



UT 2. Instalación y uso de entornos de desarrollo

ENTORNO DE DESARROLLO

I.E.S. Luis Vives - Desarrollo de Aplicaciones Web

Introducción

IDE

Los entornos de desarrollo son las herramientas con las cuales los programadores crean aplicaciones.

Es cierto que pueden programarse con un editor y un compilador (a veces, con un depurador), pero, en entornos profesionales, casi siempre se utiliza un IDE.

Un IDE consta de las siguientes herramientas:

1. Editor. Generalmente, se utilizan editores que colorean la sintaxis para ayudar al programador a comprender mejor el programa y detectar los errores más fácilmente.
2. Compilador o intérprete. Dependiendo del tipo de lenguaje utilizado, se necesitará para ejecución el intérprete o el compilador para generar código ejecutable.
3. Depurador (intérprete). Un buen depurador siempre tiene un intérprete detrás para ir ejecutando órdenes paso a paso, inspeccionar el valor de variables, etc.
4. Constructor de interfaces gráficos. Con él, el desarrollador podrá crear ventanas, botones, campos de texto, literales, pestañas, tablas, etc. Tiene todos los componentes que pueden encontrarse en una interfaz.

Historia

IDE

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 utilizó 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 y depuración; es perfectamente compatible con los IDE actuales.

A nivel popular, el primer IDE puede considerarse que fue el IDE llamado **Maestro I**. Nació a principios de los 70 y fue instalado por unos 22.000 programadores en todo el mundo. Lideró este campo durante los años 70 y 80. Uno de los últimos Maestro I puede ser encontrado en el Museo de Tecnología e Informática en Arlington.

Turbo Pascal. Lo lanzó la empresa Borland en el año 1983 y fue el IDE más potente de su época. Al principio, funcionaba en MS-DOS, CP/M y CP/M 86 y Macintosh, aunque posteriormente se creó una versión para Windows que tuvo mucho éxito.

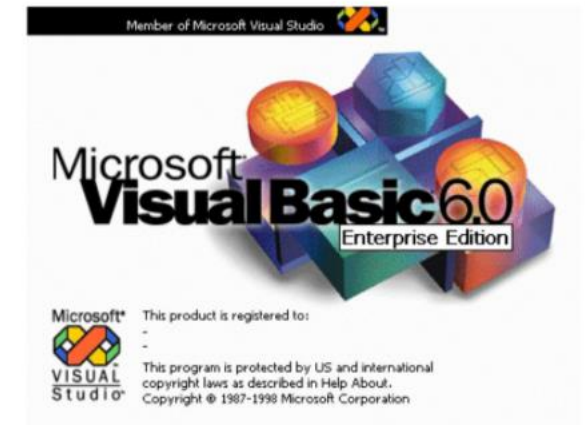
Se lanzaron siete versiones y, en las últimas, podía utilizarse el ratón. Soportaba múltiples archivos en el mismo editor (diferentes ventanas) y podía programarse orientado a objetos. También poseía una herramienta llamada Turbo Profiler que permitía optimizar el código.



Fue una revolución en su época. La rapidez de compilación era asombrosa. De hecho, los compiladores actuales son más lentos. Tras el éxito de esta herramienta, Borland creó nuevas herramientas, como Delphi, basadas en el mismo lenguaje de programación: **Pascal**

Visual Basic 6 fue uno de los IDE más utilizados en su época, si no el que más. Este nuevo tipo de herramientas creó el paradigma de desarrollo RAD, acrónimo del inglés rapid application development (desarrollo rápido de aplicaciones).

Un paradigma en el que primero se desarrollaban de una manera rápida las interfaces y se consensuaban con el usuario. Cuando se tenía el visto bueno, empezaban a crearse la base de datos y el código. Fue un cambio en el modelo de programar.

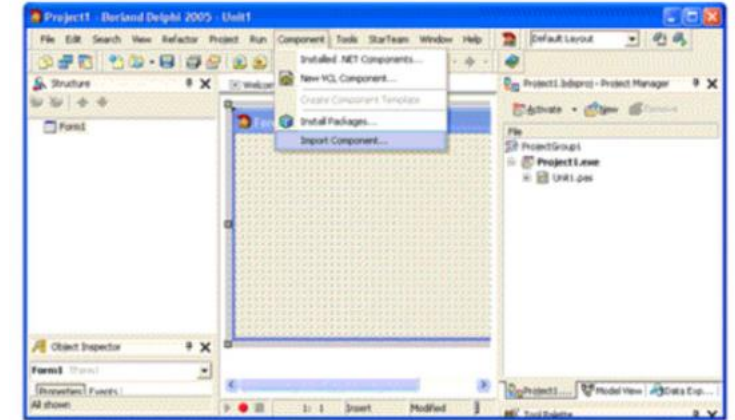


Los programadores creaban las interfaces a partir de una serie de componentes que ofrecía la propia herramienta. También podían utilizarse componentes de terceros, con lo cual se ganaba en funcionalidad y potencia. El acceso a las bases de datos se realizaba utilizando DAO, RDO o ActiveX Data Objects, este último más rápido y más optimizado. Visual Basic se utiliza en la actualidad gracias a que las macros realizadas en Office utilizan un dialecto suyo: Visual Basic for applications (VBA). Las macros son una herramienta muy potente, dado que combinan las características de Office con la potencia de todo un lenguaje de programación orientado a objetos.

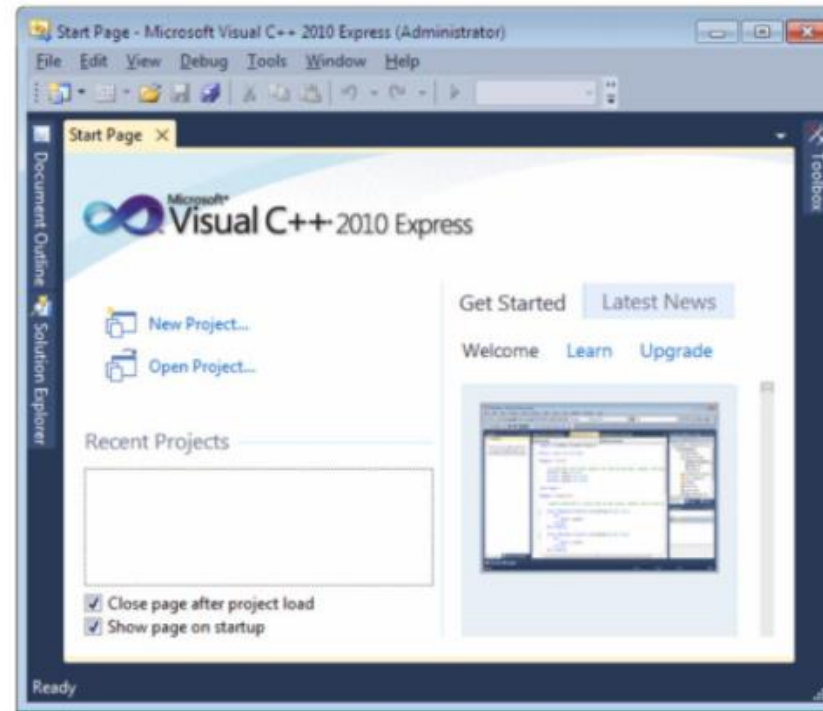
Delphi Turbo Pascal fue un líder en su época y otro grande de la informática (Microsoft) sacó al mercado Visual Basic. Visual Basic era un IDE para Windows que hizo que Borland sacara algo más tarde al mercado Delphi, que fue una evolución del Turbo Pascal hacia el sistema Windows

igual que Builder C++ fue la evolución del Turbo C. Además de Delphi, Borland también sacó al mercado el JBuilder. Un IDE de Java que tenía la ventaja de estar disponible también en Linux.

Delphi también tuvo su hermano de Linux llamado Kylix, que, desafortunadamente, se abandonó tras la versión 3.0, pero tenía la ventaja de que cualquier proyecto realizado en Windows podía recompilarse en Linux, y viceversa (siempre que se utilizasen los controles estándar).



Visual C++ es un IDE para programar en C y C++. Su potencia radica en que incluye las bibliotecas de Windows, las Microsoft Foundation classes (MFC) y el framework .NET. Es un IDE pesado, pero a la vez potente, puesto que, además de las bibliotecas propias, pueden añadirse otras nuevas como DirectX, wxWidgets o SDL. Al igual que Java, .NET ha incluido una herramienta bastante útil para autogestionar la memoria: el recolector de basura o garbage collector.



Entorno de desarrollo actuales



Xcode es la herramienta para realizar aplicaciones (app) para dispositivos Apple. Con esta herramienta, podrán realizarse aplicaciones nativas para iOS y OS X. Si desea descargarse una versión antes de que se encuentre disponible para todo el mundo, hay que hacerse desarrollador de Apple. Actualmente, no cuesta nada darse de alta como desarrollador, es gratuito, lo que cuesta es subir una aplicación a la App Store (la suscripción es de unos dólares al año y pueden subirse todas las aplicaciones que se desee).

Con las nuevas versiones, ya puede programarse en Swift, mientras que, con las versiones anteriores, solamente puede programarse con Objective C. Objective C es un lenguaje parecido a Java/ C/C++, pero con una sintaxis algo diferente. Muy potente y orientado a objetos.



NetBeans está escrita en Java, lo que la convierte en una plataforma disponible para un gran número de sistemas operativos (Windows, Linux o Mac OS X). Se creó para desarrollar aplicaciones en Java, pero también puede

programarse con ella en Python, PHP, HTML5 y C/C++. Es open source lo que hace que muchos programadores se decanten por este IDE. De hecho, cuando sale una nueva versión al mercado, suele estar bastante probada.

Se basa en la modularidad. Todas las funciones las realizan módulos, los cuales pueden ir añadiéndose según necesidades del programador. De hecho, cuando se descarga, tiene todos los módulos de Java incluidos por defecto. Muchas herramientas están basadas en NetBeans como Sun Studio, Sun Java Studio Creator y otras más. Contiene una herramienta para crear interfaces de usuario (llamada al comienzo Matisse). Esta herramienta permite crear aplicaciones basadas en la librería Swing. En el editor, puede programarse también en JavaScript, Ajax y CSS.



Es un IDE de código abierto. Al contrario que otros clientes livianos, es una plataforma potente con un buen editor, depurador y compilador (el ECJ). El JDT (Java development toolkit) es de los mejores que existen en el mercado y

tiene detrás una gran comunidad de usuarios que van añadiendo mejoras al software. Fue desarrollado por IBM como evolución de su VisualAge, pero ahora lo mantiene la fundación Eclipse, que es independiente y sin ánimo de lucro. Tenía licencia CPL (common public license), pero luego la fundación cambió dicha licencia por una EPL (Eclipse public license).



IntelliJ IDEA es un entorno de desarrollo integrado para el desarrollo de programas informáticos. Es desarrollado por JetBrains, y está disponible en dos ediciones: edición para la comunidad y edición comercial. La primera versión de IntelliJ la IDEA fue publicada en enero de 2001, y en aquel momento fue uno de los primeros IDE Java disponibles con navegación avanzada de código y capacidades de refactorización de código integrado.

En un informe de Infoworld en 2010, IntelliJ recibió la puntuación más alta entre las cuatro mejores herramientas de programación de Java: Eclipse, IntelliJ IDEA, NetBeans y Oracle JDeveloper.

En diciembre de 2014, Google anunció la versión 1.0 de Android Studio, un IDE de código abierto para aplicaciones de Android basado en el código abierto de la edición comunitaria de IntelliJ IDEA. Otros entornos de desarrollo se basaron en IntelliJ incluidos AppCode, PhpStorm, PyCharm, RubyMine, WebStorm, y MPS

[User interface](#) | [IntelliJ IDEA Documentation \(jetbrains.com\)](#)

El I.E.S. Luis Vives es centro con licencia para los alumnos y profesores. La licencia es por un año y es para la versión ultimate.

Además se están tramitando otras muchas licencias como la de Tabnine y OpenWebinars. Debéis indicar el email del centro (el que acaba en luisvives.org)

A continuación se os deja un enlace para apuntaros y obtener la licencia
<https://forms.office.com/r/qmKR8c3cXy>



Funciones

IDE

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

- Editor de código: coloración de la sintaxis.
- Auto-completado 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 autore-cuperación 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 plugins.
- Administración de las interfaces de usuario (menús y barras de herramientas).
- Administración de las configuraciones del usuario.

IDEs Libres y propietarios

Diferencias entre software libre o propietario.

Software Propietario. Encontramos otros nombres para el software propietario, como software privativo o software de código cerrado. Este tipo de software limita las posibilidades del usuario a modificarlo e incluso en su uso.

Software Libre. En contraposición al anterior, es aquel que se distribuye libremente. También es conocido como software de código abierto. En él, el usuario tiene plena libertad a la hora de usar el software, distribuirlo y/o modificarlo, con lo que se consigue un mayor desarrollo en las mejoras del primitivo. Cuando hablamos de software libre hacemos referencia a las tres libertades que este posee, que son:

- Libertad de uso.
- Libertad de distribución.
- Libertad de adaptación y mejora.

El término **libre no es sinónimo de gratuito.**

Existen muchos IDEs, dependiendo de la popularidad del lenguaje, habrá más o menos opciones. Podemos diferenciar IDEs libres

Entornos Integrados Libres. Normalmente gratuitos para desarrollar

Tipos de entornos de desarrollo libres más relevantes en la actualidad.

IDE	Lenguajes que soporta	Sistema Operativo
NetBeans.	C/C++, Java, JavaScript, PHP...	Windows, Linux, Mac OS X.
Eclipse	C/C++, Java, JavaScript, PHP...	Windows, Linux, Mac OS X.
Geany.	C/C++, Java, JavaScript, PHP...	Windows, Linux, Mac OS X.

Existen muchos IDEs, dependiendo de la popularidad del lenguaje, habrá más o menos opciones. Podemos diferenciar IDEs libres

Entornos Integrados Propietario. Normalmente de pago, aunque suelen tener algún tipo de versión reducida y gratuita.

Tipos de entornos de desarrollo propietario más relevantes en la actualidad.

IDE	Lenguajes que soporta	Sistema Operativo
Microsoft Visual Studio.	Basic, C/C++, C#...	Windows.
JBuilder	Java.	Windows, Linux, Mac OS X.
Xcode.	Swift, C/C++, Objective C, Java...	Mac OS X.
IntelliJ	Java, Kotlin, Groovy...	Windows, Linux, Mac OS X.

Estructura

IDE

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: 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 plugins, aumentando las opciones de nuestros programas

Instalación

IDE PARA JAVA

Para instalar correctamente el entorno de programación Eclipse, vamos a instalar previamente un JDK de java.



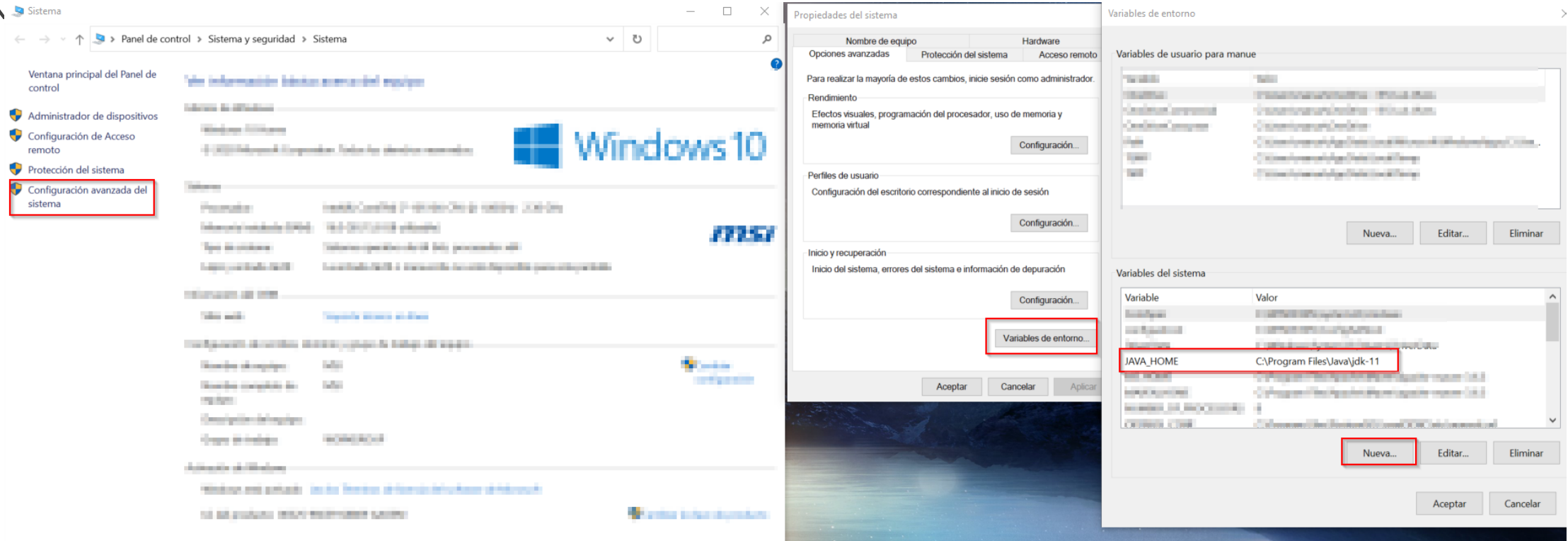
Para poder programar en Java, antes de nada es imprescindible que el Java Development Kit o JDK que, entre otras muchas cosas, incluye el compilador de Java, el depurador, las bibliotecas de servicios y la JVM (Java Virtual Machine) que permitirán convertir el código fuente en bytecode y ejecutarlo.

