TEMA 2:

Entornos Integrados de Desarrollo.

Ciclo de Desarrollo de Aplicaciones Web

Módulo: Entornos de Desarrollo

Fray Paulino s/n 33600 MIERES - ASTURIAS Tfno. 985464982 Fax 985452969 / www.ibq.es ibq@ibq.es

C.I.F.: Q-3368149E

1- Introducción

Un entorno de desarrollo integrado (*Integrated Development Environment o IDE*) es un programa compuesto por una serie de herramientas que utilizan los programadores para desarrollar código.

Estos entornos nos permiten realizar diferentes tareas:

- Crear, editar y modificar el código fuente del programa.
- Compilar, montar y ejecutar el programa.
- Examinar el código fuente.
- Ejecutar el programa en modo depuración.
- Generar documentación.
- Realizar control de versiones.
- Etc.

La mayoría de los IDEs actuales proporcionan un entorno de trabajo visual formado por ventanas, barras de menús, barras de herramientas, paneles laterales para presentar la estructura en árbol de los proyectos o del código del programa que estamos editando, etc. Los editores suelen ofrecer facilidades como el resaltado de la sintaxis utilizando diferentes colores y tipos de letra, etc.

Un mismo IDE puede funcionar con varios lenguajes de programación, este es el caso de Eclipse, Netbeans, IntelliJ IDEA o, incluso, **Visual Studio Code**, que mediante la instalación de plugins proporcionan soporte a lenguajes adicionales.

Generalmente, un IDE consta de los siguientes componentes:

- **Editor**. Generalmente, se utilizan editores que colorean la sintaxis para ayudar al programador a comprender mejor el programa y detectar los errores más fácilmente.
- **Compilador o intérprete**. Dependiendo del tipo de lenguaje utilizado, se necesitará para ejecución el intérprete o el compilador para generar código ejecutable o bytecode.
- **Depurador (Debugger)**. Es una funcionalidad de los IDEs que nos permite probar y depurar (eliminar los errores) de nuestros códigos. Para ello, permite ir ejecutando las instrucciones paso a paso, inspeccionar el valor de variables, establecer que se pare al llegar a determinada instrucción o cuando cierta variable alcance un valor concreto, etc.
- **Constructor de interfaces gráficos.** Con él, el desarrollador podrá crear de una manera visual (pinchando y arrastrando el componente) ventanas, botones, campos de texto, literales, pestañas, tablas, etc. Tiene todos los componentes que pueden encontrarse en una interfaz.
- **Refactorización (Refactoring)** de código. La Refactorización es una técnica de la ingeniería de software para reestructurar un código fuente, alterando su estructura interna (para que sea más eficiciente) sin cambiar su comportamiento externo.

2- Los primeros Entornos de Desarrollo

2.1. 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.

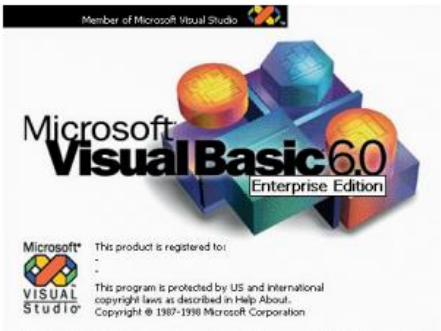
Se lanzaron siete versiones y, **en las últimas, podía utilizarse el ratón**. Soportaba múltiples archivos en el mismo editor (diferentes ventanas) y podía programarse orientado a objetos.

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.

2.2. Visual Basic 6

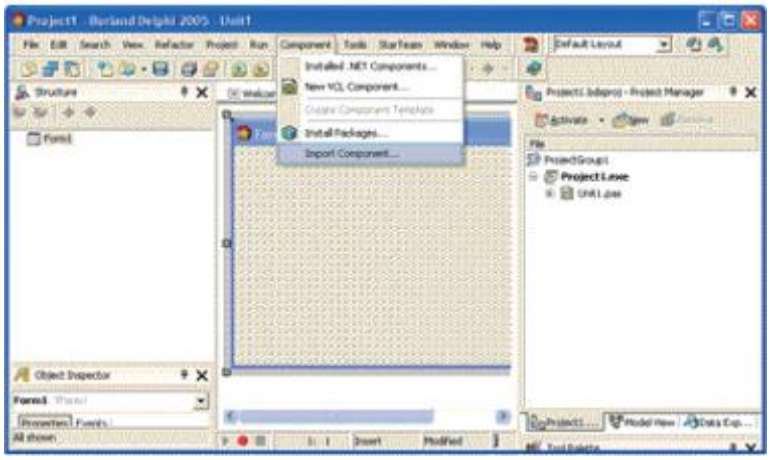
Visual Basic 6 fue uno de los IDE más utilizados en su época, si no el que más. Este nuevo tipo de herramientas creó el **paradigma de desarrollo RAD**, acrónimo del inglés rapid application development (desarrollo rápido de aplicaciones). Un paradigma en el que primero se desarrollaban de una manera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

2.3. 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.

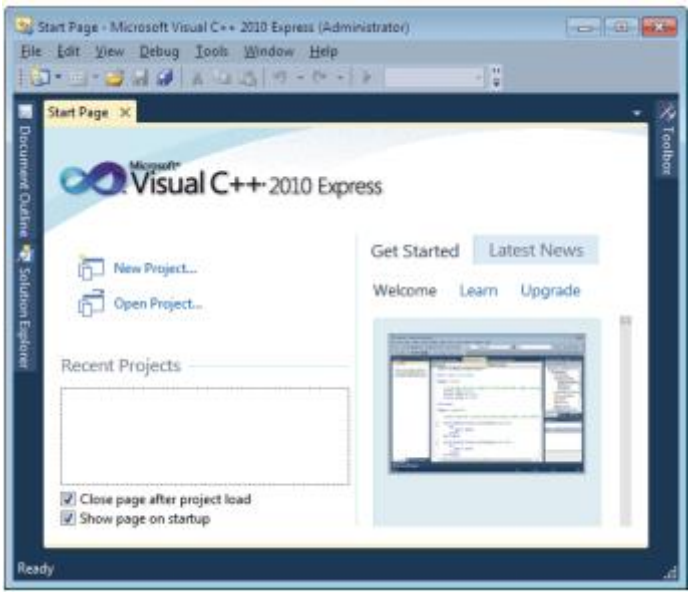


Además de Delphi, Borland también saco al mercado el JBuilder. Un IDE de Java que tenía la ventaja de estar disponible también en Linux (hoy en día, el IDE Java dominantes es **IntelliJ IDEA** de Android Studio)**.**

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).

2.4. Visual C++

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 añadírsele 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.

3- Entornos de desarrollo actuales

3.1. Xcode

Xcode es la herramienta para realizar aplicaciones (app) para dispositivos Apple. Con esta herramienta, podrán realizarse aplicaciones nativas para iOS, macOS, watchOS o tvOS.

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 100 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 Objetive C. Objetive C es un lenguaje parecido a Java/C/C++, pero con una sintaxis algo diferente. Muy potente y orientado a objetos.

3.2. NetBeans

Ahora es **Apache NetBeans y está escrita en Java**, lo que la da cierta versatilidad y 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, HTML 5, CSS, PHP o incluso C++.

3.3. Eclipse

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).

En el editor, puede programarse en Java,C/C++, Python, PHP, Ruby o Perl.

3.4. IntelliJ IDEA

IntelliJ IDEA es un IDE de JetBrains centrado en el ecosistema **JVM** (Java, **Kotlin**, Scala…).

Destaca por sus **refactorizaciones seguras**, **inspecciones de código** y navegación inteligente. Existe edición **Community** (gratuita) y **Ultimate** (de pago) con soporte avanzado para frameworks web y bases de datos. **Android Studio** está basado en IntelliJ.

Características destacadas:

- Depurador avanzado, cobertura de tests, y runners para **JUnit/TestNG**.
- Soporte para **Spring**, Jakarta EE, Micronaut, Quarkus (Ultimate).
- Herramientas para **SQL/DB** (Ultimate) y cliente HTTP integrado.
- **VCS** integrado (Git, GitHub, GitLab) y **Marketplace** de plugins.
- Integración con **Maven/Gradle**, gestor de dependencias y perfiles de compilación.
- Atajos, "**intenciones**/bombillitas" y **auto-imports** que aceleran el desarrollo.

3.5. Visual Studio Code

Visual Studio Code (VS Code) es un **editor extensible** que, mediante el **Language Server Protocol (LSP)** y su **Marketplace**, puede comportarse como un **IDE ligero**. Es multiplataforma (Windows, macOS, Linux) y especialmente ágil para **desarrollo web** (HTML, CSS, JavaScript/TypeScript) y múltiples lenguajes mediante extensiones.

Características destacadas:

- Editor potente con **resaltado**, **autocompletado**, **refactorización** y **Go to Definition**.
- **Depurador** integrado (navegador/Node, Python, C/C++… según extensión).
- **Terminal** integrada y tareas de construcción (npm, Maven/Gradle, etc.).
- **Git** integrado (commits, ramas, PRs con extensiones).
- Extensiones recomendadas para **LMSGI**: *Live Server/Live Preview*, **Prettier**, **HTMLHint**, **stylelint**, **ESLint** (si hay JS), **XML Tools**.
- Se integra con los **runtimes**/compiladores externos (no los incluye): Node.js, Java, Python, etc.

Nota: VS Code no es un IDE "monolítico" como Eclipse/IntelliJ; su fuerza está en **extensiones + LSP** para convertirlo en el IDE que necesites.

