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. Sintaxis básica de Java
2. Compilación en línea
3. La herramienta NetBeans

Índice

1. **Sintaxis básica de Java**
   - **Comentarios**
   - **Tipos de datos**
   - **Operadores**
   - **Estructuras de control**
   - **“Hola Mundo” tradicional**
   - **“Hola Mundo” orientado a objetos**

Lenguaje Java
Sintaxis básica de Java

- **Comentarios “normales”**
  - **// De una sola línea**
  - **/* De más de una línea */**
- **Comentarios de documentación**
  - **Construcción básica**

Opcional

```
/**
* Comentario de documentación
*/
```

  - **Tokens de javadoc**
    - **@see, @version, @author, @param, @return, @exception, @deprecated**
  - **Empleo de etiquetas HTML**

```
/**
@param <b>args</b> cadena de
   argumentos
*/
```

```
import java.util.*;

/**
 * DateDoc.java: Ej. de documentacion.
 * Presenta la fecha y hora del Sistema
 * @author Eduardo Mosqueira
 * @version 1.0
 */
public class DateDoc
{

 /**
  * Método principal de la aplicacion
  * @param <b>args</b> cadena de argumentos
  * @return No devuelve ningun valor
  * @exception ninguna excepcion
  */
  public static void main (String args[])
  {
    System.out.println (new Date());
  }
}
```

Lenguaje Java

Sintaxis básica de Java

- **Tipos de datos**

Tipo	Características	Utilización
byte	8 bits – complemento a 2	
short	16 bits – complemento a 2	
int	32 bits – complemento a 2	
long	64 bits – complemento a 2	
float	32 bits – IEEE 754	
double	64 bits – IEEE 754	
boolean	valores true o false	
char	16 bits - carácter	
String	Se trata como una clase No se pueden cambiar los caracteres que lo forman	String s; s = “Cadena” s = new String(“Cadena”);
Arrays	Se distinguen tres operaciones: definir el array, dar tamaño al array y asignar elementos al array Realiza comprobaciones exhaustivas del correcto funcionamiento del array Comienzan por cero	**Definición:** Int[] lista; **Dar tamaño:** lista = new int[10]; **Asignación:** lista[0] = 6; **Asignación objetos:** listaS[0]=new String(“c”); **Matrices:** int tabla[][] = new int[4][5]; **Definición y asignación:** int [] Lista ={ 1, 2, 3 };

Los tipos byte, short, int, long, float, double, boolean y char son tipos primitivos, es decir, no son clases. Para poder trabajar con los tipos primitivos como clases existen clases contenedoras que se definen con nombres similares pero empezando con mayúsculas (ej. Integer, Character, Boolean, Double, etc.)

Lenguaje Java
Sintaxis básica de Java

- **Operadores**

Tipo	Operador	Descripción
Aritméticos	+, -, *, / y %	Suma, Resta, Multiplicación, División y Resto
	++, --	Incremento, Decremento
Relacionales	<, <=	Menor que, Menor o igual que
	>, >=	Mayor que, Mayor o igual que
	==, !=	Distinto de
Condicionales	&	AND
	|	OR
	&&	AND condicional
	||	OR condicional
	!	NOT
Manejo de bits	>>, <<	Desplaz. a la derecha, Desplaz. a la izquierda
	>>>	Desplazamiento sin signo
	&, |, ^	AND, OR, XOR
	~	Complemento a
Operadores de asignación	=	Operador de asignación básico
	Operador= (+=, -=, *=, /=, %=, &=, |=, ^=, <<=, >>=, >>>=)	A operador= B equivale a A = A operador B
Conversión de tipos	(tipo) variable o expresion	
Operador ternario	Expresion ? sentencia1 : sentencia2	if expresion then sentencia1 else sentencia2

Lenguaje Java
Sintaxis básica de Java

- **Estructuras de control**

Tipo	Estructura	Sintaxis
Bifurcaciones	if ... else	if (condicion) { ... } else { ... }

	switch	switch (variable)
		{
		  case n1: ... break;
		  case n2: ... break;
		  default: ... break;
		}