El procedimiento a seguir para instalarlo es prácticamente el mismo con independencia del sistema operativo en que estemos trabajando, si bien hay métodos alternativos en algunos casos como ocurre con GNU/Linux.

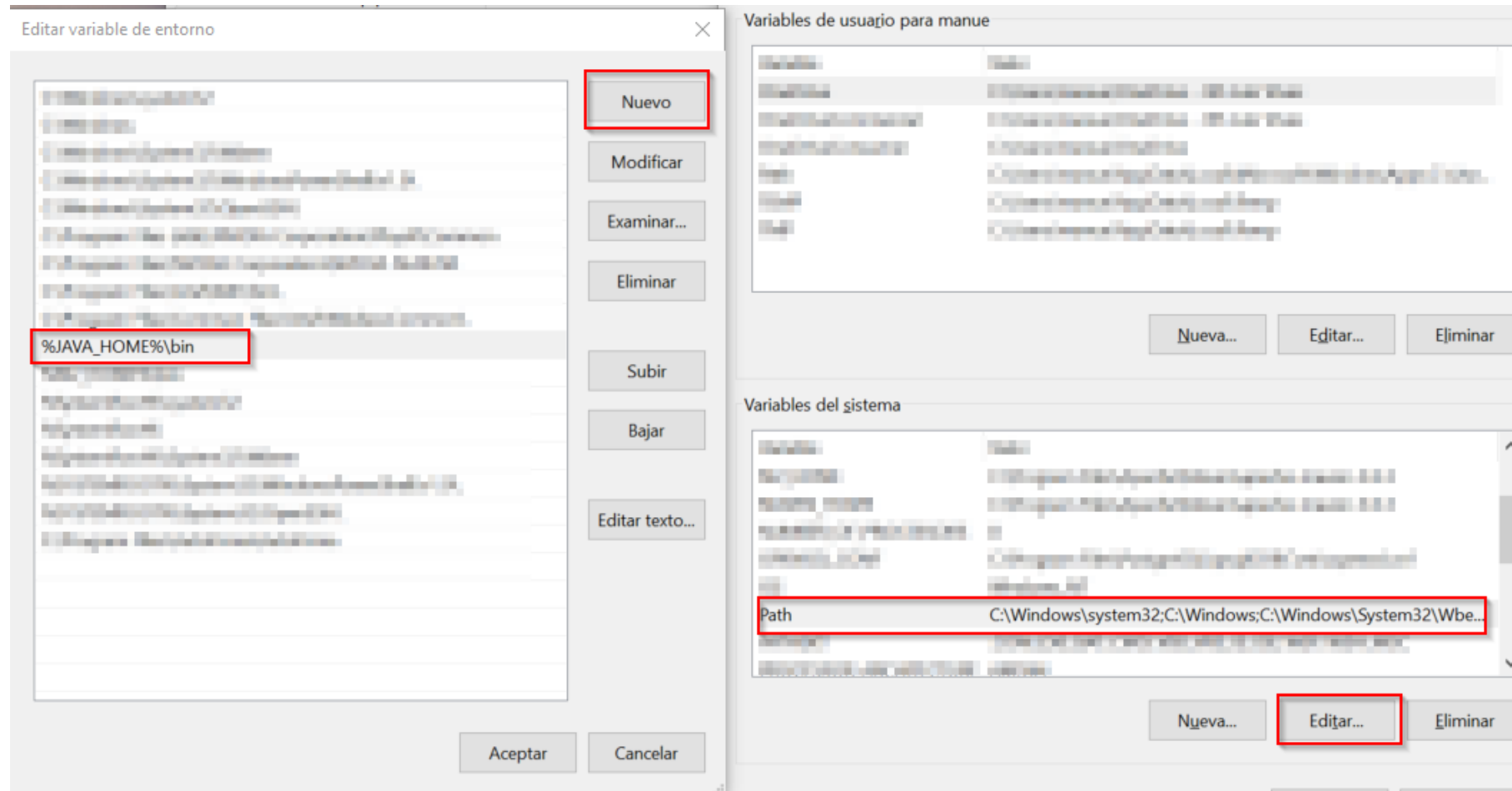
Para poder instalar el JDK lo primero que hemos de hacer es, lógicamente, descargar el correspondiente paquete. Podemos elegir entre distintos orígenes: Oracle, OpenJDK, IBM, etc., así como entre múltiples versiones. Lo habitual es que obtengamos un ejecutable (.exe para Windows), una imagen (.dmg para OS X) o un paquete (tar.gz para GNU/Linux) comprimido.

Puedes descargar la última versión del JDK oficial de Oracle [desde aquí](#). Si es una versión más antigua necesitas registrarte con Oracle (es gratuito). Otra opción mejor, desde que salió Java 11, es [descargarse el OpenJDK](#), la versión open source del kit de desarrollo.

Una vez instalado el JDK en el ordenador siguiendo los pasos del ejecutable, si queremos poder invocar el compilador desde cualquier lugar en la línea de comandos, es necesario que la carpeta bin figure en el PATH del sistema, para ello acceda a las variables de entorno y cree la variables del sistema JAVA_HOME, especificando la ruta de instalación del JDK

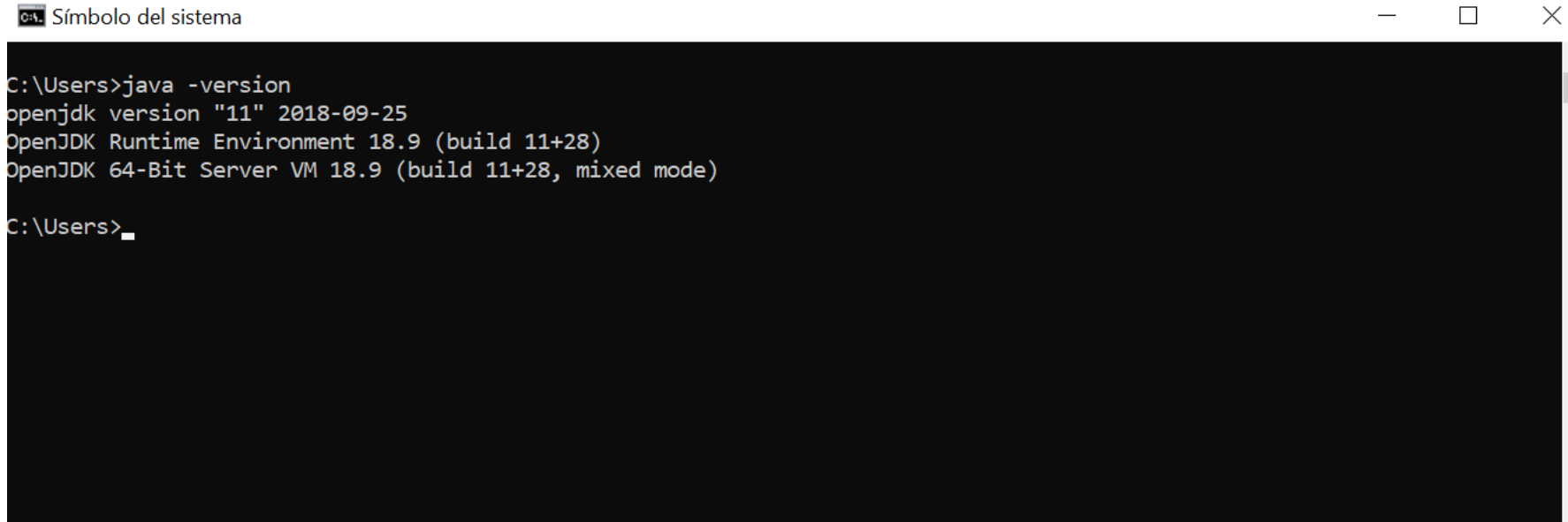


Por último modifica la variable del sistema PATH añadiendo %JAVA_HOME%\bin



Para verificar que habéis instalado correctamente, será necesario abrir un cmd y ejecutar

➤ `java -version`.



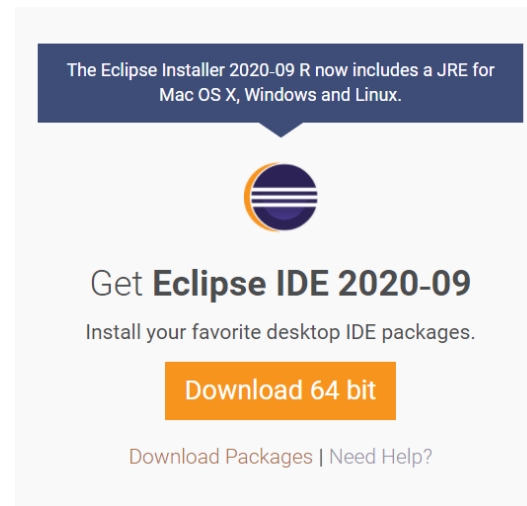
```
C:\Users>java -version
openjdk version "11" 2018-09-25
OpenJDK Runtime Environment 18.9 (build 11+28)
OpenJDK 64-Bit Server VM 18.9 (build 11+28, mixed mode)

C:\Users>
```


Para instalar el Eclipse, basta con descargarse el .exe de eclipse de la web <https://www.eclipse.org/eclipseide/>

Eclipse tiene diferentes paquetes dependiendo de que lenguaje o herramientas quieras usar, para la mayoría de los proyectos Java basta con usar Eclipse IDE for Enterprise Java Developers, además puedes añadirle plugins a la versión que hayas descargado completando tu entorno de desarrollo con las mejoras que consideres.

La mejor opción para instalar el Eclipse es descargarse el instalador que descargará y hará uso de los paquetes necesarios para instalar la versión que elijas.



También podrás descargar la versión que estimes oportuna directamente, en cuyo caso solo hará falta descomprimir el fichero zip.




Eclipse IDE for Java Developers

197 MB 530,962 DOWNLOADS

The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Maven and Gradle integration

Windows 64-bit
Mac Cocoa 64-bit
Linux 64-bit




Eclipse IDE for Enterprise Java Developers

382 MB 334,723 DOWNLOADS

Tools for developers working with Java and Web applications, including a Java IDE, tools for Web Services, JPA and Data Tools, JavaServer Pages and Faces, Mylyn, Maven and Gradle, Git, and more.

Windows 64-bit
Mac Cocoa 64-bit
Linux 64-bit




Eclipse IDE for C/C++ Developers

308 MB 314,425 DOWNLOADS

An IDE for C/C++ developers.

Windows 64-bit
Mac Cocoa 64-bit
Linux 64-bit




Eclipse IDE for Eclipse Committers

382 MB 23,874 DOWNLOADS

Package suited for development of Eclipse itself at Eclipse.org, based on the Eclipse Platform adding PDE, Git, Marketplace Client, source code and developer documentation.

Windows 64-bit
Mac Cocoa 64-bit
Linux 64-bit




Eclipse IDE for PHP Developers

252 MB 15,210 DOWNLOADS

The essential tools for any PHP developer, including PHP language support, Git client, Mylyn and editors for JavaScript, TypeScript, HTML, CSS and XML.

Windows 64-bit
Mac Cocoa 64-bit
Linux 64-bit



Get Eclipse IDE 2020-09

Install your favorite desktop IDE packages.

[Download 64 bit](#)

[Download Packages](#) | [Need Help?](#)

RELATED LINKS

- [Compare & Combine Packages](#)
- [New and Noteworthy](#)
- [Install Guide](#)
- [Documentation](#)
- [Updating Eclipse](#)
- [Forums](#)
- [Simultaneous Release](#)

MORE DOWNLOADS

- [Other builds](#)
- [Eclipse 2020-09 \(4.17\)](#)
- [Eclipse 2020-06 \(4.16\)](#)
- [Eclipse 2020-03 \(4.15\)](#)
- [Eclipse 2019-12 \(4.14\)](#)
- [Eclipse 2019-09 \(4.13\)](#)
- [Eclipse 2019-06 \(4.12\)](#)
- [Eclipse 2019-03 \(4.11\)](#)
- [Eclipse 2018-12 \(4.10\)](#)
- [Eclipse 2018-09 \(4.9\)](#)
- [Eclipse Photon \(4.8\)](#)
- [Eclipse Oxygen \(4.7\)](#)
- [Eclipse Neon \(4.6\)](#)
- [Older Versions](#)

All downloads are provided under the terms and conditions of the [Eclipse Foundation Software User Agreement](#) unless otherwise specified.

[Download](#)

Download from: Germany - University of Applied Sciences Esslingen (<https>)

File: [eclipse-java-2020-09-R-win32-x86_64.zip](#) SHA-512

[>> Select Another Mirror](#)

Actividad 1

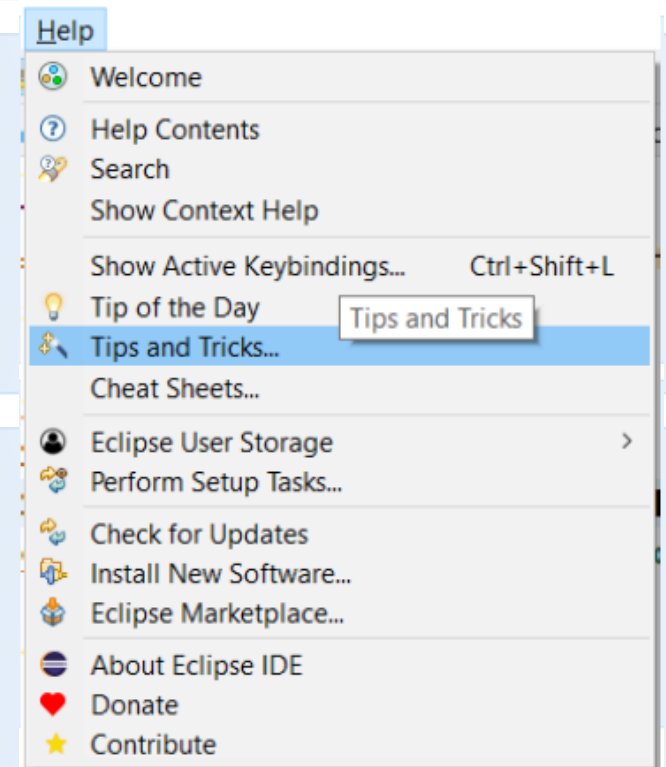
Comprueba la versión de java que hay en el equipo e instala la última versión de java (OpenJDK) disponible.

Actividad 2

Descarga e instala la versión de Eclipse IDE for PHP Developers.

