



CASO PRÁCTICO

Tras el éxito del anterior proyecto, en GoyoProg están recibiendo más peticiones de creación de software que nunca. Ana y Antonio, que ya hace unas semanas que están estudiando el Ciclo de Diseño de Aplicaciones, piensan que este es un buen momento para participar activamente en los proyectos, pues a sus compañeros no les vendría nada mal un poco de ayuda. La fase de codificación es compleja, pero Ana y Antonio están aprendiendo a dominar los llamados entornos integrados de desarrollo de software... Ana se muestra muy ilusionada y no piensa desperdiciar esta gran oportunidad.

Todos en la empresa están sorprendidos del entusiasmo de Ana ante los nuevos proyectos que GF Programación tiene por delante. Juan, que acabó el Ciclo Superior de DAM hace algunos años, se muestra inquieto porque es consciente de que en sólo unos cuatro años han salido muchas herramientas nuevas en el mercado y necesita reciclarse...

1. Concepto de Entorno de Desarrollo. Evolución Histórica.

En la unidad anterior hablábamos de las fases en el proceso de desarrollo de software. Una de ellas era la **fase de codificación**, en la cual se hacía uso de algún lenguaje de programación para pasar todas las acciones que debía llevar a cabo la aplicación a algún lenguaje que la máquina fuera capaz de entender y ejecutar. También se hizo alusión a herramientas de apoyo al proceso de programación.

En esta unidad vamos a analizar, instalar y ejecutar estas herramientas para entender su acción y efecto. Muchas personas aprenden a programar utilizando un **editor de texto simple, compilador y depurador**. Pero la mayoría, finalmente, terminan haciendo uso de algún **entorno de desarrollo integrado (IDE)** para crear aplicaciones.

Un **entorno integrado de desarrollo (IDE)**, es un tipo de software compuesto por un conjunto de herramientas de programación.

Los **primeros entornos de desarrollo integrados nacen** a principios de los **años 70**, y se popularizan en la década de los 90. Tienen el objetivo de ganar fiabilidad y tiempo en los proyectos de software.

Proporcionan al programador una serie de componentes con la misma interfaz gráfica, con la consiguiente **comodidad, aumento de eficiencia y reducción de tiempo** de codificación.

Normalmente, un IDE está dedicado a un determinado lenguaje de programación. No obstante, las **últimas versiones** de los **IDEs** tienden a ser **compatibles** con **varios lenguajes** (por ejemplo, Eclipse, NetBeans, Microsoft Visual Studio...) mediante la instalación de Plugin adicionales.

En este tema, nuestro interés se centra en conocer los entornos de desarrollo, los tipos, en función de su licencia y del lenguaje de programación hacia el cual están enfocados. Instalaremos NetBeans bajo Windows y veremos cómo se configura y cómo se generan ejecutables, haciendo uso de sus componentes y herramientas.

1.1. Evolución Histórica

En las décadas de utilización de la tarjeta perforada como sistema de almacenamiento el concepto de Entorno de Desarrollo Integrado sencillamente no tenía sentido. Los programas estaban escritos con diagramas de flujo y entraban al sistema a través de las tarjetas perforadas. Posteriormente, eran compilados.

El **primer lenguaje** de programación que **utiliza** un **IDE** fue el **BASIC** (que fue el primero en abandonar también las tarjetas perforadas o las cintas de papel). Éste primer IDE estaba **basado en consola de**

comandos exclusivamente (normal por otro lado, si tenemos en cuenta que hasta la década de los 90 no entran en el mercado los sistemas operativos con interfaz gráfica). Sin embargo, el uso que hace de la gestión de archivos, compilación, depuración... es perfectamente compatible con los IDE actuales.

A nivel popular, el **primer IDE** puede considerarse que fue el IDE llamado **Maestro**. Nació a principios de los 70 y fue instalado por unos 22000 programadores en todo el mundo. Lideró el campo durante los años 70 y 80. El uso de los entornos integrados de desarrollo se ratifica y afianza en los 90 y hoy en día contamos con infinidad de IDE, tanto de licencia libre como no.

IDE más relevantes hoy en día		
Entorno de desarrollo	Lenguajes que soporta	Tipo de licencia
NetBeans	C/C++, Java, JavaScript, PHP, Python.	De uso público.
Eclipse	Ada, C/C++, Java, JavaScript, PHP.	De uso público.
Microsoft Visual Studio.	Basic, C/C++, C#.	Propietario.
C++ Builder.	C/C++.	Propietario.
JBuilder	Java.	Propietario.
Spring Tools Suite	Spring	De uso público

DESTACADO

No hay unos entornos de desarrollo más importantes que otros. La **elección** del IDE más adecuado **dependerá** del **lenguaje** de **programación** que vayamos a **utilizar** para la codificación de las aplicaciones y el tipo de licencia con la que queramos trabajar.

2. Funciones del entorno de Desarrollo

CASO PRÁCTICO Juan, que asume por fin su desconocimiento, habla con Ana para que le pase sus apuntes de entornos de desarrollo. Ésta se muestra encantada, y le anima a reciclarse. Juan se muestra reacio (ya he estudiado el ciclo... y durante cuatro años he cumplido con éxito en la empresa). Pero piensa que quizás debería reciclarse si no quiere quedarse atrás en los proyectos. Juan aprendió a programar usando un editor simple de textos, ¿qué ventajas tendrá programando con un IDE?

Como sabemos, los **entornos de desarrollo** están **compuestos** por una serie de **herramientas** software de programación, necesarias para la consecución de sus objetivos. Estas **herramientas** son:

- ✓ Un Editor de código fuente.
- ✓ Un compilador y/o Intérprete.
- ✓ Automatización de generación de herramientas.
- ✓ Un depurador.

Las **funciones** de los IDE son:

- ✓ **Editor de Texto:** Coloración de la sintaxis.
- ✓ **Autocompletado** de código, atributos y métodos de clases.
- ✓ **Identificación automática** de código.
- ✓ **Herramientas de concepción visual** para crear y manipular componentes visuales.
- ✓ **Asistentes y utilidades de gestión y generación** de código.
- ✓ **Archivos fuente en unas carpetas y compilados a otras.**
- ✓ **Compilación de proyectos complejos** en un solo paso.
- ✓ **Control de versiones:** tener un único almacén de archivos compartido por todos los colaboradores de un proyecto. Ante un error, mecanismo de **auto-recuperación** a un estado anterior estable.
- ✓ **Soporta cambios de varios usuarios** de manera simultánea
- ✓ **Generador de documentación** integrado.
- ✓ **Detección de errores de sintaxis** en tiempo real.

Otras **funciones** importantes son:

- ✓ **Ofrece refactorización de código:** cambios menores en el código que facilitan su legibilidad sin alterar su funcionalidad (por ejemplo, cambiar el nombre a una variable).
- ✓ **Permite introducir automáticamente tabulaciones y espaciados** para aumentar la legibilidad.
- ✓ **Depuración:** seguimiento de variables, puntos de ruptura y mensajes de error del intérprete.
- ✓ **Aumento de funcionalidades** a través de la gestión de sus módulos y plugin.
- ✓ **Administración de las interfaces de usuario** (menús y barras de herramientas).
- ✓ **Administración de las configuraciones del usuario.**

3. Estructura de Entornos de Desarrollo

Los entornos de desarrollo, ya sean libres o propietarios, están formados por una serie de componentes software que determinan sus funciones.

Estos **componentes** son:



Editor de textos: Se resalta y colorea la sintaxis, tiene la función de autocompletar código, ayuda y listado de parámetros de funciones y métodos de clase. Inserción automática de paréntesis, corchetes, tabulaciones y espaciados.

Compilador/intérprete: Detección de errores de sintaxis en tiempo real. Características de refactorización.

Depurador: Botón de ejecución y traza, puntos de ruptura y seguimiento de variables. Opción de depurar en servidores remotos.

Generador automático de herramientas: Para la visualización, creación y manipulación de componentes visuales y todo un arsenal de asistentes y utilidades de gestión y generación código.

Interfaz gráfica: Nos brinda la oportunidad de programar en varios lenguajes con un mismo IDE. Es una interfaz agradable que puede acceder a innumerables bibliotecas y plugin, aumentando las opciones de nuestros programas.

3. Entornos Integrados libres y propietarios

Entornos Integrados Libres

Son aquellos con **licencia de uso público**.

No hay que pagar por ellos, y aunque los **más** conocidos y **utilizados** son **Eclipse** y **NetBeans**, hay bastantes más.

IDE	Lenguajes que soporta	Sistema Operativo
NetBeans	C/C++, Java, JavaScript, PHP, Python.	Windows, Linux, Mac OS X.
Eclipse	Ada, C/C++, Java, JavaScript, PHP.	Windows, Linux, Mac OS X.
Gambas	Basic.	Linux.
Anjuta	C/C++, Python, Javascript.	Linux.
Geany	C/C++, Java	Windows, Linux, Mac OS X.
GNAT Studio	Fortran	Windows, Linux, Mac OS X.

DESTACADO El aspecto de la licencia del IDE que se elija para el desarrollo de un proyecto es una cuestión de vital importancia. En su elección prevalecerá la decisión de los supervisores del proyecto y de la dirección de la empresa.

Entornos Integrados Propietarios

Son aquellos entornos integrados de desarrollo que **necesitan licencia**. No son free software, hay que pagar por ellos. El **más** conocido y **utilizado** es **Microsoft Visual Studio**, que usa el Framework .NET y es desarrollado por Microsoft.

IDE	Lenguajes que soporta	Sistema Operativo
Microsoft Visual Studio	Basic, C/C++, C#.	Windows
FlashBuilder	ActionScript.	Windows, Mac OS X.
C++ Builder	C/C++.	Windows
Turbo C++ Professional	C/C++.	Windows
JBuilder	Java	Windows
JCreator	Java	Windows, Linux, Mac OS X.
Xcode	C/C++, Java.	Mac OS X.

4. Instalación de Entornos Integrados de Desarrollo.

Vamos a realizar la instalación del IDE NetBeans sobre Windows 10. La instalación del IDE *NetBeans* ya sea en Linux, Windows o Mac OS X, **requiere** la **instalación** previa del **JDK compatible** con la versión de NetBeans que se quiera instalar.

4.1. Instalación de JDK

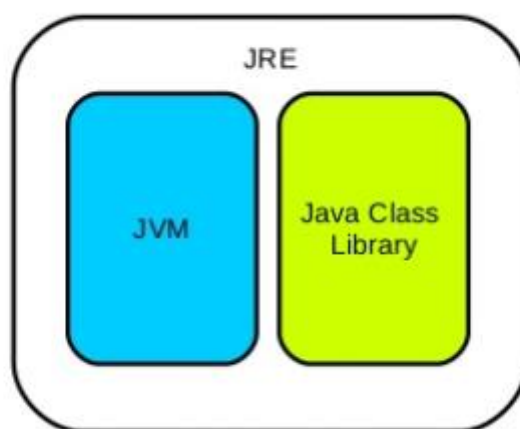
Para poder simplemente ejecutar un programa en JAVA, necesitamos tener como mínimo el **Java Runtime Environment (JRE)** instalado. **El JRE solo nos permitirá ejecutar programas una vez compilados.**

En el caso de querer desarrollar programas con Java, necesitaremos el **Java Development Kit (JDK)** o **Kit de Desarrollo de Java**, que simplemente es un software que provee las herramientas de desarrollo para la creación de programas en Java.

Nosotros instalaremos el JDK ya que con **el JRE solo podríamos ejecutar los programas una vez compilados y con extensión .class**

Elementos del JRE

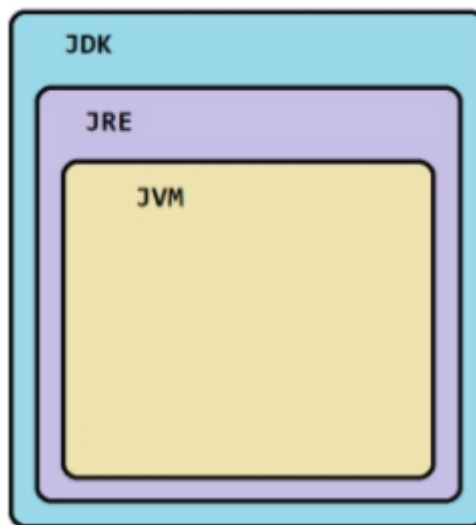
El JRE está compuesto por la máquina virtual de Java también conocida por JVM o Java Virtual Machine, un conjunto de bibliotecas Java y algunos componentes necesarios para que una aplicación en Java pueda ser ejecutada.



Elementos del JDK

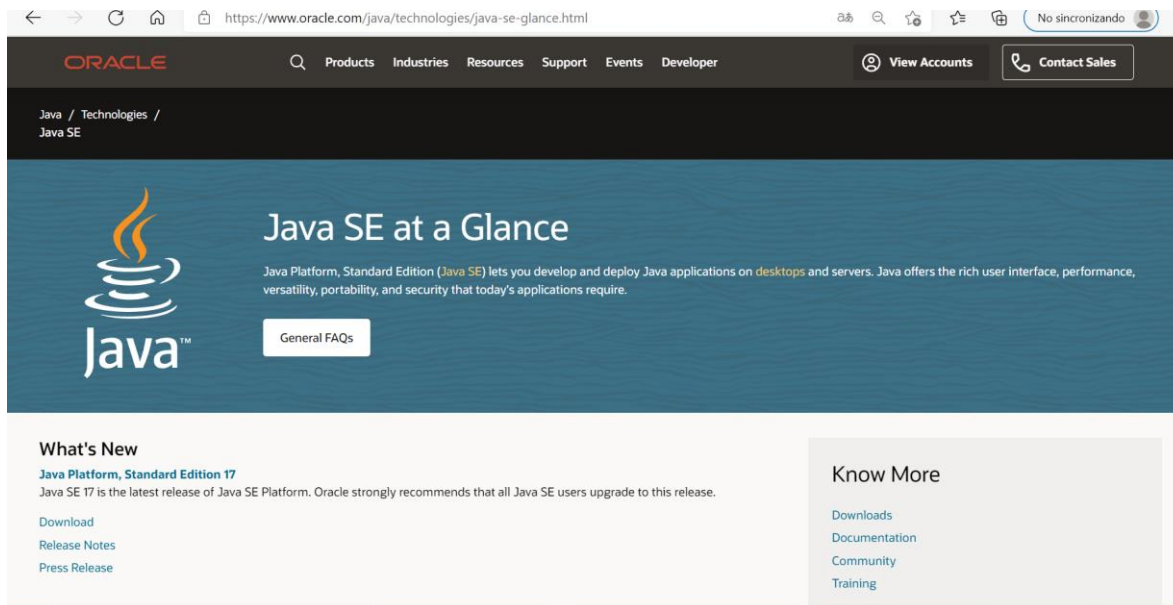
El JDK está compuesto principalmente por:

- **Appletviewer.exe:** nos permite generar vistas previas visor de applets (pequeñas aplicaciones web). Necesitamos generar vistas previas debido a que al carecer de un método main no se puede ejecutar desde el programa de Java.
- **Javac.exe:** el compilador que nos permitirá crear el archivo .class para más tarde ejecutarlo con la JVM de JAVA.
- **Java.exe:** El intérprete de Java.
- **Javadoc.exe:** nos permite generar la documentación de las clases que contiene un programa en Java.
- **JRE al completo:** Si instalamos JDK automáticamente tendremos instalado el JRE al completo (JVM, sus bibliotecas Java y sus componentes).

