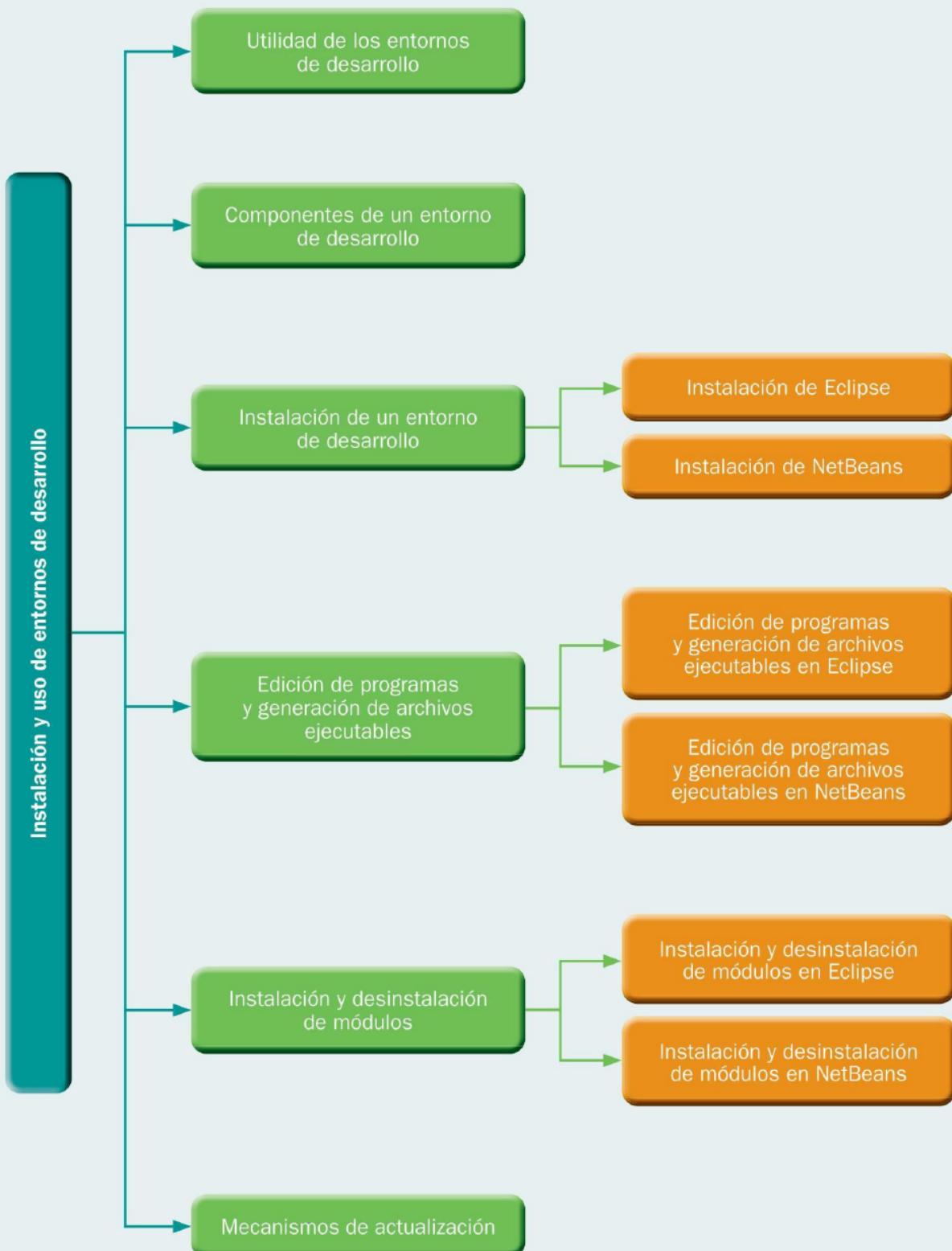


Figura 2.34. Actualizaciones disponibles para Apache NetBeans. Para llevar a cabo su instalación, se hace clic en el botón Next.



