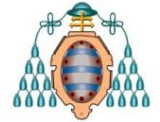




Escuela de Ingeniería Informática



UNIVERSIDAD DE OVIEDO

INTRODUCCIÓN A ECLIPSE

Metodología de la programación
Curso 2023-2024

¿Qué es Eclipse?

- [Eclipse](#) es una comunidad de código abierto que se centra en la construcción de una *plataforma compuesta por marcos extensibles (Frameworks)* y herramientas para la construcción, despliegue y gestión del software.
- La [Fundación Eclipse](#), es una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios y servicios.
- [Proyectos eclipse](#), existen multitud de proyectos:
 - El IDE (Entorno de Desarrollo Integrado) de java (JDK), aplicaciones de modelado, software para dispositivos, herramientas para la generación de informes, etc.

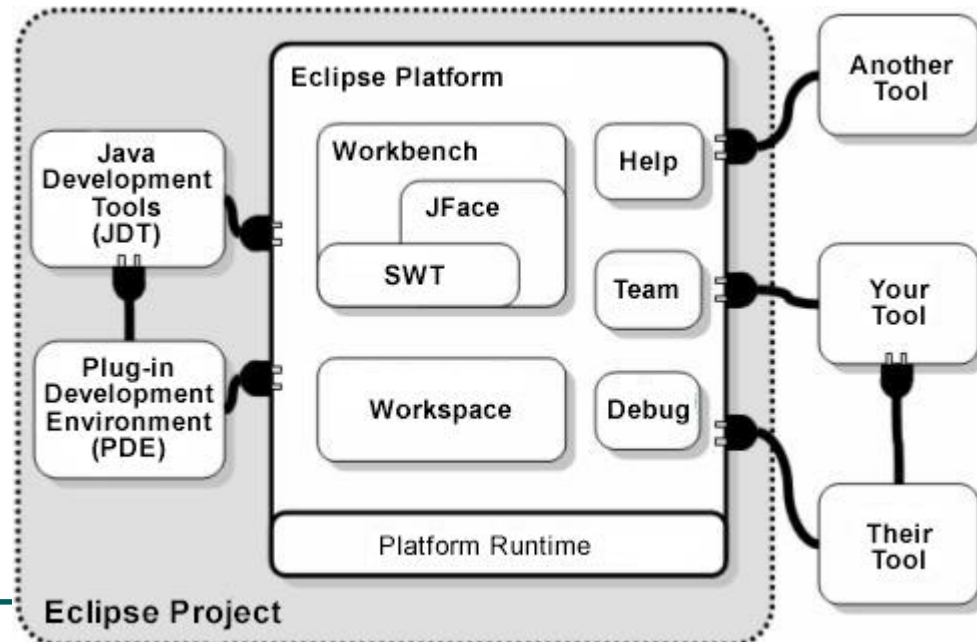
Arquitectura

- Esta basada sobre el concepto de **plug-in**
 - Código que extiende la funcionalidad del IDE.
 - Existen plug-ins para GUIs, pruebas, modelado ...

La **plataforma** está **desarrollada en java**

Soporta diferentes lenguajes:

Java, C/C++, Cobol, PHP, AspectJ, JavaScript ...



Distribución

- Descarga gratuita en la dirección:

<https://www.eclipse.org/downloads/packages/release/2022-03>

Versiones para Windows, Linux y Mac OS X.

- Cada versión (release) se libera anualmente y se identifica con un nombre.

- 2022-12

- **2022-03**

- 2021-12

- 2020-12

- Photon año 2018

- Oxigen año 2017

- Neon año 2016

- Mars año 2015

- Versión de Java Estándar Edition

- **JSE-16**

Eclipse IDE 2022-03 R Packages



Eclipse IDE for Java Developers

308 MB 1,226,611 DOWNLOADS

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



Windows x86_64
macOS x86_64 | AArch64
Linux x86_64 | AArch64

MORE DOWNLOADS

- Other builds
- Eclipse 2022-12 (4.26)
- Eclipse 2022-09 (4.25)
- Eclipse 2022-06 (4.24)
- **Eclipse 2022-03 (4.23)**
- Eclipse 2021-12 (4.22)
- Eclipse 2021-09 (4.21)
- Eclipse 2021-06 (4.20)
- Eclipse 2021-03 (4.19)
- Older Versions

Entorno de trabajo- Workbench

- El IDE (Integrated Development Environment) incluye:
 - Explorador de ficheros.
 - Editor.
 - Compilador.
 - Depurador.
- También incluye herramientas de ayuda al programador:
 - Refactorización (Refactoring)
 - Generación de código (Code generation)
 - Pruebas (Testing)

Entorno

The screenshot displays the Eclipse IDE interface with the following components:

- Package Explorer:** Located on the left, it shows the project structure. The project is named "apellido1_apellido2_nombre_session01_hellowold". Inside, there is a "src" folder containing a package "uo.mp.s1.hellowold", which in turn contains a class "HelloWorld" with a method "main(String[]): void".
- Editor:** The central workspace shows the source code of "HelloWorld.java". The code includes a Javadoc comment for the "main" method, which describes it as "Sesión 1. Imprime por pantalla Hola Mundo". The code is as follows:

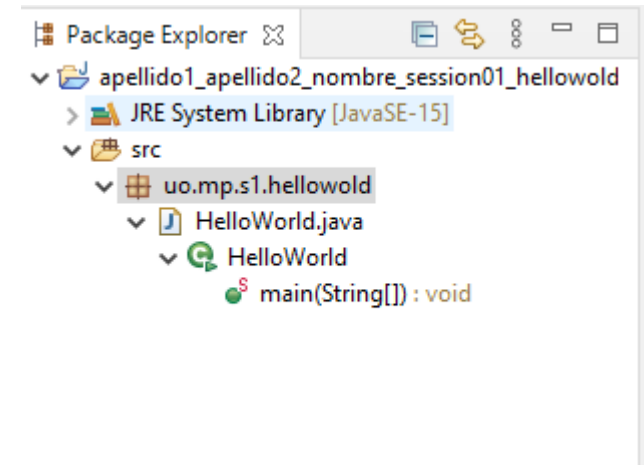
```
2 // **
3 *
4 * @author mp22
5 *
6 */
7 public class HelloWorld {
8
9     /**
10      * Sesión 1. Imprime por pantalla Hola Mundo
11      *
12      * @param args
13      * Los parámetros de entrada no se usan en este caso
14      */
15
16     public static void main(String[] args) {
17         System.out.println("Hola Mundo!");
18     }
19
20 }
```
- Outline:** Located on the right, it shows the class hierarchy. It lists "uo.mp.s1.hellowold" and "HelloWorld", with the "main(String[]): void" method listed under "HelloWorld".
- Console:** At the bottom, it shows the output of the program, which is "Hola Mundo!".

Four blue boxes with white text are overlaid on the image, each with a line pointing to a specific part of the IDE:

- Explorador de paquetes:** Points to the Package Explorer on the left.
- Editor:** Points to the central code editor.
- Perspectivas:** Points to the Outline view on the right.
- Vistas:** Points to the Console view at the bottom.

Package Explorer

- Muestra el contenido del workspace (espacio de trabajo) actual.
 - Contiene un conjunto de proyectos
 - Cada proyecto contiene diversos paquetes y librerías.
 - Los paquetes contienen clases que tienen relación entre si.
- Está sincronizado automáticamente con el sistema de ficheros.
- **Presionar F5** para actualizar la sincronización.

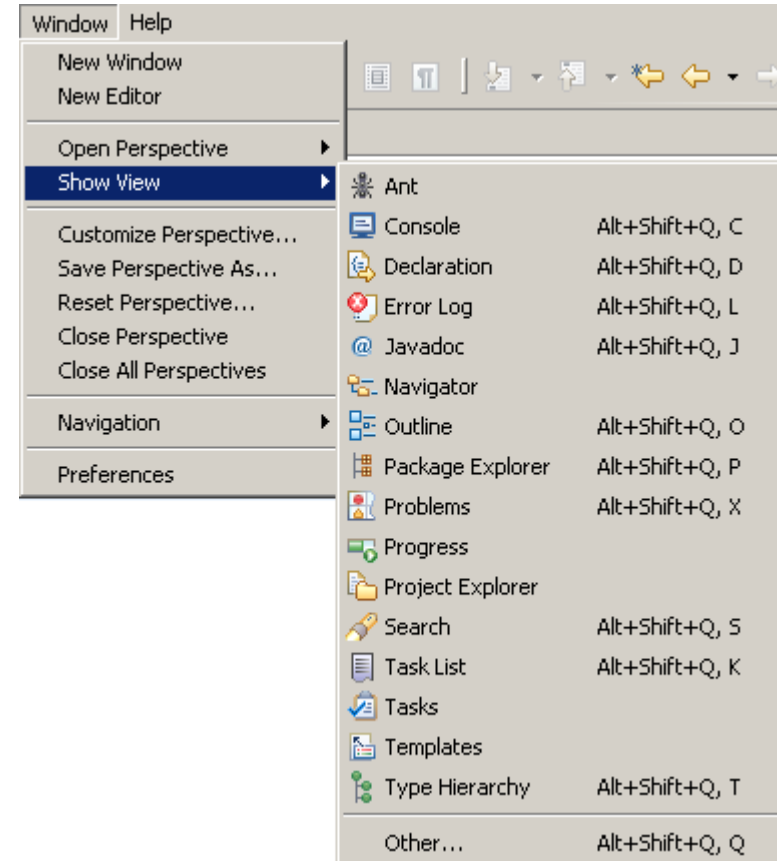


Paquete

- Un **paquete** puede ser definido como un **contenedor de tipos relacionados** (clases, interfaces, enumeraciones y anotaciones).
- Los componentes del mismo paquete están relacionados entre sí; por ejemplo, están enfocados a una función común (interfaz con el usuario, lógica de negocio, interfaz con bases de datos....)
- Los paquetes también **proporcionan protección de acceso** y gestión del espacio de nombres.

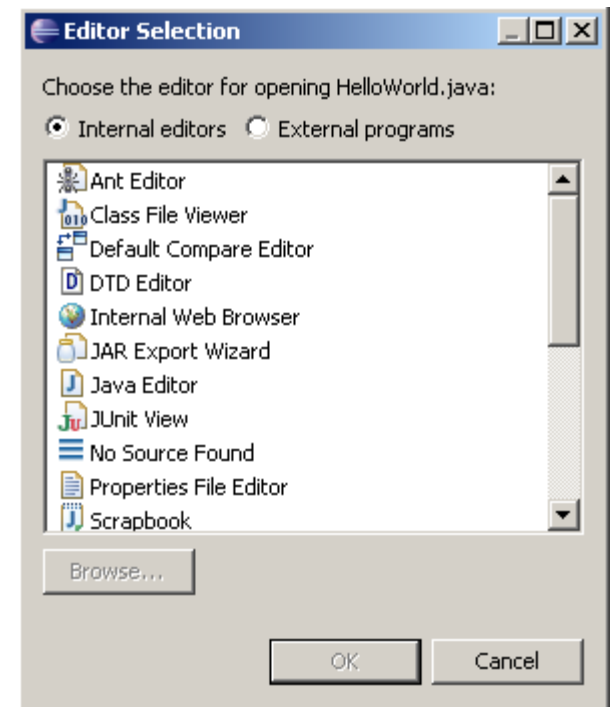
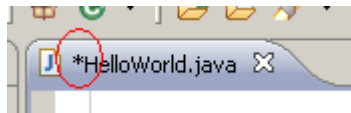
Vistas

- Es una ventana que muestra algo concreto del proyecto.
 - Consola de Java.
 - Errores de compilación.
 - etc...
- Se puede abrir cualquier vista desde el menú.
- Aparecen agrupadas y solo se puede ver una.
- No permite modificar.



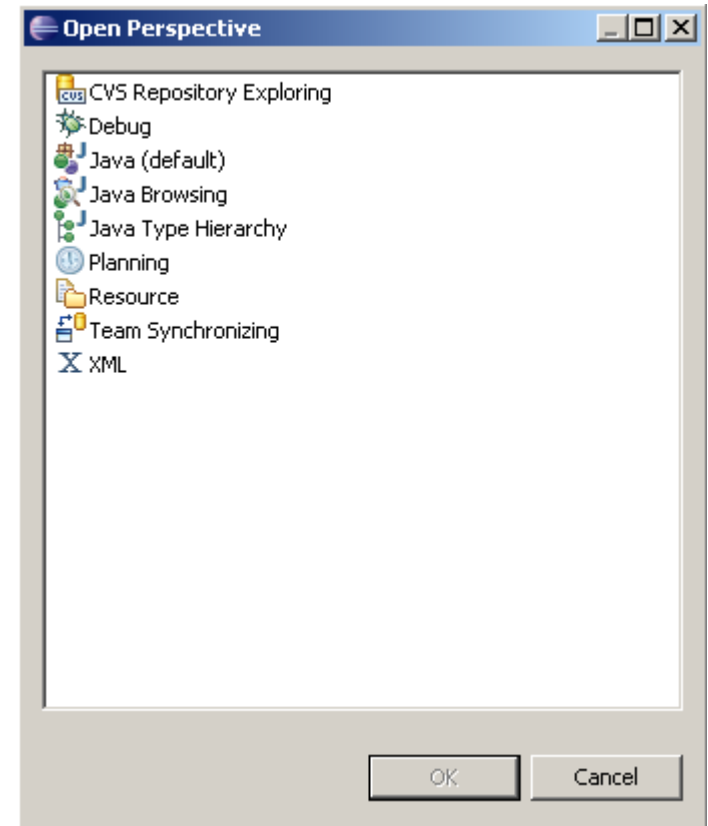
Editores

- Similares a las vistas.
- Múltiples editores, gráficos o de texto.
- Asociados por la extensión.
 - Configurable desde las preferencias.
- Es posible forzar el uso del editor que queramos.
- Un “*” al lado del nombre del archivo => sin salvar.



Perspectivas

- Adaptan el entorno para una tarea de alto nivel.
- Seleccionan un conjunto de vistas, editores y barras de herramientas adecuados a la tarea.
 - **Java:** Desarrollo en java.
 - **Debug:** Depuración.
 - **CVS:** Control de versiones.
 - etc...
- En cualquier momento se puede cambiar la perspectiva.

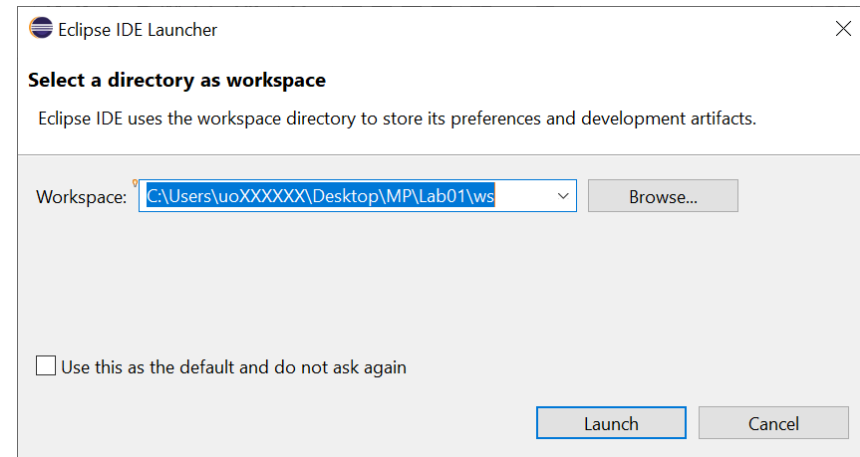


Workspace (Espacio de trabajo)

- Un **workspace** es una carpeta donde se almacena un conjunto de proyectos
- Guarda el estado de cada proyecto .
- Se pueden mantener tantos espacios de trabajo como se quiera.
- **Usaremos un workspace para cada sesión de laboratorio**
- Llamaremos al espacio de trabajo ws para la sesión.
- Después de cada clase de laboratorio se deben exportar los proyectos del workspace y subirlos a OneDrive, Dropbox...

Arrancar el entorno

- Crea la **carpeta lab01** dentro de **carpeta MP** y guarda aquí el material de esta sesión bajado del campus.
- Crea el espacio de trabajo (carpeta ws) dentro de lab01
`c:\Users\uoXXXXXX\desktop\MP\lab01\ws`
- Ejecuta Eclipse.
 - eclipse.exe (icono eclipse del escritorio)
- Selecciona el espacio de trabajo.





CÓMO SE CREA UN NUEVO PROYECTO

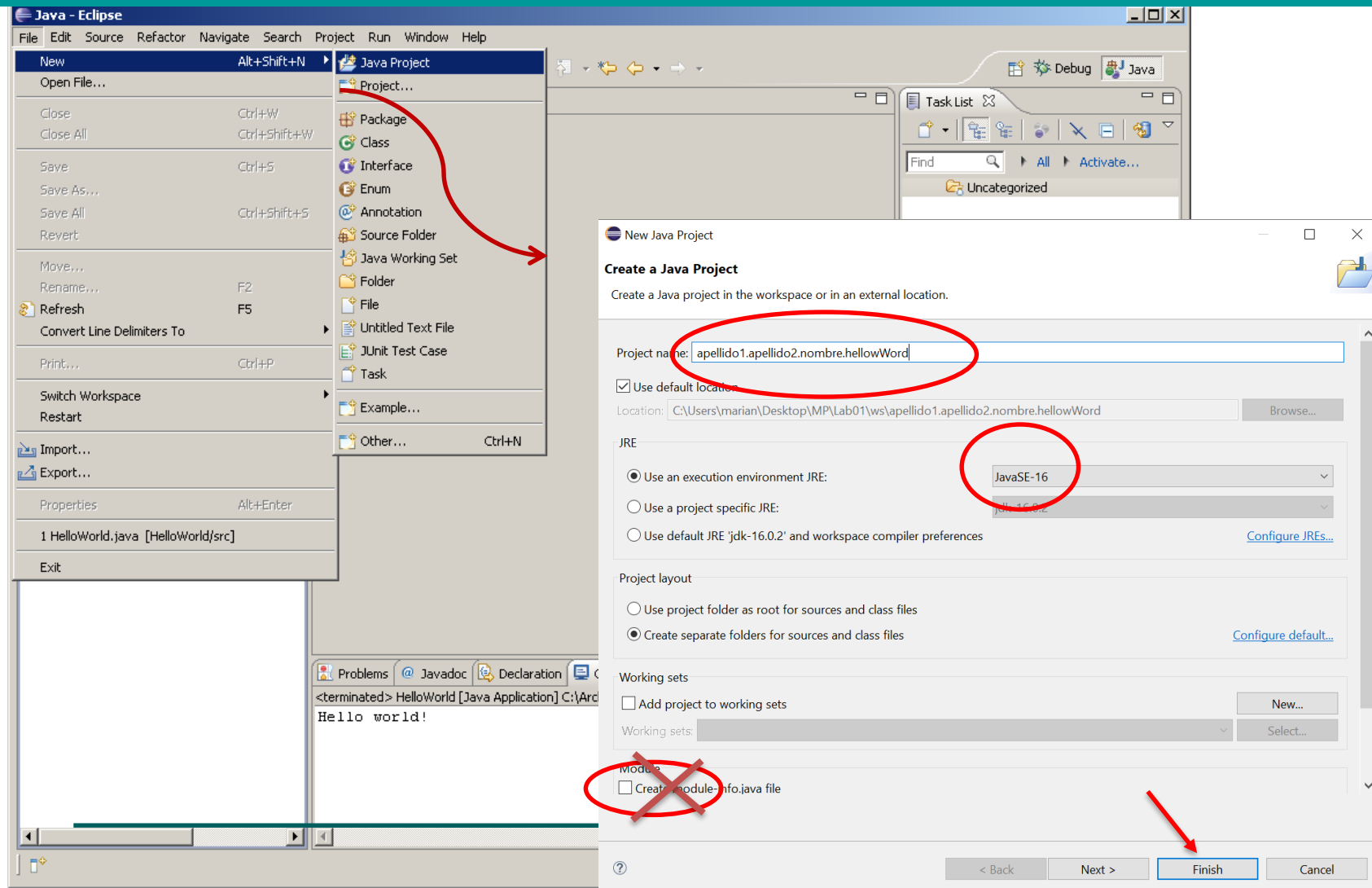
Organización y nomenclatura

- **Carpeta para la asignatura:** MP
- **Carpeta para cada sesión del curso (semanal):** lab01
lab01, lab02...lab10, (para que queden ordenadas)
 - *Guarda toda la documentación que se baje del campus y los proyectos que se hagan.*
- **Espacio de Trabajo para la sesión:** ws
 - Carpeta que guarda todos los proyectos que se realicen
- **Nombre de los proyectos**
 - *apellido1_apellido2_nombre_lab01_helloworld*
*Todo en **minúsculas** y en **inglés***
 - *apellido1_apellido2_nombre_lab01_task_game*
*Las tareas (a entregar) llevan la palabra **task***
- **Nombre del paquete**
 - *uo.mp.lab01.game.model* (siempre uo.mp. Además laboratorio, proyecto, paquete)
 - Su último nombre depende del contenido. Se suelen incluir paquetes como **model**, **service**, **etc**. Según la función
 - Ejemplo: uo.mp.lab01.game.model

Ejercicio

- Crear un proyecto llamado helloworld que al ejecutarlo muestre por pantalla “Hola Mundo”
 - Se crea un proyecto:
 - apellido1_apellido2_nombre_lab01_helloworld
 - Se crea un paquete en la carpeta src
 - uo.mp.lab01.helloworld
 - Como el proyecto es muy simple, el nombre del paquete coincide con el nombre del proyecto
 - Se crea una **clase con el método estático Main**
 - HelloWorld (Sigue norma de denominación de clases)

Crear un nuevo proyecto



Crear un paquete

New Java Package

Java Package

Create a new Java package.

Creates folders corresponding to packages.

Source folder: [Browse...](#)

Name:

☒ Create package-info.java

☐ Generate comments (configure templates and default value [here](#))

[?](#) [Finish](#) [Cancel](#)

Crear una clase

Sesion1 - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

New Alt+Shift+N >

Open File...

Open Projects from File System...

Recent Files

Close Ctrl+W

Close All Ctrl+Shift+W

Save Ctrl+S

Save As...

Save All Ctrl+Shift+S

Revert

Move...

Rename... F2

Refresh F5

Convert Line Delimiters To

Print... Ctrl+P

Import...

Export...

Properties Alt+Enter

Switch Workspace >

Restart

Exit

Java Project

Project...

Package

Class

Interface

Enum

Annotation

Source Folder

Java Working Set

Folder

File

Untitled Text File

Task

JUnit Test Case

Example...

Other... Ctrl+N

New Java Class

Java Class

Create a new Java class.

Source folder: apellido1_apellido2_nombre_lab01_helloworld/src Browse...

Package: uo.mp.lab01.helloworld Browse...

☐ Enclosing type: Browse...

Name: HelloWorld

Modifiers: ☒ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass: java.lang.Object Browse...

Interfaces: Add... Remove

Which method stubs would you like to create:

☒ public static void main(String[] args)

☐ Constructors from superclass

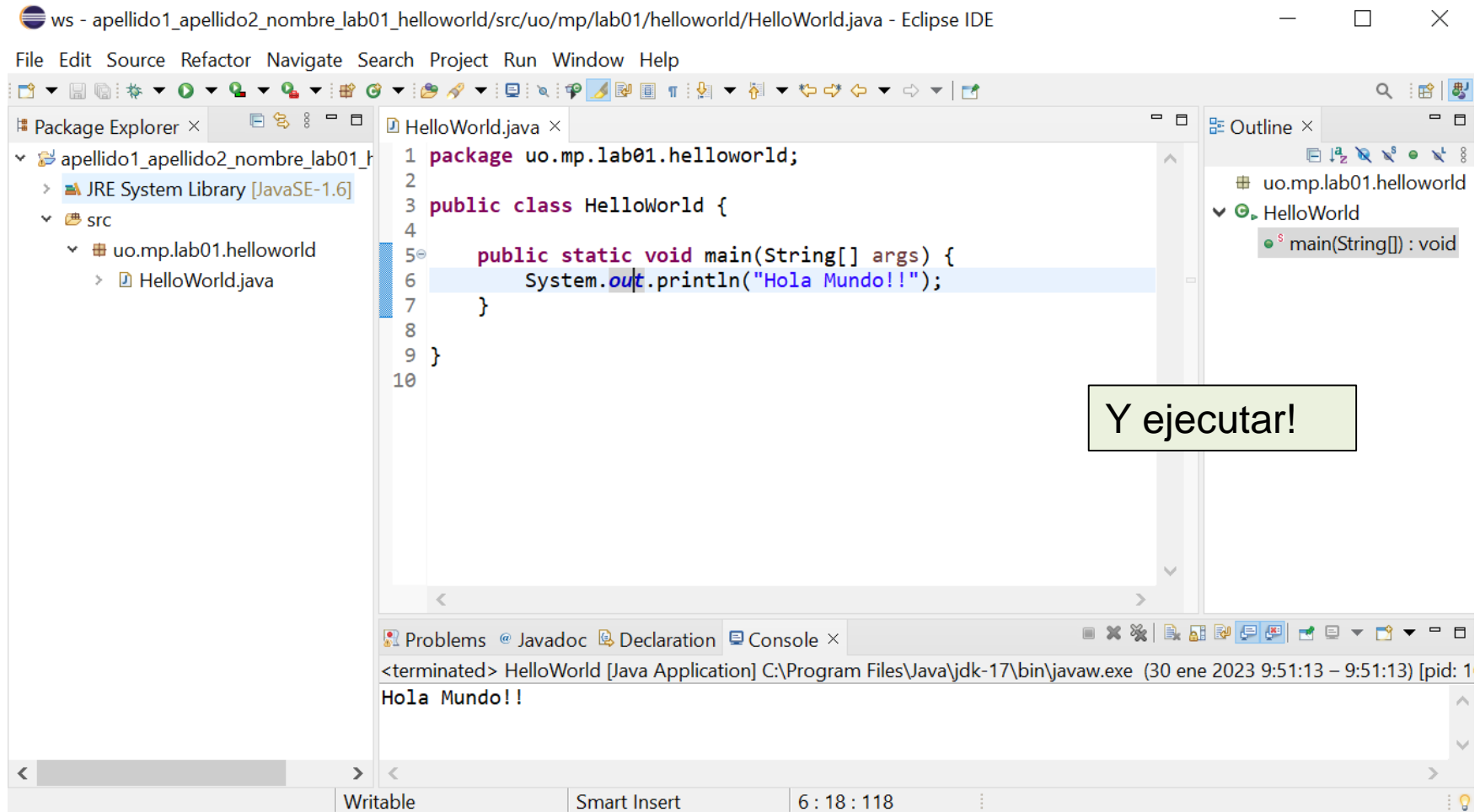
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

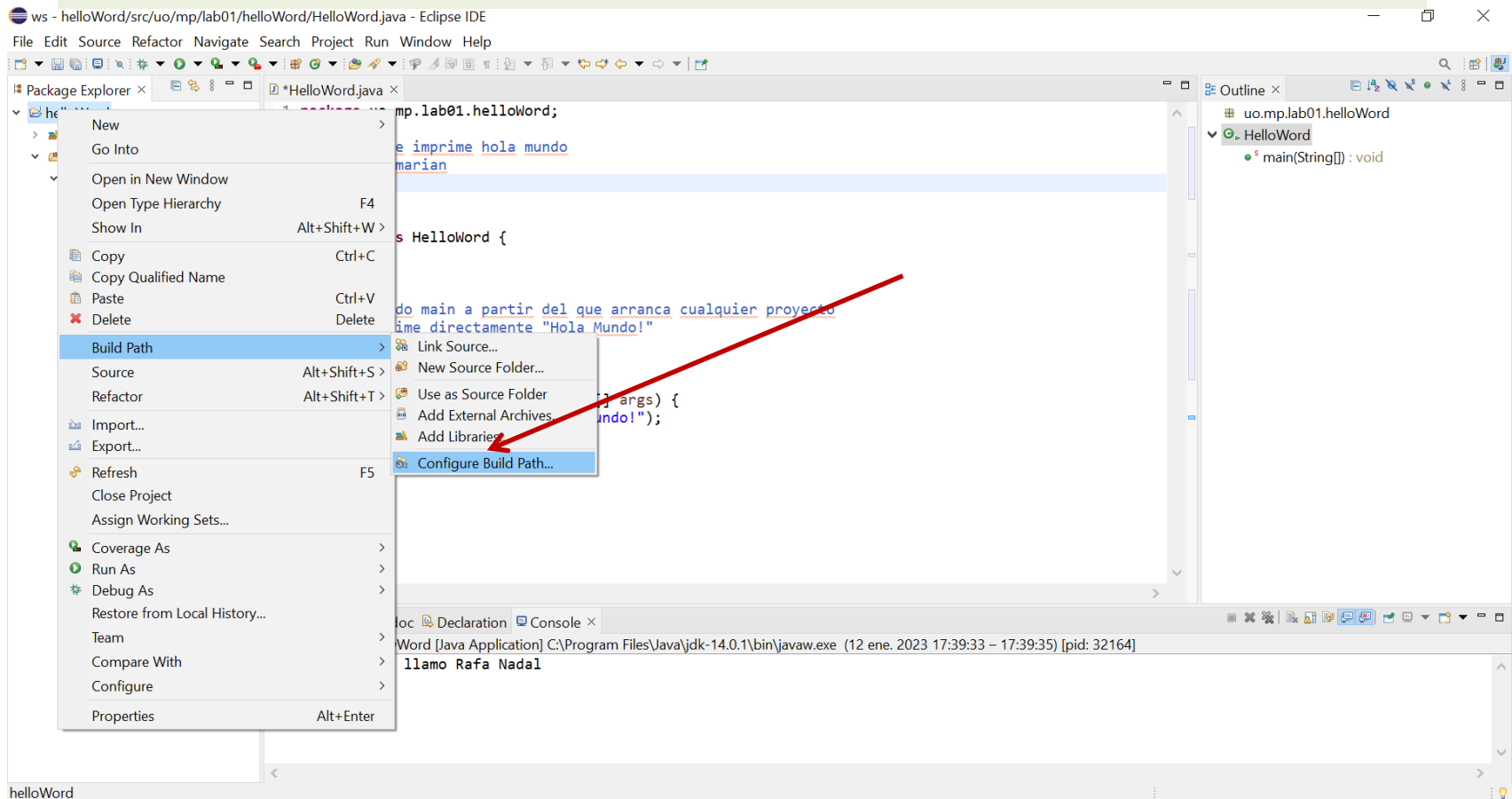
Finish Cancel

Editar para añadir el código



Configuración del proyecto

Sobre el proyecto botón derecho
build Path /configure build path



helloWord

Configuración del proyecto

Asegurarse que tiene la versión de **Java JSE-16**

The screenshot shows the Eclipse IDE interface with the following elements:

- Package Explorer:** Shows the project structure with 'src' and 'JRE' folders.
- Editor:** Displays the code for 'HelloWord.java' with the following content:

```
lab01.helloWord;  
prime hola mundo  
ian  
HelloWord {  
    main a partir del que arranca cualquier  
    directamente "Hola Mundo!"  
}
```
- Context Menu:** A right-click menu is open over the code, with 'Configure Build Path...' selected. Other options include 'New', 'Go Into', 'Open in New Window', 'Open Type Hierarchy', 'Show In', 'Copy', 'Copy Qualified Name', 'Paste', 'Delete', 'Build Path', 'Source', 'Refactor', 'Import...', 'Export...', 'Refresh', 'Close Project', 'Assign Working Sets...', 'Coverage As', 'Run As', 'Debug As', 'Restore from Local History...', 'Team', 'Compare With', 'Configure', and 'Properties'.
- Build Path Dialog:** A dialog titled 'Add Library' is open, showing the 'JRE System Library' tab. The 'Execution environment' is set to 'CDC-1.0/Foundation-1.0 (jdk-16.0.2)'. A list of installed JREs is shown, with 'JavaSE-16 (jdk-16.0.2)' selected. A red arrow points to the 'Classpath' tab in the 'Build Path' dialog.
- Taskbar:** The Windows taskbar at the bottom shows the search bar with the text 'Escribe aquí para buscar' and the system clock displaying '18:20' and '12/01/2023'.

Uso de parámetros

ws - helloWord/src/uo/mp/lab01/helloWord/HellowWord.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer ×

helloWord

JRE System Library [JavaSE-8]

src

*HellowWord.java ×

```
1 package uo.mp.lab01.helloWord;  
2  
3 /**  
4  * Clase que imprime hola mundo  
5  */
```

Run Configurations

Create, manage, and run configurations

Run a Java application

Name: HelloWorld

Main Arguments

Program arguments:

Rafa Nada

VM arguments:

Show Command Line Revert Apply

Run Close

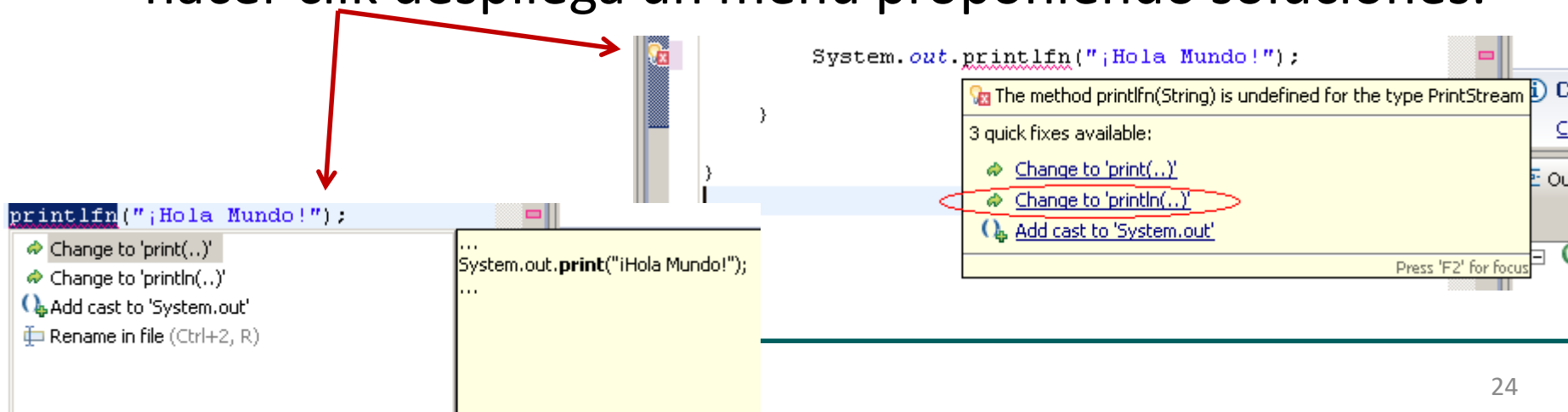
alquier proyecto
tamente para imprimir
no se usan los argumentos (o parámetros)

```
ing[] args) {  
a Mundo!, me llamo " + args[0] + " " + args[1]);
```

Y ejecutar!

Detección y corrección de errores

- Similar a un corrector ortográfico.
- Subraya el error con una línea roja ondulada (amarilla si es un warning).
- Situando el ratón sobre la línea muestra una descripción del error y puede proponer soluciones.
- A la izquierda muestra un icono de error en el que al hacer clic despliega un menú proponiendo soluciones.



Manipulación del código fuente

- Opciones agrupadas en el menú “Source”
- Source → Format : Formatea e indenta el código fuente.
- Shortcut: Control + Shift + F

The image displays two side-by-side screenshots of an IDE editor window titled `*HelloWorld.java`. A red arrow points from the left window to the right one, indicating a transformation of the code.

Left Window (Before Format):

```
1 package uo.mp.s1.helloworld;
2 /**
3  *
4  * @author mp22
5  *
6  */
7 public class HelloWorld {
8
9     /**
10      * Sesión 1. Imprime por pantalla Hola Mundo
11      *
12      * @param args
13      * Los parámetros de entrada no se usan en este caso
14      */
15
16     public static void main(String[] args) {
17         System.out.println
18 ("Hola Mundo!");
19         if (true) System.out.println("Siempre se imprime");
20     }
21
22 }
23
```

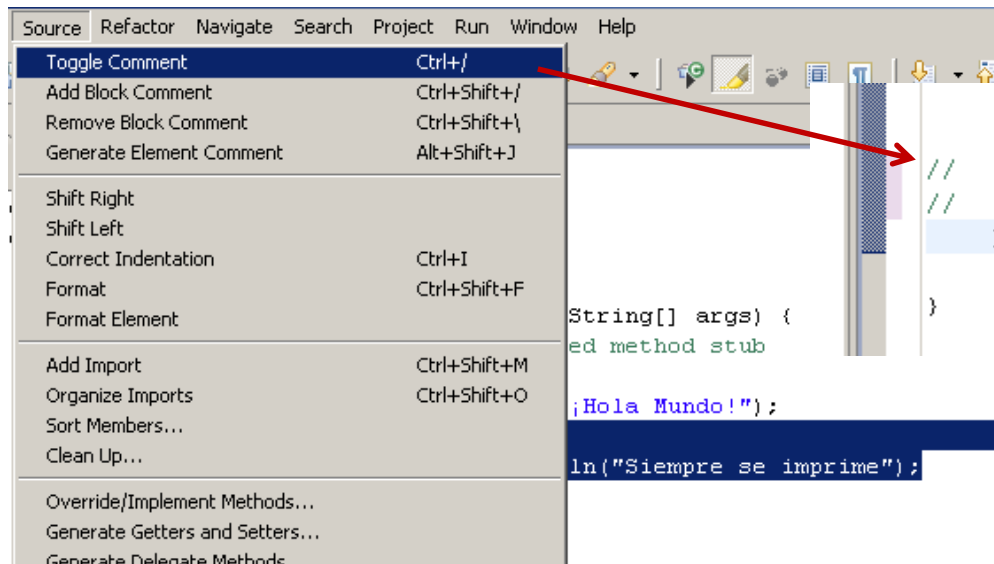
Right Window (After Format):

```
1 package uo.mp.s1.helloworld;
2
3 /**
4  *
5  * @author mp22
6  *
7  */
8 public class HelloWorld {
9
10     /**
11      * Sesión 1. Imprime por pantalla Hola Mundo
12      *
13      * @param args Los parámetros de entrada no se usan en este caso
14      */
15
16     public static void main(String[] args) {
17         System.out.println("Hola Mundo!");
18         if (true)
19             System.out.println("Siempre se imprime");
20     }
21
22 }
```

The transformation shows that the code has been formatted with consistent indentation and line wrapping. Specifically, the `if` statement in the `main` method is now properly indented, and the long comment line is wrapped across multiple lines.

Código fuente: Comentar bloques

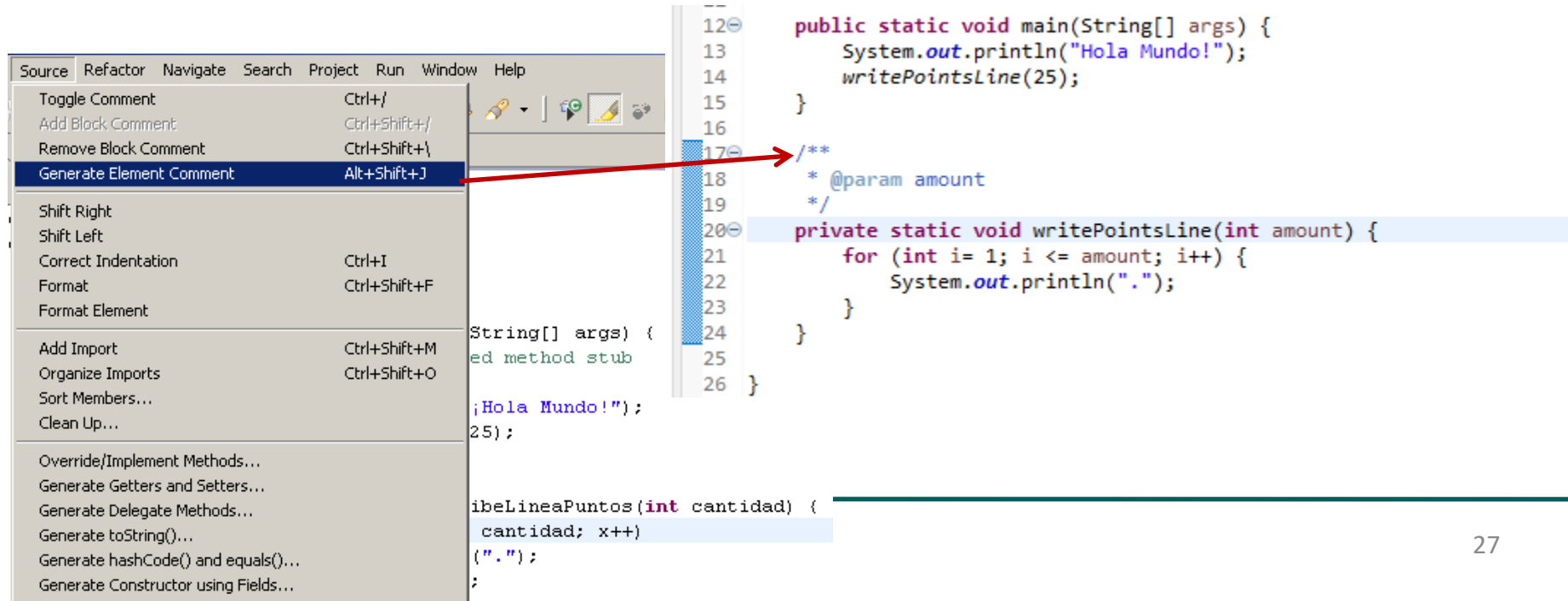
- Comentar y (des) comentar bloques de código previamente seleccionados.
- Source → Toggle Comment : Utiliza `//`
Shortcut: **Control + /**
- Source → Add Block Comment: Utiliza `/* ... */`



```
// System.out.println(";Hola Mundo!");  
// if (true)  
//     System.out.println("Siempre se imprime");  
}  
  
String[] args) {  
    ed method stub  
  
    ;Hola Mundo!);  
  
    ln("Siempre se imprime");
```

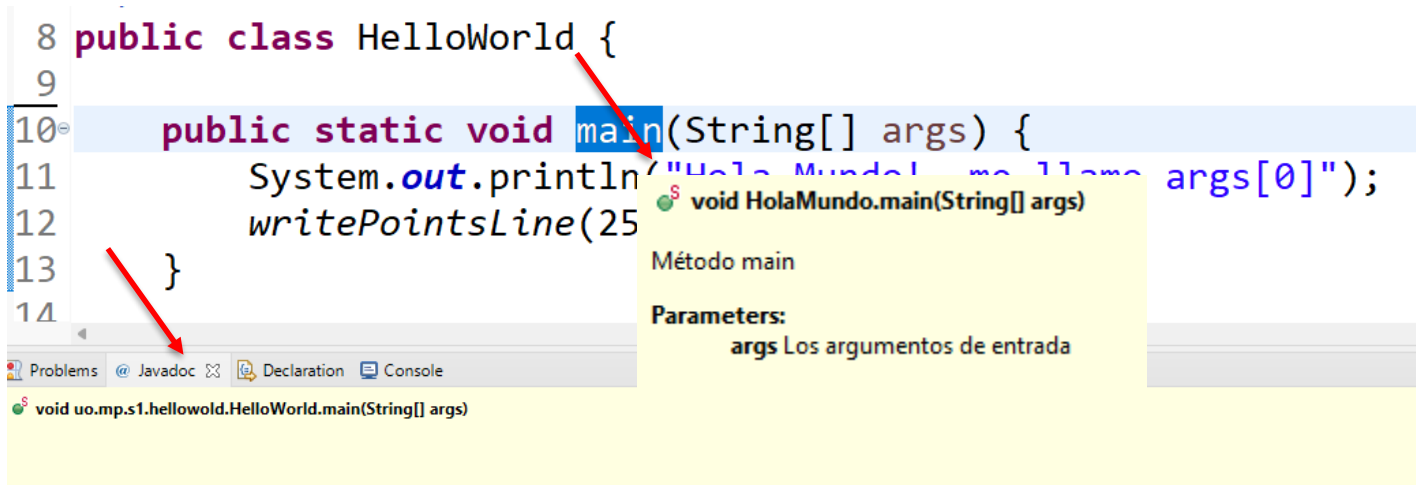
Código fuente: Comentarios Javadoc

- Añadir bloques apropiados para generar Javadoc a un elemento (método o clase).
 - Source → Generate Element Comment
 - o bien `/**` y presionando enter



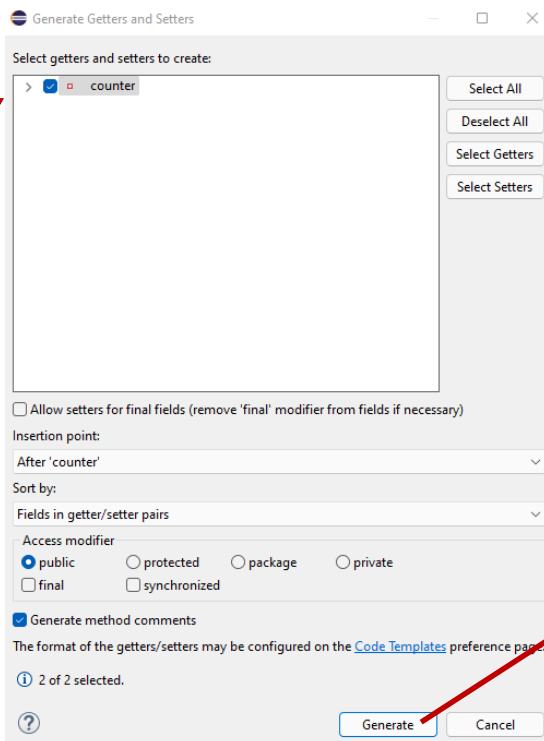
Comentarios Javadoc

- La información de Javadoc es usada:
 - En la **vista Javadoc** cuando se selecciona un elemento
 - En la **ventana emergente** cuando se coloca el ratón sobre un elemento.



Código fuente: Generar código

- Utiliza plantillas para añadir código. Ver `Source`
- Por ejemplo para generar métodos get y set:
 - `Source` → `Generate Getters and Setters ...`



```
private int counter;

/**
 * @return the counter
 */
public int getCounter() {
    return counter;
}

/**
 * @param counter the counter to set
 */
public void setCounter(int counter) {
    this.counter = counter;
}
```

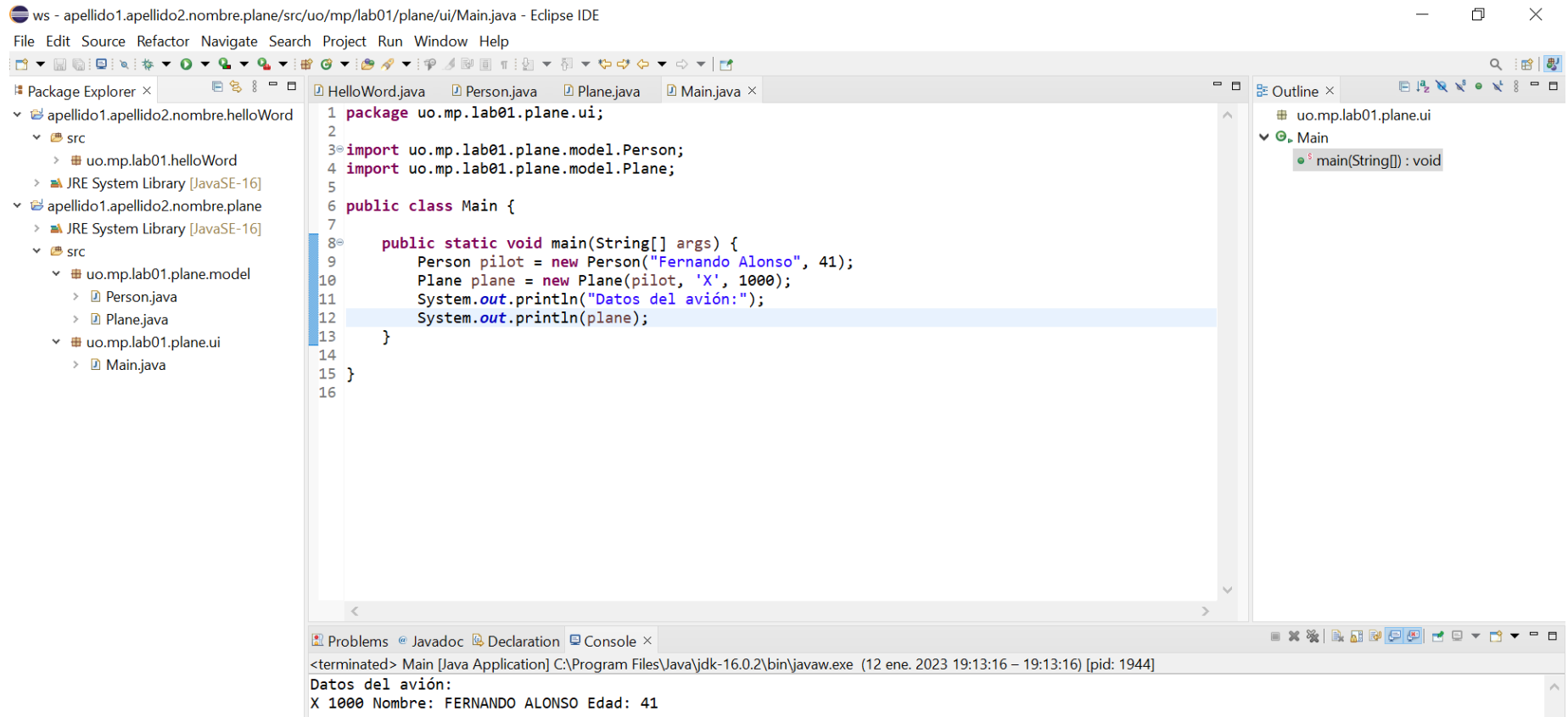


CÓMO EJECUTAR UN PROYECTO CREADO EN BLUEJ

Proyecto plane

- Creamos un nuevo proyecto para cargar las clases Plane y Person desarrolladas en IP y usarlas.
 - Creamos nuevo proyecto
apellido1_apellido2_nombre_lab01_plane
 - Creamos paquete `uo.mp.lab01.plane.model`
 - Copiamos en él las clases Plane y Person
 - Creamos paquete `uo.mp.lab01.plane.ui`
 - Creamos en él la clase Main con el método estático main
 - Dentro del método main
 - Creamos un piloto
 - Creamos un avión pasando el piloto, un identificador y combustible
 - Imprimimos por consola los datos del avión

Proyecto plane



ws - apellido1.apellido2.nombre.plane/src/uo/mp/lab01/plane/ui/Main.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer ×

- apellido1.apellido2.nombre.helloWord
 - src
 - uo.mp.lab01.helloWord
- apellido1.apellido2.nombre.plane
 - JRE System Library [JavaSE-16]
 - src
 - uo.mp.lab01.plane.model
 - Person.java
 - Plane.java
 - uo.mp.lab01.plane.ui
 - Main.java

1 package uo.mp.lab01.plane.ui;
2
3 import uo.mp.lab01.plane.model.Person;
4 import uo.mp.lab01.plane.model.Plane;
5
6 public class Main {
7
8 public static void main(String[] args) {
9 Person pilot = new Person("Fernando Alonso", 41);
10 Plane plane = new Plane(pilot, 'X', 1000);
11 System.out.println("Datos del avión:");
12 System.out.println(plane);
13 }
14
15 }
16

Outline ×

- uo.mp.lab01.plane.ui
 - Main
 - main(String[]) : void

Problems × Javadoc Declaration Console ×

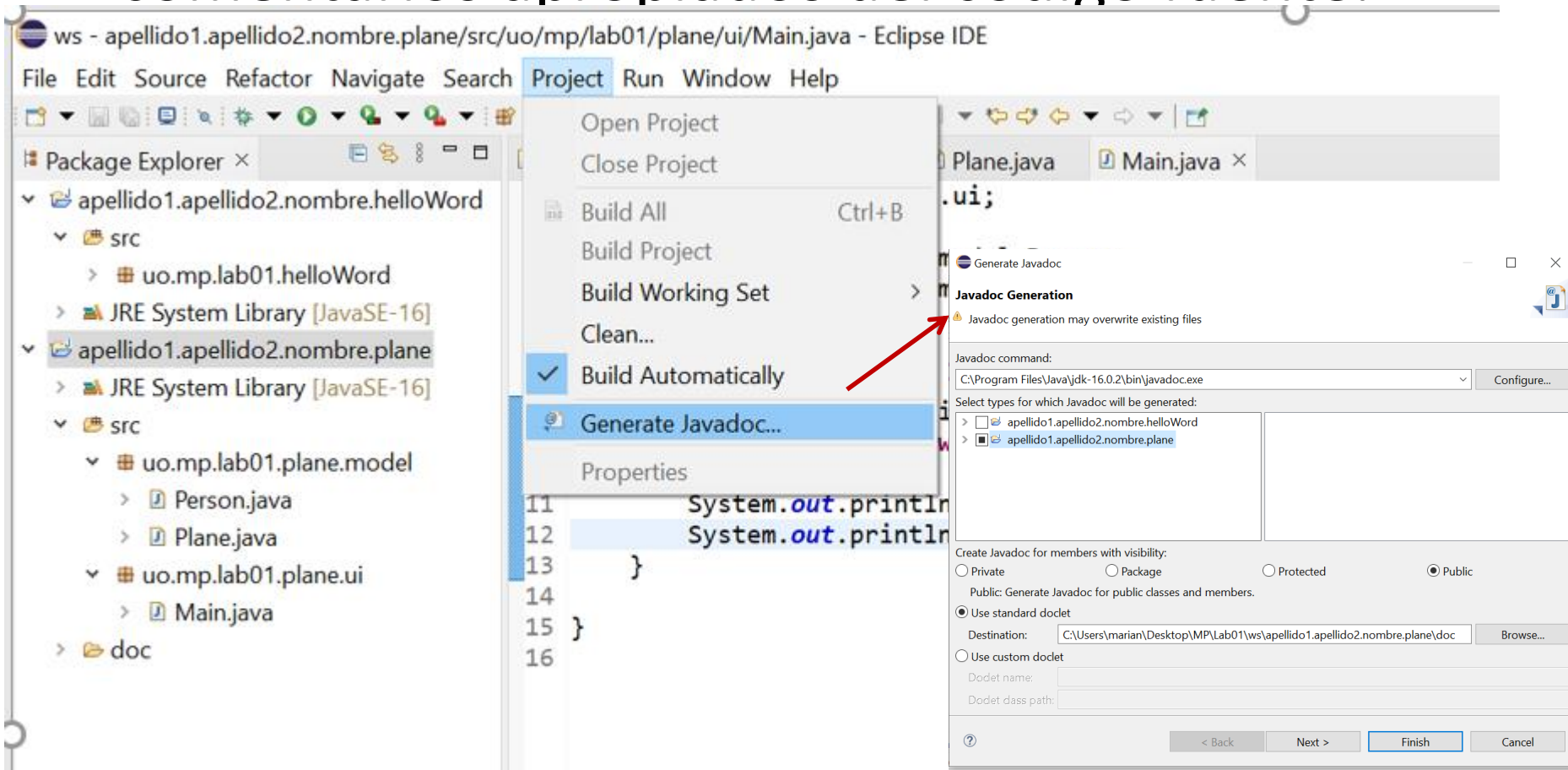
<terminated> Main [Java Application] C:\Program Files\Java\jdk-16.0.2\bin\javaw.exe (12 ene. 2023 19:13:16 – 19:13:16) [pid: 1944]
Datos del avión:
X 1000 Nombre: FERNANDO ALONSO Edad: 41



CÓMO SE GENERA LA DOCUMENTACIÓN CON JavaDoc

Generar Javadoc

- Me permite generar el Javadoc a partir de los comentarios apropiados del código fuente.

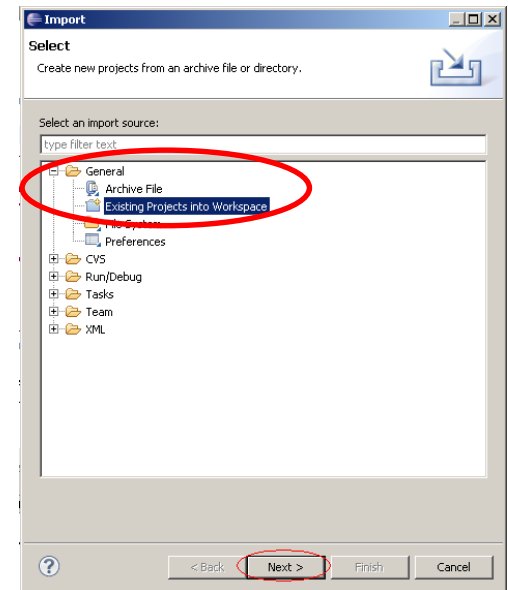




CÓMO SE ABRE UN PROYECTO YA CREADO

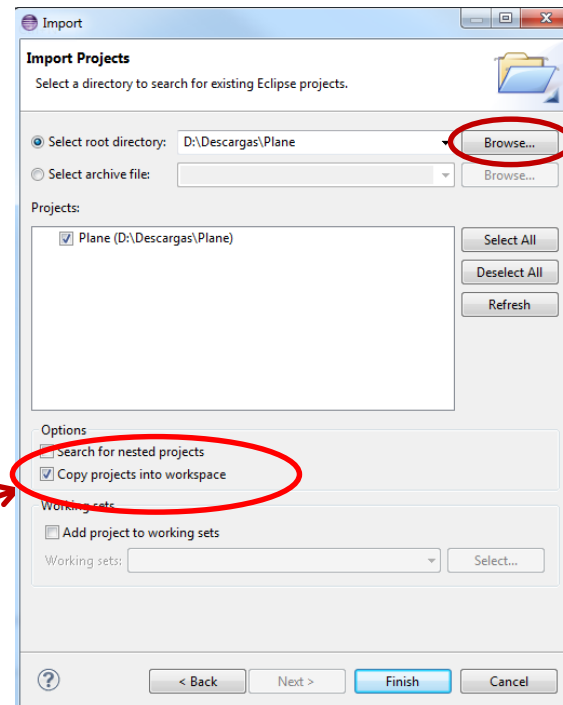
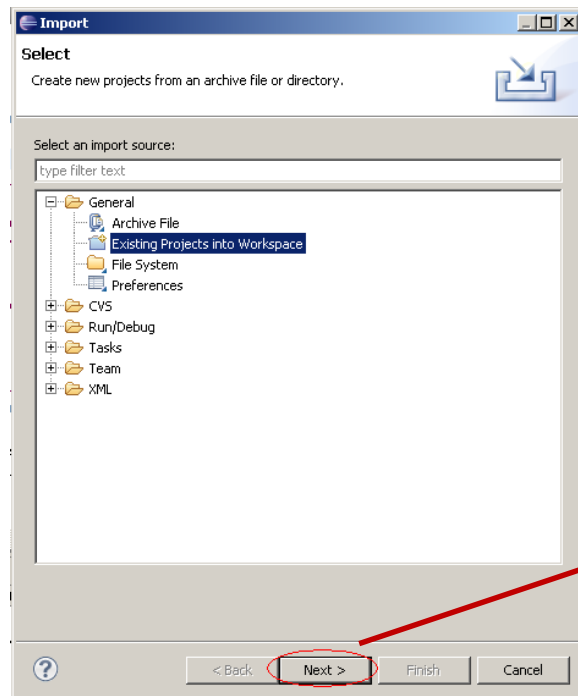
Importar un proyecto

- Menú File → Import
/General/Existing Projects into Workspace...
 - Dos posibilidades
 1. Seleccionar directorio
La carpeta con el proyecto
 2. Seleccionar fichero
El proyecto comprimido
- > Más cómodo usar segunda opción



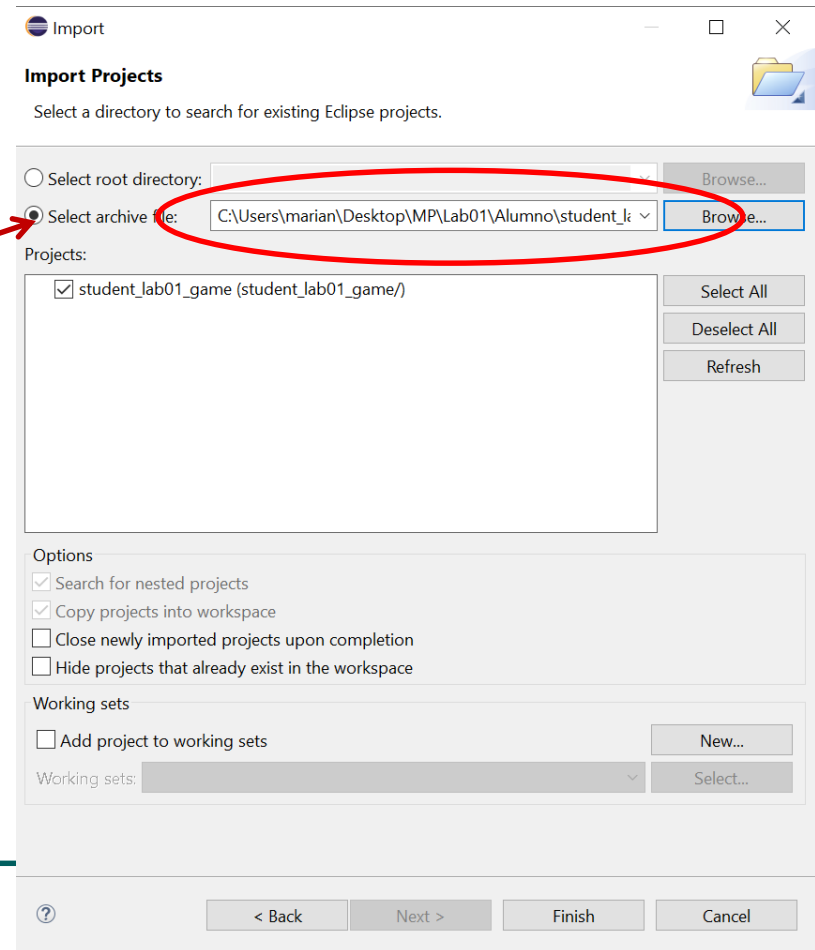
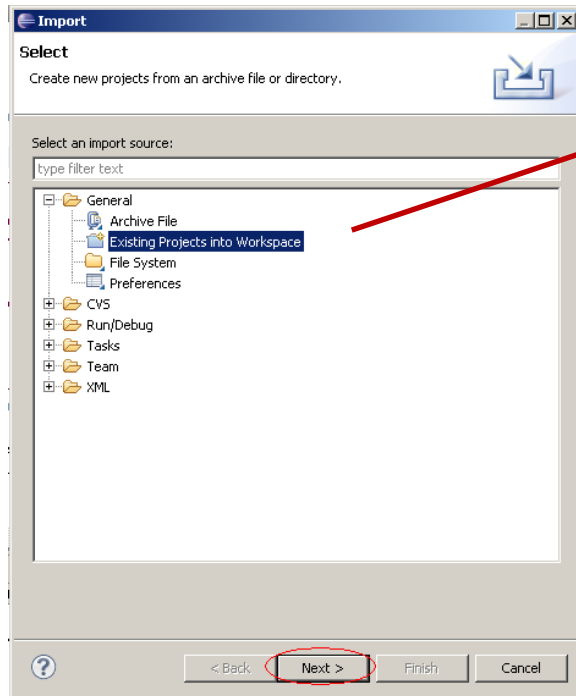
Importar un proyecto

- Si se importa proyecto sin comprimir es importante hacer una copia en el espacio de trabajo
- Si se importa proyecto comprimido ya la hace el sistema

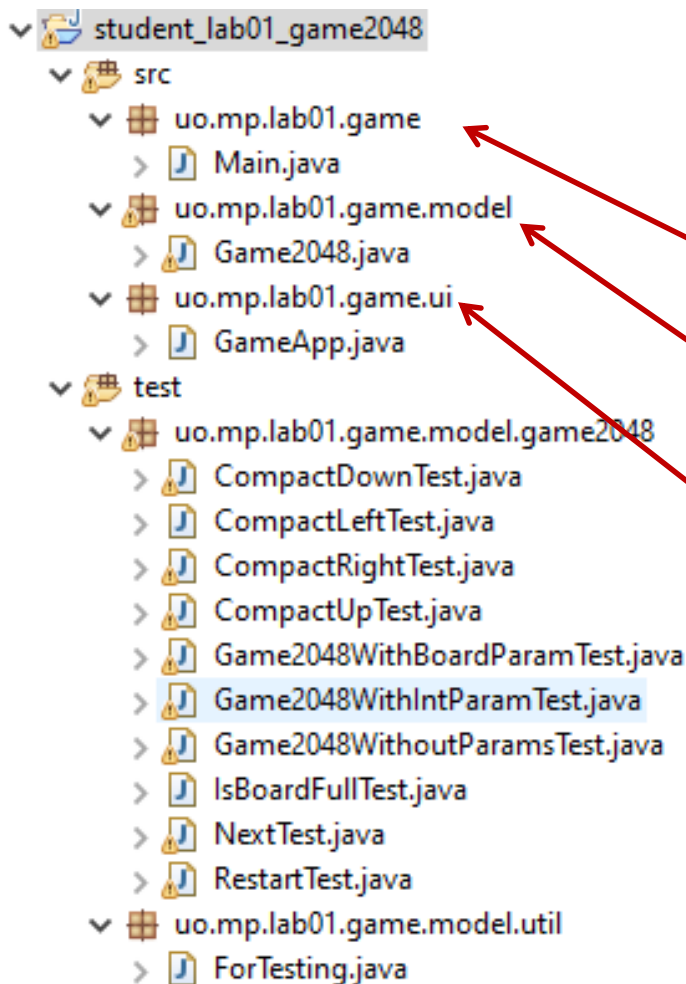


Importar un proyecto

- Importaremos el proyecto para la tarea student_lab01_game2048.



Analizar el proyecto



```
1 package uo.mp.lab01.game;  
2  
3 import uo.mp.lab01.game.ui.GameApp;  
4  
5 public class Main {  
6  
7     public static void main(String[] args) {  
8         new GameApp().run();  
9     }  
10  
11 }
```

Paquete raíz

Modelo de datos

Interfaz de usuario

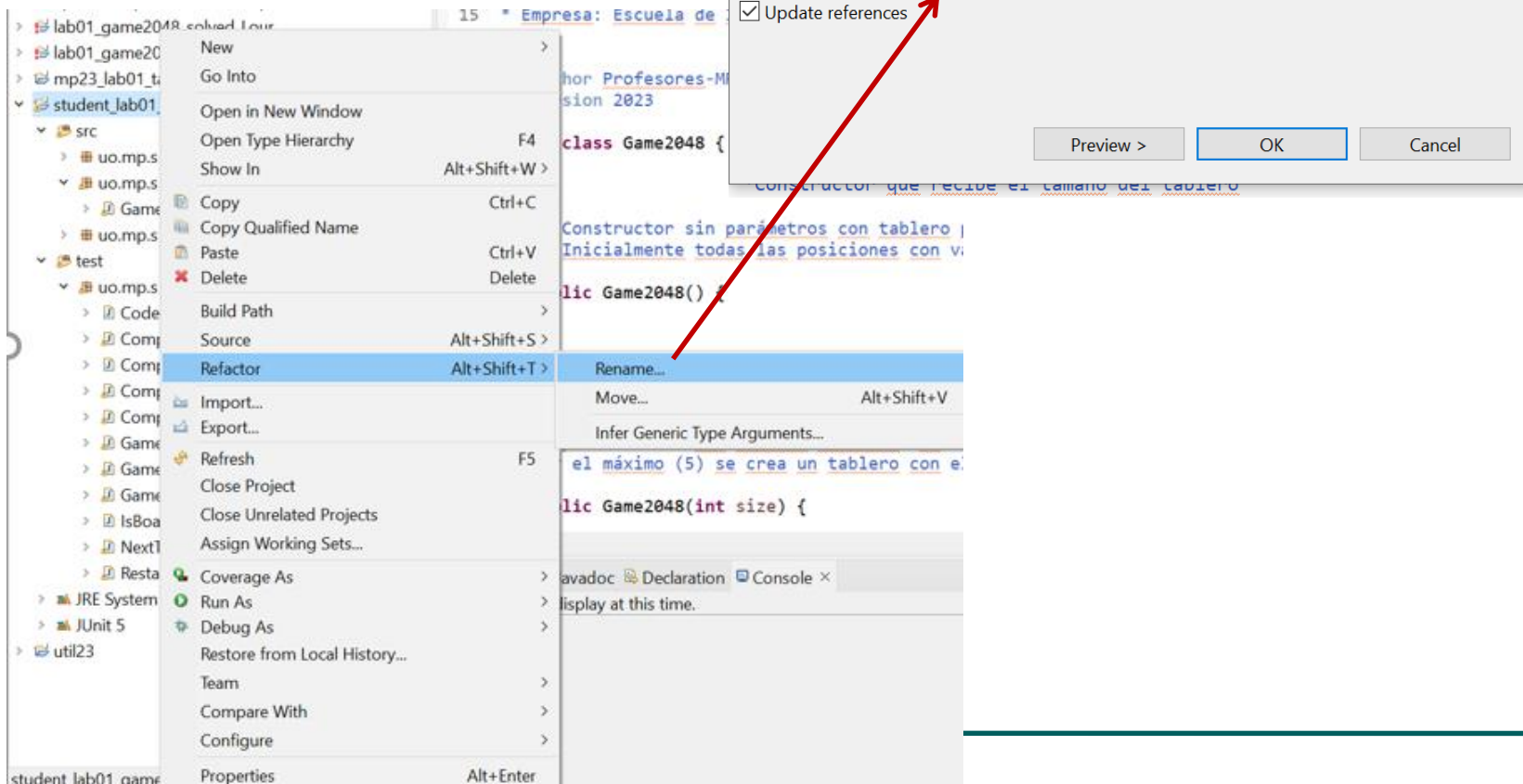
PAQUETES

El main solo crea la aplicación y la lanza

Encadenamiento de llamadas

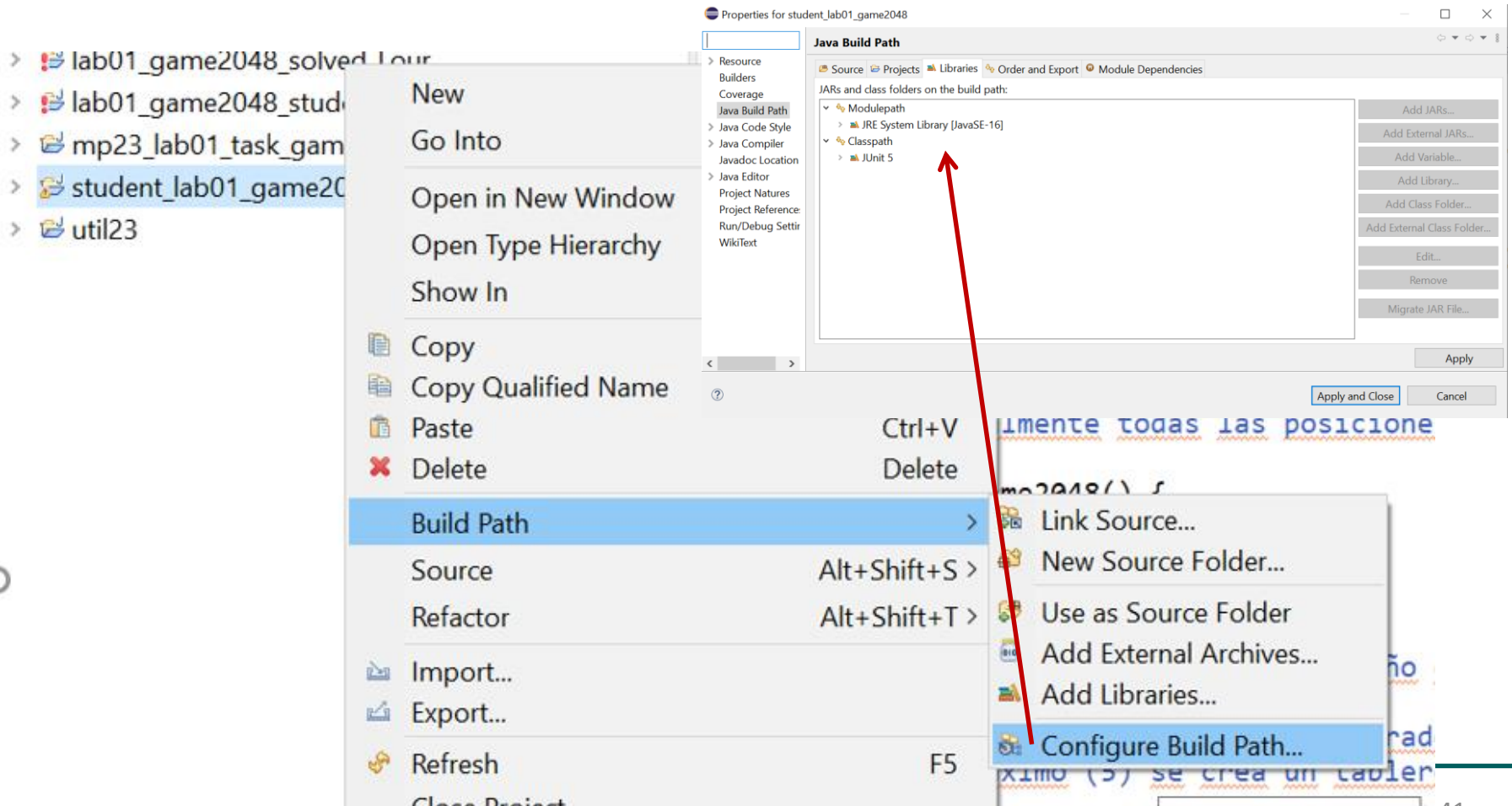
Analizar el proyecto

- **Renombrar** proyecto sustituyendo student por apellido1_apellido2_nombre



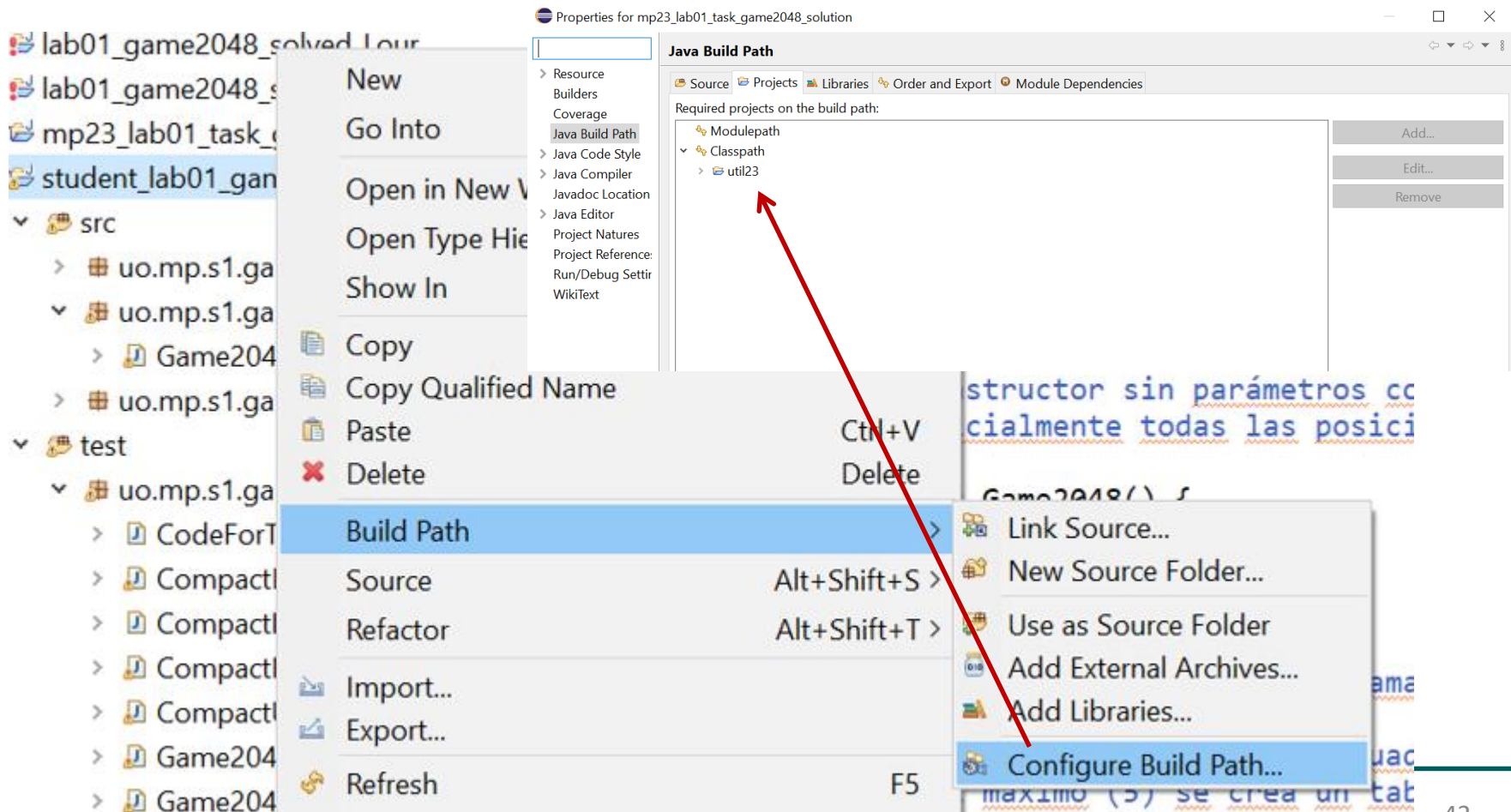
Analizar el proyecto

- Revisar versiones de Java y JUnit



Analizar el proyecto

- **Enlazar proyecto util23 con proyecto Game2048**

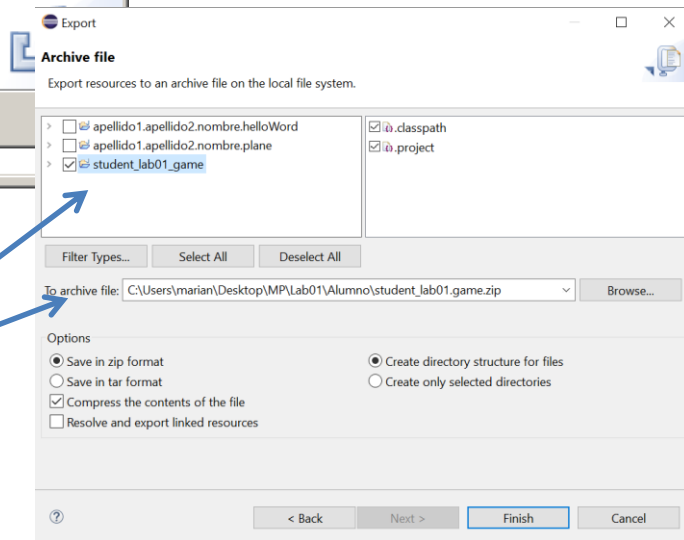
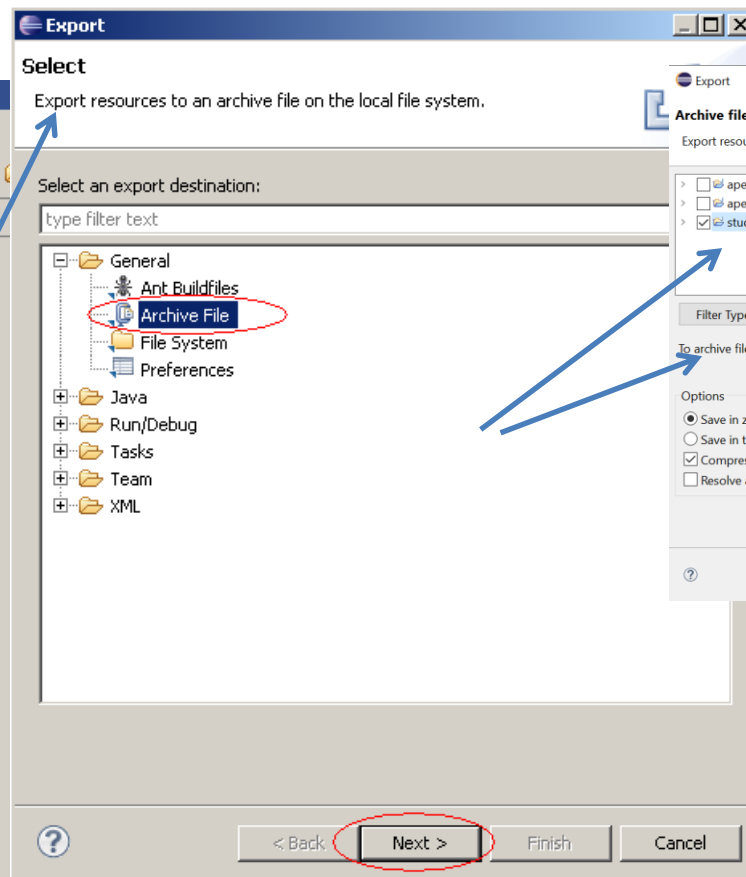
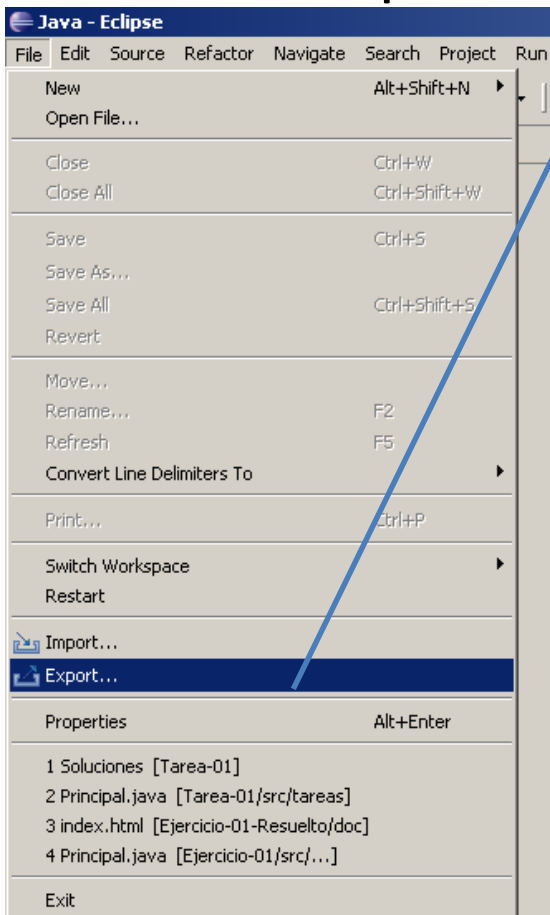




CÓMO SE EXPORTA UN PROYECTO Y SE CREA UN FICHERO COMPRIMIDO

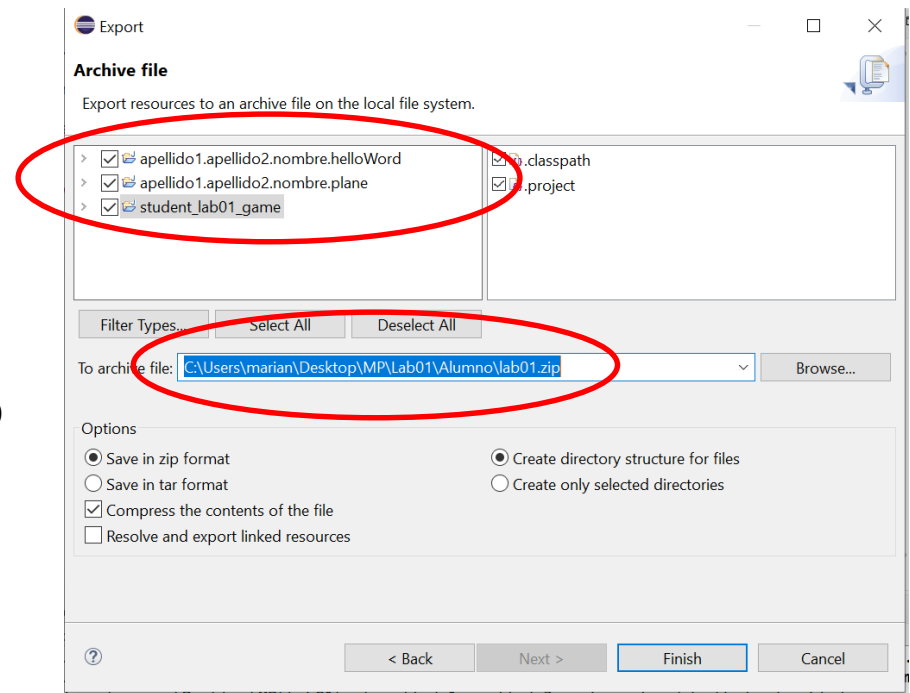
Exportar proyecto

- Permite copiar el proyecto a un archivo comprimido.
- File → Export



Exportar todos

- Cuando se finalice la sesión
 - Se exportan TODOS los proyectos juntos en fichero comprimido lab01
 - **Se guarda el comprimido en la nube**
- En casa
 - Se crea carpeta MP
 - Se crea carpeta lab01 en MP
 - Se crea carpeta ws en lab01
 - Se arranca eclipse y se le asigna ws
 - Se importa el fichero comprimido zip



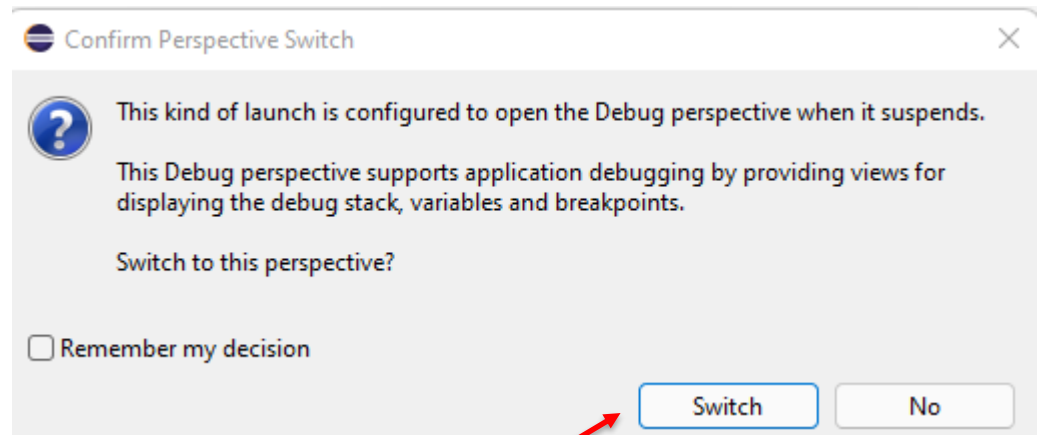
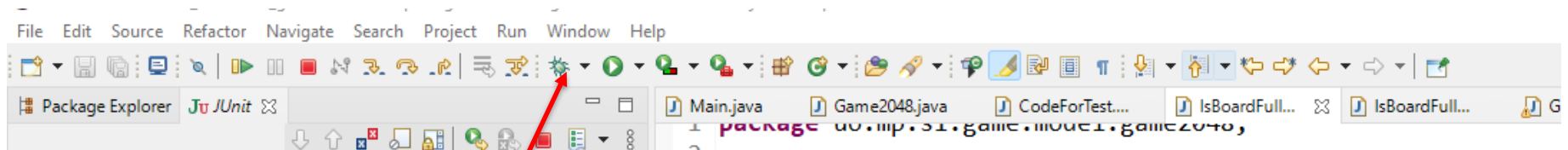
Depuración

Depuración

Punto de ruptura en el código. Igual que en BlueJ

Ejecución en modo depuración. Salta a la perspectiva de depuración

Ejecución paso a paso

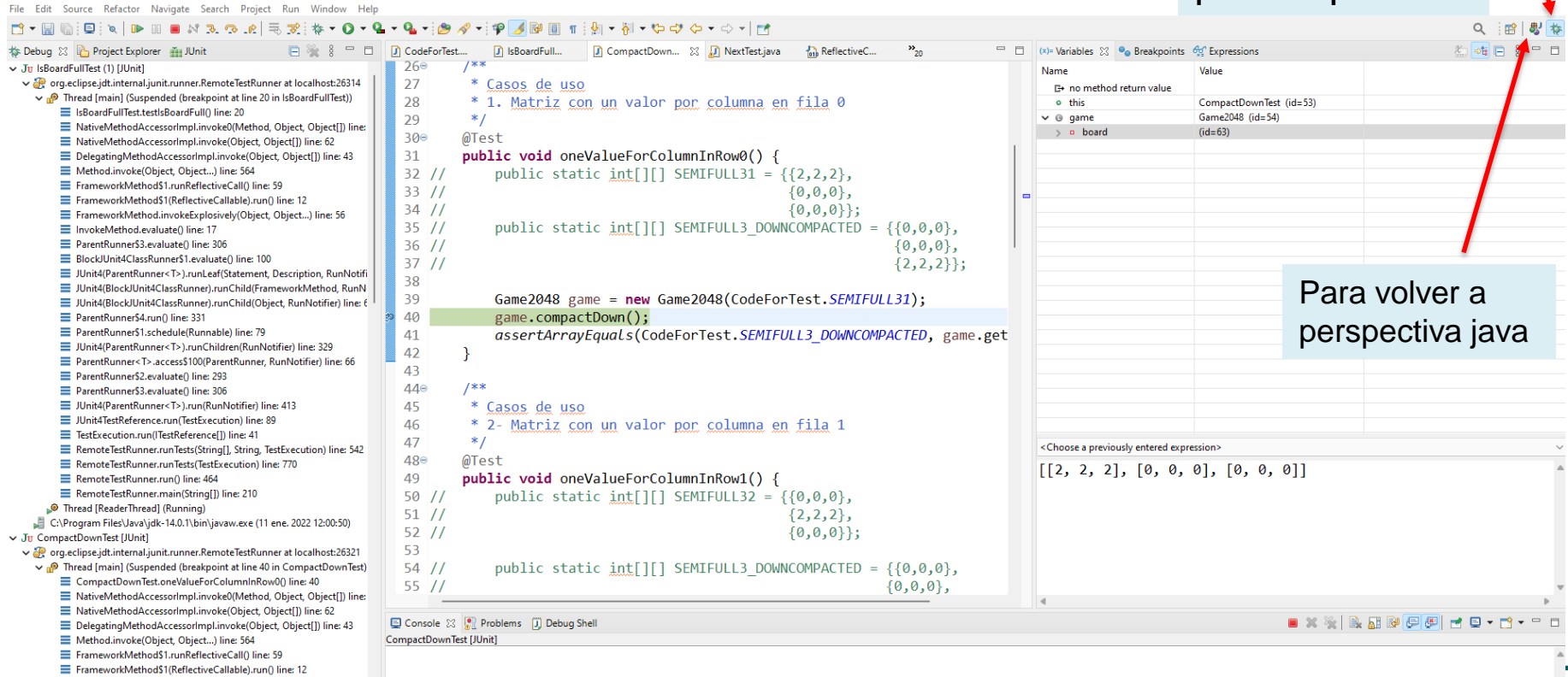


Depuración



Perspectiva depuración

Para ejecutar
paso a paso



Para volver a
perspectiva java

Tarea a entregar

- **Proyecto game2048 completado** con las operaciones que se piden en el enunciado y con el siguiente nombre
apellido1_apellido2_nombre_lab01_task_game2048

Para renombrar (un proyecto, paquete, Sobre él pulsar botón derecho y

Refactor/rename

- **Antes de entregar, revisa tu tarea** de forma autónoma **usando la checkList proporcionada** para asegurarte de que está bien.
- **Entregar en el campus virtual**, en el enlace habilitado para ello, **a lo sumo 24 horas antes de la siguiente tarea.**

Normas generales

- Cada semana hay tarea publicada en el campus
- Las tareas deben estar acabadas 24 horas antes de la siguiente clase de laboratorio.
- **Podrán ser usadas en la siguiente clase.**
- **Podrán ser usadas en los exámenes de laboratorio.**
- **Todas las tareas de trabajo autónomo (no presencial) que se pidan, deberán subirse comprimidas en un único fichero al campus virtual.**
- Más de 2 tareas inválidas o no entregadas supone **la pérdida de la evaluación continua**