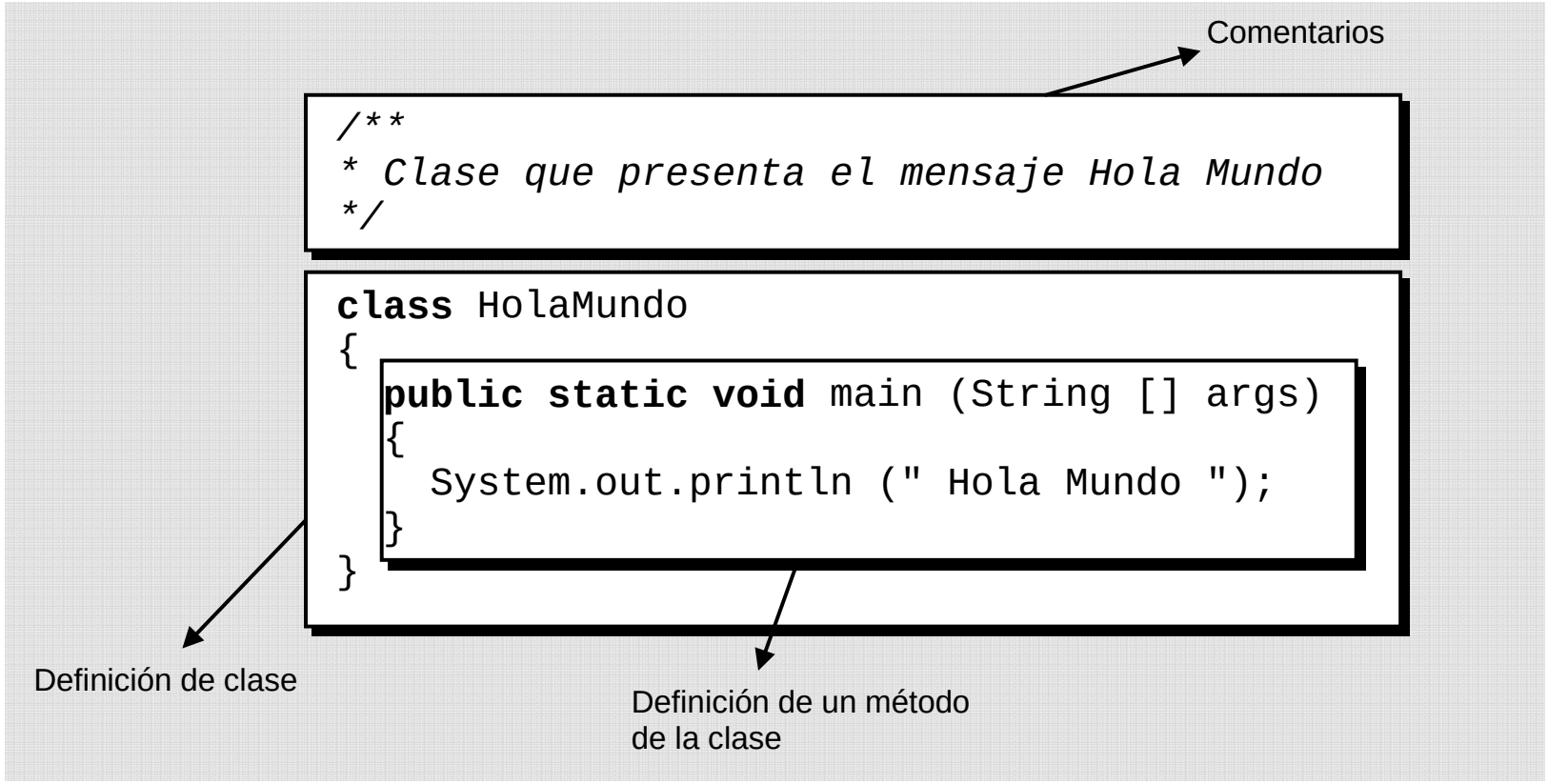
Bucles	for	for (var=min; var<max; inc) { ... }
		
		for (Object o : miArray)
		  { System.out.println(o); }

	while	while (condicion) {... }
	do ... while	do { ... } while (condicion)
Manejo de excepciones	try ... catch ... finally	try { ... } catch (excepción) { ... } finally { ... }
	throw	throw excepción;
	throws	type NombreMetodo (argumentos) throws excepciones { ... }
Control del flujo	break	break [etiqueta];
	continue	continue [etiqueta];
	return	return expresión;

