Instalación y uso de IDE's

Álvaro González Sotillo

18 de diciembre de 2025

Índice

1. IDE

- **I** ntegrated **D** evelopment **E** nvironment
- Una aplicación con conjunto de herramientas que facilitan la tarea al programador
  - Editor de texto
  - Compilador
  - Enlazador
  - Control de versiones
  - Gestión de tareas
  - Depurador
  - Integración con IA generativa
  - Empaquetado, generador de instalaciones

1.1. Editor de código

Proporciona una interfaz para escribir y editar el código fuente. Suele incluir características como resaltado de sintaxis, autocompletado, indentación automática y otras utilidades para facilitar la escritura del código.

1.2. Compilador/Intérprete

Permite compilar o interpretar el código fuente en un formato ejecutable o en bytecode, dependiendo del lenguaje de programación utilizado. Algunos IDEs también ofrecen la capacidad de ejecutar el código directamente desde el entorno.

1.3. Depurador

Es una herramienta que ayuda a identificar y corregir errores en el código. Permite establecer puntos de interrupción, examinar el estado de las variables, ejecutar el código paso a paso y realizar otras operaciones para analizar y solucionar problemas en el programa.

1.4. Gestión de proyectos/ficheros

Permite crear, organizar y administrar proyectos de desarrollo de *software*. Esto incluye la capacidad de crear estructuras de directorios, agregar o eliminar archivos, gestionar dependencias y realizar otras tareas relacionadas con la organización del proyecto.

1.5. Control de versiones

Algunos IDEs incluyen integración con sistemas de control de versiones como Git, que permiten realizar seguimiento de cambios en el código, realizar confirmaciones (commits), fusionar (merge) ramas y otras operaciones relacionadas con la gestión del código fuente.

1.6. Herramientas de construcción

Algunos IDEs proporcionan herramientas para automatizar el proceso de construcción del *software*, como la generación de archivos de configuración, la compilación, el empaquetado y otras tareas relacionadas con la construcción del proyecto.

1.7. Herramientas externas

Los IDEs suelen ofrecer integración con otras herramientas y servicios externos, como sistemas de gestión de bases de datos, terminal, servidores web, frameworks, bibliotecas, entre otros, para facilitar el desarrollo y la integración con otros componentes del sistema.

1.8. *Agentic code*

Integración con IA generativa, que permite realizar funciones que antes se realizaban manualmente o con plugins específicos

- Generación de código
- Modificación de código (*refactor*)

2. Plugins

- Complementos que se relacionan con otras herramientas para agregarle una nueva función.
- Esta aplicación adicional es ejecutada por la aplicación principal.
- Ejercicio: Probar Keymap Exporter

3. Proyecto de ejemplo

- Proyecto de ejemplo
- Descomprime el ZIP

3.1. Construcción sin IDE

- Carpeta `Tetris`
- Fichero `Makefile`
  - Es un sistema básico de construcción
  - Antiguo, pero muy estándar y muy utilizado

```
src=src/*.java
pkg=tetris.jar
main=PlayTetris


all:
  mkdir -p bin
  javac ${src} -d bin/

package: all
  ../package.sh ${pkg} ${main}

doc:
  javadoc ${src} -d doc/

play: all
  java -cp bin/ ${main}

clean:
  rm -rf bin
  rm -rf doc
  rm -f ${pkg}
```

3.2. Ejercicio

- Utiliza solo la línea de comandos
- Construye la aplicación con los comandos que se deducen del fichero `Makefile`
- Ejecuta la aplicación
- Extra: utiliza el comando `make`

4. Intellij

- IDE profesional
  - Originalmente para Java
  - Pero soporta otros lenguajes

4.1. IDEs basados en IntelliJ

IDE	Enfoque Principal	Características Clave
IntelliJ IDEA Ultimate	Desarrollo Java y web empresarial	- IDE más completo, Soporte total para desar
IntelliJ IDEA Community Edition	Desarrollo Java de código abierto	- Versión gratuita, Características básicas de J
PyCharm	Desarrollo en Python	- Entorno especializado para Python, Herrami
WebStorm	Desarrollo web y JavaScript	- Soporte avanzado para JavaScript Herramier
CLion	Desarrollo en C y C++	- Soporte multiplataforma- Herramientas de o
GoLand	Desarrollo en Go	- Soporte para microservicios
Rider	Desarrollo .NET	
Android Studio	Desarrollo de aplicaciones Android	- Herramienta oficial de Google, Soporte nativ

4.2. Por qué tantas versiones

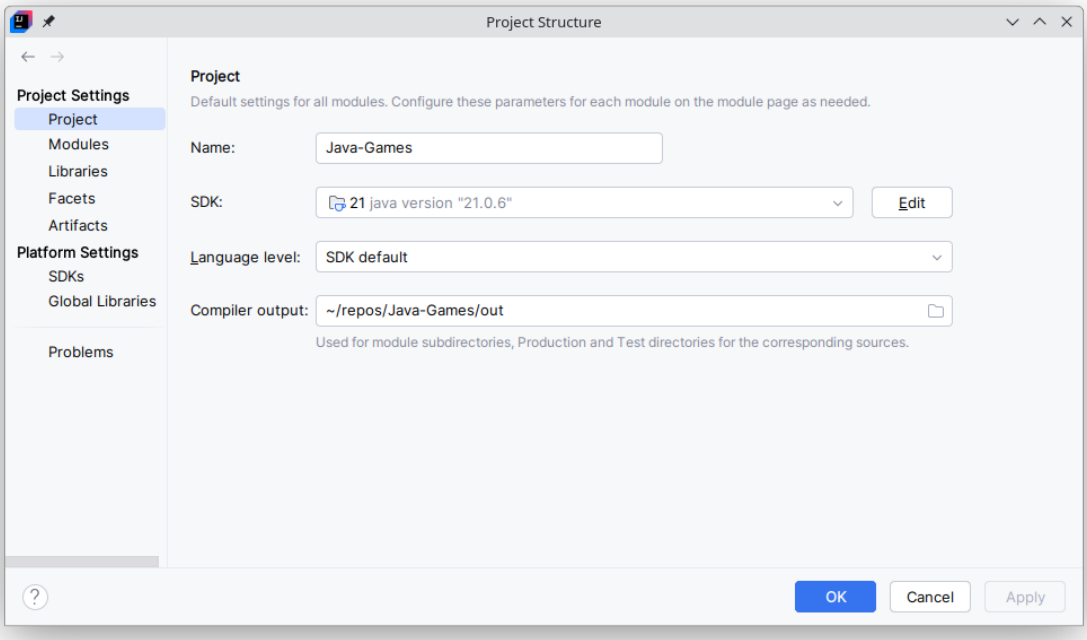
- La aplicación IntelliJ es prácticamente la misma
- Las diferentes versiones se diferencian en los *plugins* instalados
- No todos los plugins se pueden instalar en todas las versiones
  - La razón no es técnica, sino comercial

5. Instalación Intellij

- Como (casi) todos los IDE
  - Instalación CE (*comunitiy edition*) o en el NAS
  - **Preferido**: Ultimate edition (tras conseguir licencia gratuita de estudiante)
  - Descomprimir la instalación
  - Ejecutar `bin/idea.sh`
  - Opcional: crear fichero `.desktop` para lanzarlo cómodamente
- Algunos IDE complejos necesitan una instalación real
  - Pero se debe más al S.O. que al IDE

6. Compilación con Intellij

- Proyecto
- Módulo
- Librería
  - Fichero `jar`: clases precompiladas de Java
  - SDK (compilador del lenguaje). En Java, se llama JDK
- *Facet*: capacidades que se añaden a un proyecto
  - Ejemplo: añadir un lenguaje
- *Artifacts*: creación de ficheros entregables/desplegables/ejecutables



6.1. Proyecto

- Un proyecto contiene código fuente y otros recursos para construir una aplicación
- La aplicación se compone de módulos
  - Debe tener al menos un módulo

6.2. Módulo

- Varios ficheros fuente y recursos en una unidad compilable
- Un módulo se compila completo o no puede ejecutarse

6.3. Libería

- Código java ya compilado
  - Un fichero `jar` ya generado
  - Otro módulo del proyecto
- Se pueden añadir a cada módulo

6.4. Artifact/entregable

- Un fichero que es el resultado de la aplicación
  - Por ejemplo, un `jar` ejecutable

6.5. Ejercicio compilación

- Crea un módulo para varios juegos
  - Compila y juega al `tetris`
  - y al `mathhero`
  - Opcional: juega con otro compañero al `pongserver`

6.6. Ejercicio de liberías

- Crea un módulo nuevo de nombre `all`
- Mostrará un menú modo texto para elegir el juego a lanzar
- Lanzará el juego y al acabar el juego se acabará el programa

6.7. Ejercicio *artifact*

- Crea un `jar` ejecutable con el módulo `all` y todas sus dependencias
- Pruébalo con `java -jar all.jar`

7. Navegación

- Una de las facilidades más importantes del IDE
- El código se escribe una vez, pero se lee muchas veces
- Facilidades para:
  - Búsqueda de cadenas/símbolos en el proyecto
  - Ir rápidamente a una clase/fichero/método/variable
  - Buscar referencias/definiciones de clases/métodos/variables
  - Conocer la cadena de llamadas a un método
  - Saber qué métodos llaman/son llamados por otros métodos
  - Saber la cadena de herencia de una clase/interfaz
  - Saber qué clases y métodos se definen en el fichero actual

7.1. Ejercicio de búsqueda de atributo

- Busca la clase `Level` entre los juegos
- Busca quién lee y quién escribe en el atributo `Level.key`
- Decide para qué sirve `Level.key`
- Úsalo en el juego
- Opcional: crea nivel *superhardcore* con 10 enemigos simultáneos y clave `1111111`
- Opcional: *hackea el juego* para tener vida infinita

7.2. Ejercicio de búsqueda de cadena

- Traduce los mensajes del juego `MathHero` a castellano
  - *Level*
  - *Key*
  - *You loose*
- Opcional: añade un texto con la vida que aún queda

7.3. Ejercicio de herencia

- Busca la clase `MathHero`
- Encuentra de qué clases hereda dicha clase
- Encuentra todos los métodos, heredados o no, de esa clase

7.4. Ejercicio

Rellena la siguiente tabla con las teclas rápidas de cada entorno

	Idea	VSCode	*Otros*
Búsqueda de cadenas en proyecto			
Ir a fichero			
Ir a clase/método			
Buscar definición de símbolo			
Buscar usos de símbolo			
Buscar métodos llamados por un método			
Buscar métodos que llaman a un método			
Cadena de herencia de una clase			
Ir rápidamente a un método del fichero			
Volver al sitio anterior			
Otras			

8. Depuración (*debug*)

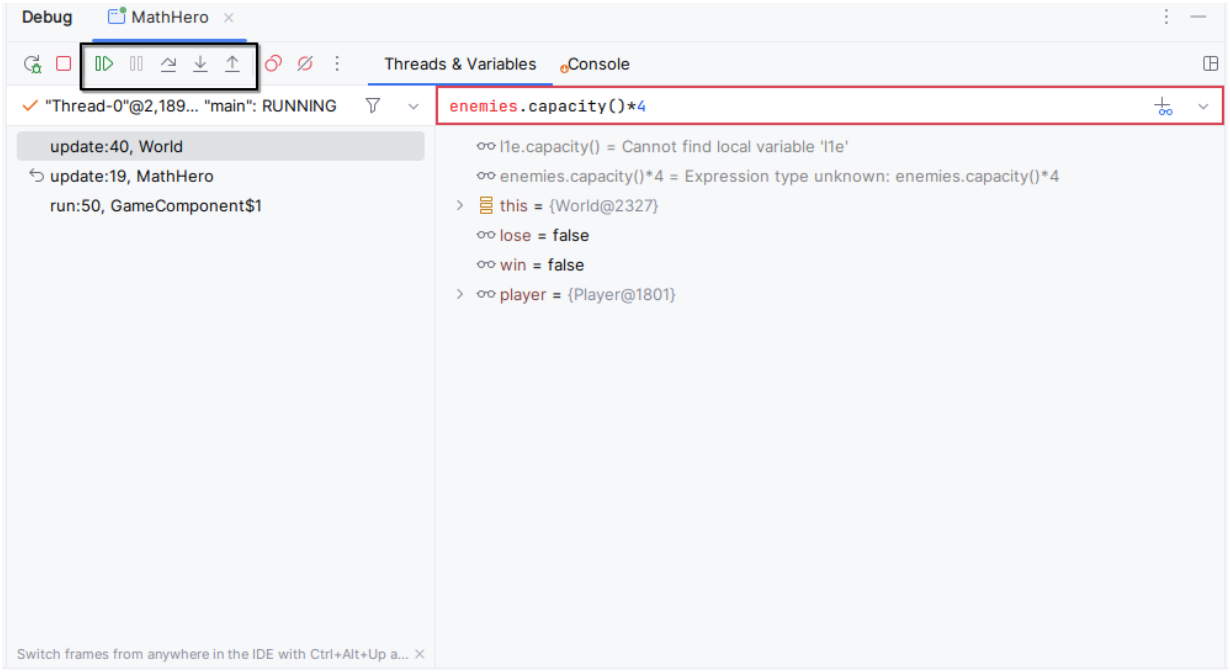
- Capacidad de avanzar paso a paso en un programa, visualizando variables
- Técnica complementaria a las trazas (`println` por el código)
- Conceptos
  - Punto de ruptura (*breakpoint*)
  - *breakpoint* condicional
  - Pila de llamadas (*call stack*)
  - Visualizar variale (*watch*, *inspect*)

8.1. *breakpoint*

- Parar la ejecución al llegar a cierta instrucción
- El *breakpoint* puede ser condicional
  - Dependiendo del valor de una expresión
  - Cuando se lanza una excepción
- Después, se puede seguir avanzando

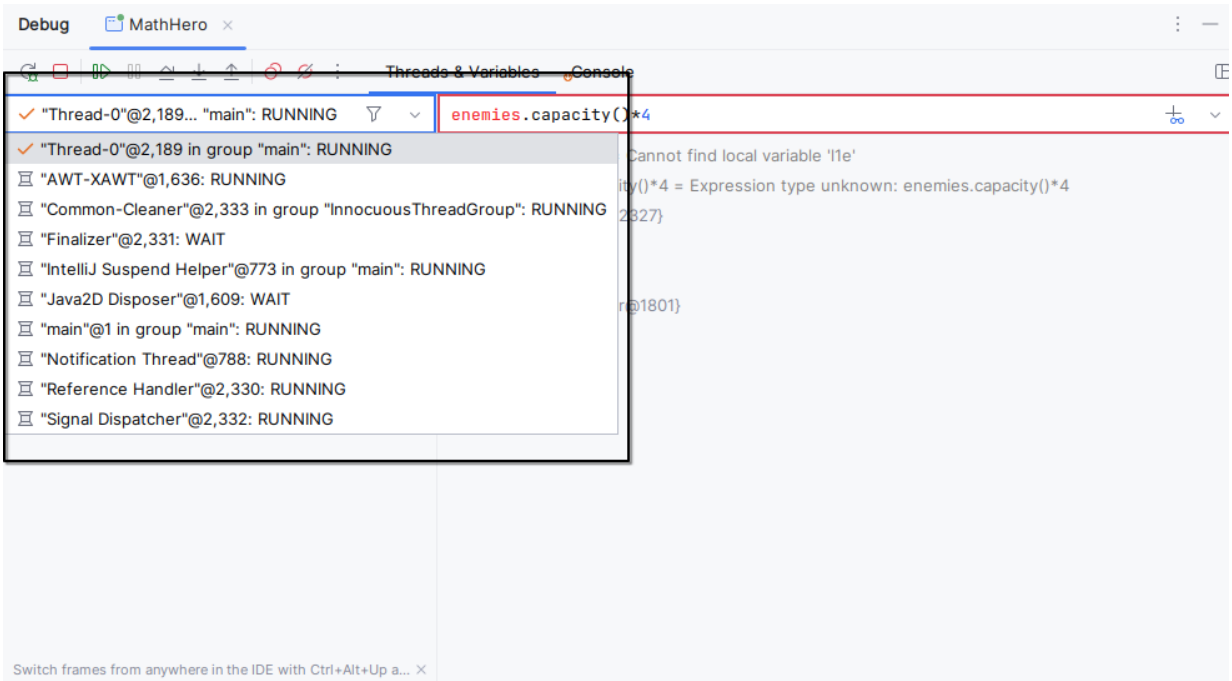
8.2. *step into*, *step out*, *step over*

- Tras parar en un breakpoint
  - *step into*: Se puede seguir ejecutando la siguiente instrucción, aunque sea en otro método
  - *step over*: Se ejecuta la siguiente instrucción del método actual
  - *step out*: Se ejecuta hasta salir del método actual (vuelve un nivel en el *stack trace*)



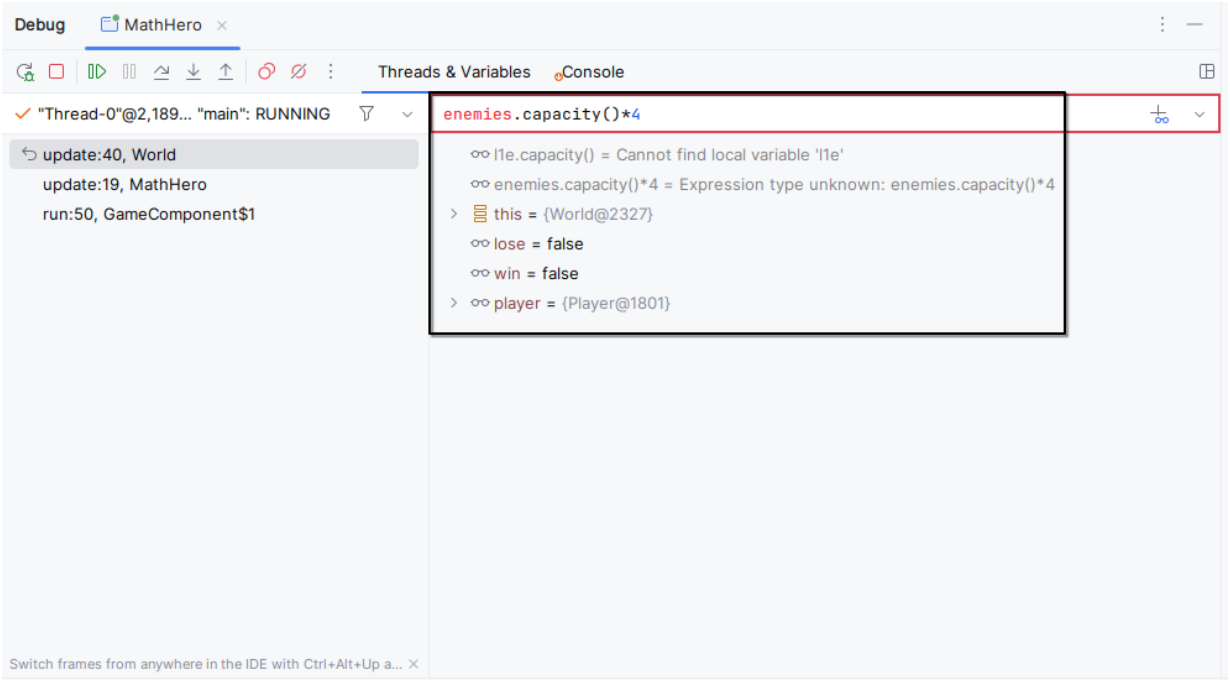
8.3. Pila de llamadas *stack trace*

- La ejecución comienza en un método `main`
- Cada método puede llamar a otros métodos
  - El método actual se ve "*encima*"
  - Los métodos que llamaron al método actual quedan "*debajo*"
- *Threads*
  - Hilos de ejecución, de forma concurrente
  - Cada uno tiene su propia pila de llamadas



8.4. *Watch*

- En un *breakpoint* se muestran por defecto todas las variables locales del método
- Pueden añadirse
  - Variables globales
  - Expresiones *java*: llamadas a métodos, aritmética...
- *inspect*: Las variables complejas (objetos) pueden expandirse para ver sus componentes



8.5. *hot reload* o *hot swap*

- Se puede cambiar el código de un método y aplicar los cambios al programa debugueado
- Hay límites:
  - No se pueden añadir métodos ni clases y atributos a clases
  - No se puede cambiar el tipo de los métodos (qué devuelven y qué reciben)
  - No se pueden cambiar variables estáticas
- Ampliación: jrebel

9. JDK vs JRE

- Java Runtime Environment
  - Herramientas para ejecutar programas java
- Java Development Kit
  - Herramientas para desarrollar programas en java
  - Incluye el compilador (y más herramientas) y un JRE

9.1. Versiones java

- Pueden tenerse múltiples versiones de JDK y JRE instaladas
- La mayoría de herramientas:
  - Usan `JAVA_HOME` para encontrar el JDK/JRE
  - Utilizan el programa `java` o `javac` que esté en el `PATH`
  - O bien, usan el camino completo al intérprete `java`

9.2. Variable PATH

- Existe en linux y windows
- Contiene los directorios donde se buscan los programas
- Se puede ver con:
  - Linux: `echo $PATH`
  - Windows: `echo %PATH%`

10. Referencias

- Formatos:
  - Transparencias
  - PDF
  - Página web
  - EPUB
- Creado con:
  - Emacs
  - org-re-reveal
  - Latex
- Alojado en Github