Actividad 3

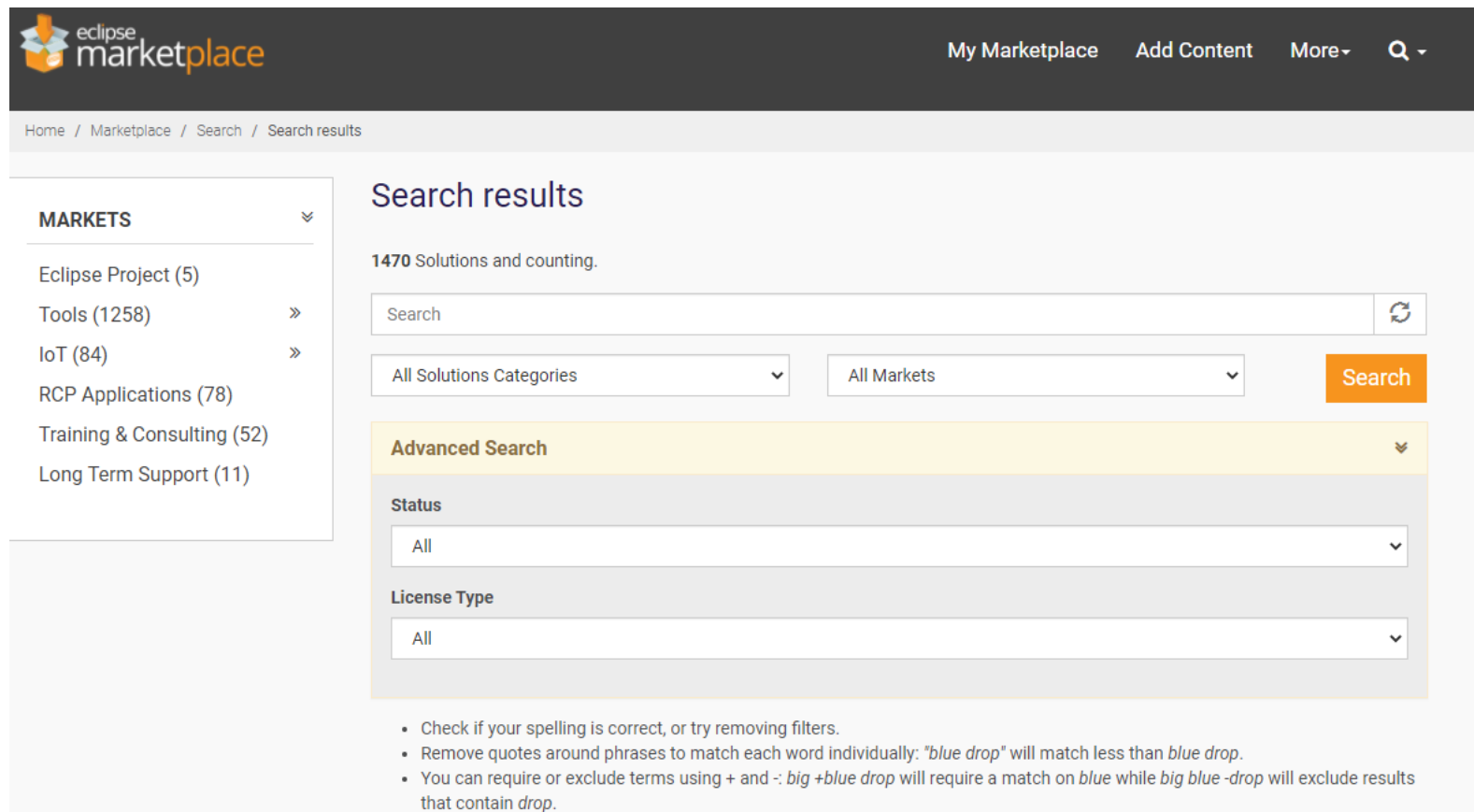
Mira el tips del día y prueba algunos atajos (Keybindings)



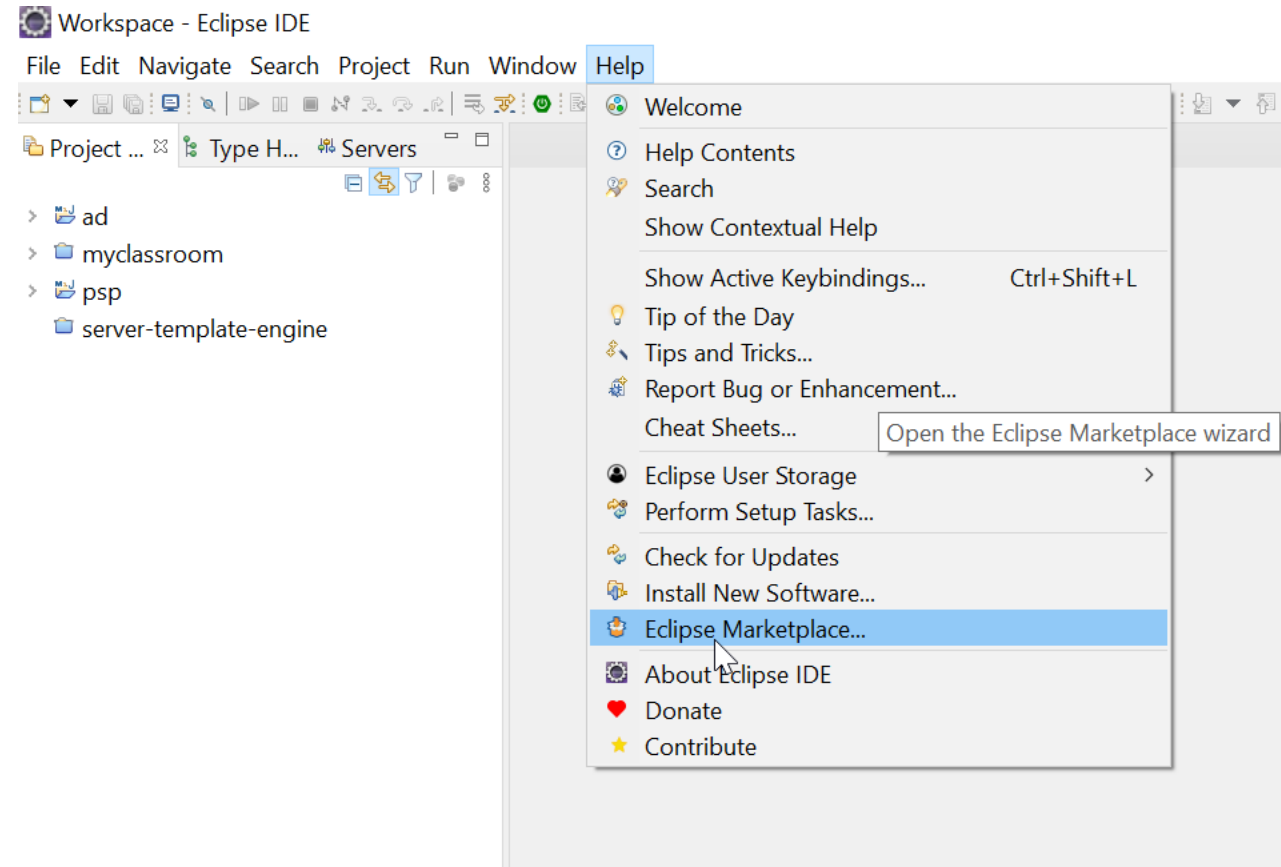
Gestión de módulos o plugins

ECLIPSE

Con la plataforma dada por un entorno de desarrollo como Eclipse podemos hacer uso de módulos y plugins para desarrollar aplicaciones. En la página oficial de marketplace <https://marketplace.eclipse.org/search/site/> podremos buscar los plugins que queramos instalar.

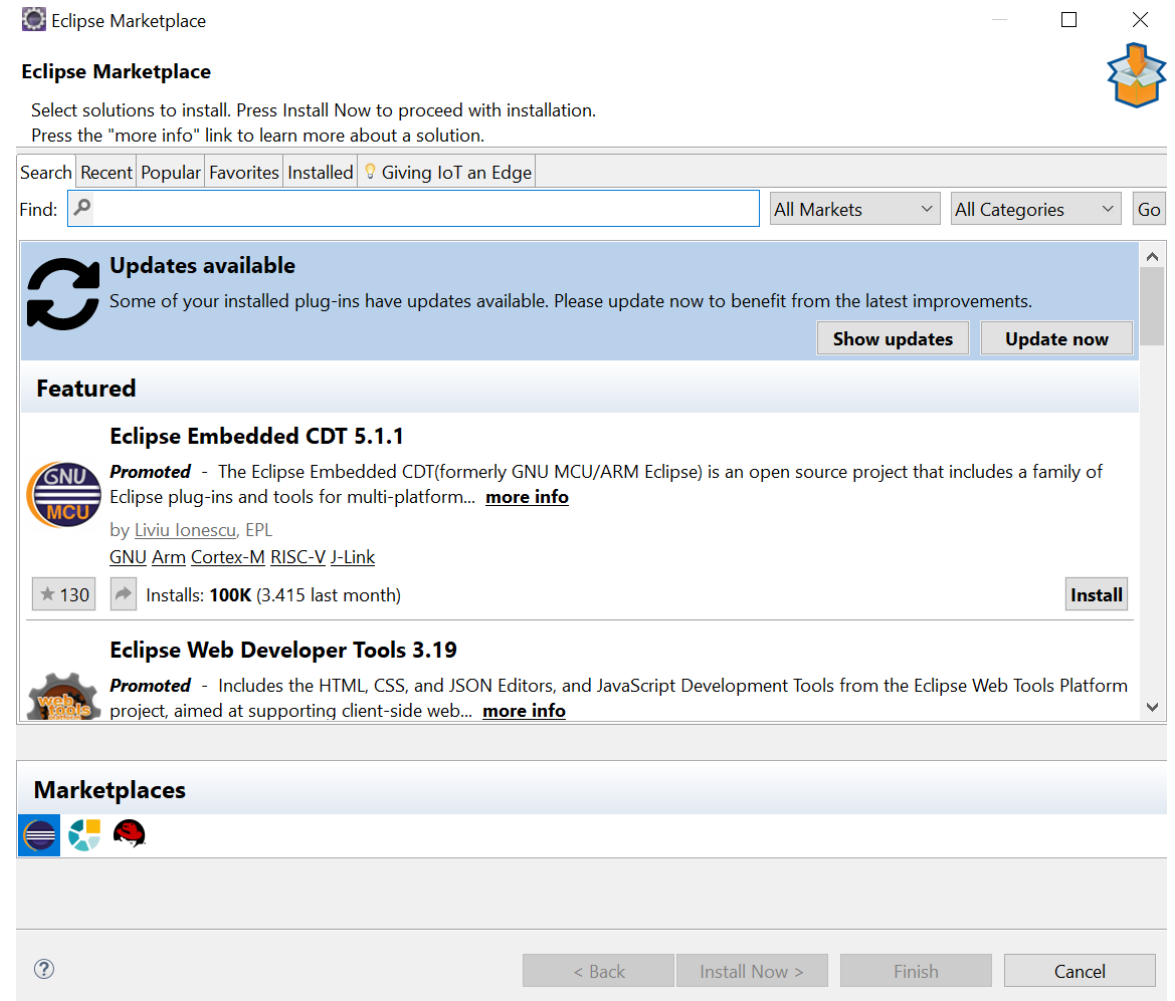


La forma más sencilla de instalar un plugin es usar el propio eclipse, accediendo al menú Marketplace en la ayuda.



Aquí podremos buscar los plugins que hagan falta por categorías o por su nombre:

- Pinchamos en **Install**.
- Nos mostrara lo que contiene el plugin.
- Nos pide que aceptemos los términos de licencia.
- Comenzará la instalación.
- Cuando termine nos pedirá que reiniciemos el programa.
- El plugin ya estará instalado y se configurará según el plugin.

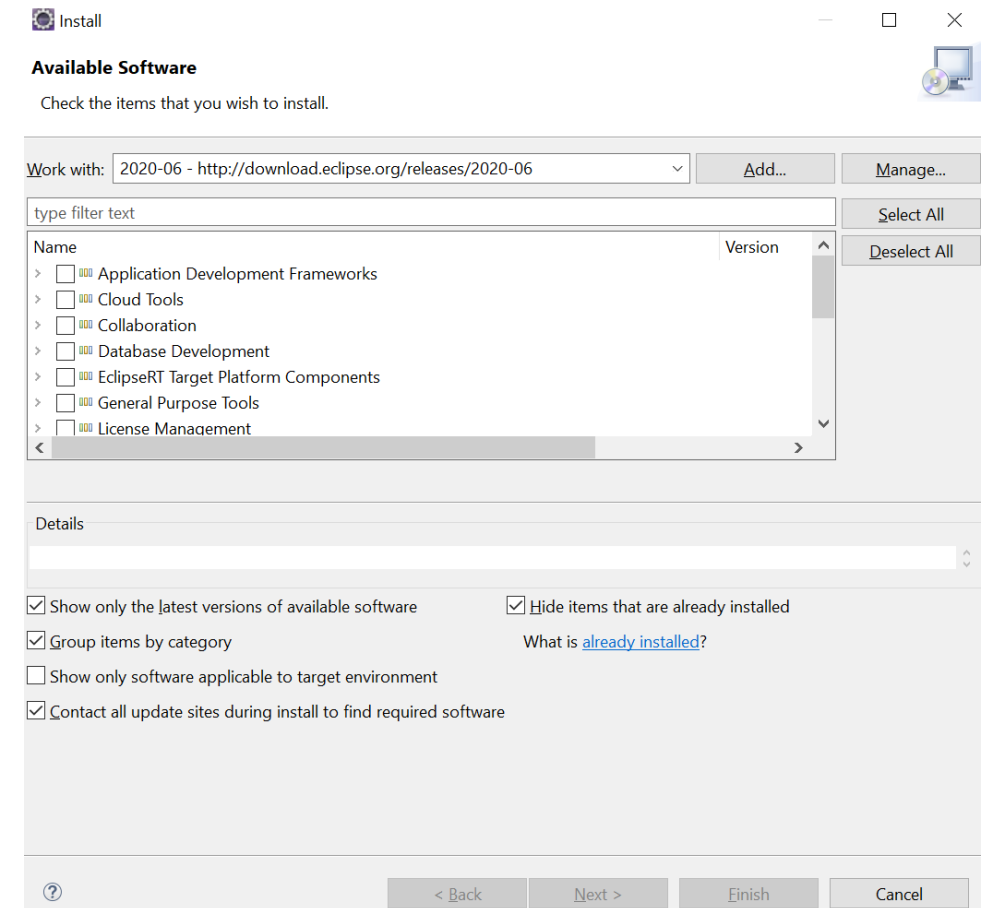
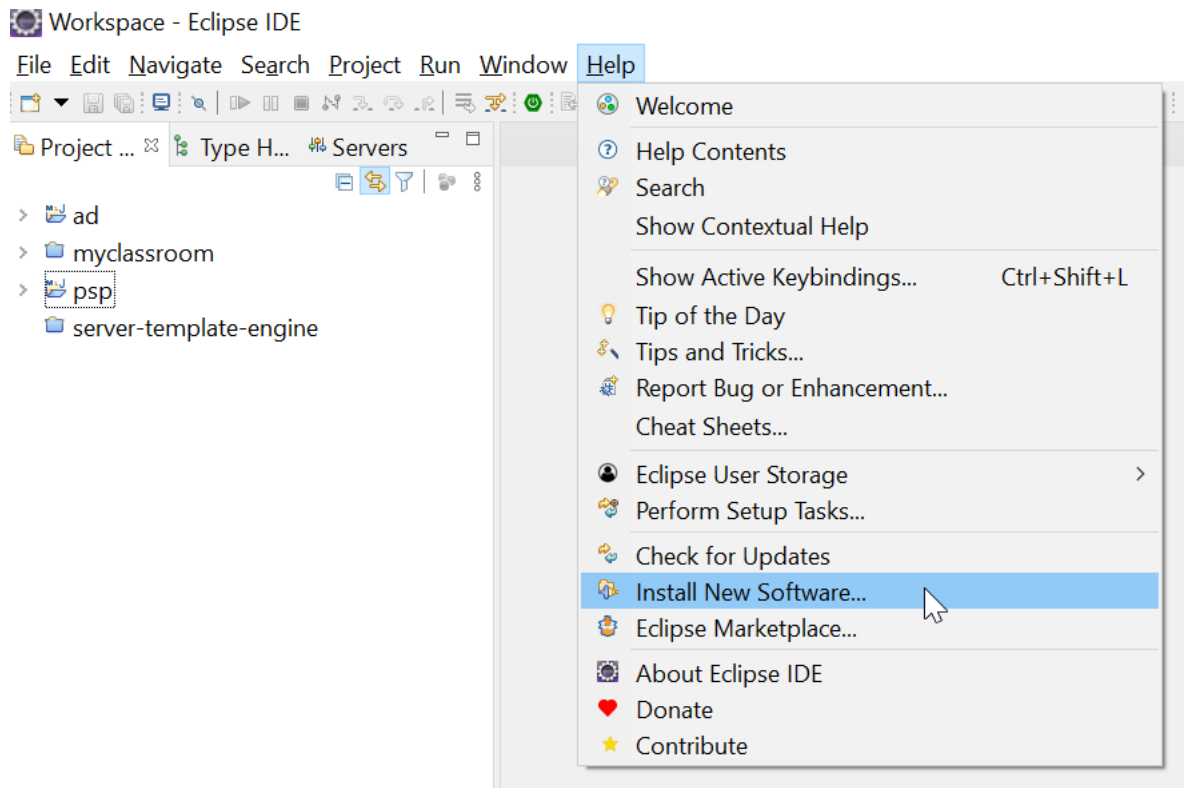


Para desinstalar plugins con MarketPlace haremos lo siguiente:

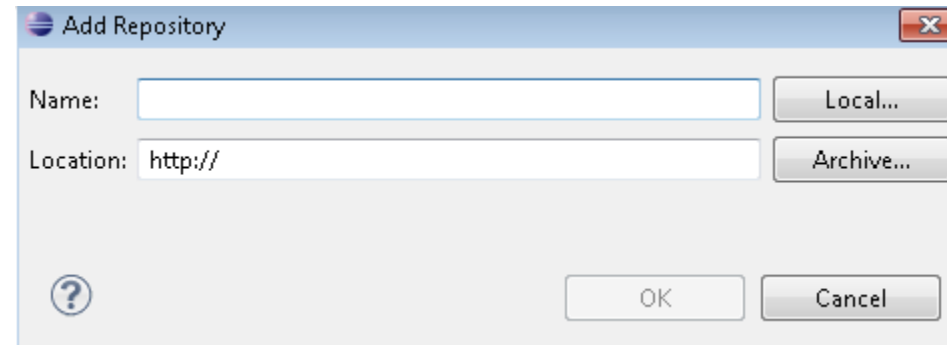
- Abrimos Eclipse.
- Pinchamos en Help -> Eclipse Marketplace -> Installed.
- Buscamos el plugin que queremos desinstalar.
- Pinchamos en Uninstall.
- Aceptamos que queremos desinstalarlo.
- Cuando termine nos pedirá que reiniciemos el programa.
- El plugin ya estará desinstalado.

