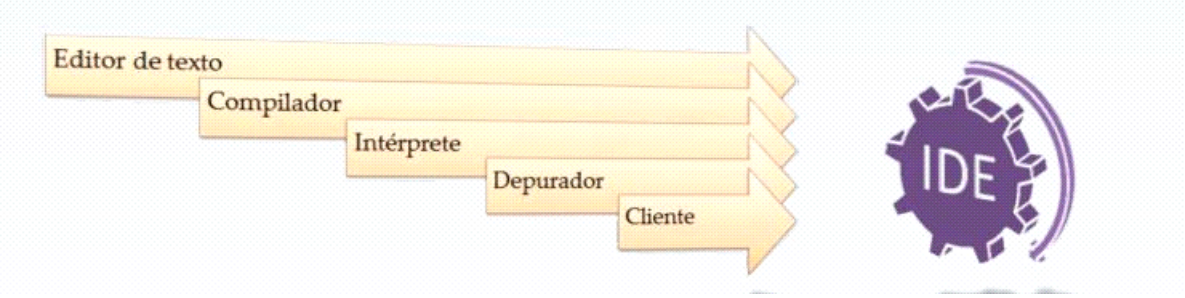
1
Instalación y uso de entornos de desarrollo

Índice:

1. Funciones de un entorno de desarrollo

Un entorno de desarrollo integrado (IDE en inglés Integrated Developmet Environmet) *es una aplicación informática que tiene el objetivo de asistir al programador en la tarea de diseñar y codificar un software mediante la inclusión de múltiples herramientas destinadas a esa tarea.*

Cada IDE tiene unas características y funcionalidades específicas que lo definen. Sin embargo, todos mantienen unos componentes comunes:



- Un **editor de texto** para crear y modificar archivos digitales compuestos únicamente por texto sin formato, conocidos comúnmente como archivos de texto o *texto plano*.
- Un **compilador** que es un programa que permite traducir el código fuente de un programa en lenguaje de alto nivel, a otro lenguaje de nivel inferior (típicamente lenguaje máquina). De esta manera un programador puede diseñar un programa en un lenguaje mucho más cercano a como piensa un ser humano, para luego *compilarlo* a un programa más manejable por una computadora.
- Un **depurador** es un programa usado para probar y eliminar los errores de otros programas (el programa "objetivo"). Ofrecen funciones más sofisticadas tales como correr un programa paso a paso (un paso o animación del programa), parar el programa (breacking), es decir, pausar el programa para examinar el estado actual en cierto evento o instrucción especificada por medio de un breakpoint, y el seguimiento de valores de algunas variables. Algunos depuradores tienen la capacidad de modificar el estado del programa mientras que está corriendo, en vez de simplemente observarlo.
- Posibilidad de ofrecer un sistema de **control de versiones**. Se llama control de versiones a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo. Los sistemas de control de versiones facilitan la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones realizadas (por ejemplo, para algún cliente específico). El control de versiones se realiza principalmente en la industria informática para controlar las distintas versiones del código fuente.
- Factibilidad para ayuda en la construcción de **interfaces gráficas de usuario**. La interfaz gráfica de usuario, conocida también como GUI (del inglés Graphical User Interface) es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz. Su

principal uso, consiste en proporcionar un entorno visual sencillo para permitir la comunicación con el sistema operativo de una máquina o computador.

1.1. Extensiones y herramientas

En un IDE tenemos muchas herramientas con las que poder realizar programas y nos facilitan la edición y depuración de los programas.

Por ejemplo. tenemos el **coloreado de sintaxis** dependiendo del tipo de elemento que se trate (palabras reservadas, comentarios, cadenas de caracteres, etc.). Esta característica tan sencilla mejora la visión y el entendimiento del código con un simple vistazo.

(Sin coloreado)

```
public class ejer1 {
            static int n1=50;
            public static void main(String [] args)
            {
                  int n2=30, suma=0, n3=5; //definición de variables
                  suma=n1+n2;
                  System.out.println("LA SUMA ES: " + suma);
                  suma=suma+n3;
                  System.out.println("LA SUMA ES: " + suma);
            }
}
```

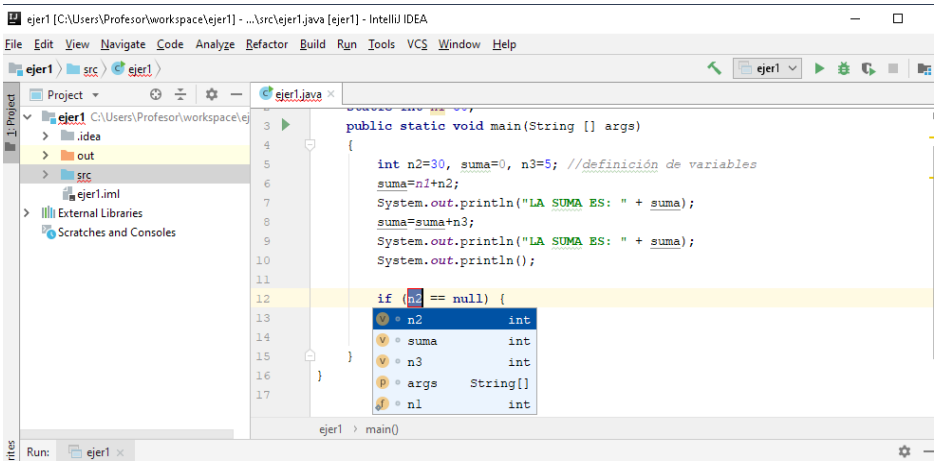
(Con coloreado)

```
public class ejer1 {
         static int n1=50;
         public static void main(String [] args)
         {
              int n2=30, suma=0, n3=5; //definición de variables
              suma=n1+n2;
              System.out.println("LA SUMA ES: " + suma);
              suma=suma+n3;
              System.out.println("LA SUMA ES: " + suma);

         }
}
```

Otra herramienta importante es el **autocompletado de código**. En IntelliJ, cada vez que comenzamos a escribir una palabra reservada podemos hacer que nos aparezca el autocompletado con las teclas **Control + Barra espaciadora**, aparece un listado de sugerencias por el que podemos navegar para elegir la que queramos. También es muy útil mostrar los métodos y propiedades de un objeto o clase cuando escribimos el punto **“.”** Para acceder a ellos.

Veamos un ejemplo donde comenzamos a escribir la instrucción **if** y con Control + Barra espaciadora, nos aparece el autocompletado:

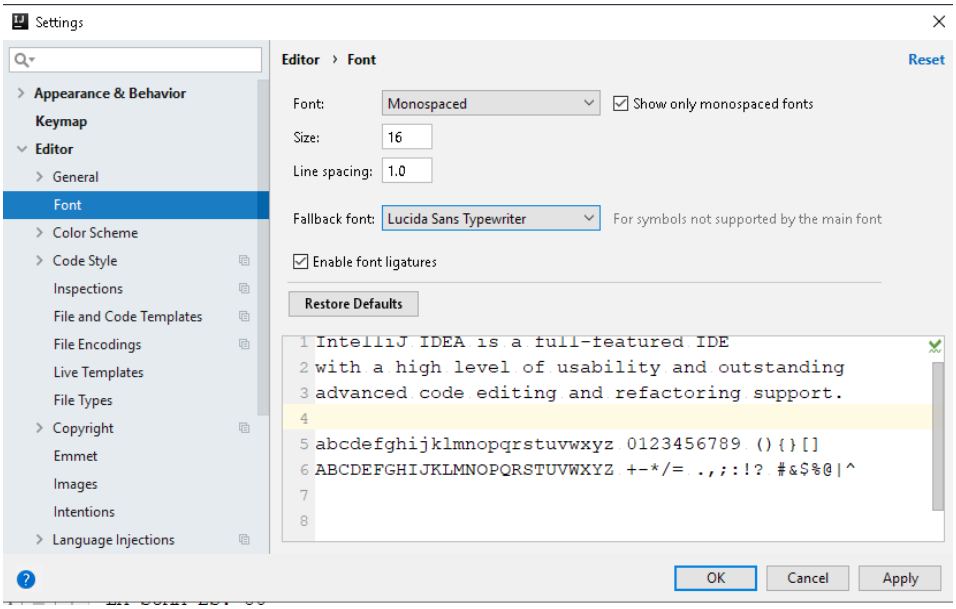


Los IDE también nos ofrecen herramientas de refactorización, una ejecución en depuración y muchas opciones más con el objetivo de facilitar y acortar el tiempo dedicado al desarrollo de software.

Muchas herramientas no vienen integradas en el IDE y se pueden incluir mediante extensiones (pluggins) que añaden, modifican y mejoran el IDE.

1.2. Personalización y configuración

Los IDE son configurables a la medida y necesidad de cada programador, su objetivo es ofrecer al usuario una aplicación amigable con la que trabajar, por eso es importante que se pueda configurar a medida. Por ejemplo, si queremos cambiar la fuente y su tamaño del código visto en el editor, pulsaremos **Ctrl+Alt+S** para abrir el IDE settings y seleccionaremos el tamaño y tipo de letra. Al presionar Apply, se visualñiza en el editor.



La configuración del IDE permite añadir y modificar las barras de herramientas, crear comandos personalizados y atajos de teclado para cada comando. Además, posicionando las ventanas y barras conjuntamente con los atajos de teclado podemos mejorar el rendimiento del IDE y aprovechar al máximo sus funciones.

También es muy efectivo poder establecer configuraciones de depuración y

compilación de proyectos, lo que permite obtener mejores resultados al manejar las interrupciones de las excepciones o manejar la pila de llamadas.

2. Criterios de elección de un IDE

Para poder elegir correctamente el IDE con el que trabajar, necesitamos saber las características que buscamos de él. Veremos diversos criterios que suelen ser los más utilizados a la hora de elegir el IDE, son: el sistema operativo, lenguaje de programación y herramientas-disponibilidad.

2.1. Sistema operativo

Es uno de los criterios más restrictivos para seleccionar el IDE y, más importante aún es para qué SO vamos a desarrollar nuestro software.

Hoy en día existen emuladores o virtualizadores de SO (p.e. VirtualBox), pero no es aconsejable usar estos métodos si nuestro software no va a ser ejecutado mediante máquina virtual. Deberemos desarrollar aplicaciones en el mismo SO en el que van a funcionar cuando estén en la fase de explotación. Esta restricción es inherente al compilador del IDE ya que es quien se encarga de traducir nuestro código fuente a código objeto que es el que se ejecutará en el sistema.

2.2. Lenguaje de programación y framework

Un IDE puede soportar uno o varios lenguajes de programación, por lo que es importante saber en qué lenguaje vamos a codificar nuestra aplicación y qué lenguajes nos ofrecen los distintos IDE disponibles.

Este criterio va de la mano del sistema operativo que usemos, ya que si quisiéramos desarrollar en Visual Basic bajo un SO Linux no sería Visual Studio la opción del IDE sino Gambas.

Lo mismo ocurre con las plataformas de trabajo o frameworks, no solo depende del framework que vayamos a usar, también necesitamos saber bajo qué SO vamos a ejecutarla. Por ejemplo, si fuéramos a desarrollar con Visual Basic en la plataforma .NET bajo Linux, tendríamos que usar Mono Develop en lugar de Visual Studio.

2.3. Herramientas y disponibilidad

Las herramientas son el criterio que completa la elección del IDE pues si nos encontramos con varios IDE que cumplen los criterios de lenguaje y SO, nos decantaremos por aquel que tenga mejores herramientas.

En ocasiones los IDE pueden ser restrictivos por las preferencias del programador, por trabajar de manera colaborativa o por el modo de utilizar los diferentes códigos entre diferentes IDE. Por ejemplo, si nuestros compañeros de trabajo usan el Team Server Foundation (TSF) como sistema de control de versiones, nosotros debemos usarlo también, y el único modo de hacerlo es con Visual Studio; pero si los compañeros están trabajando en Java con un sistema de control de versiones Subversión (SVN), podríamos usar indistintamente Netbeans, Eclipse o InteliJ IDEA, entre otros, ya que SVN está disponible para todos esos IDE.

Dentro de nuestras necesidades y preferencias podría estar la de tener una funcionalidad para crear archivos de ayuda y documentación. En ese caso deberíamos buscar un IDE con esa funcionalidad o buscar una extensión (pluggin) para ese IDE que soporte la funcionalidad deseada.

Se debe invertir una importante cantidad de tiempo para investigar y documentarse a fondo sobre los IDE potenciales que podemos elegir para no elegir incorrectamente o no saber qué funcionalidades tiene el IDE que hemos escogido.

Después de comprobar los criterios anteriores, el aspecto más restrictivo de la disponibilidad es el precio de la aplicación; dependiendo de nuestro presupuesto podremos acceder a diferentes IDE.

Si no disponemos de presupuesto, existen ciertas soluciones gratuitas para poder disponer de un IDE con el que trabajar que cumplirán la mayoría de criterios que necesitamos. En este capítulo instalaremos tres de estos IDE gratuitos: Eclipse, NetBeans e IntelliJ IDEA.

3. Uso básico de un IDE

La funcionalidad básica de los IDE como hemos visto es la de desarrollar software. Pero la tarea del IDE no se queda ahí, ya que permite realizar una serie de operaciones que no se podrían realizar de otro modo. Esta funcionalidad añadida que se ofrece al desarrollador se realiza por medio de herramientas integradas en el propio IDE o añadidas mediante la inclusión de pluggins.

Muchas herramientas que usamos en un IDE, también se encuentran disponibles fuera de ellos, como por ejemplo aplicaciones para crear modelados y diagramas o aplicaciones para automatizar las pruebas unitarias. Pero se debe tener en cuenta que es mucho más sencillo realizar todas estas funciones desde dentro del IDE que mediante aplicaciones externas para cada nueva funcionalidad.

3.1. Edición de programas y generación de ejecutables

Todo IDE debe cubrir la necesidad básica de edición de programas y convertir ese código fuente en código ejecutable. Sin un IDE también es posible realizar por un lado la edición del código con un editor de texto, compilar el código fuente con un compilador y usar un enlazador para combinar ese código objeto con las librerías, obteniendo como resultado el programa ejecutable. Un IDE realiza esas tareas de forma conjunta y compacta.

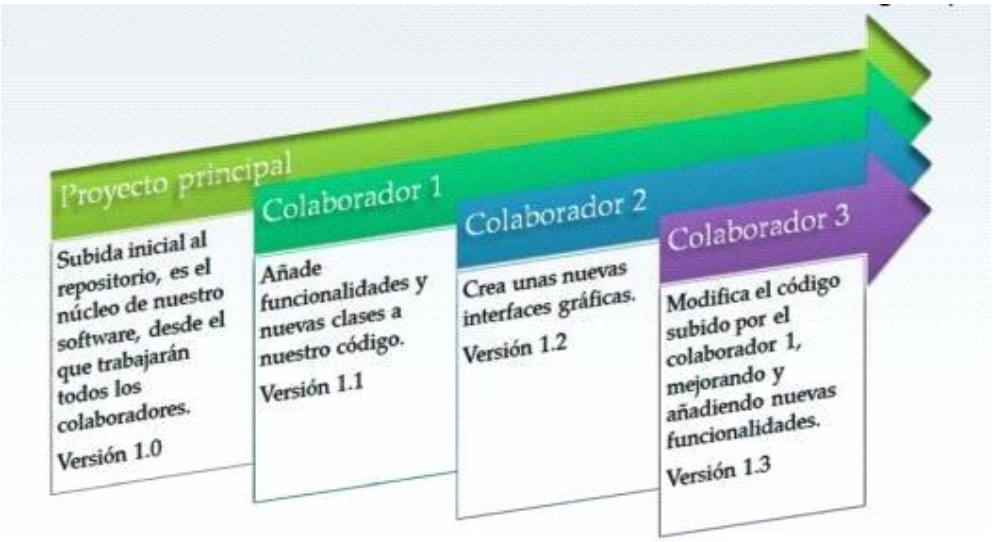
Los IDE, además, suelen ofrecer una funcionalidad añadida, ya que permiten ejecutar de manera virtual el programa que se está codificando en cualquier momento, de ese modo permiten comprobar la funcionalidad del programa sin tener que crear una publicación o despliegue de la aplicación para probar cualquier cambio añadido.

3.2. Desarrollo colaborativo

El desarrollo colaborativo hace referencia al desarrollo de un software de manera descentralizada y distribuida, donde los desarrolladores no necesitan conocerse ni hablar el mismo idioma, pero trabajan en el mismo proyecto. Para poder llevar un control del código realizado desde tantas y diversas fuentes, se utilizan los controles de versiones.

Los programas de controles de versiones son aplicaciones que constan de servidor y cliente, donde en la parte del servidor se crean repositorios para que los clientes se puedan descargar y subir código. Son herramientas asíncronas que permiten controlar y gestionar las fuentes y versiones del código del repositorio.

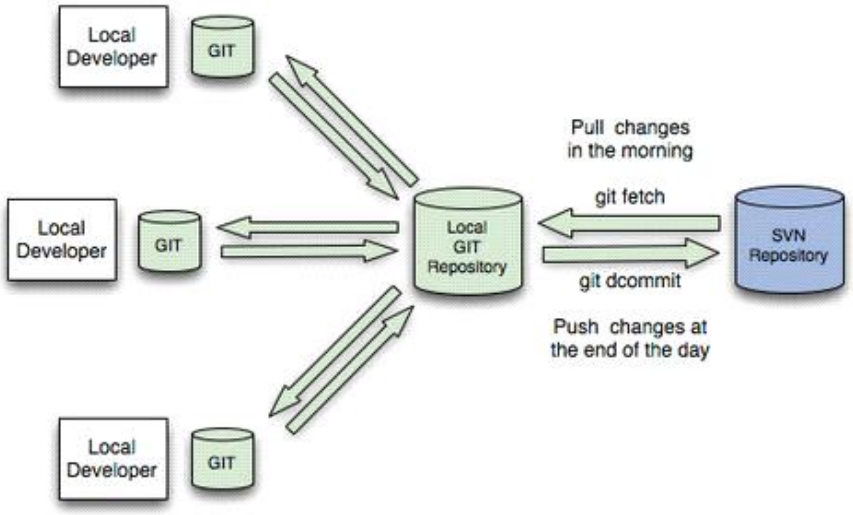
Veamos con un diagrama el funcionamiento de un control de versiones:



El proyecto parte del mismo núcleo y se va modificando a lo largo del tiempo, los desarrolladores van trabajando sobre el código realizado por los anteriores colaboradores. Además. podemos elegir sobre qué versión trabajar descargándola del repositorio.

