UNIDAD 2:INSTALACIÓN Y USO DE ENTORNOS DE DESARROLLO

PARTE 2: JAVA EN VISUAL STUDIO CODE Y EN ECLIPSE

ÍNDICE

3. VISUAL STUDIO CODE Y JAVA

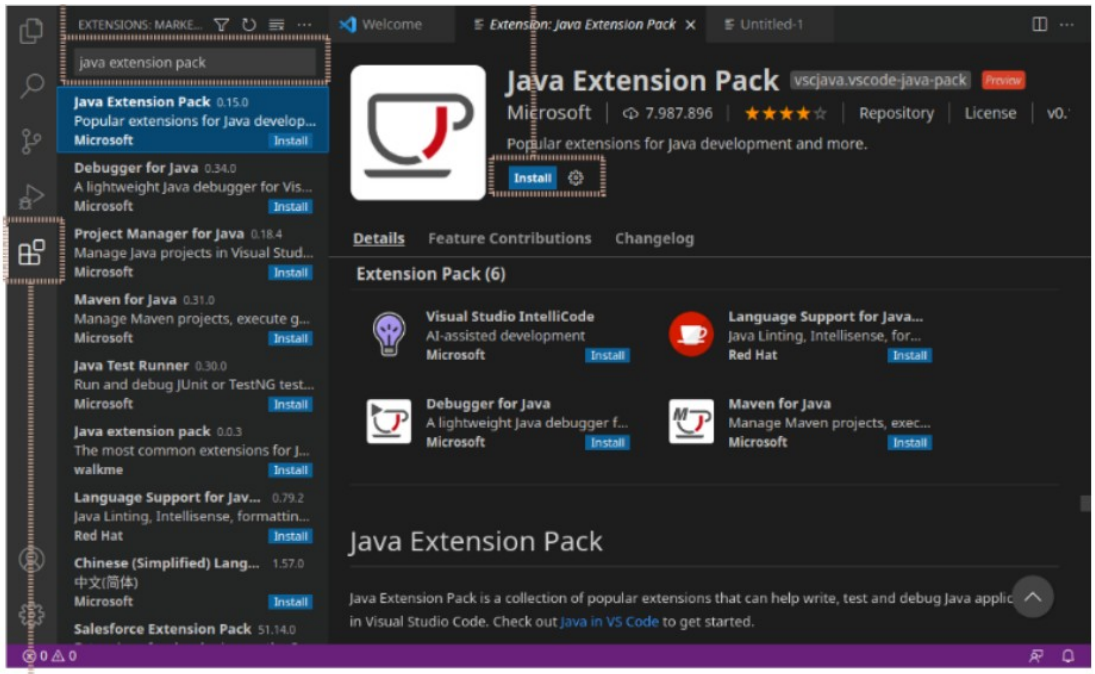
Visual Studio Code es un editor de código **muy ligero.**

Permite el **desarrollo en cientos de tecnologías mediante extensiones**.

3.1 INSTALACIÓN Y CONFIGURACIÓN DE LAS EXTENSIONES PARA JAVA

Visual Studio Code **dispone de cientos de extensiones relacionadas con Java**.

Vamos a utilizar la extensión **Java Extension Pack**.



1 Actividad de extensiones

Realmente, se trata de un conjunto de extensiones :

• **Language Support for Java**(TM) **by Red Hat**, con el soporte de Java para el IDE Eclipse: ofrece resaltado y comprobación de la sintaxis y los errores de compilación, autocompletado de código, refactorización, soporte a JavaDoc o pequeños fragmentos de código reutilizable de uso común (snippets).

• **Project Manager** f**or Java:** proporciona herramientas de exploración, creación y construcción de proyectos Java.

• **Debugger for Java,** con un depurador ligero que permite lanzar las aplicaciones, añadir puntos de ruptura, pausar y continuar la ejecución, seguimiento de variables, etcétera.

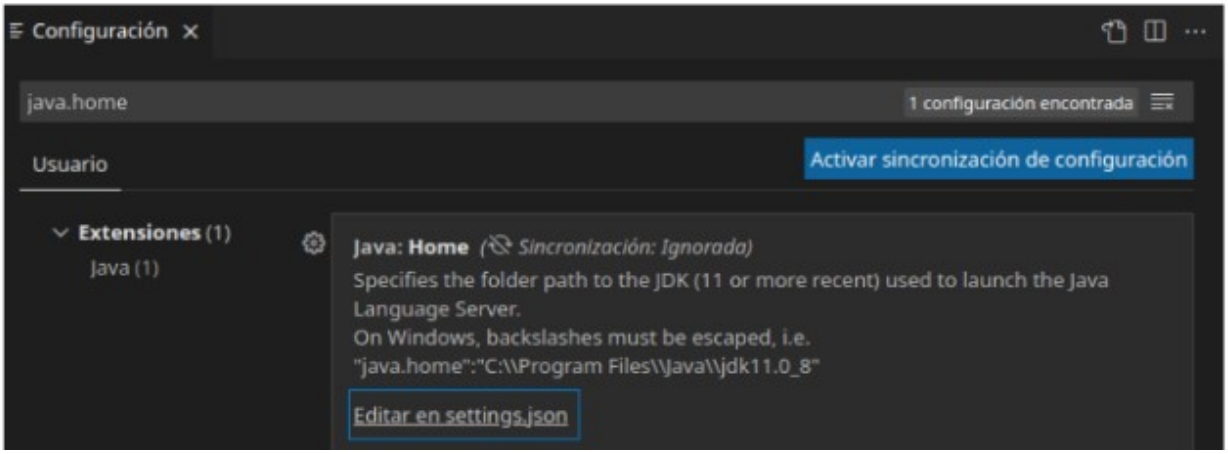
• J**ava Test Runner**, para ejecutar diferentes pruebas de funcionamiento, así como tests unitarios de JUnit, depuración o logs.

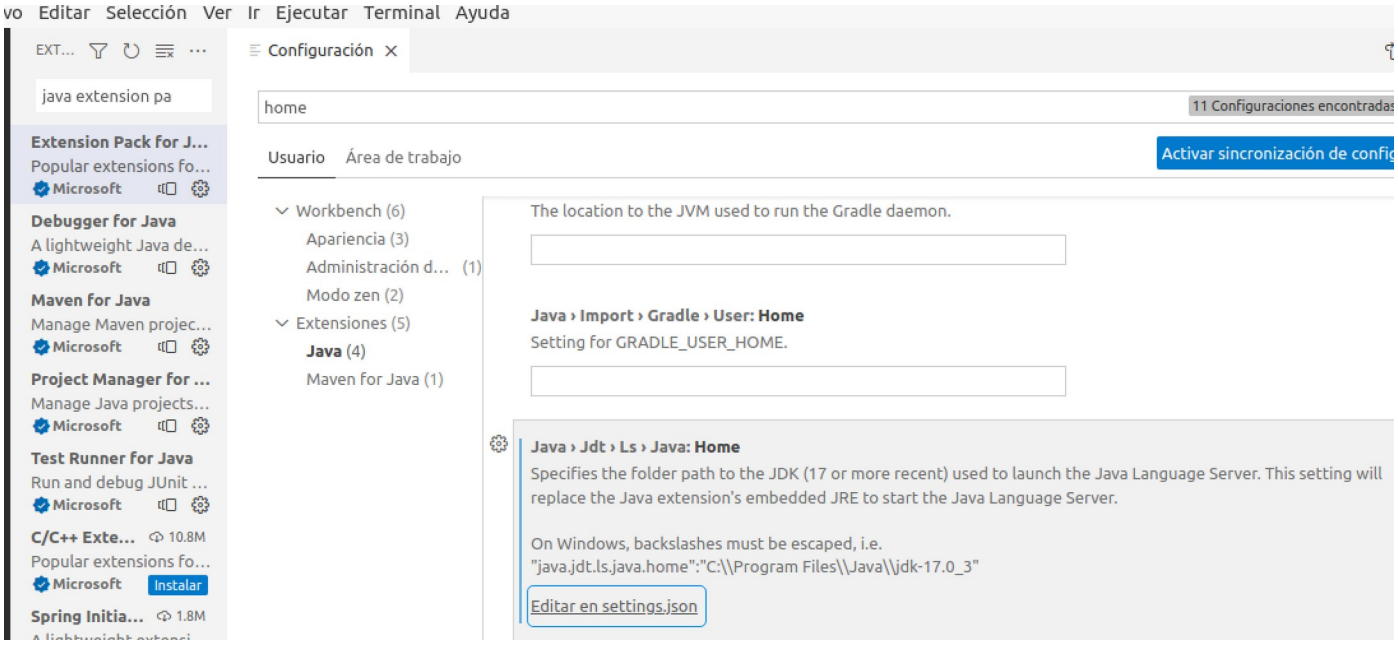
• **Maven for Java**, para la gestión de proyectos Maven. Facilita un explorador de proyectos y proporciona herramientas para su generación, validación, compilación, realización de tests o despliegue.

• **Visual Studio IntelliCode,** ofrece algunas características de desarrollo basadas en la inteligencia artificial, de modo que es capaz de inferir qué vamos a escribir según el contexto del código y el aprendizaje automático a partir del código ya generado.

CONFIGURACIÓN

- Configurar la variable de entorno **JAVA_HOME** requerida por la extensión de soporte de Java.
- Accedemos a la configuración de usuario
- En **Extensiones > Jav**a: Buscar el parámetro **Java:Home**.

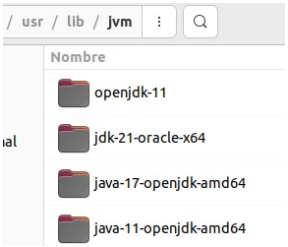




Editar en settings.json para añadir la **ubicación del JDK** (variable de entorno JAVA_HOME, previamente configurada desde el teminal con el comando export).

En Linux:

- Desde el explorador de archivos



- Desde el terminal: mostrando el valor de la variable $JAVA_HOME

```
usuario@usuario-VirtualBox:~$ echo $JAVA_HOME
/usr/lib/jvm/java-11-openjdk-amd64
```

En Windows, deberéis indicar la ruta donde se instaló el JDK: **C:\java\jdk- 16.0.1**

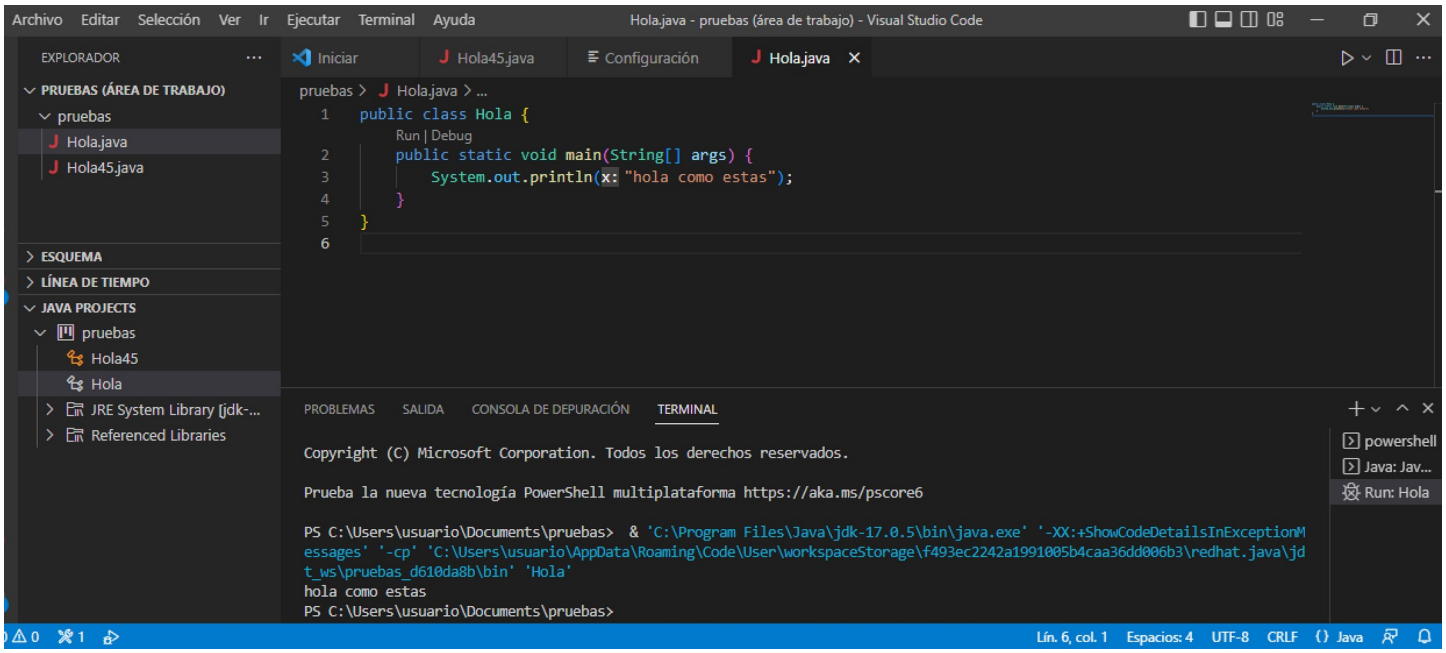
Configuración | {} settings.json

```
{
    "workbench.editor.enablePreview": false,
    "editor.minimap.enabled": false,
    "diffEditor.wordWrap": "on",
    "debug.console.wordWrap": false,
    "workbench.colorTheme": "Visual Studio Light",
    "editor.wordWrap": "on",
    "liveServer.settings.donotShowInfoMsg": true,
    "breadcrumbs.enabled": false,
    "java.jdt.ls.java.home": "/usr/lib/jvm/java-11-openjdk-amd64"
}
```