Actividades de comprobación

- 2.1.** Las actividades de desarrollo de software que más asistencia reciben por los entornos de desarrollo son:
- a) Programación y pruebas.
 - b) Análisis y diseño.
 - c) Diseño y programación.
 - d) Programación y mantenimiento.
- 2.2.** El conjunto de utilidades que permite la ejecución de programas escritos en Java recibe el nombre de:
- a) JDK.
 - b) JRE.
 - c) JVM.
 - d) IDE.
- 2.3.** ¿Qué nombre recibe el componente de un entorno de desarrollo que permite controlar los cambios que se realizan sobre los programas, creando distintas versiones de estos?
- a) Constructor de interfaz gráfica.
 - b) Depurador.
 - c) Editor de textos.
 - d) Control de versiones.
- 2.4.** Para compilar un programa escrito en Java desde el indicador del sistema se usa el compilador llamado:
- a) `javac`.
 - b) `java`.
 - c) `compiler`.
 - d) Ninguna de las respuestas anteriores es correcta.
- 2.5.** Los ficheros que contienen el código fuente de los programas escritos en Java tienen extensión:
- a) `.java`.
 - b) `.class`.
 - c) `.exe`.
 - d) `.txt`.
- 2.6.** Los entornos de desarrollo Eclipse y Apache NetBeans son:
- a) De código cerrado y gratuitos.
 - b) De código cerrado y de pago.
 - c) De código abierto y gratuitos.
 - d) De código abierto y de pago.

- 2.7.** Por cada proyecto Java creado en Eclipse, se crea una carpeta en el espacio de trabajo (*workspace*) especificado y, dentro de esta carpeta, hay una carpeta cuyo nombre es _____. Esta carpeta contiene una subcarpeta por cada paquete del proyecto y, dentro de cada una de estas subcarpetas, hay un archivo con la extensión _____ para cada clase creada en el paquete. Elige la opción correcta para llenar los huecos en el texto anterior:
- a) bin / class.
 - b) src / class.
 - c) bin / java.
 - d) Ninguna de las respuestas anteriores es correcta.
- 2.8.** Indica la afirmación correcta en relación con las similitudes y diferencias entre los entornos de desarrollo Eclipse y Apache NetBeans:
- a) En los dos entornos de desarrollo se debe indicar la ubicación del espacio de trabajo o *workspace* donde se almacenarán los proyectos.
 - b) En Eclipse, se puede ejecutar cada clase de manera independiente seleccionando, desde su menú contextual, la opción de ejecución correspondiente, mientras que en NetBeans esto no es posible.
 - c) En los dos entornos de desarrollo, tras instalar un módulo, es posible modificar los elementos de los que consta el módulo instalado.
 - d) Ninguna de las respuestas anteriores es correcta.
- 2.9.** En cuanto a la creación de aplicaciones con interfaces gráficas de usuario en Apache NetBeans:
- a) Para crear estas aplicaciones es necesario instalar un módulo o *plug-in*.
 - b) Para crear este tipo de aplicaciones, se debe crear un proyecto usando la opción de menú *File* → *New Project* → *Java with Maven* → *Java Application* y luego se añade un panel a la aplicación.
 - c) Para crear este tipo de aplicaciones, se crea un proyecto empleando la opción de menú *File* → *New Project* → *Java with Maven* → *Java Frontend Application*.
 - d) Las respuestas b y c son correctas.
- 2.10.** En Eclipse y en Apache NetBeans, _____ puede constar de _____ y, en _____, puede haber _____. Elige la opción correcta para llenar los huecos en el texto anterior:
- a) un paquete / varios proyectos / cada proyecto / varias clases.
 - b) un proyecto / varios paquetes / cada paquete / varias clases.
 - c) un paquete / varias clases / cada clase / varios paquetes.
 - d) un proyecto / varias clases / cada clase / varios paquetes.

Actividades de aplicación

- 2.11.** Se sabe que es posible crear y ejecutar programas en Java sin el uso de ningún entorno de desarrollo. ¿Qué ventajas aporta el empleo de un entorno de desarrollo?
- 2.12.** ¿Cuáles son las diferencias entre un editor de textos como el bloc de notas y el editor de textos que incorpora un entorno de desarrollo?