Todas estas operaciones realizadas desde el propio IDE permiten gestionar el control de versiones de forma rápida, pudiendo elegir qué archivos actualizar en servidor o cliente, omitir cambios para no pisar nuestro trabajo con el de otros, y viceversa, y otras operaciones de la misma índole.

La sincronización de un IDE con el repositorio del proyecto permite saber qué archivos han cambiado y tener un control de versiones más allá del servidor, viendo qué archivos se han modificado, borrado o añadido desde la última actualización al repositorio.



4. Instalación de un entorno de desarrollo

4.1. Descarga del JRE

1.- Entramos en la página principal de Java en español en la sección descargas: https://www.java.com/es/download/ie_manual.jsp

2.- Hacer clic en el botón rojo "**Aceptar e iniciar descarga gratuita**".

3.- Se inicia la descarga inmediatamente y nos baja el archivo **JavaSetup8u301.exe** a la carpeta de descargas.

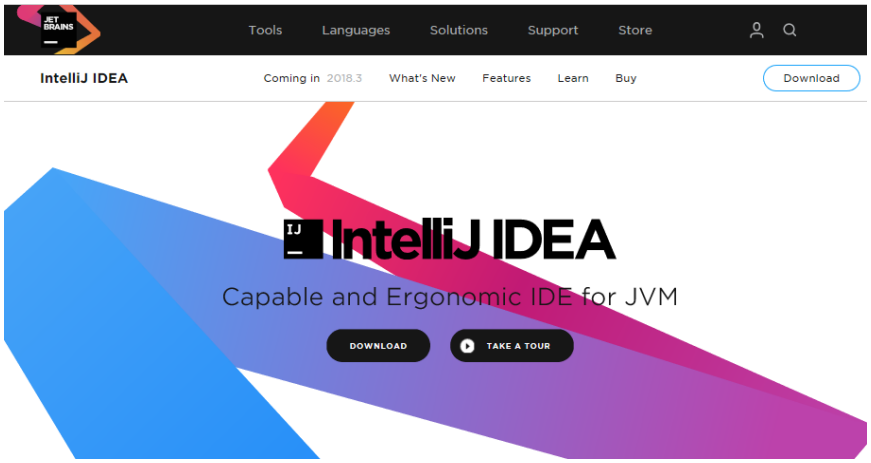
4.2. Descarga del IDE IntelliJ IDEA

IntelliJ IDEA es un **IDE** (entorno de desarrollo integrado) creado principalmente para la programación de Java. Es una herramienta que los programadores pueden usar para escribir mejor código de manera más eficiente. Al automatizar gran parte del código repetitivo, aplicando y aplicando fácilmente los estándares de codificación, integrando la finalización del código, la depuración y la implementación, además de muchas más características, ayuda a los programadores a escribir mejores códigos en menos tiempo.

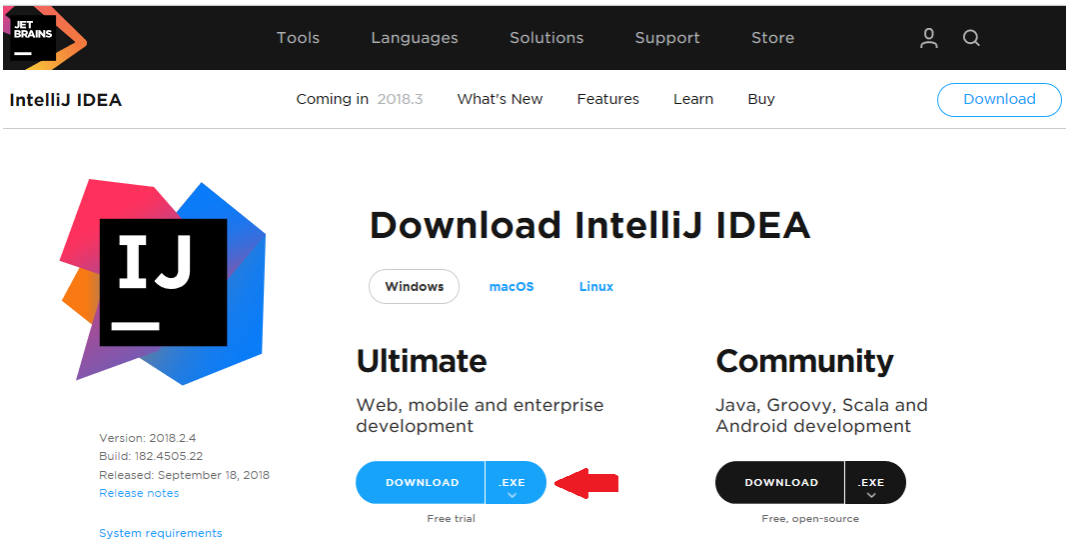
En este IDE tenemos todas las funciones para que los programadores de Java creen fácilmente nuevos proyectos, los desarrollen, prueben, depuren y finalmente los implementen. Los programadores de Java saben que escribir código puede resultar engorroso al seguir convenciones de nomenclatura estándar, por lo que características como la finalización del código ahorran una gran cantidad de ayuda al escribir y marcos específicos, lo que significa que no tienes que buscar manualmente las clases y los métodos de cada biblioteca.

Las herramientas de depuración ayudan a rastrear la fuente de errores complejos y la integración total con software de control de versiones significa que los desarrolladores tienen virtualmente todas las herramientas que necesitan

Lo podemos encontrar en: https://www.jetbrains.com/idea



Si pulsamos sobre DOWNLOAD nos lleva a la página:



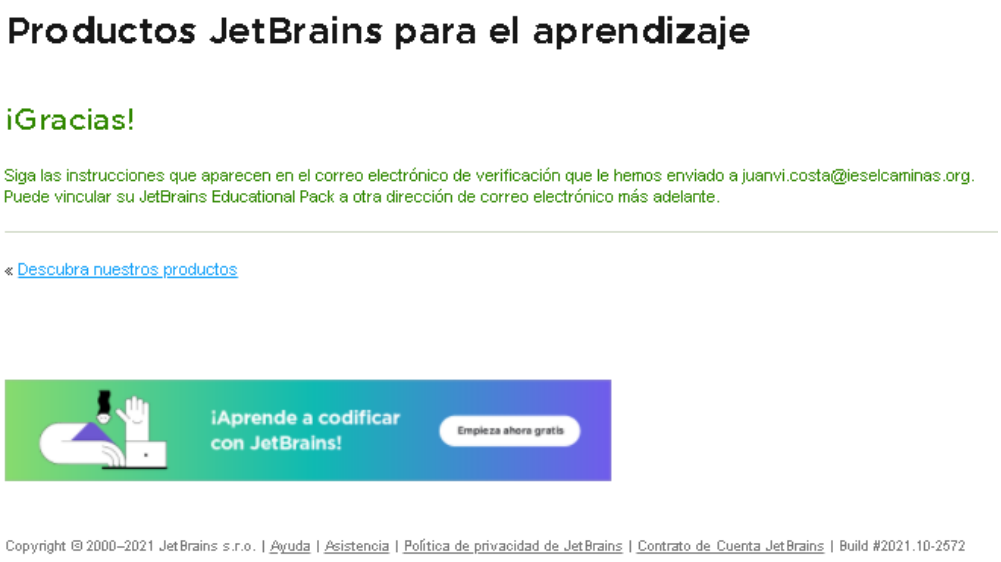
Pulsando en **DOWNLOAD** de la versión **Community** se nos descarga el IDE o podemos instalarlo directamente.

Tenemos la posibilidad de descargar la versión **Ultimate** con una licencia de estudiante. Para ello debéis entrar en el siguiente enlace:

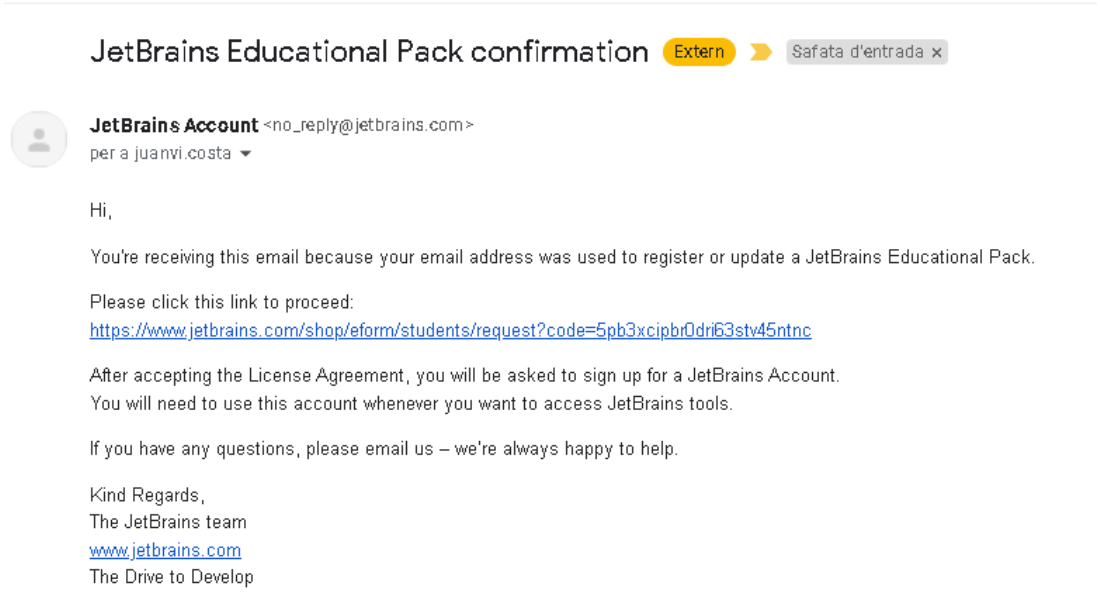
https://www.jetbrains.com/es-es/community/education/#students

Una vez allí debemos buscar el botón que permite solicitar licencias individuales para estudiantes y docentes.

Cuando presionemos en **Solicitar ahora**, se nos abrirá un formulario para rellenar con nuestros datos. Debéis poner el correo de ieselcaminas.org sino no os dejará. Una vez rellenados los datos y enviado el formulario os responderá asi:



Si todo ha ido bien, recibiréis en vuestro correo un mensaje de JetBrains para que finalicéis la suscripción.



JetBrains Educational Pack confirmation Extern Safata d'entrada ×

JetBrains Account <no_reply@jetbrains.com>
per a juanvi.costa

Hi,

You're receiving this email because your email address was used to register or update a JetBrains Educational Pack.

Please click this link to proceed:
https://www.jetbrains.com/shop/eform/students/request?code=5pb3xcipbr0dri63stv45ntnc

After accepting the License Agreement, you will be asked to sign up for a JetBrains Account.
You will need to use this account whenever you want to access JetBrains tools.

If you have any questions, please email us – we're always happy to help.

Kind Regards,
The JetBrains team
www.jetbrains.com
The Drive to Develop

Solo tenéis que hacer click en el enlace y os llevará a otra página que es el TOOLBOX SUBSCRIPTION AGREEMENT FOR STUDENTS AND TEACHERS. Bajáis hasta abajo hasta que se active el botón **I Accept** y hacéis clic en él siguiendo las instrucciones que os propone.

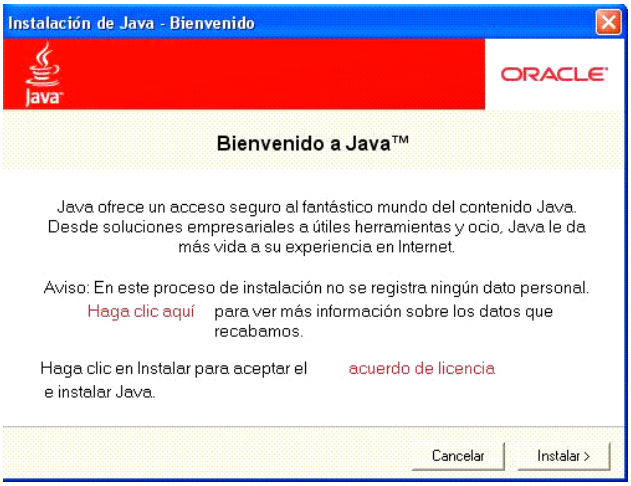
Este proceso os permitirá descargar el archivo ideaIE-2021.2.2.exe.

4.3. Instalación del JRE

La mayoría no hará falta que realicéis este paso, pues si tenéis Internet, es muy probable que ya tengáis instalado Java y actualizado, ya que periódicamente nos aparece en la barra de tareas (normalmente debajo de la pantalla) el icono de actualización de Java.

Para quien no lo tenga instalado, deberá seguir los siguientes pasos:

1.- Hacemos doble clic en el JDK descargado de la instalación de Java. Dependiendo del navegador en que se haya descargado, tendrá un nombre, pero la instalación que hace es la misma. Nos aparecerá una ventana como esta:

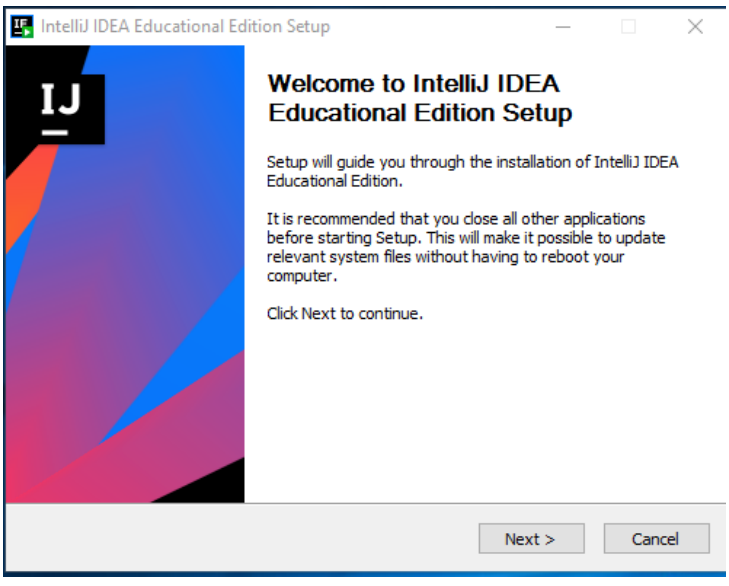


Y cuando le demos al botón de aceptar, se iniciará la instalación de Java

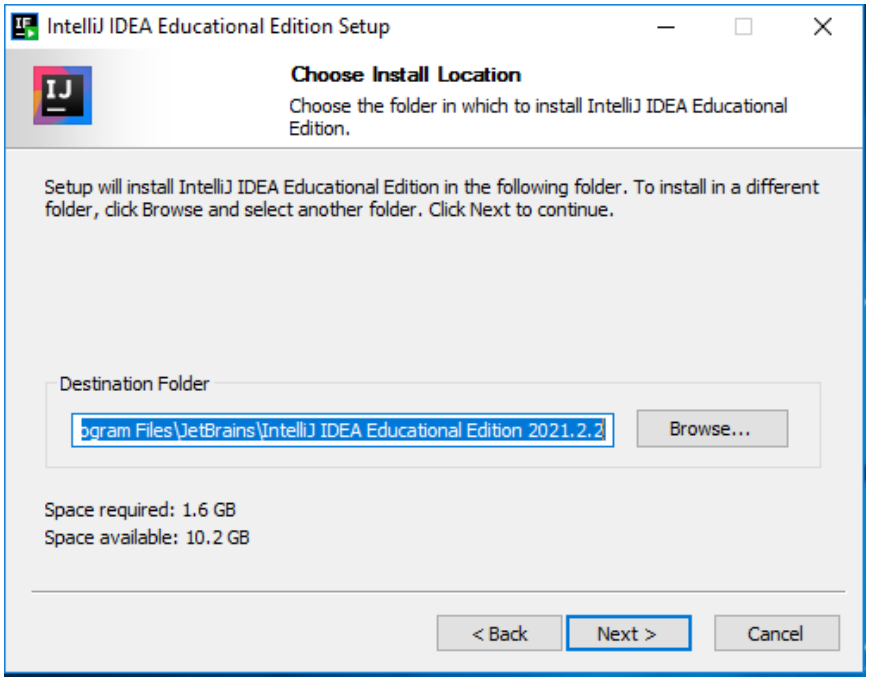
que nos indica el proceso de instalación de Java. Al finalizar esta instalación nos aparece una ventana que nos informa de su correcta instalación.