3.6. Entornos de desarrollo online

Los entornos de desarrollo online o en la nube están extendiéndose cada vez más. Pese a la desventaja de la potencia, poseen muchas otras ventajas como el trabajo colaborativo, los repositorios comunes, el poder trabajar con cualquier dispositivo, etc. Estas ventajas hacen que muchos desarrolladores y empresas de desarrollo opten por entornos en la nube.

Por ejemplo, **AWS Cloud9** es un entorno de desarrollo integrado (IDE) en la nube que se puede utilizar para escribir, ejecutar y depurar código directamente en el navegador. **Es un IDE multilenguaje que es compatible con más de 40 lenguajes, incluidos Java, JavaScript, Python, PHP, Ruby, Go y C++**.

¿Qué inconvenientes puede tener para una empresa *trabajar en la nube*?

3.7. Entornos de desarrollo libres y propietarios

Existen muchos IDE, dependiendo de la popularidad del lenguaje, habrá más o menos opciones. En el siguiente cuadro, se ofrece una lista de IDEs para los lenguajes Java y JavaScript

	IDE	Licencia	Windows	Linux	Mac OS X
Lenguaje Java	Eclipse	EPL	Sí	Sí	Sí
	NetBeans	CDDL/GPL2	Sí	Sí	Sí
	Visual Studio	Propietario	Sí	No	No
	JDeveloper	Propietario	Sí	Sí	Sí
Lenguaje JavaScript	Eclipse	EPL	Sí	Sí	Sí
	NetBeans	CDDL/GPL2	Sí	Sí	Sí
	Geany	GPL	Sí	Sí	Sí
	KDevelop	GPL	No	Sí	No
	JBuilder	Propietario	Sí	Sí	Sí
	JCreator	Propietario	Sí	No	No
	JDeveloper	Propietario	Sí	Sí	Sí

EPL - Eclipse Public License – Licencia de código abierto utilizada principalmente para distribuir el IDE Eclipse.

CDDL (Common Development and Distribution License) – Licencia de código abierto que se utiliza fundamentalmente para productos Sun Microsystem y Oracle.

GPL (Licencia Pública General de GNU) – Es la licencia más utilizada para distribuir software libre.

4- ECLIPSE

Eclipse es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores.

> **NOTA:** Las "Aplicaciones de Cliente Enriquecido" se refieren a aplicaciones de software que se ejecutan en la computadora del usuario final y suelen ser más potentes y completas.
>
> Por otro lado, las aplicaciones "Cliente-liviano" se ejecutan en un navegador web y dependen en gran medida de los recursos y capacidades del servidor web.

Esta plataforma, típicamente ha sido usada para desarrollar aplicaciones en lenguaje de programación JAVA para lo cual nos proporciona el Java Development Toolkit (JDT) y el compilador ECJ (que realmente forma parte de JDT), que se entregan como parte de Eclipse en sus últimas versiones.

ACTIVIDAD: ¿Sabrías decirme la diferencia entre el JDT y el JDK? En caso negativo, investiga por tu cuenta para obtener la respuesta. ¿Cuál crees que contiene al otro?

4.1. Instalación

La herramienta puede descargarse de http://www.eclipse.org/downloads/

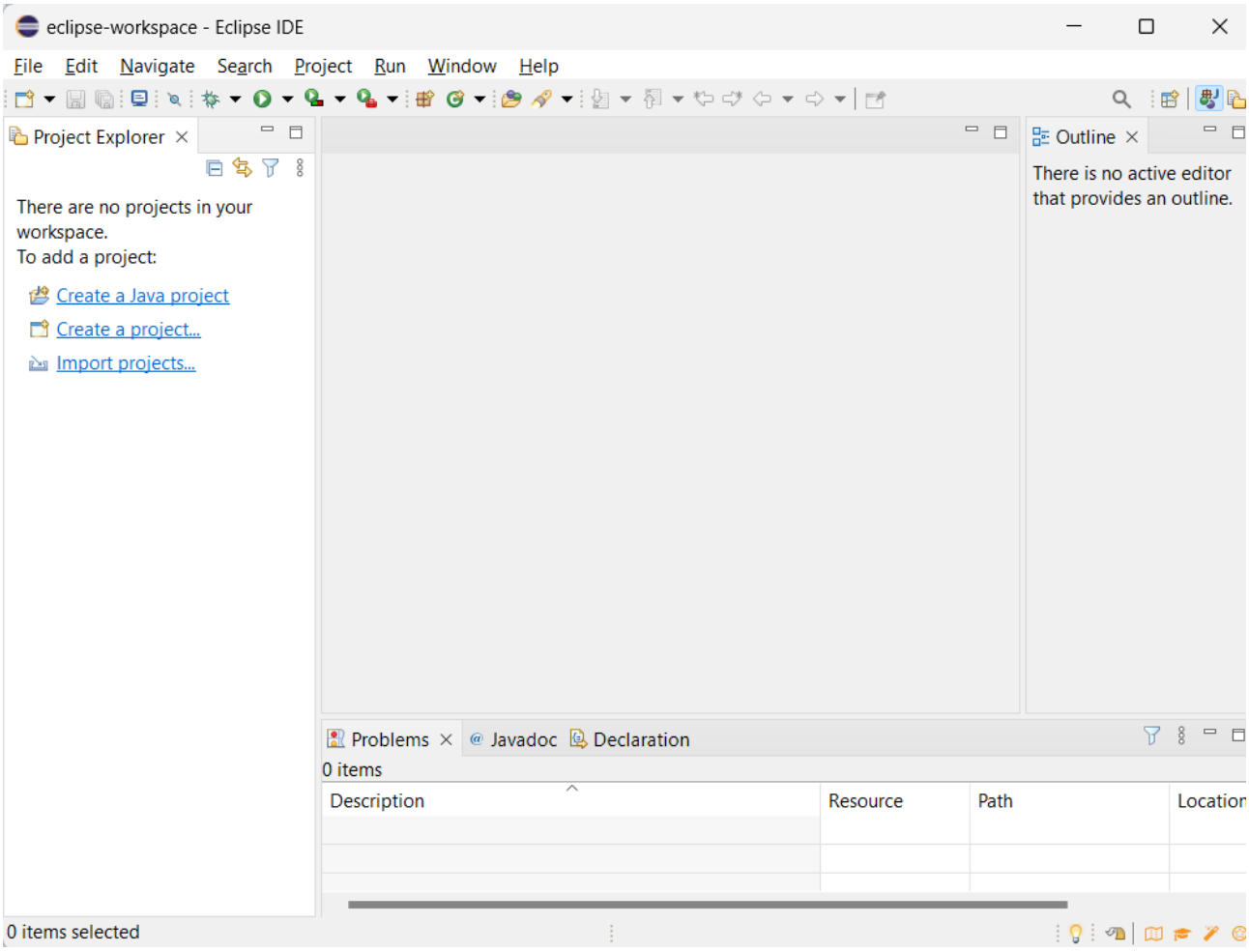
El siguiente paso es descomprimir el archivo en el directorio de destino. No es necesaria la instalación y puede iniciarse la herramienta haciendo doble click en .\eclipse\eclipse.exe

Eclipse almacena los proyectos en espacios de trabajo (Workspace). Se puede seleccionar uno existente o crear una carpeta nueva para asignar la ubicación. Lo que aparece después es el entorno de trabajo (Workbench).

El entorno dispone de varias configuraciones visuales o perspectivas (perspectives). Arriba a la derecha puede cambiarse la perspectiva en uso (**por defecto es java**). Si quiere abrirse otra, hay un icono para hacerlo: 

Seleccionamos, por ejemplo, la perspectiva "Resource" (destinada principalmente a la gestión de recursos y archivos en tus proyectos) y veremos como la disposición de los marcos (partes) de trabajo cambia. Esas partes pueden ser vistas o editores. **Las vistas son componentes visuales (es decir, componentes que me visualizan "algo").**

En cada momento sólo habrá una parte activa, mostrándose en azul su título.



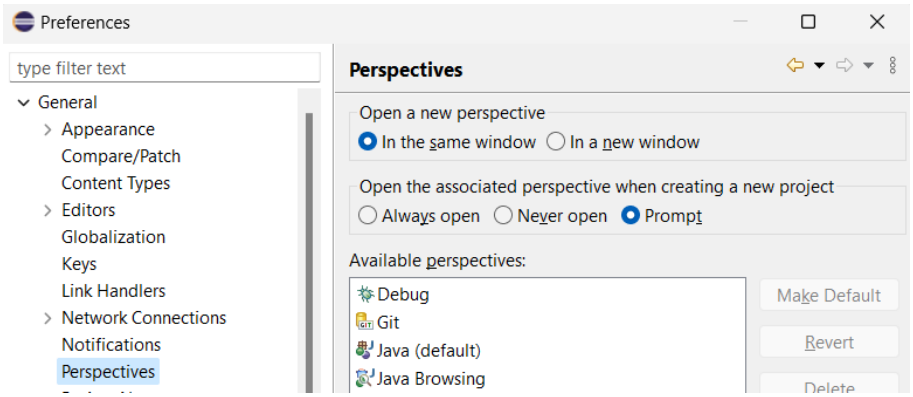
4.2. Perspectivas

Constituye un conjunto de vistas y editores visibles con una particular disposición acorde con el objetivo de cada perspectiva. Cada perspectiva está diseñada para una tarea específica, como desarrollo de Java, depuración, modelado UML, desarrollo web, etc.

Por ejemplo, si seleccionas la perspectiva "Java", Eclipse mostrará vistas como el "Explorador de Paquetes", el "Explorador de Tipos" y el "Editor de Java" para ayudarte en el desarrollo de aplicaciones Java. Las perspectivas **Java, Debug, Git, Java EE/Web** son las más útiles.