- 2.13. ¿Qué software es necesario instalar en un ordenador antes de poder crear y ejecutar programas en Java independientemente de que se use para ello un entorno de desarrollo o no?
- 2.14. ¿Qué extensión tienen los programas fuente escritos en Java? Y una vez compilados, ¿qué extensión tiene el archivo resultado de dicha compilación?
- 2.15. ¿En qué consiste un constructor de interfaz gráfica? ¿Incorporan este componente Eclipse y Apache NetBeans tras su instalación?
- 2.16. ¿En qué área de la pantalla de Eclipse se puede ver el resultado de la ejecución de un programa abierto? Y ¿en qué área se pueden visualizar los proyectos del espacio de trabajo (*workspace*) y se puede navegar por sus elementos?
- 2.17. ¿Qué controles se emplean dentro de una aplicación con interfaz gráfica de usuario para que la persona usuaria pueda elegir una opción de entre varias que son excluyentes entre sí? Y ¿cuáles se pueden usar para introducir texto que se desea que quede oculto?
- 2.18. Muchos entornos de desarrollo incluyen entre sus componentes un *debugger* o depurador, ¿para qué sirve este componente?
- 2.19. ¿Cuál es la utilidad de los módulos o *plug-ins* en un entorno de desarrollo?
- 2.20. ¿Cómo se pueden instalar módulos o *plug-ins* en Eclipse y en Apache NetBeans?
- 2.21. ¿Es posible modificar los elementos o componentes de un módulo instalado en Eclipse sin desinstalarlo y volver a instalarlo? En caso afirmativo, ¿cómo se puede realizar?
- 2.22. ¿En qué lenguajes de programación es posible escribir programas en Apache NetBeans una vez instalado, es decir, sin instalar ningún módulo adicional o *plug-in*?

Actividades de ampliación

- 2.23. Crea en Eclipse una pantalla como la de la Figura 2.35, para la introducción de los datos de una nueva persona usuaria de una aplicación.

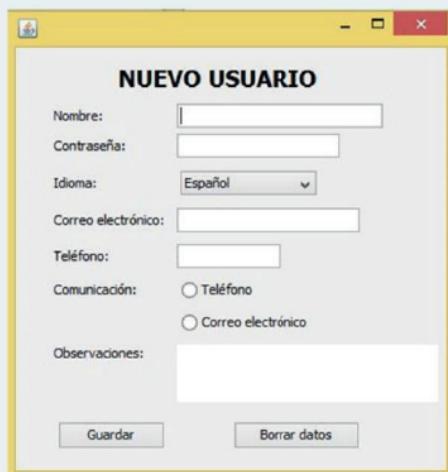


Figura 2.35. Pantalla creada con Swing gracias a la instalación del módulo WindowBuilder en Eclipse.

- 2.24.** Busca en internet información sobre el entorno de desarrollo BlueJ. ¿Qué le diferencia de entornos de desarrollo como Eclipse y Apache NetBeans? ¿Con qué lenguajes de programación se puede utilizar?
- 2.25.** Busca información en la red sobre el entorno de desarrollo Microsoft Visual Studio. ¿Con qué lenguajes de programación se puede utilizar?, ¿es de código abierto o cerrado?, ¿es gratuito?
- 2.26.** Averigua en internet qué lenguajes de programación se pueden utilizar con el entorno de desarrollo Oracle JDeveloper, si se trata de un entorno de código abierto o cerrado y si es gratuito.
- 2.27.** Accede a la página web <https://marketplace.eclipse.org/>. ¿Qué información se muestra en ella y para qué crees que sirve?
- 2.28.** Crea la ventana correspondiente a la Figura 2.35 en Apache NetBeans.

Enlaces web de interés

-  **Eclipse** - <https://www.eclipse.org/>
(Sitio web de Eclipse Foundation con información sobre los productos que pone a disposición del público y la posibilidad de descargarlos)
-  **Linuxtopia** - https://www.linuxtopia.org/online_books/eclipse_guides.html
(Página web con guías sobre Linux y desarrollo de software. Incluye una sección sobre Eclipse)
-  **Apache NetBeans** - <https://netbeans.apache.org/>
(Página web con información sobre el IDE Apache NetBeans, desde la que se puede descargar)
-  **Nube Colectiva** - <https://nubecolectiva.com/>
(Portal web que ofrece tutoriales y recursos sobre desarrollo de software)