4.4. Instalación del IDE IntelliJ IDEA

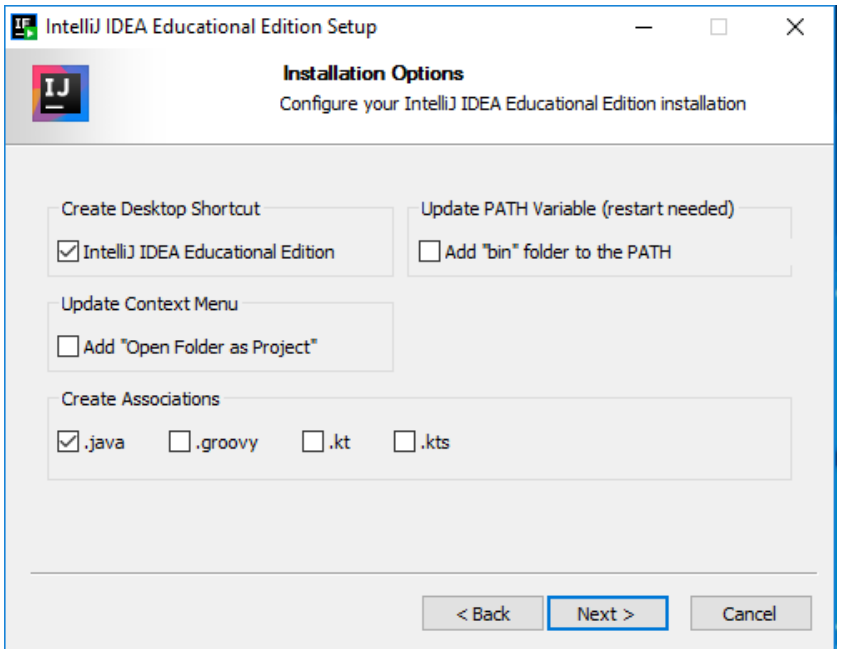
Para comenzar la instalación ejecutaremos el archivo descargado en el paso 4.2 ideaIC-2021.2.2.exe. Esta es una licencia gratuita de la versión Ultimate que es más completa que la Community. Al ejecutar el archivo y después de preguntarnos si queremos que IntelliJ haga cambios en nuestro dispositivo, comenzara la instalación con la siguiente ventana:



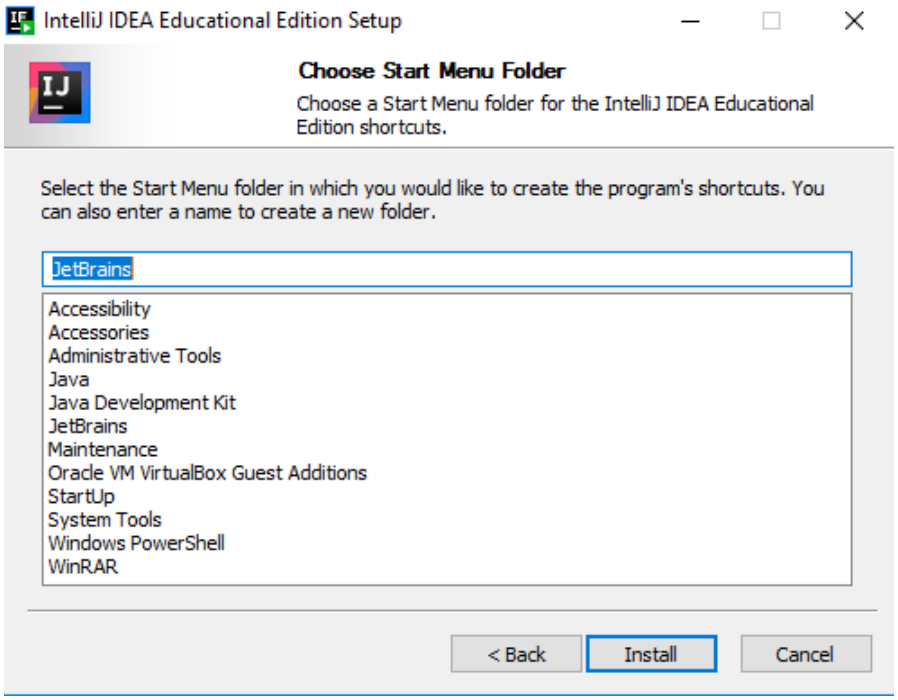
Al pulsar en **Next** nos pedirá la carpeta de instalación:



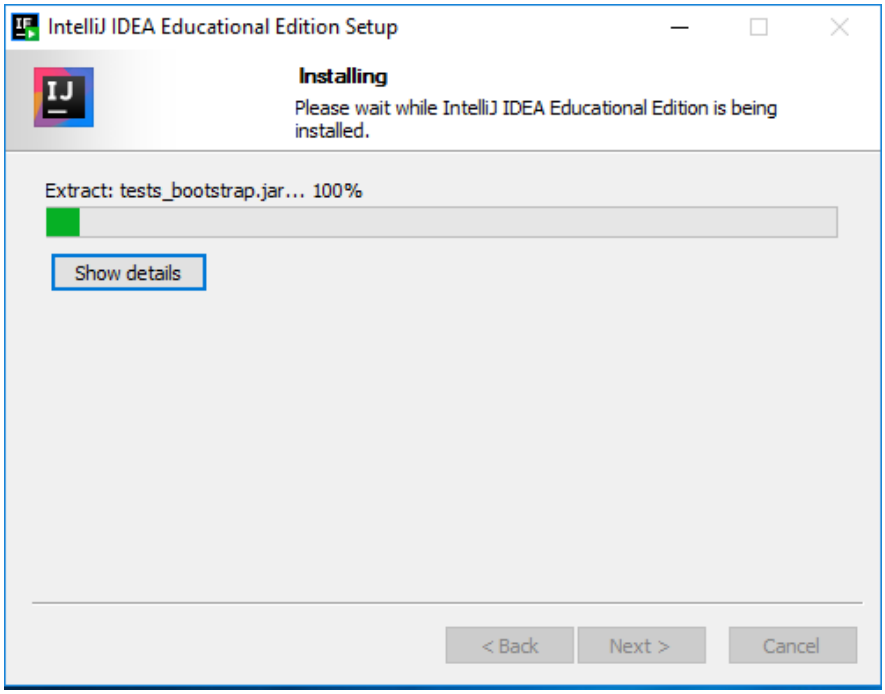
A continuación, nos preguntará si queremos crear un acceso directo al programa, qué tipo de aplicaciones vamos a crear (seleccionaremos las .java) y si queremos descargar el JRE de Java (como seguramente ya lo tendremos de las otras instalaciones, no seleccionamos la opción correspondiente).



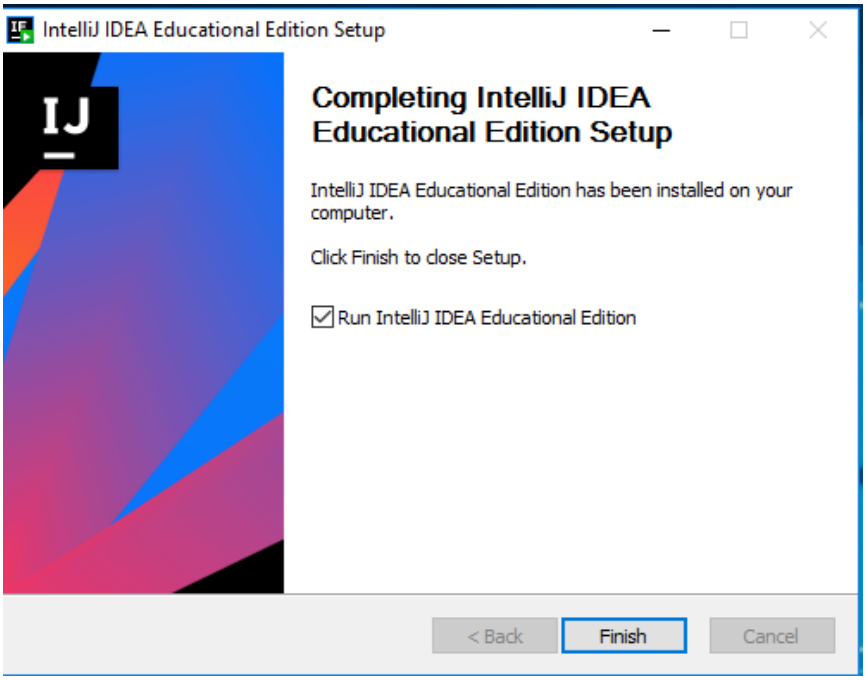
Seguidamente nos pregunta la carpeta en la que se instalará el IDE dentro de Archivos de programa:



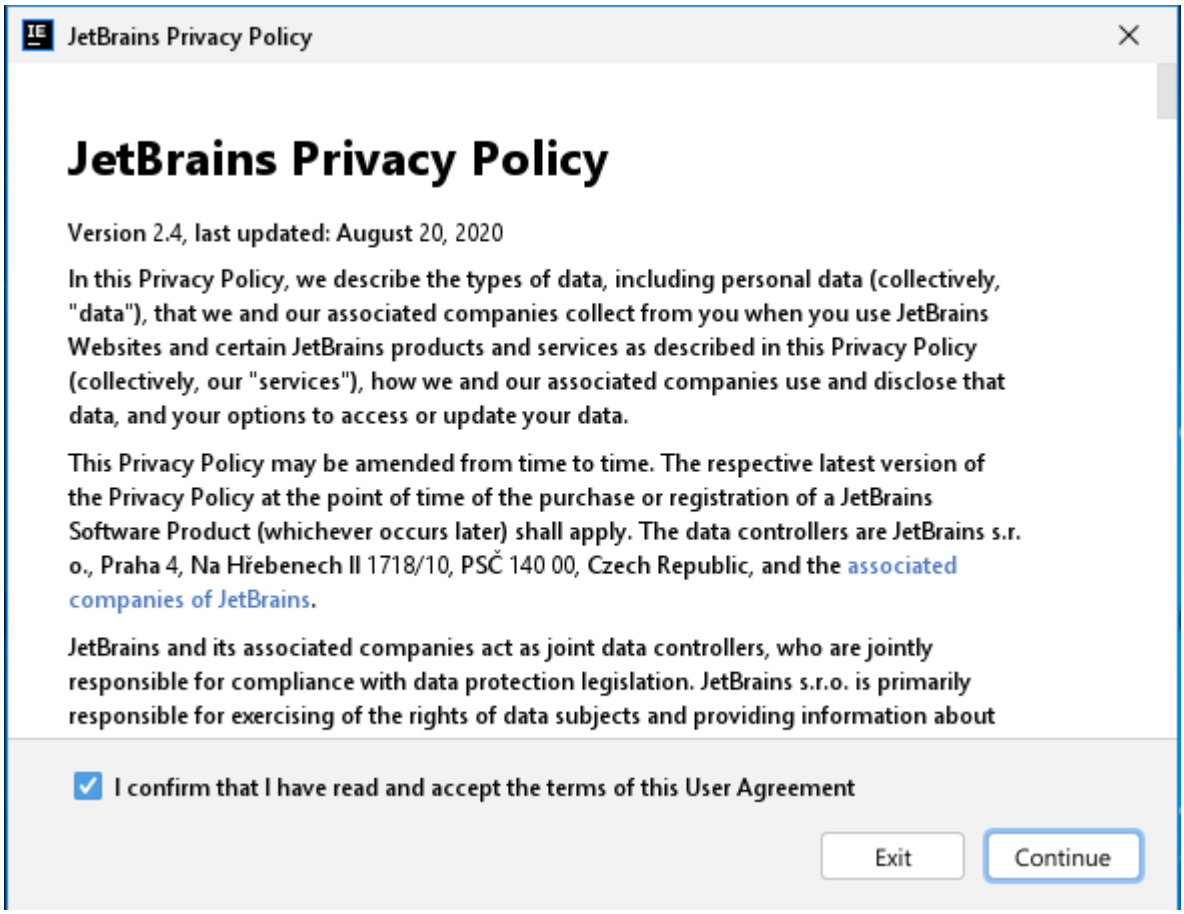
Una vez hechas estas selecciones pulsamos sobre **Install** y comenzará la instalación.



Una vez acabada la instalación se nos muestra la siguiente ventana, que nos informa que todo ha ido correctamente.

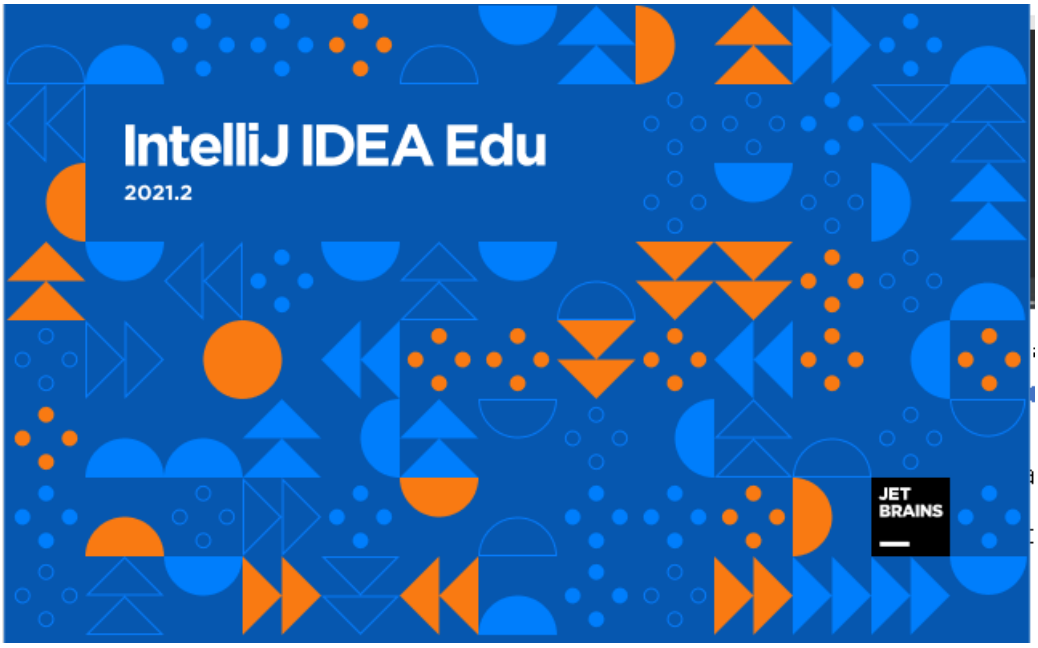


Activamos la casilla Run IntelliJ IDEA y pulsamos en **Finish**..

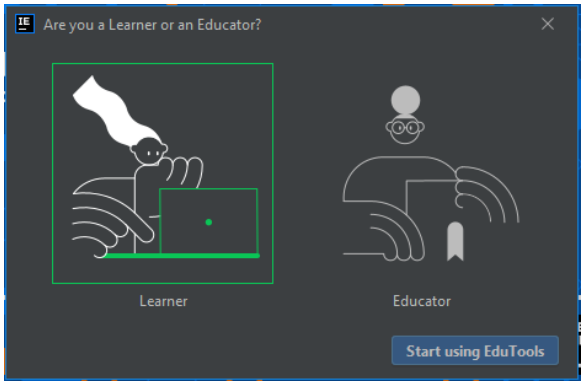


La primera vez que lo ejecutamos hemos de aceptar la política de privacidad pulsando en **Continue**

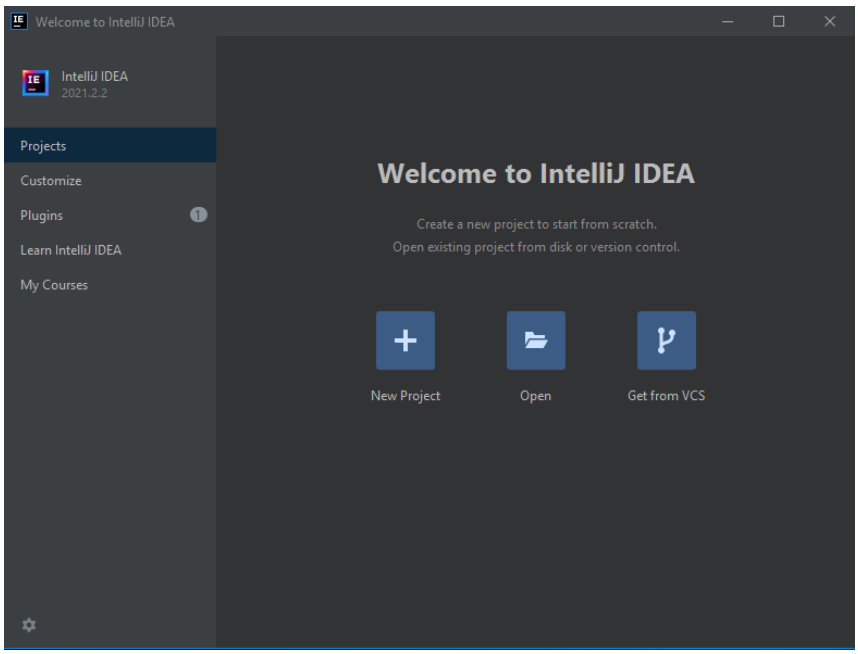
El programa se inicia mostrando la siguiente imagen:



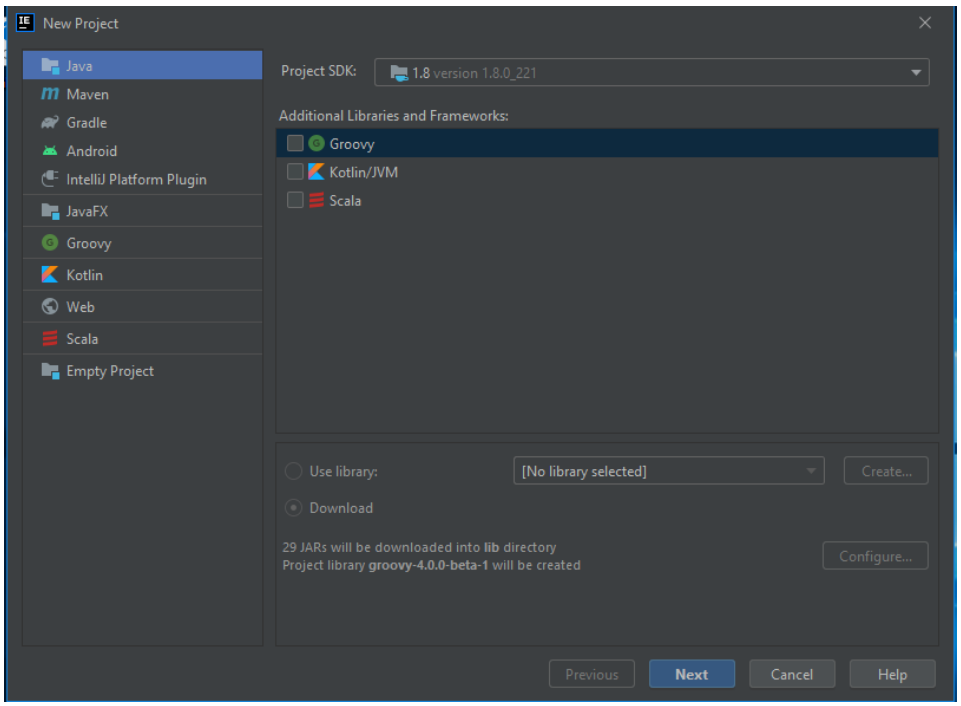
Luego nos pregunta si somos estudiante o docente:



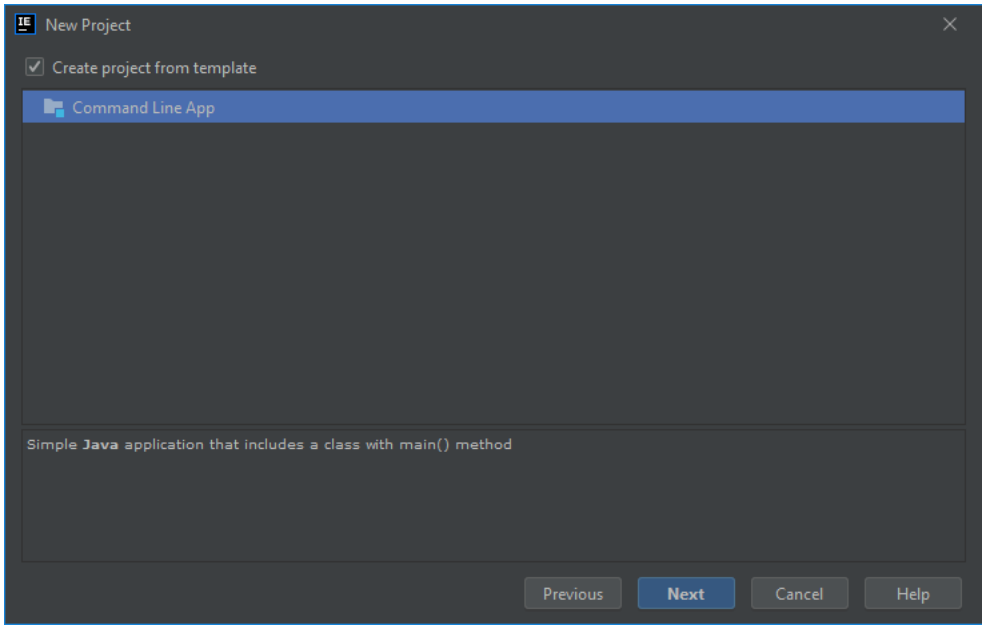
A continuación, ya podemos empezar a utilizarlo seleccionando un nuevo proyecto o algún proyecto existente:



Si seleccionamos un proyecto nuevo, nos aparece la siguiente ventana para poder añadir librerías adicionales si hacen falta.