Puede abrirse una perspectiva desde el botón de la barra de herramientas o desde *Window → Open Perspective*. Al abrir la nueva perspectiva se abrirá en el entorno de trabajo (Workbench). Puede

configurarse las *Preferencias* para que se abra en un nuevo Workbench (*Window → Preferences → General → Perspectives*):



El usuario puede disponer las vistas a su gusto y crear una nueva perspectiva, guardándola desde *Window → Perspective → Save Perspective As...*

Además es posible configurar los iconos visibles en la barra de herramientas, las opciones de la barra de menús y sus desplegables, etc, en *Window → Perspective → Customize Perspective*

Window → Perspective → Reset Perspective restaura la configuración de la perspectiva actual.

4.3. Vistas

Una vista es una ventana específica dentro de una perspectiva que muestra información o funcionalidad particular relacionada con la tarea en curso.

Las vistas muestran información de una manera organizada y se pueden mover, redimensionar, apilar o acoplar dentro del entorno de Eclipse según tus preferencias.

Ejemplos de vistas incluyen el "Explorador de Proyectos", "Explorador de Paquetes", "Consola", "Depurador" y "Consola de Git". Cada una de estas vistas proporciona información o funcionalidad específica para tareas de desarrollo particulares.

Puedes abrir, cerrar, mover y personalizar las vistas según tus necesidades y preferencias de diseño de la interfaz de usuario.

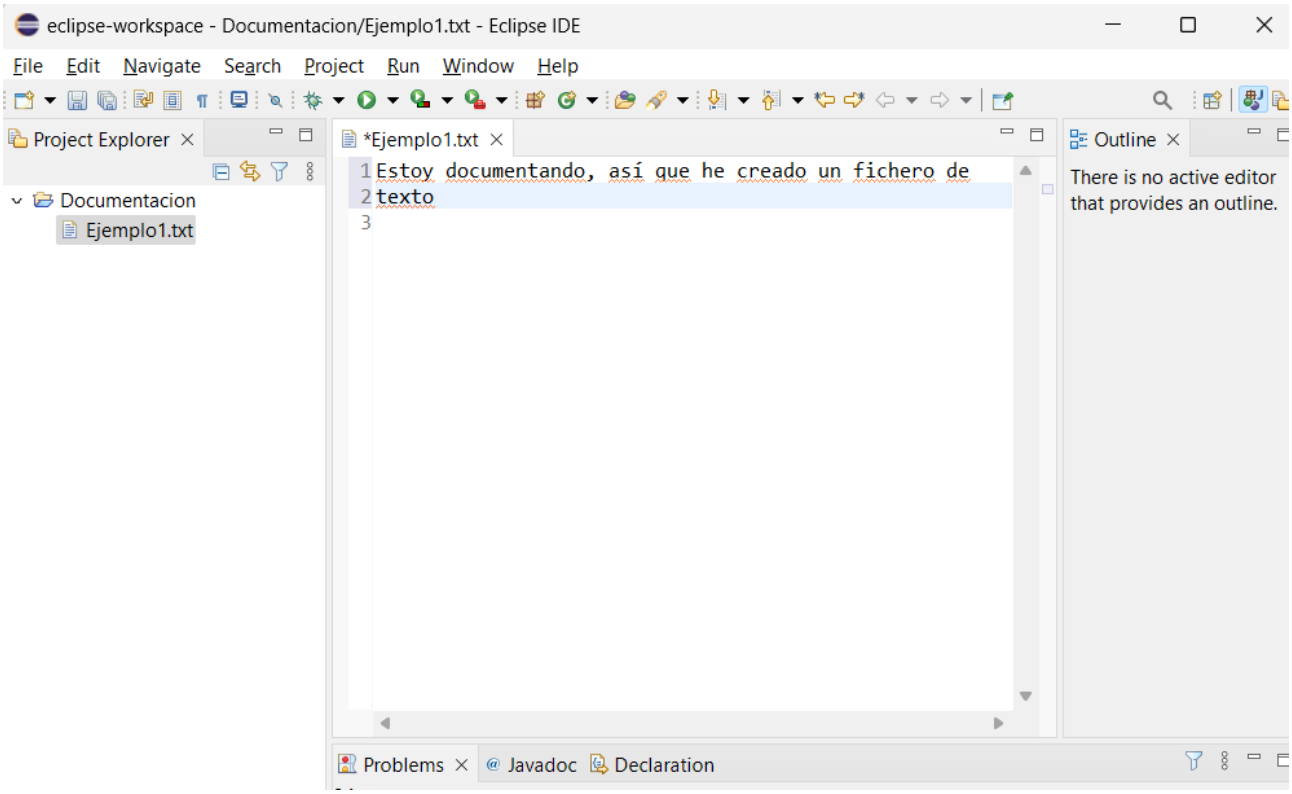
Windows → Perspective → Reset perspective retorna la apariencia a la colocación de vistas por efecto.

La lista de todas las vistas posibles aparecen en *Window → Show View* o pulsando en el icono de la esquina inferior izquierda de la pantalla.

4.4. Editores

Dependiendo del archivo a editar se usará el editor adecuado a la tarea.

En la barra de título, si aparece un asterisco "*" al lado del nombre de archivo, indica que el archivo en edición tiene cambios pendientes de grabarse en disco.



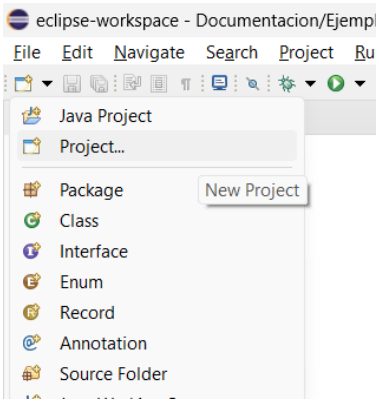
Si se abren varias partes con editores, éstas pueden colocarse de diversas maneras pulsando la barra de título y arrastrando a una nueva posición.

Puede navegarse a través de las diferentes editores de modo circular con Ctrl+F6.

Puede intentarse abrir cualquier tipo de archivo. Si Eclipse no dispone de un editor propio, abrirá el que tenga asociado el Sistema Operativo.

4.5. Proyectos

Un proyecto es un contenedor lógico que agrupa y organiza todos los archivos y recursos relacionados con un proyecto de desarrollo de software en particular. Es decir, una carpeta dentro de la cual podemos crear otras carpetas y archivos, y en la que iremos guardando todo lo que vayamos creando dentro de nuestro proyecto software.



Un proyecto puede ser creado usando el menú *File → New → Project* o directamente desde el Botón de *New*:

4.6. Navegando entre recursos

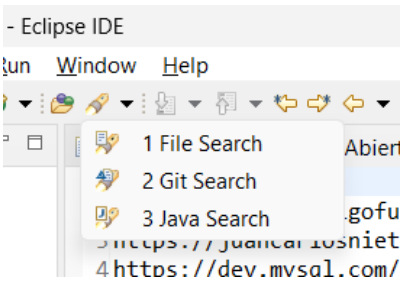
Los proyectos, carpetas y archivos se llaman en conjunto "recursos".

Un recurso puede abrirse seleccionándolo en la vista de navegación (*Project Explorer*) y haciendo doble click o bien desplegando con el botón derecho y eligiendo con qué se abre (puede elegirse incluso una aplicación externa). Las asociaciones de archivos y programas se pueden modificar en *Window → Preferences → General → Editors → File associations*.

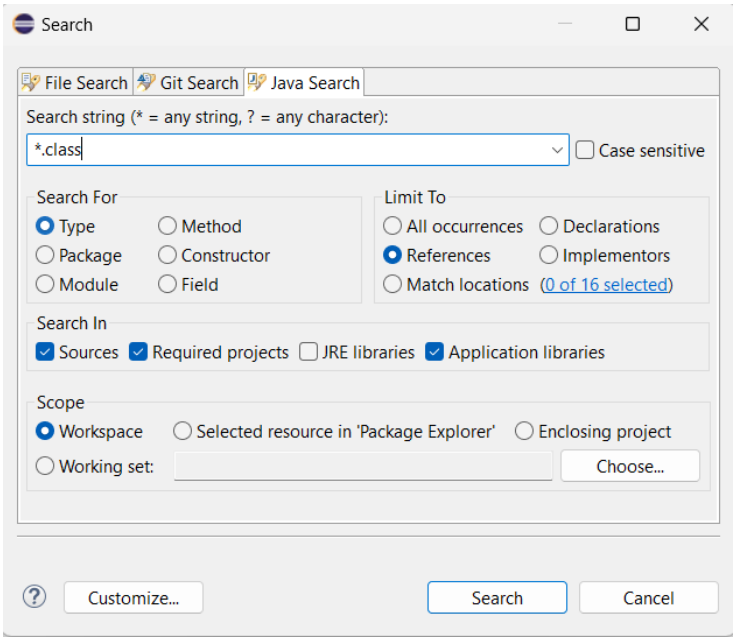
Cuando se ha trabajado con diversos archivos o recursos, es posible pasar de uno a otro en la opción *Navigate → Go to → Resource*.

Todos los recursos se localizan físicamente en el directorio que se indicó al iniciar Eclipse. Si no se recuerda la ubicación o se desea añadir otra ubicación o cambiarla, se debe hacer en *File → Switch Workspace*. IMPORTANTE: tras grabar la localización, se debe pulsar Cancel para cerrar el diálogo, o Eclipse se cerrará y se reabrirá en el nuevo espacio de trabajo.