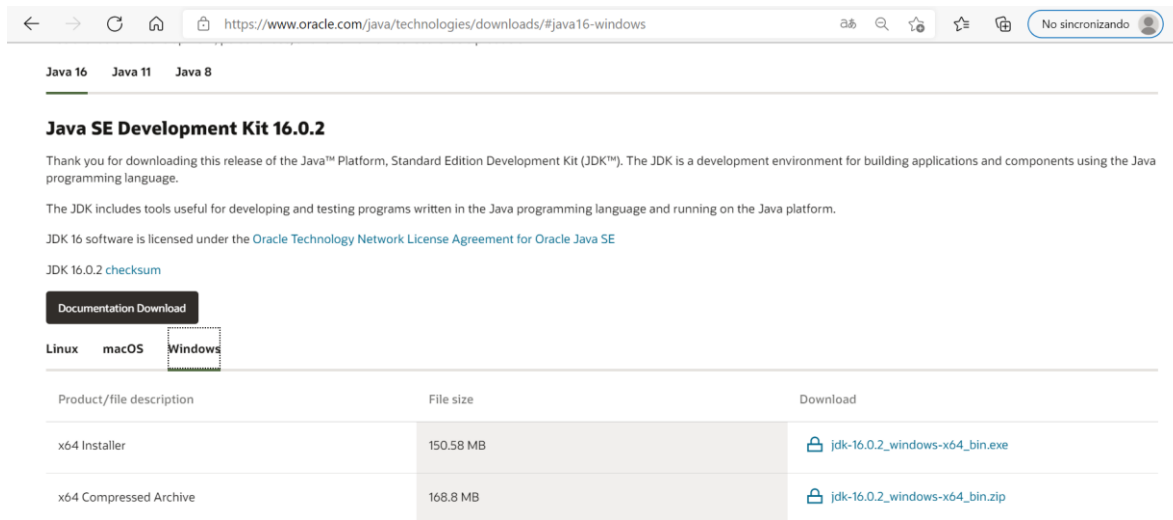


El primer paso es instalar el JDK en el sistema operativo. Esta será la **plataforma** del **entorno**, imprescindible para que éste pueda ser **instalado** en el sistema operativo y **funcionar**.

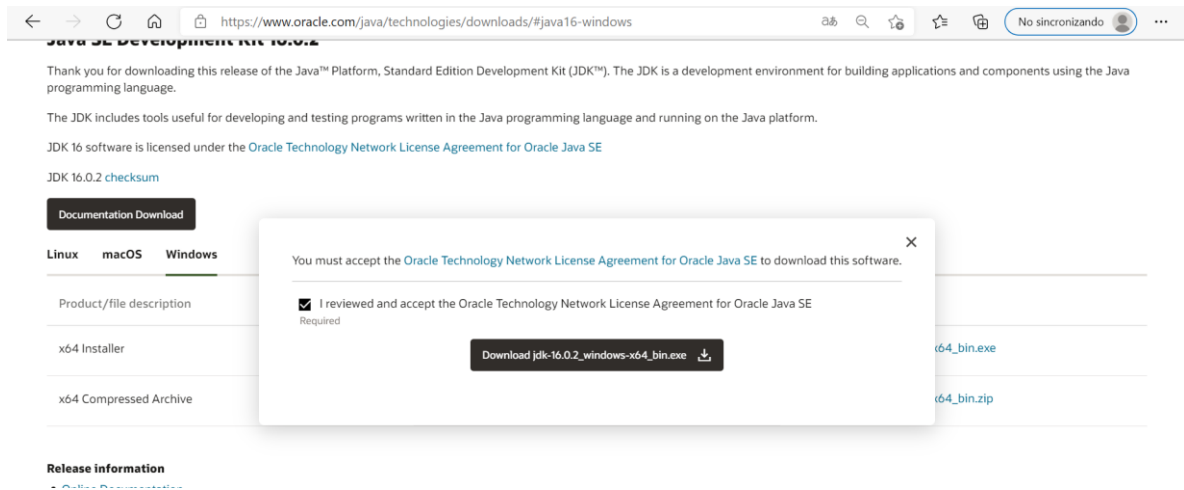
Lo primero que hay que hacer es descargar el JDK de la siguiente dirección [Java SE | Oracle Technology Network | Oraclehtml](https://www.oracle.com/java/technologies/java-se-glance.html) y se muestra la siguiente página:



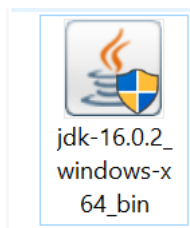
A continuación se selecciona el Download de Java SE Development Kit 16.0.2. Se selecciona el instalador .exe para la versión del sistema operativo deseado, en este caso para Windows 10 de 64 bits.



Aceptamos el **Accept License Agreement** y descargaremos la versión **jdk-16.0.2_windows-x64_bin.exe**.



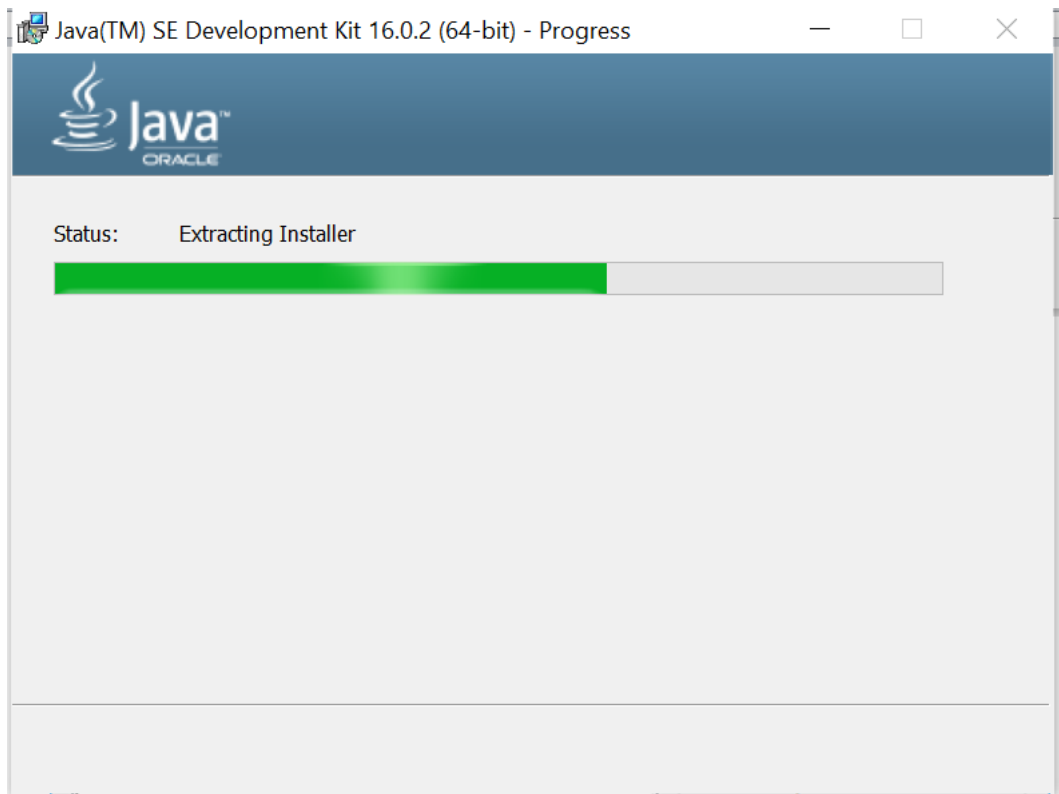
Iniciamos la descarga y guardamos el instalador en el lugar deseado.



Ejecutamos nuestro archivo descargado y nos aparece la ventana "**Java SE Development Kit 16.0.2 (64-bit)**", que nos da la bienvenida al instalador, damos click en el botón Next>:



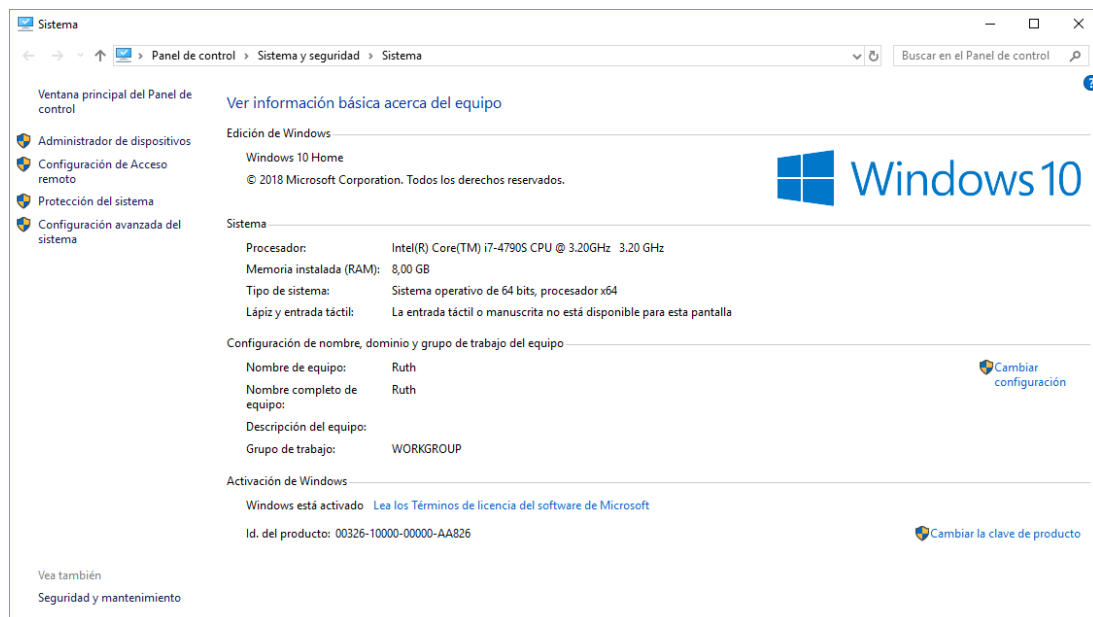
Nos aparece la ventana "**Java SE Development Kit 16.0.2 (64-bit) - Progress**" donde nos muestra el progreso de la instalación.



Nos aparece finalmente la ventana "**Java SE Development Kit 16.0.2 (64-bit) - Complete**", donde se nos indica que se ha instalado satisfactoriamente, damos click en el botón Close.

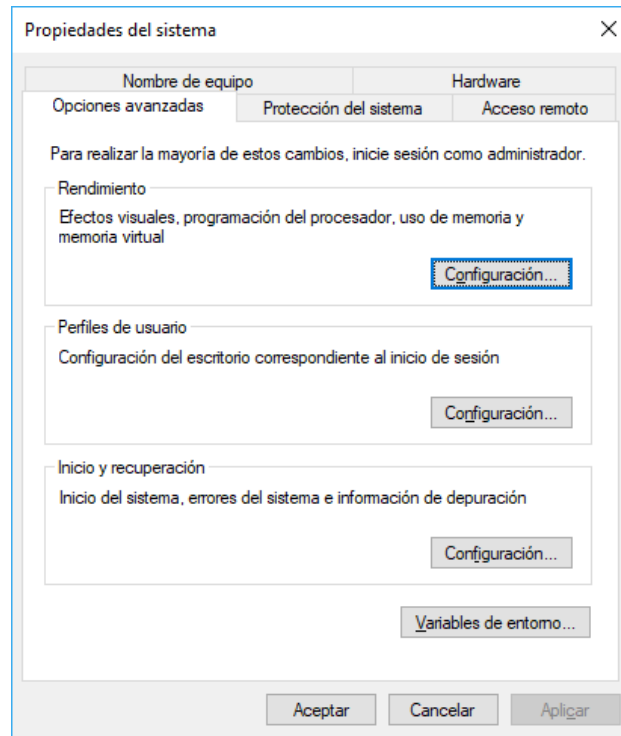


El siguiente paso será declarar la variable de entorno **JAVA_HOME** y modificar la variable **Path**, para esto hacemos click derecho sobre **Equipo** y luego click en **Propiedades** a como se muestra en la imagen a continuación:

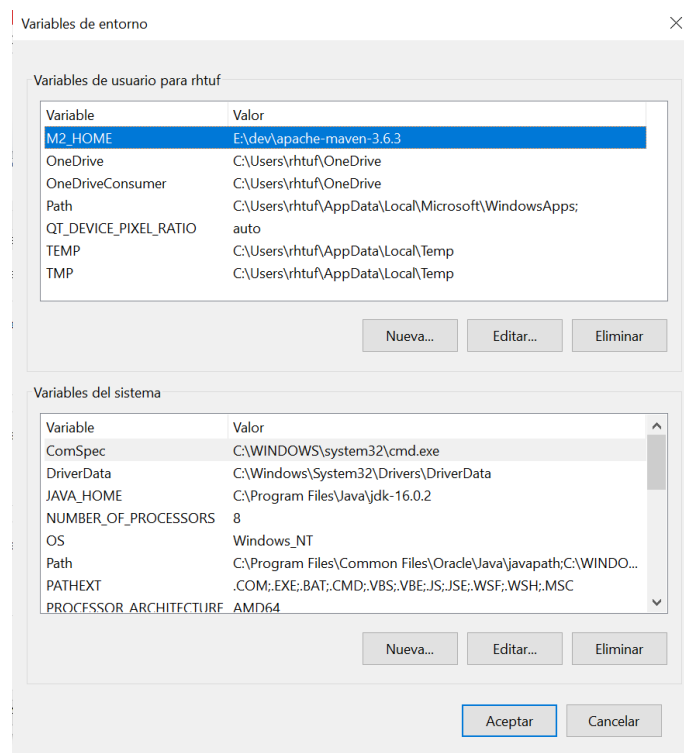


Se nos abre la ventana llamada **Ver información básica acerca del equipo**, del lado izquierdo hacemos click en **Configuración avanzada del sistema**.

