

# UD 2: Entornos Integrados de Desarrollo (NetBeans)

## Contornos de desenvolvimento (CODE)

**Víctor Blanco**

Curso 2023-2024



XUNTA  
DE GALICIA

IES Muralla Romana

Dpto. de Informatica

# Índice

**1** Introducción

**2** La Herramienta NetBeans

**3** Refactorización



IES MURALLA ROMANA

# Índice

## 1 Introducción

- Compilación sin IDEs
- Entornos Integrados de Desarrollo (IDEs)

## 2 La Herramienta NetBeans

## 3 Refactorización

# “Hola Mundo” Orientado a Objetos

## “Hola Mundo” orientado a objetos

```
class HolaMundoOO {  
    private String mensaje = "Hola Mundo";  
    public String devuelveMensaje() {  
        return mensaje;  
    }  
  
    public static void main(String[] args) {  
        HolaMundoOO miHola = new HolaMundoOO();  
        System.out.println(miHola.devuelveMensaje());  
    }  
}
```

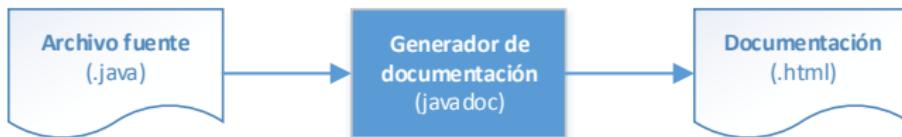


# Operaciones del entorno de desarrollo

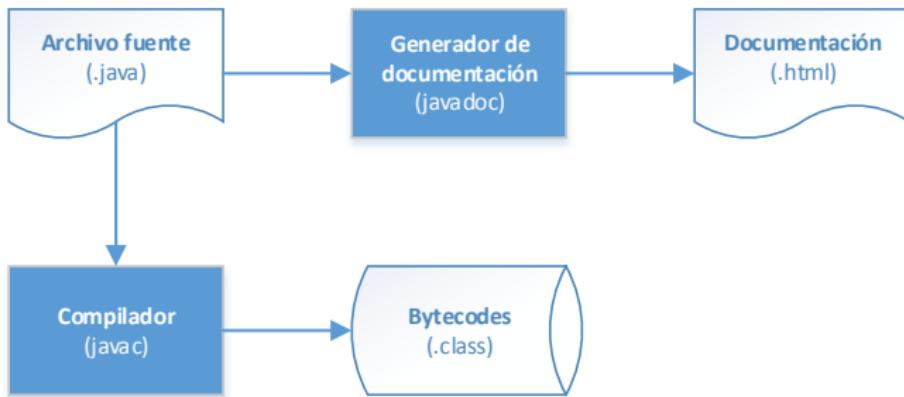
Archivo fuente  
(.java)



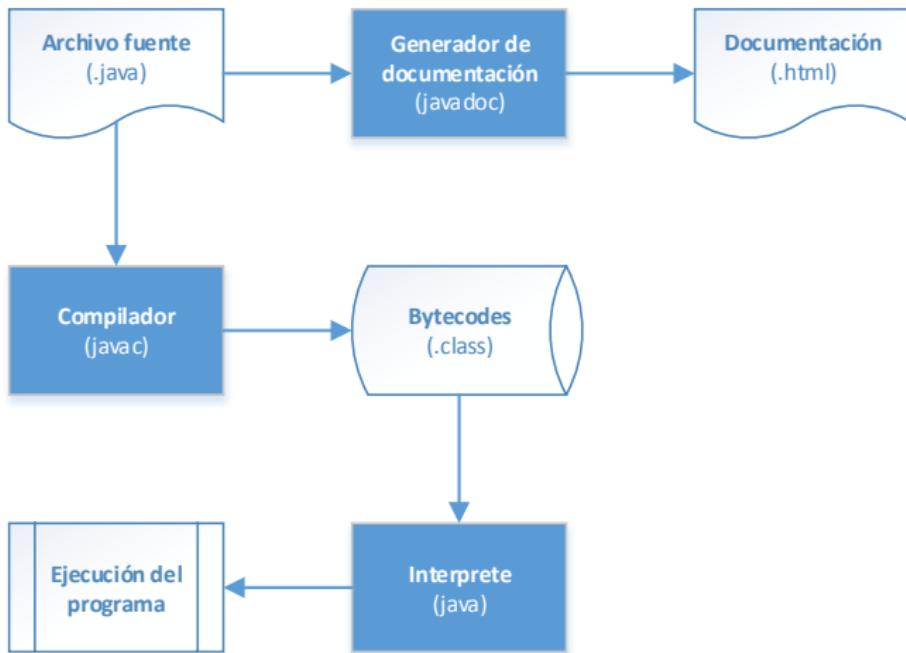
# Operaciones del entorno de desarrollo



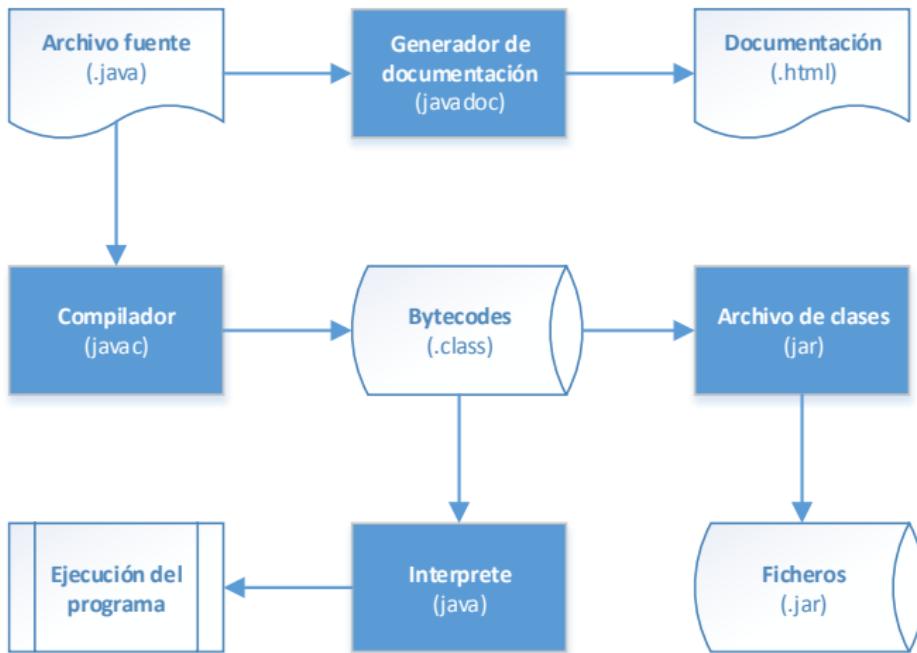
# Operaciones del entorno de desarrollo



# Operaciones del entorno de desarrollo



# Operaciones del entorno de desarrollo



# Compilación Simple

```
C:\java>dir
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: 1CD8-2C84

Directorio de C:\java

10/08/2005  02:33    <DIR>          .
10/08/2005  02:33    <DIR>          ..
10/08/2005  02:33           257 HolaMundo.java
               1 archivos          257 bytes
               2 dirs   10.514.345.984 bytes libres

C:\java>javac HolaMundo.java

C:\java>dir
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: 1CD8-2C84

Directorio de C:\java

10/08/2005  02:33    <DIR>          .
10/08/2005  02:33    <DIR>          ..
10/08/2005  02:33           329 HolaMundo.class
10/08/2005  02:33           257 HolaMundo.java
10/08/2005  02:33           408 HolaMundo00.class
               3 archivos          994 bytes
               2 dirs   10.514.341.888 bytes libres

C:\java>java HolaMundo
Hola Mundo

C:\java>
```

# Compilación Simple

```
C:\WINDOWS\system32\cmd.exe

C:\java>dir
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: 1CD8-2C84

Directorio de C:\java
10/08/2005 02:33    <DIR>      .
10/08/2005 02:33    <DIR>      ..
10/08/2005 02:33           257 HolaMundo.java
               1 archivos        257 bytes
               2 dirs   10.514.345.984 bytes libres

C:\java>javac HolaMundo.java

C:\java>dir
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: 1CD8-2C84

Directorio de C:\java
10/08/2005 02:33    <DIR>      .
10/08/2005 02:33    <DIR>      ..
10/08/2005 02:33           329 HolaMundo.class
10/08/2005 02:33           257 HolaMundo.java
10/08/2005 02:33           408 HolaMundo00.class
               3 archivos        994 bytes
               2 dirs   10.514.341.888 bytes libres

C:\java>java HolaMundo
Hola Mundo

C:\java>
```

Compilamos el fichero  
HolaMundo.java  
usando javac

# Compilación Simple

```
C:\WINDOWS\system32\cmd.exe

C:\java>dir
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: 1CD8-2C84

Directorio de C:\java
10/08/2005  02:33    <DIR>      .
10/08/2005  02:33    <DIR>      ..
10/08/2005  02:33           257 HolaMundo.java
              1 archivos        257 bytes
              2 dirs   10.514.345.984 bytes libres

C:\java>javac HolaMundo.java

C:\java>dir
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: 1CD8-2C84

Directorio de C:\java
10/08/2005  02:33    <DIR>      .
10/08/2005  02:33    <DIR>      ..
10/08/2005  02:33           329 HolaMundo.class
10/08/2005  02:33           257 HolaMundo.java
10/08/2005  02:33           408 HolaMundo00.class
              3 archivos        994 bytes
              2 dirs   10.514.341.888 bytes libres

C:\java>java HolaMundo
Hola Mundo

C:\java>
```

Compilamos el fichero  
HolaMundo.java  
usando javac

Obtenemos tantos  
ficheros class como  
clases hay en el fichero

# Compilación Simple

```
C:\WINDOWS\system32\cmd.exe

C:\java>dir
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: 1CD8-2C84

Directorio de C:\java
10/08/2005  02:33    <DIR>      .
10/08/2005  02:33    <DIR>      ..
10/08/2005  02:33           257 HolaMundo.java
              1 archivos        257 bytes
              2 dirs   10.514.345.984 bytes libres

C:\java>javac HolaMundo.java

C:\java>dir
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: 1CD8-2C84

Directorio de C:\java
10/08/2005  02:33    <DIR>      .
10/08/2005  02:33    <DIR>      ..
10/08/2005  02:33           329 HolaMundo.class
10/08/2005  02:33           257 HolaMundo.java
10/08/2005  02:33           408 HolaMundo00.class
              3 archivos        994 bytes
              2 dirs   10.514.341.888 bytes libres

C:\java>java HolaMundo
Hola Mundo
C:\java>
```

Compilamos el fichero  
HolaMundo.java  
usando javac

Obtenemos tantos  
ficheros class como  
clases hay en el fichero

Usamos el comando java para ejecutar el main de una clase particular (no se añade la extensión class)

# Compilación Simple

■ **El caso anterior es tan sencillo como poco realista para aplicaciones reales porque:**

- Mezcla los ficheros .java con los ficheros .class, algo generalmente poco recomendable
- No trabaja con paquetes (módulos) de Java ⇒ Los paquetes lógicos de Java se asocian con directorios físicos en el disco (Al no existir paquetes todos los fuentes necesarios residen en el mismo directorio)
- No se utilizan librerías externas aparte del API de Java



# Compilación Compleja

## ■ **Imaginemos un nuevo ejemplo más real en el que:**

- Los fuentes se sitúan en el directorio `src` y los compilados en el directorio `build`
- La clase `HolaMundo` si sitúa en el paquete `poo.holamundo` lo que implica que los fuentes tienen que estar en el subdirectorio `poo/holamundo`
- Utilizamos una clase `Libreria` del paquete `utilidades` con un método `imprime` que dado un `String` lo imprime por pantalla
- La librería se empaqueta en un fichero `jar` que se sitúa en el directorio `lib`

# Compilación Compleja

## Clase HolaMundoOO

```
package poo.holamundo;
import utilidades.Libreria;
public class HolaMundoOO {
    private String mensaje = "Hola Mundo";
    public String devuelveMensaje() { return mensaje; }

    public static void main(String[] args) {
        HolaMundoOO miHola = new HolaMundoOO();
        Libreria.imprime(miHola.devuelveMensaje());
    }
}
```