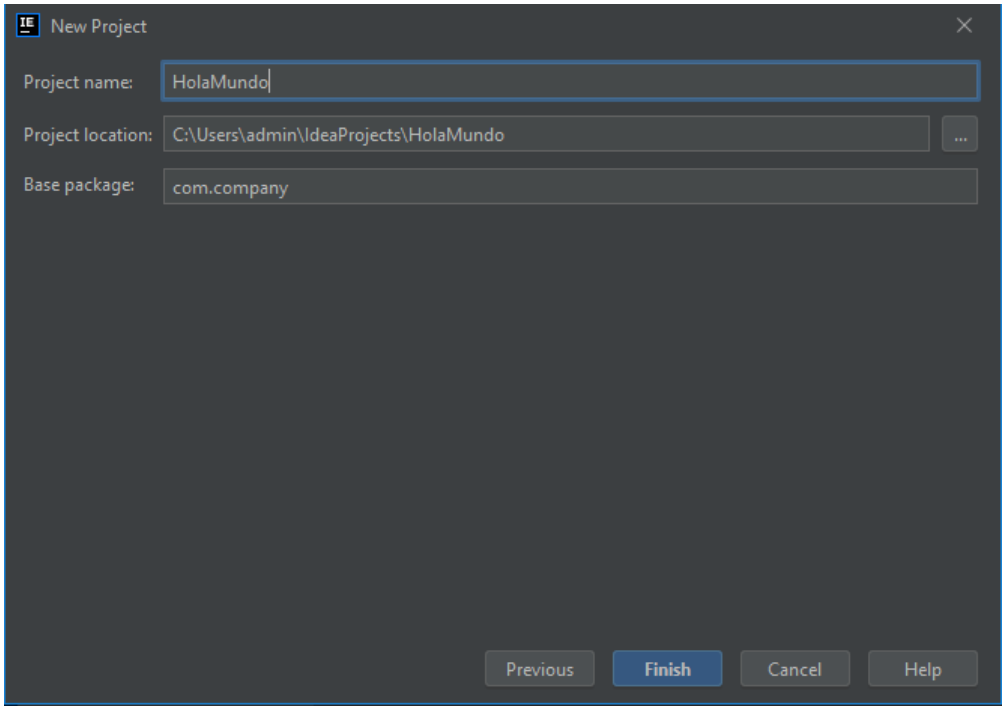


Al pulsar en **Next** nos permite crear un nuevo proyecto usando una plantilla si activamos la casilla “Create project from template”. El primer proyecto lo crearemos así.

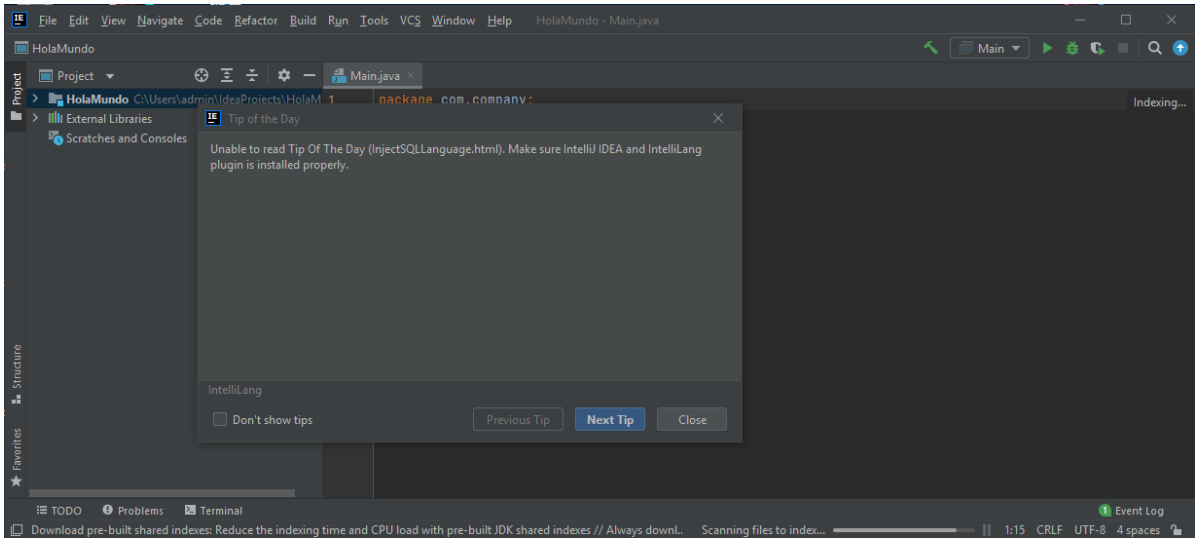


Al pulsar en **Next** nos pedirá el nombre del nuevo proyecto le pondremos **HolaMundo** y nos mostrará su ubicación en nuestro ordenador. En mi caso estará en la carpeta:

C:\Users\administrador\IdeaProjects\HolaMundo.



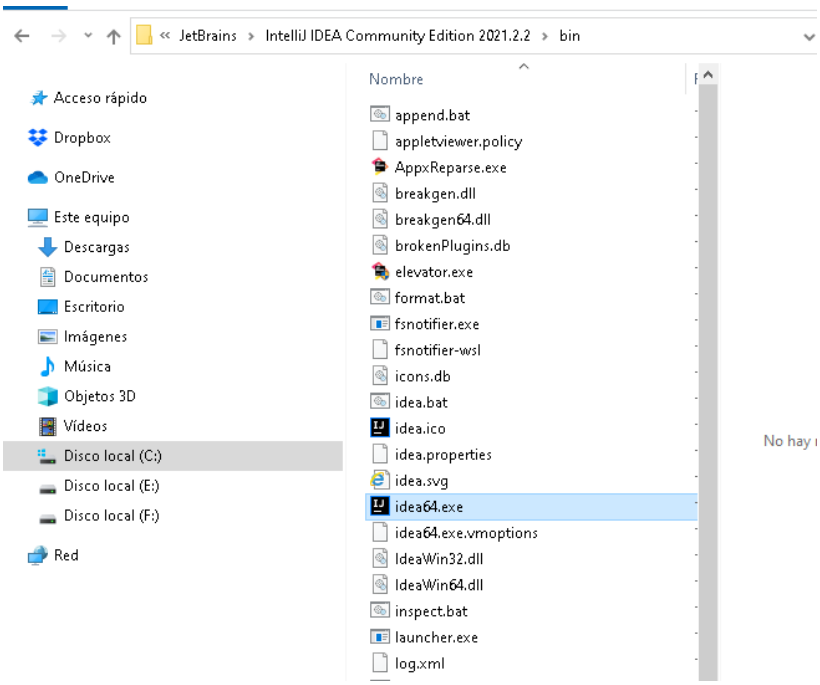
Cuando presionemos el botón **Finish**, accederemos a la ventana principal de IntelliJ IDEA que tiene el siguiente aspecto:



Con esto ya tenemos instalado IntelliJ IDEA en nuestro ordenador.

5. Uso básico de un entorno de desarrollo

1.- Ahora solo falta comprobar que se ha instalado correctamente. Para ello vamos a la carpeta de Archivos de programa y comprobamos que esta la carpeta de JetBrains\IntelliJ IDEA Community Edition 2021.2.2\bin. Entramos en ella y veremos lo siguiente:



2.- Si no lo hemos hecho en la instalación, creamos en el escritorio un acceso directo a idea64.exe o ejecutamos IntelliJ directamente haciendo doble clic en **idea64.exe**. Nos aparecerá la ventana siguiente:

Si se ve el logo de IntelliJ IDEA sin la aparición de ningún mensaje de error, significa que la Máquina virtual de Java se ha instalado correctamente.

3.- La primera acción que he realizado en el IDE ha sido la de configurar un fondo blanco para el editor ya que para realizar las capturas de pantalla queda mejor. Para ello Desde la opción del menú **File→Settings→Appearance** elegimos como tema el de **Windows 10 Ligth**.

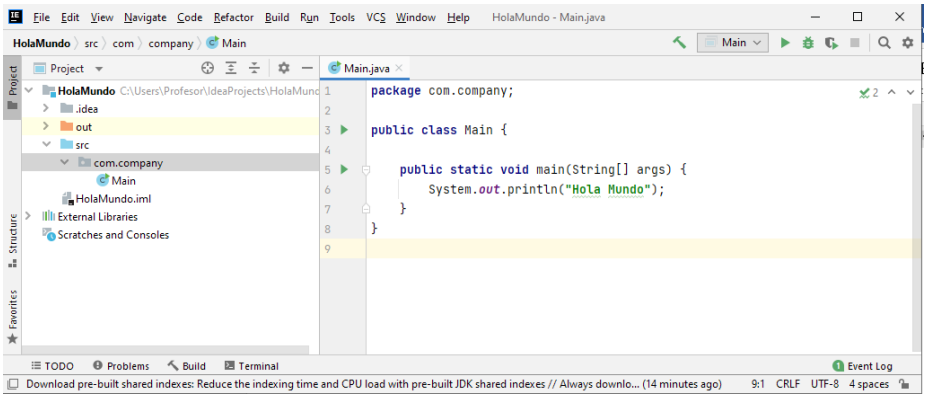
4.- Para crear un proyecto nuevo, seleccionamos del menú **File→New→Project**, añadimos librerías si las necesitamos, usamos la plantilla si queremos, le damos nombre al proyecto “hola mundo” y obtendremos la siguiente imagen:

6. Edición de programas

1.- Ejecuta IntelliJ como se ha indicado en el punto anterior.

2.- Vamos a abrir el proyecto creado "**HolaMundo.java**". Este proyecto se guardará en la carpeta **IdeaProjects** del usuario. Un proyecto es la unidad básica que utiliza IntelliJ para poder trabajar, y es el lugar donde guarda tanto los archivos del código fuente (.java) como archivos necesarios para que ese código fuente funcione.

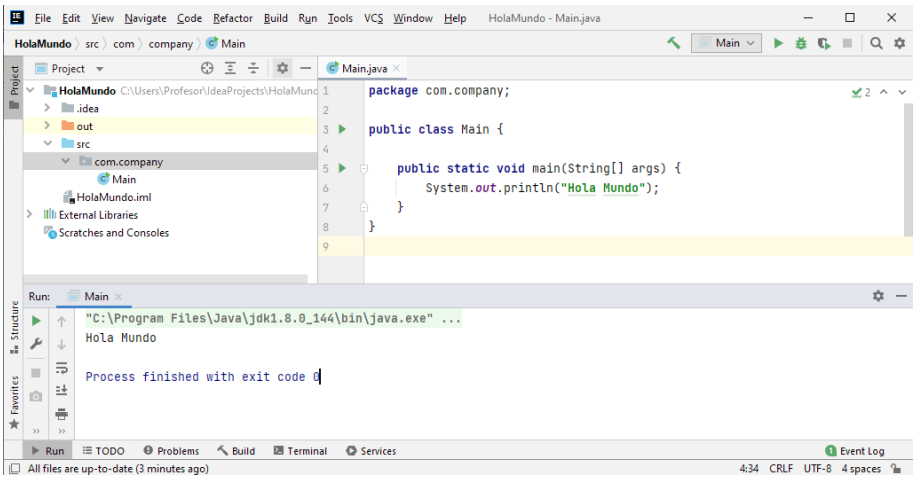
3.- Una vez abierto el proyecto, vamos a editar la clase creada por defecto dentro de él que se llama **Main**. Más adelante veremos que se pueden crear nuevas clases en el mismo proyecto desde la opción del menú **File→New→Java class.** Modificamos el método main poniendo al final una instrucción que muestra por consola la frase Hola Mundo.



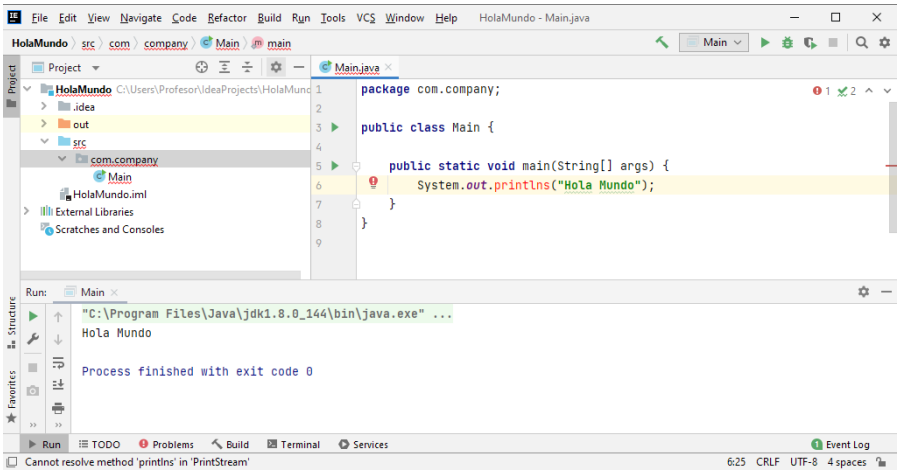
4.- Por último, vamos a guardar los cambios en la clase Main mediante la opción del menú **File/Save all** o con la combinación de teclas **Ctrl + S**.

7. Herramientas básicas en la edición de programas

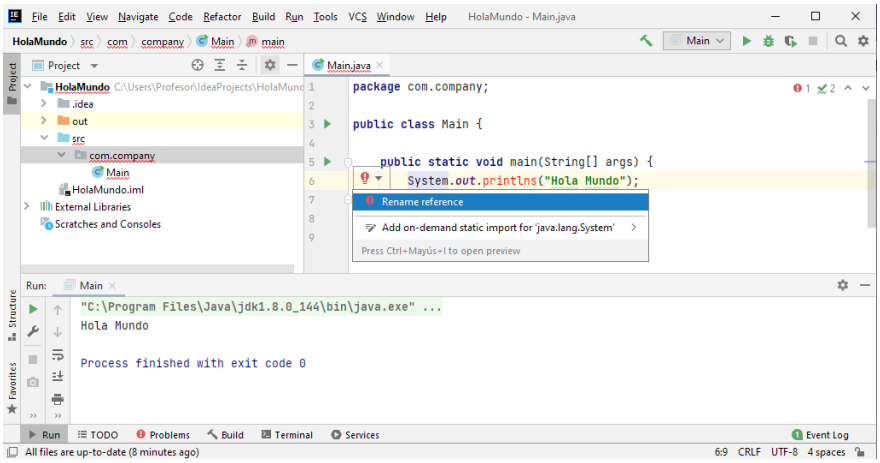
1.- **Compilación**: Esta herramienta permite detectar los errores del código y mostrarlos para que puedan ser depurados. Si un programa tiene errores no podrá ser ejecutado hasta que se corrijan. En IntelliJ se compila mediante la opción del menú **Run/Run main** o la combinación de teclas **Mayusc + F10**; compilamos y ejecutamos al mismo tiempo un programa. El resultado de la compilación se muestra en la ventana **Run**.



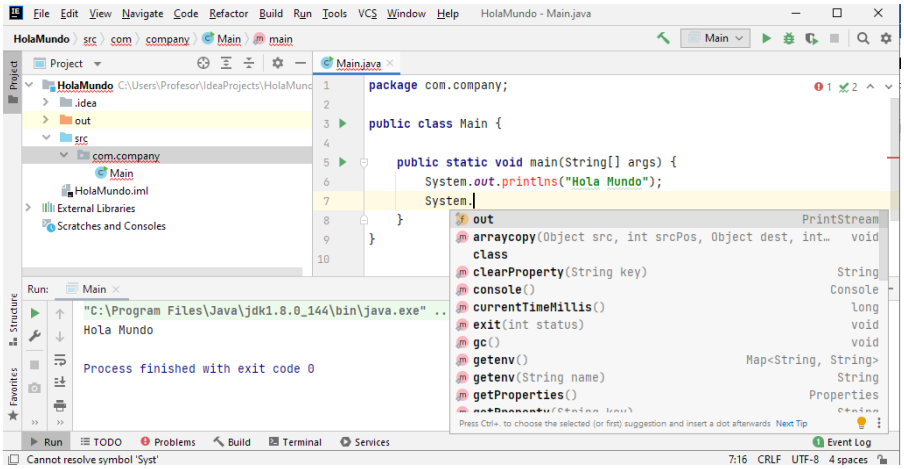
2.- **Detección de errores**: IntelliJ detecta los errores en tiempo real, y los marca en rojo, con una bombilla roja en la parte izquierda de la línea de código donde está el error y una bombilla roja en el margen izquierdo del código.



3.- **Corrección de errores**: Si hacemos clic en la bombilla roja te lleva a alguna posible solución de este error (o presionando **Ctrl +Mayucs + I**). Solo debes seleccionar una de esas opciones para modificar el código con la solución elegida.