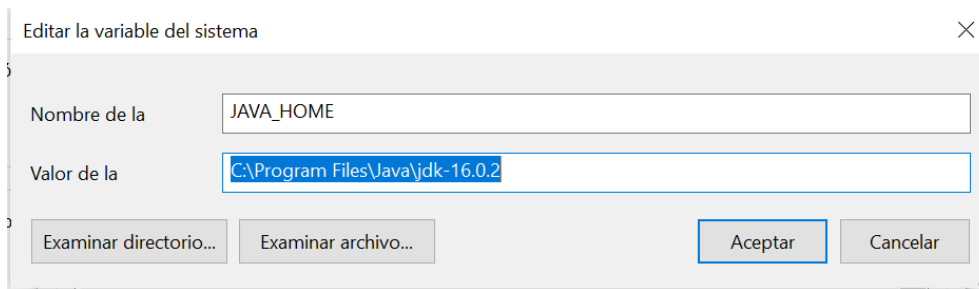
Se nos abre la ventana **Propiedades del sistema**, nos dirigimos a la pestaña **Opciones avanzadas** y hacemos click en el botón **Variables de entorno ...**.



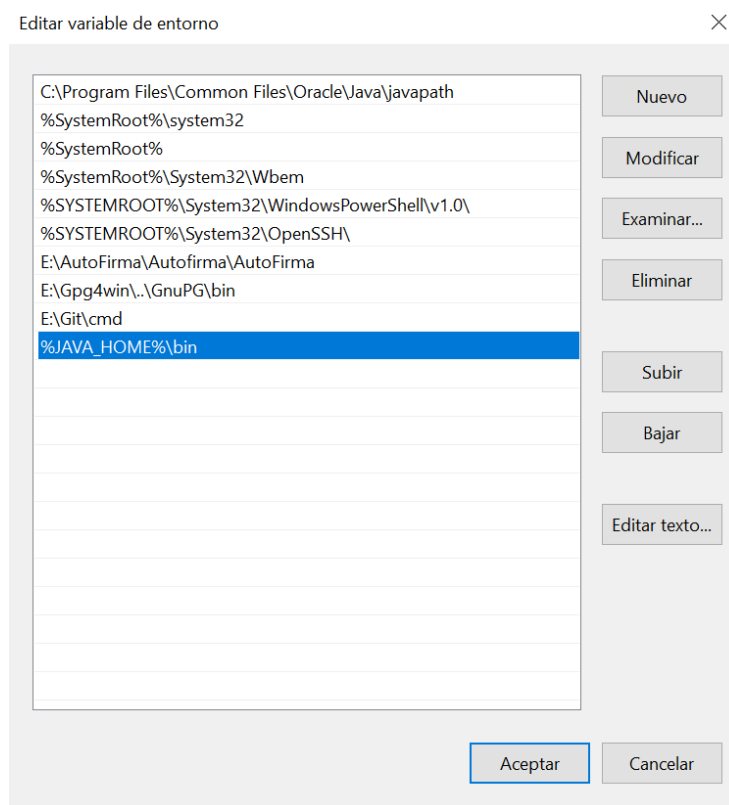
Se nos abre a continuación la ventana llamada **Variables de entorno**, y en el apartado **Variables del sistema** hacemos click en el botón **Nueva ...**.



En la ventana que se nos despliega llamada **Nueva variable del sistema** daremos de alta a nuestra variable entorno **JAVA_HOME**. Para realizar esto, en el campo **Nombre de la variable** ingresamos el texto **JAVA_HOME**, y en el campo **Valor de la variable** ingresamos la ruta de nuestro JDK.



Volvemos a la ventana anterior y en las variables del sistema seleccionamos la variable de entorno **Path** y hacemos click sobre el botón **Editar ...**, se abrirá la ventana llamada **Editar la variable del sistema**, y desde aquí añadimos un nuevo valor de la variable desde el botón **Nueva**, nos ubicamos al final del texto que contiene dicho campo e introducimos al final del texto "%JAVA_HOME%\bin". Damos en el botón **Aceptar**.



Regresamos a la ventana de **Variables de entorno** y damos click en el botón **Aceptar**. Ahora se nos mostrará la ventana de **Propiedades del sistema** y hacemos click en el botón **Aceptar**.

Ahora que todas las variables de entorno que necesitamos están configuradas, ha llegado el momento de verificar si todo ha sido configurado correctamente, para esto, abrimos el **Command Prompt** de Windows. Ejecutamos la orden `java -version` para ver que se ha instalado todo correctamente.

```
Símbolo del sistema
Microsoft Windows [Versión 10.0.19042.1237]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\rhtuf>java -version
java version "16.0.2" 2021-07-20
Java(TM) SE Runtime Environment (build 16.0.2+7-67)
Java HotSpot(TM) 64-Bit Server VM (build 16.0.2+7-67, mixed mode, sharing)

C:\Users\rhtuf>
```

4.2. Instalación de Netbeans

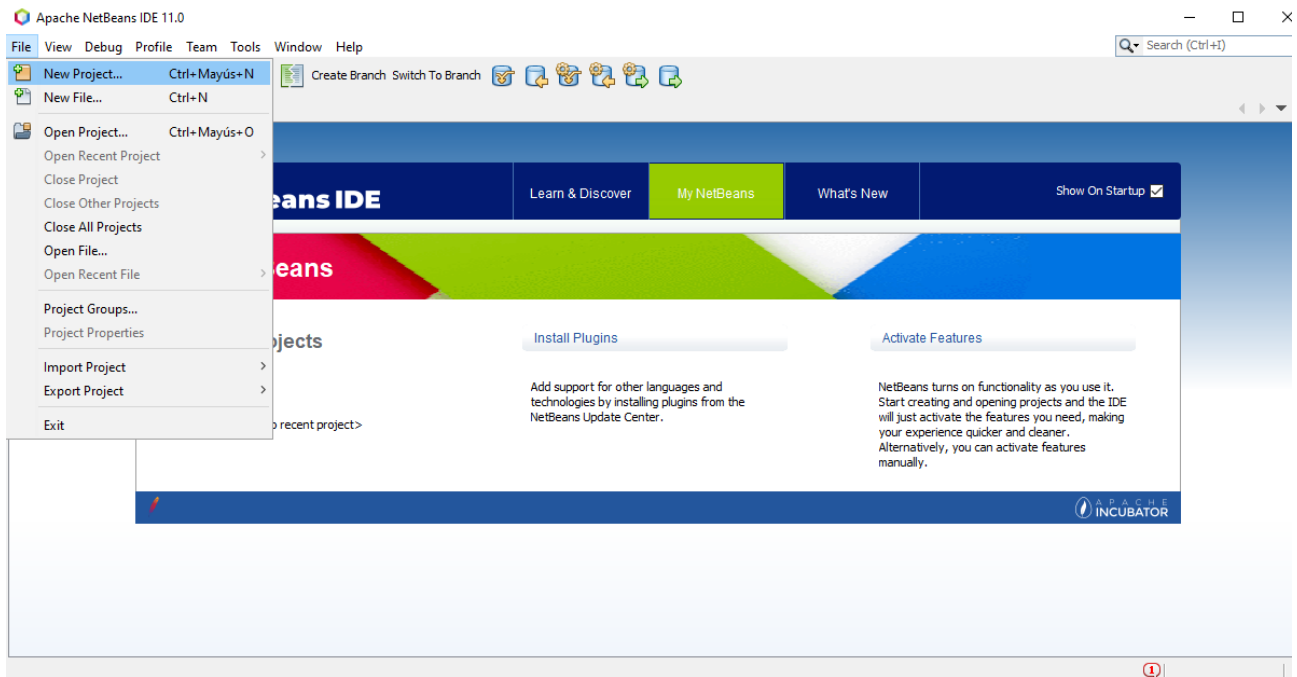
Descargamos el IDE <https://netbeans.org/> de la web oficial y procedemos a su instalación.



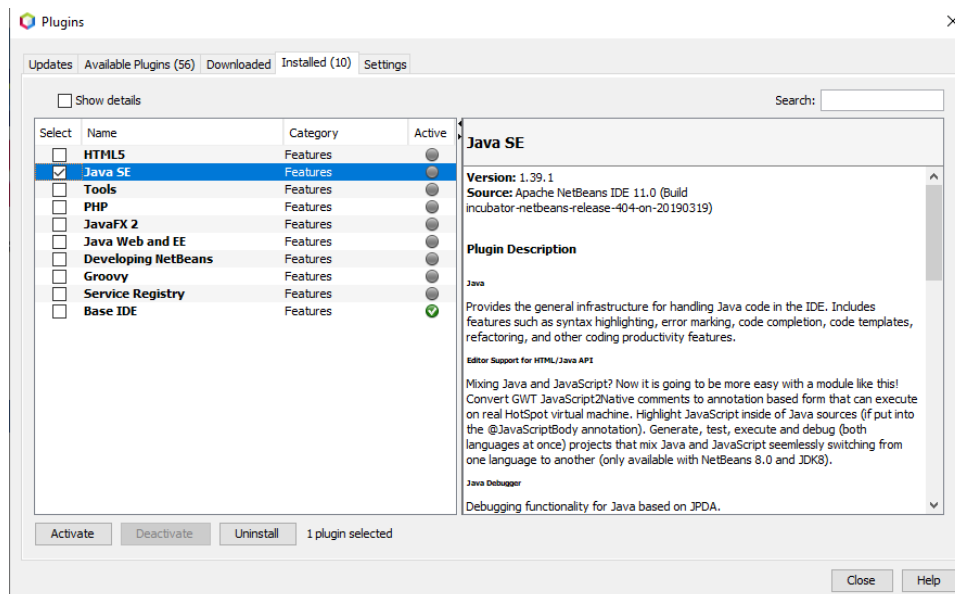
Una vez descargado el fichero comprimido de **Apache Netbeans**, lo guardamos en C:\ y lo descomprimos, para a continuación abrir el IDE de Netbeans a través del enlace: `Netbeans\netbeans\bin`



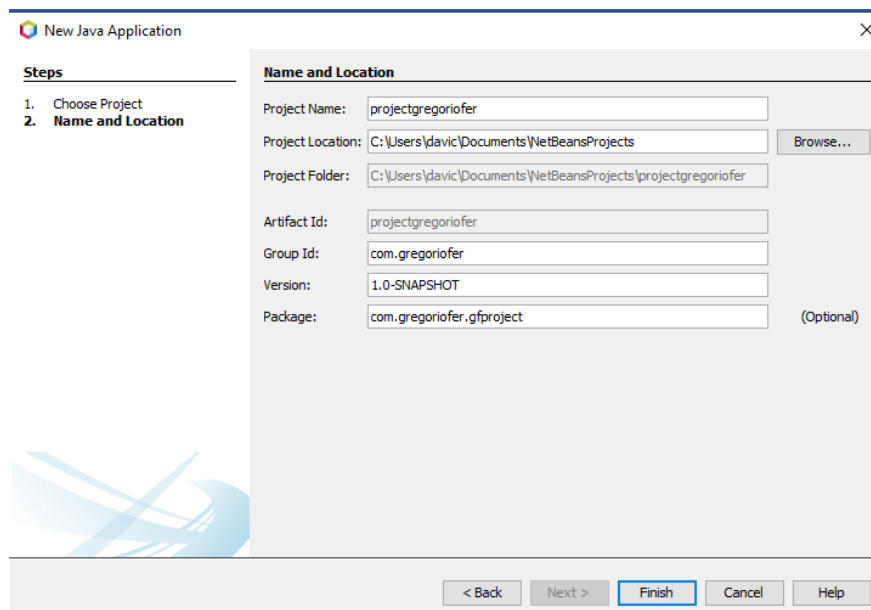
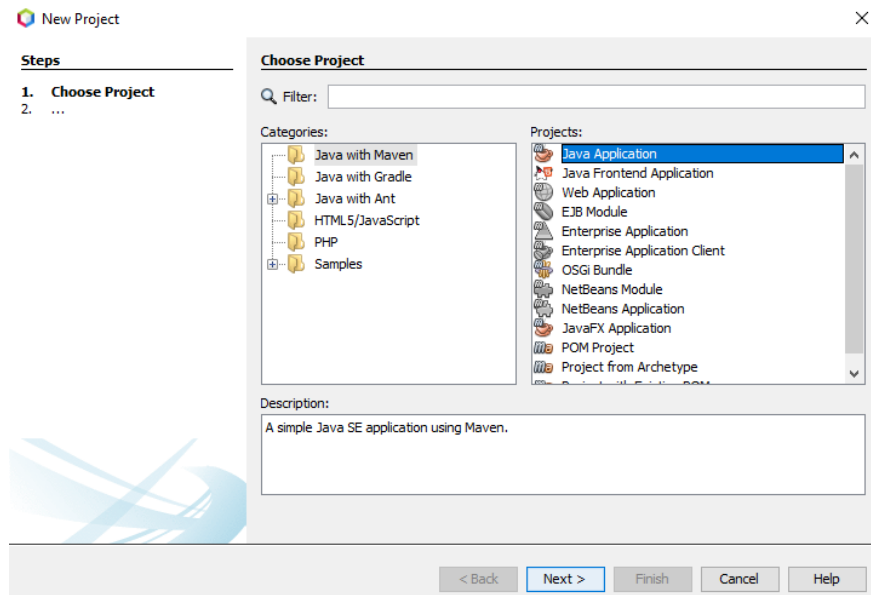
Para comenzar, seleccionamos **File->New Project...**



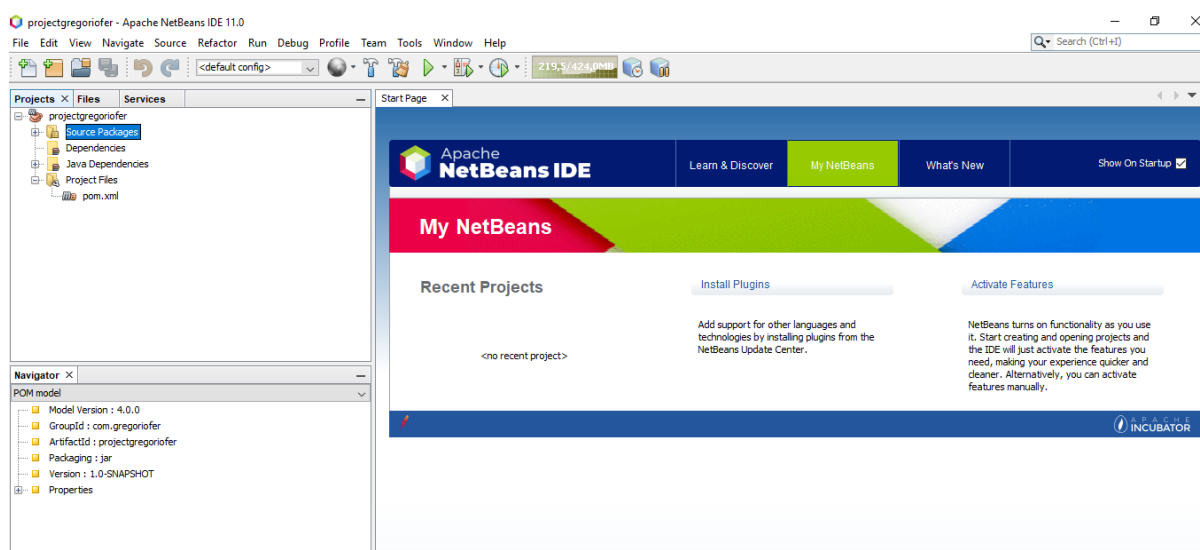
Si la opción de proyecto Java aparece deshabilitada, tendremos que activar la característica vamos a la sección '**Plugins**' desde la **opción de menú Tools** y seleccionamos la que nos interese y a continuación pulsamos sobre siguiente.