Para instalar plugins desde un repositorio, sabiendo su dirección, haremos lo siguiente:

- Abrimos Eclipse.
- Pinchamos en Help -> Install New Software



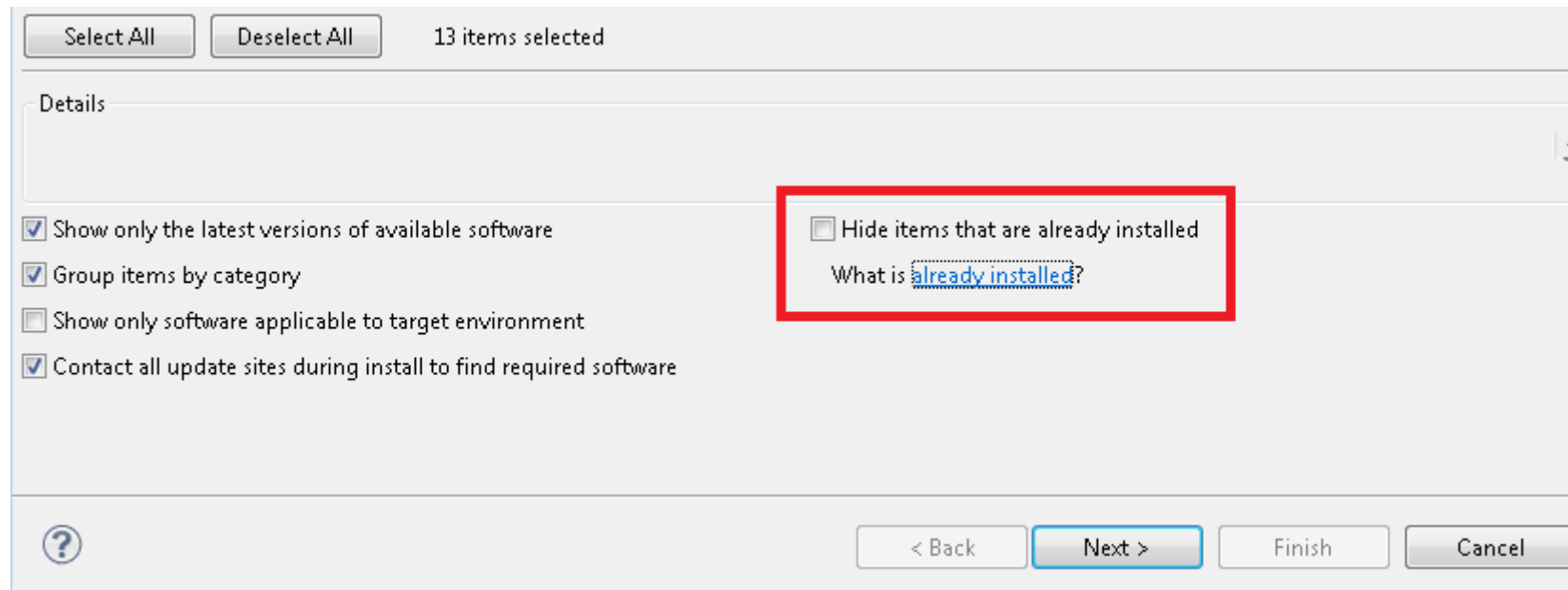
- En Work with copiamos la dirección del repositorio y en el cuadro de texto podemos filtrar el contenido del repositorio. Si ese repositorio lo vamos a usar muy a menudo, lo podemos guardar con un nombre. Para ello, pinchamos en Add y nos aparecerá un dialogo donde debemos indicar el nombre (opcional) y su localización (la dirección del repositorio). Estos aparecerán en la lista de Work with.



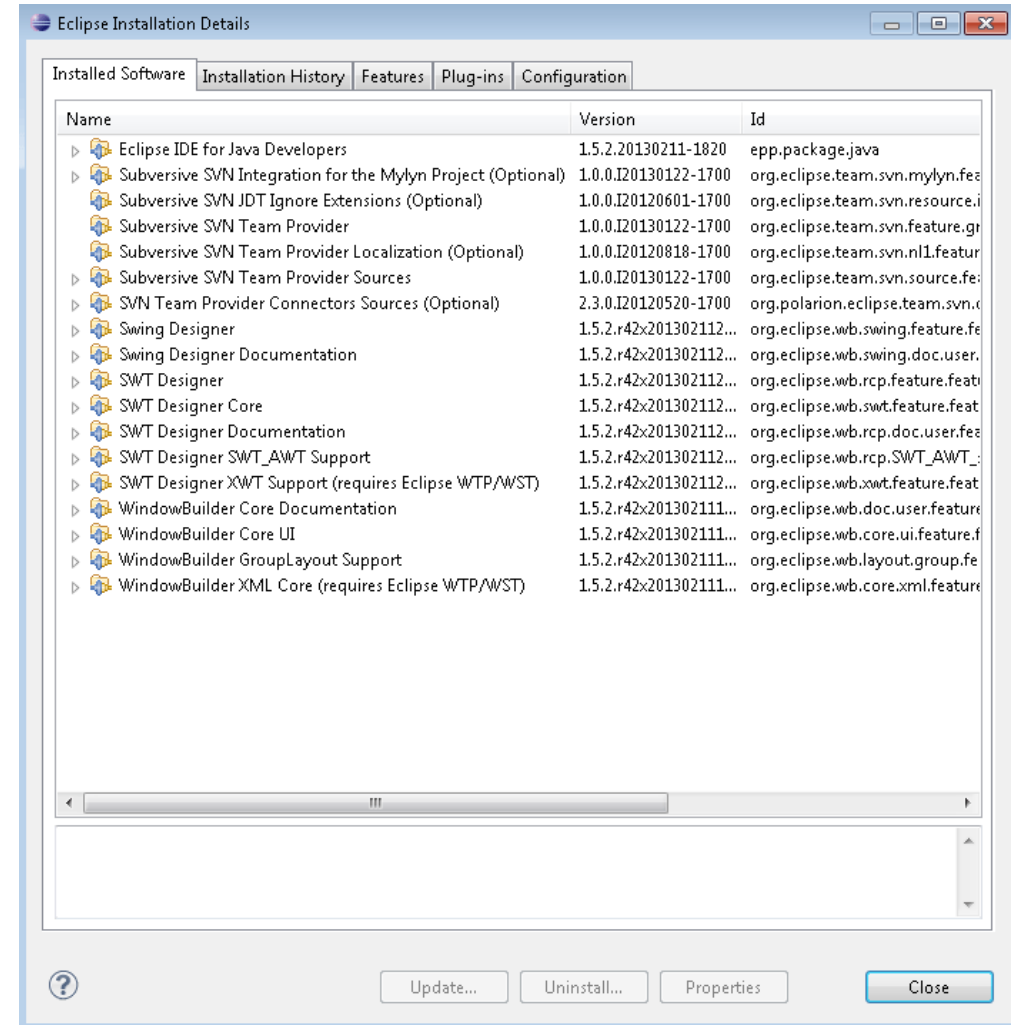
- Cuando encontremos los plugins necesarios, los seleccionamos y pinchamos en **Next**.
- Nos mostrara lo que contiene el plugin.
- Nos pide que aceptemos los términos de licencia.
- Comenzará la instalación.
- Cuando termine nos pedirá que reiniciemos el programa.
- El plugin ya estará instalado y se configurara según el plugin.

Para desinstalar plugins instalados desde un repositorios, haremos lo siguiente:

- Abrimos Eclipse.
- Pinchamos en Help -> Install New Software
- Pinchamos en el enlace “already installed”.



- Veremos la lista de los plugins instalados, seleccionamos los que queremos desinstalar.
- Pinchamos en Uninstall
- Aceptamos que queremos desinstalarlo.
- Cuando termine nos pedirá que reiniciemos el programa.
- El plugin ya estará desinstalado



Para instalar plugins manualmente haremos lo siguiente:



- Descargamos un plugin.
- Lo descomprimos.
- Abrimos la carpeta plugins buscamos el plugin que necesitamos (puede ser mas de uno).
- Los copiamos a la carpeta plugins dentro de nuestro eclipse.
- Hacemos lo mismo en la carpeta feature.
- Reiniciamos el programa.
- NOTA: no se llaman igual los de la carpeta plugins y feature pero si tienen el mismo código en su nombre

Para desinstalar plugins manualmente haremos lo siguiente:

- Borramos los archivos copiados anteriormente, tanto en plugins como en feature.
- Reiniciamos el programa.
- Espero que os sea de ayuda. Si tenéis duda, preguntad estamos para ayudarte..

Actividad 4

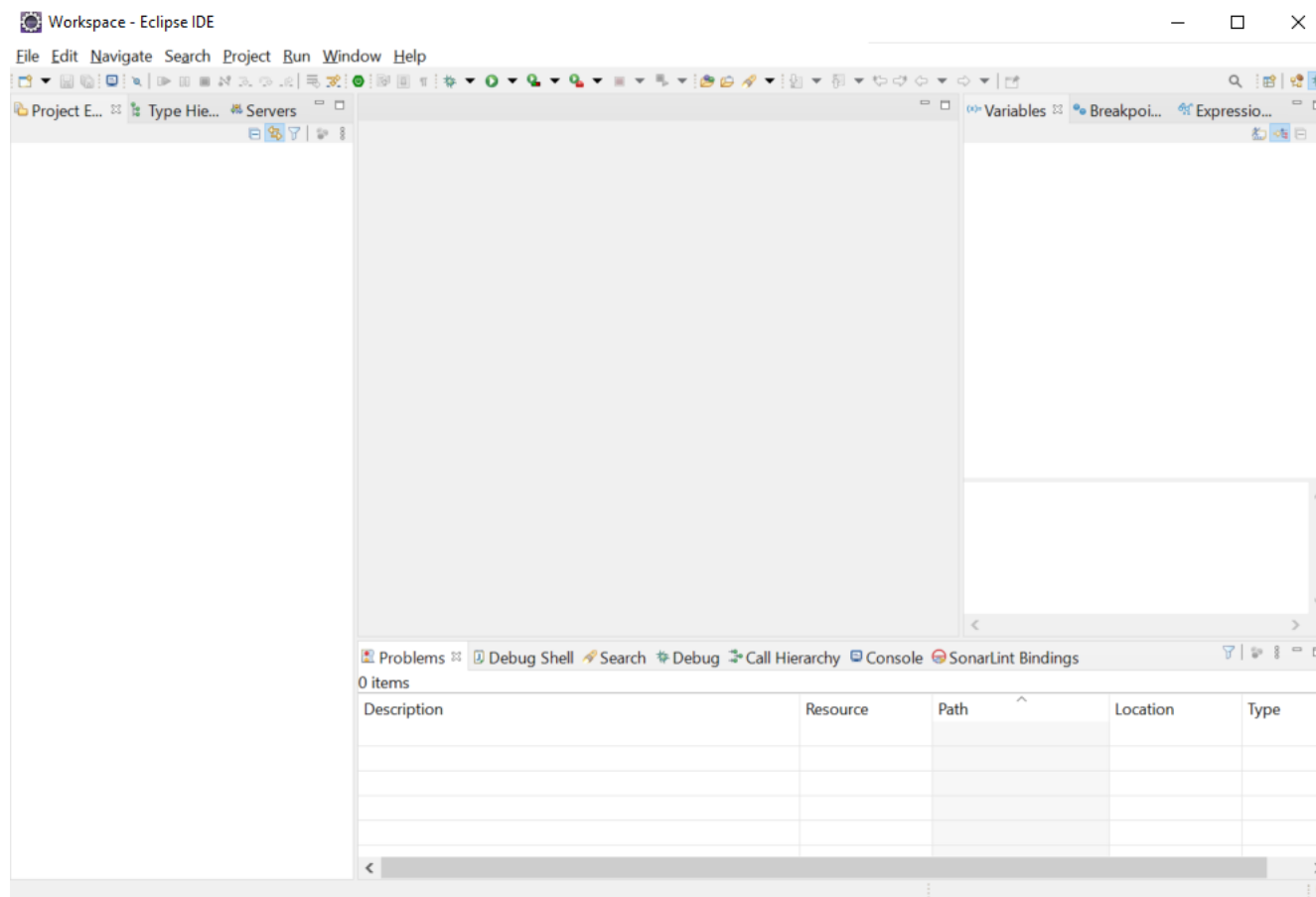
Instala los plugins:

- MoreUnit  para realizar test unitarios
- EcEmma.  para ver la cobertura de test
- Window builder para entorno gráfico con AWT y Swing
- Papyrus SysUML 1.6 para diseños UML

Estructura

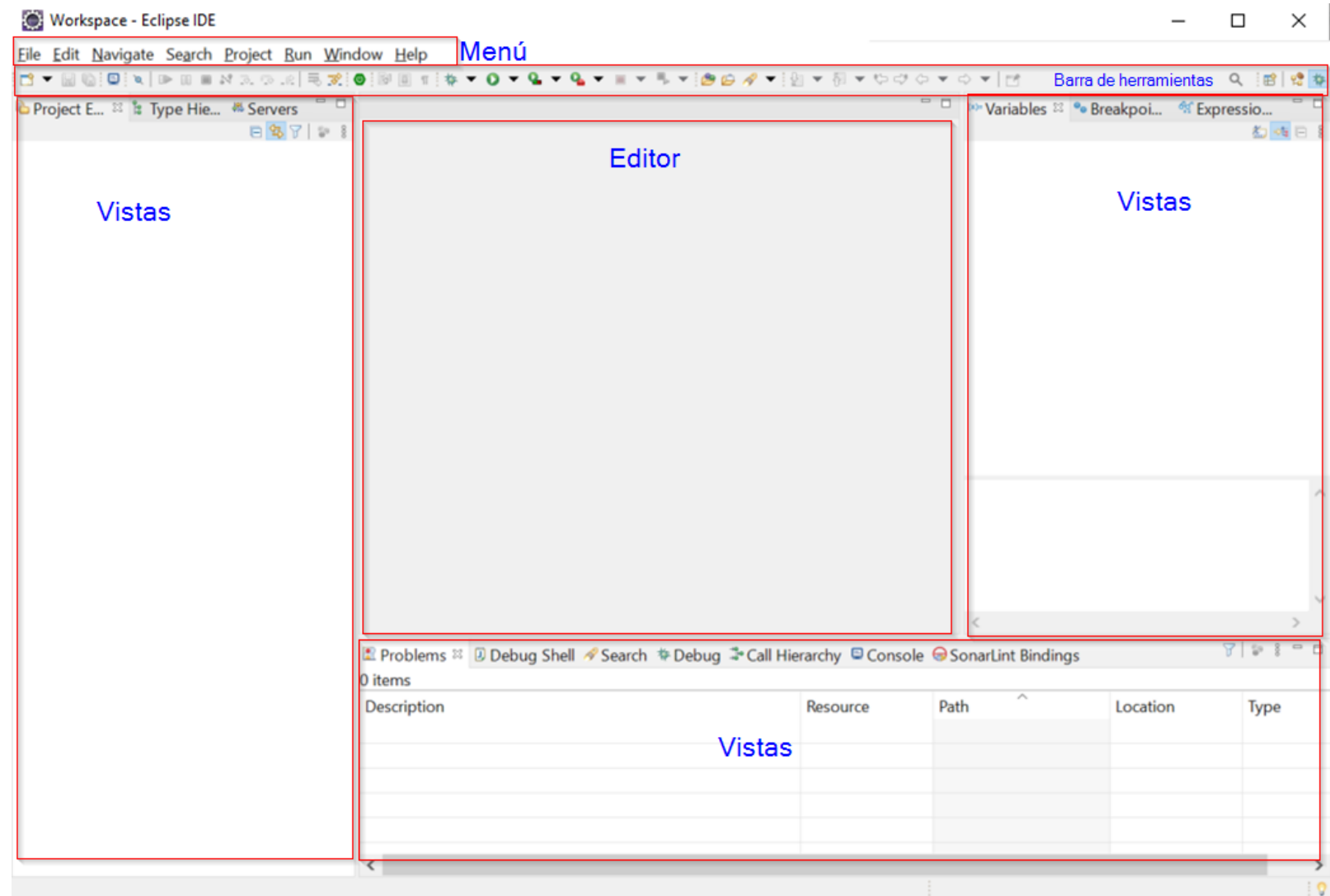
ECLIPSE

Una vez iniciado eclipse tendremos que elegir un workspace, el workspace es el espacio de trabajo, es decir una carpeta donde se crearan los diferentes proyectos, una vez elegido se abrirá el eclipse y el aspecto será parecido a este:



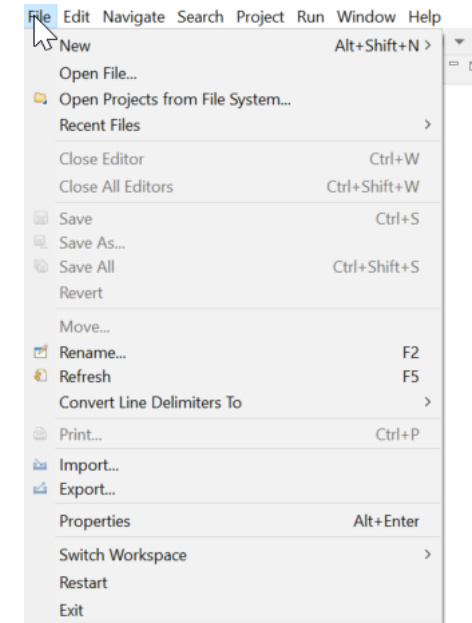
En Eclipse podemos encontrar:

- Editor
- Vistas y perspectivas
- Barra de herramientas
- Menú

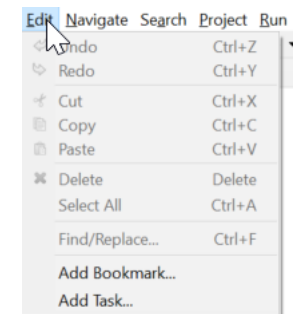


Menú. En el menú podemos encontrar diferentes opciones para controlar el IDE. Tenemos:

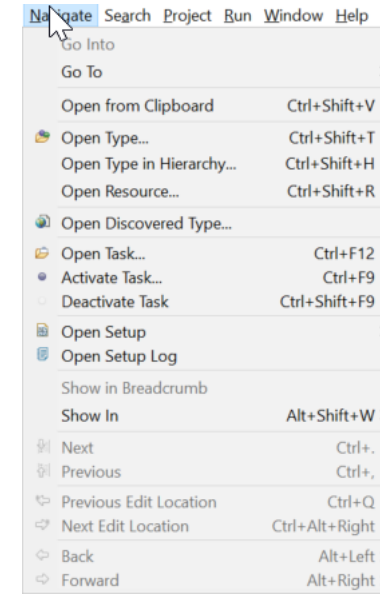
- File. Dónde podremos crear un nuevo proyecto, una nueva clase, enum, interface, fichero... en definitiva un nuevo elemento. Además podremos, modificar el workspace, abrir un proyecto ya creado en el sistema, importar o exportar elementos.



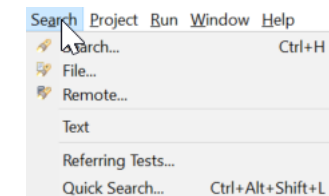
- Edit. Dónde podremos copiar, cortar, pegar, eliminar, buscar, deshacer, rehacer...



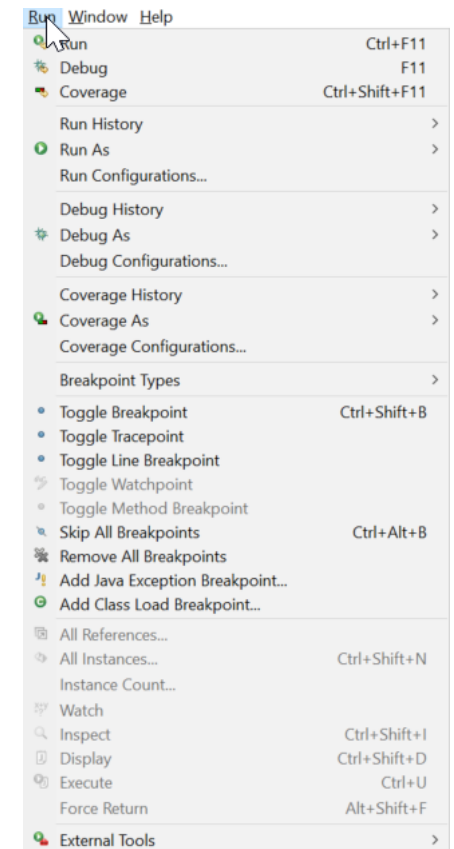
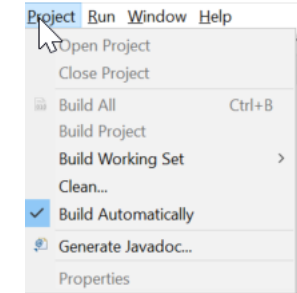
- Navigate. Para abrir un tipo, activar y desactivar tareas



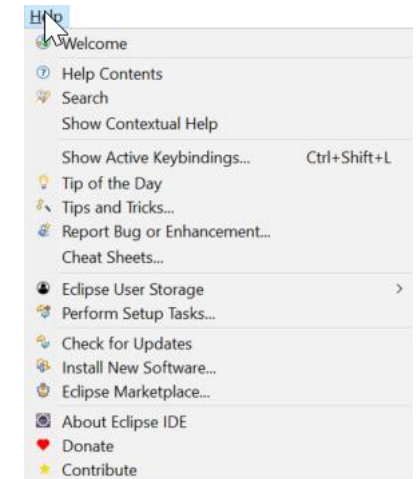
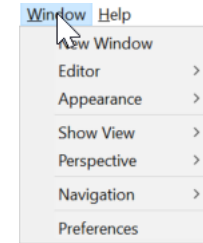
- Search. Para poder buscar dentro de ficheros o clases



- Project. Para gestionar el proyecto, interesante la opción de construcción automática (Build Automatically) que nos permite compilar el proyecto y ver los errores sin necesidad de ejecutar Build, otra opción interesante es la de Clean, que permite limpiar los proyectos para eliminar referencias, y carpetas de compilación. Otra opción es la de generar automáticamente Javadoc
- Run. Esta opción nos ayudará a ejecutar los proyectos según su tipo. Además tendremos las opciones para debuguear y crear puntos de ruptura, mirar la cobertura de código, etc...



- Window. Para abrir nuevas ventanas o cambiar perspectivas
- Help. El menú de ayuda con el que además podremos instalar/desinstalar plugis, así como buscar actualizaciones o conocer la versión de Eclipse instalada.

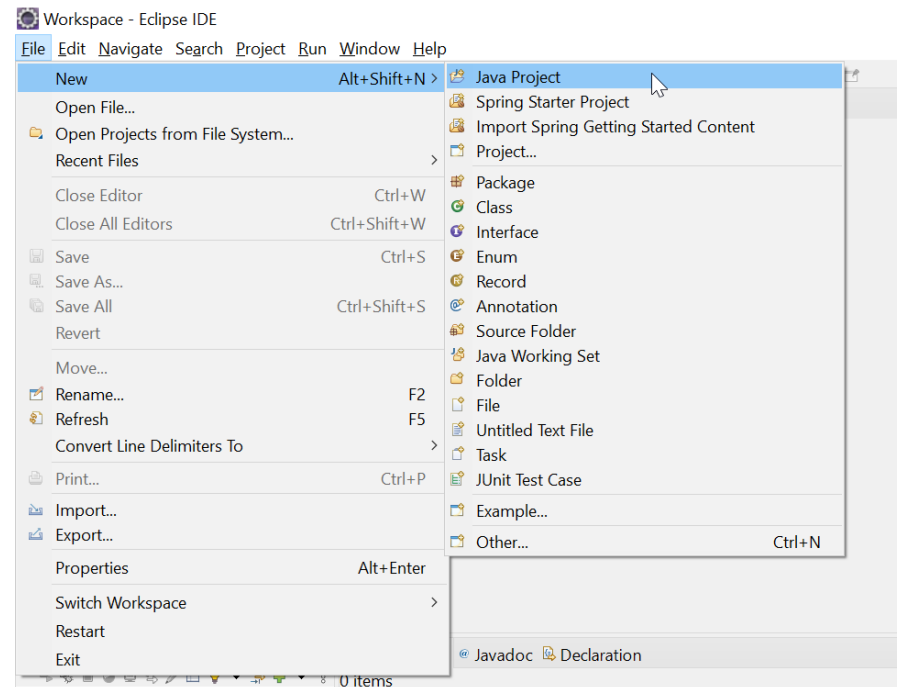


Uso básico

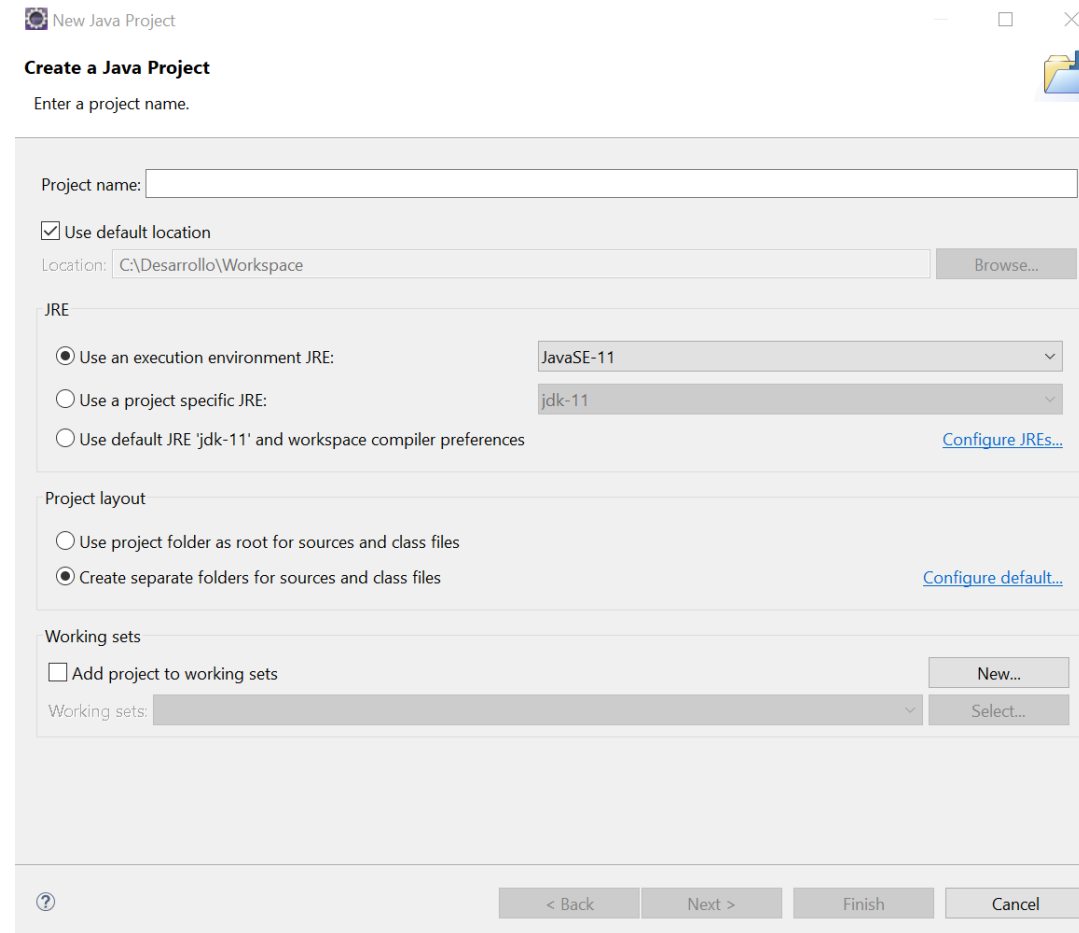
IDE

Dentro de eclipse se pueden crear diferentes tipos de proyectos para diferentes tipos de lenguajes, así para java podemos crear un proyecto java simple, proyecto Maven, proyectos Web. Dependiendo de lo que quieras hacer, se puede crear un tipo de proyecto y el propio IDE te creará las estructuras de carpeta, así como la configuración básica.

Para crear un proyecto java tendremos que acceder a File > New > Java project



En la siguiente pantalla bastará con indicarle una serie de información básica, como el nombre del proyecto o la JRE en la que se tiene que ejecutar



New Java Project

Create a Java Project

Enter a project name.

Project name:

☒ Use default location

Location: [Browse...](#)

JRE

☒ Use an execution environment JRE:

☐ Use a project specific JRE:

☐ Use default JRE 'jdk-11' and workspace compiler preferences [Configure JREs...](#)

Project layout

☐ Use project folder as root for sources and class files

☒ Create separate folders for sources and class files [Configure default...](#)

Working sets

☐ Add project to working sets [New...](#)

Working sets: [Select...](#)

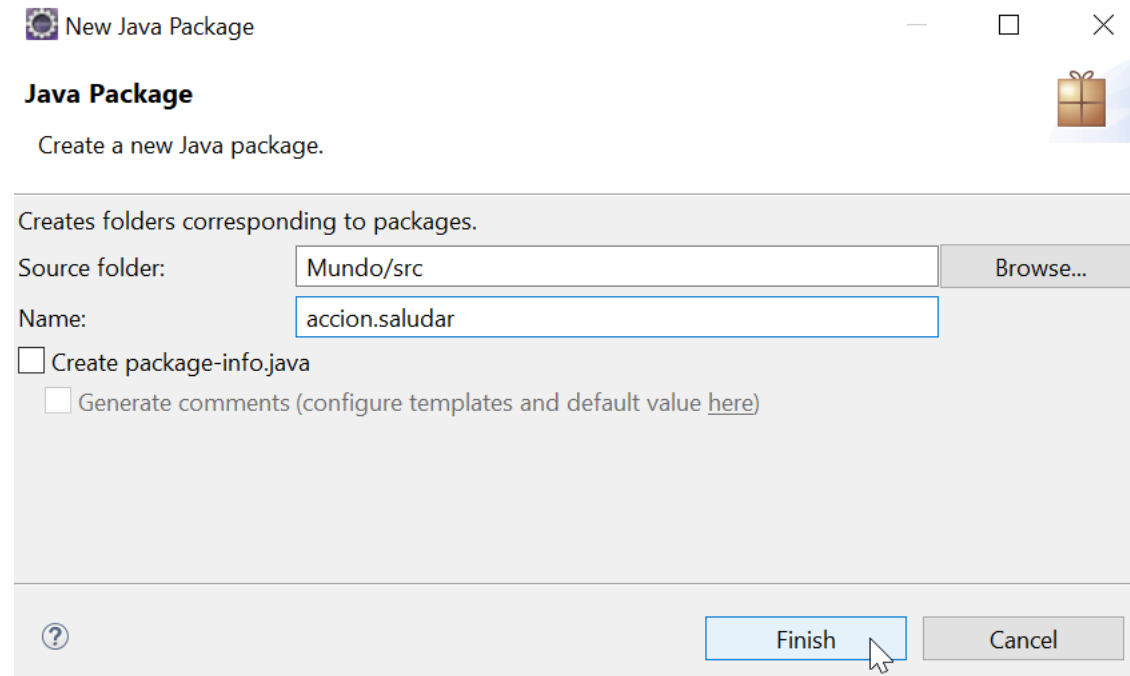
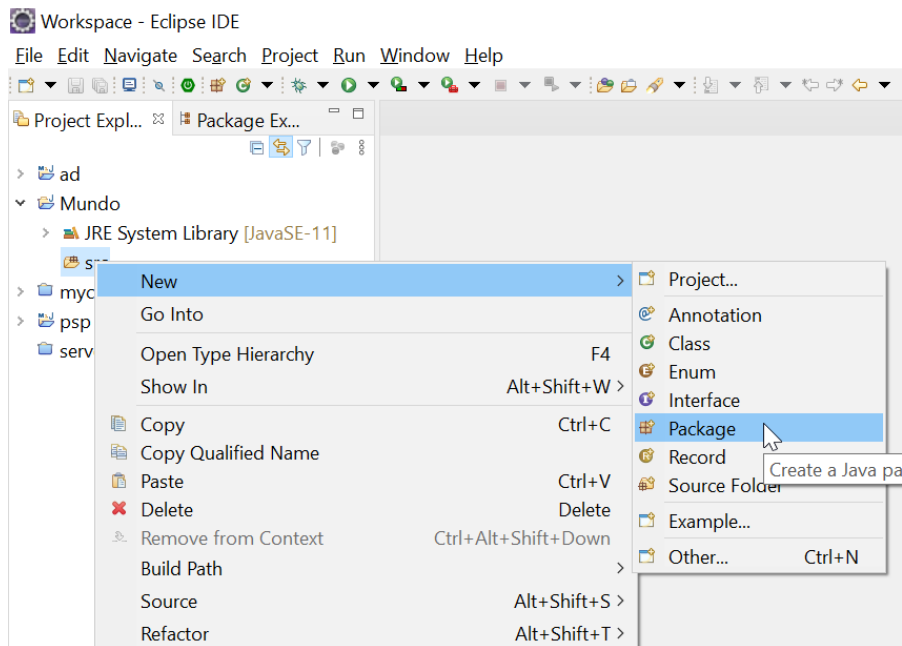
[? < Back](#) [Next >](#) [Finish](#) [Cancel](#)