4.- **Autocompletar**: A medida que escribimos el código, IntelliJ intenta adivinar lo que vas a escribir y muestra una lista con las opciones entre las que elegir. Esto es muy cómodo y rápido por ejemplo al escribir el nombre de una clase y el punto, nos muestra un listado que de palabras que pueden seguir a la clase. Por ejemplo, si introducimos la clase **System.** nos aparece la lista siguiente:



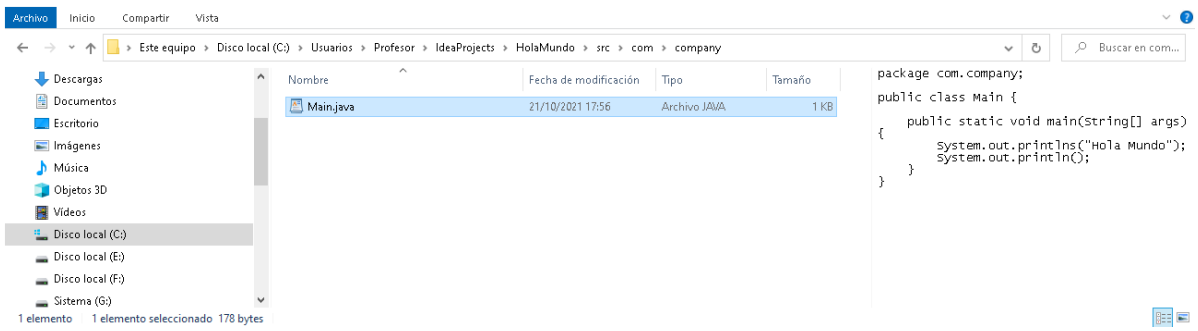
Elegimos out y al poner el punto nos vuelve a aparecer otra lista para volver a elegir la palabra println. A continuación, solo hace falta completar parámetro de entrada de tipo String (el texto que se va a imprimir por pantalla entre comillas).

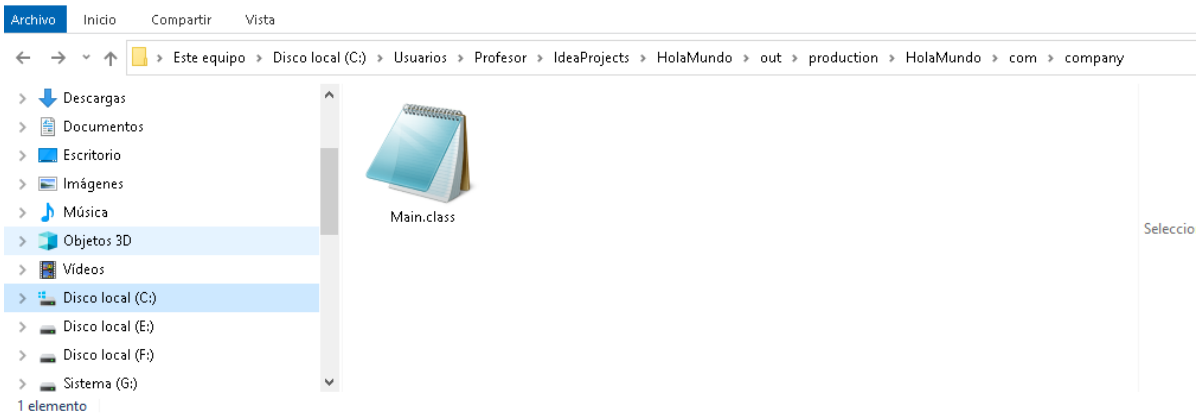
8. Generación de programas ejecutables

A diferencia de otros lenguajes de programación de alto nivel que generan un código ejecutable después de la compilación y el enlazado, en Java no se generan directamente los archivos .exe que conocemos de otros IDEs.

En Java se generan unos archivos en un lenguaje intermedio o bytecode que luego interpretará la Máquina Virtual de Java. Estos archivos tienen extensión .class y se encuentran en la carpeta **IdeaProjects\HolaMundo** del proyecto

Si nuestro proyecto Java se llamaba HolaMundo, se crea una carpeta HolaMundo que a su vez tiene otras dos carpetas: **src** donde se encuentra el código fuente del programa “HolaMundo.java” y otra **out\production\HolaMundo \com\company** donde se encuentra el bytecode “Main.class” que interpreta el JRE de Java.





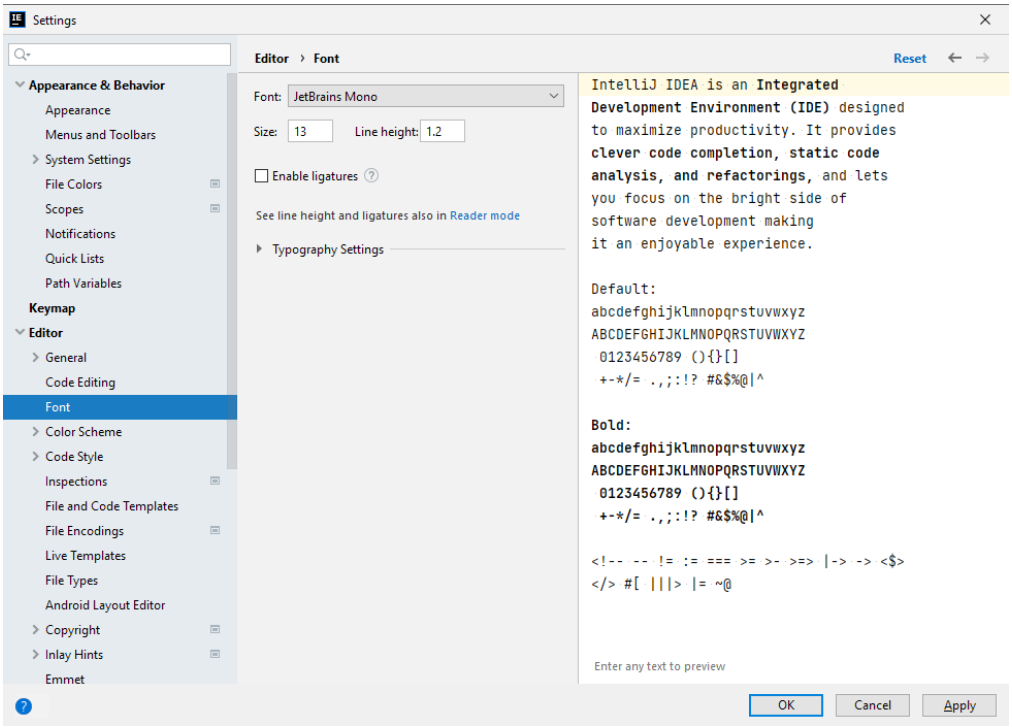
9. Personalización y configuración del IDE

En IntelliJ como en la mayoría de IDE se puede personalizar y añadir controles para personalizar el entorno. Esta personalización facilita al programador su tarea con acciones como redimensionar, reposicionar y modificar los elementos del IDE.

La personalización del IDE no sólo consta de la configuración gráfica, sino que además permite establecer valores por defecto y configuraciones específicas del comportamiento del IDE para ajustarlo a nuestras necesidades.

OPCIONES DEL ENTORNO

Si deseamos modificar las opciones del entorno, debemos ir al menú **File → Settings** y seleccionar el apartado que deseemos configurar a nuestra medida.



Por ejemplo, desde el apartado **Editor** podemos configurar muchísimas opciones del editor para que nos resulte más agradable la edición de programas.

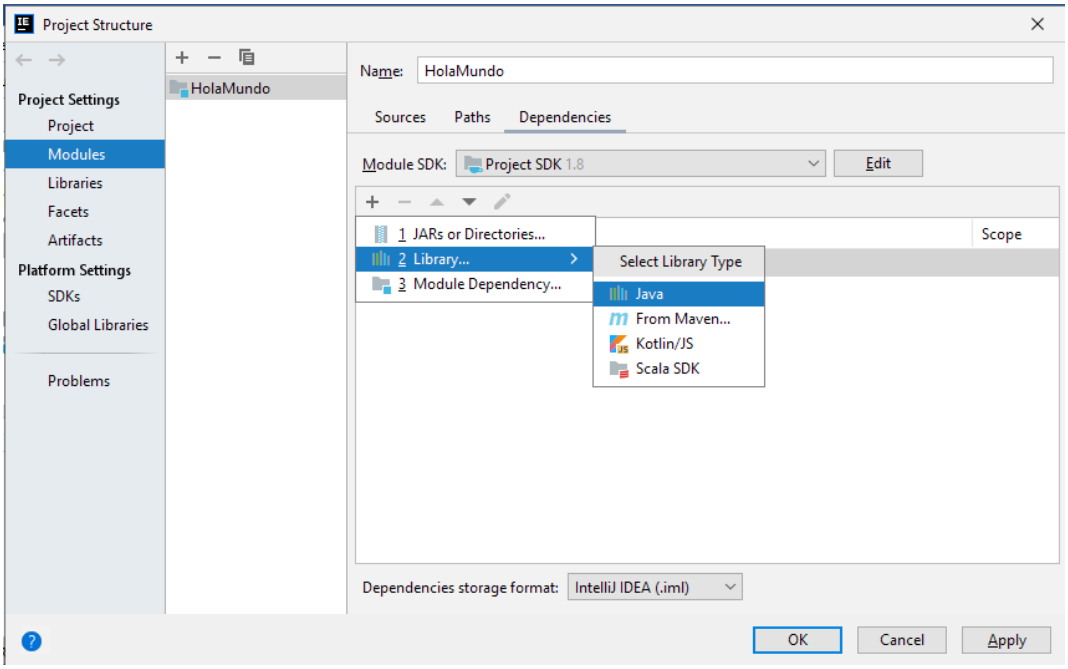
OPCIONES DEL PROYECTO

Para cambiar las opciones particulares de un proyecto, debemos abrir la configuración del proyecto, esto se consigue con la siguiente combinación de teclas: **Ctrl + Alt + Shift + S**. Por ejemplo, para añadir librerías externas a nuestro proyecto (por ejemplo javadesigner.jar), deberíamos realizar los siguientes pasos:

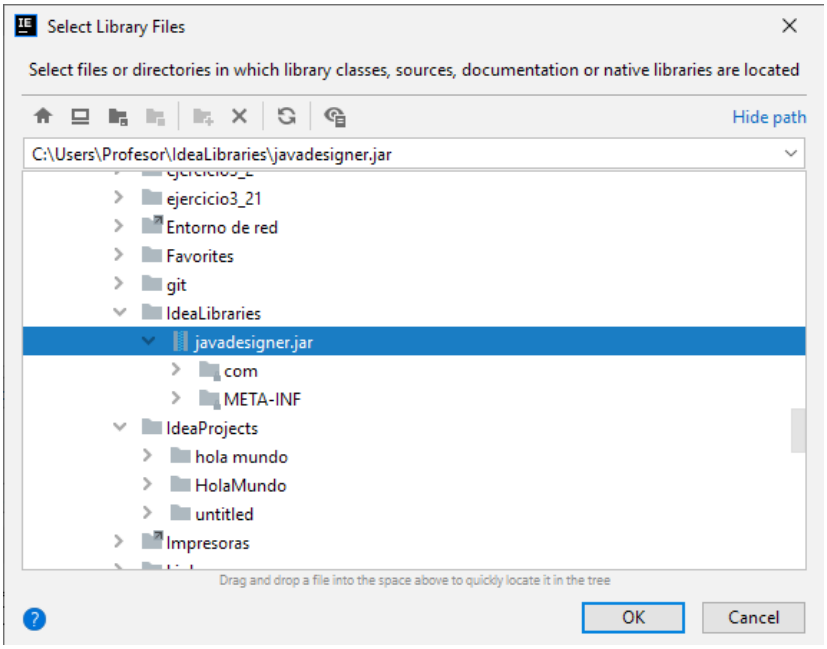
- Abrir Configuración del proyecto (**Ctrl Alt Shift S**)
- Bajo el panel de Configuración del Proyecto a la izquierda, elegir **Módules**
- En el panel más grande de la derecha, elegir la pestaña **Dependencies**

- Presionar el botón **Add...** en el extremo derecho de la pantalla hay un botón con el signo **+.**

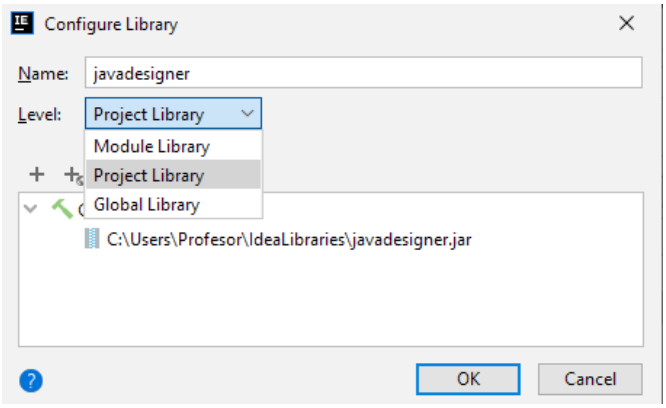
- En el menú desplegable de Agregar opciones, elija "**Library…**". Elegimos de tipo "**Java** ".



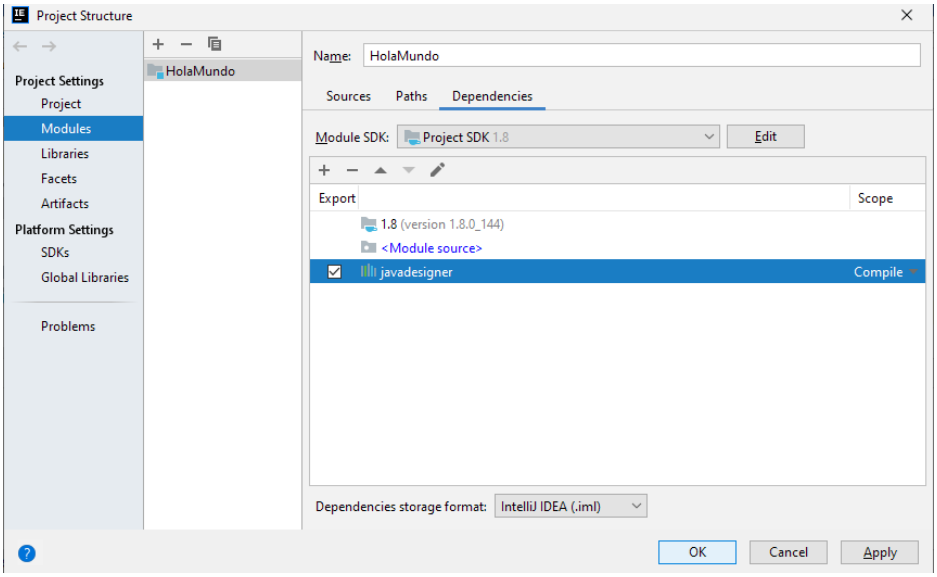
- A continuación, se abrirá una ventana de navegación para buscar la biblioteca a incorporar.



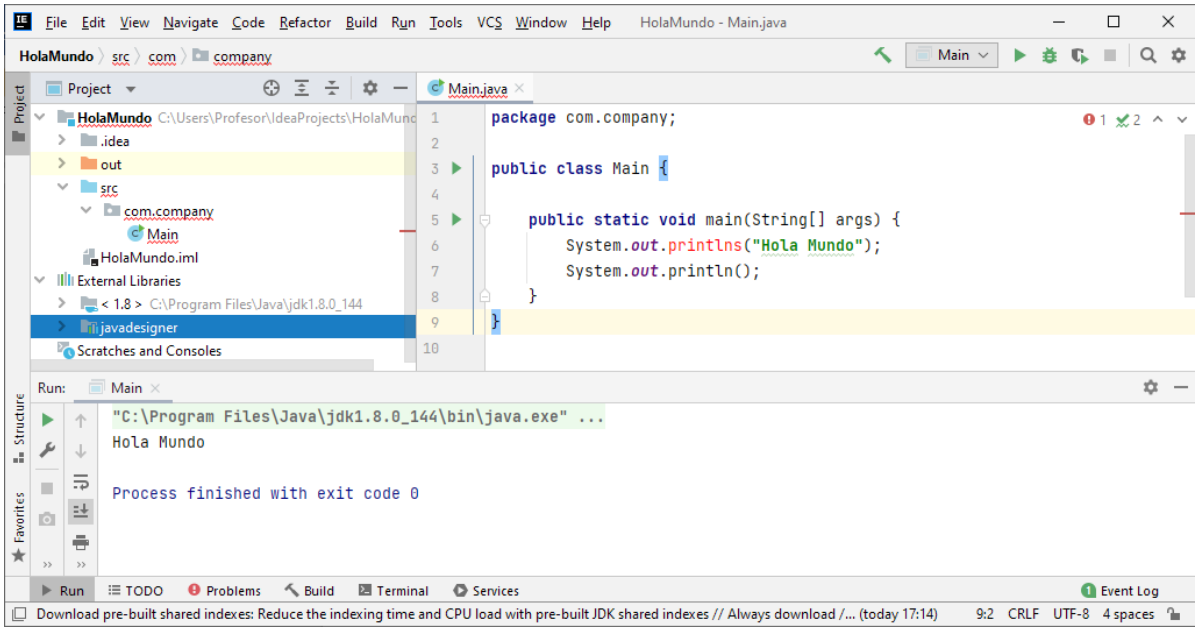
- Elegimos la biblioteca **javadesigner.jar** de la carpeta adecuada.



- Podemos configurar la biblioteca, elegimos **Proyect Library** y presionamos **OK**.