Una vez reiniciado el IDE, después de la instalación, ahora sí creamos un *nuevo proyecto*. Le damos un nombre y la ubicación donde se va a guardar: Finalmente, pulsamos **Terminar**.



La apariencia inicial del proyecto sería la siguiente:



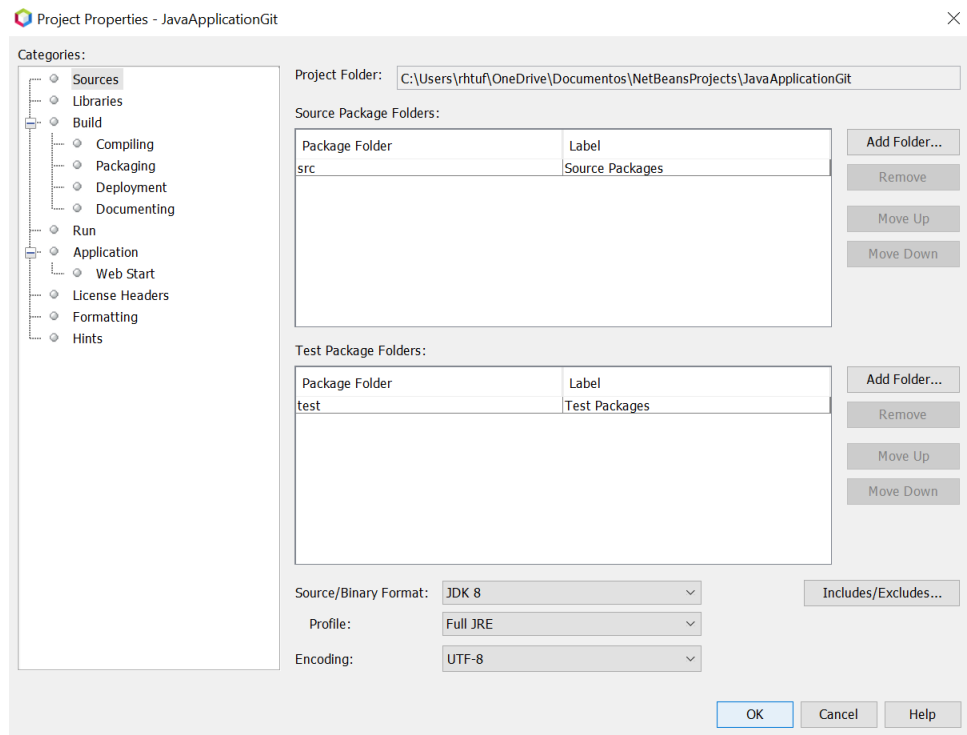
5. Configuración y Personalización de Entornos de Desarrollo

5.1. Configuración y Personalización de NetBeans

Una vez tenemos instalado nuestro entorno de desarrollo podemos acceder a personalizar su configuración. Al abrir un proyecto existente, o bien crear un nuevo proyecto, seleccionaremos un desplegable con el nombre de “configuración” desde el que podremos personalizar distintas opciones del proyecto. Podemos personalizar la configuración del entorno sólo para el proyecto actual, o bien para todos los proyectos, presentes y futuros. Parámetros configurables del entorno:

- ✓ Carpeta/s donde se alojarán todos los archivos de los proyectos (es importante la determinación de este parámetro, para tener una estructura de archivos ordenada).
- ✓ Carpetas de almacenamiento de paquetes fuente y paquetes prueba.
- ✓ Administración de la plataforma del entorno de desarrollo.
- ✓ Opciones de la compilación de los programas: compilar al grabar, generar información de depuración...
- ✓ Opciones de empaquetado de la aplicación: nombre del archivo empaquetado (con extensión .jar, que es la extensión característica de este tipo de archivos empaquetados) y momento del empaquetado.
- ✓ Opciones de generación de documentación asociada al proyecto.
- ✓ Descripción de los proyectos, para una mejor localización de los mismos.
- ✓ Opciones globales de formato del editor: número de espaciados en las sangrías, color de errores de sintaxis, color de etiquetas, opción de autocompletado de código, propuestas de insertar automáticamente código...
- ✓ Opciones de combinación de teclas en teclado.
- ✓ Etc.

Para entrar a la aplicación podemos seleccionar “*Nuevo Proyecto*” y, una vez abierto, personalizar la configuración de NetBeans para ese proyecto. En la barra de iconos de la aplicación, seleccionamos el desplegable de configuración Seleccionamos “*Propiedades*” y nos aparecerá la siguiente ventana:



Aquí vemos todo lo que podemos personalizar de la aplicación:

- ✓ Fuentes.
- ✓ Bibliotecas.
- ✓ Generación de código.
- ✓ Ejecución de código.
- ✓ Opciones de la aplicación.
- ✓ Formato del código en el editor de textos.

FUENTES: Podemos modificar:

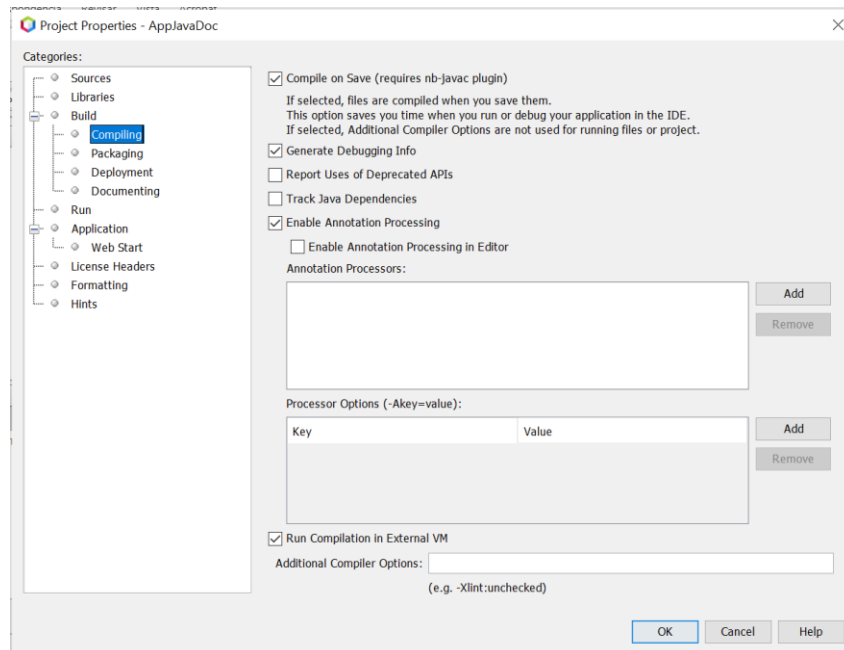
- ✓ La carpeta que contendrá el proyecto.
- ✓ La carpeta que almacenará los paquetes fuentes.
- ✓ La carpeta que contendrá los paquetes prueba.

BIBLIOTECAS: Desde esta ventana podemos elegir la plataforma de la aplicación. Toma por defecto el JDK, pero se puede cambiar si se quiere, siempre y cuando sea compatible con la versión de NetBeans utilizada. También en esta ventana se puede configurar el paquete de pruebas que se realizará al proyecto.

GENERACIÓN DE CÓDIGO - COMPILANDO Las opciones que nos permite modificar en cuanto a la compilación del programa son:

- ✓ Compilar al grabar: al guardar un archivo se compilará automáticamente.
- ✓ Generar información de depuración: para obtener la documentación asociada.

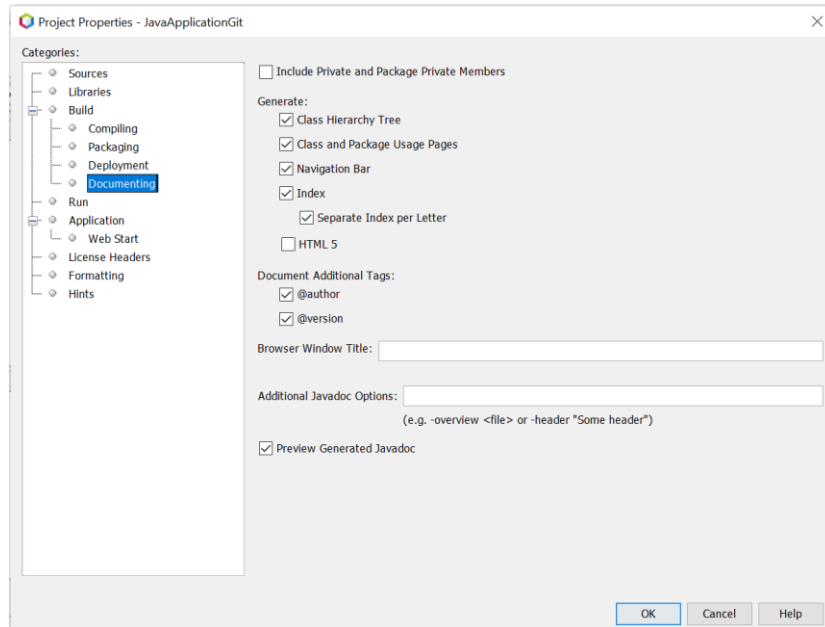
- ✓ Reportar uso de APIs deprecadas: permitir indicar que alguna API de las utilizadas está deprecada y ya no está recomendada su utilización.



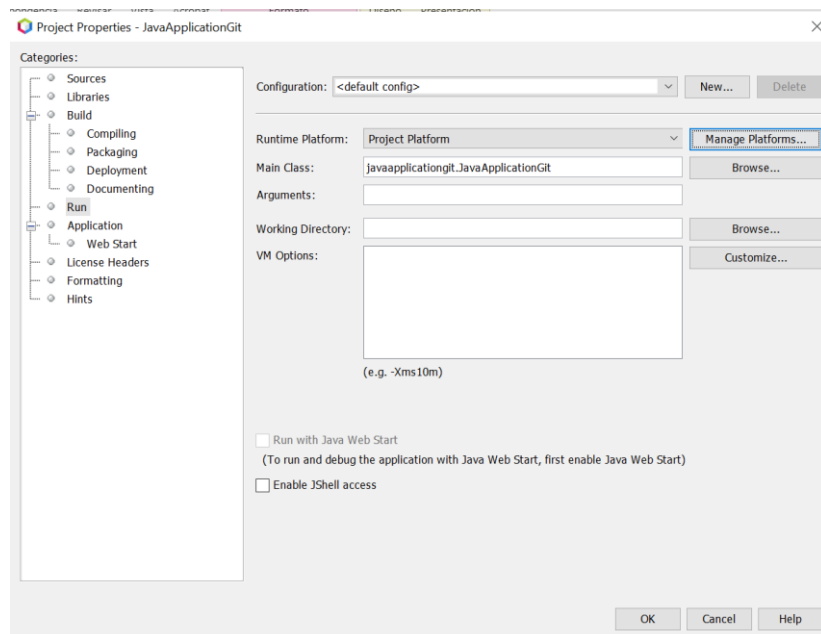
GENERACIÓN DE CÓDIGO - EMPAQUETANDO Las aplicaciones resultado de la compilación del código deben ser empaquetadas antes de su distribución, con objeto de tener un único archivo, generalmente comprimido, que contenga en su interior todos los archivos de instalación y configuración necesarios para que la aplicación pueda ser instalada y desarrollada con éxito por el usuario cliente.

Dentro de esta opción podemos modificar el lugar donde se generará el archivo resultante del empaquetado, así como si deseamos comprimirlo. También podemos elegir que el archivo empaquetado se construya tras la compilación, que es lo habitual (por eso esta opción aparece como predeterminada)

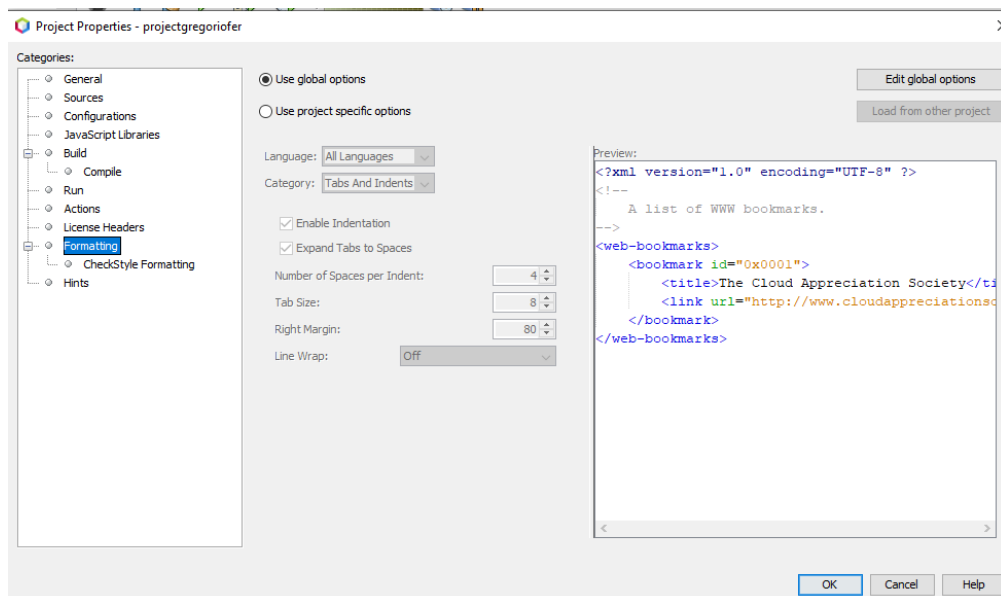
GENERACIÓN DE CÓDIGO - DOCUMENTANDO Como ya vimos, la documentación de aplicaciones es un aspecto clave que no debemos descuidar nunca. NetBeans nos ofrece una ventaja muy considerable al permitirnos obtener documentación de la fase de codificación de los programas de forma automática (**Javadoc**).



EJECUTANDO CÓDIGO Esta opción nos permite definir una nueva configuración de ejecución de código, elegir la clase principal, los paquetes de trabajo del proyecto y otras opciones de la máquina virtual.