Existe una herramienta para buscar recursos, accesible desde la barra de herramientas:



Pueden hacerse búsquedas usando patrones o expresiones regulares tanto de nombre de recurso como de texto o elementos contenidos en ellos. En el tab "Java search" se permite la búsqueda de elementos lógicos de las clases, por ejemplo métodos, paquetes, constructores, etc



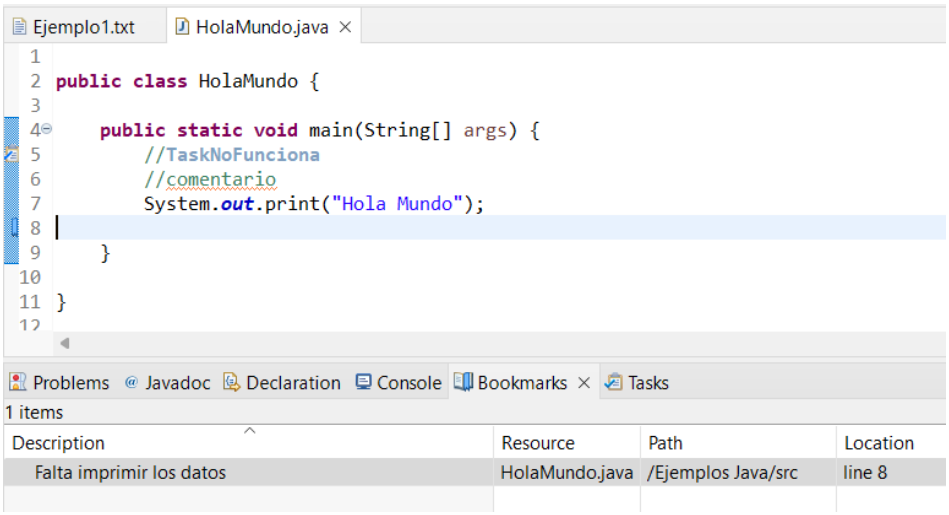
4.7. Task y Bookmarks (Tareas y Marcadores)

Tanto "Task Tags" como "Bookmarks" son funcionalidades que te ayudan a administrar y seguir tu código, pero tienen propósitos ligeramente diferentes:

Las "Task Tags" son anotaciones que te permiten marcar líneas de código con comentarios especiales que indican tareas pendientes, problemas o recordatorios relacionados con ese código. Eclipse identifica estos comentarios como tareas y las muestra en la vista "Tasks" (Tareas):

Por lo tanto, **son útiles para llevar un registro de las áreas del código que requieren atención o trabajo adicional**.

Por su lado, los "Bookmarks" en Eclipse son una forma de marcar ubicaciones específicas en tu código para que puedas volver a ellas más tarde. Son comentarios personales que dejamos en nuestros códigos: no viajan con el código a Git, por ejemplo. Los marcadores **son útiles cuando deseas regresar a ubicaciones específicas en tu código de manera rápida y fácil**. Puedes ver y gestionar tus marcadores en la vista "Bookmarks" (Marcadores):



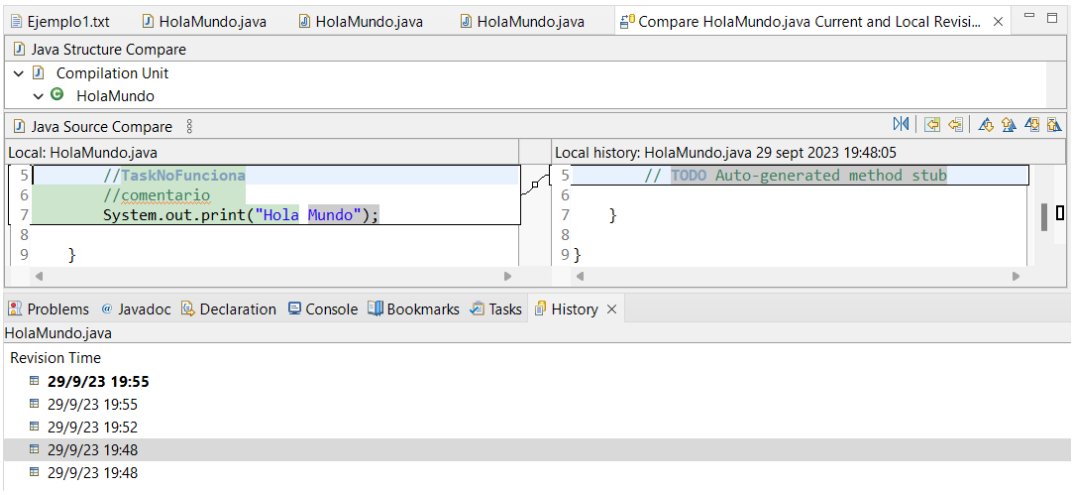
Por lo tanto, las "Task Tags" se utilizan principalmente para identificar tareas pendientes o problemas en tu código, mientras que los "Bookmarks" se utilizan para marcar ubicaciones específicas que deseas volver a visitar. Ambas características son útiles para la gestión y navegación del código en Eclipse.

4.8. Historial de Versiones

Eclipse proporciona una funcionalidad bastante potente en cuanto a la gestión de versiones. Son copias locales que Eclipse guarda en tu workspace cada vez que guardas un archivo. Es independiente de Git/EGit.

Puede verse el historial de modificaciones de un fichero pulsando sobre él botón derecho *Team → Show Local History*

En él se muestra cada modificación hecha como una entrada en una línea diferente. La última entrada es el estado actual del fichero. Puede verse el estado del fichero en cada entrada del historial haciendo doble click sobre la entrada (se abrirá una nueva vista). Si se quieren comparar los cambios del fichero en un instante dado con respecto al estado actual, en la entrada correspondiente, en el menú contextual puede seleccionarse la opción "*Compare current with Local*" → se abrirá la ventana de comparación de ficheros y podrán recuperarse selectivamente las líneas que se requiera:

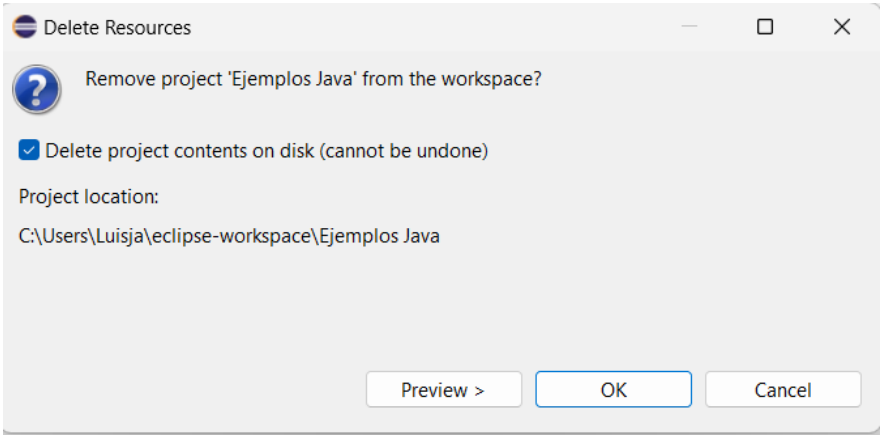


Ojo, la opción "Get contents" sustituye la versión actual por la que se haya seleccionado.

4.9. Gestión de los elementos de un proyecto

Ahora vamos a ver como borrar los proyectos es bastante sencillo, lo único que debemos hacer es seleccionar aquellos proyectos que queremos borrar y pulsar el botón *supr* o, con el botón secundario, pinchamos en *delete*. De la misma forma, podemos borrar ficheros .java específicos, pero en lugar de los proyectos seleccionamos el fichero a eliminar.

NOTA: es muy importante seleccionar la opción que nos indica al borrar, ya que si no lo hacemos, se quedara el proyecto en nuestro disco duro y cuando creemos un proyecto que se llame de la misma manera no nos dejara crearlo.

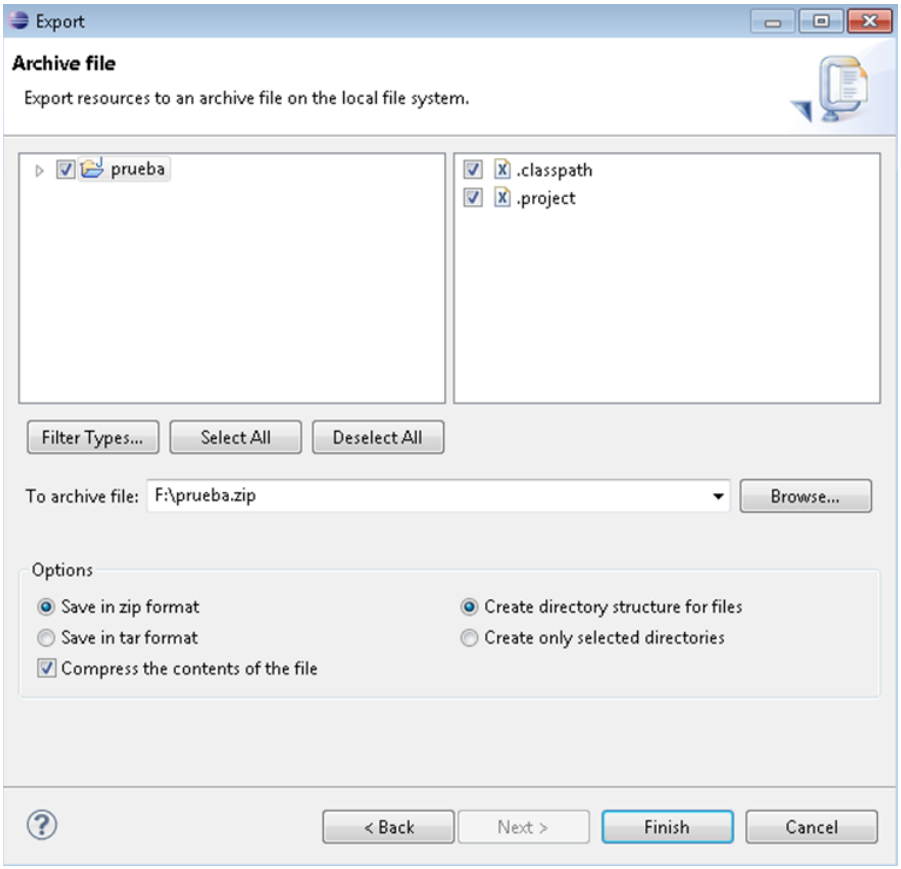


También podemos **copiar ficheros** de un proyecto a otro, arrastrando ese fichero a donde quieres copiarlo. Otra forma de hacerlo es ir a la carpeta del *workspace* y copiar los ficheros **.java** al proyecto que queramos, que son como carpetas, de la misma forma que cuando copiamos un archivo a otro lugar de nuestro sistema de archivos.