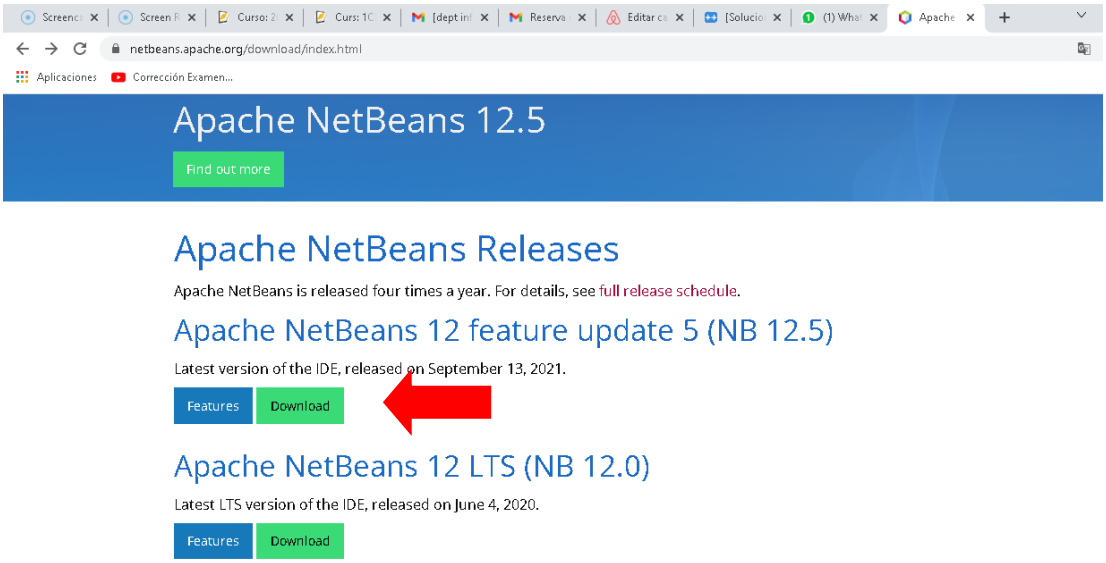
- La biblioteca se añade a nuestro proyecto y podemos verlo en el IDE a través de **External Libraries** del proyecto HolaMundo.



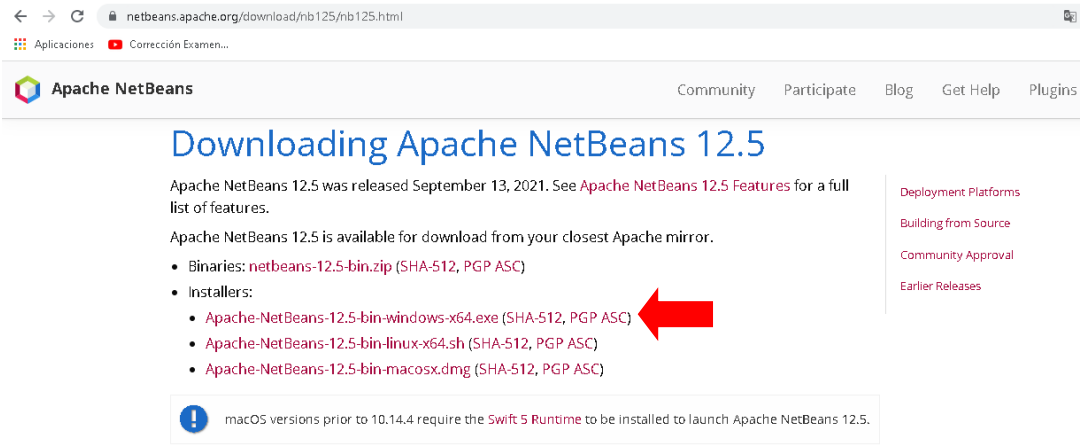
10. Instalación del IDE Netbeans

NetBeans es un **entorno de desarrollo integrado libre**, construido principalmente para el **lenguaje de programación Java**. Existe además un número importante de módulos para extenderlo. NetBeans IDE es un producto libre y gratuito sin restricciones de uso. La página web es:

https://netbeans.apache.org/download/index.html

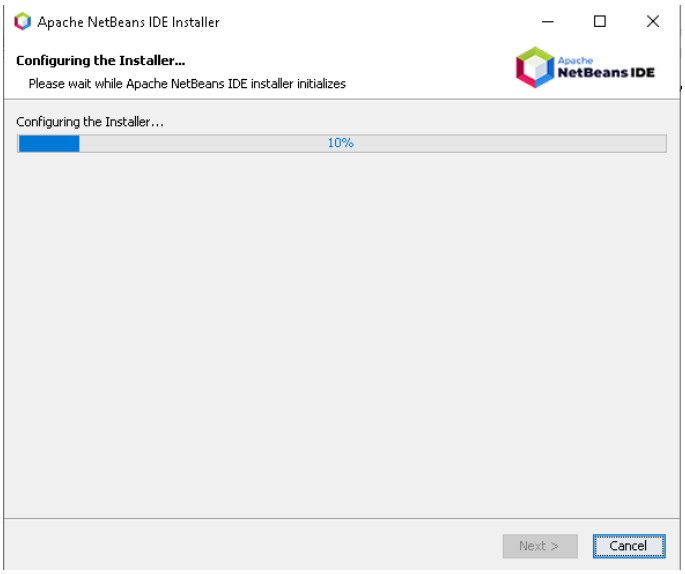


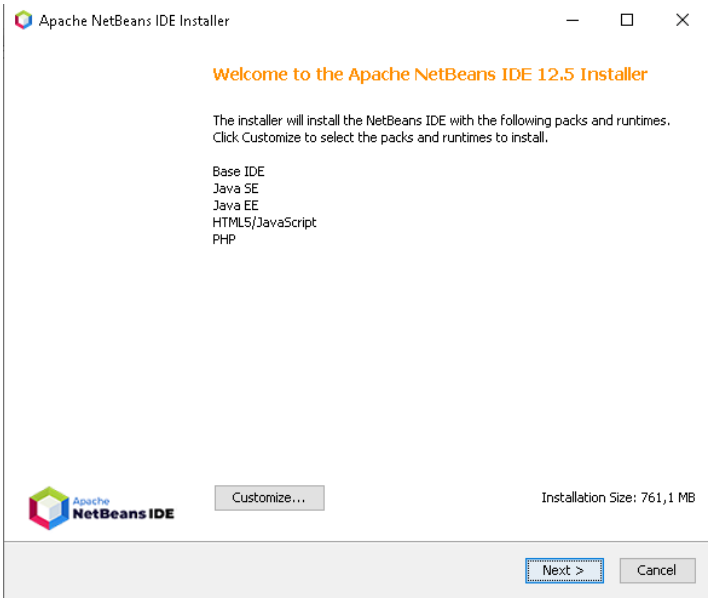
Elegimos la última versión y nos podemos descargar el IDE en su versión 12 desde el botón **Download**:



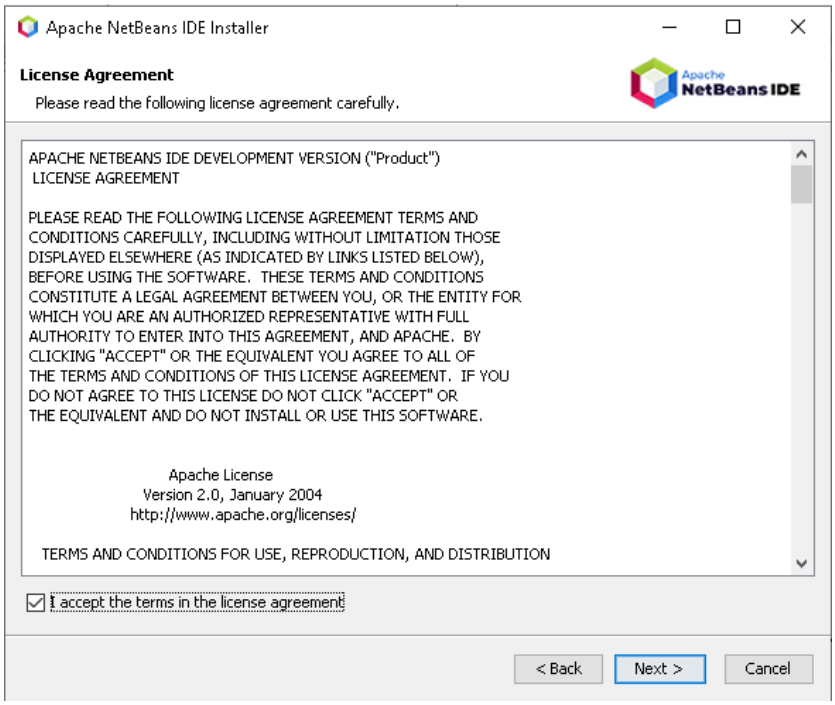
Elegimos el archivo marcado que es un .exe con la instalación de Netbeans. A continuación, nos aparece una ventana en la que nos sugiere un sitio (mirror) desde donde nos podemos descargar el IDE.

Finalizada la descarga, obtenemos el siguiente archivo: **Apache-NetBeans-12.5-bin-windows-x64.exe** que podremos instalar haciendo doble clic en él. Después de unos instantes se configurará el instalador y a continuación nos da la bienvenida a la instalación de Netbeans IDE 12.5.

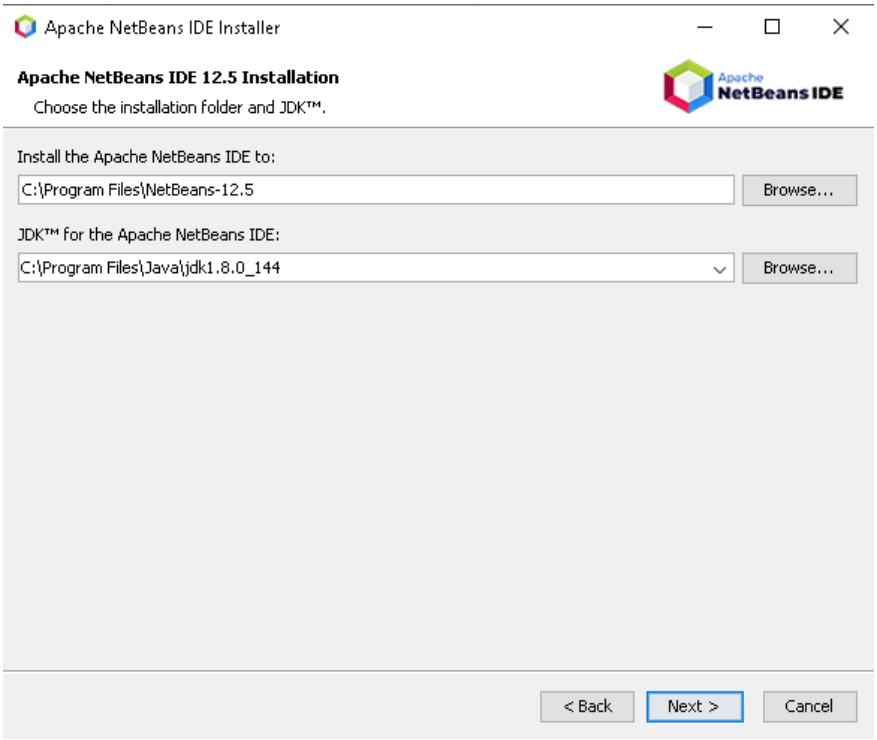




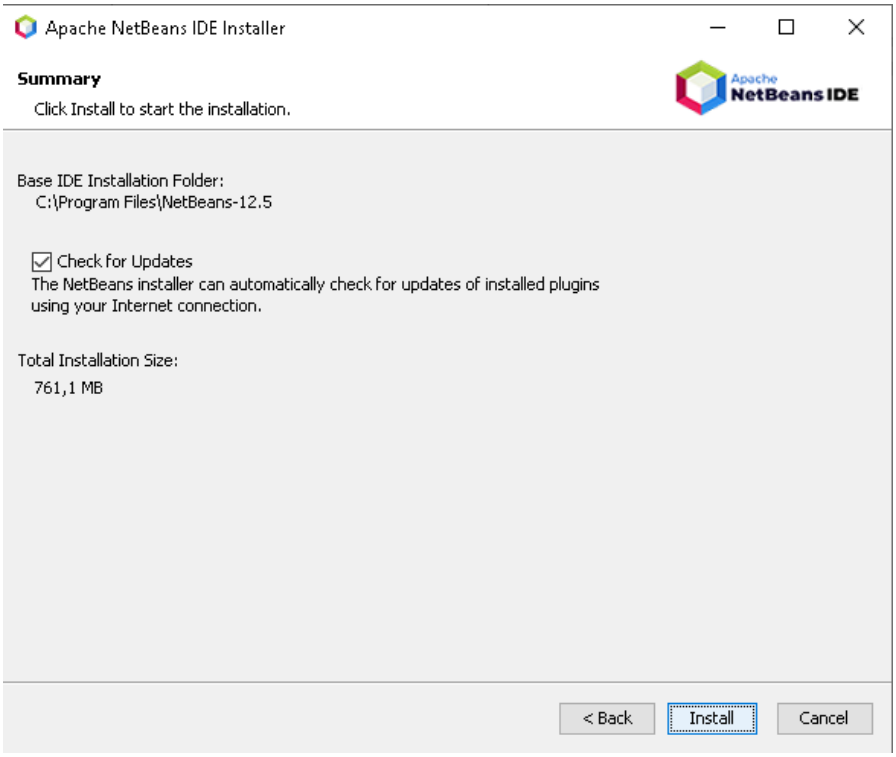
Si pulsamos en **Next** nos ofrece los datos de la licencia y debemos aceptar sus términos.



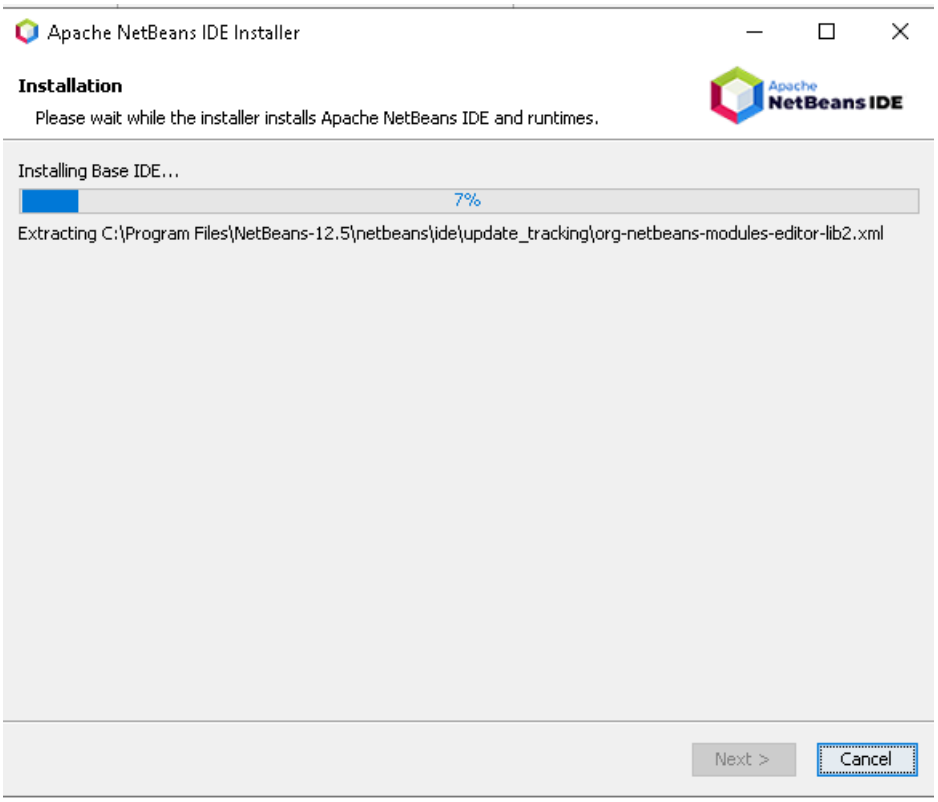
A continuación nos informa de la carpeta donde instalará el IDE y el JDK que confirmaremos (o podremos cambiar) dándole al botón **Next**:

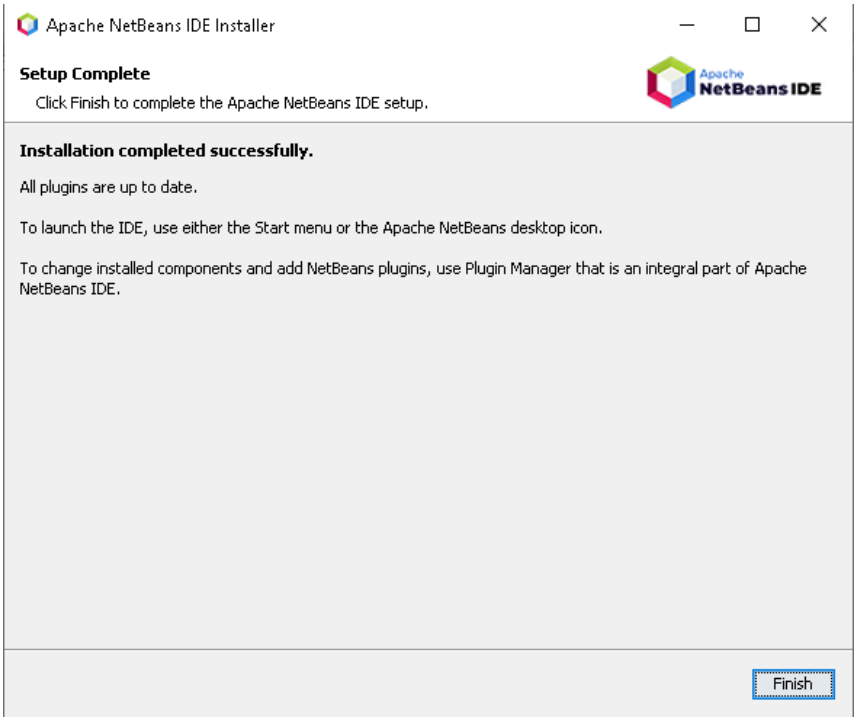


Finalmente nos hace un breve resumen de los datos de instalació:



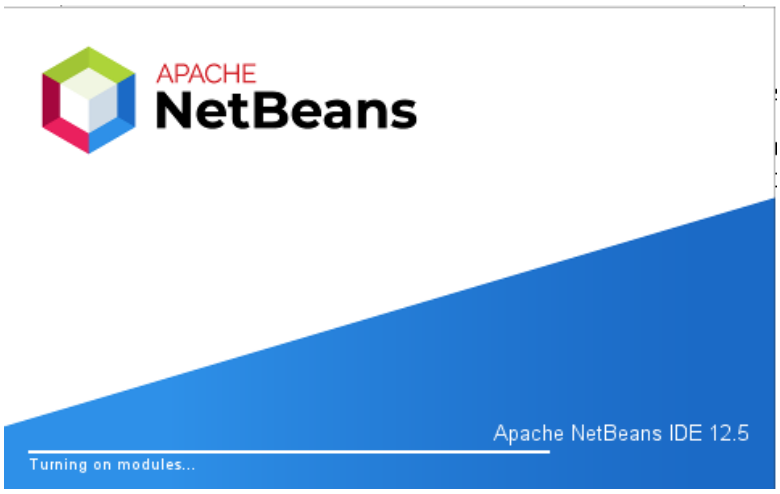
Cuando presionemos el botón **Install** comenzará la instalación en nuestro equipo



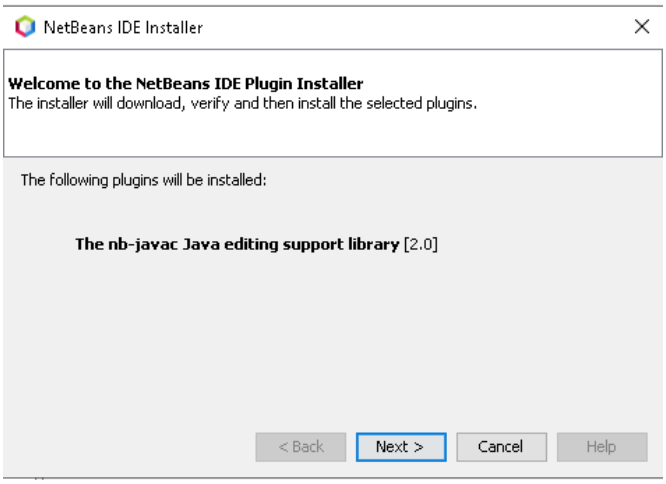


Al acabar la instalación una nueva ventana nos informa del éxito de esa instalación.