FORMATO: Aquí podemos personalizar aspectos globales del formato del código fuente en la aplicación.



6. Gestión de Módulos

Con la plataforma dada por un entorno de desarrollo como NetBeans **podemos hacer uso de módulos y plugin para desarrollar aplicaciones**. En la página oficial de NetBeans encontramos una relación de módulos y plugin, **divididos en categorías**. Seleccionando la categoría Lenguajes de Programación, encontraremos aquellos módulos y plugin que nos permitan añadir nuevos lenguajes soportados por nuestro IDE.

Un **módulo** es un **componente software** que contiene clases de Java que pueden interactuar con las APIs del entorno de desarrollo y el **manifest file**, que es un archivo especial que lo identifica como módulo. Los módulos se pueden construir y desarrollar de forma independiente. Esto posibilita su reutilización y que las aplicaciones puedan ser construidas a través de la inserción de módulos con finalidades concretas. Por esta misma razón, una aplicación puede ser extendida mediante la adición de módulos nuevos que aumenten su funcionalidad.

Existen en la actualidad multitud de módulos y plugin disponibles para todas las versiones de los entornos de desarrollo más utilizados. En las secciones siguientes veremos dónde encontrar plugin y módulos para NetBeans que sean de algún interés para nosotros y las distintas formas de instalarlos en nuestro entorno. También aprenderemos a desinstalar o desactivar módulos y plugin cuando

preveamos que no los vamos a utilizar más y cómo podemos estar totalmente actualizados sin salir del espacio de nuestro entorno. Veremos las categorías de plugin disponibles, su funcionalidad, sus actualizaciones...

6.1. Añadir un nuevo módulo

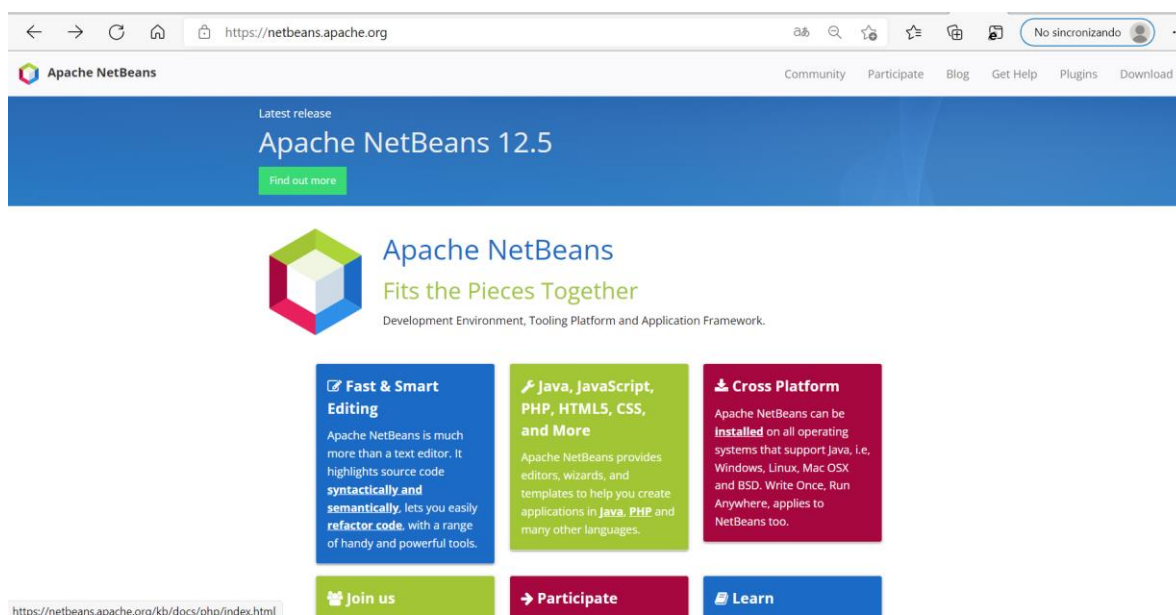
Añadir un módulo va a provocar dotar de mayor funcionalidad a nuestros proyectos desarrollados en NetBeans.

Para añadir un nuevo módulo tenemos varias opciones:

- a) Añadir algún módulo de los que NetBeans instala por defecto.
- b) Descargar un módulo desde algún sitio web permitido y añadirlo.
- c) Instalarlo on-line en el entorno de desarrollo.

Por supuesto, una cuarta posibilidad es crear el módulo nosotros mismos (aunque eso no lo veremos aquí). Sin embargo, lo **más usual** es **añadir** los módulos o **plugin** que realmente nos **interesan** desde la **web oficial** de NetBeans ([Apache NetBeans Plugin Portal](https://netbeans.apache.org)). El plugin se descarga en formato **.nbm** que es el propio de los módulos en NetBeans. Posteriormente, desde nuestro IDE, cargaremos e instalaremos esos plugin. A esta manera de añadir módulos se le conoce como **adición off-line**.

También es habitual **instalarlos on-line**, sin salir del IDE. La adición on-line requiere tener instalado el plugin *'Portal Update Center'* en NetBeans y consiste en instalar complementos desde nuestro mismo IDE, sin tener que descargarlos previamente. A modo de ejemplo, a continuación, se explican los pasos para añadir un módulo o plugin de forma on-line.



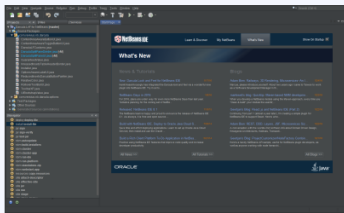
Sitio Oficial de Descarga de Plugins para instalación OFF-LINE

NetBeans NetBeans IDE NetBeans Platform Enterprise **Plugins** Docs & Support Community

HOME / Plugin Portal

Darcula LAF for NetBeans - plugin detail

A NetBeans Look And Feel plugin using Darcula of IntelliJ IDEA.



Versions available

NetBeans 8.2 NetBeans 8.1 NetBeans 8.0

[Download](#) Download size: 0.34 MB Last Update: 2017-07-31

This plugin is also available on the [NetBeans Plugin Portal Update Center](#). Use 'Tools > Plugins' action from NetBeans IDE main menu for convenient installation of this plugin

What's new in this version
 Thank you everybody! [markiewb](#), [granella](#) and [AlexFalappa](#) for their contributions and all others for issues/suggestions on GitHub. Please use and provide defects/suggestions.
 Sources available here: <https://github.com/Revivius/nb-darcula>.
 Plugin is compiled with Java 7 and should work with NetBeans on JDK7 without problems.

Change Log

[1.6] - 2017-07-31
 - Plugin now works on OSes Thanks to [Griffith](#) and [markiewb](#)

Plugin owner: [Revivius](#)
 Website: <https://github.com/Revivius/nb-darcula>
 Added: 2016-01-14

Ejemplo de Plugin *Darcula* (Modifica apariencia del IDE) para instalación OFF-LINE

Plugins

Updates (2) Available Plugins (55) Downloaded Installed (10) Settings

[Check for Newest](#)

Install	Name	Category	Source
<input checked="" type="checkbox"/>	Backlog	Base IDE	
<input type="checkbox"/>	GitHub Issues	Base IDE	
<input type="checkbox"/>	NetBeans CapsLock Notifier	Base IDE	
<input type="checkbox"/>	NetBeans: Statusline Clock	Base IDE	
<input type="checkbox"/>	NetBeans Use System Desktop	Base IDE	
<input type="checkbox"/>	NetBeans System Properties	Base IDE	
<input type="checkbox"/>	NetBeans Antr	com.github.mch...	
<input type="checkbox"/>	Netbeans Minimap	com.github.mch...	
<input type="checkbox"/>	SQL DAL Maker	Database	
<input type="checkbox"/>	MultiProperties	Data Files	
<input type="checkbox"/>	Color Codes Preview	Editing	
<input type="checkbox"/>	nb-noext-mime-resolver	Editing	
<input type="checkbox"/>	Change Line Endings on Save	Editing	
<input type="checkbox"/>	NB MindMap Editor	Editing	
<input type="checkbox"/>	NB CSV Editor	Editing	
<input type="checkbox"/>	Rainbow Braces	Editing	
<input type="checkbox"/>	BinEd - Binary/Hexadecimal Editor	Editing	
<input type="checkbox"/>	No Newline Resolver	Editing	
<input type="checkbox"/>	jvi for NB-7.0 Update Center	Editing	
<input type="checkbox"/>	NB-ChangeCase	Editing	
<input type="checkbox"/>	Quick Insert	Editing	

[Install](#) 1 plugin selected, 3MB

GitHub Issues

Community Contributed Plugin

Version: 0.5.1
Author: junichi11(Junichi Yamamoto)
Date: 9/09/19
Source: Plugin Portal
Homepage: <https://github.com/junichi11/netbeans-github-issues-plugin>

Plugin Description
 This plugin provides support for GitHub Issue Tracker.

Features

- Create a new issue
- Edit an issue
- Create queries
- Edit an issue comment
- Delete an issue comment
- Create a new pull request
- Change an existing issue to a pull request
- Search issues with issue number or keywords

[Close](#) [Help](#)

Cuadro de selección de Plugin para descarga en instalación ON-LINE.

NetBeans IDE Installer

Installation completed successfully
 Click Finish to quit the NetBeans IDE installer.

The NetBeans IDE Installer has successfully installed the following plugins:

Backlog

[Finish](#) [Help](#)

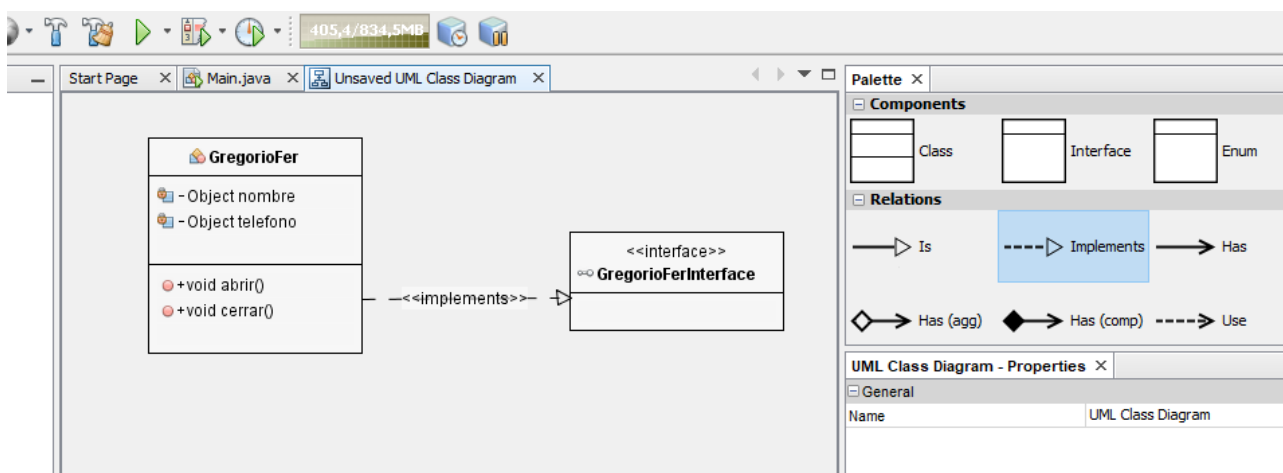
Finalización correcta de Instalación Online

Algunos Plugins para Apache Netbeans

Easyuml

UML significa lenguaje de modelado unificado, y se utiliza para representar relaciones entre clases como herencia, agregación, etc.

El complemento *easyUML* proporciona una herramienta para trabajar con UML como diagramas de clases. Es de gran importancia, ya que cuando se está aprendiendo programación orientada a objetos, se puede usar el plugin *easyUML* para hacer los diagramas de clase para evitar cualquier confusión, como cuáles son las relaciones entre clases, etc.



Gittoolbar

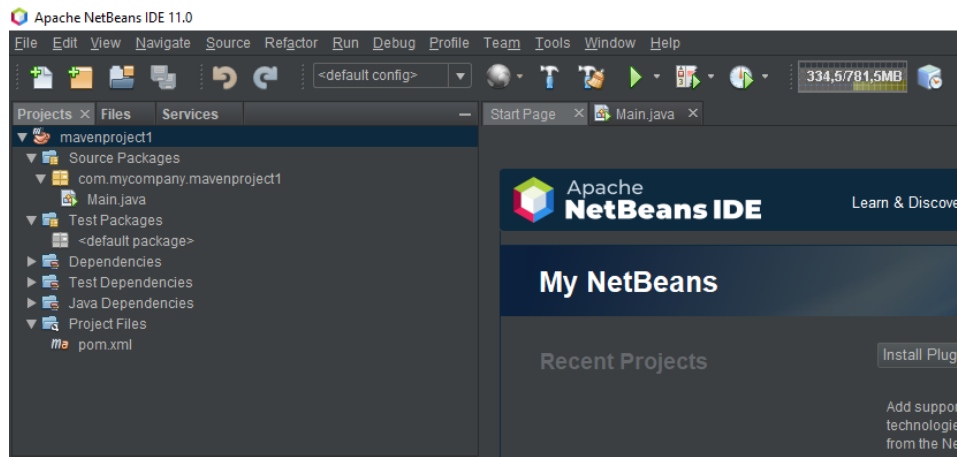
Este complemento agrega una barra de herramientas personalizada en la parte superior de Apache netbeans que contiene todos los comandos GIT comunes.

Esta barra de herramientas con comandos GIT ahorrará mucho tiempo al desarrollador en el momento de guardar cambios en el repositorio remoto.



Darcula LAF for netbeans

Este complemento le da un esquema de color más oscuro al editor de Apache Netbeans. Hace más fácil programar por períodos más largos de tiempo.