4.10. Importar / Exportar proyectos

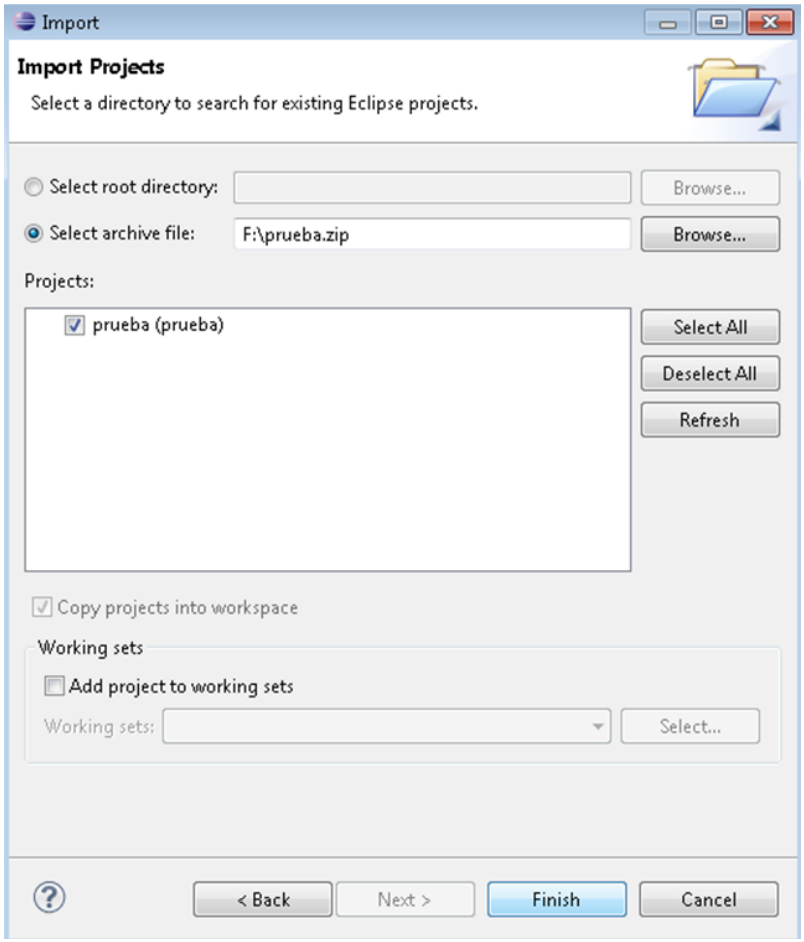
Otra acción importante es **importar y exportar proyectos**.

Imaginemos que debemos llevar una serie de proyectos hechos en un lugar a otro distinto → para ello se usa exportar. Seleccionamos los proyectos que queramos exportar, en el menú *File -> Export -> General -> Archive File*. Pinchamos en *Next* y nos aparecerá los proyectos seleccionados, en "To archive file" buscaremos la ruta donde queremos guardarlo.



En las opciones podemos seleccionar el tipo de formato y el modo de crearlo. Pinchamos en *Finish* y ya tenemos nuestros proyectos exportados, ahora vamos a importarlos como si estuvieras en un equipo diferente.

Ahora pinchamos en menú *File -> import -> Existing Projects in Workspace* y en el dialogo que nos aparecerá, elegimos la opción *Select archive file* y buscamos el archivo que antes hemos exportado, al seleccionarlo nos aparecerá todos los proyectos que contiene el fichero:



Una vez pinchemos en Finish, los proyectos se importaran a nuestro **workspace**.

OJO: Si importas copiando directamente carpetas al **workspace** con el explorador del sistema, Eclipse **no "ve"** el proyecto hasta que lo **Importas** (*Existing Projects…*).

5- JAVA en ECLIPSE

Vamos a verificar la instalación del JRE (Java Runtime Environment). Para que Eclipse funcione correctamente debemos tenerlo instalado. Para ello comprobamos *Windows → Preferences → Java → Installed JREs*

Por defecto estaremos usando el JRE proporcionado por el propio entorno de desarrollo de Eclipse (para abrir Eclipse es necesario tener instalado un JRE).

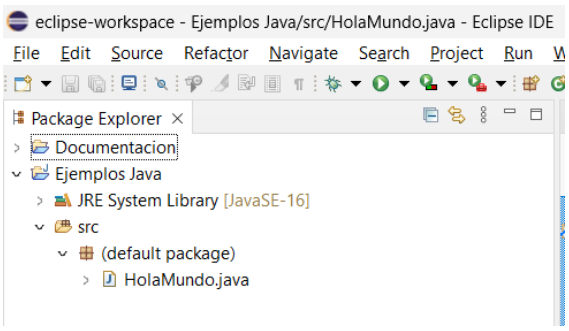
RECUERDA:

- ✓ **JDK** es el Java development kit. Es el software utilizado por los desarrolladores. Incluye el compilador de Java (javac), el Debugger y el JRE (entorno de ejecución).
- ✓ **JRE** es el Java runtime environment. Es el software utilizado por los usuarios. Este software incluye la JVM.
- ✓ **JVM** o Java virtual machine. Es el programa que ejecuta el código Java previamente compilado con el compilador de Java (javac), es decir, el bytecode.

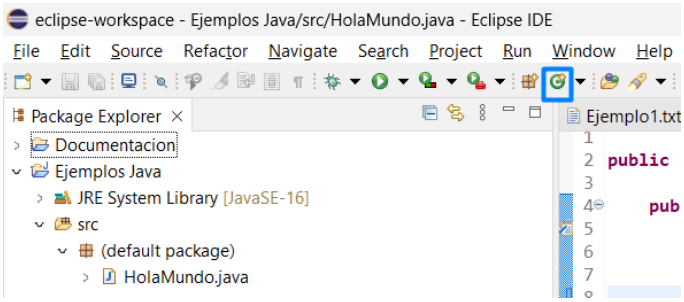
5.1.Crear un proyecto de Java

Vamos a crear nuestro primer proyecto Java, para ello vamos a *File -> New -> Java Project*, escribimos el nombre del proyecto en **Project Name,** en principio el resto de opciones no será necesario modificarlas.

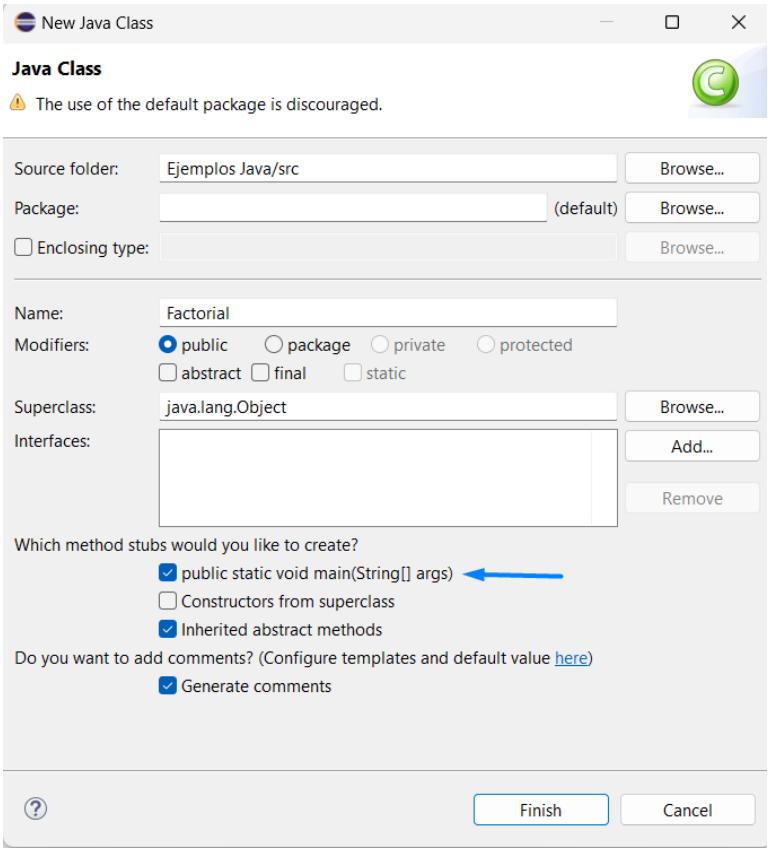
Si queremos personalizar más opciones pinchamos en *Next*. Si estamos empezando y, de momento, no tenemos mucha idea sobre las opciones avanzadas, pinchamos en *Finish*. Una vez hecho, veremos que aparece una carpeta en la parte de la izquierda: pinchamos en la flecha y nos aparecerá las subcarpetas que contiene. Escribiremos nuestros programas en Java en la carpeta **src** (source, fuente en ingles).



Para añadir una nueva clase, seleccionamos la carpeta **src** y, con el botón derecho o desde el menú *File*, vamos a *New -> Class*. También podemos pinchar en el icono de una **C** con un *mas(+)* en un circulo verde en la zona debajo del menú *File*. Recordar que debemos tener la carpeta ***src*** seleccionada.