- Clic en **Guardar**

COMPROBAR QUE EL PROYECTO ESTÁ FUNCIONANDO

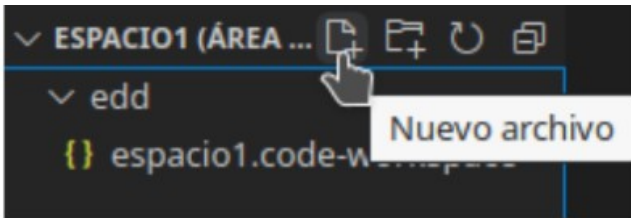
- Crear un fichero llamado **Hola.java** y comprobar que se ha creado automáticamente la clase.
- Escribir main para añadir el método **main**.
- Gracias a la confiuración se autocompleta el código y aparece Run | Debug.
- Añadir la línea **System.out.println ("Hola como estás");** escribiendo sólo las iniciales de las instrucciones y el mensaje que se desea que aparezca en el terminal.
- Pulsar **Run** para ver los resultados



NOTA: SI NO FUNCIONA: HAY QUE INSTALAR JDK A UNA VERSIÓN MÁS RECIENTE

3.2 EJECUCIÓN Y DEPURACIÓN

- Desde la carpeta de nuestro workspace , seleccionamos el **Nuevo Archivo**
- Nombramos el archivo a crear: **Hola.java**



- Al detectar que se trata de un fichero java, la extensión de Java ya nos crea el esqueleto de lo que sería la clase.
- Dentro escribimos **main**, y nos ofrecerá varias posibilidades para completar el código.
  - Elegimos el **primer snippet o fragmento de código** útil, que nos añade las siguientes líneas:

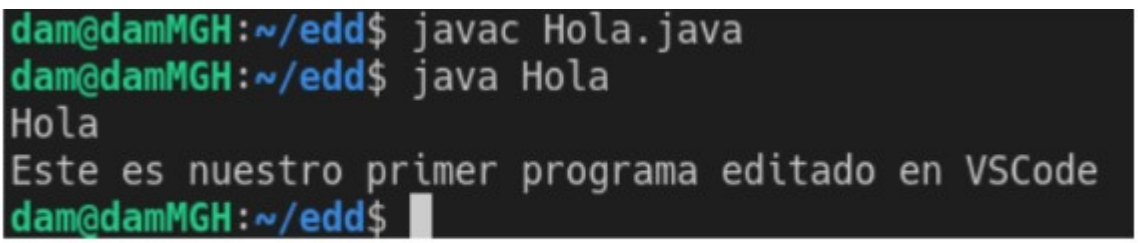
- Completamos el código con el siguiente contenido:

- Entre la línea 1 y la 2 aparecen dos opciones **Run | Debug**, que nos van a permitir ejecutar y depurar el programa.

EJECUTAR

- **Clic en Run** (o Ctrl+F5), se nos abrirá una terminal y la extensión de Java hará la compilación , mostrando el resultado de la ejecución:

- **Desde una terminal,** también podemos **compilar** y **ejecutar** el programa:



En este caso se ha creado el fichero ejecutable **Hola.class**

DEPURAR

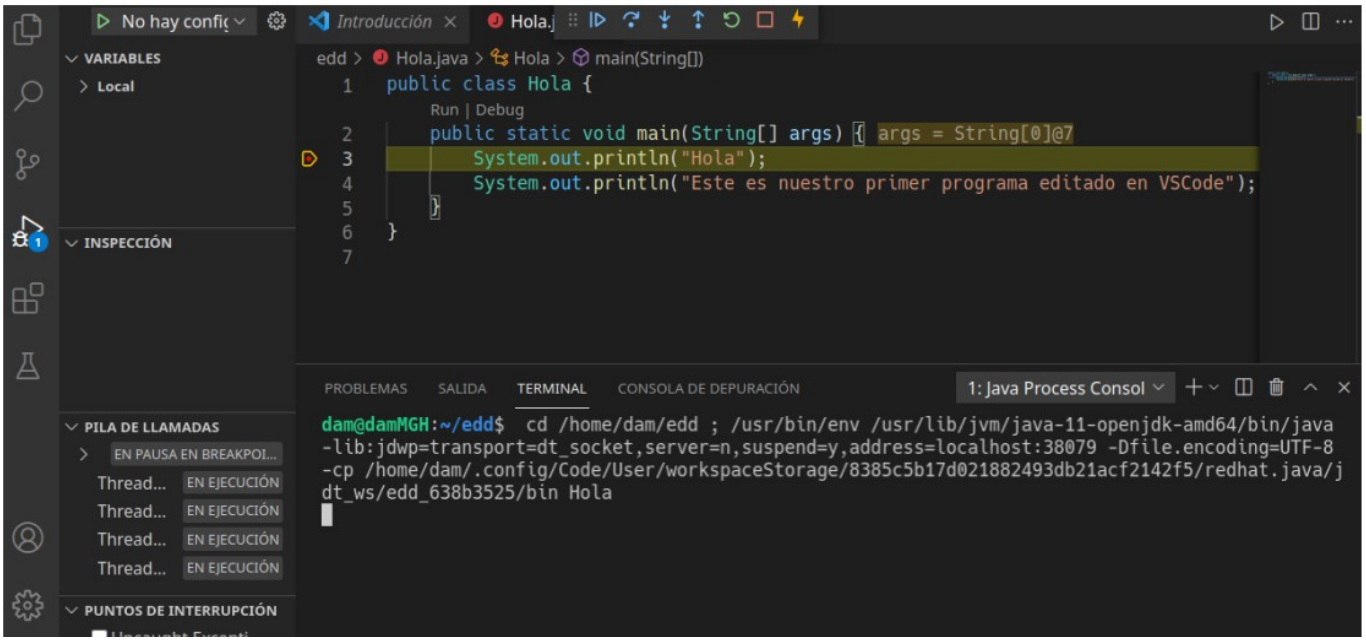
Es Ejecutar paso a paso el programa, estableciendo un punto de ruptura o breakpoint, entonces la ejecución se para cuando llegue a él.

1. Establecer puntos de ruptura

- Cuando **pasamos el ratón por el lado izquierdo de los números de línea**, aparecen unos puntos de color rojo oscuro.
- Si hacemos **clic** en uno de ellos, se **establecerá un breakpoint** en esa línea.
- No todas las líneas de código son susceptibles de ser puntos de ruptura (por ejemplo, las líneas donde abrimos o cerramos llaves)
- Añadimos un **breakpoint en la línea del primer println**

2. Empezar la depuración, pulsando Debug

  - veremos que la ejecución se para antes de mostrar esta línea, y que la barra de estado ha pasado a ser de color anaranjado, indicando que estamos en modo depuración.
  - La interfaz muestra en la parte derecha algunos detalles de la ejecución, como el valor de las variables que se encuentren activas o la pila de llamadas:



Si hay más puntos de ruptura debemos utilizar otras opciones disponibles en la barra de debug

Para **continuar con la ejecución hasta el final**: clic en el **primer icono** (o F5) 

Para **ejecutar paso a paso**, usaremos el **segundo icono** (o F10). 

3. Parar la ejecución paso a paso: 

4. Eliminar un breakpoint: volver a hacer **clic sobre el círculo rojo**.

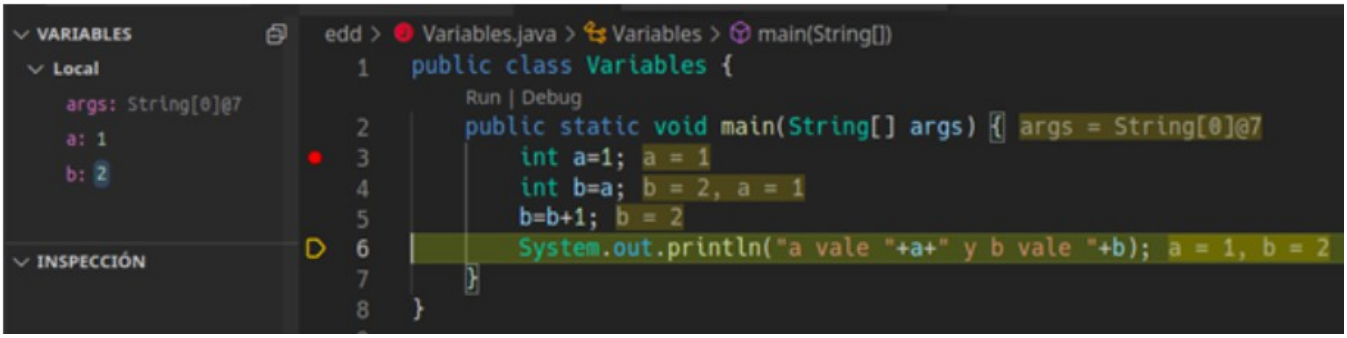
4. CASO PRÁCTICO:Depurando con variables

Crea el siguiente programa llamado **Variables.java**

PruebasCode > Variables.java > ...

```
1   public class Variables {
        Run | Debug
2       public static void main(String[] args) {
3           int a=1;
4           int b=a;
5           b=b+1;
6           System.out.println("a vale: "+a+" y b vale: "+b);
7       }
8   }
9
```

- **Establecer puntos de ruptura**
- Utilizar ***Debug*** para depurar el programa.
- **Ver cómo va cambiando el valor de las variables usando la barra debug**:



- Parar la depuración
- Eliminar los breakpoints.

5. ENTORNOS DE DESARROLLO INTEGRADOS(IDE)

Son aplicaciones que integran herramientas para el desarrollo de software, ofreciendo un completo soporte para las tareas más comunes.

Como norma general, **se compone** de un editor de código, alguna herramienta para la gestión y la construcción de proyectos, y un depurador de código.

De manera opcional, **incluyen otras herramientas**, como control de versiones o asistentes para el desarrollo de interfaces gráficas.

5.1 ENTORNO DE DESARROLLO INTEGRADO ECLIPSE

Eclipse es uno de los IDE más **potentes** y **utilizados** para el desarrollo en Java.

Es de código **abierto**, **multiplataforma** y **extensible**, desarrollado inicialmente por IBM como sucesor de VisualAge.

Desde 2003 se encarga de su desarrollo la **Fundación Eclipse**, quien lo distribuye bajo la licencia Eclipse Public License.

Eclipse va más allá de un IDE.

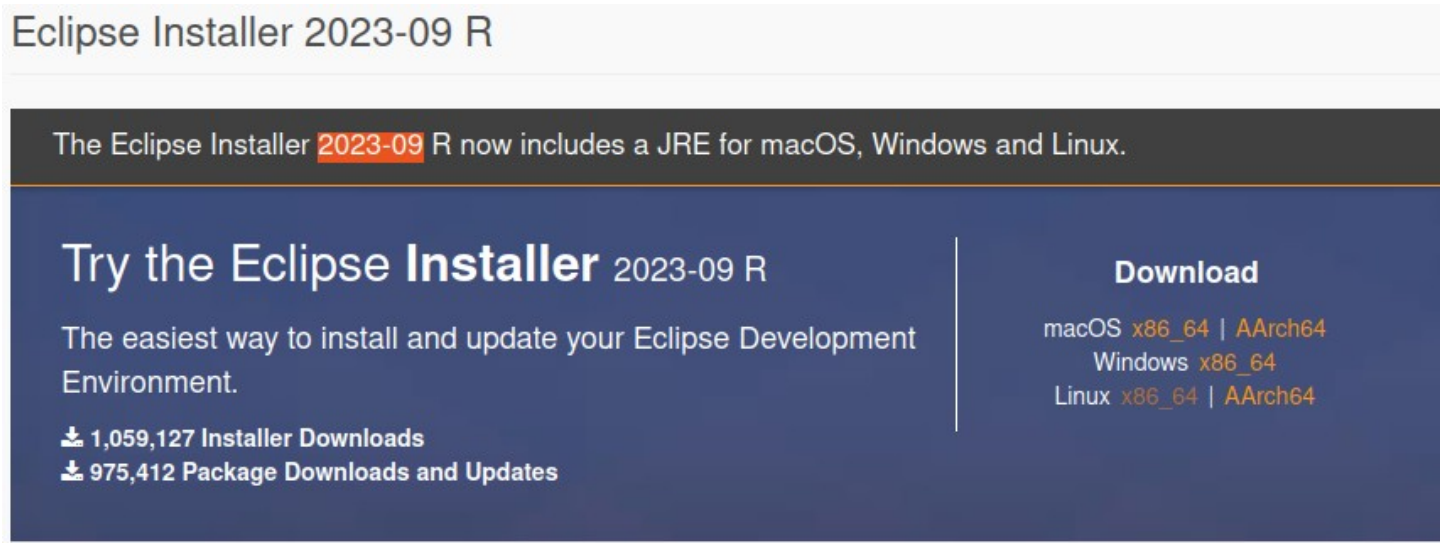
- Se trata de una **plataforma** concebida para la integración de herramientas de desarrollo de software, con un diseño que permite su extensión mediante **plugins** o extensiones, y sin tener en mente ningún lenguaje específico.
- Actualmente **tiene una gran popularidad en el desarrollo de aplicaciones Java** gracias al **plugin** *JDT* (Java Development Toolkit) incluido en la distribución estándar.
- Proporciona herramientas para **gestionar espacios de trabajo, editar, compilar, desplegar, depurar y ejecutar aplicaciones.**
- La última versión de Eclipse es **Eclipse IDE 2023-09 https://www.eclipse.org/**

5.2 INSTALACIÓN

El modo más fácil es mediante el instalador que se proporciona en su página de descargas:
http://www.eclipse.org/downloads/packages

Para más información:

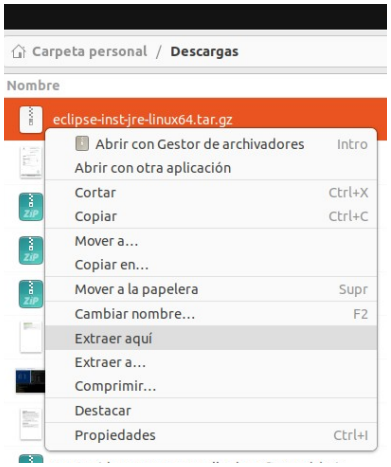
Definición, características, proceso de instalación: https://www.genbeta.com/desarrollo/eclipse-ide
Proceso de instalación: https://www.youtube.com/watch?v=eEXvOjtauxI
Instalación y primeros pasos:https://www.youtube.com/watch?v=PT8AziBfrd



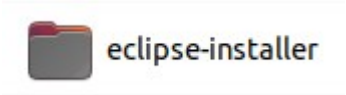
NOTA: En Ubuntu podemos instalarlo desde la **tienda**.

Seleccionamos el enlace correspondiente a nuestro sistema operativo y lo **descargamos**.

- El siguiente paso es **descomprimirlo**.
  - Mediante el **explorador de archivos**, haciendo clic en el botón derecho sobre el fichero descargado y descomprimiéndolo,
  - Mediante la **terminal**, en el caso de **Ubuntu**.
    - Nos situamos en el directorio donde tenemos descargado el fichero (generalmente **Descargas**) y lanzamos el comando **tar -xvzf eclipse-inst-jre-linux64.tar.gz** (también es posible en modo gráfico, clic derecho y Extraer)



    - Se genera la carpeta **eclipse-installer** con el instalador de Eclipse.

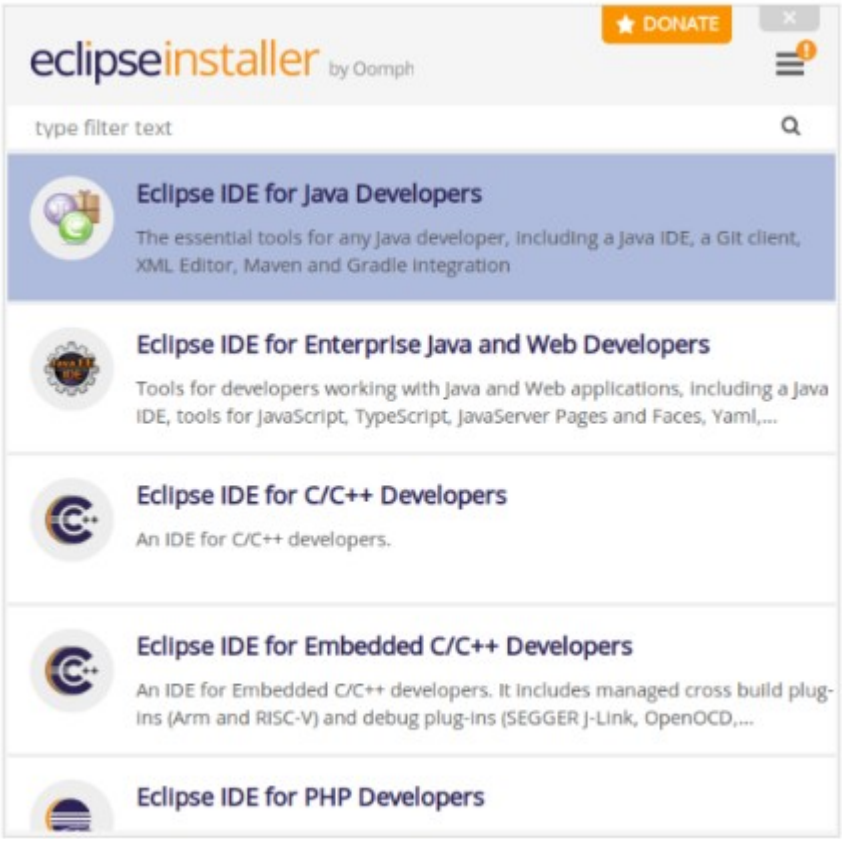


    - Nos situamos en ella con el comando **cd eclipse-installer**
    - Lanzamos el script **eclipse-inst** **./eclipse-inst**

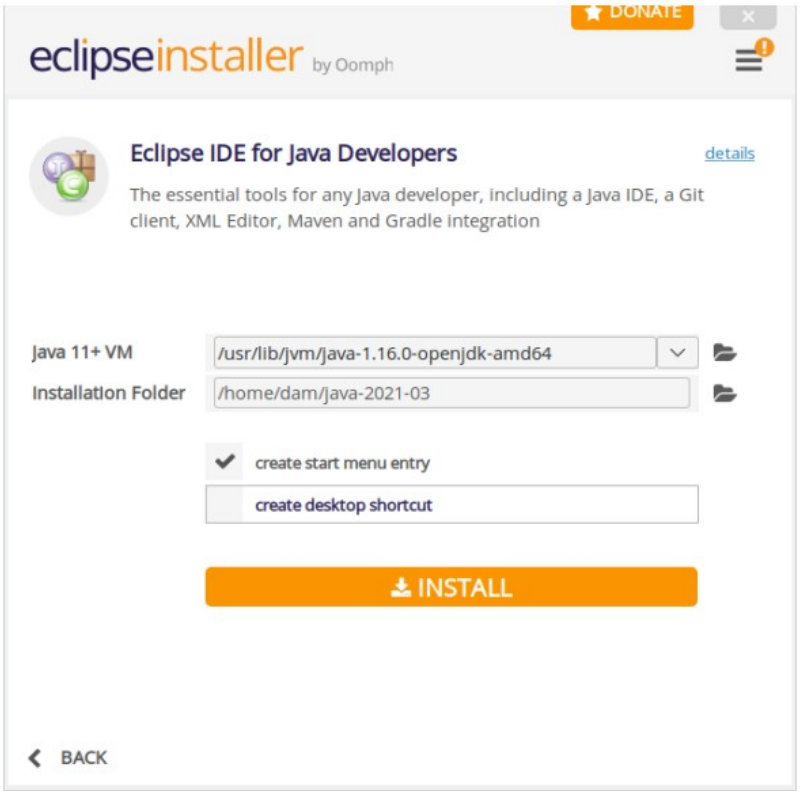
```
usuario@HP-ENVY-15-Notebook:~/Descargas/eclipse-installer$ ls
artifacts.xml  configuration  eclipse-inst  eclipse-inst.ini  features  icon.xpm  p2  plugins  readme
usuario@HP-ENVY-15-Notebook:~/Descargas/eclipse-installer$ ./eclipse-inst
```

- Se **iniciará el asistente de instalación**.

- Seleccionar el paquete que deseemos instalar: **Eclipse para desarrolladores Java**



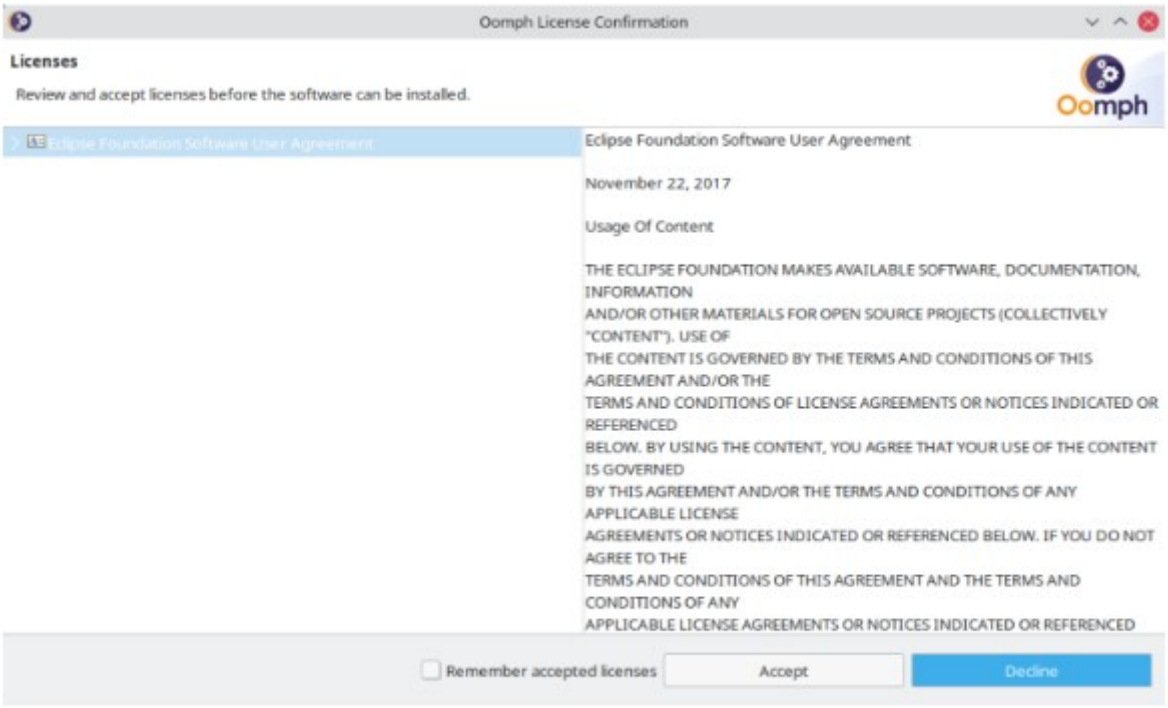
- La segunda ventana del asistente



- **Indicar dónde se encuentra la máquina virtual de Java**. Esto coincide con el JAVA_HOME que tenemos definido.

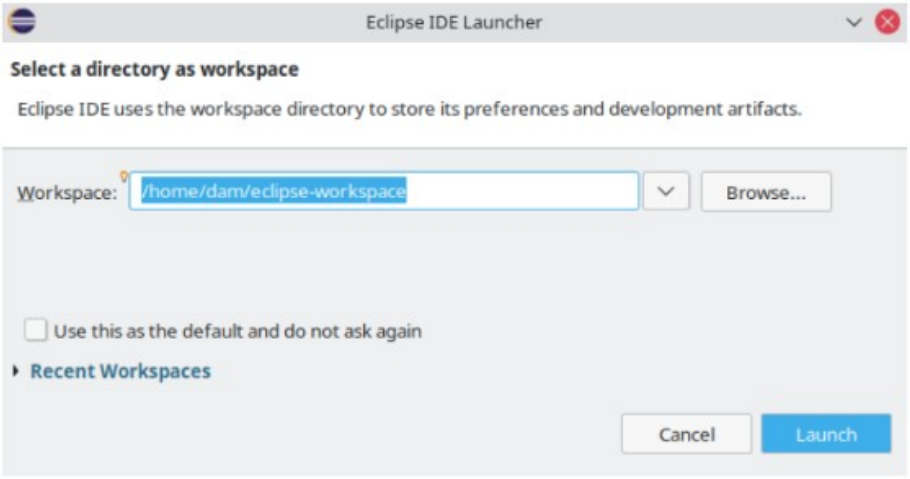
- Si no detecta este valor, podemos seleccionarlo manualmente seleccionando el icono de la carpeta ubicado a su derecha.
- Seleccionar el JRE a utilizar en la ventana Preferencias (Window >Preferences), en el apartado Java > Installed JRE.

- **Indicar la carpeta donde instalar el IDE**.
  - Dado que vamos a instalar el IDE solamente para nuestro usuario, dejamos la carpeta que se propone por defecto
- Marcamos que se **cree una entrada en el menú**
- Opcionalmente, si deseamos un acceso directo o no en el escritorio.
- Clic en **Install** para empezar la instalación.
- Aceptar las licencias.



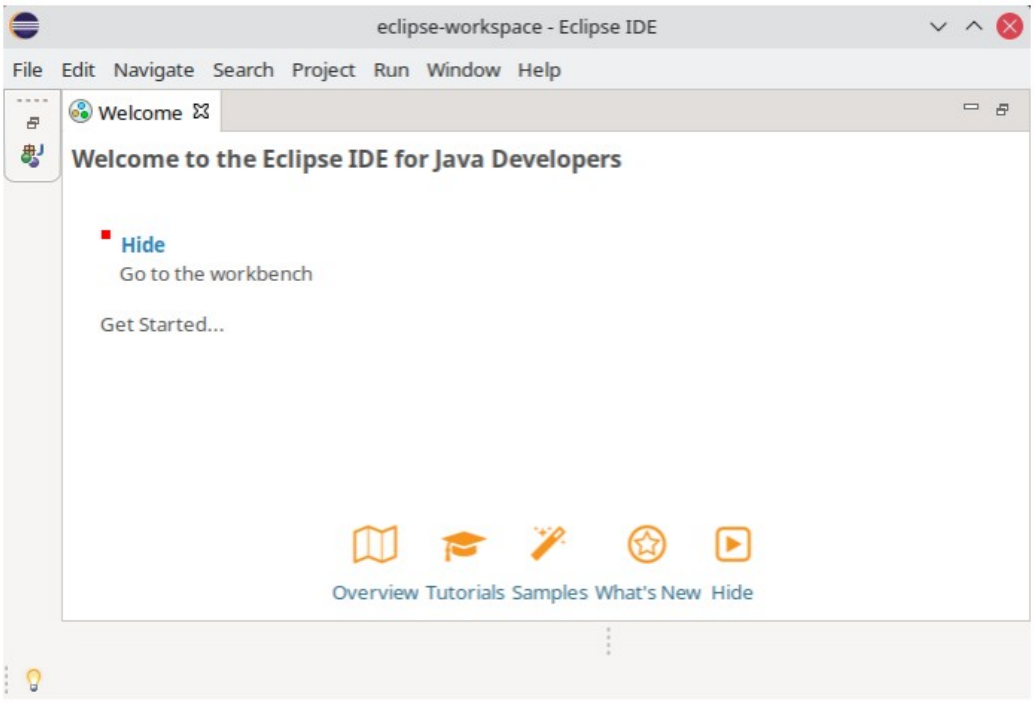
5.3 PRIMERA EJECUCIÓN

- La primera vez que ejecutemos Eclipse nos pedirá que **seleccionemos el directorio de trabajo** (workspace directory), para guardar también las preferencias y extensiones.
- **Dejamos la ruta mostrada por defecto**
- Si lo deseamos podemos marcar la opción para que siempre se use este espacio de trabajo.



- Pulsamos en **Launch**

5.4 INTERFAZ DE ECLIPSE



Es una interfaz bastante limpia, con una pantalla de bienvenida desde la que podremos acceder a documentación, tutoriales, ejemplos o novedades.

- En Eclipse el **concepto de trabajo se basa en las perspectivas**.
  - una preconfiguración de subventanas, conocidas como vistas, y
  - editores, con relación entre ellos,
  - y que permiten trabajar de forma óptima en un determinado entorno.
- A la **izquierda** de la ventana se encuentran las **diferentes perspectivas** que soporta nuestro entorno.
  - En este caso, tenemos disponible únicamente la perspectiva de Java.
- Para **abrir una perspectiva:** menú Window > Perspective > Open Perspective > Other:

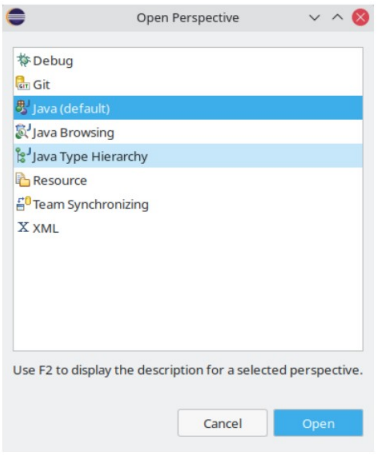
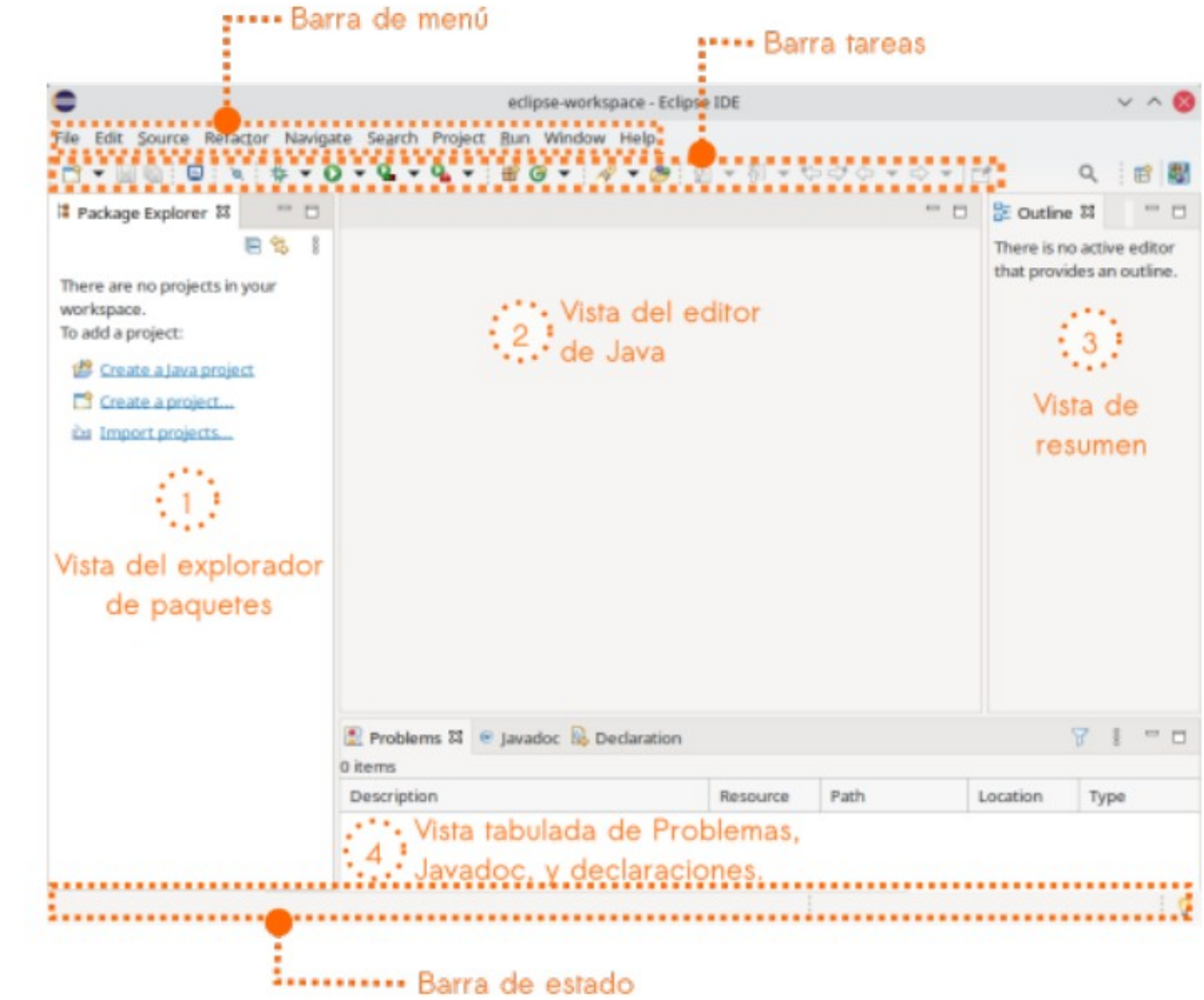


Figura 2.25. Perspectivas.

- Si activamos la **perspectiva Java**, la interfaz que se nos muestra:



La **Perspectiva de Java** se compone de las siguientes **vistas**:

1. **Vista del explorador del paquete (**Package Explorer View): muestra la estructura del proyecto que tengamos abierto, o algunas opciones para crear proyectos en caso contrario.

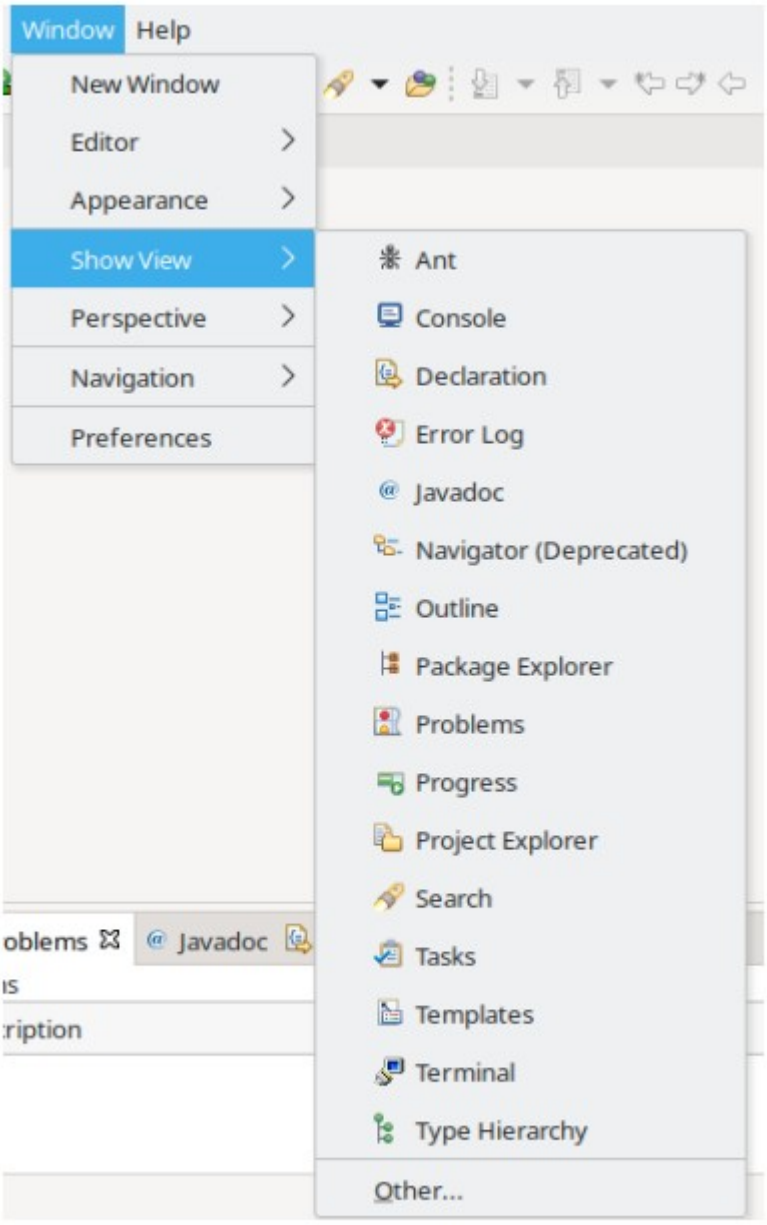
2. **Editor de Java**, donde escribiremos el código de nuestras aplicaciones. Cabe decir que soporta

autocompletado y algunos snippets de código.

3. **Vista de Resumen** (Outline), donde se mostrará un resumen con los principales métodos y atributos que tenemos definidos dentro de una clase.

4. **Vistas de Problemas, Javadoc y Declaraciones**.

Esta interfaz es configurable y podemos tanto cerrar como activar nuevas vistas mediante el menú **Window > Show View.**



5.5 CASO PRÁCTICO:Múltiples JRE para Eclipse

Disponemos de una máquina con Ubuntu con distintas versiones de Java.

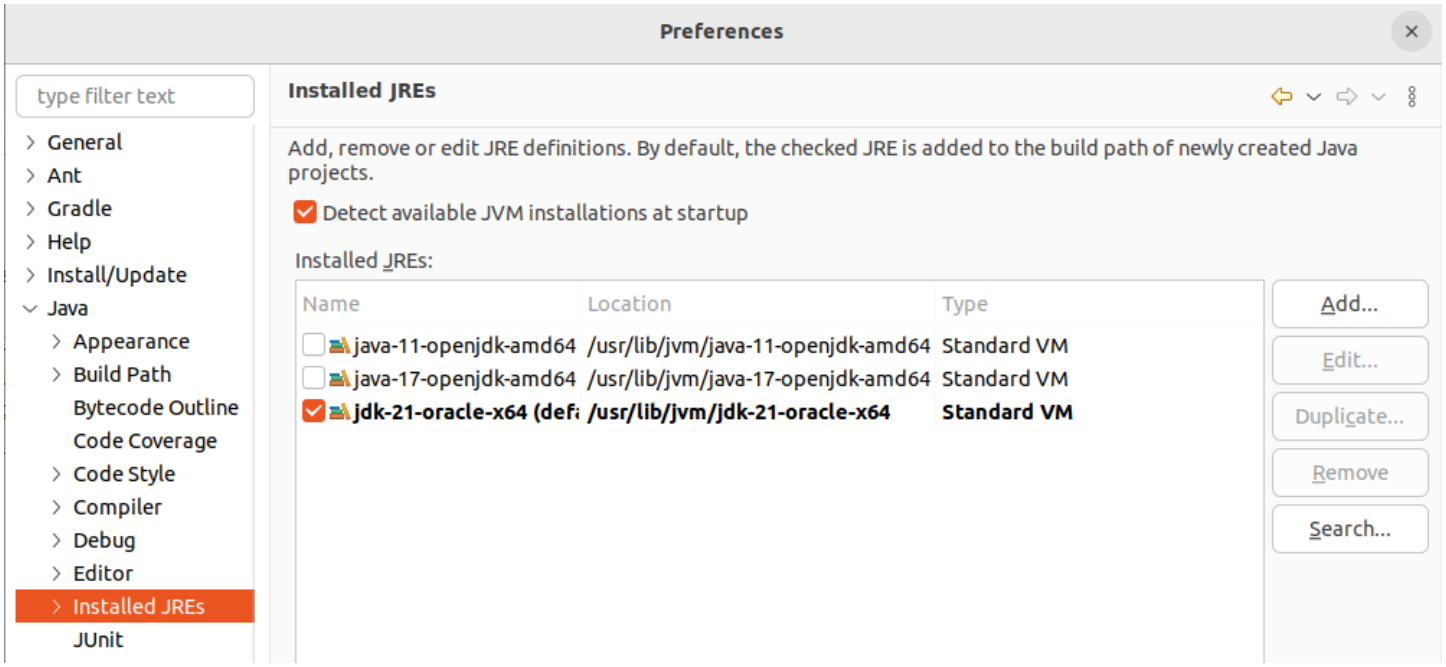
Si hacemos un ***update-alternatives***, comprobamos las que hay instaladas.

```
usuario@HP-ENVY-15-Notebook:~$ sudo update-alternatives --config java
[sudo] contraseña para usuario:
Existen 3 opciones para la alternativa java (que provee /usr/bin/java).

  Selección   Ruta                                            Prioridad  Estado
------------------------------------------------------------
* 0            /usr/lib/jvm/jdk-21-oracle-x64/bin/java        352321536 modo automático
  1            /usr/lib/jvm/java-11-openjdk-amd64/bin/java    1111      modo manual
  2            /usr/lib/jvm/java-17-openjdk-amd64/bin/java    1711      modo manual
  3            /usr/lib/jvm/jdk-21-oracle-x64/bin/java        352321536 modo manual

Pulse <Intro> para mantener el valor por omisión [*] o pulse un número de selección:
```

- Abrimos ***Window*** > ***Preferences*** y dentro de la categoría Java buscamos ***Installed JRE***



Si se desea instalar el **paquete de idiomas de español**: https://www.programaenlinea.net/cambiar-idioma-eclipse-espanol-primera-parte/

6. ECLIPSE Y JAVA

El IDE de Eclipse está pensado principalmente para el desarrollo de aplicaciones Java.

Durante la instalación, ya configuramos el entorno para que utilizase la instalación de JDK de nuestro equipo.

Vamos a ver cómo generar **proyectos Java** en Eclipse.

Aparte del código de la aplicación, en un proyecto se **organizará la documentación y otro tipo de recursos, como imágenes o configuraciones.**

Eclipse no es el único IDE especializado en Java. Existen otros entornos, como **NetBeans** o **IntelliJ**, que también juegan un importante papel en el ecosistema de IDE para Java.

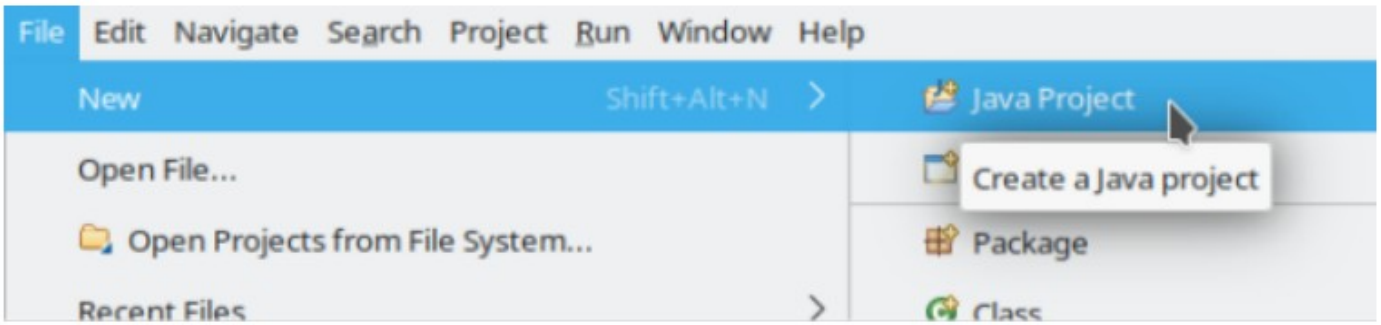
6.1 CREACIÓN DE PROYECTOS

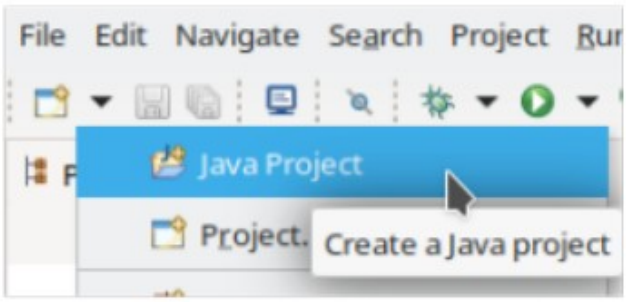
El desarrollo en Eclipse se basa en proyectos.

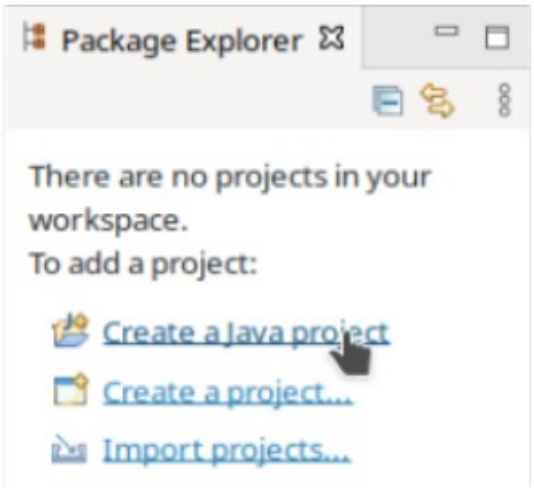
Un proyecto se compone de un conjunto de recursos relacionados, como el **código**, la **documentación**, los **directorios** o los ficheros de configuración o imágenes, entre otros.

Podemos crear un proyecto Java nuevo de diversas formas.

- el menú **File > New >Java Project**;
- con el icono New de la barra de herramientas, o
- desde el propio Explorador de paquetes:

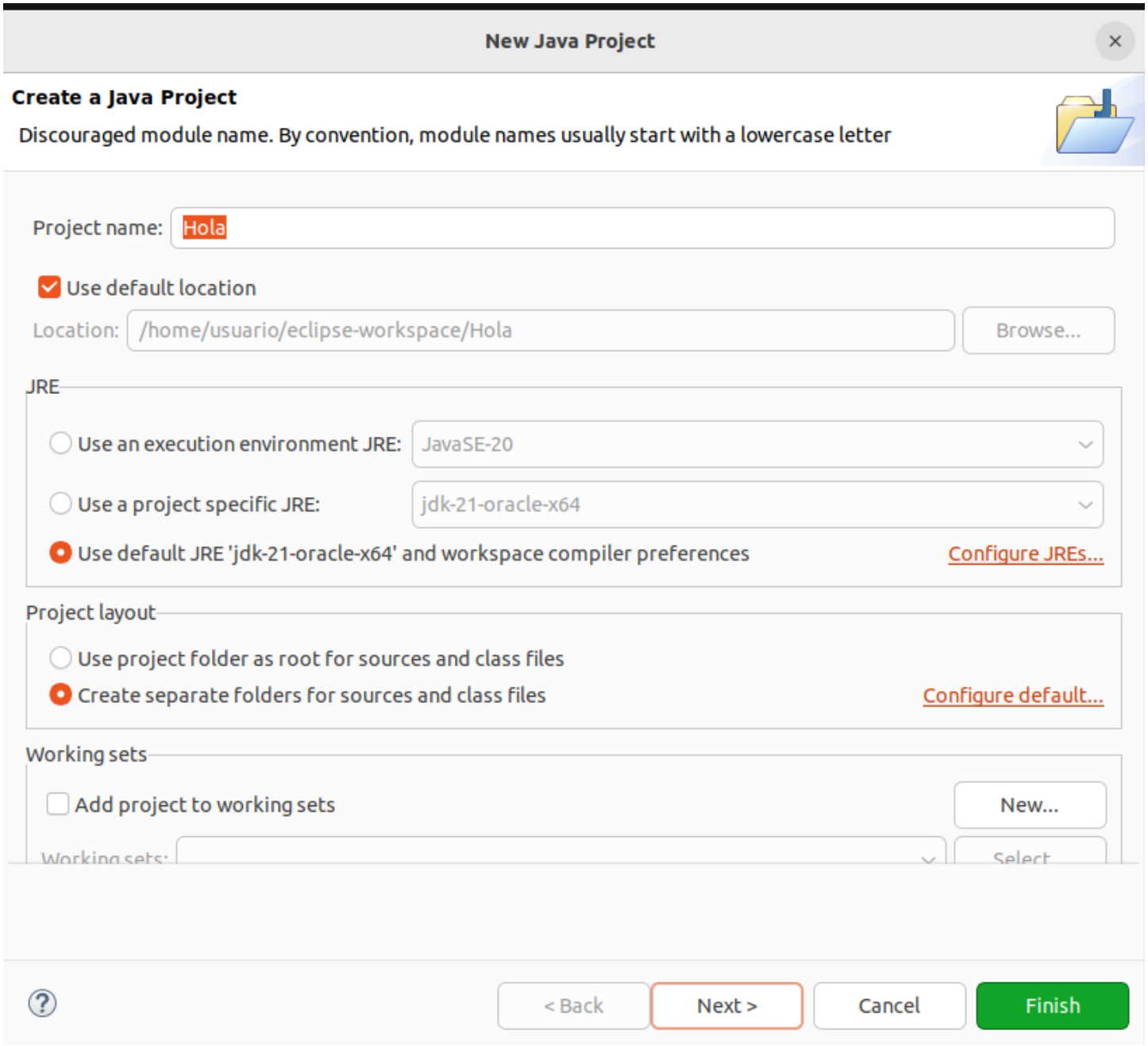




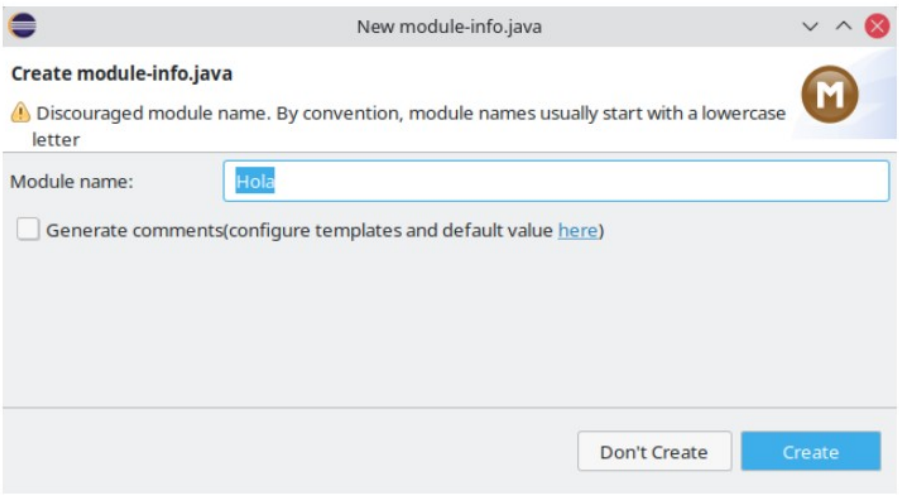


Abrirá una ventana para indicar algunos parámetros de configuración: **nombre**, la **JVM** para la que se desarrollará o el **diseño** (layout) que tendrá.

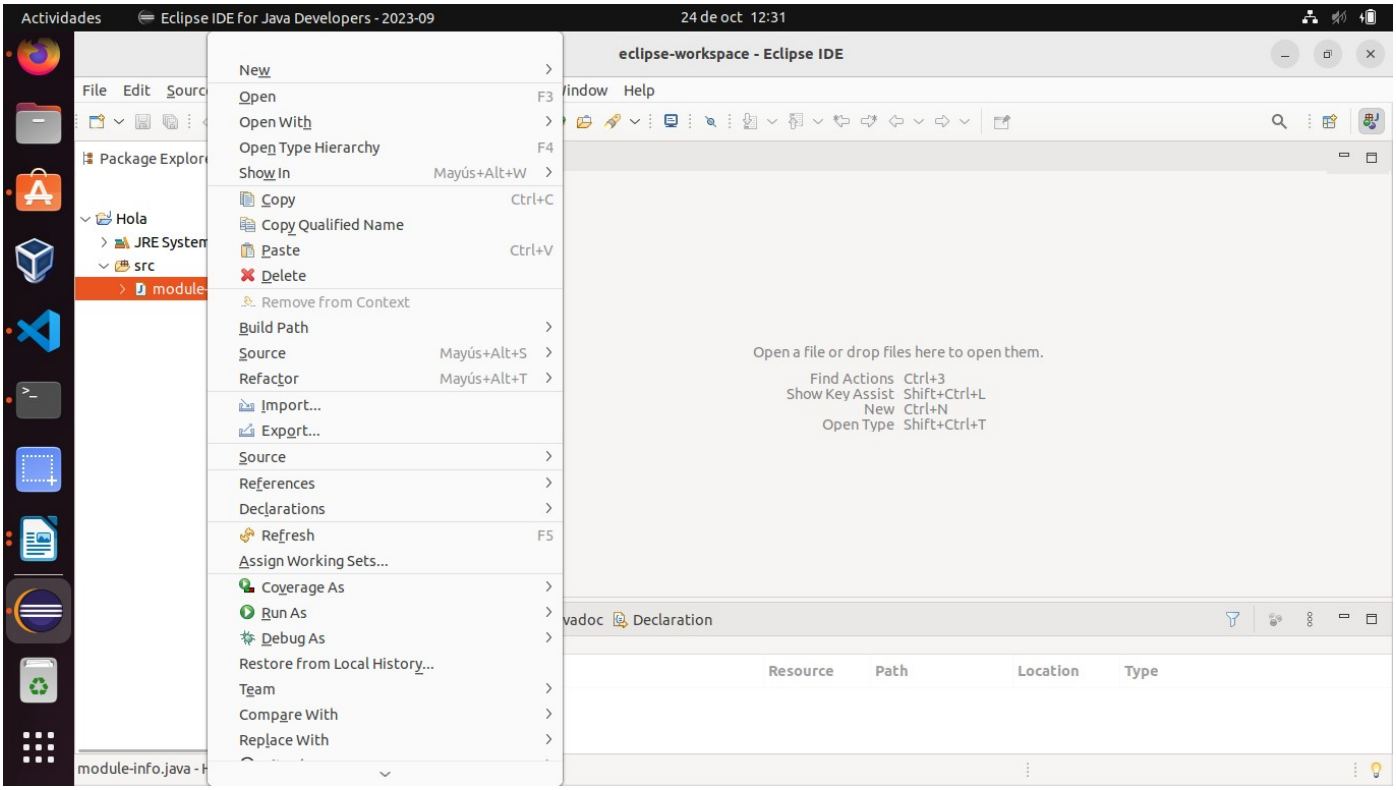
Ejemplo, crearemos el proyecto **Hola**, que usa el JRE por defecto del sistema, y en el que la estructura de carpetas separará las fuentes de los ficheros compilados en bytecode.



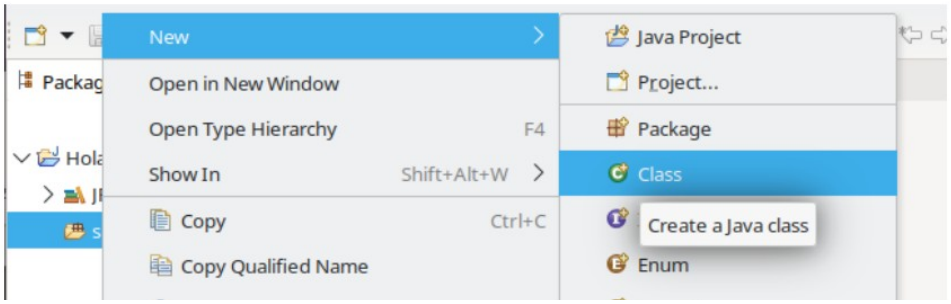
- **Finish** y pedirá si deseamos crear un proyecto modular. Si es así, creará un fichero llamado module-info.java con la configuración de este.
    - No vamos a hacer un proyecto modular: **Don't Create**



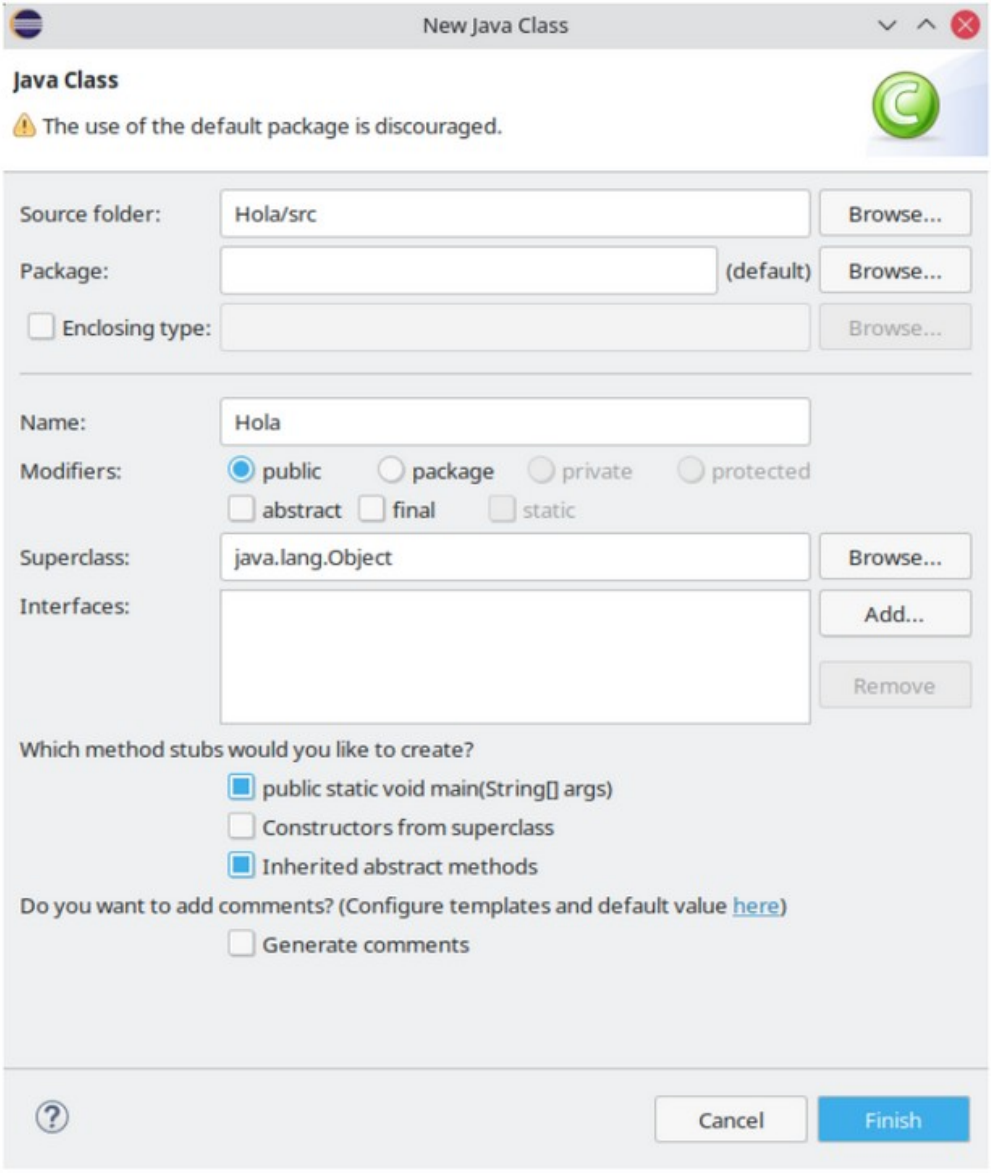
- Tendremos la estructura de nuestro proyecto creada y visible desde el Package Explorer.
- Si se ha creado el module-info.java lo podemos eliminar(clic derecho, delete)



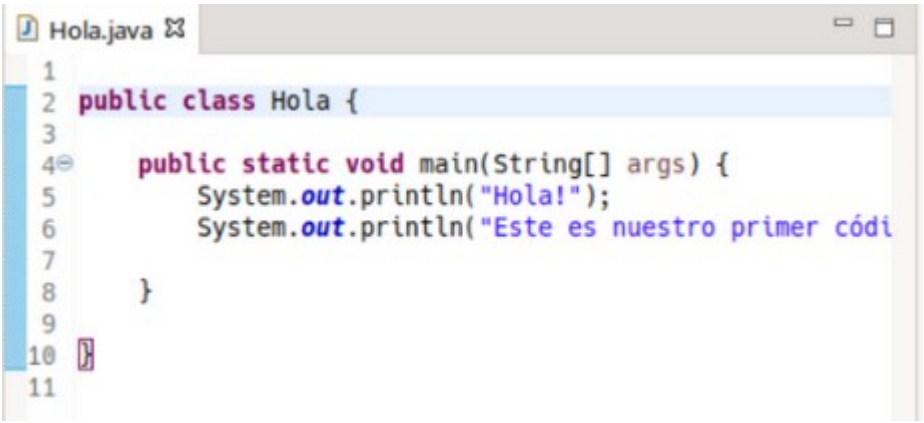
- Veremos la **carpeta del proyecto**, que contiene
  - la referencia a la biblioteca del sistema del JRE, y
  - la carpeta con las fuentes (src, de source).
- Para **añadir código**, tenemos que **crear una clase** que contenga el método **main** de la aplicación:
  - **Clic derech**o sobre la carpeta **src** y seleccionamos la opción **New > Class**:



- **Indicar la carpeta donde se guardará la clase, un nombre de paquete y el nombre de la clase, además de otros parámetros de configuración** que iremos conociendo poco a poco.
  - Lo dejaremos todo como está indicando:
    - en Name el nombre de la clase Hola,
    - que nos cree el método main:



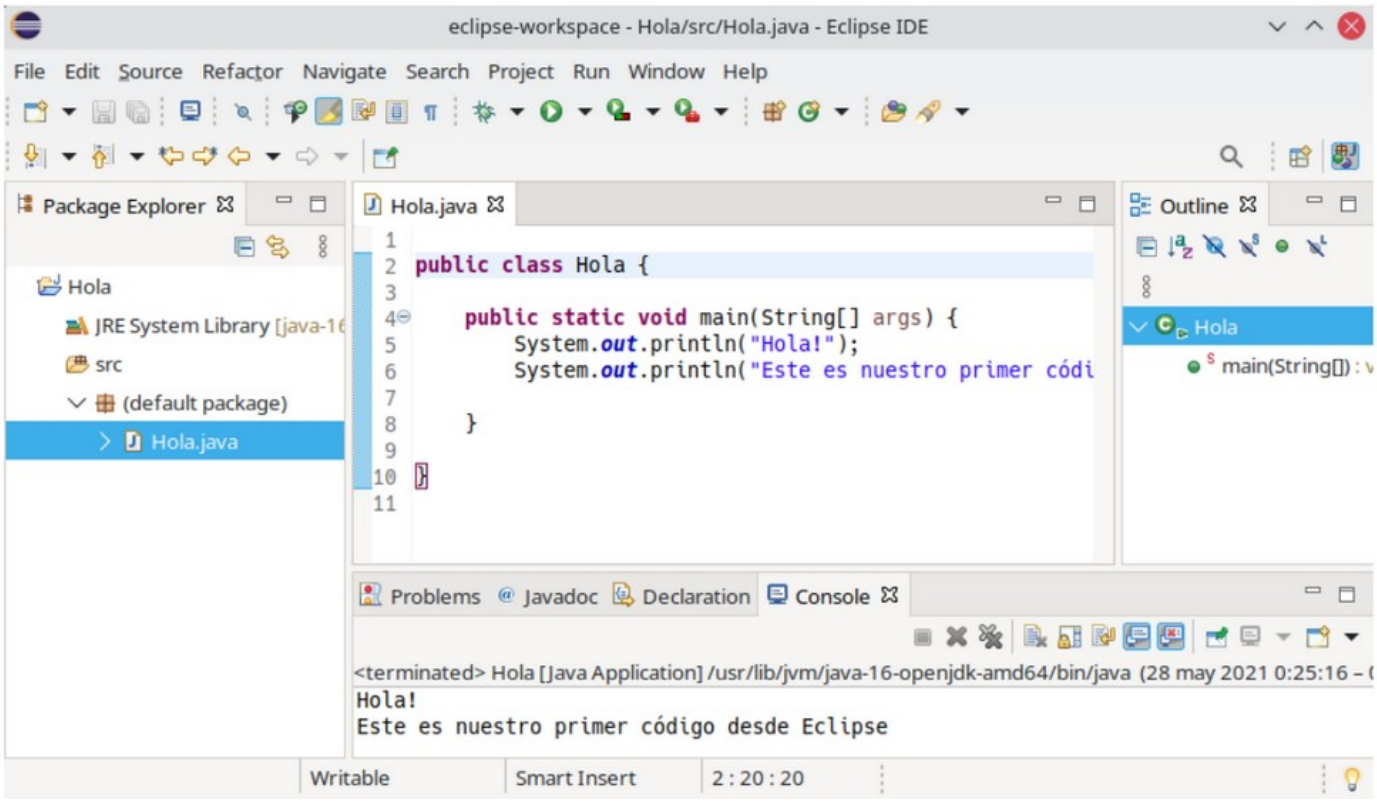
A**ñadir las líneas de código**:



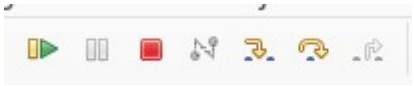
Ejecutar el código:

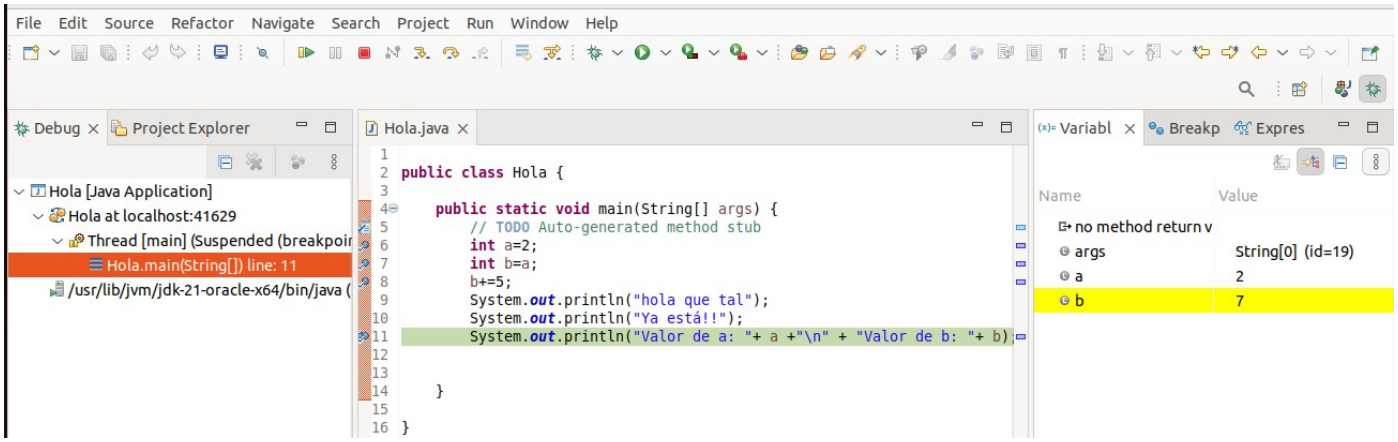
- clic menú **Run > Run**, o
- clic en el **botón de ejecución** de la barra de tareas , o
- **Control+F11.**

Podemos ver la salida en la vista de Console.



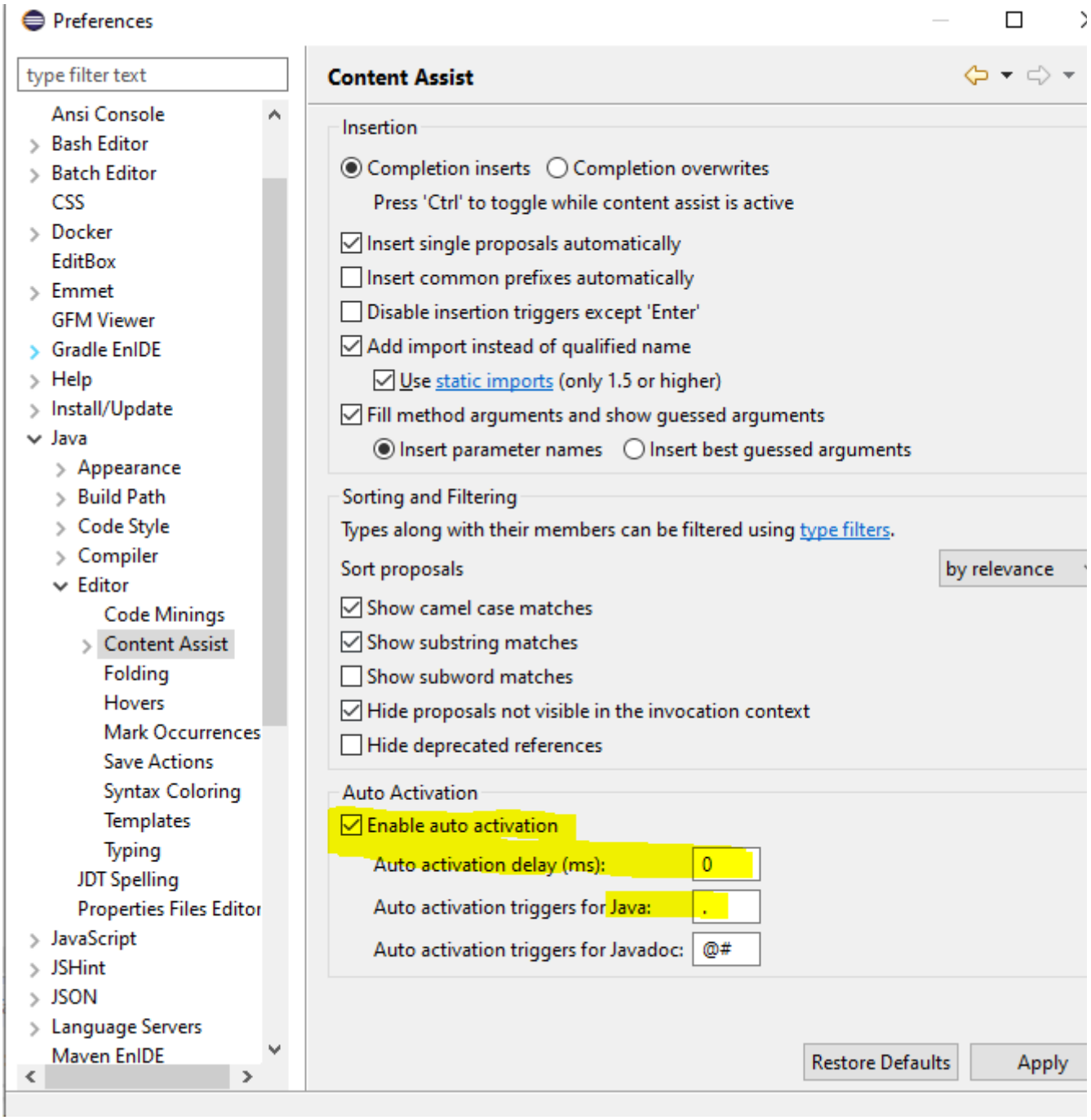
PARA DEPURAR: Opción **Debug** o F11.

- **Establecer puntos de ruptura** haciendo doble clic a la izquierda del número de línea
- Pulsar Debug y abrir la perspectiva de Debug (si no se abre automáticamente) 
- Utilizar botones de Debug 
- Comprobar la evolución de las variables
- Comprobar los breakpoints



6.2 CONFIGURACIONES

Autocompletado



https://es.stackoverflow.com/questions/360878/hay-alg%C3%BAn-intellisense-o-autocompletado-para-eclipse

6.3 REFUERZO ECLIPSE

Instalación y primeros pasos:https://www.youtube.com/watch?v=PT8AziBfrdQ

6.4 CASO PRÁCTICO: Árbol de directorios del proyecto Explorador de paquetes

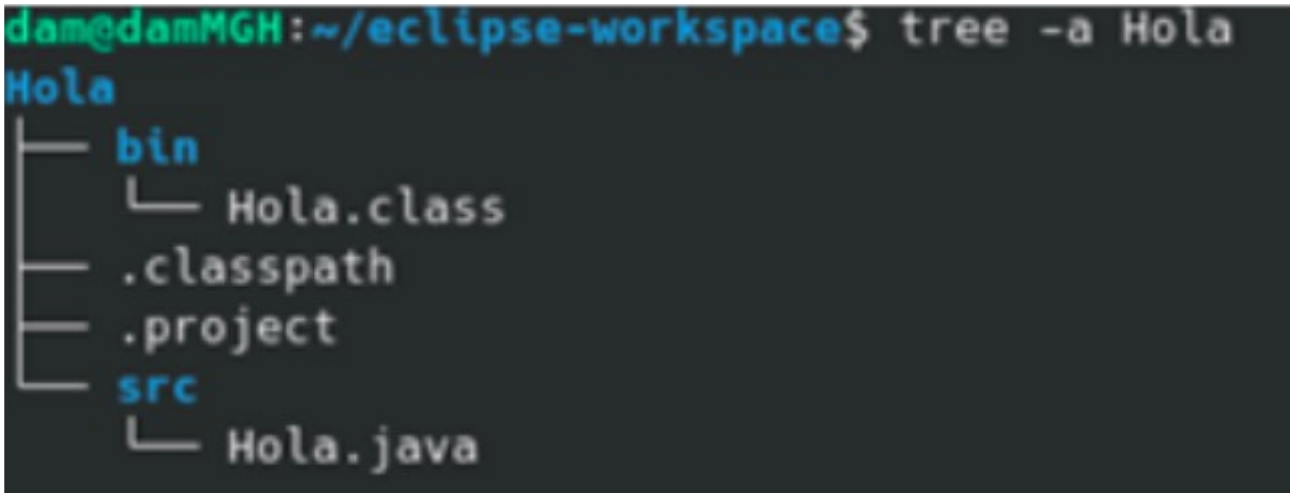
Hemos creado un proyecto *Hola* mediante Eclipse. Funciona correctamente, mostrando el mensaje que esperamos. Eclipse nos muestra en el explorador de paquetes como se ve en la imagen. Ahora nos preguntamos

¿Esto se corresponde con la estructura de carpetas generadas en la carpeta del proyecto? ¿Cómo lo averiguamos?

En el **explorador de paquetes**, Eclipse nos muestra una vista parcial y organizada por *paquetes* de nuestro proyecto.

Si accedemos a la carpeta donde tenemos guardado el proyecto y examinamos el contenido, encontraremos varias cosas interesantes.

Desde la **línea de comandos** podemos utilizar el comando ***tree -a*** (probablemente debamos instalarlo antes):



Se han creado:

dos carpetas:

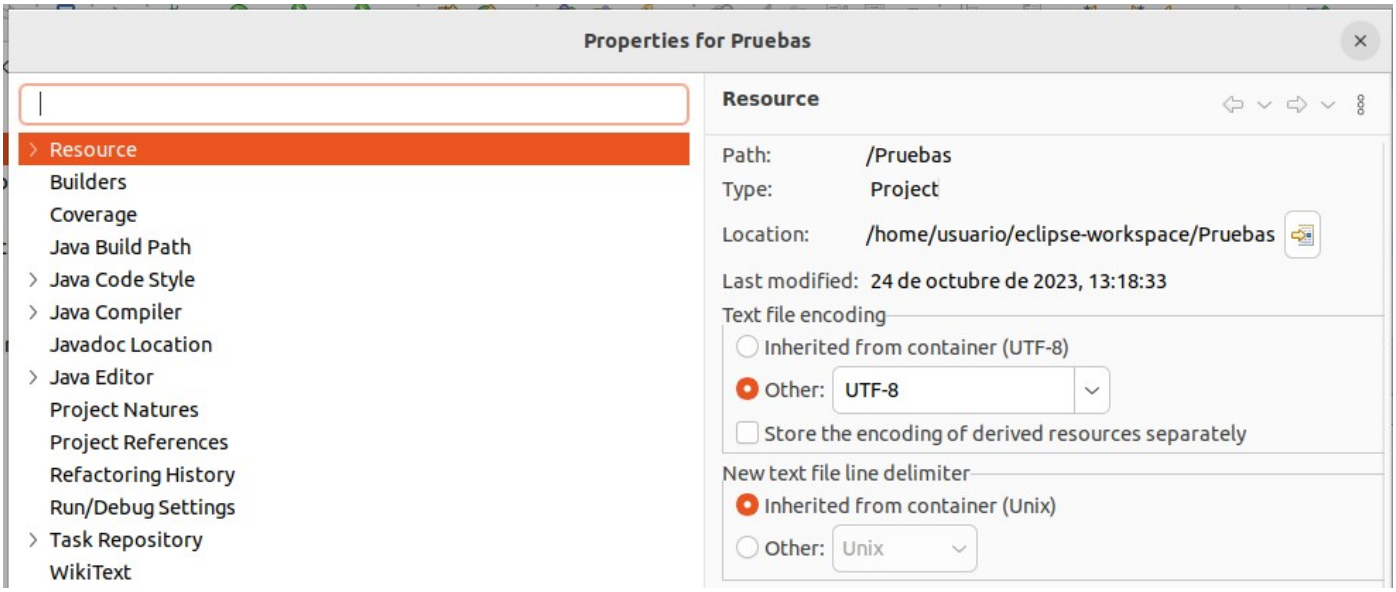
- ***src*,** con el código fuente
- ***bin***, que no se muestra en el explorador de paquetes de Eclipse y que contiene los .class, el bytecode ejecutable en la máquina virtual.

dos archivos ocultos en formato XML:

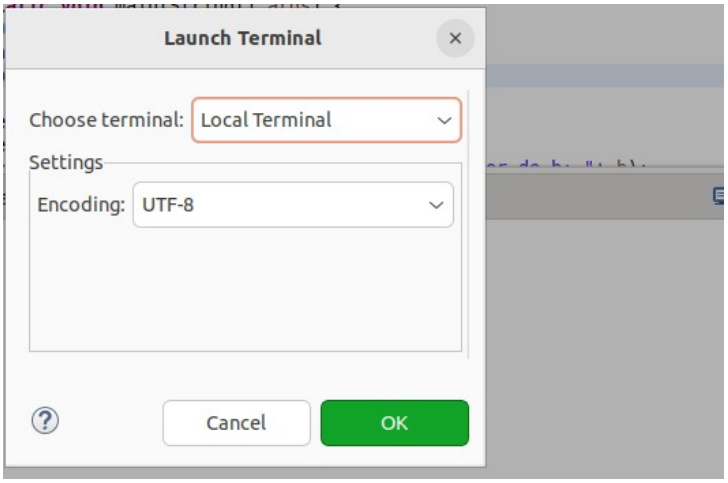
- **.project:** contiene información sobre el proyecto, como el nombre y los argumentos, entre otros;
- **.classpath:** contiene información sobre el CLASSPATH, esto es la configuración de los PATHs de las clases, indica en qué carpetas se encuentran las fuentes y dónde se generan los ficheros .class. Esto será necesario cuando un programa necesite funcionalidades que ofrecemos en otros programas.

Para abrir un terminal en Eclipse:

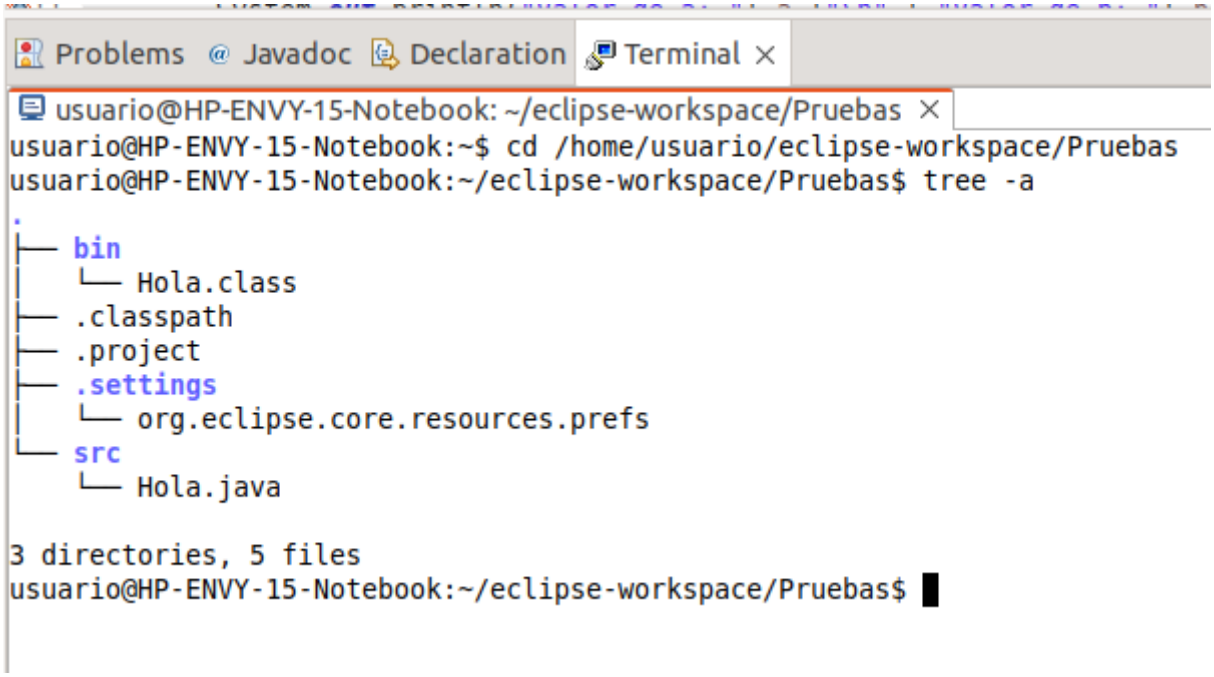
- Comprobar la ruta donde está el proyecto: clic derecho en nombre del proyecto, Properties



- Abrir el terminal: Window> Show View>Terminal>botón 



- Una vez en el directorio del proyecto, escibir los comandos necesarios(cd, tree -a)



7. ACTUALIZAR VISUAL STUDIO CODE Y ECLIPSE

- **VSCode**, el propio paquete se configuró una nueva fuente de software en nuestro sistema, **con los repositorios de Microsoft**, de modo que cada vez que se incluya una actualización en éstos y actualicemos nuestro equipo, **VSCode se actualizará automáticamente**.
- **Eclipse**, se instaló de forma manual, y éste, en principio **no se actualiza cuando actualizamos el equipo.**

VIDEO DE ACTUALIZACIÓN DE VISUAL STUDIO CODE Y ECLIPSE:
https://player.vimeo.com/video/598955274?byline=0&badge=0&portrait=0&title=0

Investiga, las posibilidades de actualización que ofrecen, tanto VSCode, como Eclipse.

• https://code.visualstudio.com/docs/

• https://wiki.eclipse.org/FAQ_How_do_I_upgrade_Eclipse_IDE%3F

8. INTEGRAR TRELLO CON VISUAL STUDIO CODE

VÍDEO EXPLICATIVO: https://player.vimeo.com/video/598955401?byline=0&badge=0&portrait=0&title=0

9. RESUMEN