6.2. Eliminar un módulo existente

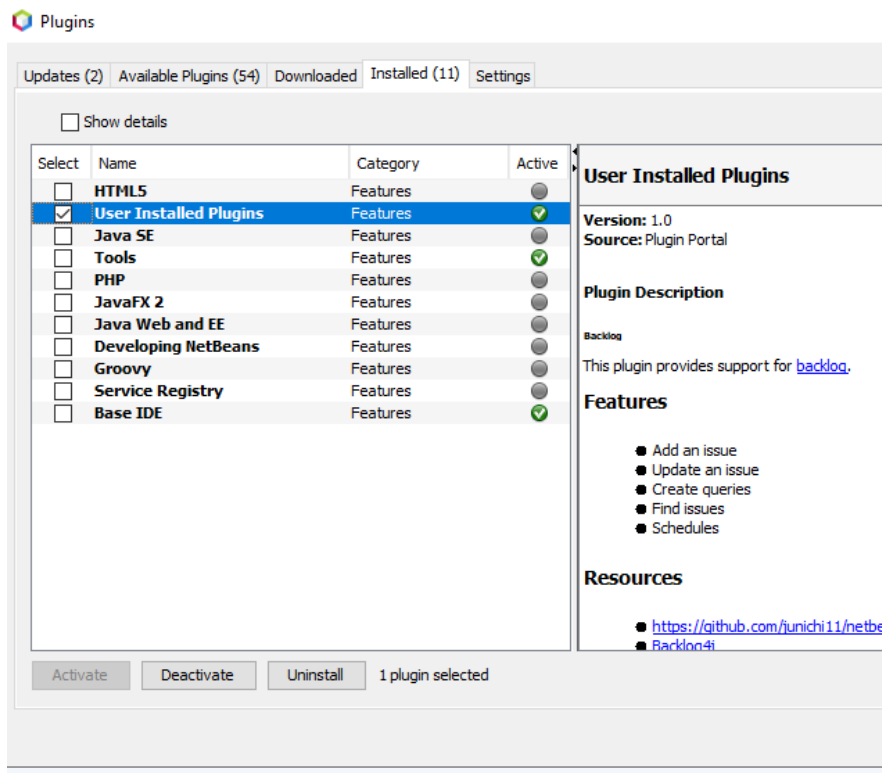
Cuando consideramos que algún módulo o plugin de los instalados no nos aporta ninguna utilidad, o bien que el objetivo para el cual se añadió ya ha finalizado, el módulo deja de tener sentido en nuestro entorno. Es entonces cuando nos planteamos eliminarlo.

Eliminar un módulo es una tarea trivial que requiere seguir los siguientes pasos:

1. Encontrar el módulo o plugin dentro de la lista de complementos instalados en el entorno.
2. A la hora de eliminarlo, tenemos dos opciones:
 - a) Desactivarlo: El módulo o plugin sigue instalado, pero en estado inactivo (no aparece en el entorno).
 - b) Desinstalarlo: El módulo o plugin se elimina físicamente del entorno de forma permanente.

Esta es la ventana, desde el gestor de complementos de NetBeans, que nos aparece cuando queremos eliminar un módulo del entorno.

Siempre nos pedirá elegir entre dos opciones: desactivar o desinstalar.



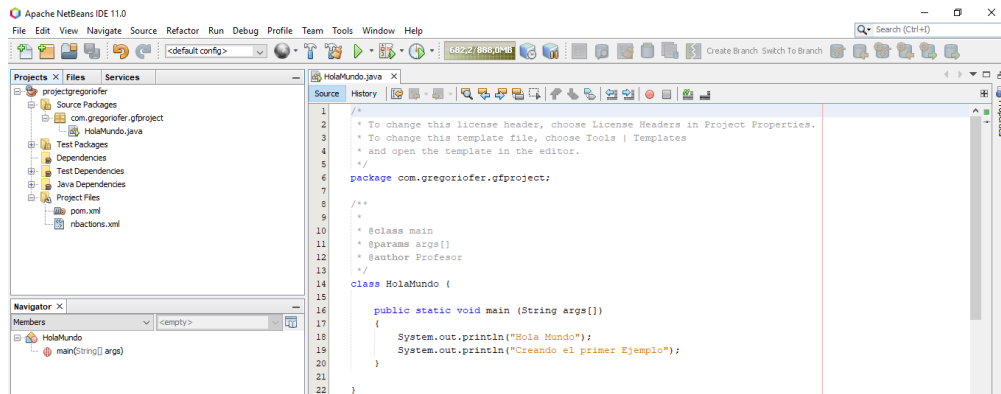
6.3. Funcionalidades

Los **módulos** y **plugin** disponibles para los entornos de desarrollo, en sus distintas versiones, tienen muchas y muy variadas funciones. Podemos **clasificar** las distintas **categorías** de **funcionalidades** de módulos y plugin en los **siguientes grupos**:

1. Construcción de código: facilitan la labor de programación.
2. Bases de datos: ofrecen nuevas funcionalidades para el mantenimiento de las aplicaciones.
3. Depuradores: hacen más eficiente la depuración de programas.
4. Aplicaciones: añaden nuevas aplicaciones que nos pueden ser útiles.
5. Edición: hacen que los editores sean más precisos y más cómodos para el programador.
6. Documentación de aplicaciones: para generar documentación de los proyectos en la manera deseada.
7. Interfaz gráfica de usuario: para mejorar la forma de presentación de diversos aspectos del entorno al usuario.
8. Lenguajes de programación y bibliotecas: para poder programar bajo un Lenguaje de Programación que, en principio, no soporte la plataforma.
9. Refactorización: hacer pequeños cambios en el código para aumentar su legibilidad, sin alterar su función.
10. Aplicaciones web: para introducir aplicaciones web integradas en el entorno.
11. Prueba: para incorporar utilidades de pruebas al software.

7. Uso básico de Entornos de Desarrollo

En la ventana principal del entorno de desarrollo de NetBeans nos encontramos con la siguiente ventana, que aparece cuando seleccionamos *Archivo -> Nuevo proyecto -> Java*:



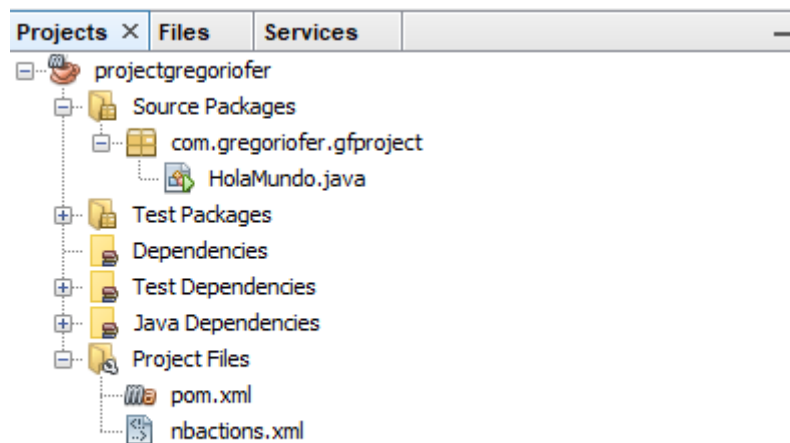
Como observamos, el espacio se divide en dos ventanas principales.

Ventana Izquierda: ventana de proyectos.

Aquí irá apareciendo la relación de proyectos, archivos, módulos o clases que vayamos abriendo durante la sesión.

Cada proyecto comprende una serie de archivos y bibliotecas que lo componen.

En este ejemplo, el principal archivo del proyecto Java es el llamado *Main.java*.



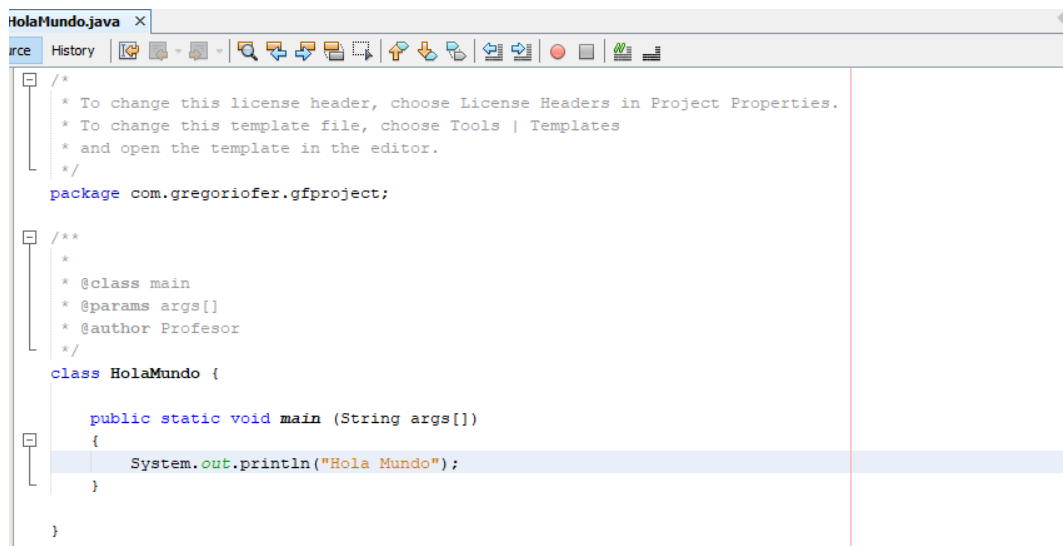
Ventana derecha: espacio de escritura del código de los proyectos.

Aquí aparece el esqueleto propio de un programa escrito en lenguaje Java.

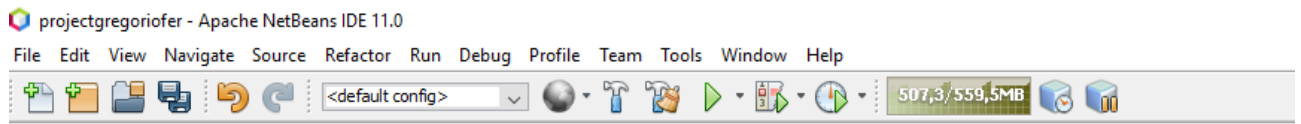
Se ha añadido el código:

```
System.out.println("Hola Mundo");
```

Y veremos su significado en las siguientes páginas. De momento, debemos saber que para escribir cualquier código, hay que hacerlo en esta ventana.

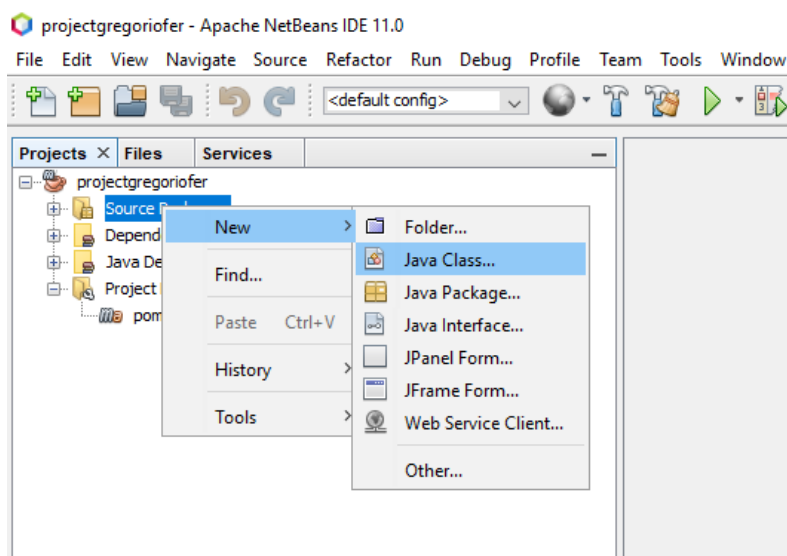


BARRA DE HERRAMIENTAS: Desde aquí podremos acceder a todas las opciones del IDE.

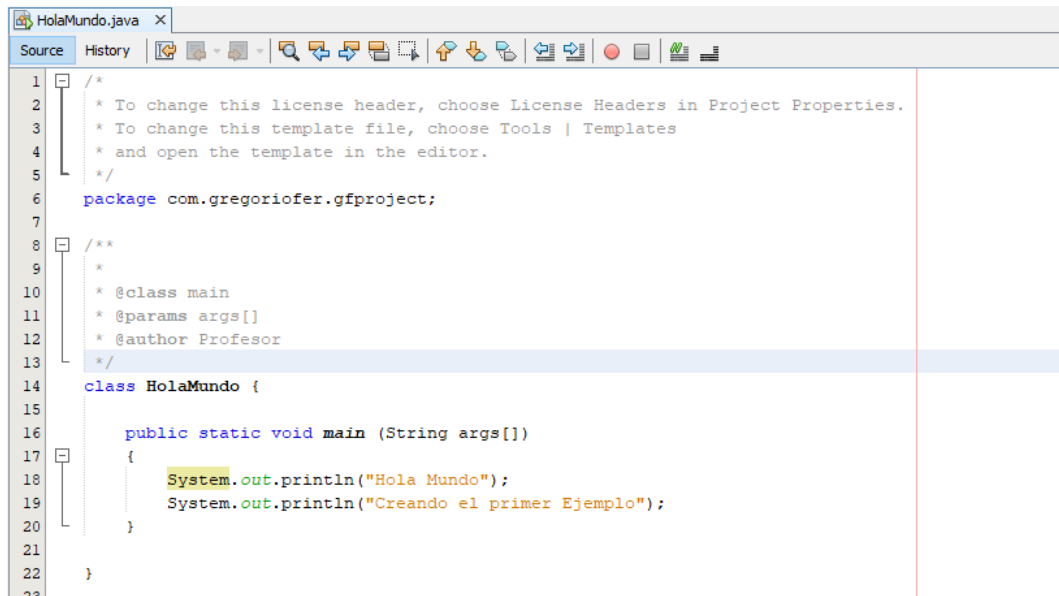


7.1. Edición de Programas

Vamos a ver un pequeño ejemplo de la edición del programa en Java *HolaMundo*, que únicamente mostrará una nueva línea por pantalla.



Creación de una nueva clase



```
1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6  package com.gregoriofer.gfproject;
7
8  /**
9   *
10   * @class main
11   * @params args[]
12   * @author Profesor
13   */
14  class HolaMundo {
15
16      public static void main (String args[])
17      {
18          System.out.println("Hola Mundo");
19          System.out.println("Creando el primer Ejemplo");
20      }
21  }
22
23  */
```

En este sencillo ejemplo se ve una modificación de las líneas de código en la ventana de codificación del archivo HolaMundo.java del proyecto ejemplo que acabamos de crear.

Las dos líneas que aparecen resaltadas se han escrito sobre la ventana y, tal y como significan en lenguaje Java, su ejecución implicará que sendos mensajes encerrados entre comillas y entre paréntesis saldrán impresos por pantalla.

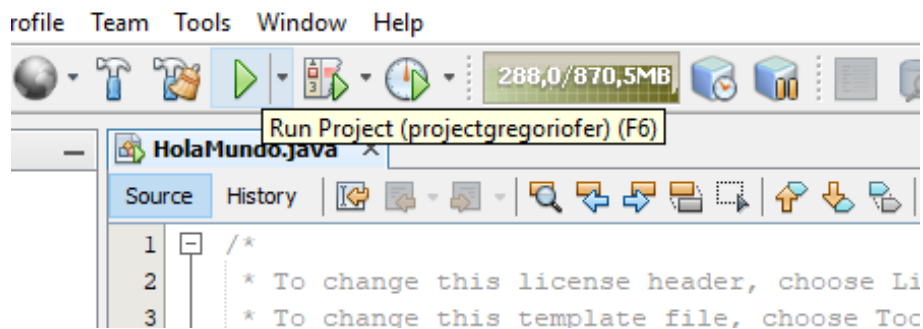
No hay que decir que la programación en Java no es objeto del presente módulo, pero puedes probar con algunos ejemplos en Java que tengas de otros módulos.

Mientras escribimos en el editor de textos nos percatamos de varias características de NetBeans que ya hemos señalado en páginas anteriores:

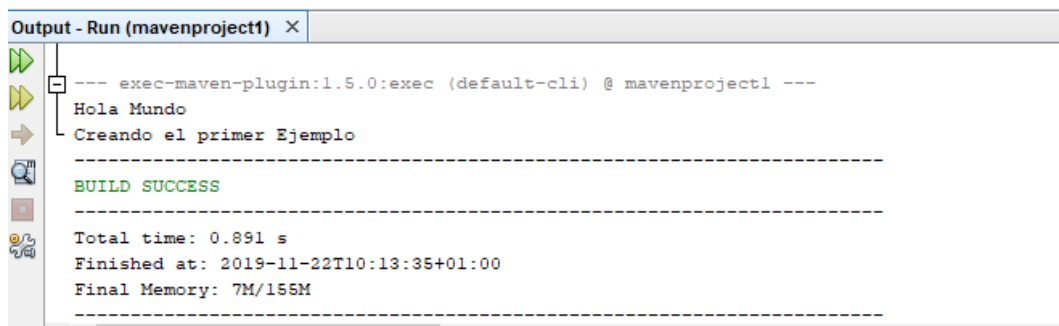
- ✓ Autocompletado de código.
- ✓ Coloración de comandos.
- ✓ Subrayado en rojo cuando hay algún error y posibilidad de depuración y corrección de forma visual, mediante un pequeño icono que aparece a la izquierda de la línea defectuosa.

7.2. Generación de Ejecutables

Una vez tenemos el código plasmado en la ventana de comandos y libre de errores de sintaxis, los siguientes pasos son: compilación, depuración, ejecución.

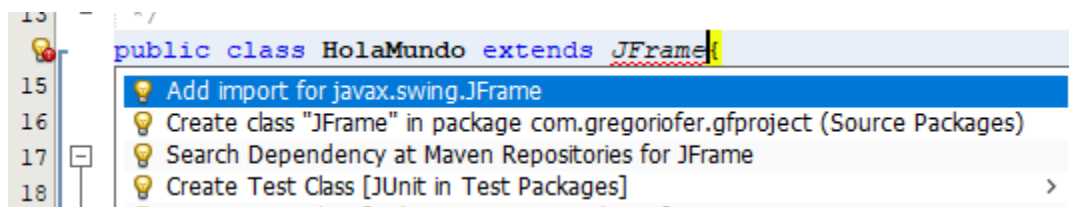


Al ejecutar el ejemplo anterior, el resultado es:



Trabajando con JFrame

Vamos a introducir la funcionalidad de JFrame dentro de nuestra clase, haciendo que herede de ella. Esta línea nos va a servir para adentrarnos en una de las utilidades más importantes de Apache Netbeans. NetBeans entiende esta orden como un error (aparece subrayada en una línea roja ondulada y con un pequeño icono al lado izquierdo, porque no encuentra la dependencia) Lo que haremos será hacer doble click para que la resuelva y añada el correspondiente 'import'.



Opciones de Resolución del error


```

package com.gregoriofer.gfproyect;

import javax.swing.JFrame;
import javax.swing.JLabel;

/**
 *
 * @author rhtuf
 */
public class HolaMundo extends JFrame {

```

Vemos cómo se ha añadido la librería de JFrame

En el caso de que se nos vuelvan a subrayar líneas en rojo, actuamos igual que en el caso anterior y vamos viendo las sugerencias que nos dan para corregir.

Añadimos el código para mostrar dentro de una etiqueta en nuestro mensaje de Hola Mundo:

```

/**
 *
 * @author rhtuf
 */
public class HolaMundo extends JFrame {

    public HolaMundo() {

        JLabel jl = new JLabel( text: "Hola mundo.Creando mi primer JFrame");
        add( comp:jl);
        this.setSize( width:300, height:300);
        this.setTitle( title: "Gregorio Fernández");
        this.setVisible( b:true);
        this.setDefaultCloseOperation( operation: JFrame.EXIT_ON_CLOSE);

    }

    public static void main(String[] args) {
        System.out.println( x: "Hola Mundo");
        HolaMundo h = new HolaMundo();
    }

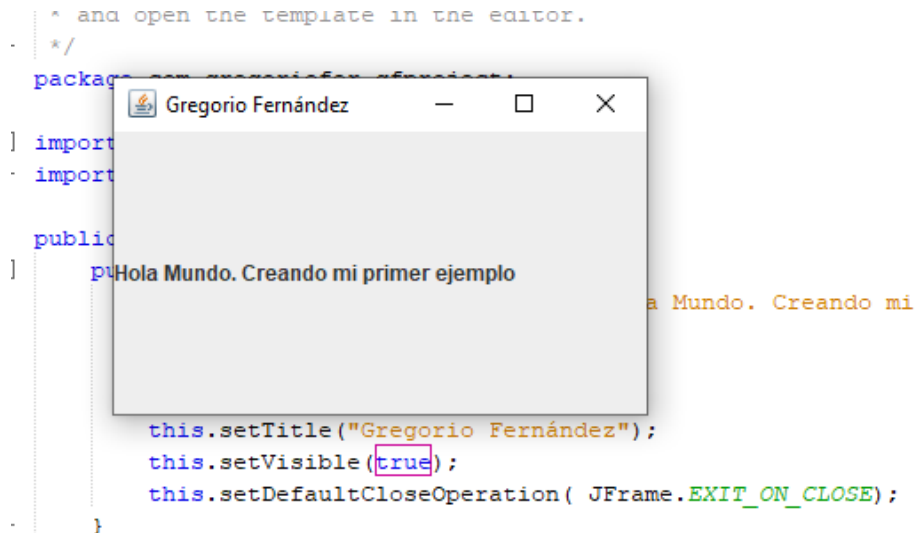
}

```

Tenemos el programa escrito en el editor libre de errores sintácticos. ¿Cómo convertir ese programa en ejecutable? Cabe destacar que, por la sencillez y pequeñez del programa, la ejecución del mismo podría ser directa sin ningún problema.

Ejecutar → En la barra de herramientas o bien mediante el icono de acceso directo en la parte superior de la ventana de edición de código.

Resultado de la Ejecución



8. Actualización y Mantenimiento de Entornos de Desarrollo

El mantenimiento del entorno de desarrollo es una tarea fundamental que requiere tener todos sus componentes periódicamente actualizados. También es de vital importancia realizar copias de seguridad sobre las bases de datos de nuestros proyectos por si ocurriera algún error o proceso defectuoso poder restaurarlos. El mantenimiento y las actualizaciones se hacen de forma on-line. En NetBeans contamos con el complemento llamado **Auto Update Services**.

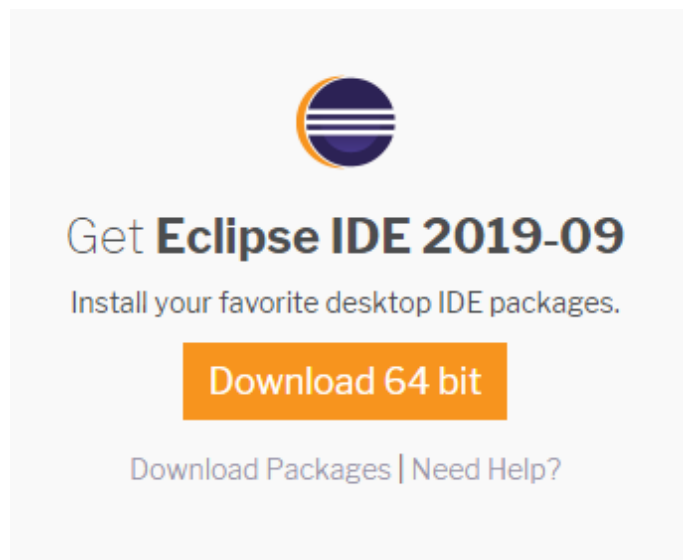
9. Alternativas al IDE Apache Netbeans

Como alternativa al IDE Apache Netbeans, vamos a ver la instalación y ejecución un fichero dentro del IDE Eclipse.

9.1. Instalación de Eclipse

De la misma forma que hicimos para Apache Netbeans, Una vez instalado el JDK, procedemos a la web oficial de *Eclipse Foundation* para la descarga del fichero instalable (<https://www.eclipse.org/>)

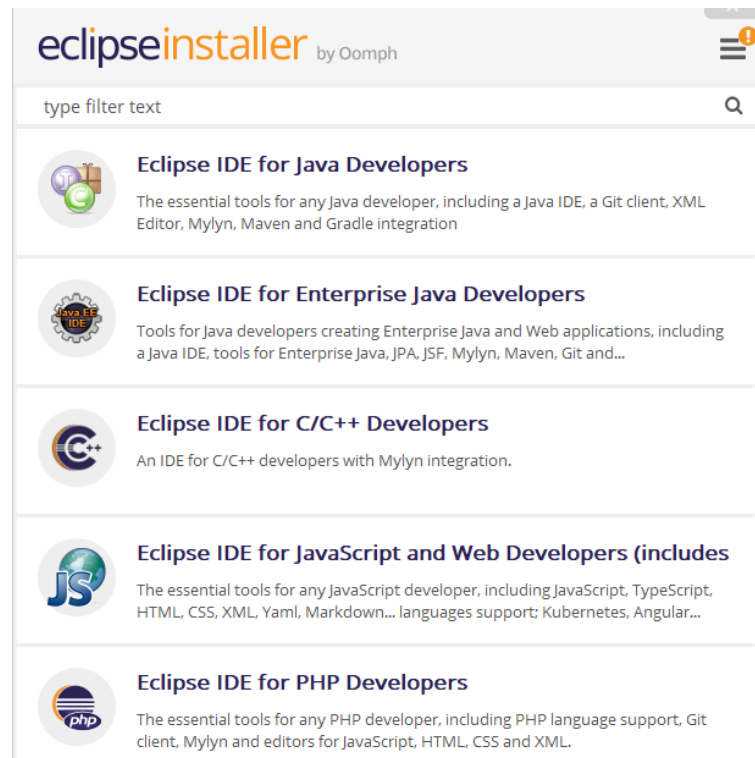




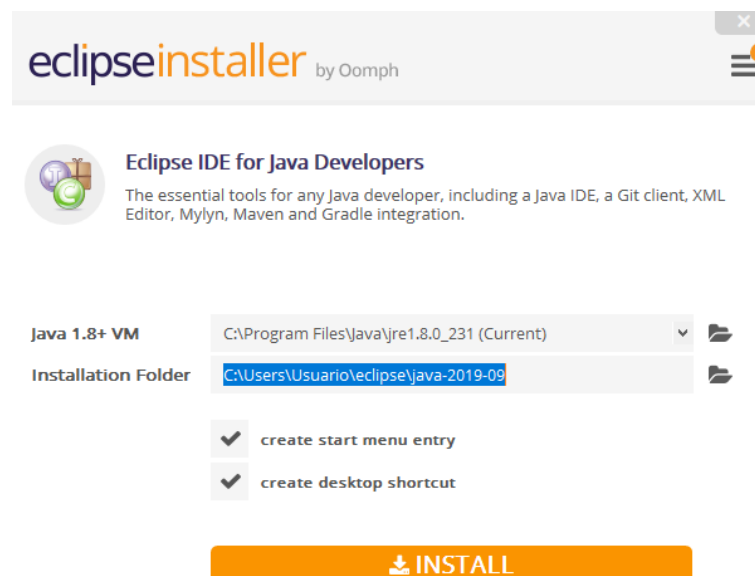
Una vez descargado, procedemos a su instalación:



Elegimos la versión del IDE que más se ajuste a nuestras necesidades, esto dependerá del proyecto que vayamos a desarrollar y del lenguaje en el que se desarrollará. En nuestro caso elegimos *Eclipse IDE for Java Developers*.



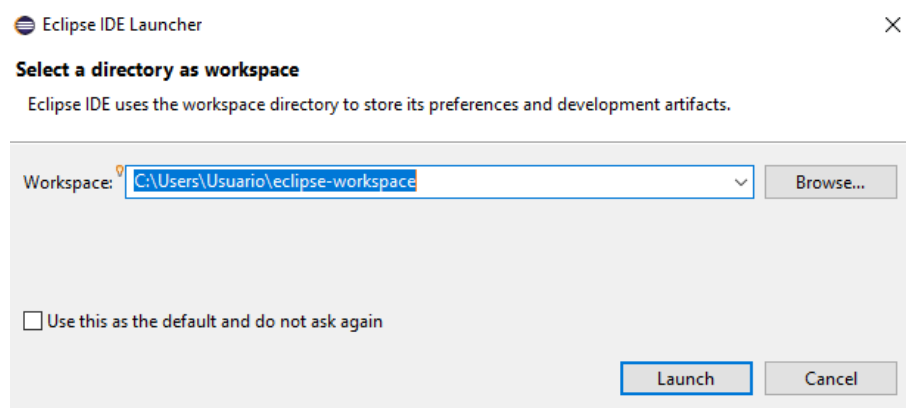
Elegimos el directorio de instalación y la versión de JDK sobre la que queremos que se ejecute nuestro IDE Eclipse.



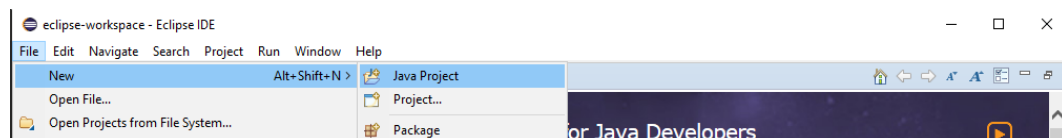
Una vez que se haya instalado, procedemos a abrir el IDE Eclipse.