Simplemente pondríamos un nombre a nuestra nueva clase y marcamos la opción "*public static void main (String[] args)*" para que nuestra nueva clase pueda ser ejecutada (es lo que nos interesa de momento):

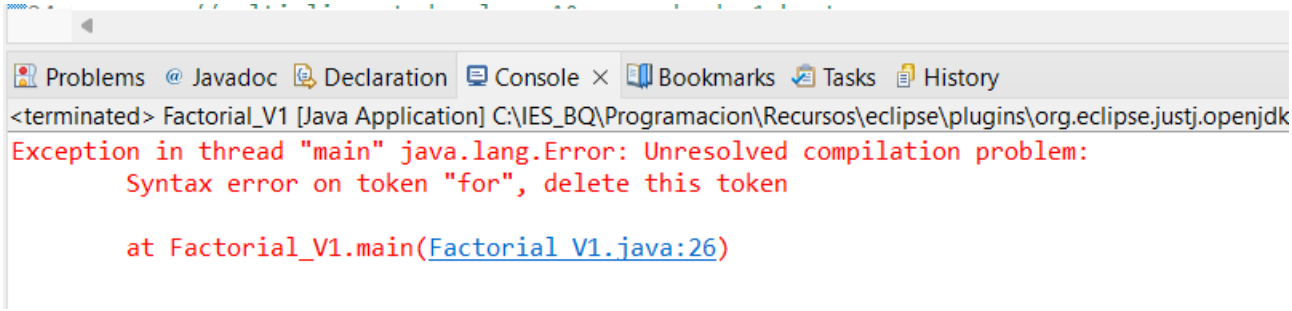


También es recomendable pinchar en la opción "*Generate comments*".

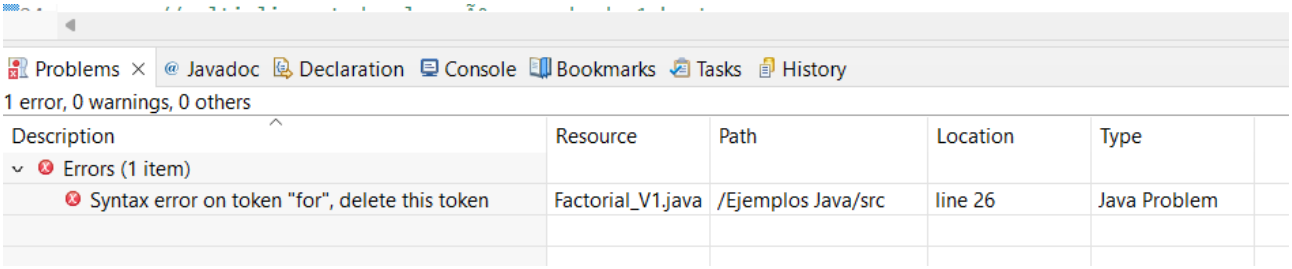
RECUERDA: Cuando escribes un programa Java en Eclipse, el código fuente se guarda en archivos con extensión ".java". Luego, cuando compiles tu proyecto, Eclipse genera los archivos ".class" correspondientes en el directorio de salida (a menudo llamado "bin" o "target") de tu proyecto. Estos archivos ".class" contienen el bytecode resultante de tu código fuente Java, y son los responsables de la portabilidad de tu código.

ACTIVIDAD: Escribe tu primer programa en Java que, como no podía ser de otra manera, simplemente mostrará el mensaje "HOLA MUNDO".

Si tenemos errores en nuestro código, el compilador de JAVA nos los mostrará cuando intentemos ejecutar el programa:



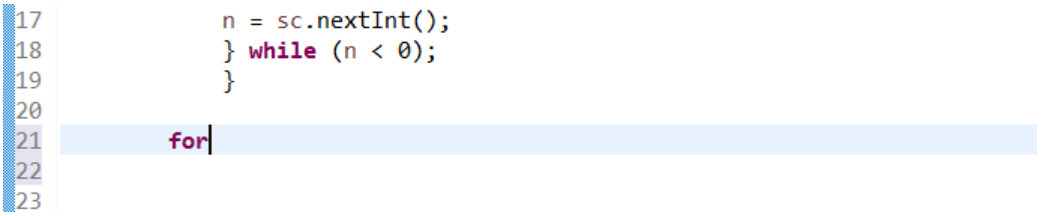
Como vemos en el ejemplo, se muestran en rojo. Podremos consultarlos en la vista *Console* y en la vista *Problems*:



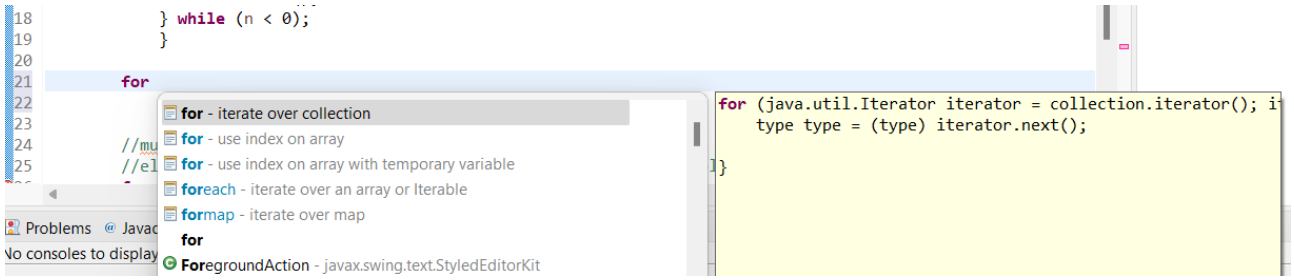
Los *warnings* (avisos) se mostrarán en amarillo.

5.2. Asistente de contenido

Eclipse te ayuda a completar los fragmentos de códigos que mas repites. Por ejemplo al momento de escribir "*for*" en una función:



Si pulsamos **Ctrl+Espacio** vamos a activar el asistente de contenido:



Se listarán las propuestas del asistente para completar el código inacabado, lo cual me va a proporcionar una interesante ayuda en línea para recordar el formato de la instrucción, en este caso un "*for*".

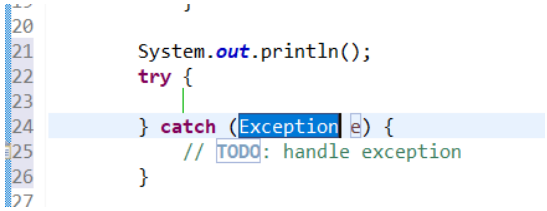
Se puede seguir tecleando código, y el asistente reducirá la lista de opciones.

5.3. Plantillas para generar código

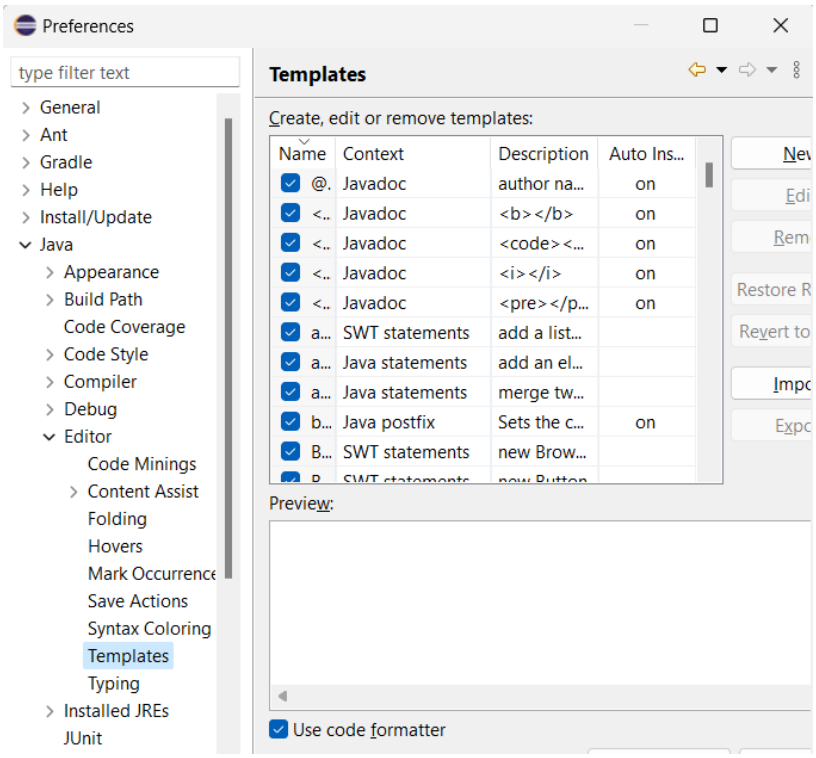
Eclipse nos permite usar palabras clave para generar automáticamente determinadas estructuras de código. **Y más aún, crearlas a nuestro gusto.**

Por ejemplo, uno de los métodos más utilizados en Java es *System.out.println()* para imprimir texto en la consola; pues nos basta con escribir en el editor "*sysout*" y pulsar el autocompletado (ctrl+espacio) para que nos inserte la línea completa y nos ponga el cursor en la posición del argumento. Lo mismo ocurre, por ejemplo, con "try" para insertar un bloque try-catch:

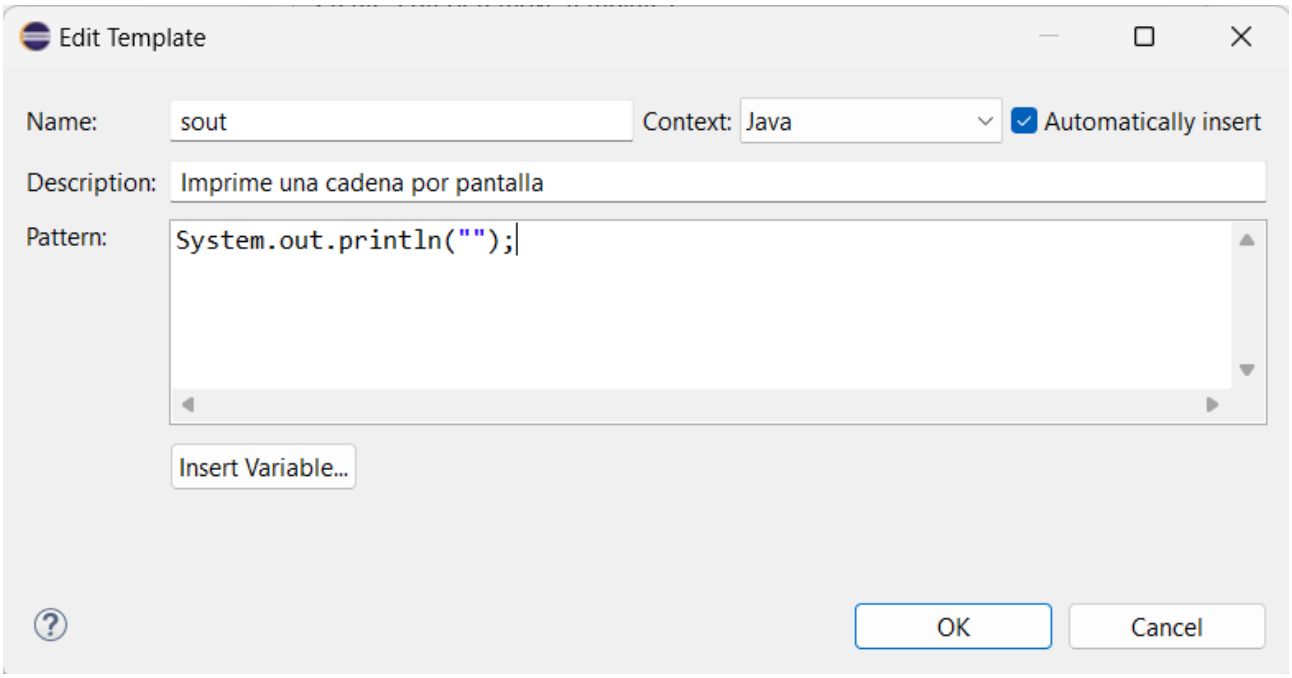
Si selecciono la propuesta que me ofrece (doble *clic*), me insertaría la estructura de código:



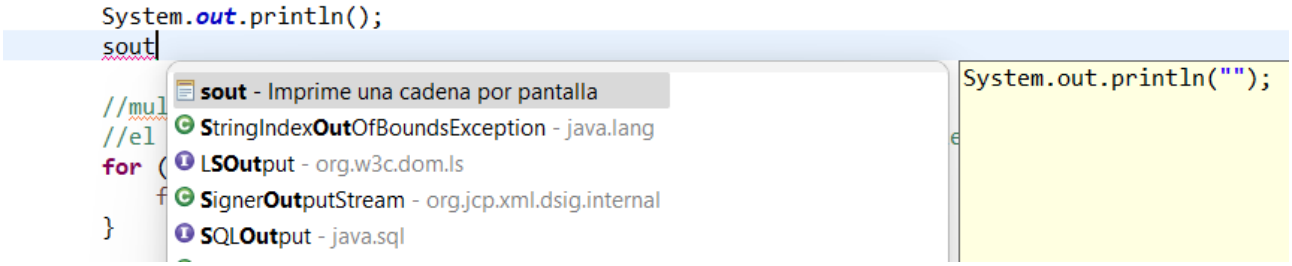
La lista completa podemos verla en *Windows/Preferences/Java/Editor/Templates* y ahí mismo podemos añadir nuestras propias plantillas:



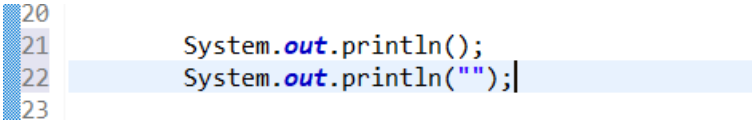
Por ejemplo, voy a crear una nueva plantilla, a la que llamare ***sout***, para imprimir cadenas por pantalla y no tener que escribir cada vez *System.out.println("");*



Ahora, puedo escribir ***sout*** y pulsar *ctrl+espacio*, y me saldrá mi nueva plantilla para seleccionarla:



La selecciono y me genera el código:

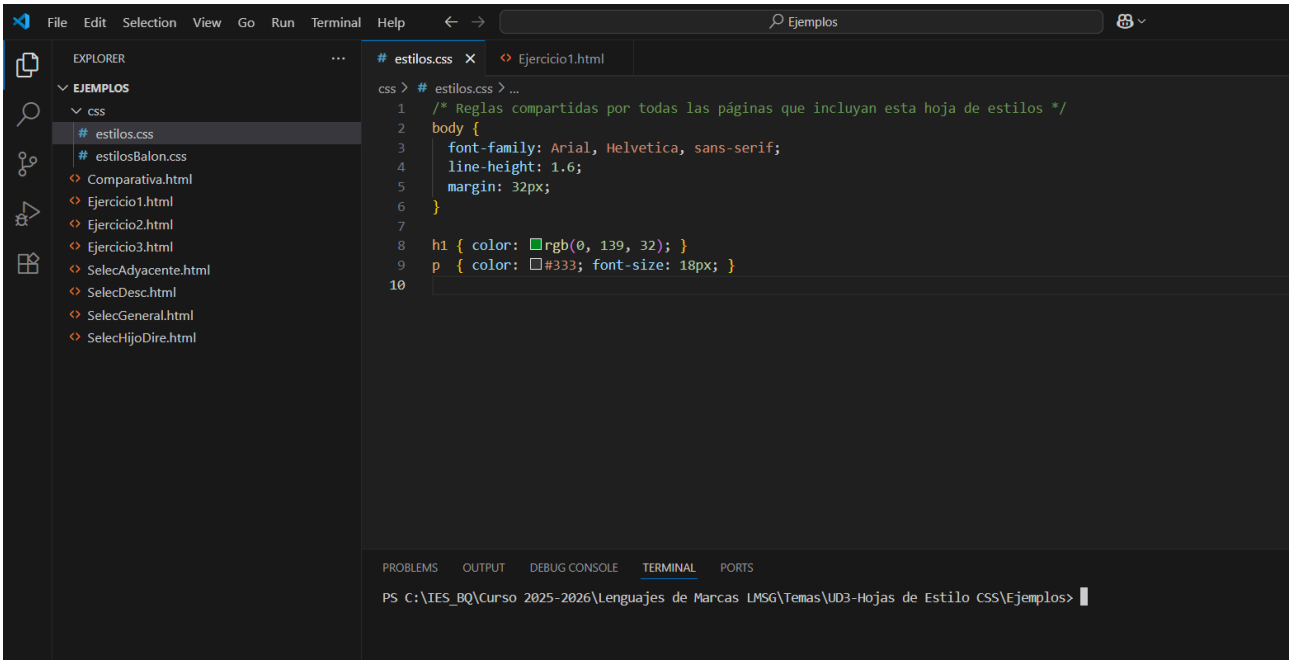


6- Visual Studio Code (VS Code) — IDE ligero para HTML/CSS

Es un IDE ligero, muy utilizado por programadores de HTML/CSS: Se trata de un editor extensible (Windows, macOS, Linux) que, con extensiones y el **Language Server Protocol (LSP)**, se comporta como un **IDE** muy cómodo para **HTML/CSS** (y JS).

Puedes pensar en VS Code como un taller: el propio editor es la mesa, y las extensiones son las herramientas que vas sacando según lo que vayas a construir. Para HTML y CSS es comodísimo: abre rápido, ayuda a escribir, y puedes ver la web en vivo sin salir del editor.

Lo primero no es crear un archivo, sino abrir una carpeta (*File* → *Open Folder*). Esa carpeta será "tu proyecto". A la izquierda verás el Explorador con los archivos; en el centro, el editor; abajo, la Terminal si la necesitas; y cuando haya algo que corregir, aparecerá el panel de Problemas.



Un truco para moverte sin menú:

Ctrl+P abre cualquier archivo escribiendo su nombre.

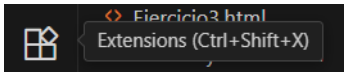
Ctrl+Shift+P te deja buscar cualquier acción: "Formatear", "Cambiar color del tema", "Open with Live Preview"… lo que sea.

Ejecutar tus páginas web es muy sencillo e inmediato. Cada vez que realices cambios en tu código fuente, tienes varias opciones para testearlo:

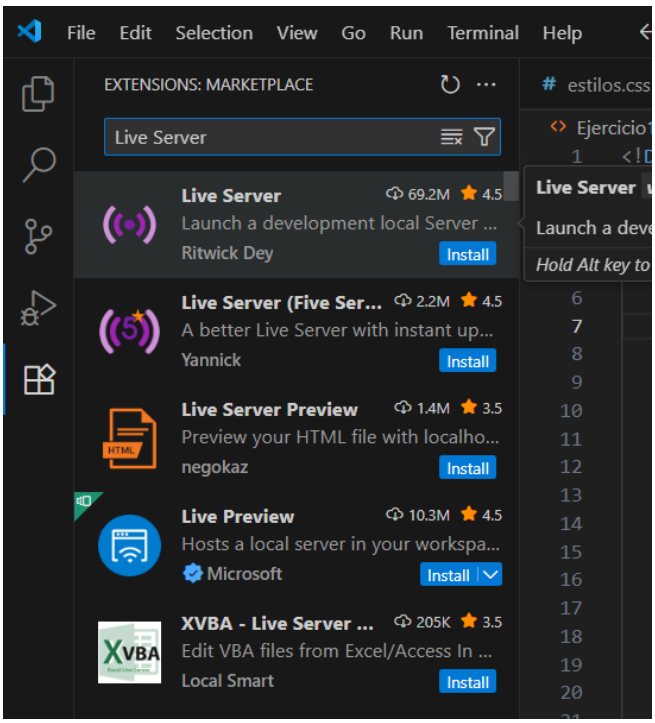
- Puedes arrastrar tu página web y soltarla en tu navegador.
- Puedes escribir la ruta donde se encuentra tu página web, en la barra de direcciones (urls) del navegador.
- Puedes instalar el **Live Server**, para *lanzar* tu página web al navegador directamente desde el VS Code.