## Clase Libreria

```
package utilidades;
public class Libreria {
    public static void imprime(String s) { System.out.println (s); }
}
```



# Compilación Compleja

```
C:\java>tree
Listado de rutas de carpetas
El número de serie del volumen es 000062DC 1CD8:2C84
C:.
└── build
└── lib
└── src
    └── poo
        └── holamundo

C:\java>javac -d build -classpath lib/Libreria.jar src/poo/holamundo/HolaMundo.j
ava

C:\java>tree
Listado de rutas de carpetas
El número de serie del volumen es 0000D01B 1CD8:2C84
C:.
└── build
    └── poo
        └── holamundo
└── lib
└── src
    └── poo
        └── holamundo

C:\java>java -classpath build;lib/Libreria.jar poo/holamundo/HolaMundo
Hola Mundo

C:\java>
```

# Compilación Compleja

Nueva estructura  
de directorios se-  
parando fuentes,  
compilados y librerías

```
C:\java>tree
Listado de rutas de carpetas
El número de serie del volumen es 0000D01B 1CD8:2C84
C:.
└── build
└── lib
└── src
    └── poo
        └── holamundo

C:\java>javac -d build -classpath lib/Libreria.jar src/poo/holamundo/HolaMundo.j
ava

C:\java>tree
Listado de rutas de carpetas
El número de serie del volumen es 0000D01B 1CD8:2C84
C:.
└── build
    └── poo
        └── holamundo
└── lib
└── src
    └── poo
        └── holamundo

C:\java>java -classpath build;lib/Libreria.jar poo/holamundo/HolaMundo
Hola Mundo

C:\java>
```

# Compilación Compleja

The screenshot shows a Windows command prompt window titled 'C:\Windows\system32' with the following content:

```
C:\java>tree
Listado de rutas de carpetas
El número de serie del volumen es 0000D01B 1CD8:2C84
C:.
├── build
├── lib
└── src
    └── poo
        └── holamundo

C:\java>javac -d build -classpath lib/Libreria.jar src/poo/holamundo/HolaMundo.java

C:\java>tree
Listado de rutas de carpetas
El número de serie del volumen es 0000D01B 1CD8:2C84
C:.
├── build
│   └── poo
│       └── holamundo
└── lib
└── src
    └── poo
        └── holamundo

C:\java>java -classpath build;lib/Libreria.jar poo/holamundo/HolaMundo
Hola Mundo

C:\java>
```

A yellow speech bubble on the left points to the directory structure and says: "Nueva estructura de directorios separando fuentes, compilados y librerías". A yellow speech bubble on the right points to the command line and says: "Compilación con javac especificando el CLASSPATH".

# Compilación Compleja

```
C:\java>tree
Listado de rutas de carpetas
El número de serie del volumen es 000
C:.
├── build
├── lib
└── src
    └── poo
        └── holamundo

C:\java>javac -d build -classpath lib/Libreria.jar src/poo/holamundo/HolaMundo.java

C:\java>tree
Listado de rutas de carpetas
El número de serie del volumen es 000
C:.
├── build
│   └── poo
│       └── holamundo
└── lib
└── src
    └── poo
        └── holamundo

C:\java>java -classpath build;lib/Libreria.jar poo/holamundo/HolaMundo
Hola Mundo
C:\java>
```

Nueva estructura de directorios separando fuentes, compilados y librerías

Compilación con javac especificando el CLASSPATH

Los compilados replican la estructura de directorios de los fuentes

# Compilación Compleja

```
C:\> C:\Windows>
C:\java>tree
Listado de rutas de carpetas
El número de serie del volumen es 000
C:.
├── build
├── lib
└── src
    └── poo
        └── holamundo

C:\java>javac -d build -classpath lib/Libreria.jar src/poo/holamundo/HolaMundo.java

C:\java>tree
Listado de rutas de carpetas
El número de serie del volumen es 000
C:.
├── build
│   └── poo
│       └── holamundo
└── lib
└── src
    └── poo
        └── holamundo

C:\java>java -classpath build;lib/Libreria.jar poo/holamundo/HolaMundo
Hola Mundo
C:\java>
```

Nueva estructura de directorios separando fuentes, compilados y librerías

Compilación con javac especificando el CLASSPATH

Los compilados replican la estructura de directorios de los fuentes

La ejecución requiere de un CLASSPATH distinto que incluya los compilados

# Soluciones

## ■ Solución 1: archivos *.bat* o *scripts Linux*:

- Solución sencilla pero poco portable (los archivos no tienen el mismo formato)
- Incomoda e ineficaz para proyectos grandes

## ■ Solución 2: ficheros *make*:

- Usadas tradicionalmente la solución utilizada por C/C++ para compilar y ejecutar programas
- Es más portable pero también presenta problemas a la hora de llevar un fichero *make* a distintas plataformas
- No tiene en cuenta las particularidades de Java (`CLASSPATH`)

# Soluciones

## ■ Solución 3: Herramientas de *build* (p.ej. Ant):

- Herramientas que facilitan las tareas de desarrollo de software
- Ant (Another Neat Tool) de Apache está adaptada a Java y fácil de usar (desarrollada en Java ⇒ multiplataforma) y extender (las tareas Ant son clases Java)
- Los ficheros de configuración están en XML, un formato fácil de leer y editar

## ■ Solución 4: Entornos integrados de desarrollo:

- También facilitan las tareas de desarrollo de software
- Normalmente trabajan sobre los sistemas de *build*
- Incorporan entornos visuales, asistentes y demás facilidades para el desarrollo de código

# Entornos integrados de desarrollo

## Entorno integrado de desarrollo

Aplicación informática que proporciona servicios integrales para facilitar al programador el desarrollo de software



# Entornos integrados de desarrollo

## Entorno integrado de desarrollo

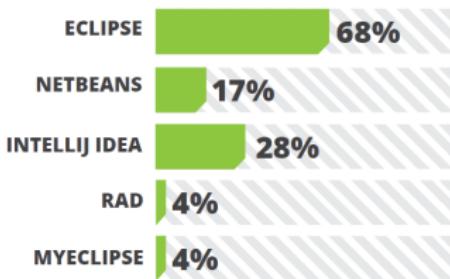
Aplicación informática que proporciona servicios integrales para facilitar al programador el desarrollo de software

- Se conocen popularmente con las siglas *IDEs* del inglés *Integrated Development Environments*
- Proporcionan servicios o ayudas para las tareas de: edición, compilación, depuración, empaquetado, desarrollo de interfaces gráficos, refactorización, etc.

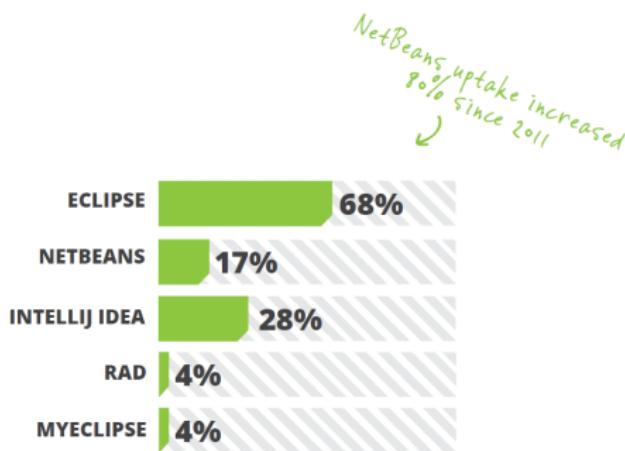


# IDEs Java en 2012

*NetBeans uptake  
8% since increased  
since 2011*

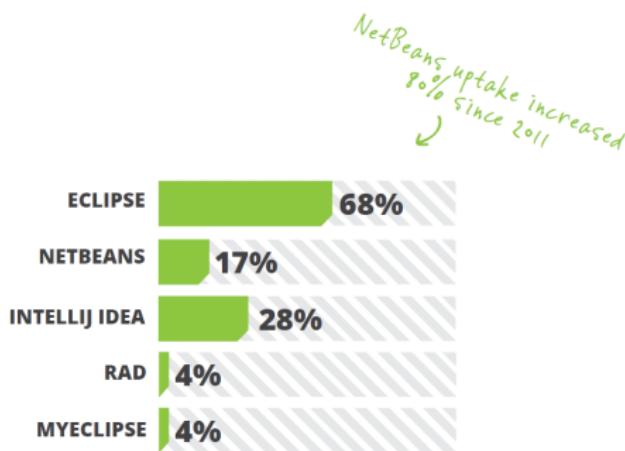


# IDEs Java en 2012



- El mercado se divide en tres IDEs: Eclipse, IntelliJ IDEA, NetBeans
- Eclipse domina el mercado pero los otros son cada vez más usados
- ¡Ojo! los porcentajes suman el 121 %

# IDEs Java en 2012

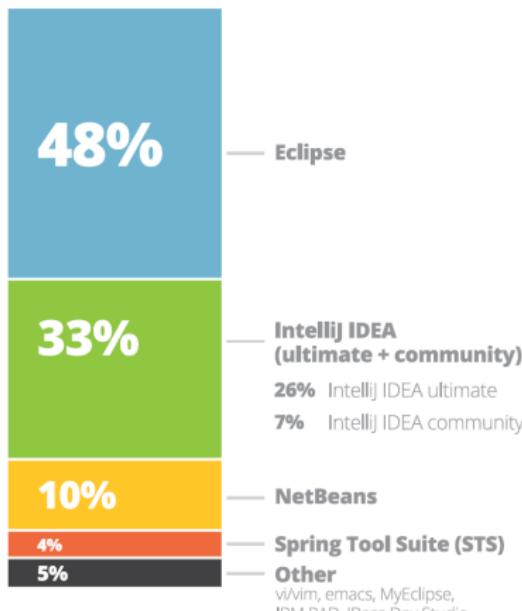


- El mercado se divide en tres IDEs: Eclipse, IntelliJ IDEA, NetBeans
- Eclipse domina el mercado pero los otros son cada vez más usados
- ¡Ojo! los porcentajes suman el 121 %

Las estadísticas mostradas en este tema y los siguientes provienen de informes obtenidos de ZeroTurnaround RebelLabs: <https://zeroturnaround.com/rebellabs/>

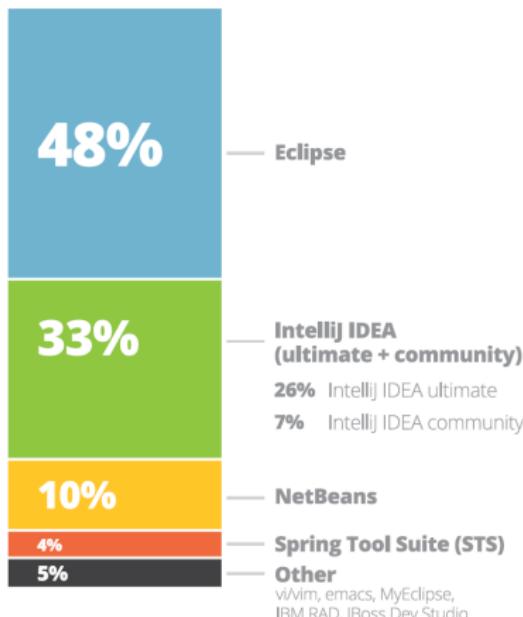
# IDEs Java en 2014

**IDE used** most often



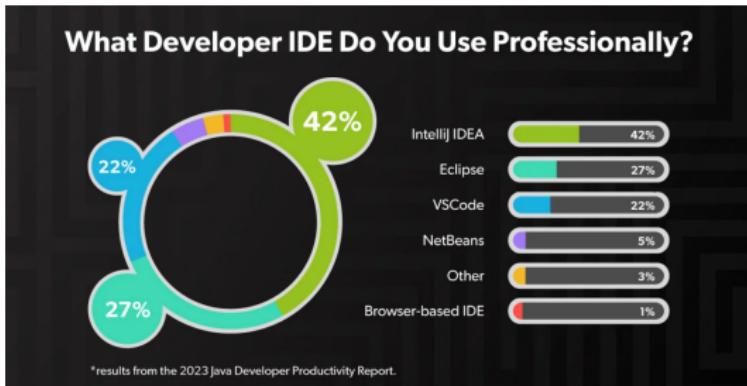
# IDEs Java en 2014

**IDE used** most often

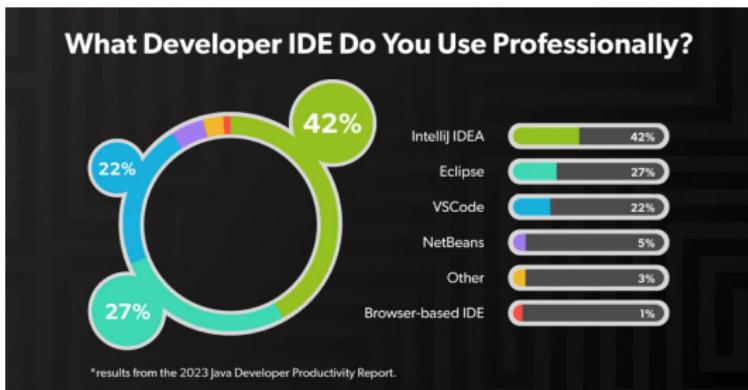


- Eclipse ha perdido cuota de mercado en favor de IntelliJ IDEA
- NetBeans continua estando en tercera posición
- Los números no son demasiado representativos (ahora sí suman el 100 %)
- Sigue existiendo gente que no usa ningún IDE

# IDEs Java en 2024

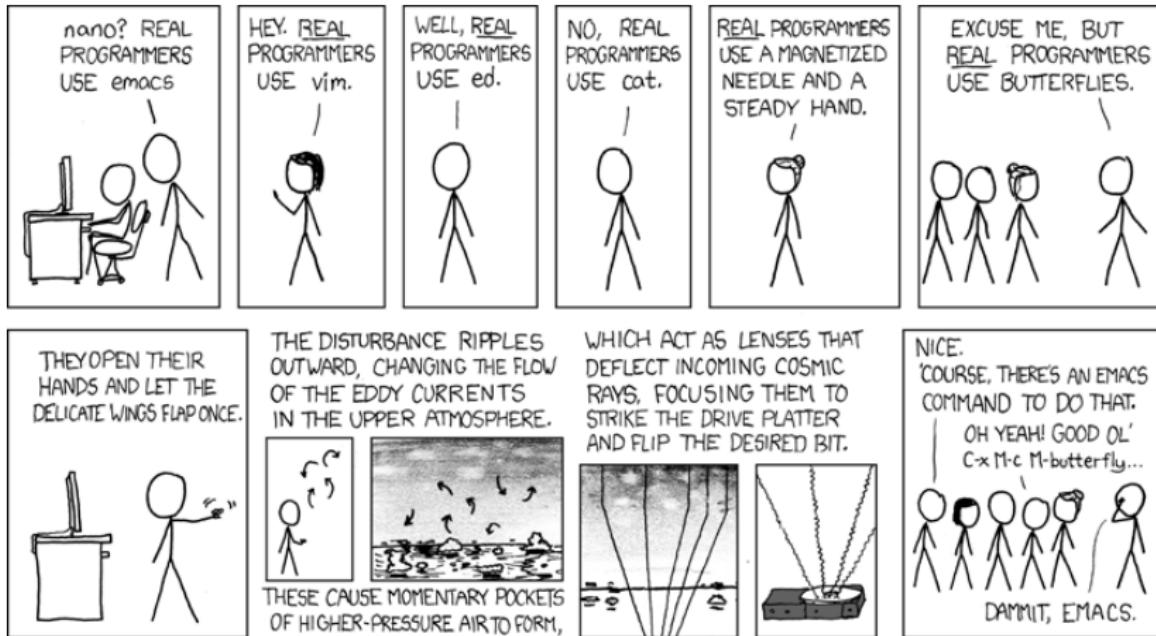


# IDEs Java en 2024



- El mercado se divide en 4 IDEs: Eclipse, IntelliJ IDEA, NetBeans y Visual Studio Code
- Visual Studio Code irrumpió con fuerza
- IntelliJ IDEA domina el mercado

# Real Programmers



<https://xkcd.com/378>



# Eclipse



- Desarrollado originalmente por IBM
- Mantenido por un consorcio que se transformó en fundación en 2004 y que incluye no solo a IBM sino a miembros como Oracle, SAP, Google, etc.
- Distribuido bajo la *Eclipse Software License* (EPL) considerada una licencia libre aunque incompatible con la *GNU General Public License* (GPL)
- Soporta numerosos lenguajes, generalmente a través del uso de *plugins*

# IntelliJ IDEA



- Desarrollado por la empresa JetBrains (antes denominada IntelliJ)
- Herramienta comercial basada en la venta de licencias perpetuas de uso (*bugfixes* gratuitos y descuentos para nuevas versiones)
- Desde el 2 de noviembre de 2015 su modelo será el de suscripción (acceso siempre a la última versión a cambio de pagos mensuales o anuales y otras restricciones -p.ej. conexión a Internet-)
- La versión *community* es distribuida mediante una licencia Apache (usada por Google para desarrollar Android Studio)
- La versión *Ultimate* (comercial) añade soporte para modelado, bases de datos, programación web, etc.
- También soporta múltiples lenguajes, algunos integrados y otros a través de *plugins*

# NetBeans



- Desarrollado inicialmente en la Universidad Carolina de Praga fue posteriormente adquirida por Sun Microsystems (propietaria de Java) que finalmente fue absorbida por Oracle
- Distribuida mediante un modelo dual a través de la *Common Development and Distribution License* (CDDL), basada en la *Mozilla Public License* (MPL) y la *GPL version 2 license* incluyendo la *GPL linking exception for GNU Classpath*.
- Soporta múltiples lenguajes y sigue un modelo similar a IntelliJ IDEA de ofrecer funcionalidades sin necesidad de *plugins* externos (funcionalidades *out-of-the-box*) aunque también soporta *plugins*.

# Visual Studio Code



## Visual Studio Code

- Desarrollado por Microsoft y anunciado en abril de 2015
- En noviembre de ese mismo año se publicó su código fuente en GitHub y se anunció el soporte para *plugins*
- En lugar de tener el clásico sistema de proyectos, permite al usuario abrir múltiples carpetas, que pueden guardadas en *workspaces* para su futura reutilización.
- Algunas de sus funcionalidades sólo son accesibles por línea de comandos

# Comparativa



- Eclipse es más que un IDE. Es una plataforma sobre la que se han construido más de 76 proyectos que van desde el desarrollo software, al modelado o a aspectos como el “Internet de las cosas” *Internet of Things (IoT)*
- El desarrollo de numerosos *plugins* para numerosas tecnologías es su gran fortaleza pero, en parte, su gran debilidad ya que no siempre funcionan adecuadamente en conjunto

# Comparativa



- Desarrollado por una única compañía en la que la influencia de la comunidad es menor
- Las múltiples características incluidas por defecto le dan un carácter más cohesivo (menos dependiente de *plugins*)
- Interfaz muy similar a la de Android Studio
- Es considerado por muchos desarrolladores el IDE más completo y usable
- Para usos profesionales es una herramienta de pago
- Se trata de un software más pesado que sus competidores

# Comparativa



- Situada en una posición residual, coge lo mejor de ambos mundos: Herramienta *open-source* cohesiva con múltiples características incluidas por defecto
- Muy utilizado en entornos educativos (universidades y centros de formación profesional)
- Pertenece a Oracle (propietarios también de Java), siendo el primero en soportar una nueva versión de Java una vez es publicada.
- Es de los pocos proyectos *open-source* que han mantenido y fomentado (al contrario que OpenOffice, Hudson, etc.) y han sabido mantener a su comunidad.

# Comparativa



Visual Studio Code

- Con un ascenso meteórico en cuota de mercado desde su publicación
- Igualmente, muy utilizado en entornos educativos (universidades y centros de formación profesional)
- Es una aplicación muy ligera
- Tiene demasiadas extensiones disponibles



IES MURALLA ROMANA

# Comparativa



## ■ Conclusión

- Cualquiera de los cuatro IDEs es adecuado para el desarrollo profesional de software
- La elección de uno u otro depende en gran medida de preferencias personales, aunque los costes también son un factor a tener en cuenta
- A nivel empresarial, es recomendable que todos los desarrolladores usen el mismo, para agilizar la resolución de posibles errores y la configuración de los proyectos.

# Índice

## 1 Introducción

## 2 La Herramienta NetBeans

- Características
- Instalación
- Proyectos NetBeans
- Ejemplo de Uso
- Accesos directos

## 3 Refactorización

# Soporte para las últimas tecnologías Java

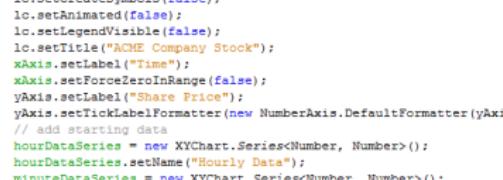
- Incluyendo lambdas, interfaces funcionales, métodos por defecto, etc.

```
44  public class Anagrams extends JFrame {  
45  
46      public static void main(String[] args) {  
47          /* Set the Nimbus look and feel */  
48          This anonymous inner class creation can be turned into a lambda expression.  
----  
(Alt-Enter shows hints)  
49          SwingUtilities.invokeLater(new Runnable() {  
50              public void run() {  
51                  new Anagrams().setVisible(true);  
52              }  
53          });  
54      }  
55  }  
  
46  public static void main(String[] args) {  
47      /* Set the Nimbus look and feel */  
48      Look and feel setting code (optional)  
49  
50      /* Create and display the form */  
51      SwingUtilities.invokeLater(() -> {  
52          new Anagrams().setVisible(true);  
53      });  
54  }
```



Edición de código rápida y sencilla

- Identado, autocompletado, resultado de errores sintácticos y semáticos, refactorización, generadores de código, plantillas, etc.
  - Soporta muchos lenguajes como Java, C/C++, XML, HTML, PHP, Groovy, Javadoc, JavaScript, JSP, etc..



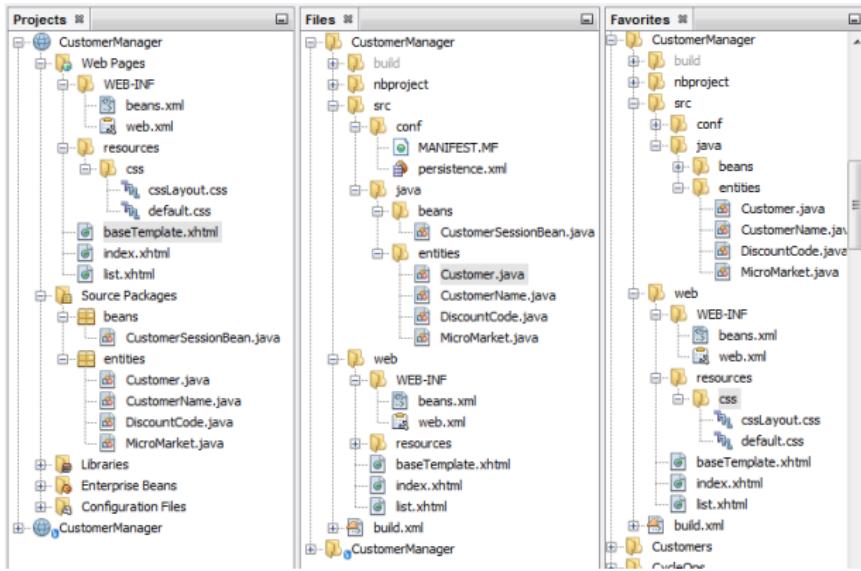
The screenshot shows the JavaFX Scene Builder interface with a Java code editor open. The code is for a chart component, specifically setting up axes and data series. A tooltip is displayed over the 'getData()' method call, providing information about its parameters and return type. The code editor has syntax highlighting and various toolbars at the top.

```
ChartAdvancedStockLine.java
Source History ...
81 lc.setAnimated(false);
82 lc.setLegendVisible(false);
83 lc.setTitle("ACME Company Stock");
84 xAxis.setLabel("Time");
85 xAxis.setForceZeroInRange(false);
86 yAxis.setLabel("Share Price");
87 yAxis.setTickLabelFormatter(new NumberAxis.DefaultFormatter(yAxis, "$"));
88 // add starting data
89 hourDataSeries = new XYChart.Series<Number, Number>();
90 hourDataSeries.setName("Hourly Data");
91 minuteDataSeries = new XYChart.Series<Number, Number>();
92 minuteDataSeries.setName("Minute Data");
93 // create some starting data
94 hourDataSeries.getData().add(new XYChart.Data<Number, Number>(timeInH ...
95 minuteDataSeries.getData().add(new XYChart.Data<Number, Number>(...
96 for (double nextTime : getClass() ...
97 nextTime = getClass() ...
98 plotTime = getName() ...
99 } ...
100 lc.getData().addAll(minuteDataSeries);
101 lc.getData().add(hourDataSeries);
102 return lc;
103 }
```



# Gestión de proyectos sencilla y eficiente

- Diferentes vistas de un proyecto (lógica, física)
- Gestión de proyectos basada en Ant



# Control de versiones integrado

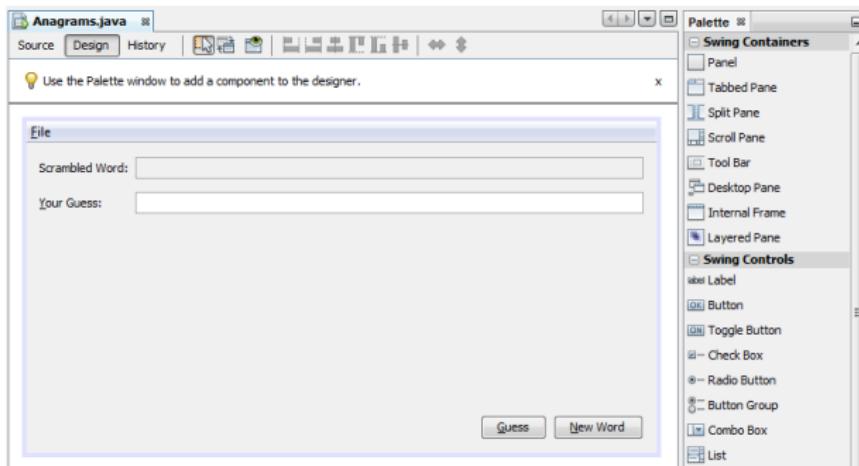
- Soporte integrado para las distintas herramientas de versionado (Subversion, Git, Mercurial, etc.)

The screenshot shows the NetBeans IDE interface for managing version control. At the top, there's a toolbar with various icons and a dropdown menu labeled "Search Subversion History". Below the toolbar is a table showing the commit history for a file named "collaboration.html". The columns in the table are Date, Revision, User, and Message. The history shows several commits from a user named "geertjan" on January 14, 2013, and January 11, 2013, with messages related to database features and editor page rewrites.

At the bottom, there's a "Graphical" tab selected, showing a diff view between the current file and the previous revision (revision 96). The diff highlights changes in the HTML code, such as the addition of a new  element with a class of "overview-c" and a source image of "Subversion". The code also includes some explanatory text about Subversion and its features.

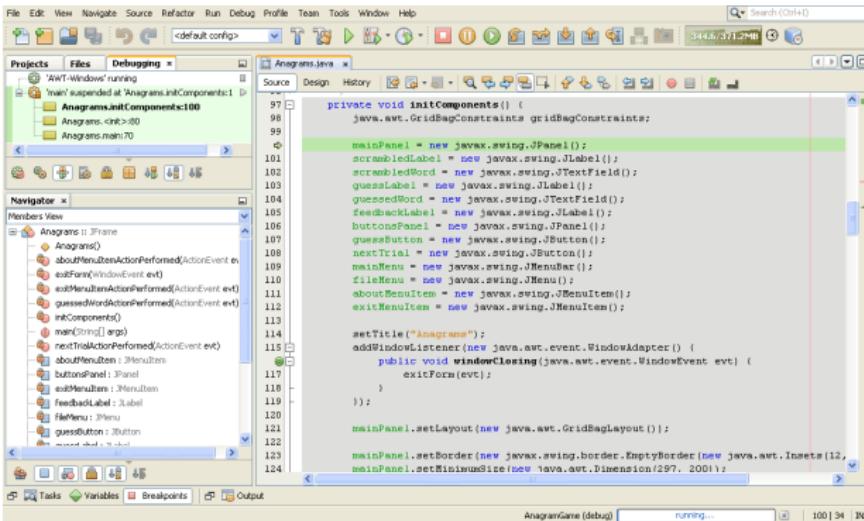
# Desarrollo rápido de interfaces de usuario

- Herramientas como Swing GUI Builder permiten crear fácilmente interfaces gráficos de usuario en Swing
- Soporte para la nueva librería JavaFX



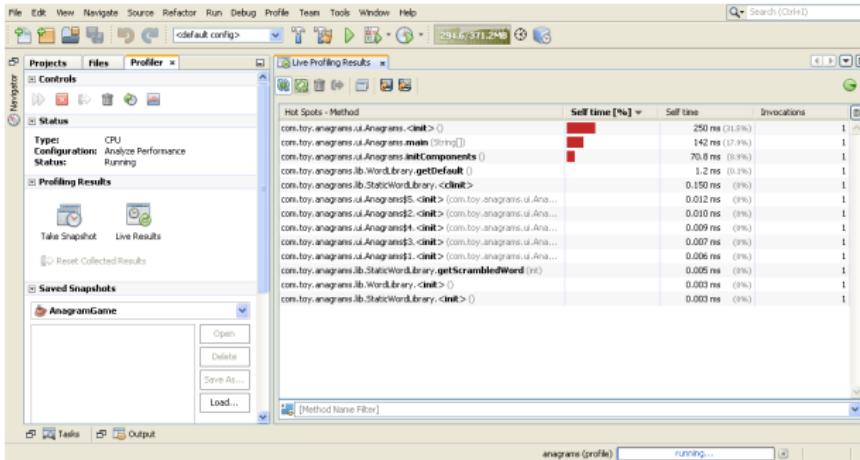
# Depurador

- Depurador avanzado con las típicas herramientas que nos permiten realizar una ejecución paso a paso y ver el contenido de variables, objetos, etc.



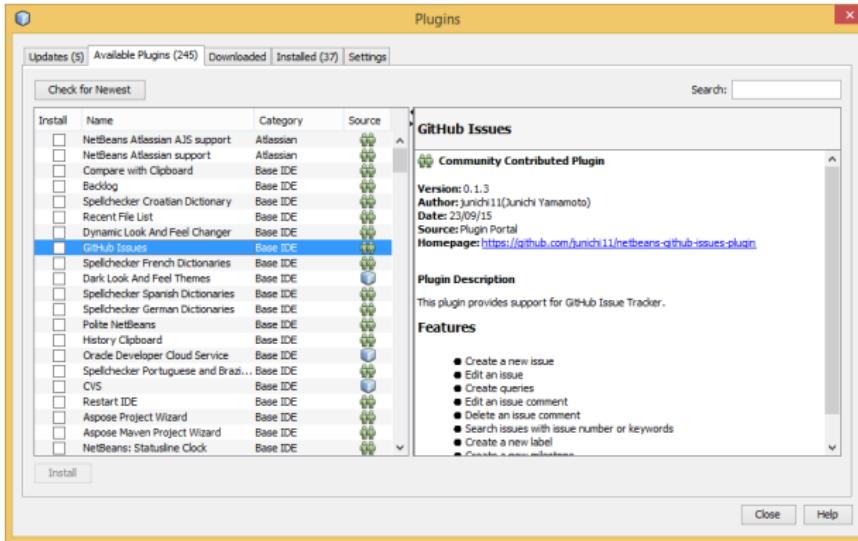
# Profiler

- Que provee de asistencia para optimizar la velocidad y el uso de memoria de nuestra aplicación



# Plugins

- Soporte para múltiples plugins desarrollados por NetBeans o por la comunidad de usuarios
- Ver lista completa en: <http://plugins.netbeans.org/>



# Plugins

- NetBeans puede descargarse en distintos paquetes (*bundles*) con distintas funcionalidades
- Posteriormente pueden añadirse nuevas funcionalidades si fuera necesario a través del gestor de *plugins*

NetBeans IDE Download Bundles					
Supported technologies *	Java SE	Java EE	C/C++	HTML5 & PHP	All
NetBeans Platform SDK	•	•			•
Java SE	•	•			•
Java FX	•	•			•
Java EE		•			•
Java ME					•
HTML5		•		•	•
Java Card™ 3 Connected					•
C/C++			•		•
Groovy					•
PHP				•	•
Bundled servers					
GlassFish Server Open Source Edition 4.1		•			•
Apache Tomcat 8.0.15	•				•
<a href="#">Download</a>		<a href="#">Download</a>	<a href="#">Download</a>	<a href="#">Download</a>	<a href="#">Download</a>
Free, 90 MB		Free, 186 MB	Free, 63 MB	Free, 63 MB	Free, 205 MB

# Compilación con la Herramienta “ant”

## ■ **NetBeans y Ant:**

- NetBeans utiliza Ant como gestor de proyectos

## ■ **Características de Ant:**

- Por defecto Ant busca un fichero de compilación denominado build.xml
- Cada fichero contiene una etiqueta <project> donde se especifican las características del proyecto
- Además tendrá un conjunto de etiquetas <target> que indican los objetivos que pueden realizarse con dicho fichero Ant (inicializar, compilar, etc.)
- Los target pueden tener dependencias entre sí, si un target A depende de otro B, al intentar ejecutar A se ejecutará primero B

# Compilación con la Herramienta “ant”

## Proyecto de ant. Parte 1: Configuración

```
<project name="Hola Mundo" default="compile" basedir=".">



    <!-- Propiedades globales -->
    <property name="src.dir" value="src"/>
    <property name="build.dir" value="build"/>
    <property name="lib.dir" value="lib"/>
    <property name="Libreria.jar" value="${lib.dir}/Libreria.jar"/>




    <!-- CLASSPATH para la compilacion -->
    <path id="compile.classpath">
        <pathelement location="${Libreria.jar}" />
    </path>




    <!-- CLASSPATH para la ejecucion -->
    <path id="run.classpath">
        <path refid="compile.classpath" />
        <pathelement location="${build.dir}" />
    </path>




    <!-- . . . -->
```



# Compilación con la Herramienta “ant”

## Proyecto de ant. Parte 2: Targets

```
<!-- ... -->
<!-- Inicia la compilacion creando el directorio build -->
<target name="init">
    <mkdir dir="${build.dir}" />
</target>
<!-- Limpia el directorio build de compilaciones pasadas -->
<target name="clean">
    <delete dir="${build.dir}" />
</target>
<!-- Compila el proyecto -->
<target name="compile" depends="init">
    <javac srcdir="${src.dir}" destdir="${build.dir}"
           classpathref="compile.classpath"/>
</target>
<!-- Ejecuta el proyecto -->
<target name="run" depends="compile">
    <java classname="poo.holamundo.HolaMundo"
          classpathref="run.classpath"/>
</target>
</project>
```



# Compilación con la Herramienta “ant”

```
C:\WINDOWS\system32\cmd.exe

C:\java>ant
Buildfile: build.xml

init:
  [mkdir] Created dir: C:\java\build

compile:
  [javac] Compiling 1 source file to C:\java\build

BUILD SUCCESSFUL
Total time: 3 seconds
C:\java>
C:\java>
C:\java>ant run
Buildfile: build.xml

init:
compile:
run:
  [java] Hola Mundo

BUILD SUCCESSFUL
Total time: 1 second
C:\java>
```



# Compilación con la Herramienta “ant”

```
C:\WINDOWS\system32\cmd.exe
C:\java>ant
Buildfile: build.xml

init:
  [mkdir] Created dir: C:\java\build

compile:
  [javac] Compiling 1 source file to C:\java\build

BUILD SUCCESSFUL
Total time: 3 seconds
C:\java>
C:\java>
C:\java>ant run
Buildfile: build.xml

init:
compile:
run:
  [java] Hola Mundo

BUILD SUCCESSFUL
Total time: 1 second
C:\java>
```

Si no se especifica un target se ejecuta el por defecto, en este caso compile

# Compilación con la Herramienta “ant”

```
C:\WINDOWS\system32\cmd.exe
C:\java>ant
Buildfile: build.xml

init:
  [mkdir] Created dir: C:\java\build

compile:
  [javac] Compiling 1 source file to C:\java\build

BUILD SUCCESSFUL
Total time: 3 seconds
C:\java>
C:\java>
C:\java>ant run
Buildfile: build.xml

init:
compile:
run:
  [java] Hola Mundo

BUILD SUCCESSFUL
Total time: 1 second
C:\java>
```

Si no se especifica un target se ejecuta el por defecto, en este caso compile

compile depende de init, por lo que es necesario ejecutar este target antes

# Compilación con la Herramienta “ant”

```
C:\WINDOWS\system32\cmd.exe
C:\java>ant
Buildfile: build.xml

init:
  [mkdir] Created dir: C:\java\build

compile:
  [javac] Compiling 1 source file to C:\java\build

BUILD SUCCESSFUL
Total time: 3 seconds
C:\java>
C:\java>
C:\java>ant run
Buildfile: build.xml

init:
compile:
run:
  [java] Hola Mundo

BUILD SUCCESSFUL
Total time: 1 second
C:\java>
```

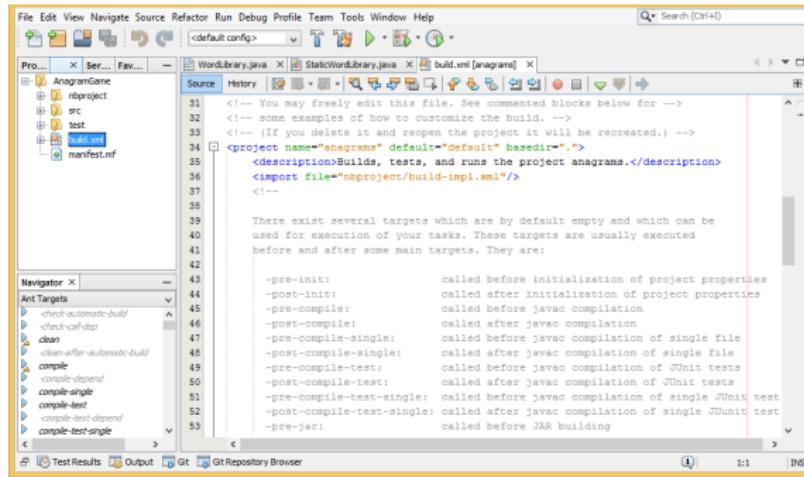
Si no se especifica un target se ejecuta el por defecto, en este caso compile

compile depende de init, por lo que es necesario ejecutar este target antes

Al ejecutar el target run debemos también ejecutar compile e init, pero como ya han sido ejecutados anteriormente no es necesario hacer nada nuevo

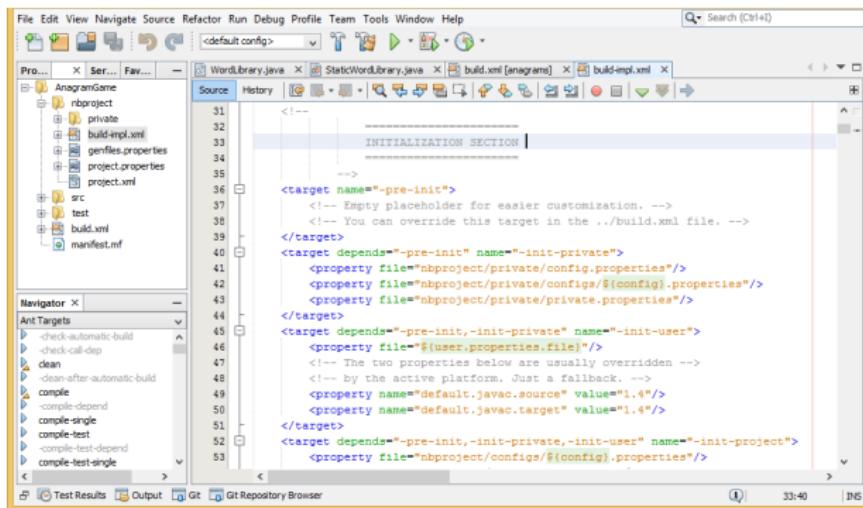
# Ant y NetBeans

- El fichero build.xml de NetBeans representa la versión Ant de nuestro proyecto, aunque en realidad lo único que hace es invocar a build-impl.xml
- El fichero build.xml está destinado a introducir aportaciones propias sin tener que modificar el fichero build-impl.xml

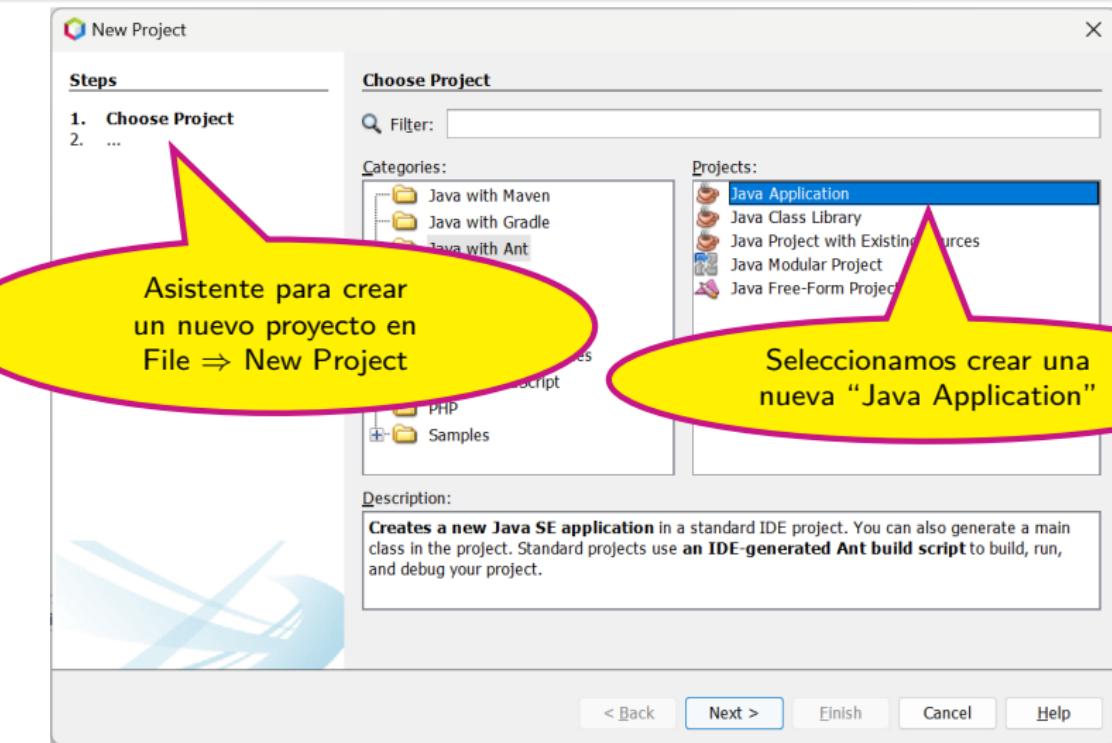


# Ant y NetBeans

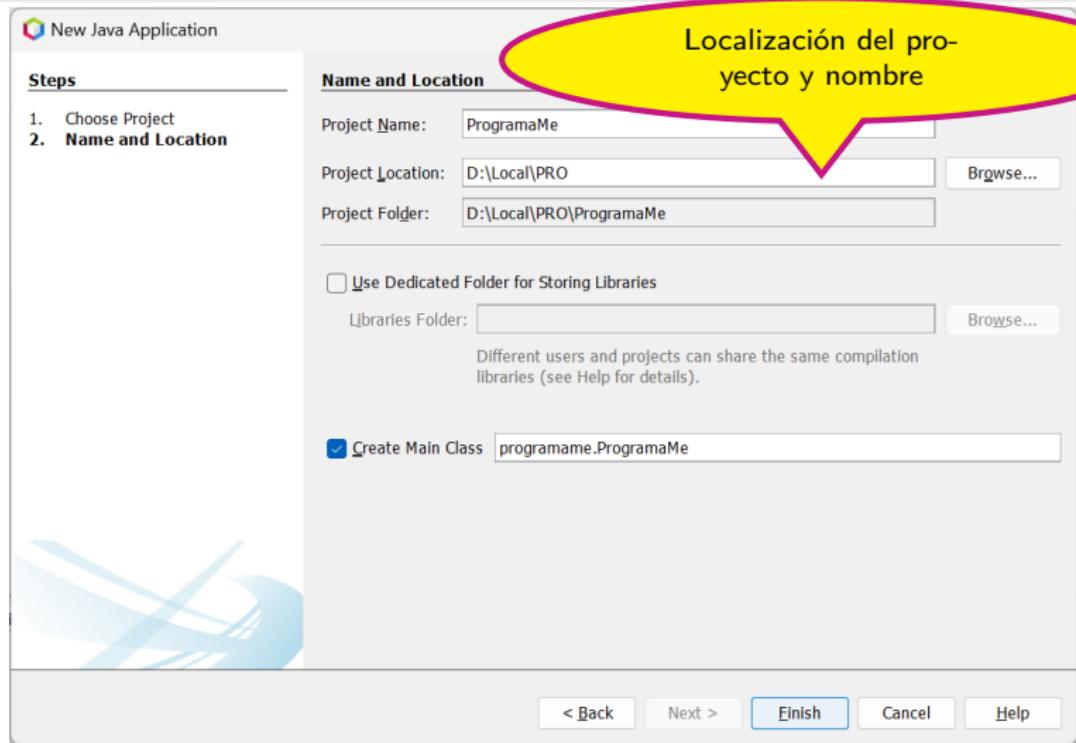
- El fichero build-impl.xml es generado automáticamente por la herramienta (podemos probar a borrarlo y ver como se regenera de nuevo) y cambia de versión en versión
- Representa el funcionamiento básico del proyecto NetBeans en Ant



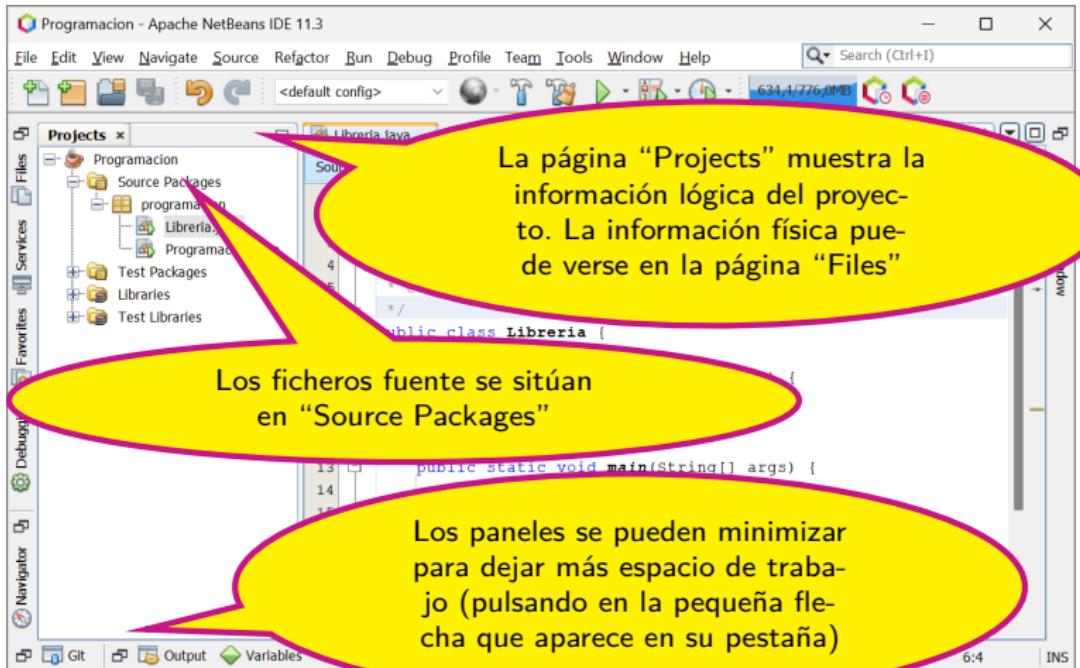
# Edición en NetBeans



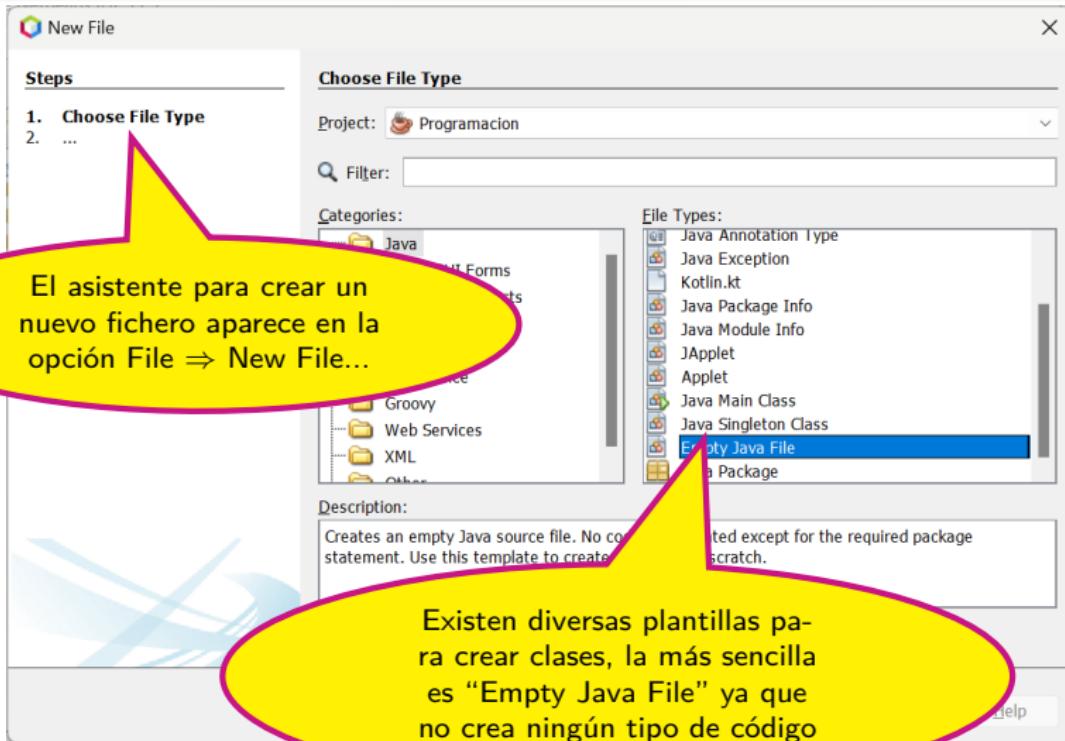
# Edición en NetBeans



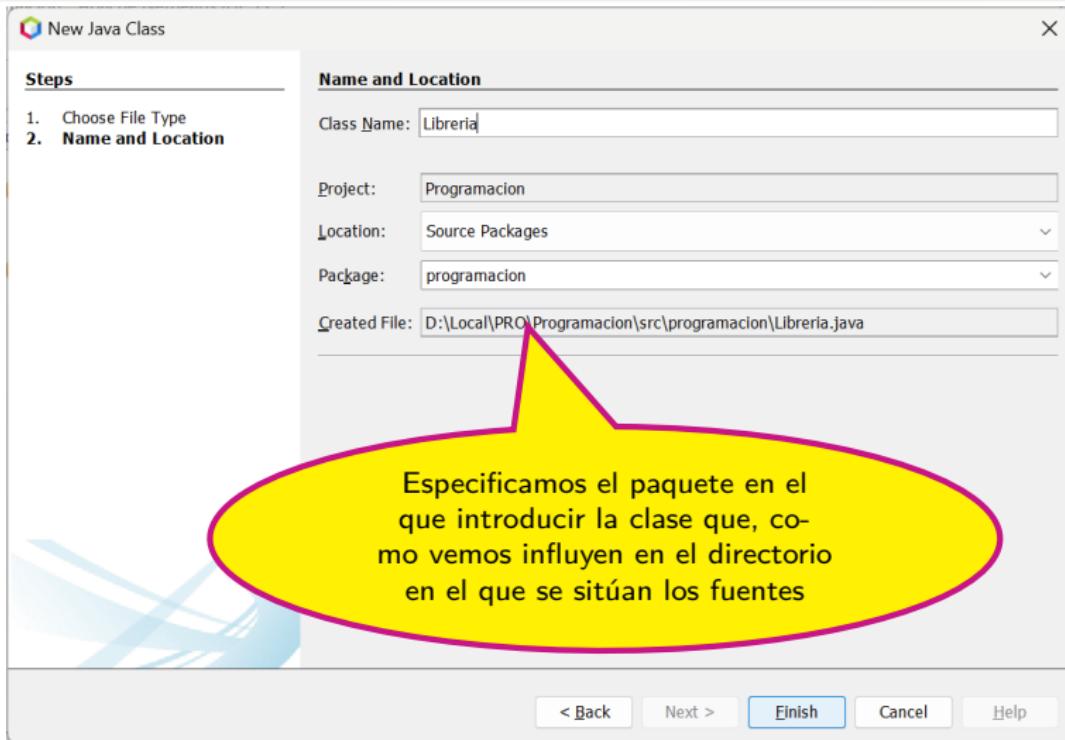
# Edición en NetBeans



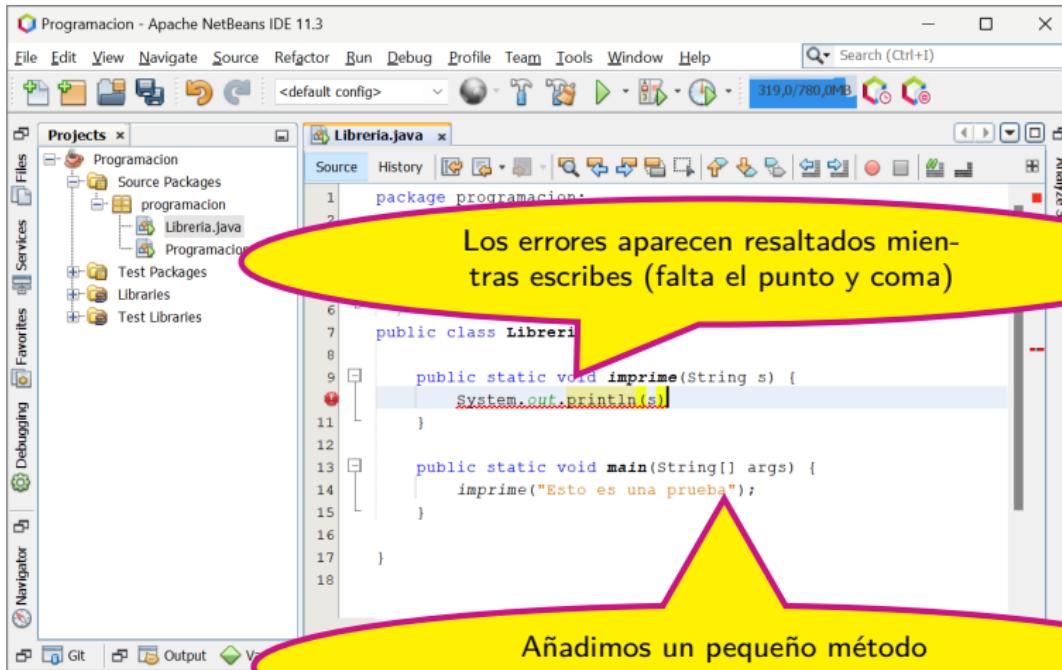
# Edición en NetBeans



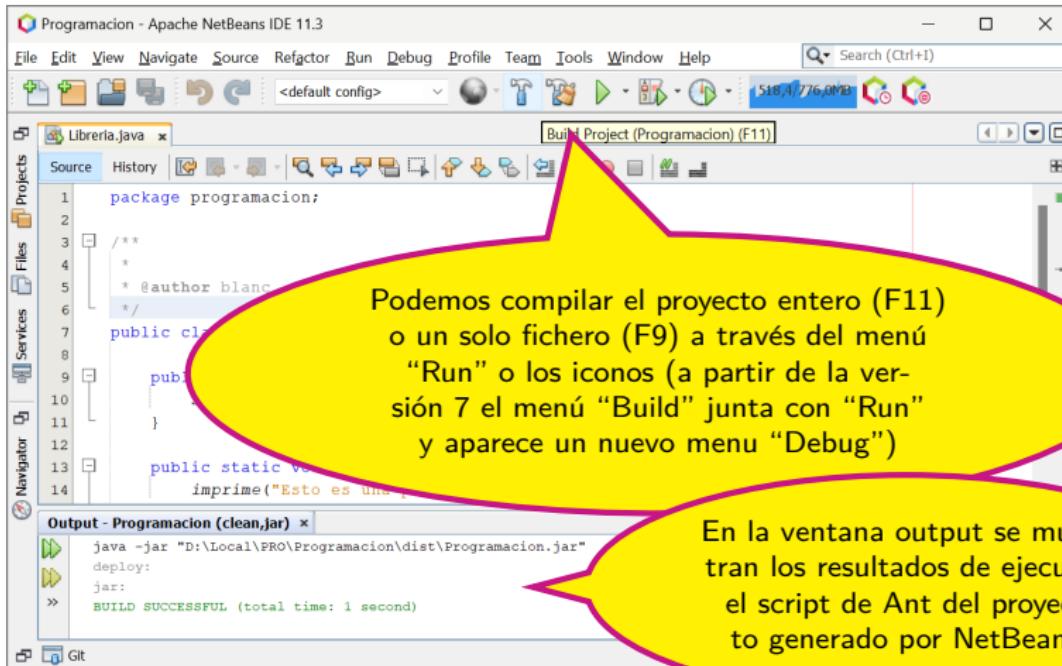
# Edición en NetBeans



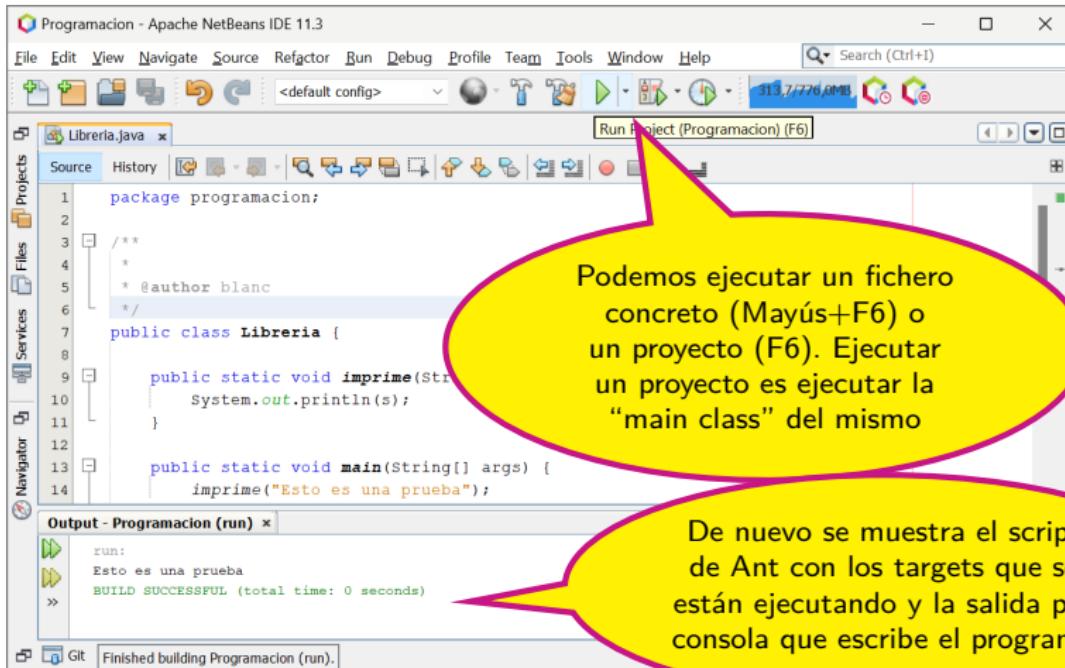
# Edición en NetBeans



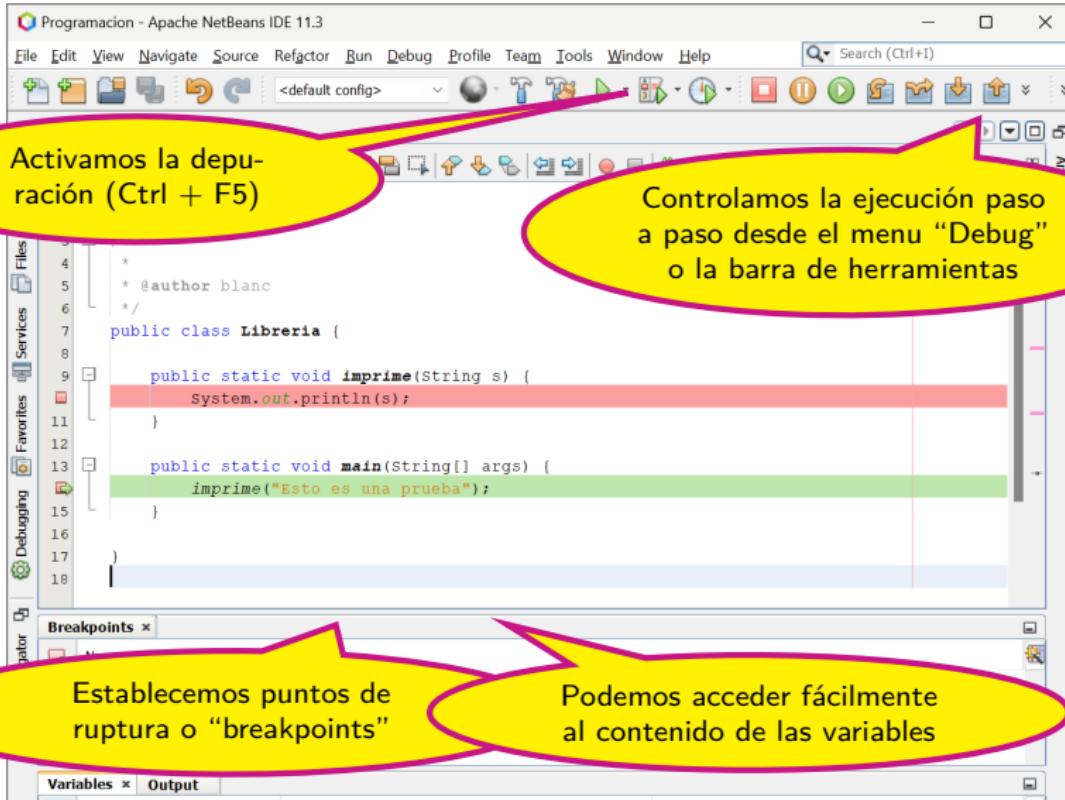
# Compilación en NetBeans



# Ejecución en NetBeans



# Depuración en NetBeans



# Índice

**1** Introducción

**2** La Herramienta NetBeans

**3** Refactorización

- Introducción
- Refactorizaciones
- Ejemplo: Refactorizando un blob
- Ejemplo: Refactorización a patrones en NetBeans



IES MURALLA ROMANA

# Refactorización

## Refactorización

Es un cambio realizado en la estructura interna del software que lo hace más fácil de entender y modificar, sin cambiar su comportamiento observable



# Refactorización

## Refactorización

Es un cambio realizado en la estructura interna del software que lo hace más fácil de entender y modificar, sin cambiar su comportamiento observable

- Generalmente las refactorizaciones realizan pocos cambios en el código, tratando de minimizar las posibilidades de hacer algo incorrecto, y manteniendo el comportamiento observable
- Aunque las refactorizaciones hagan pocos cambios, una serie encadenada de refactorizaciones puede conseguir una reestructuración significante del código



# Ejemplo

- En la redacción de la declaración de independencia de los EEUU, Benjamin Franklin revisó los escritos de Thomas Jefferson simplificándolos. Ante sus quejas le puso el siguiente ejemplo:
- Cartel de una tienda de sombreros:



# Ejemplo

- En la redacción de la declaración de independencia de los EEUU, Benjamin Franklin revisó los escritos de Thomas Jefferson simplificándolos. Ante sus quejas le puso el siguiente ejemplo:
- Cartel de una tienda de sombreros:



# Ejemplo

- En la redacción de la declaración de independencia de los EEUU, Benjamin Franklin revisó los escritos de Thomas Jefferson simplificándolos. Ante sus quejas le puso el siguiente ejemplo:
- Cartel de una tienda de sombreros:



# Ejemplo

- En la redacción de la declaración de independencia de los EEUU, Benjamin Franklin revisó los escritos de Thomas Jefferson simplificándolos. Ante sus quejas le puso el siguiente ejemplo:
- Cartel de una tienda de sombreros:



# Ejemplo

- En la redacción de la declaración de independencia de los EEUU, Benjamin Franklin revisó los escritos de Thomas Jefferson simplificándolos. Ante sus quejas le puso el siguiente ejemplo:
- Cartel de una tienda de sombreros:



# Ejemplo

- En la redacción de la declaración de independencia de los EEUU, Benjamin Franklin revisó los escritos de Thomas Jefferson simplificándolos. Ante sus quejas le puso el siguiente ejemplo:
- Cartel de una tienda de sombreros:



# Refactorización e infra/sobre-ingeniería

- La idea es mantener el diseño correcto, pero simple, sin adelantarnos a futuras e inciertas necesidades futuras.
- Evitamos así la sobre-ingeniería, siempre tratando de no caer en el otro extremo
- Si estas nuevas necesidades aparecen la refactorización permitirá adaptar el diseño a las nuevas funcionalidades
- Las pruebas de unidad automatizadas permitirán comprobar fácilmente que el código refactorizado tiene la misma funcionalidad que el anterior



# Principales motivaciones de la refactorización

- Hacer el código fácil de cambiar o que sea fácil de añadir una nueva característica
- Reducir la complejidad para facilitar la comprensión
- Eliminar repeticiones innecesarias
- Mejorar el diseño del código existente
- Mejorar el rendimiento del código existente
- Permitir el uso del código para otras necesidades distintas (o más generales) de las iniciales



# Algunas refactorizaciones

- **Componer métodos:** Extraer método,
- **Mover características entre objetos:** Mover método, Ocultar delegado, Extraer clase
- **Organizar datos:** Encapsular campo
- **Simplificar expresiones condicionales:** Reemplazar condicional con polimorfismo
- **Simplificar las llamadas a los métodos:** Renombrar método
- **Tratando con la generalización:** Extraer interfaz, Reemplazar herencia con delegación
- **Reorganización de un proyecto:** Recomponer la estructura de paquetes, adaptándola a nuevas necesidades o simplificando su organización

# Algunas refactorizaciones

- **Componer métodos:** Extraer método,
- **Mover características entre objetos:** Mover método, Ocultar delegado, Extraer clase
- **Organizar datos:** Encapsular campo
- **Simplificar expresiones condicionales:** Reemplazar condicional con polimorfismo
- **Simplificar las llamadas a los métodos:** Renombrar método
- **Tratando con la generalización:** Extraer interfaz, Reemplazar herencia con delegación
- **Reorganización de un proyecto:** Recomponer la estructura de paquetes, adaptándola a nuevas necesidades o simplificando su organización

## Catálogo

<http://refactoring.com/catalog/>



# Extraer método

## Situación

Un fragmento de código puede ser agrupado

## Refactorización

Convertir dicho fragmento en un método y darle un nombre que describa su propósito

```
void printOwing() {  
    printBanner();  
  
    //print details  
    System.out.println ("name: " + _name);  
    System.out.println ("amount " + getOutstanding());  
}
```



```
void printOwing() {  
    printBanner();  
    printDetails(getOutstanding());  
}  
  
void printDetails (double outstanding) {  
    System.out.println ("name: " + _name);  
    System.out.println ("amount " + outstanding);  
}
```

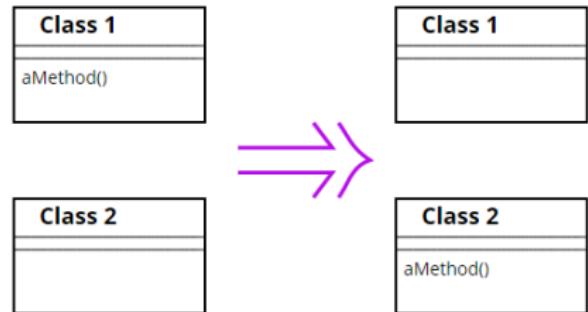
# Mover Método

## Situación

Un método usa más características de otra clase que de la clase propia en la que está definido

## Refactorización

Crear una copia del método en la clase que es más usada por el mismo. Borrar el método antiguo o mantenerlo como una delegación



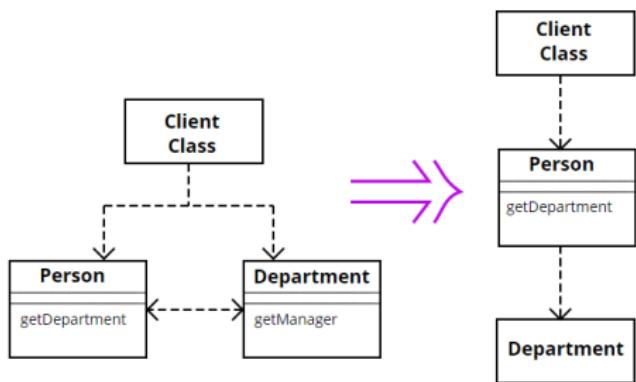
# Ocultar delegado

## Situación

Un cliente está llamando a la clase delegada de un objeto

## Refactorización

Crear métodos en el servidor que oculten el delegado



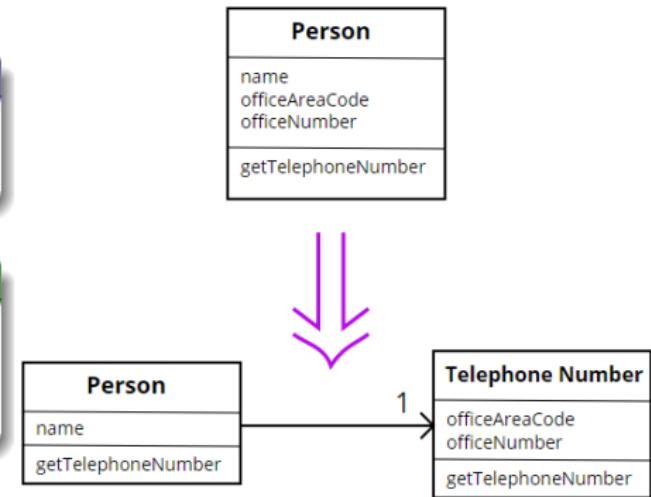
# Extraer clase

## Situación

Tenemos a una clase haciendo el trabajo que deberían hacer dos

## Refactorización

Crear una nueva clase y mover los campos y métodos relevantes de la vieja clase a la nueva



# Encapsular campo

## Situación

Existe un campo (atributo) con visibilidad pública

## Refactorización

Cambiar la visibilidad a privada y proveer de métodos de lectura y escritura

```
public String _name
```



```
private String _name;  
public String getName() {return _name;}  
public void setName(String arg) {_name = arg;}
```

# Reemplazar condicional con polimorfismo

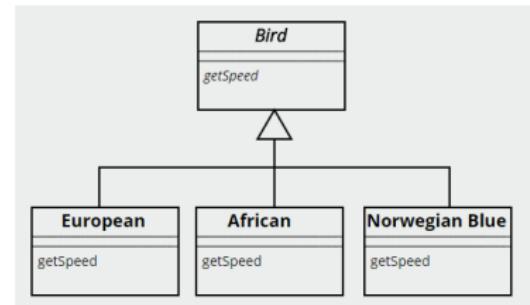
## Situación

Tenemos un método con una sentencia condicional que elige un comportamiento diferente dependiendo del tipo de un objeto

## Refactorización

Hacer el método original abstracto y mover cada rama condicional a un método que sobrescriba el método original

```
double getSpeed() {  
    switch (_type) {  
        case EUROPEAN:  
            return getBaseSpeed();  
        case AFRICAN:  
            return getBaseSpeed() - getLoadFactor() * _numberOfCoconuts;  
        case NORWEGIAN_BLUE:  
            return (_isNailed) ? 0 : getBaseSpeed(_voltage);  
    }  
    throw new RuntimeException ("Should be unreachable");  
}
```



# Renombrar método

## Situación

El nombre del método no refleja su propósito

## Refactorización

Cambiar el nombre del métodos

Customer
getinvcdtlmt



Customer
getInvoiceableCreditLimit



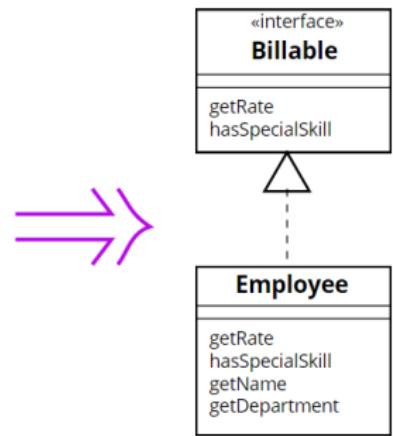
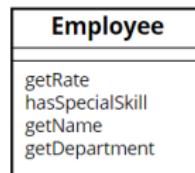
# Extraer interfaz

## Situación

Varios clientes usan un determinado subconjunto del interfaz público de una clase. O dos clases tienen parte de su interfaz en común

## Refactorización

Separar este subconjunto en un interfaz y hacer que la clase original implemente dicho interfaz



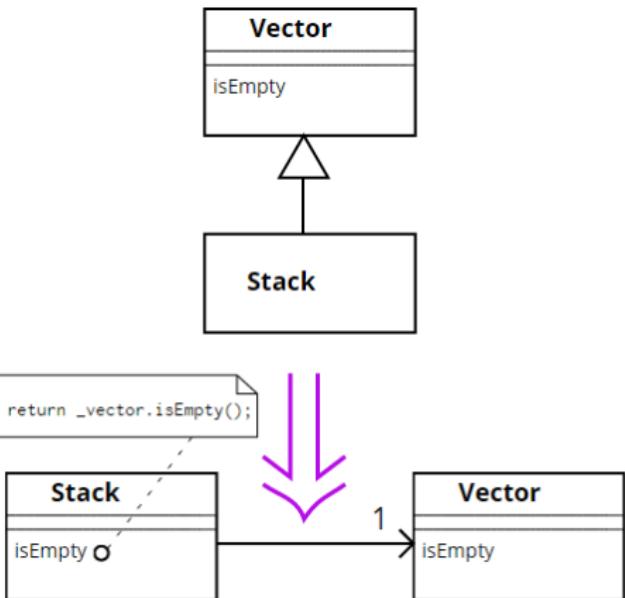
# Reemplazar herencia con delegación

## Situación

Una subclase sólo usa parte del interfaz de la superclase o no desea heredar ningún tipo de dato

## Refactorización

Crear un atributo para la superclase, ajustar los métodos para que deleguen en dicho atributo y eliminar la relación de herencia



# Ejemplo: Refactorizando un *blob*

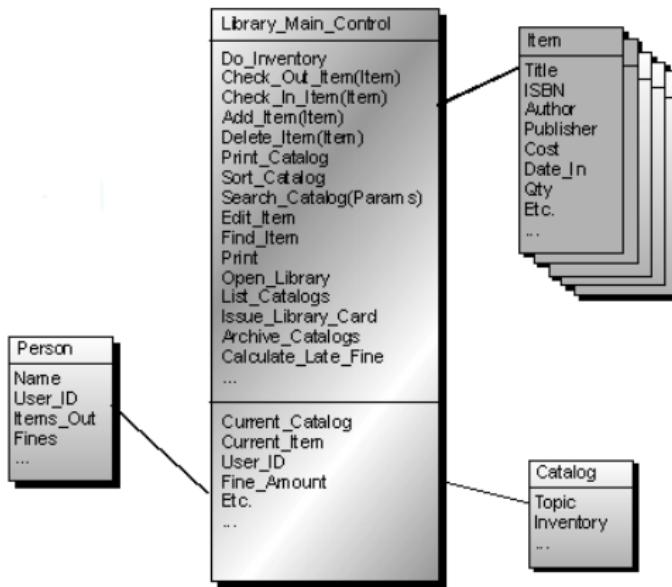
## Antipatrón: The Blob (la masa)

Un *blob* es un objeto que, como el alien de la película, crece engullendo responsabilidades pertenecientes a otros objetos hasta convertirse en el objeto predominante de la arquitectura (Clase Suprema)



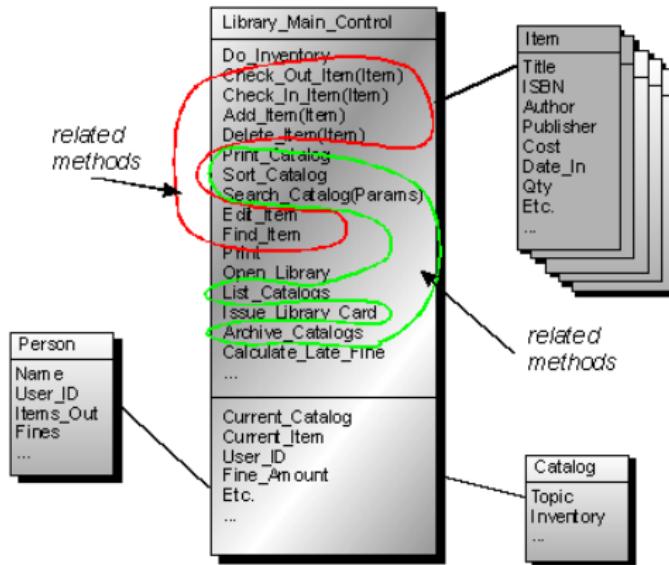
# Ejemplo: Refactorizando un *blob*

- El blob almacena la mayoría de los procesos mientras que las otras clases actúan como meros repositorios de datos



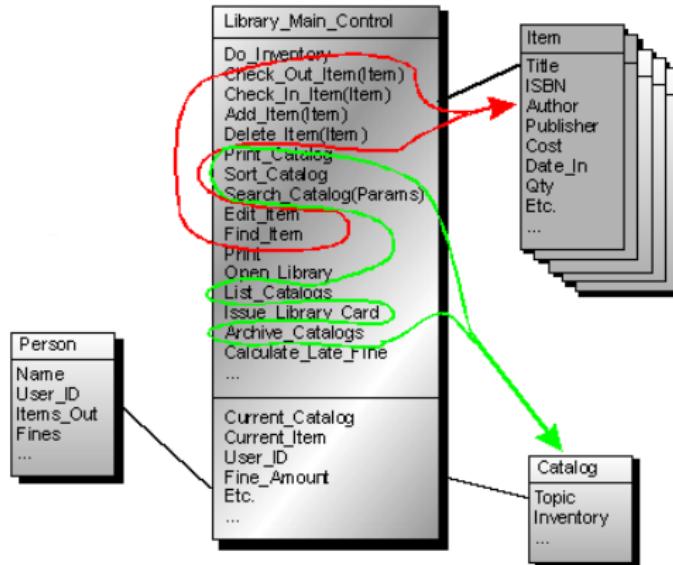
# Ejemplo: Refactorizando un *blob*

- 1<sup>er</sup> Paso: Identificamos operaciones relacionadas entre sí



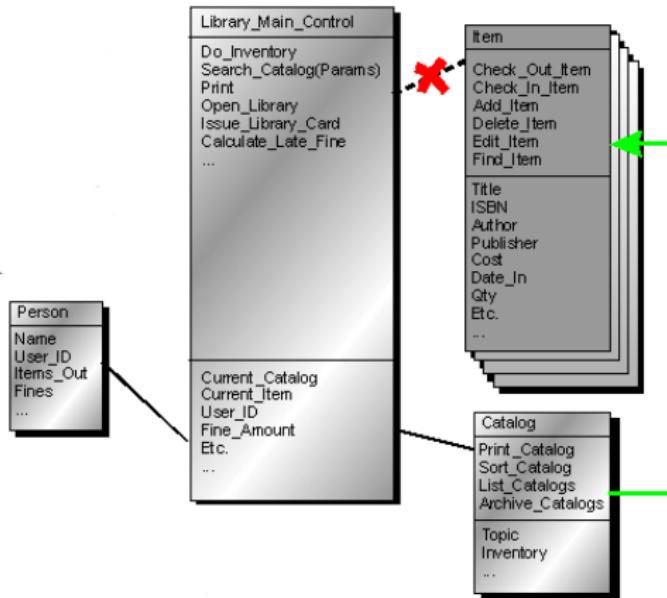
# Ejemplo: Refactorizando un *blob*

- **2º Paso:** Movemos estas operaciones a sus lugares naturales ⇒ Extraer método



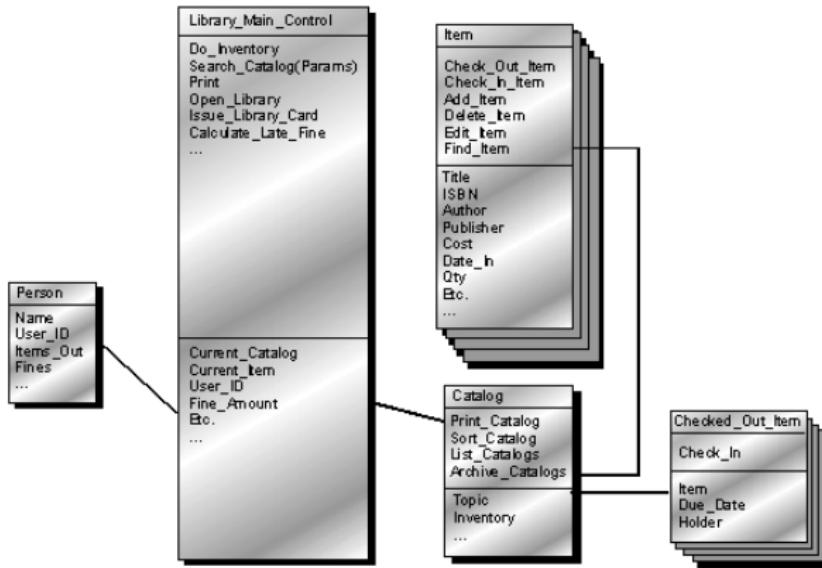
# Ejemplo: Refactorizando un *blob*

- **3<sup>er</sup> Paso:** Eliminamos asociaciones redundantes desacoplando objetos ⇒ **Ocultar delegado**



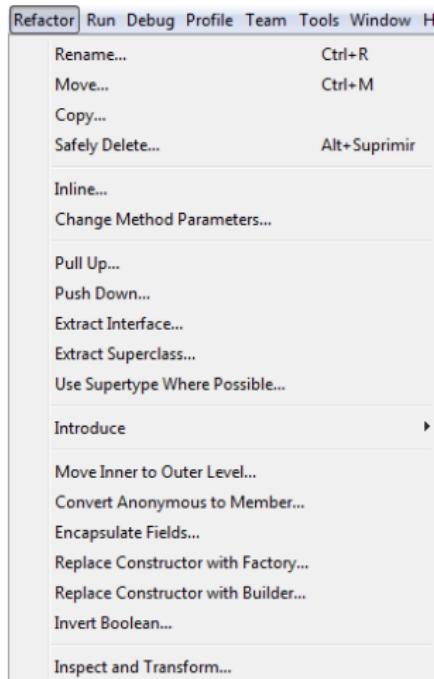
# Ejemplo: Refactorizando un *blob*

- **4º Paso:** Movemos los procesos transitorios a una clase transitoria que los represente mejor ⇒ **Extraer clase**



# Refactorizaciones en NetBeans

- El menú de refactorizaciones de NetBeans es muy completo y permite llevar a cabo muchas de las mismas de forma automática
- No sólo pueden llevarse a cabo sin que su funcionamiento puede configurarse y su resultado puede deshacerse si no fuese satisfactorio
- La refactorización a patrones consiste en cambiar un código existente para incluir un patrón de diseño



# UD 2: Entornos Integrados de Desarrollo (NetBeans)

## Contornos de desenvolvimento (CODE)

**Víctor Blanco**

Curso 2023-2024



XUNTA  
DE GALICIA

IES Muralla Romana  
Dpto. de Informatica