Actividad 5

Crea un proyecto java, llamado ed, que se ejecute con la versión de java instalada en tu PC

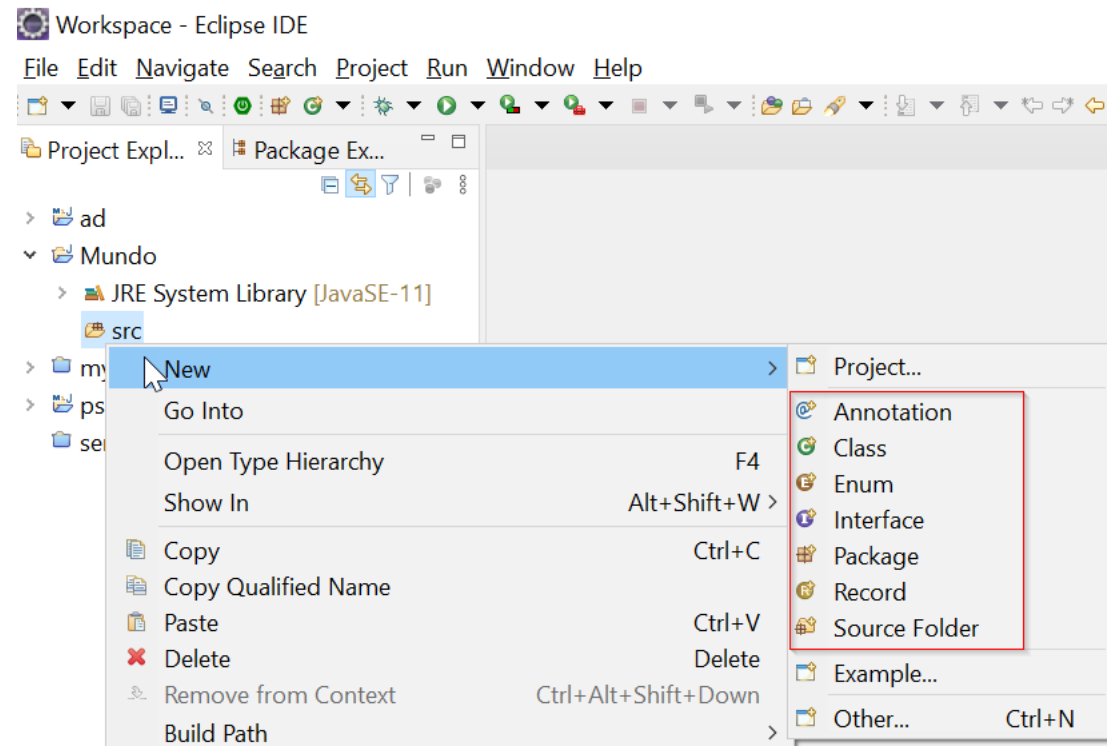
Nota: No crear java.info

Para ordenar un proyecto Java se suele crear un conjunto de directorio llamados paquetes. Para ello será necesario colocarte en la carpeta en la que quieras crear un nuevo paquete y clicar con el botón derecho y dentro del menú contextual elegir la opción New > Package



Para crear un sistema de directorios basados en paquetes, basta con separar por puntos, en el ejemplo anterior se creará el paquete accion.saludar, que estará compuesto por una carpeta acción y dentro otra carpeta saludar.

Para crear una clase, una interfaz o un enum, el proceso es similar al de la creación de un paquete y basta con seleccionar el elemento dentro de la opción new que se desee



Actividad 6

Dentro de nuestro proyecto Java ed crea

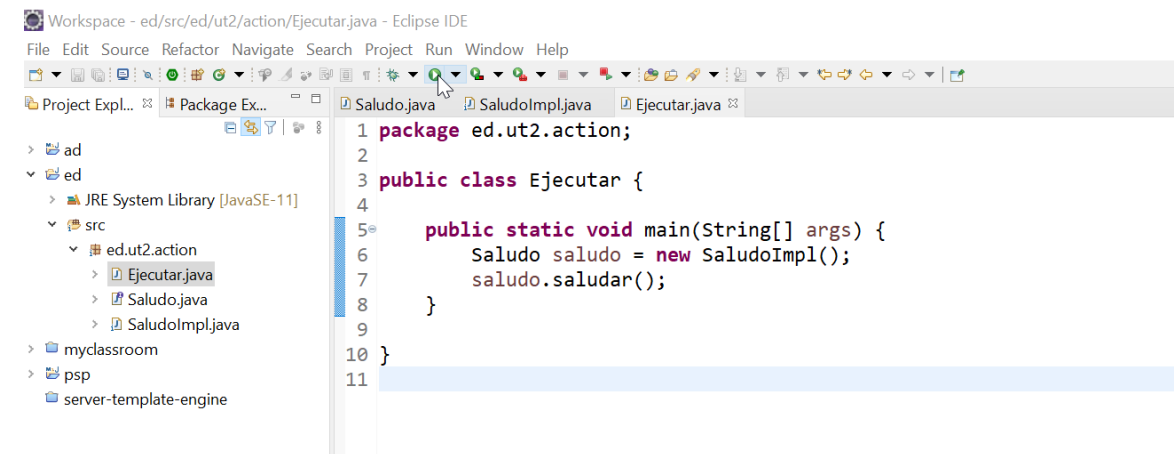
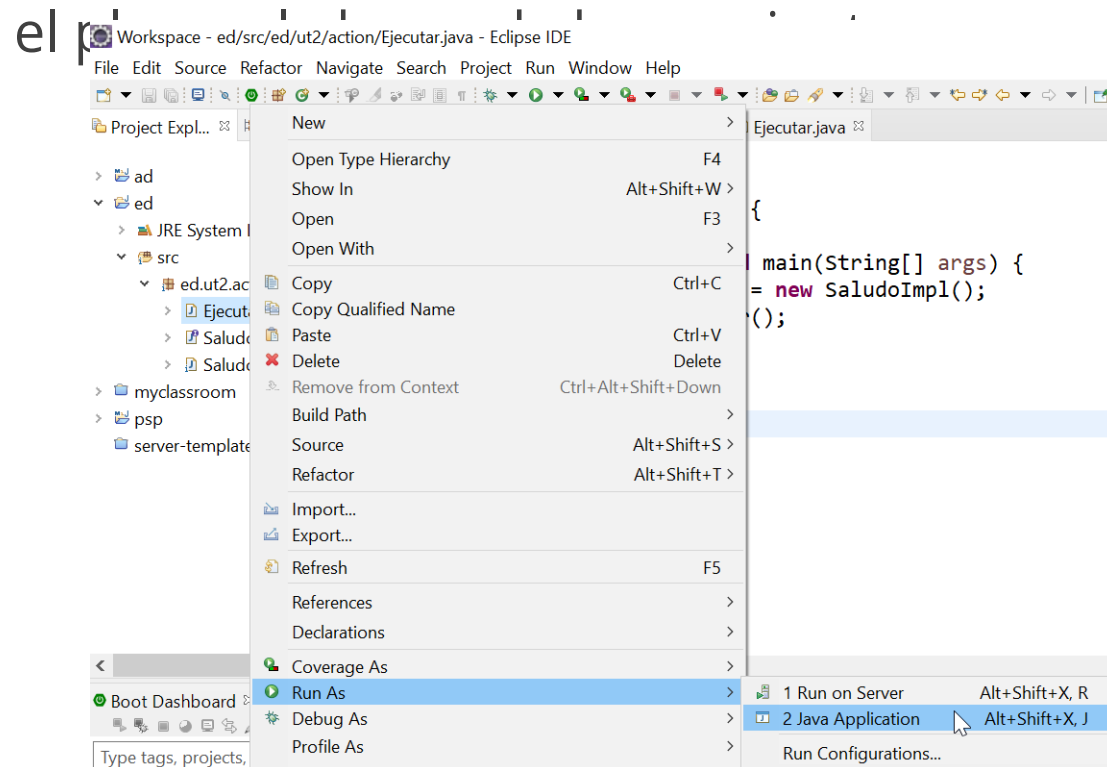
- Un paquete llamado ed.ut2.action.
- Una Interface llamada Saludo que contenga un método llamado saludar
- Una clase que llamada SaludoImpl que implemente la interfaz y que contenga la implementación del método saludar()
- Una clase llamada Ejecutar que tenga un main y que cree un objeto Saludo e invoque a saludar

```
Saludo.java SaludoImpl.java Ejecutar.java
1 package ed.ut2.action;
2
3 public interface Saludo {
4
5     /**
6      * Metodo usado para saludar
7      */
8     public void saludar();
9 }
```

```
Saludo.java SaludoImpl.java Ejecutar.java
1 package ed.ut2.action;
2
3 public class SaludoImpl implements Saludo {
4
5     public void saludar() {
6         System.out.println("Hola mundo");
7     }
8
9 }
```

```
Saludo.java SaludoImpl.java Ejecutar.java
1 package ed.ut2.action;
2
3 public class Ejecutar {
4
5     public static void main(String[] args) {
6         Saludo saludo = new SaludoImpl();
7         saludo.saludar();
8     }
9
10 }
```

Una vez que hayamos construido un proyecto java, podremos ejecutarlo para ver una respuesta. Normalmente un proyecto java tiene unos parámetros de entrada y una salida esperada. Para ejecutar un proyecto con Eclipse basta con colocarte en la clase que contenga el método de ejecución (normalmente es el main) y con el botón derecho acceder a la opción Run As > Java application. O directamente, con la clase abierta, pulsar el botón de ejecución en la barra de herramientas.



Actividad 7

Ejecuta nuestro proyecto ed, situándote en la clase Ejecutar.java

En muchas ocasiones puede ser interesante depurar el código para encontrar un bug, o incluso para entender cómo se comporta un código complejo. Usando Eclipse basta con colocar un punto de ruptura (breakpoint) en el código y ejecutarlo en modo depuración (debug), y el programa parará automáticamente su ejecución al llegar al punto de ruptura. Una vez parada la ejecución del programa, podremos analizar el valor de las variables en ese momento, crear expresiones para analizar cómo cambian dichas expresiones durante la ejecución del programa, así como interactuar con cierta parte del código para ver cómo se comporta el sistema.

[Manual depuración local en eclipse](#)

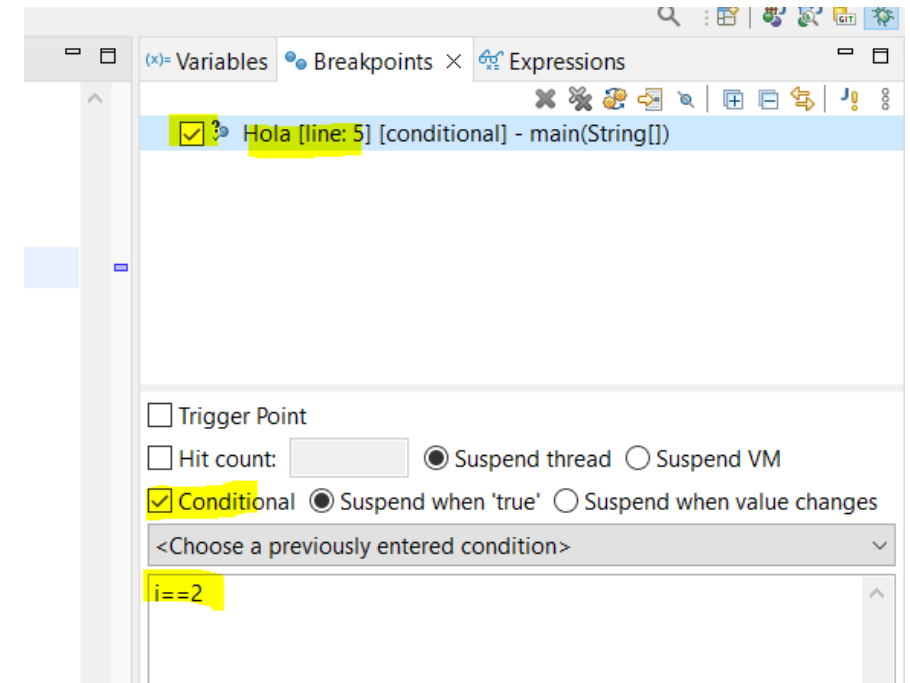
<https://www.youtube.com/watch?v=ymV7IUUHkUU>

[Depuración IntelliJ](#)

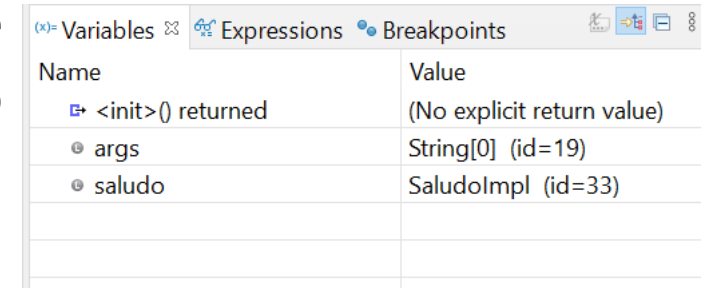
Para añadir un punto de ruptura basta con colocarse en el número de línea y pulsar dos veces con el botón izquierdo del ratón, y aparecerá un círculo indicando que se ha creado un breakpoint.

Puede ser de gran ayuda la vista de breakpoints, donde se puede ver de un simple vistazo todos los breakpoints de todos los proyectos que tenemos abiertos. Además, puedes colocar breakpoints condicionales

```
Saludo.java SaludoImpl.java Ejecutar.java
1 package ed.ut2.action;
2
3 public class Ejecutar {
4
5     public static void main(String[] args) {
6         Saludo saludo = new SaludoImpl();
7         saludo.saludar();
8     }
9
10 }
11
```

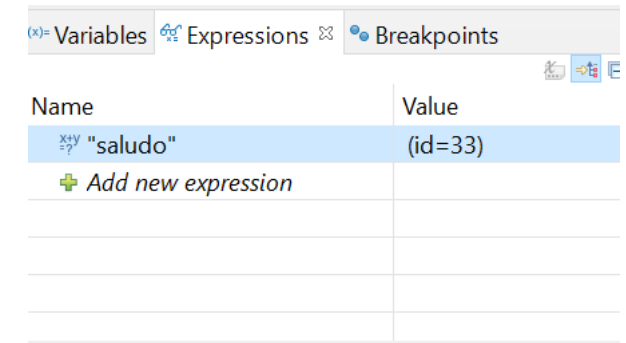


Otra vista que puede ayudarnos es la vista Variables. Dónde se irán mostrando las variables que tengamos en el código mientras se depura.



Name	Value
<init>() returned	(No explicit return value)
args	String[0] (id=19)
saludo	SaludoImpl (id=33)

También cabe destacar la vista Expresiones dónde podremos crear nuestras propias Expresiones para ver un valor concreto dentro de una variable.



Name	Value
"saludo"	(id=33)
+ Add new expression	

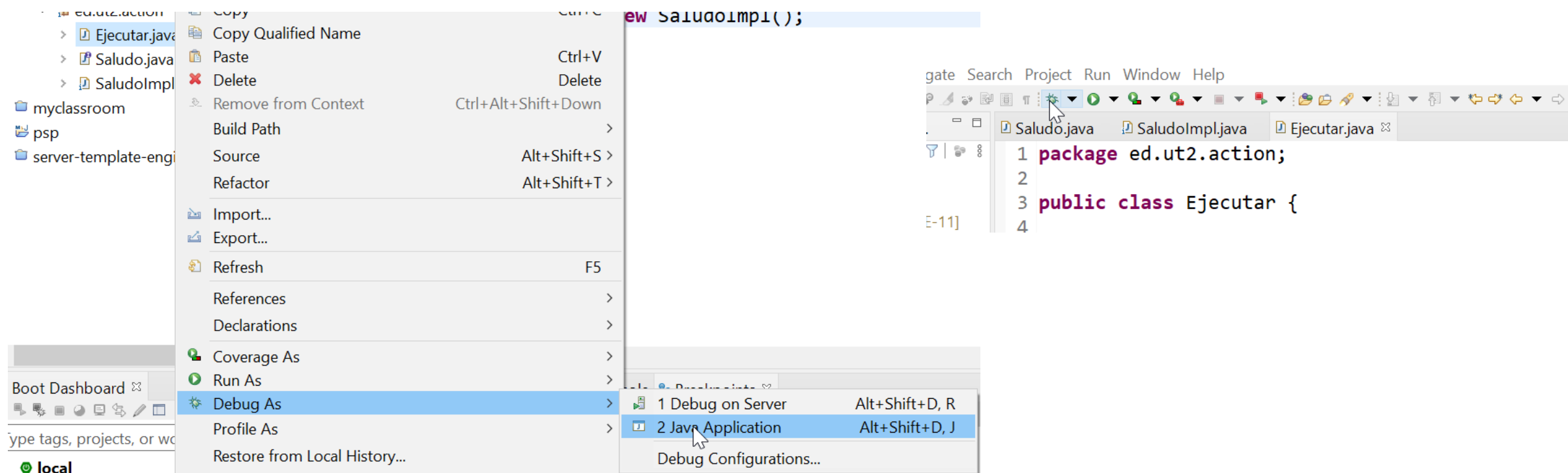
ed.ut2.action.SaludoImpl@30b8a058

Si durante la depuración escribes una instrucción en tu programa, la seleccionas y pulsas CTRL+MAY+i se ejecutará esa instrucción aunque inicialmente no estuviera en el programa.

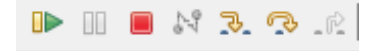
La instrucción `System.out.println ("OTRA");` se ha escrito durante la depuración :






```
public static void main (String a[]) {  
    for (int i=0; i<5;i++) {  
        System.out.println("HOLA");  
        i=i+1;  
        System.out.println("OTRA");  
        System.out.println(i);  
    }  
}
```

Para depurar código, una vez creado los breakpoints, basta con colocarte en la clase que contenga el método de ejecución (normalmente es el main) y con el botón derecho acceder a la opción **Debug As > Java Application**. O directamente, con la clase abierta, pulsar el escarabajo en la barra de herramientas.



Una vez parado el código, podremos controlar como se avanza a través del código.



-  Step into (F5). Si estamos parado en un método, continuará la ejecución accediendo al interior del método.
- Step into Selection: . Seleccionas un método y se realiza la ejecución hasta entrar en ese método.
-  ➤ Step return (F7). Si hemos entrado en un método usando step into y queremos regresar al punto de ruptura anterior, podemos usar step return.
-  ➤ Step over (F6). Si estamos parado en un método, continuará la ejecución saltando a la siguiente línea de nuestro código sin acceder al método, o a un código de ruptura interno si el método que estamos saltando o algún método invocado desde dentro de este lo tuviera.
-  Terminate (Ctrl + F2). Termina la ejecución del programa.
-  Resume (F8). Continúa la ejecución del programa hasta el final, o el siguiente punto de

Actividad 8

Modifica las clases de nuestro proyecto para que incorpore un bucle para saludar a 30 personas.

- Crear un punto de ruptura dentro del bucle
- Analizar la variable i que valor toma.
- Crea una expresión que analice la variable nombre (saludo.getName())

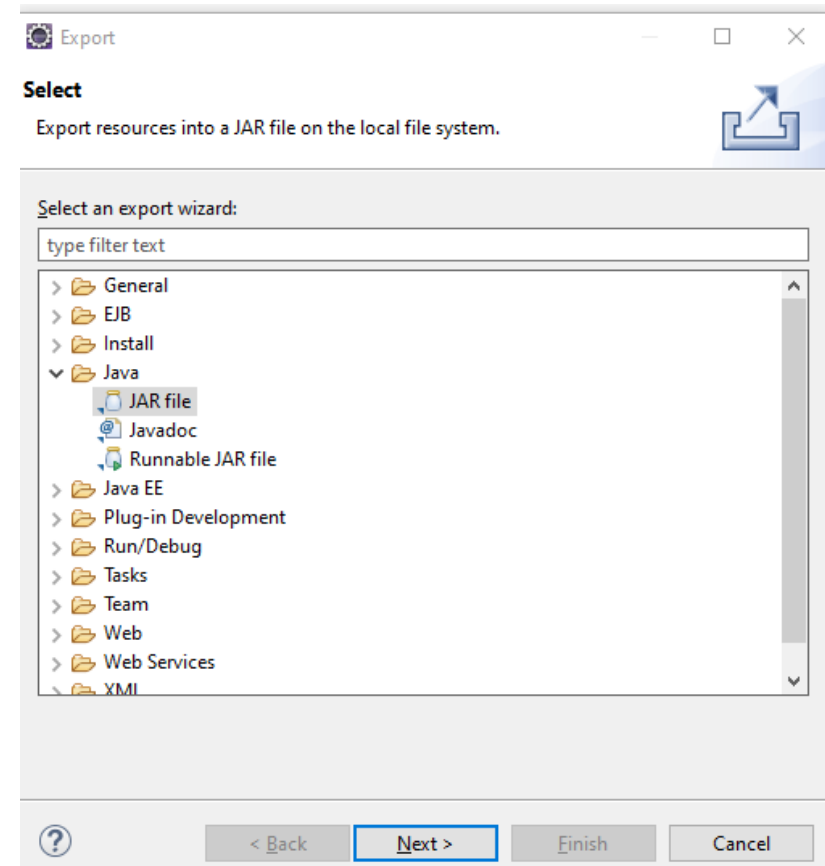
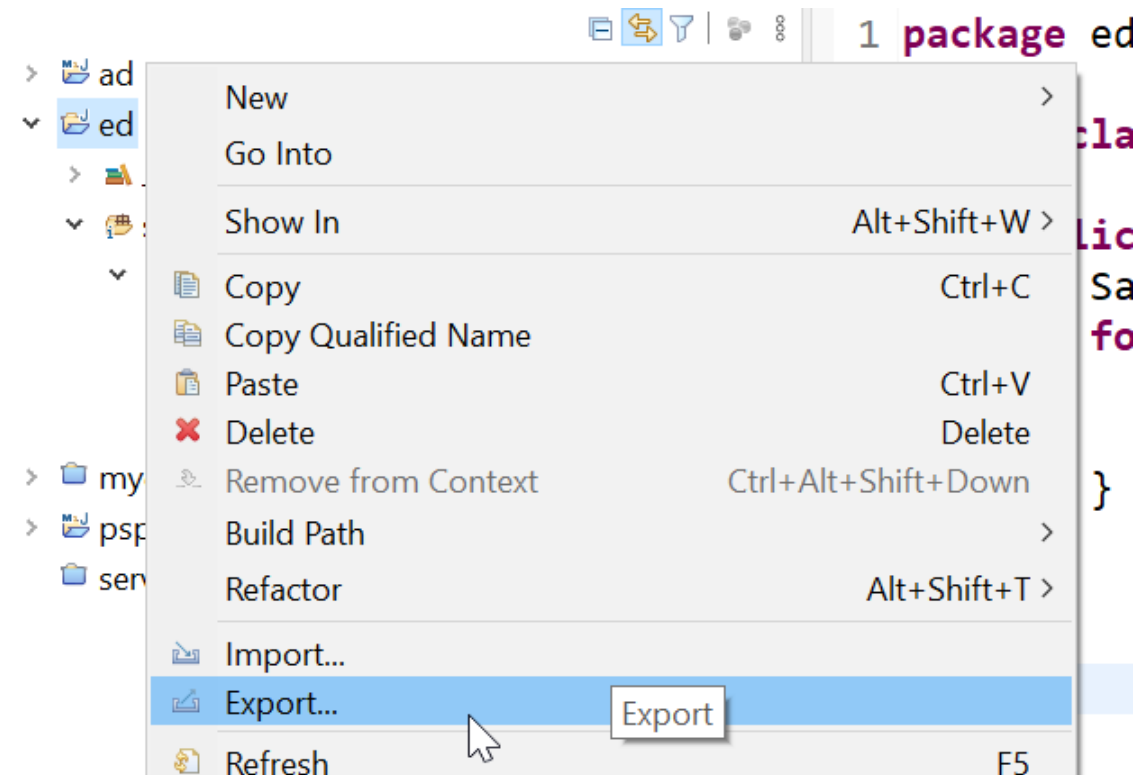
```
Saludo.java SaludoImpl.java Ejecutar.java
1 package ed.ut2.action;
2
3 public class SaludoImpl implements Saludo {
4
5     //Nombre a saludar
6     private String name;
7
8     public void saludar() {
9         System.out.println("Hola " + name);
10    }
11
12    public String getName() {
13        return name;
14    }
15
16    public void setName(String name) {
17        this.name = name;
18    }
19
20 }
```

```
Saludo.java SaludoImpl.java Ejecutar.java
1 package ed.ut2.action;
2
3 public class Ejecutar {
4
5     public static void main(String[] args) {
6         SaludoImpl saludo = new SaludoImpl();
7         for (int i = 1; i <= 30; i++) {
8             saludo.setName("Persona" + i);
9             saludo.saludar();
10        }
11    }
12
13 }
```

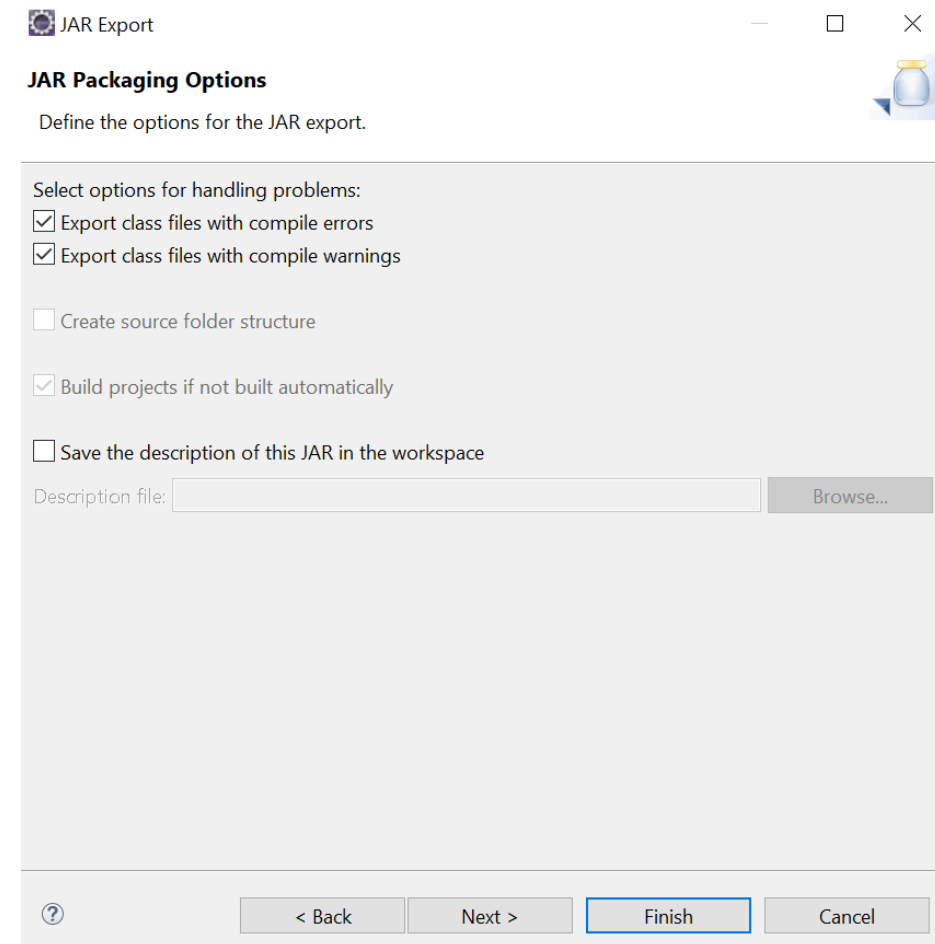
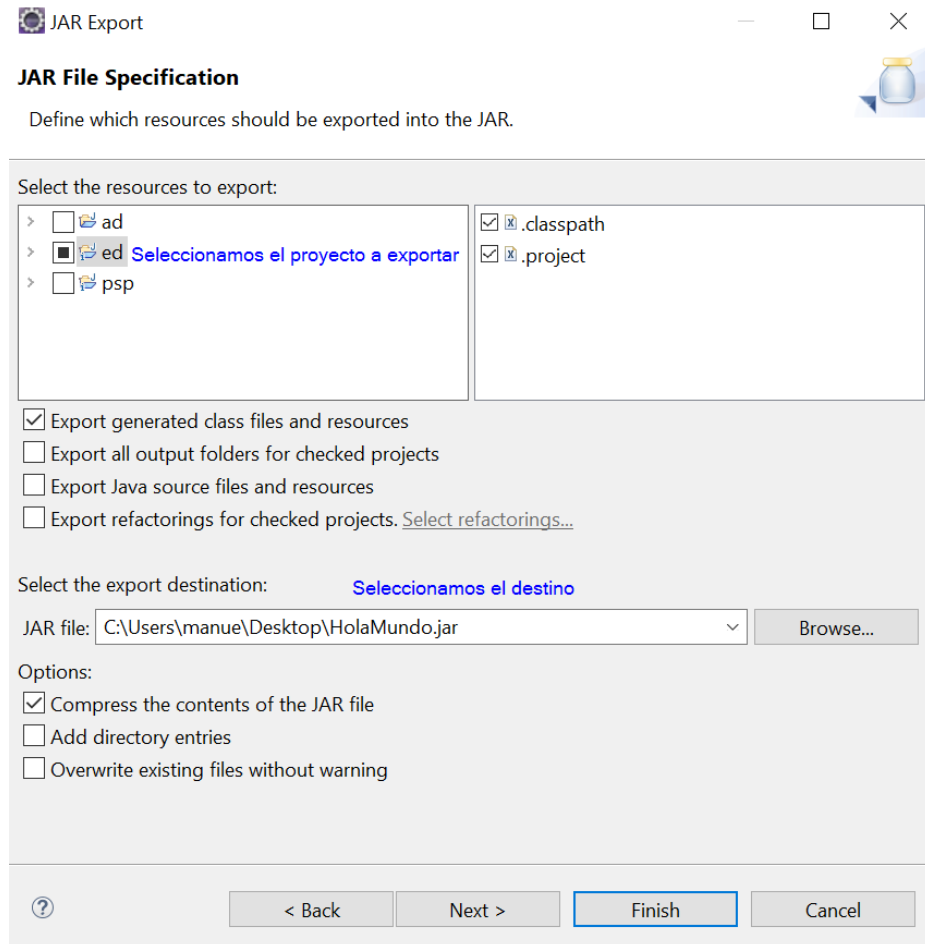
<https://www.jetbrains.com/es-es/objc/features/run-and-debug.html>

Generación de ejecutables

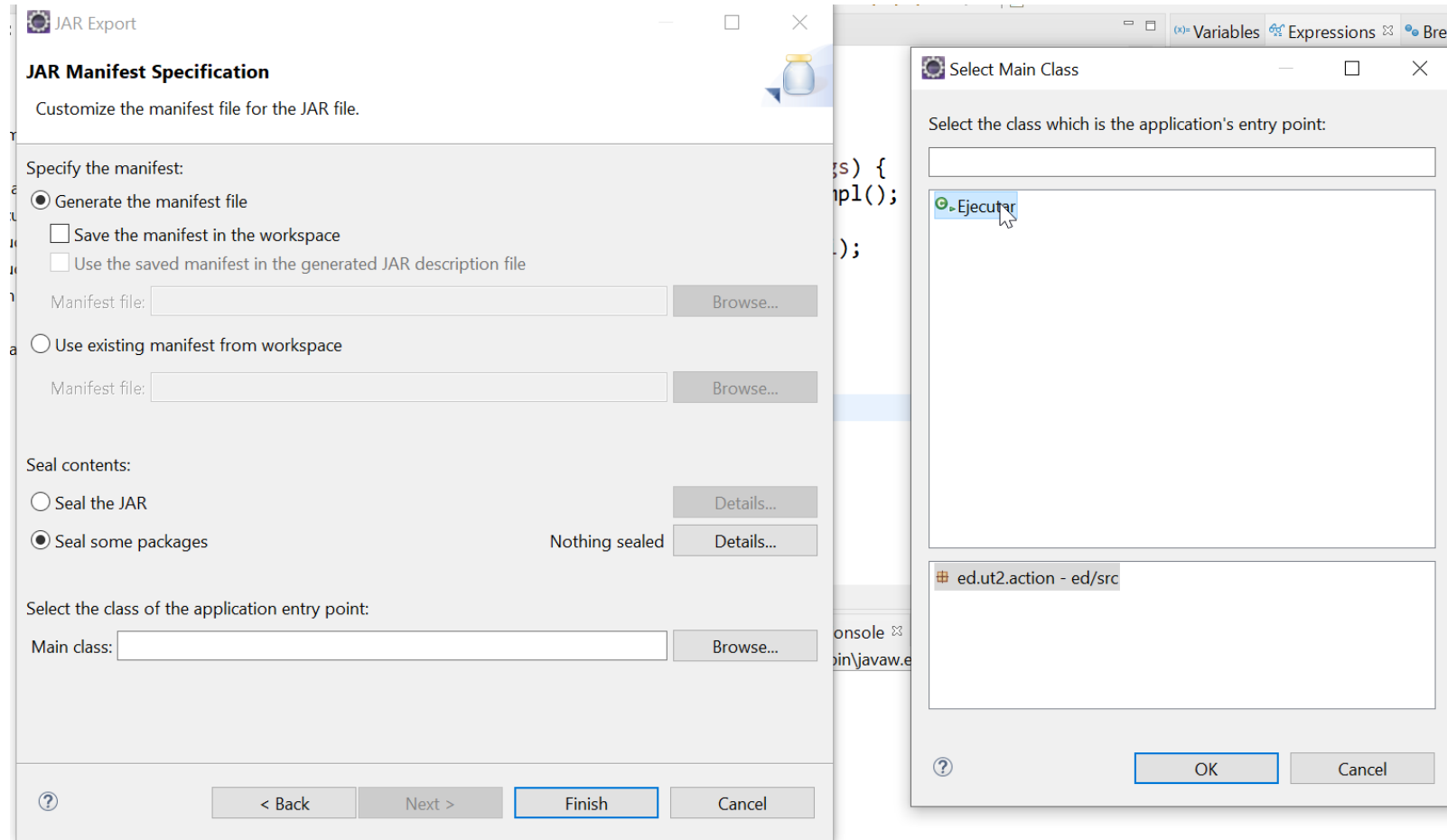
Una vez terminado nuestro proyecto java, podremos crear ejecutables para que puedan ser ejecutados sin necesidad de usar un IDE como eclipse, sino directamente desde cualquier máquina que tenga Java, independientemente del sistema operativo. Para crear un .jar ejecutable desde Eclipse pulsaremos con el botón derecho en el proyecto y seleccionaremos Export



Posteriormente seleccionamos JAR file y se nos abrirá un wizard dónde podremos exportar nuestro proyecto. Hay que pulsar Next para ir avanzando por las pantallas.



Una vez llegado al JAR Manifest Specification será necesario elegir la Main class, del proyecto



Ejecuta con doble click sobre el archivo .jar

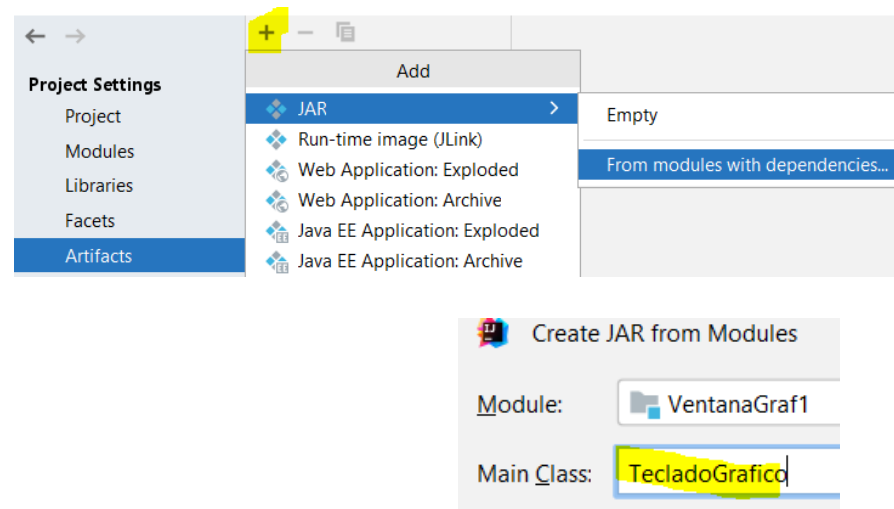
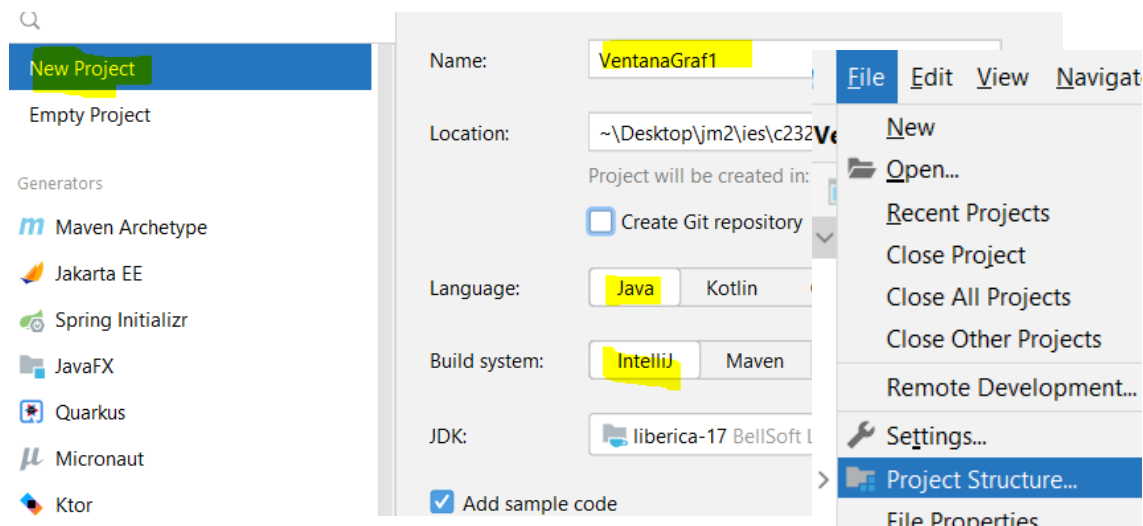
Ejecuta en línea de comandos con `java -jar nomfichero.jar`

Errores en la ejecución: Ver <https://answers.microsoft.com/es-es/windows/forum/all/no-puedo-ejecutar-archivos-jar/0f14295b-4eaf-4ba5-9d34-dc605cb5cb34>

Usaremos un proyecto de IntelliJ

File --> Project Structure. ->artifacts seleccionamos el simbolo “+” y seleccionamos el formato .jar from modules with dependencies

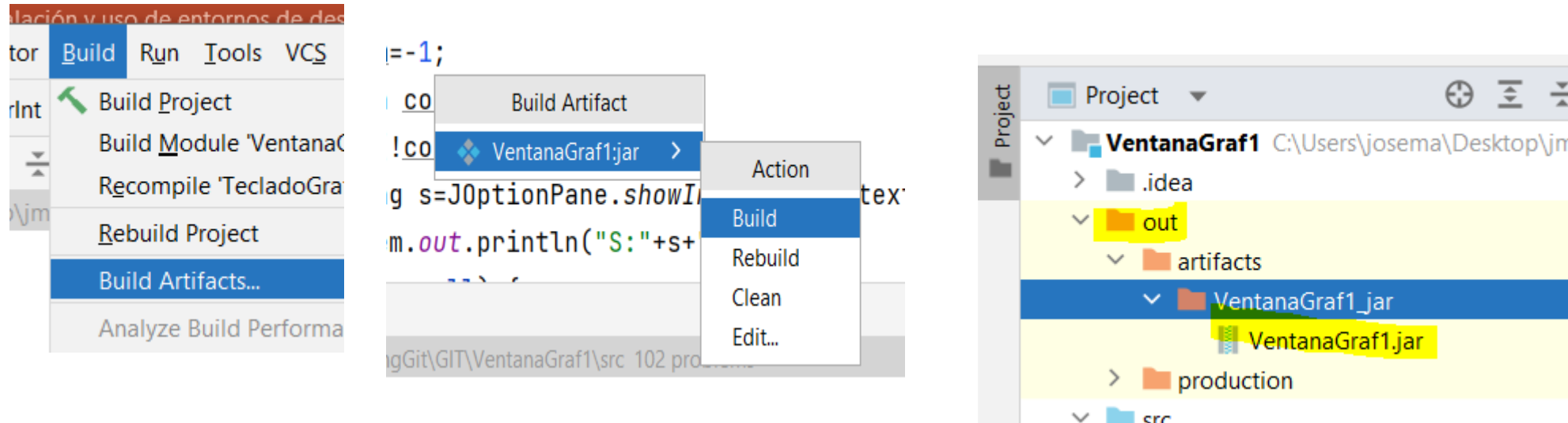
Añades la clase main y pulsas apply y ok



<https://platzi.com/tutoriales/1222-java-basico-2018/4706-como-crear-un-archivo-jar-en-intellij/>

Seleccionas Build -> Build Artifacts

En el desplegable elijes Build y se generará una carpeta out



https://www.youtube.com/watch?v=d1CT7_WZGB8

Actividad 9.a

Crea un archivo ejecutable para nuestro proyecto, utilizando la clase TecladoGrafico, en el que pidas el nombre de una persona y muestres un mensaje de Hola nombrePersona. El fichero se llamará saludo.jar y deberá ser ejecutable al hacer doble click sobre él.

Actividad 9.b**Actividad 9.b**

Crea un fichero .jar ejecutable que, utilizando la clase TecladoGrafico, pida por pantalla el nombre de una persona y después pida las notas de ese alumno, con una ventana de botones o un menú, hasta que se seleccione la opción de FIN. Después mostrará la nota del alumno, su nota máxima y su nota mínima.

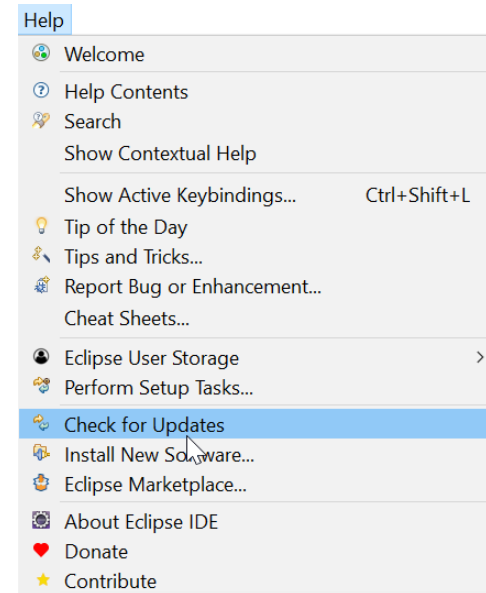
Actualización y mantenimiento

IDE

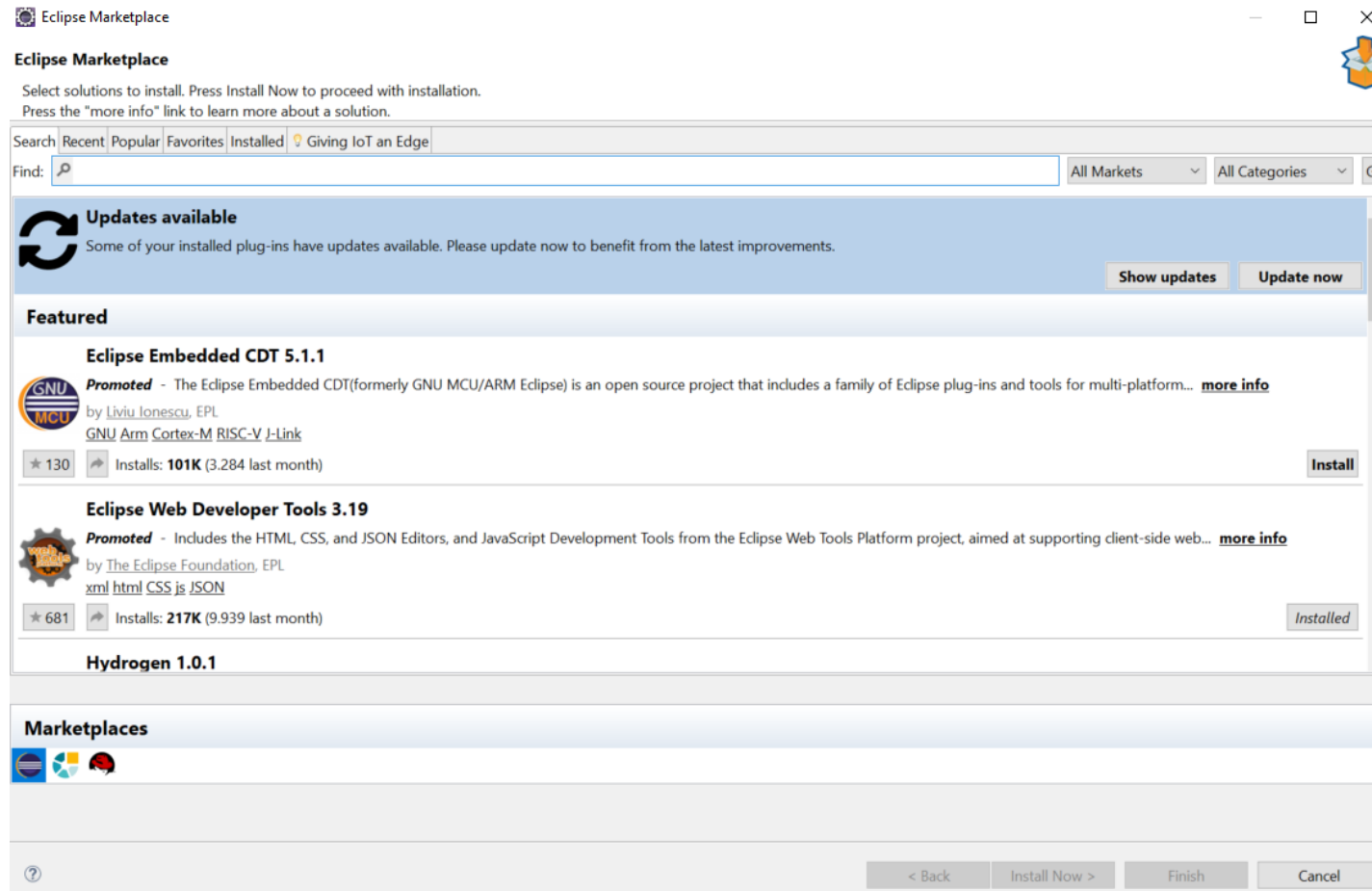
El mantenimiento de un entorno de desarrollo es una tarea fundamental que requiere tener todos sus componentes periódicamente actualizados.

También es de vital importancia realizar copias de seguridad sobre las bases de datos de nuestros proyectos, si la tuviéramos, por si ocurriera algún error o proceso defectuoso poder restaurarlos.

Para comprobar en el eclipse la actualizaciones pendientes del software base basta con usar la opción Help > Check for updates



Si queremos actualizar los plugins deberemos acceder a eclipse Marketplace y allí se nos indicará si hay actualizaciones disponibles. Bastará con pulsar sobre Update now para iniciar el proceso de actualización, el cual será similar al de instalación de un plugin



El proceso de actualización constará de los siguientes pasos:

- Seleccionamos los plugins a actualizar y pinchamos en **Next**.
- Nos mostrara lo que contiene el plugin.
- Nos pide que aceptemos los términos de licencia.
- Comenzará la actualización.
- Cuando termine nos pedirá que reiniciemos el programa.
- El plugin ya estará actualizado y listo para ser usado.

Actividad 10

- Verifica si existe una versión del software eclipse superior.
- Verifica si existe alguna actualización pendiente de algún plugin, en caso de que exista actualízalo.

Actividad 12

Instala otro entorno de desarrollo de entre los siguientes, u otro que propongas y sea aceptado por el profesor, y realiza una presentación de su funcionamiento básico (al menos crear aplicación java, ejecutar y depurar). Deberás exponerla en clase a tus compañeros. Puedes hacer un vídeo o entregar la documentación en un fichero de texto (Word, powerpoint o similar). El trabajo será en grupos de 2.

- Netbeans
- Jbuilder
- Visual Studio Code
- IntelliJ IDEA
- BlueJ
- XCode

otros lenguajes:
[runjs](#) (javascript)