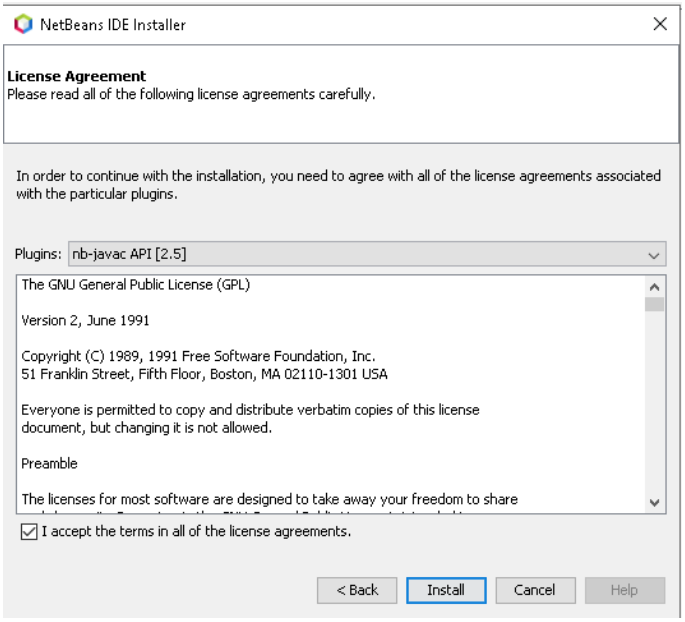
Todo listo. Tendremos un nuevo acceso directo en escritorio en el que hacer doble clic para comprobar que se ha instalado correctamente, y debería aparecer la pantalla de carga:



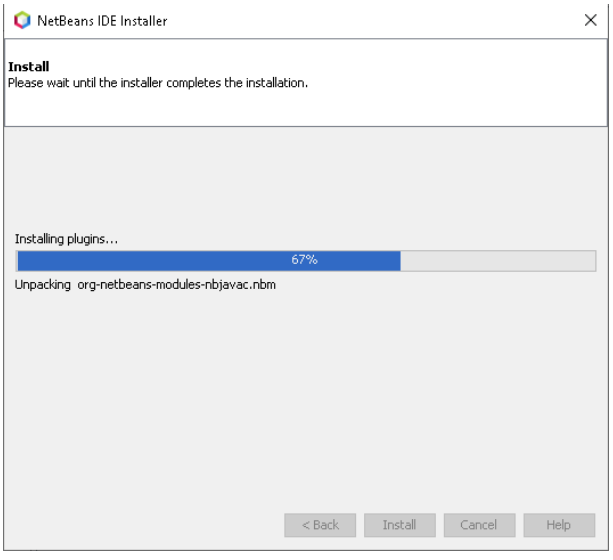
Si es necesario, Netbeans nos informa de los pluggins que son necesarios instalar para su funcionamiento:



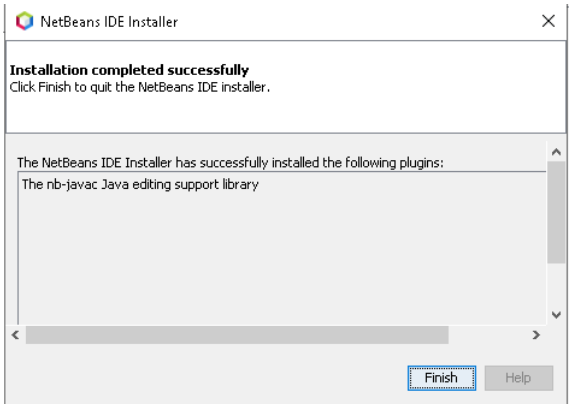
Presionamos el botón **Next** y nos lleva a la pantalla



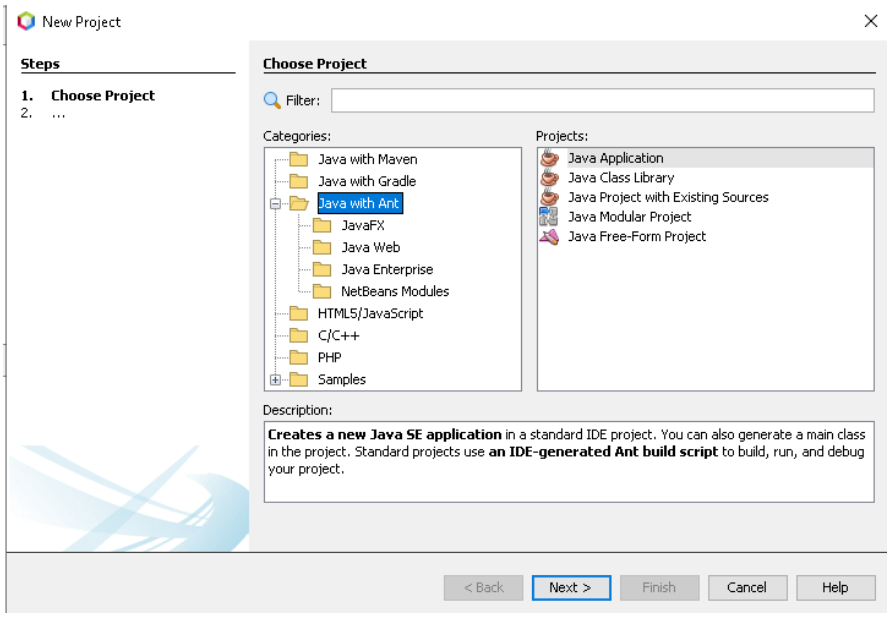
Aceptamos los términos y luego **Install** para proseguir con la descarga del plugin.

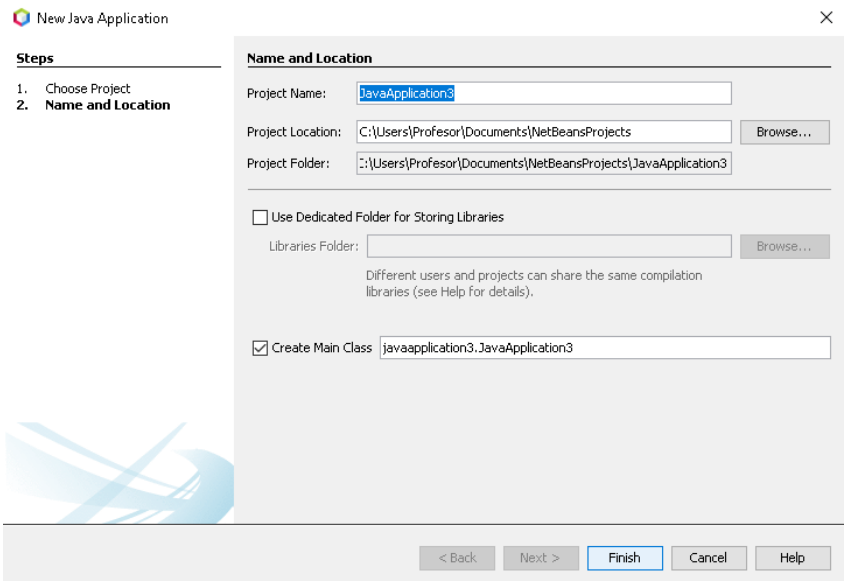


Cuando acaba su instalación nos informa de que ya se ha completado la instalación del IDE.

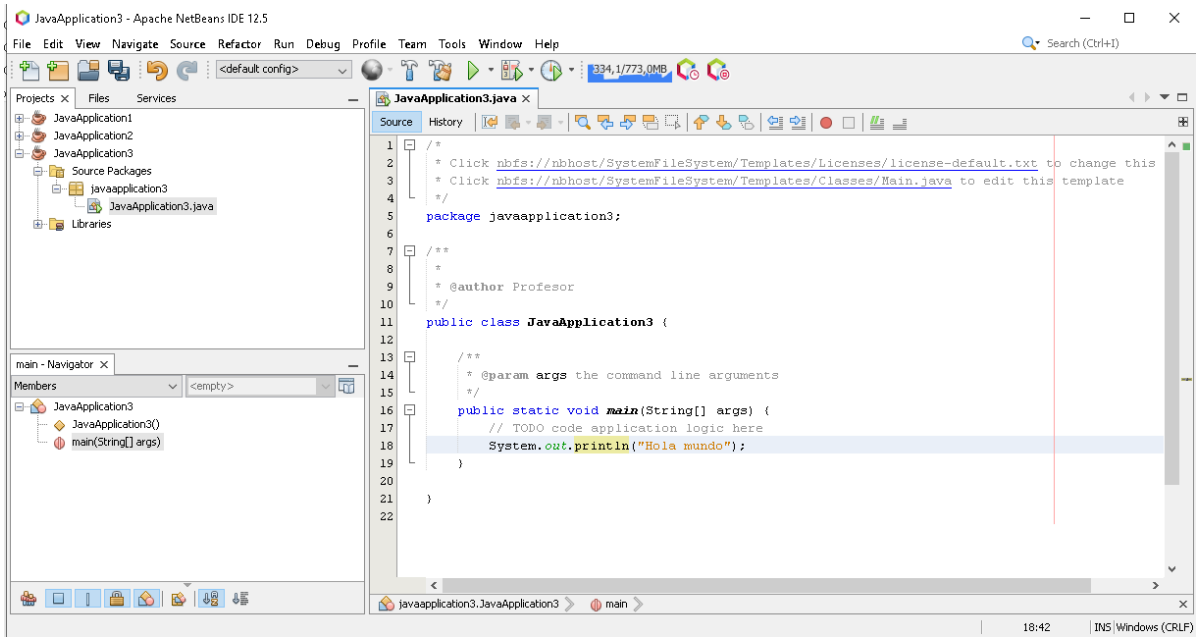


Ya estaríamos listos para empezar a crear nuestro primer programa en Java, para ello debemos seleccionar **File → New project**…→ **Java with Ant →Java Application** y ponerle el nombre (por defecto nos saldrá JavaApplication1.java





Al pulsar **Finish** ya estaremos en nuestro proyecto y podremos escribir el código de nuestro programa.

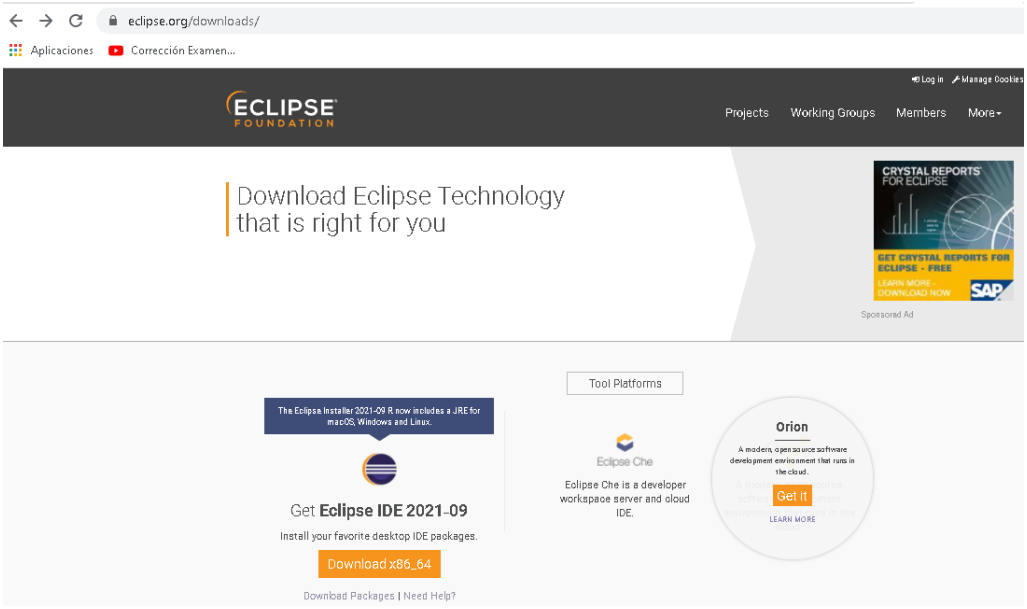


11. Instalación del IDE Eclipse

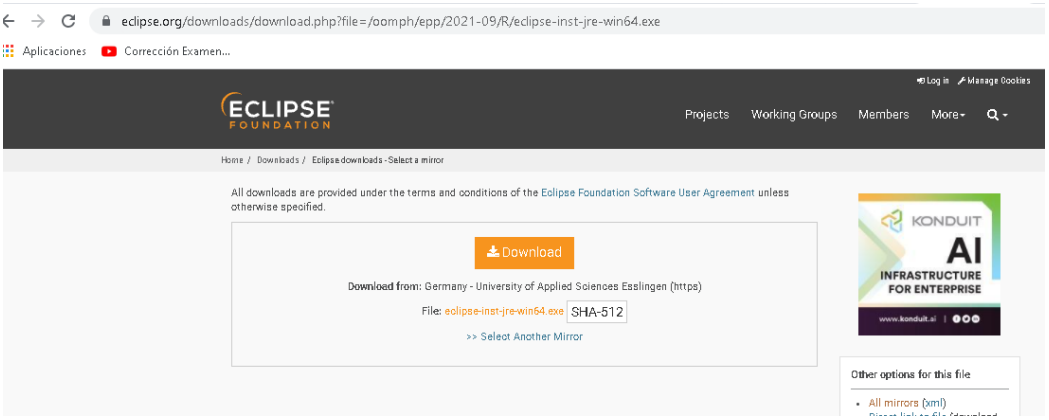
1.- Entramos en la página de descargas de Eclipse:

https://www.eclipse.org/downloads/

2.- Normalmente Eclipse detecta el sistema operativo y propone descargar la última versión del IDE. En este caso es la **Eclipse IDE 2021-09**



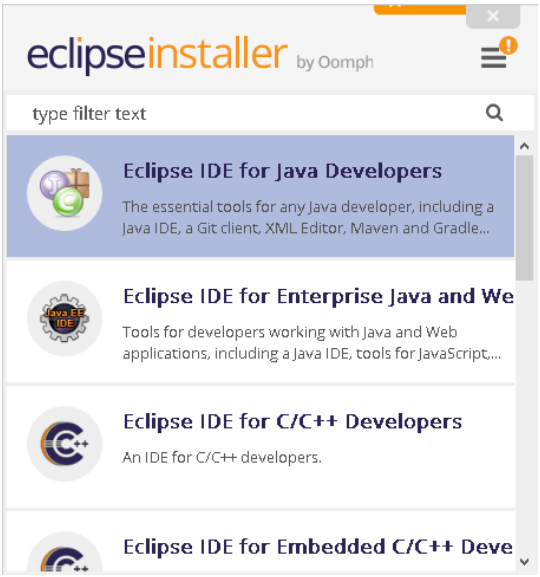
Pulsamos en el botón **Download x86_64** y aparece la ventana:



3.- Hacemos clic en **Download** y se nos descargará el archivo **eclipse-inst-jre-win64.exe**.

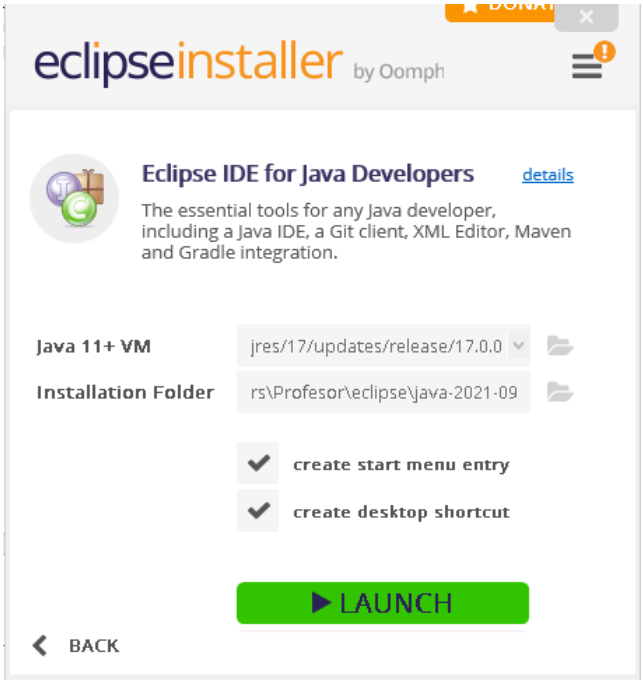
Para instalar el IDE solo tenemos que ejecutar este archivo descargado.

Nos preguntará qué instalación queremos realizar, elegiremos **Eclipse IDE for Java Developers.**



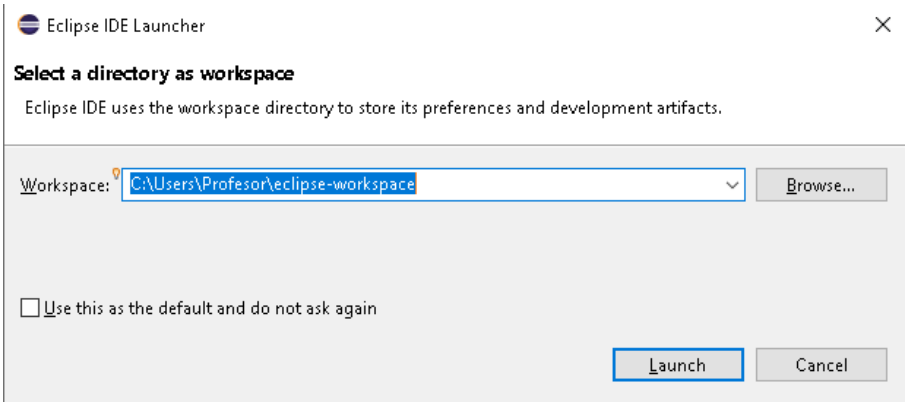
Nos mostrará el producto a instalar y si queremos acceso directo y desde el inicio.

Le damos al botón **INSTALL** y nos instalará Eclipse. Al finalizar nos muestra el final del proceso.

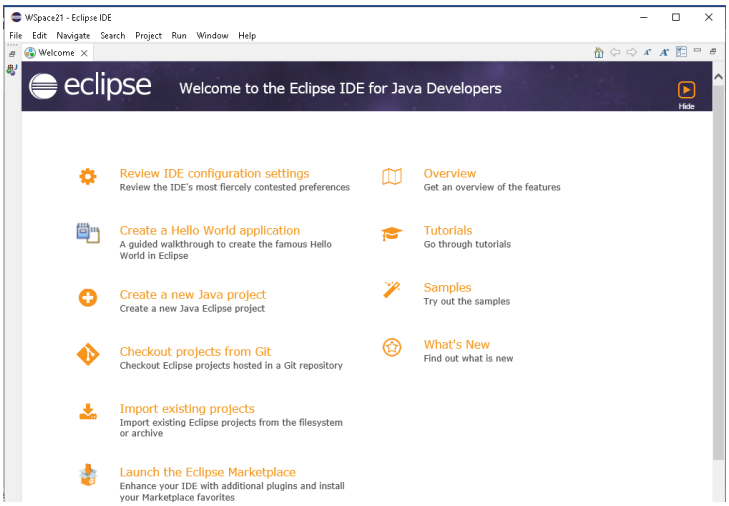


Al presionar el botón LAUNCH, nos iniciará Eclipse.

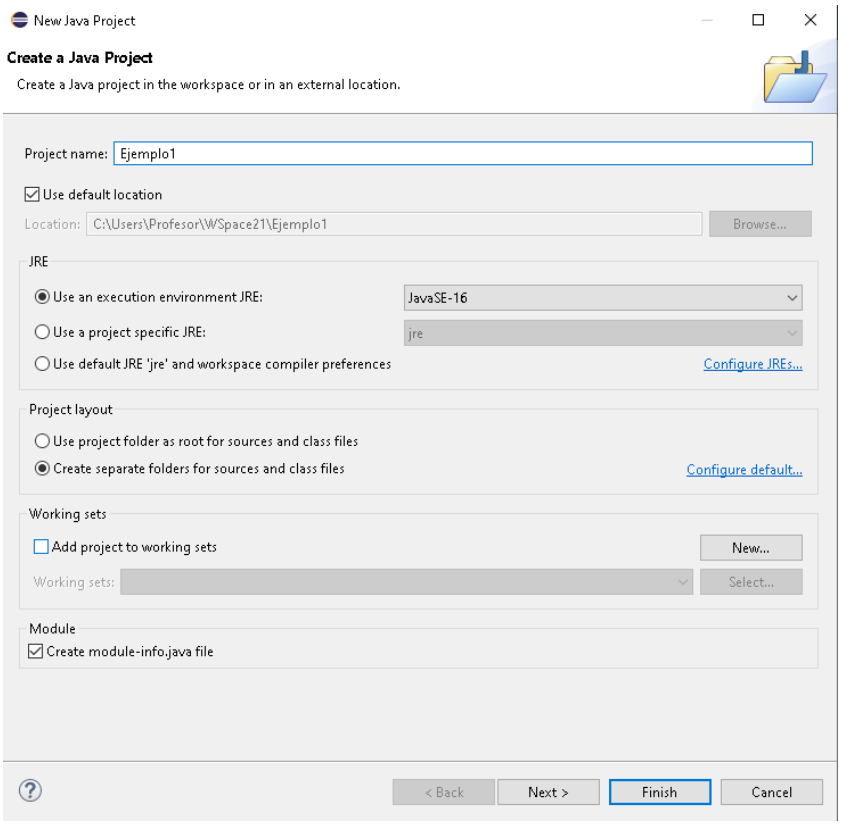
La primera operación a realizar en IDE es seleccionar el directorio de trabajo (Workspace) en el que vamos a trabajar.



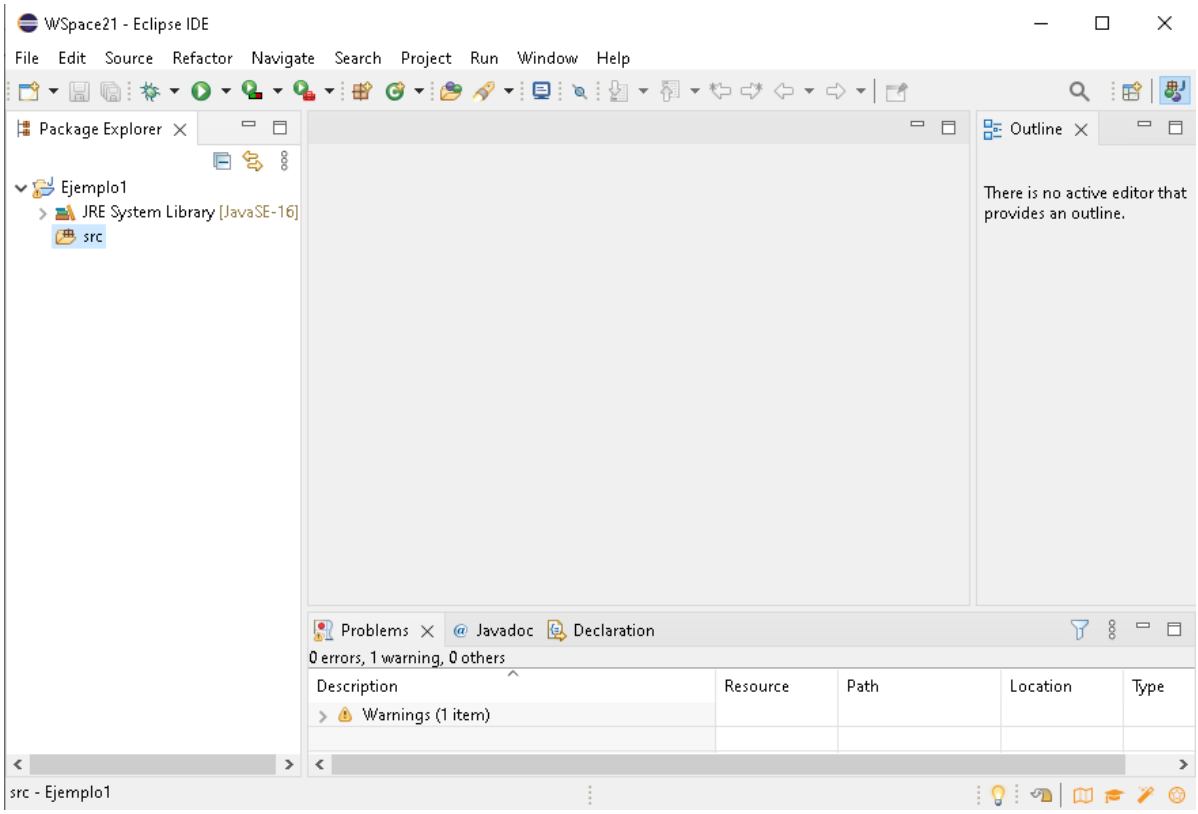
La primera vez que ejecutamos Eclipse nos aparece una ventana desde la que podemos realizar una serie de tareas como: Revisar la configuración del IDE, crear una aplicación Hola Mundo, crear un nuevo proyecto Java, ver tutoriales, etc. Nosotros elegimos crear un nuevo proyecto Java.



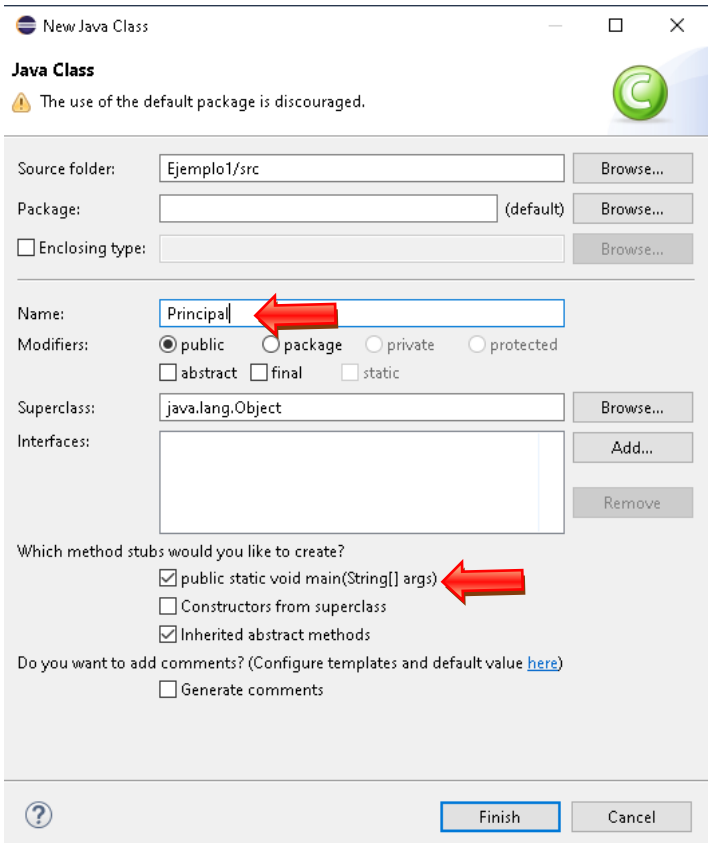
Se abre una ventana para poner el nombre del nuevo proyecto Java.



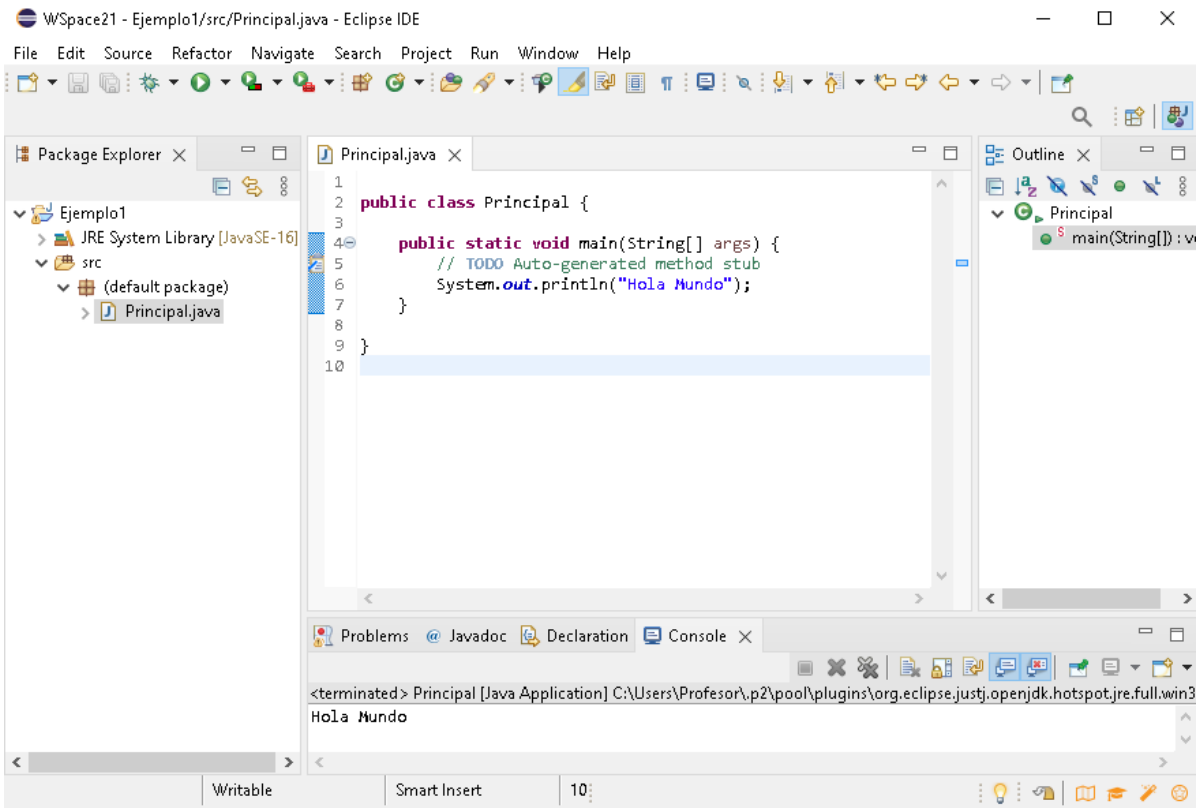
Y al presionar en **Finish** nos abre el IDE y podemos ver el aspecto que tiene. Ya estaríamos listos para empezar a crear nuestro primer programa en Java, para ello debemos seleccionar **File → New → Java Project** y ponerle el nombre Ejemplo1.java



Ahora que ya tenemos el proyecto, vamos a crear la primera clase que llamaremos **Principal** que contendrá un **método public static void main** que será el comienzo de nuestra aplicación. Esto se consigue seleccionando **File → New → Class,** poniendo el nombre a la clase y activando la casilla del método main.



Al pulsar **Finish** ya estaremos en nuestro proyecto y podremos escribir el código de nuestro programa.



13. Comparación de los IDES de Java: Eclipse vs NetBeans vs IntelliJ

Fuente: https://www.lifewire.com/comparing-java-ides-eclipse-vs-netbeans-vs-intellij-2373152 Actualizado el 24 de junio de 2018.

Seleccionar y trabajar con el IDE correcto o el entorno de desarrollo integrado es un aspecto vital para convertirse en un exitoso desarrollador de aplicaciones móviles. El IDE correcto permite a los desarrolladores manejar classpath; crear archivos; construir argumentos de línea de comandos y mucho más. En este apartado comparamos de 3 IDE de Java muy populares, a saber, Eclipse, NetBeans e IntelliJ.

Eclipse

Eclipse existe desde el año 2001, desde que IBM lanzó Eclipse como una plataforma de código abierto. Gestionado por la fundación sin fines de lucro Eclipse, se utiliza tanto en proyectos de código abierto como en proyectos comerciales. Comenzando de una manera humilde, esto ahora se ha convertido en una plataforma importante, que también se usa en varios otros idiomas.

La mayor ventaja de Eclipse es que cuenta con una gran cantidad de complementos, lo que lo hace versátil y altamente personalizable. Esta plataforma funciona para usted en segundo plano, compila el código y muestra errores cuando ocurren. El IDE completo está organizado en Perspectivas, que son esencialmente contenedores visuales, que ofrecen un conjunto de vistas y editores.

Las tareas múltiples, el filtrado y la depuración de Eclipse son otras ventajas. Diseñado para adaptarse a las necesidades de grandes proyectos de desarrollo, puede manejar diversas tareas como análisis y diseño, gestión de productos, implementación, desarrollo de contenido, pruebas y documentación también.

NetBeans

NetBeans se desarrolló de forma independiente en la segunda mitad de los años noventa. Surgió como una plataforma de código abierto después de que Sun la adquiriera en 1999. Ahora, como parte de Oracle, este IDE se puede usar para desarrollar software para todas las versiones de Java, desde Java ME hasta Enterprise Edition. Al igual que Eclipse, NetBeans también cuenta con una variedad de complementos con los que puede trabajar.

NetBeans le ofrece varios paquetes diferentes: 2 ediciones C / C ++ y PHP, una edición de Java SE y la edición de Java EE que ofrece todo lo que necesitará para su proyecto. Este IDE también ofrece herramientas y editores que se pueden usar para HTML, PHP, XML, JavaScript y más. Ahora puede encontrar soporte para HTML5 y otras tecnologías web también.

NetBeans obtiene puntuaciones sobre Eclipse en el sentido de que cuenta con soporte de base de datos, con controladores para Java DB, MySQL, PostgreSQL y Oracle. Su Explorador de bases de datos le permite crear, modificar y eliminar tablas y bases de datos fácilmente dentro del IDE.

En gran parte, visto en el pasado como una especie de sombra de Eclipse,

NetBeans ahora se ha convertido en un competidor formidable para el primero.

IntelliJ IDEA

En existencia desde 2001, IntelliJ IDEA JetBrains está disponible en una edición comercial, así como también en una edición gratuita de comunidad de código abierto. JetBrains es una empresa establecida y es más conocida por su complemento Resharper para Visual Studio y es especialmente beneficiosa para el desarrollo de C #.

IntelliJ ofrece soporte para una variedad de idiomas, incluyendo Java, Scala, Groovy, Clojure y más. Este IDE incluye características como la finalización inteligente de códigos, el análisis de códigos y la refactorización avanzada. La versión comercial "Ultimate", que se dirige principalmente al sector empresarial , también admite SQL, ActionScript, Ruby, Python y PHP. La versión 12 de esta plataforma también viene con un nuevo diseñador de IU de Android para el desarrollo de aplicaciones para Android.

IntelliJ también cuenta con varios complementos escritos por el usuario. Actualmente ofrece 947 complementos, más 55 adicionales en su versión empresarial. Los usuarios siempre son bienvenidos a enviar más complementos utilizando sus componentes Swing incorporados.

En conclusión

Todos los IDE anteriores vienen con sus propias ventajas. Si bien Eclipse sigue siendo el IDE más utilizado, NetBeans ahora está ganando popularidad entre los desarrolladores independientes. Si bien la edición empresarial de IntelliJ funciona como una maravilla, algunos desarrolladores pueden considerar que es un gasto innecesario.

Todo depende de lo que esté buscando, como desarrollador, y cómo planea seguir adelante con su trabajo. Instala todos los 3 IDE y pruébalos antes de hacer tu elección final.