  Live Server es una extensión para Visual Studio Code que nos sirve para lanzar un servidor local de desarrollo con recarga automática ("live reload") para páginas web estáticas (HTML, CSS, JS). Cada vez que haces un cambio en tus archivos (HTML/CSS/JS) y los guardas, Live Server vuelve a cargar automáticamente la página en el navegador para que veas los cambios de inmediato.

  Nos ayuda a un flujo de trabajo más rápido y eficiente, ya que ves los resultados inmediatamente.

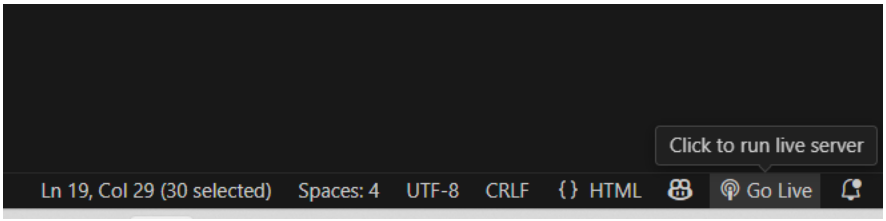
  Para instalarlo, vamos al apartado "*Extensions*" 

y buscamos *Live Server*:

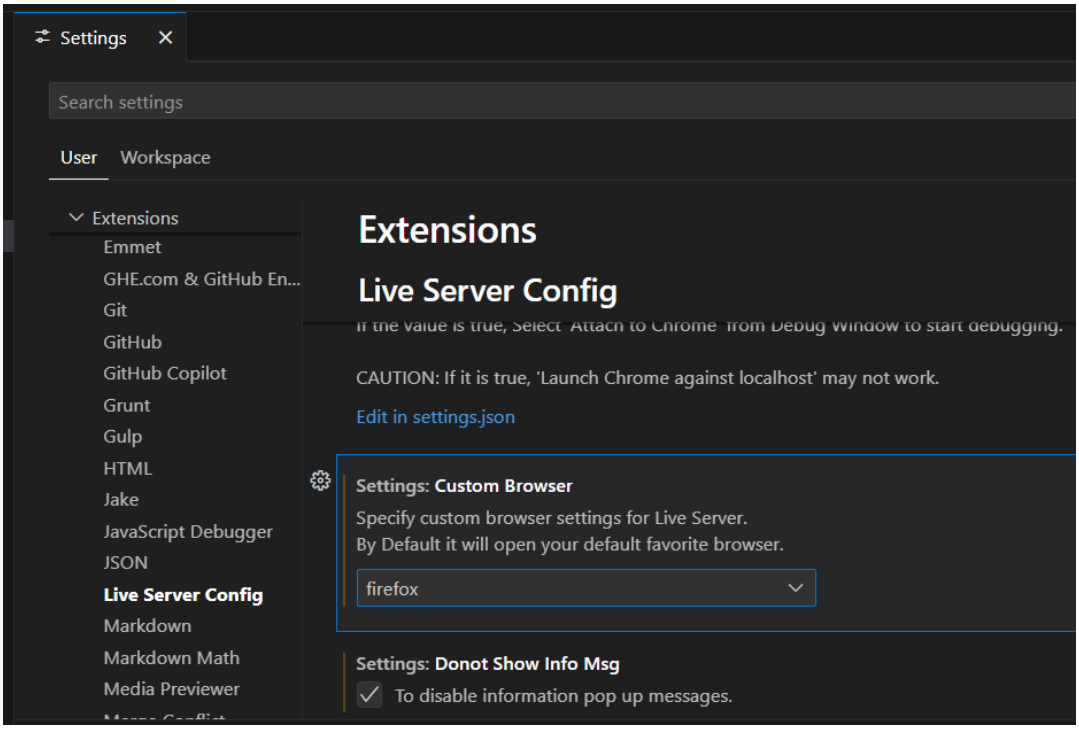


Escogemos la primera opción (la desarrollada por *Ritwick Dey*) y la instalamos.

Una vez instalado, ya nos aparece la opción de lanzar nuestra página con Live Server:



Esto nos abrirá nuestra página web en el explorador que tengamos configurado por defecto. Es posible especificarle a Live Server el explorador que lanzará nuestras páginas, desde *Settings* → *Extensions* → *Live Server Config* → *Custom Browser* (en mi caso, he especificado el *Firefox*):



Investiga por tu cuenta la herramienta **Hemmet**. En tus manos está la decisión sobre si la quieres incorporar a tus recursos a la hora de programar.

6.1. Indentación, formato y comentarios (HTML/CSS en VS Code)

Cuando una página crece, **leer el código** se parece a leer un texto sin párrafos: cansado y confuso. Dos hábitos nos salvan: **indentar y comentar**. Con ellos, tú y cualquiera de tu equipo (u otro equipo) entendéis el archivo de forma más clara.

Indentación y formato: que el editor haga el trabajo duro.

La sangría (espacios al inicio de cada línea) es el **mapa visual** de la estructura: ves qué etiqueta contiene a cuál, dónde termina un `div`, qué reglas dependen de una media query, etc.

Un HTML/CSS bien indentado:

- se **lee más rápido**,
- reduce **errores tontos** (etiquetas sin cerrar, llaves perdidas),
- y evita **discusiones de estilo** cuando trabajamos en grupo.

Para formatear de manera automática, puedes usar la combinación de teclas ***Ctrl+Alt+F***

Comentar: explica el *por qué* y orienta al lector

Los comentarios no son para repetir lo obvio, sino para **orientar** ("esta sección es el menú"), **explicar decisiones** ("usamos Grid por accesibilidad en móviles") y dejar **recordatorios**.

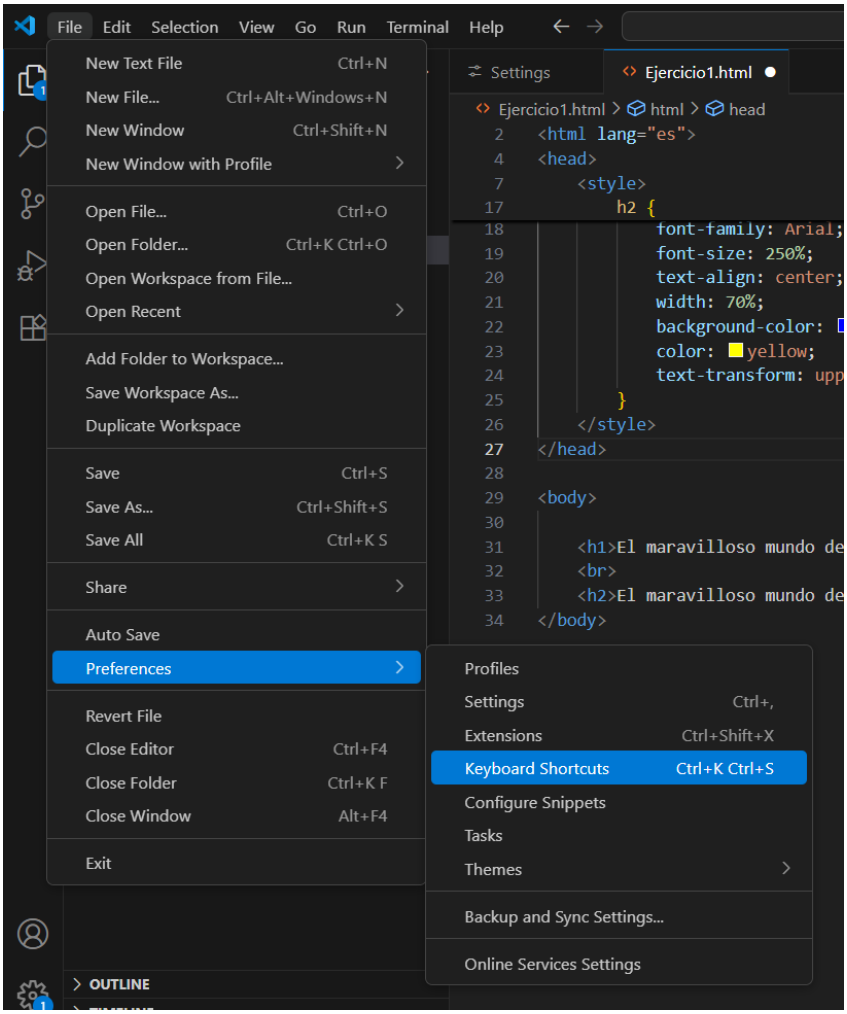
Sintaxis:

HTML: `<!-- comentario -->`

CSS: `/* comentario */`

Con la combinación de teclas ***Ctrl+K+C*** podemos comentar una línea automáticamente.

Nota: puedo configurar mis propios atajos de teclado, o cambiar los ya existentes a mi gusto, desde la opción *File → Preferences → Keyboard Shortcuts*



Buscaremos la acción que queremos atajar, y especificaremos la combinación de teclas: