

Programación Orientada a Objetos

Tema de Prácticas 1: Introducción a la compilación Java y a la herramienta NetBeans

Eduardo Mosqueira Rey



LIDIA
**Laboratorio de Investigación y
desarrollo en Inteligencia Artificial**



Departamento de Computación
Universidade da Coruña, España



Índice



- 1. “Hola Mundo” en Java**
- 2. Compilación en línea**
- 3. La herramienta NetBeans**



Índice



1. “Hola Mundo” en Java

- “Hola Mundo” tradicional
- “Hola Mundo” orientado a objetos



“Hola Mundo” en Java

“Hola Mundo” tradicional



- **Características**

- Tradicionalmente el primer programa que se escribe en un lenguaje de programación suele ser imprimir por pantalla un lacónico “Hola Mundo”
- Como todo en Java debe ir dentro de una clase es necesario crear una clase para imprimir el mensaje, en nuestro caso será la clase HolaMundo
- Si queremos ejecutar el código de una clase necesitamos un método (función) main que sirva como punto de partida, en este método pondremos el mensaje “Hola Mundo”.
- El método main tiene que llamarse así obligatoriamente (Java distingue entre mayúsculas y minúsculas) además de ser obligatoriamente público y estático, devolver void y aceptar un array de Strings como parámetro

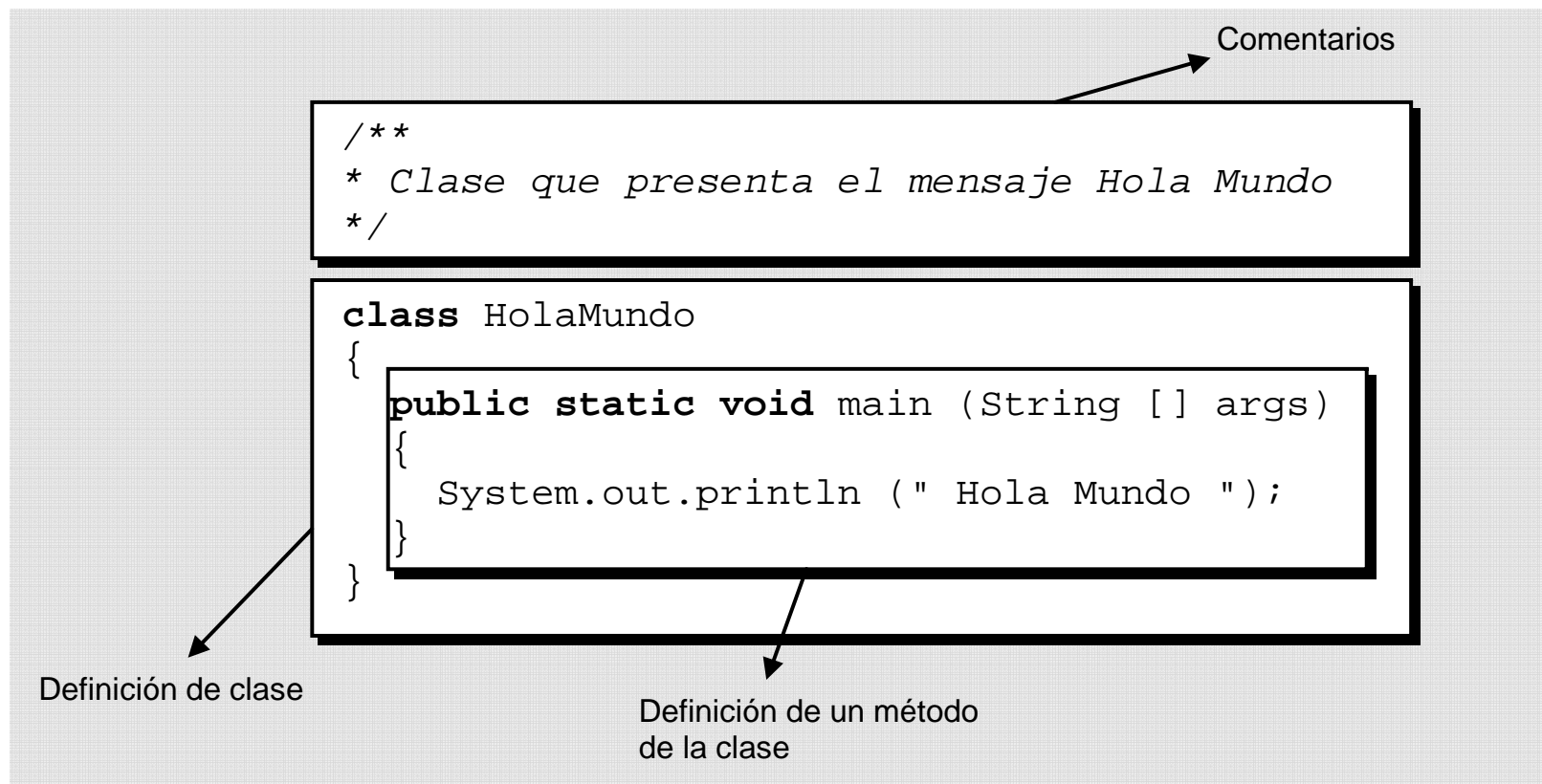


“Hola Mundo” en Java

“Hola Mundo” tradicional



- Programa HolaMundo





“Hola Mundo” en Java

“Hola Mundo” orient. a objetos



- **El ejemplo del “Hola Mundo” es un mal ejemplo de la orientación a objetos porque:**
 - Se crea una clase pero no se crea un objeto de la clase
 - El intérprete llama al método main de la clase pero no manda ningún mensaje a una instancia de una clase
- **Un ejemplo orientado a objetos debería incluir:**
 - La creación de objetos además de la definición de clases
 - El llamamiento a métodos de instancia (no estáticos) sobre el objeto creado
- **Por ello vamos a crear una nueva versión del HolaMundo**



“Hola Mundo” en Java

“Hola Mundo” orient. a objetos



- Programa HolaMundo (versión OO)

```
class HolaMundoOO
{
    public void imprimeHola()
    {
        System.out.println ( " Hola Mundo " );
    }
}
```

La nueva clase HolaMundo incluye un método no estático (necesita un objeto para ser ejecutado) denominado imprimeHola

Creamos una nueva clase únicamente para almacenar el método main

```
class HolaMundo
{
    public static void main(String[] args)
    {
        HolaMundoOO miHola = new HolaMundoOO();
        miHola.imprimeHola();
    }
}
```

Creamos una instancia de la clase HolaMundo a través del operador new

Llamamos al método de instancia imprimeHola



Índice



2. Compilación en línea

- Compilación simple**
- Compilación compleja**
- Compilación con ant**



Compilación en línea

Compilación simple



- **Como compilar un programa Java**
 - El directorio en el que se encuentran las herramientas Java debe estar en el path del sistema
 - Teclear “javac nombrefichero.java”
 - Obtendremos tantos ficheros .class como clases existen en el fichero del código fuente
- **Como ejecutar un programa Java**
 - Teclear “java nombreclase”
 - El fichero nombreclase.class debe estar en un directorio incluido en el CLASSPATH
 - CLASSPATH es una variable de entorno que indica el camino por defecto en el que están las clases Java
 - Generalmente el directorio actual está en el CLASSPATH por lo que lo más sencillo es ejecutar el intérprete en el mismo directorio en el que está el fichero .class



Compilación en línea

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>_
```



Compilación en línea

Compilación compleja



- **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 (y los subpaquetes con subdirectorios)
 - Al no existir paquetes todos los fuentes necesarios residen en el mismo directorio
 - No se utilizan librerías externas aparte del API de Java



Compilación en línea

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 “Librería” 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”
 - La instrucción de compilación sería:
 - `javac -d build -classpath lib/Libreria.jar src/poo/holamundo/HolaMundo.java`
 - En el directorio “build” se crea una estructura de directorios similar a la existente en el directorio src
 - La instrucción de ejecución sería
 - `java -classpath build;lib/Libreria.jar poo/holamundo/HolaMundo`



Compilación en línea

Compilación compleja



- Clases HolaMundo

- Clase Librería

El paquete al que pertenecen las clases se incluye como la primera instrucción del fichero con el formato **package nombrepaquete**

```
package poo.holamundo;

import utilidades.Libreria;

class HolaMundoOO
{
    public String devuelveHola()
    {
        return " Hola Mundo ";
    }
}

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

```
package utilidades;

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

La sentencia **import** permite usar la clase **Libreria** en el código sin necesidad de precederla del nombre de su paquete



Compilación en línea

Compilación compleja



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

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.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>
```



Compilación en línea

Compilación compleja



- **Problema**
 - A medida que va creciendo en complejidad un proyecto también crece la complejidad de las instrucciones de compilación y ejecución
- **Solución 1: archivos .bat**
 - La primera y sencilla solución puede ser poner estas instrucciones en sencillos ficheros de procesamiento por lotes (.bat) de Windows o scripts de Unix/Linux
 - Puede ser una solución ideal para problemas sencillos como el visto en las transparencias anteriores
 - Es una solución no portable entre distintas plataformas
 - A medida que el proyecto crece y el número de tareas a realizar aumenta (compilar, ejecutar, documentar, empaquetar, etc.) se vuelve una solución incómoda e ineficaz.



Compilación en línea

Compilación compleja



- **Solución 2: ficheros make**
 - La herramienta make (en sus distintas versiones) ha sido tradicionalmente la solución utilizada por C/C++ para compilar y ejecutar programas
 - Está presente por defecto en las plataformas Unix/Linux
 - Problemas de portabilidad
 - Las distintas versiones de make (GNU, BSD) no son totalmente compatibles entre sí
 - Las herramientas pueden extenderse creando programas para el SO operativo sobre el que se trabaja, pero eso nos liga a dicho SO
 - Adecuación a Java
 - Make no fue creado para trabajar con Java, por lo que no tiene un conocimiento específico de las particularidades de Java (por ejemplo, el CLASSPATH)
 - El formato de make es propio, bastante estricto y desconocido para todo aquel que no lo haya usado nunca.



Compilación en línea

Compilación compleja



- **Solución 3: ficheros Ant**
 - Ant (Another Neat Tool) es una herramienta desarrollada por Apache para construir programas Java (<http://ant.apache.org>)
 - Sus principales objetivos son sustituir a make para construir programas Java eliminando las incomodidades del mismo y favoreciendo el desarrollo multiplataforma:
 - Las tareas de Ant son ejecutadas por clases Java, esto garantiza su portabilidad en toda aquella plataforma que tenga una JVM disponible además de ser fácil de extender
 - Ant tiene un conocimiento especial de las necesidades de Java permitiendo configurarlas de forma sencilla
 - El formato usado por Ant es XML, un formato popular y conocido y para el cual existen múltiples herramientas disponibles



Compilación en línea

Compilación compleja

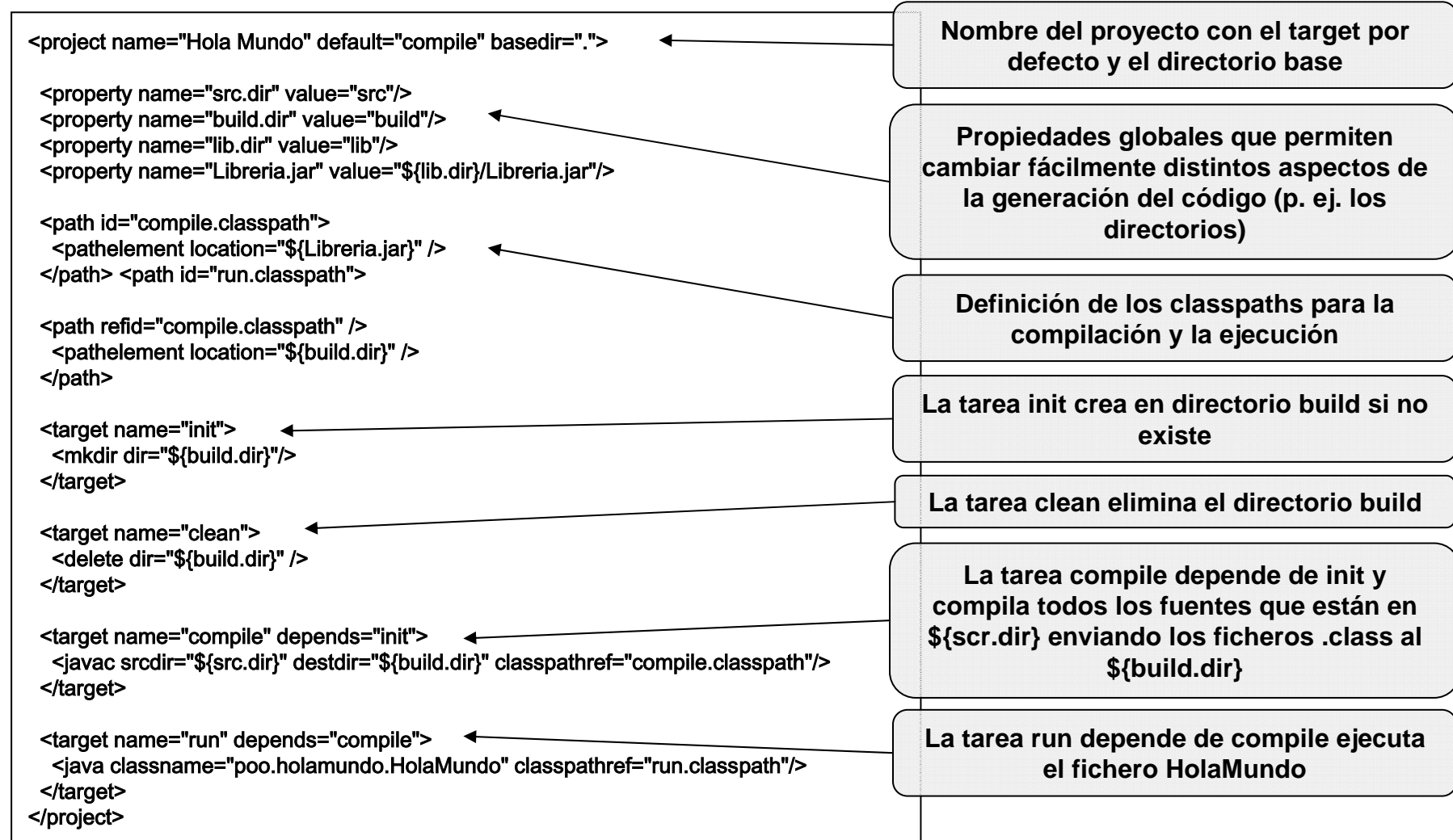


- **Características de Ant**
 - Por defecto Ant busca un fichero de compilación denominado “build.xml”
 - Cada fichero contiene una etiqueta <project> donde se especifican las características del proyecto
 - Además tendrá un conjunto de etiquetas <target> que indican los objetivos que pueden realizarse 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 en línea

Compilación compleja





Compilación en línea

Compilación compleja



```
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 la indicada por defecto

Como “compile” depende de “init” es necesario ejecutar antes el target “init”

Para ejecutar una tarea específica es necesario teclear “ant nombre_tarea”

“run” depende de “compile” y “compile” de “init” sin embargo en estas dos últimas tareas no se realiza nada porque no es necesario



Índice



3. La Herramienta NetBeans

- Introducción**
- Proyectos**
- Compilación**
- Ejecución**
- Depuración**
- Configuración**
- Pruebas de unidad**



La Herramienta NetBeans

Introducción



- **IDE OpenSource para el desarrollo de código Java mantenido por Sun (<http://www.netbeans.org>)**
- **Muy completo permitiendo el desarrollo en las plataformas Micro, Standard y Enterprise. No tiene nada que envidiar a otros IDEs comerciales**
- **El IDE Eclipse le ha robado una buena cuota de mercado basandose en defectos evidentes de versiones previas de NetBeans (eficiencia, usabilidad, etc.)**
- **Desde la versión 4.1 se han mejorado muchos de los problemas anteriores permitiendo al IDE recuperar parte de su cuota de mercado**
- **Elegido para la asignatura por dos razones:**
 - **Al estar mantenido por Sun los cambios en el lenguaje tienen su reflejo más inmediato en NetBeans**
 - **Su estructura compacta (aunque permite el uso de plug-ins) la hace mas sencilla de utilizar para el usuario neófito.**



La Herramienta NetBeans

Proyectos



- **NetBeans siempre trabaja sobre proyectos, no puede compilar ficheros que no estén integrados dentro de un proyecto**
- **Los proyectos NetBeans se basan en Ant pero no es necesario conocer Ant para manejarlos**
- **Una estructura típica de un directorio de un proyecto NetBeans incluye los siguientes subdirectorios**
 - **build: donde se sitúan los ficheros .class compilados**
 - **dist: donde se sitúan el fichero empaquetado .jar**
 - **nbproject: incluye la información del proyecto NetBeans y generalmente no debe tocarse**
 - **src: donde se incluyen los fuentes**
 - **test: donde se incluyen los fuentes de los tests JUnit para realizar pruebas de unidad**
- **La herramienta provee de asistentes para empezar proyectos desde cero o para crear un proyecto con fuentes ya existentes**



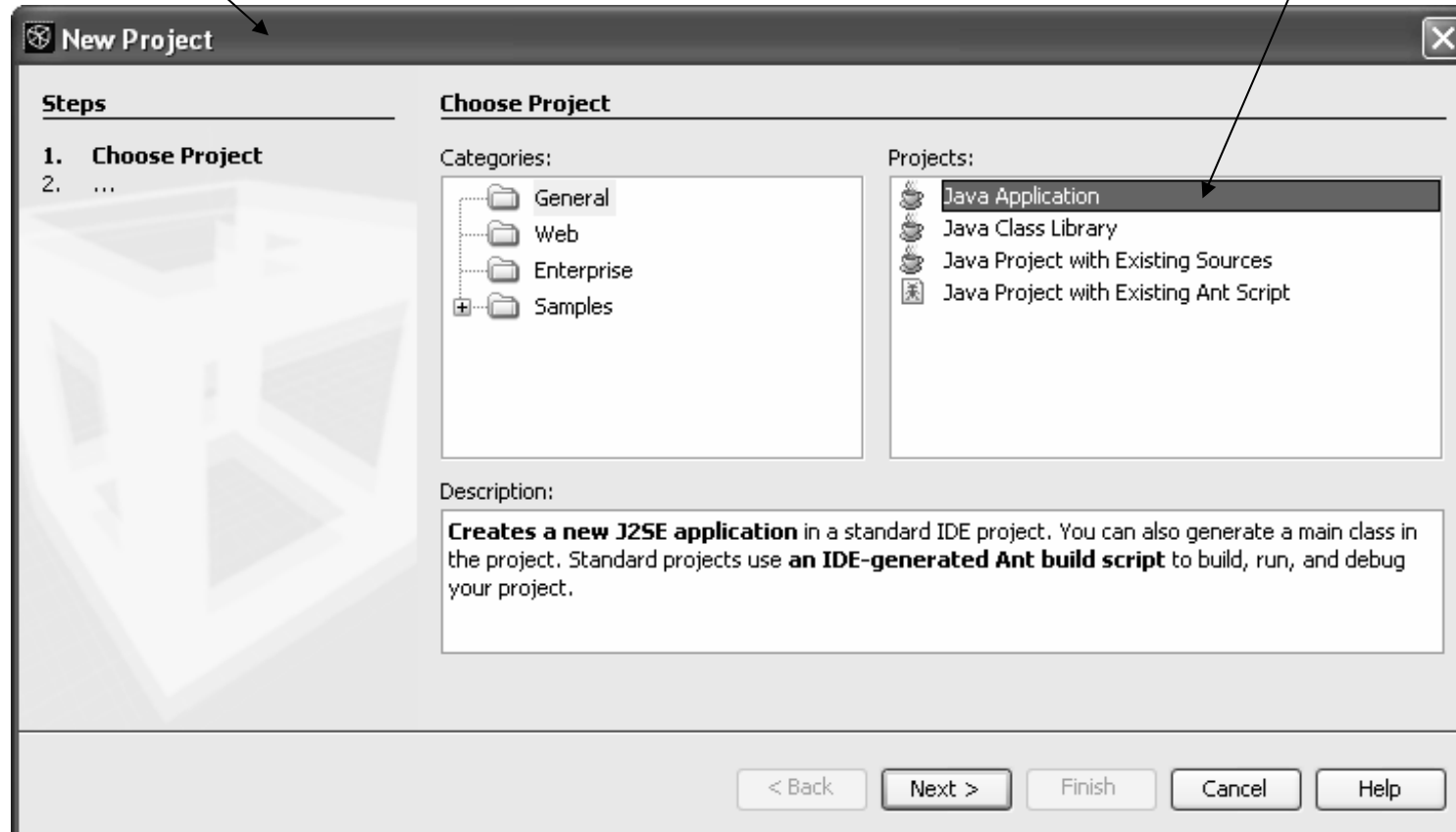
La Herramienta NetBeans

Proyectos



El asistente para crear un nuevo proyecto aparece en la opción File → New Project...

Seleccionamos crear una nueva Java Application





La Herramienta NetBeans Proyectos



Elegimos la localización del proyecto y su nombre

New Java Application

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name:

Project Location:

Project Folder:

☒ Set as Main Project

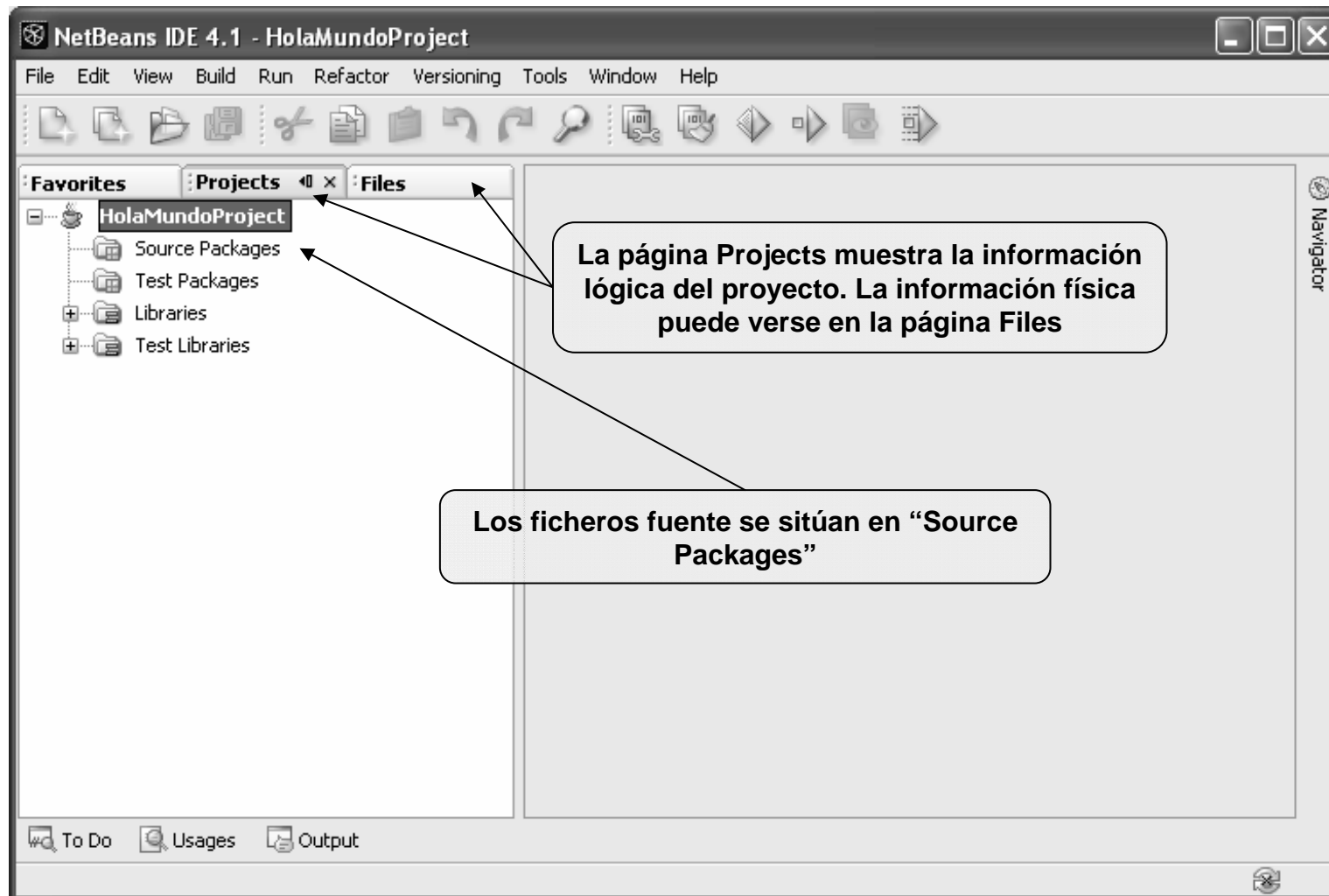
☐ Create Main Class

< Back Next > Finish Cancel Help

Fijamos el proyecto como Main Project para que sea tenido en cuenta en los comandos que hacen referencia a los proyectos (compilar, ejecutar, etc.)



La Herramienta NetBeans Proyectos





La Herramienta NetBeans

Proyectos



Existen diversas plantillas para crear clases, la más sencilla es Empty Java File ya que no crea ningún tipo de código

El asistente para crear un nuevo fichero aparece en la opción File → New File...

New File

Steps

1. Choose File Type
2. ...

Choose File Type

Project: HolaMundoProject

Categories:

- Web
- Java Classes
- Java GUI Forms
- JavaBeans Objects
- JUnit
- XML
- Ant Build Scripts
- Other

File Types:

- Java Class
- Empty Java File
- Java Interface
- Java Enum
- Java Annotation Type
- Java Exception
- Java Main Class
- JApplet
- Applet

Description:

Creates an empty Java source file. No code is generated except for the required package statement. Use this template to create a class from scratch.

< Back Next > Finish Cancel Help



La Herramienta NetBeans

Proyectos



New Empty Java File

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name:

Project:

Location:

Package:

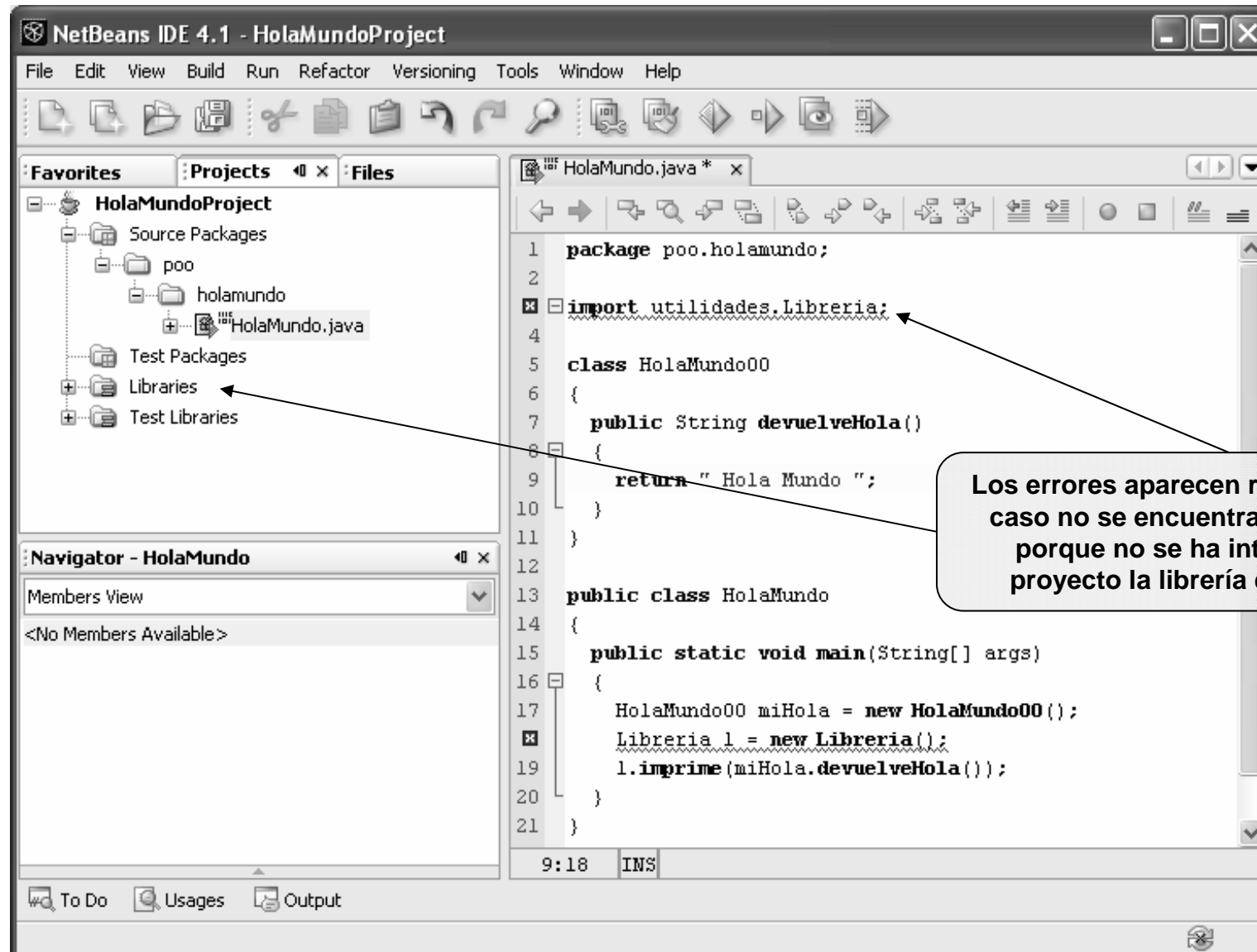
Created File:

Especificamos el paquete en el que introducir la clase que, como vemos influyen en el directorio en el que se sitúan los fuentes

< Back Next > Finish Cancel Help



La Herramienta NetBeans Proyectos





La Herramienta NetBeans

Compilación



Podemos compilar un proyecto y empaquetar su resultado en un Jar

Pulsando sobre el botón derecho en "Libraries" y eligiendo "Add Jar/Folder..." podemos añadir las librerías que nos hagan falta

En la ventana del navegador podemos ver el contenido de una clase y navegar por los distintos elementos

También podemos compilar el fichero actual (eligiendo Build → Compile "HolaMundo.java" o F9). El resultado se puede ver como el resultado de ejecutar un script de ant



La Herramienta NetBeans

Ejecución



NetBeans IDE 4.1 - HolaMundoProject

File Edit View Build Run Refactor Versioning Tools Window Help

Favorites **Projects** **Files**

holamundo
+ HolaMundo.java
Test Packages
Libraries
+ Libreria.jar
+ JDK 1.5 (Default)
Test Libraries

Navigator - HolaMundo

Members View

HolaMundo.main(String[] args)
devuelveHola()

Output - HolaMundoProject (run-single)

```
init:|
deps-jar:
compile-single:
run-single:
  Hola Mundo
BUILD SUCCESSFUL (total time: 0 seconds)
```

Code Editor: HolaMundo.java

```
1 package poo.holamundo;
2
3 import utilidades.Libreria;
4
5 class HolaMundo00
6 {
7     public String devuelveHola()
8     {
9         return "Hola Mundo ";
10    }
11 }
```

Annotations:

- Podemos ejecutar un proyecto pero para ello necesitaremos indicar qué clase contiene el main principal que debemos ejecutar
- También podemos ejecutar el fichero actual (eligiendo Run → RunFile → Run "HolaMundo.java" o Mayúsculas+F6). El resultado también se puede ver como el resultado de ejecutar un script de ant



La Herramienta NetBeans

Depuración



NetBeans IDE 4.1 - HolaMundoProject

File Edit View Build Run Refactor Versioning Tools Window Help

Podemos hacer una ejecución del proyecto que tenga en cuenta los breakpoints (necesitamos fijar cuál es la clase principal del proyecto)

La ejecución paso a paso se controla desde los botones de la barra de menú o las teclas rápidas como F7 y F8 (como vemos al entrar en depuración aparecen barras y apartados no visibles en edición)

Pulsando sobre los números podemos establecer puntos de ruptura o breakpoints

Podemos acceder fácilmente al contenido de las variables para consultar su valor

```
12
13 public class HolaMundo
14 {
15     public static void main(String[] args)
16     {
17         HolaMundo00 miHola = new HolaMundo00();
18         Libreria l = new Libreria();
19         l.imprime(miHola.devuelveHola());
20     }
21 }
22
```

14:1 INS

Watches

Name	Type	Value
args	String[]	#126(length=0)
miHola	HolaMundo00	#127
l	Libreria	#128

Local Variables

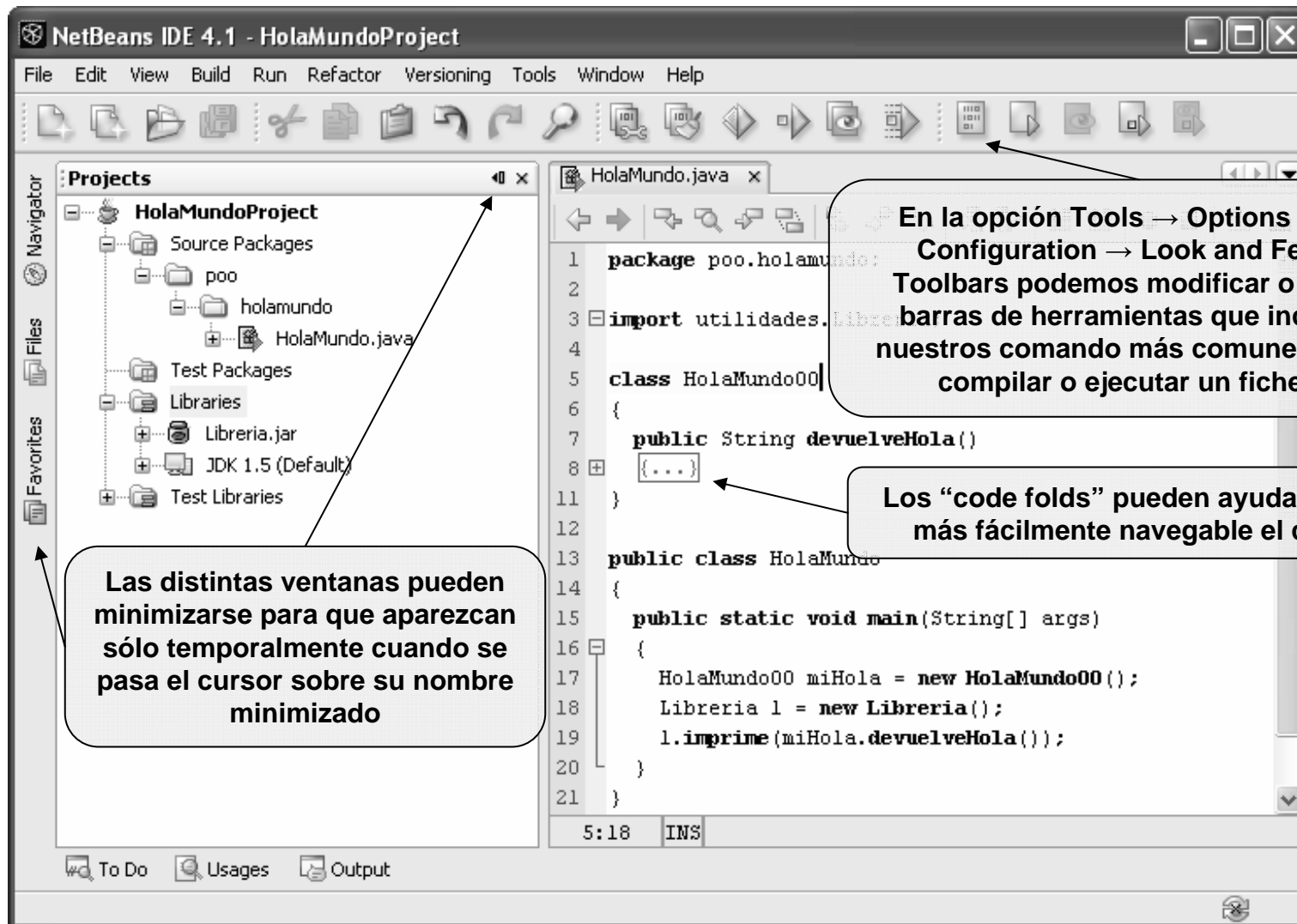
Name	Type	Value
args	String[]	#126(length=0)
miHola	HolaMundo00	#127
l	Libreria	#128

To Do Usages Output

Thread main stopped at HolaMundo.java:19.



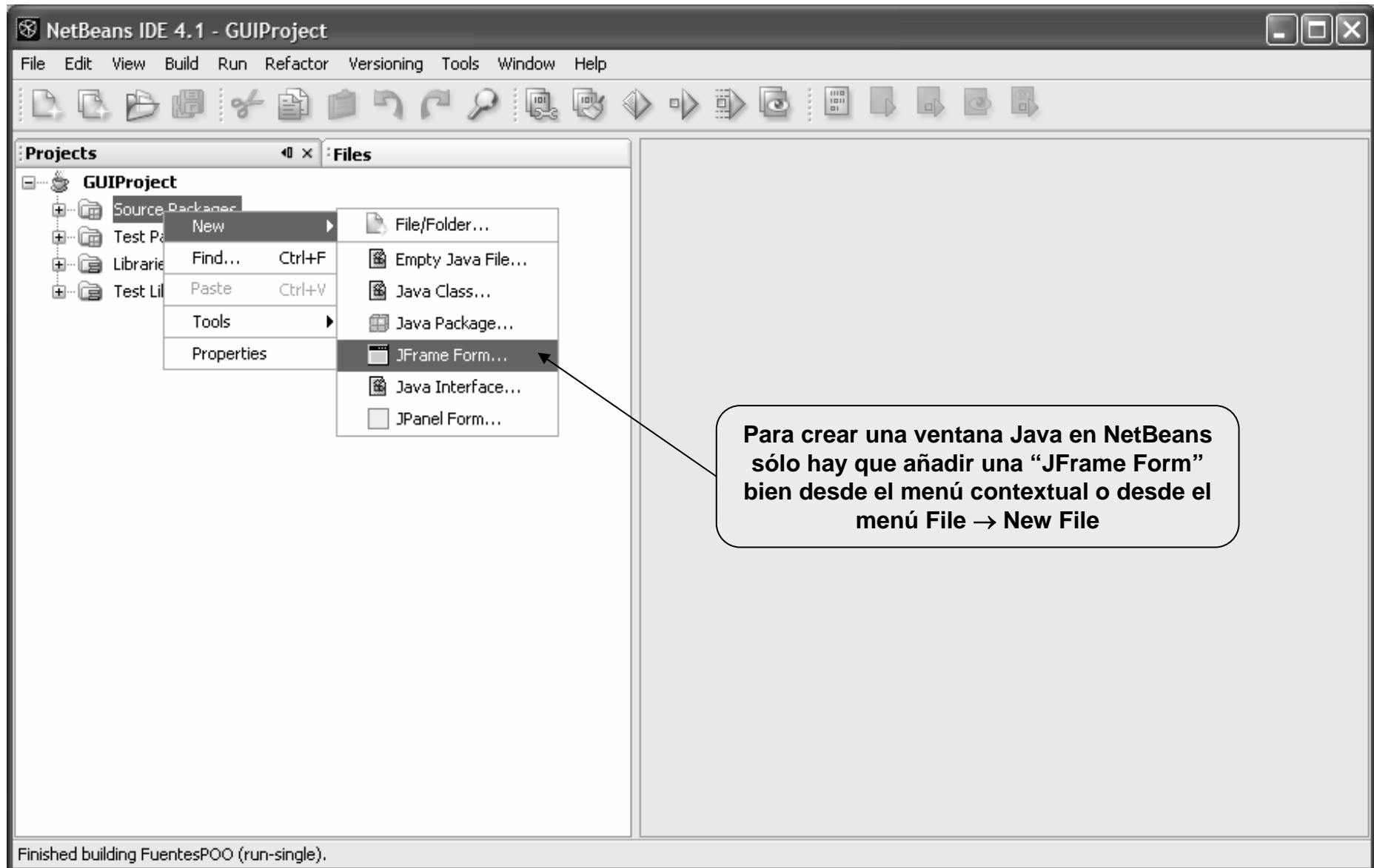
La Herramienta NetBeans Configuración





Interfaces de usuario en Java

Diseñador NetBeans





Interfaces de usuario en Java

Diseñador NetBeans



The screenshot displays the NetBeans IDE 4.1 interface. The main window is titled 'MiFrame' and is in 'Design' mode. The interface is divided into several panes:

- Projects/Files:** Shows a project named 'GUIProject' with source packages, test packages, and libraries. A file 'MiFrame.java' is selected.
- Inspector:** Shows a hierarchical tree of components for 'Form MiFrame', including 'Other Components', '[JFrame]', and 'BorderLayout'.
- Palette:** A list of Swing components available for addition, including JLabel, JButton, JToggleButton, JCheckBox, JRadioButton, ButtonGroup, and JComboBox.
- No Properties:** A pane for viewing and editing the properties of the selected component.

Three callout boxes provide additional information:

- El diseñador de formularios incluye dos vistas: una vista gráfica y otra vista del código fuente. La mayoría de las acciones se llevarán a cabo a través de la vista gráfica** (The form designer includes two views: a graphical view and a source code view. Most actions will be performed through the graphical view).
- La paleta nos permite seleccionar los componentes Swing que queremos añadir a nuestra JFrame** (The palette allows us to select the Swing components we want to add to our JFrame).
- El inspector nos muestra los componentes de la JFrame de forma jerárquica. Es muy útil porque visualmente a veces resulta complicado dilucidar la estructura de los componentes** (The inspector shows the components of the JFrame in a hierarchical manner. It is very useful because visually it can sometimes be complicated to clarify the structure of the components).



Interfaces de usuario en Java

Diseñador NetBeans



The screenshot shows the NetBeans IDE 4.1 GUIProject window. The main area is in Design view, showing a JFrame with components: jTextField1, Boton 1, and jButton2. The Palette on the right lists Swing components: JLabel, JButton, JToggleButton, JCheckBox, JRadioButton, ButtonGroup, and JComboBox. The Inspector on the left shows the component hierarchy: Form MiFrame, Other Components, [JFrame], BorderLayout, jTextField1 [JTextField], jButton1 [JButton], and jButton2 [JButton]. The Properties window on the right shows the properties for jButton1 [JButton], including action, background, componentPopupMenu, font, foreground, icon, mnemonic, and text (set to Boton 1).

Seleccionando un componente en la paleta y posteriormente pulsando sobre la JFrame añadiremos dicho componente a la ventana

El inspector de propiedades permite modificar las propiedades de cada componente gráfico



Interfaces de usuario en Java

Diseñador NetBeans



NetBeans IDE 4.1 - GUIProject

File Edit View Build Run Refactor Versioning Tools Window Help

Projects Files

GUIProject

- Source Packages
 - <default package>
 - MiFrame.java
- Test Packages
- Libraries
- Test Libraries

Inspector

Form MiFrame

- Other Components
- [JFrame]
 - BorderLayout
 - Set Layout
 - FlowLayout
 - BorderLayout
 - GridLayout
 - GridBagLayout
 - CardLayout
 - BoxLayout
 - AbsoluteLayout
 - Null Layout
 - Properties
 - jTextField
 - jButton1 [JButton]
 - jButton2 [JButton]

MiFrame * x

Source Design

Boton 1 Boton 2

A través del inspector podemos modificar el layout por defecto de la JFrame

Palette

Swing

- JLabel
- JButton
- JToggleButton
- JCheckBox
- JRadioButton
- ButtonGroup
- JComboBox

BorderLayout - Properties

Properties

Horizontal Gap	0
Vertical Gap	0

BorderLayout



Interfaces de usuario en Java

Diseñador NetBeans



NetBeans IDE 4.1 - GUIProject

File Edit View Build Run Refactor Versioning Tools Window

Projects Files

GUIProject

- Source Packages
 - <default package>
 - MiFrame.java
- Test Packages
- Libraries
- Test Libraries

Source Design

En todo momento podemos ver el código generado pulsando en "source". Este código es 100% Java por lo que puede usarse perfectamente fuera de NetBeans

El código marcado en azul es gestionado por NetBeans, por lo que no puede ser modificado si queremos que el diseñador de formularios funcione correctamente

```

jButton1.setText("Boton 1");
jButton1.addActionListener(new java.awt.event.ActionListener()
{
    public void actionPerformed(java.awt.event.ActionEvent evt)
    {
        jButton1ActionPerformed(evt);
    }
});

getContentPane().add(jButton1, java.awt.BorderLayout.WEST);

jButton2.setText("Boton 2");
getContentPane().add(jButton2, java.awt.BorderLayout.EAST);

pack();
}
// </editor-fold>

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
{
    jTextField1.setText("Has pulsado el boton 1");
}

```

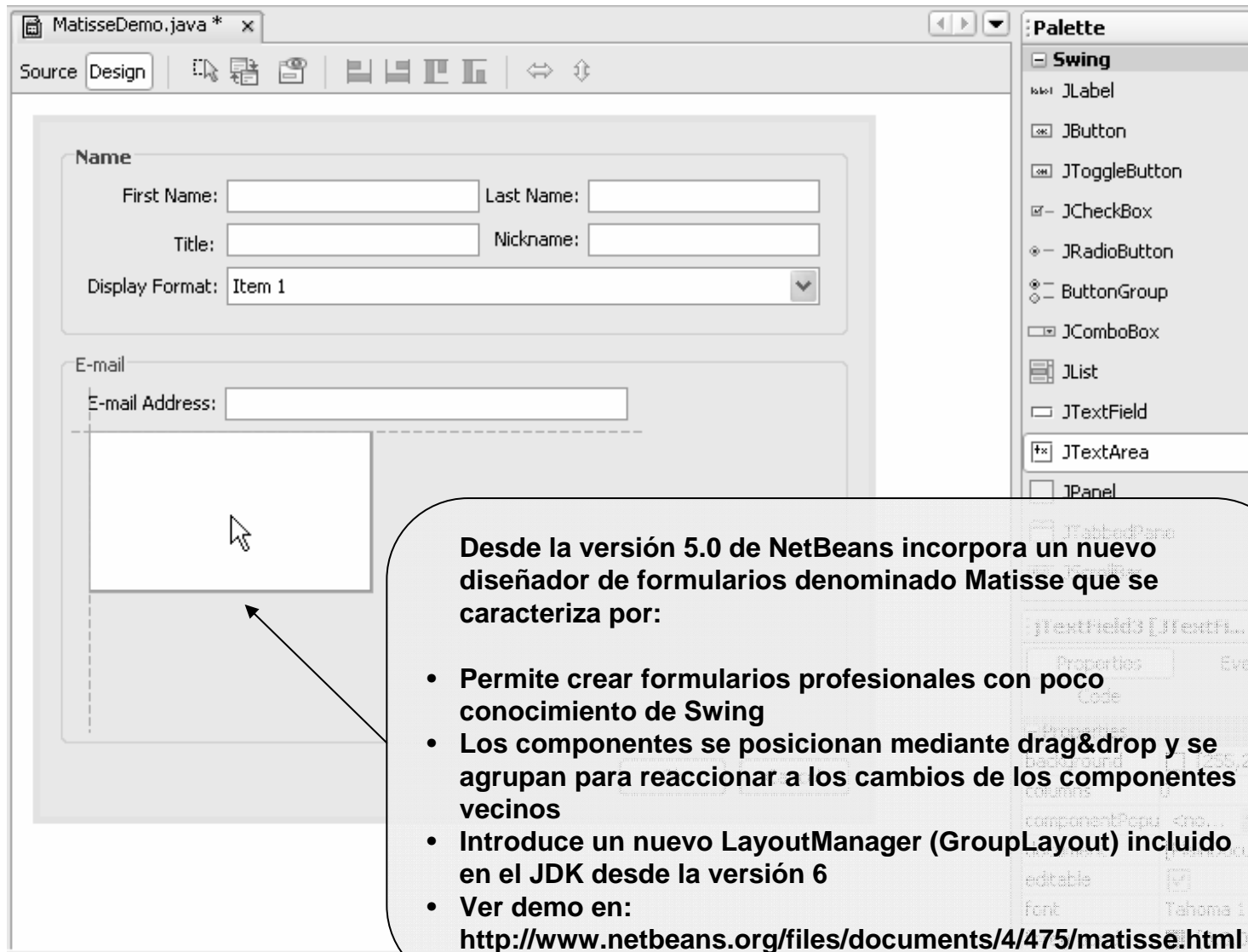
Como vemos NetBeans utiliza la estrategia de clases internas anónimas con rellamadas a métodos privados para gestionar los eventos

55:51 INS



Interfaces de usuario en Java

Diseñador NetBeans



Desde la versión 5.0 de NetBeans incorpora un nuevo diseñador de formularios denominado Matisse que se caracteriza por:

- Permite crear formularios profesionales con poco conocimiento de Swing
- Los componentes se posicionan mediante drag&drop y se agrupan para reaccionar a los cambios de los componentes vecinos
- Introduce un nuevo LayoutManager (GroupLayout) incluido en el JDK desde la versión 6
- Ver demo en:
<http://www.netbeans.org/files/documents/4/475/matisse.html>