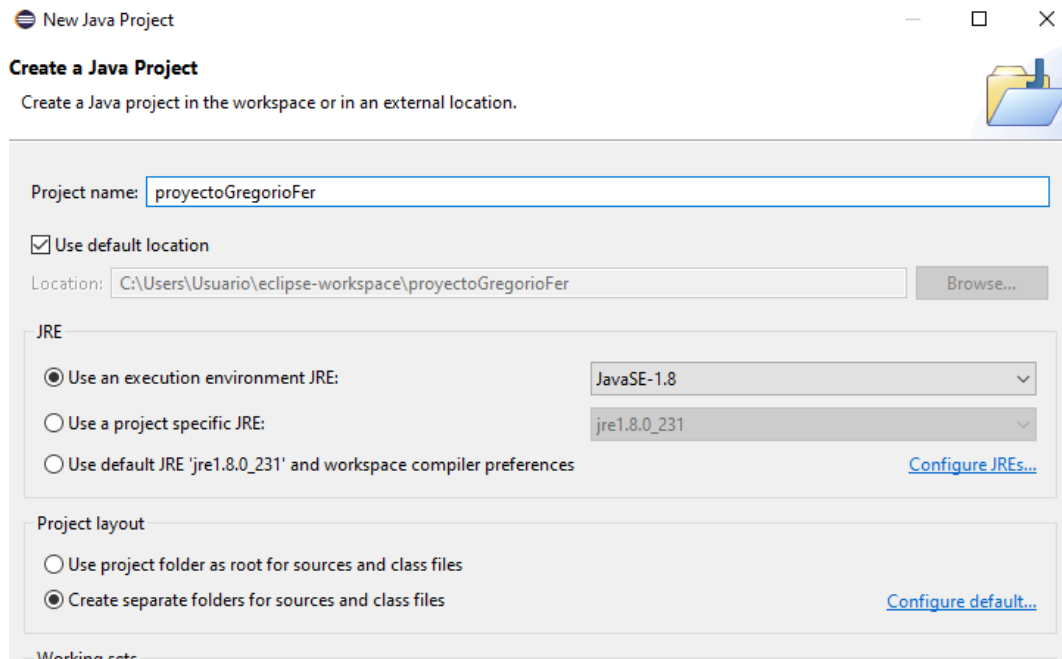
Seleccionamos nuestro Workspace, que será donde se guardarán los proyectos que creemos a partir de este momento.



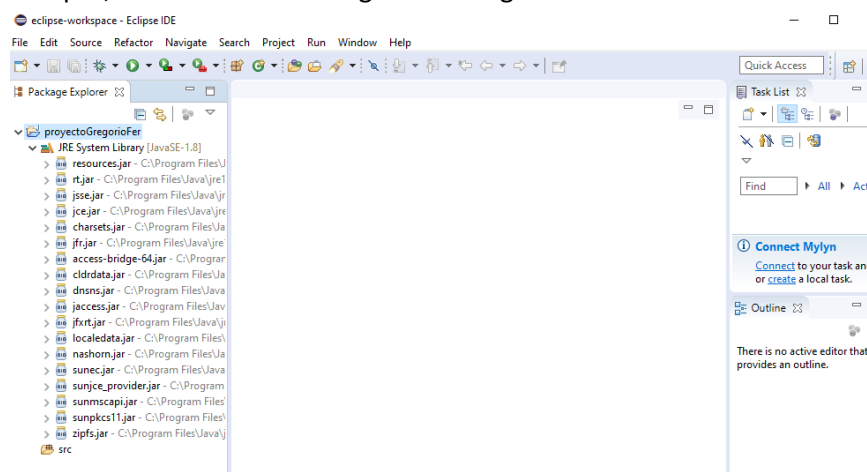
Procedemos a crear un nuevo proyecto, desde el Menú *File ->New -> Java Project*



Seleccionamos el nombre del proyecto, el *Workspace* donde se guardará y la versión del JRE que utilizará.

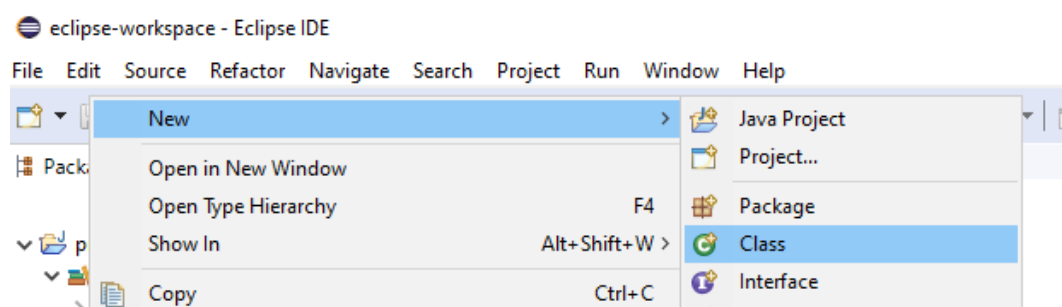


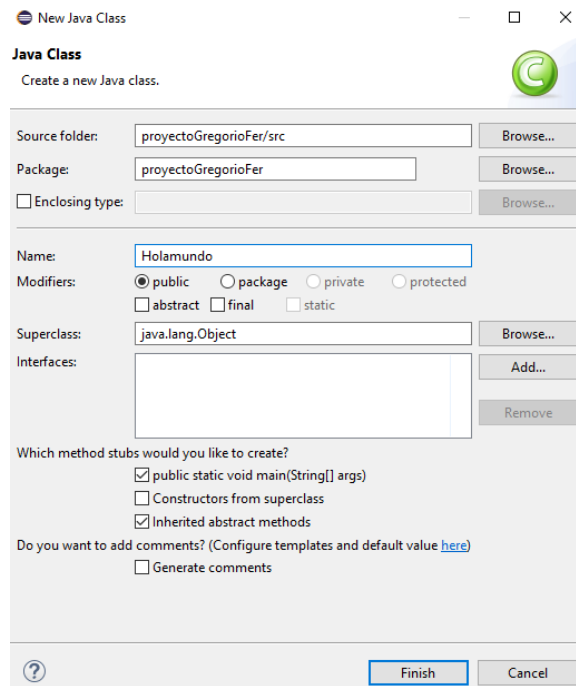
Y una vez aceptado y cerrado el menú, ya tendremos completamente operativo nuestro proyecto dentro del IDE Eclipse, como vemos en la siguiente imagen.



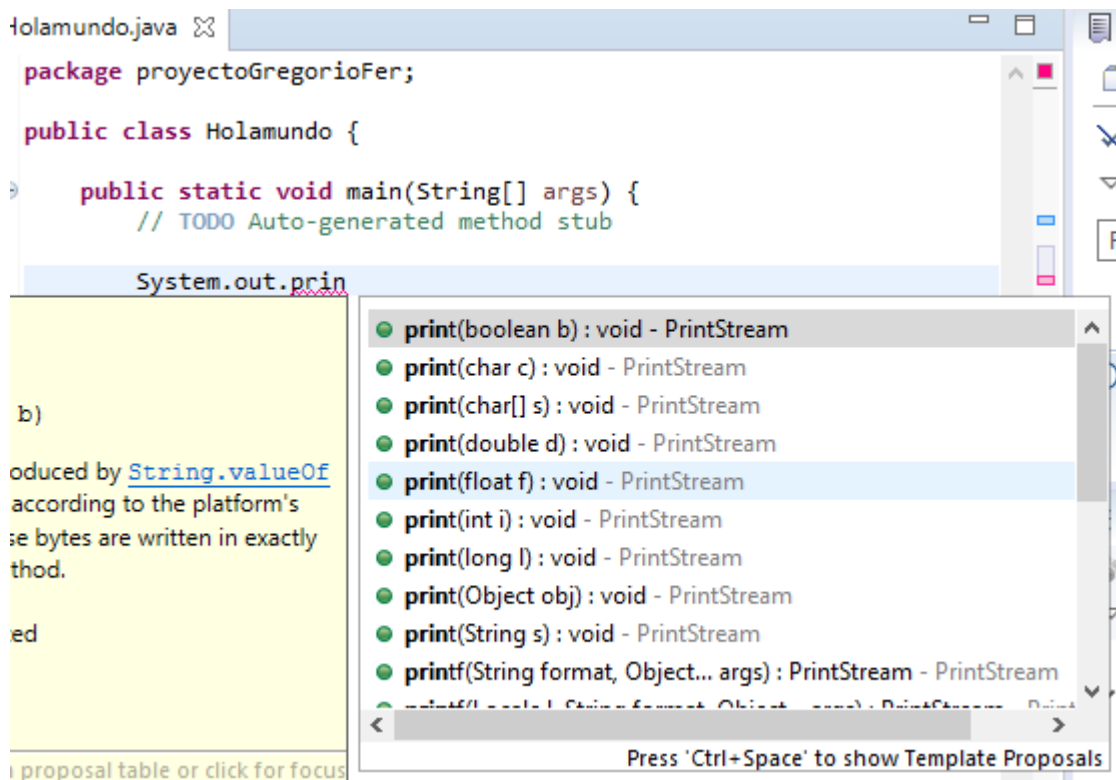
9.2. Ejecución del Primer programa ('Hola Mundo')

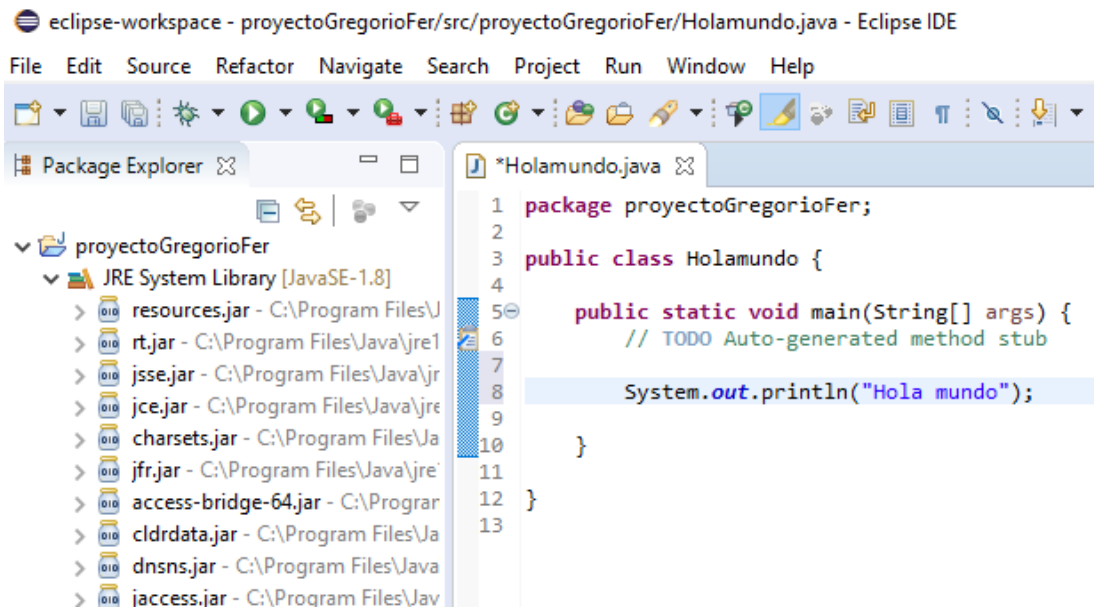
De la misma forma que hicimos para *Apache NetBeans*, añadimos una nueva clase a nuestro proyecto:



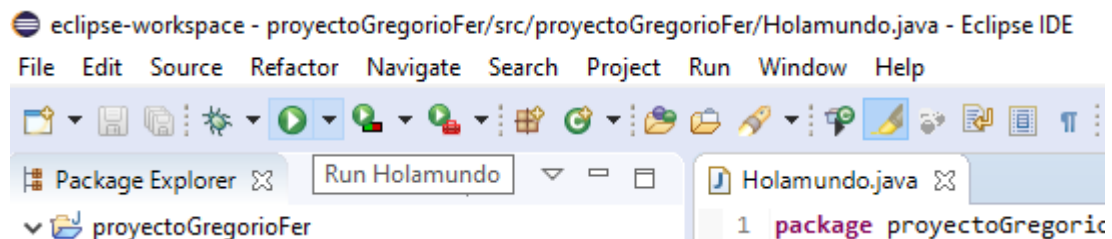


Como podemos observar, Eclipse también tiene la opción de autocompletado.

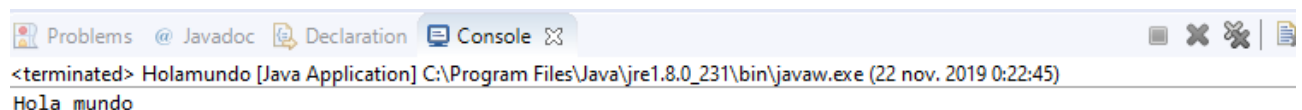




Para poder ejecutar nuestro proyecto, hacemos click en la flecha verde superior para ejecutar nuestra clase.



Y finalmente vemos el resultado el de la ejecución en la consola inferior del *IDE Eclipse*.

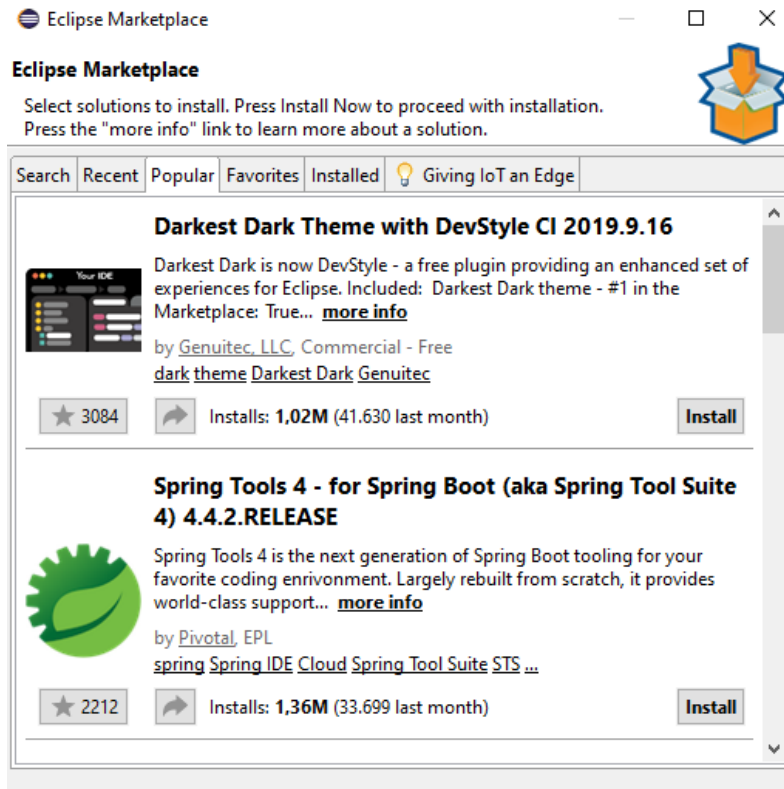


9.3. Instalación de Plugins en Eclipse

La instalación de Plugins dentro del *IDE Eclipse*, es muy similar a la instalación de Plugin en *Apache Netbeans*. Dentro de la pantalla de bienvenida de Eclipse, seleccionaremos '*Eclipse Marketplace*', para acceder al buscador de Plugin.



Eclipse Marketplace
Install Eclipse extensions and solutions



El proceso de instalación de estos plugin dentro de *Eclipse*, se iniciará pulsando el botón '*Install*' que vemos a la derecha de la captura. Después de la instalación, será necesario reiniciar nuestro IDE para que se active nuestro nuevo plugin.