“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 ");
  }
}

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

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

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

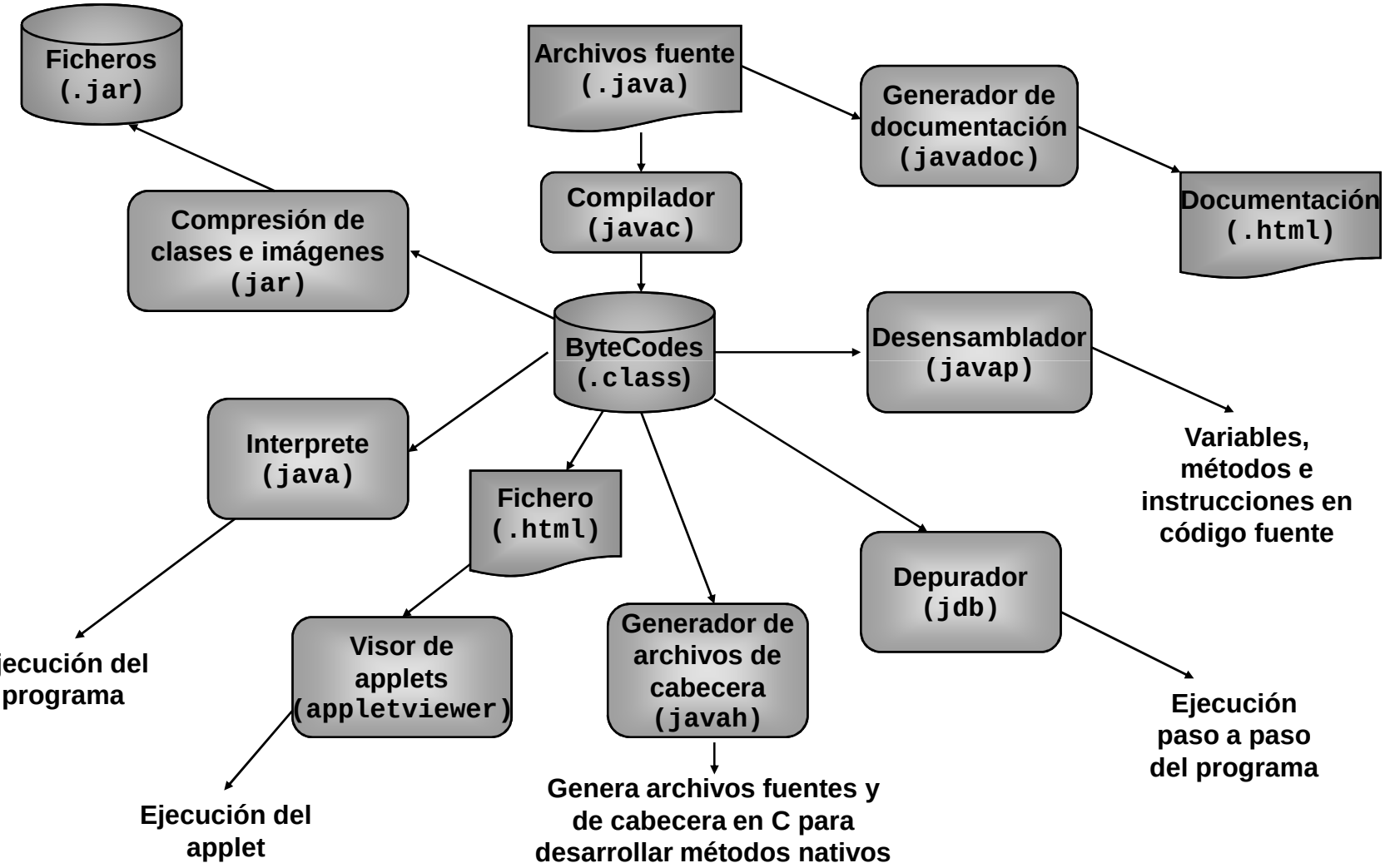
Llamamos al método de instancia imprimeHola

Índice

2. Compilación en línea

- Entorno de desarrollo
- Compilación simple
- Compilación compleja
- Compilación con ant

Lenguaje Java
Entorno de desarrollo



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 HolaMundoOO.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”**
  - **En el directorio “build” se crea una estructura de directorios similar a la existente en el directorio src**

Compilación en línea
Compilación compleja

- **Clases HolaMundo**

```
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());
  }
}
```

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

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

- **Clase Librería**

```
package utilidades;

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

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.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 en línea
Compilación compleja

- **Solución 1: archivos .bat o scripts Linux**
  - **Solución sencilla pero poco portable**
  - **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 tambien presenta problemas a la hora de llevar un fichero make a distintas plataformas**
  - **No tiene en cuenta las particularidades de Java (CLASSPATH)**
- **Solución 3: ficheros Ant**
  - **Ant (Another Neat Tool) de Apache (http://ant.apache.org)**
  - **Desarrollado en Java → multiplataforma**
  - **Adaptado a Java y fácil de extender (las tareas son clases)**
  - **Los ficheros de configuración están en XML, un formato popular para el cual existen múltiples herramientas**

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

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

  <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"/>

  <path id="compile.classpath">
    <pathelement location="${Libreria.jar}" />
  </path> <path id="run.classpath">

  <path refid="compile.classpath" />
    <pathelement location="${build.dir}" />
  </path>

  <target name="init">
    <mkdir dir="${build.dir}"/>
  </target>

  <target name="clean">
    <delete dir="${build.dir}" />
  </target>

  <target name="compile" depends="init">
    <javac srcdir="${src.dir}" destdir="${build.dir}" classpathref="compile.classpath"/>
  </target>

  <target name="run" depends="compile">
    <java classname="poo.holamundo.HolaMundo" classpathref="run.classpath"/>
  </target>
</project>
```

Nombre del proyecto con el target por defecto y el directorio base

Propiedades globales que permiten cambiar fácilmente distintos aspectos de la generación del código (p. ej. los directorios)

Definición de los classpaths para la compilación y la ejecución

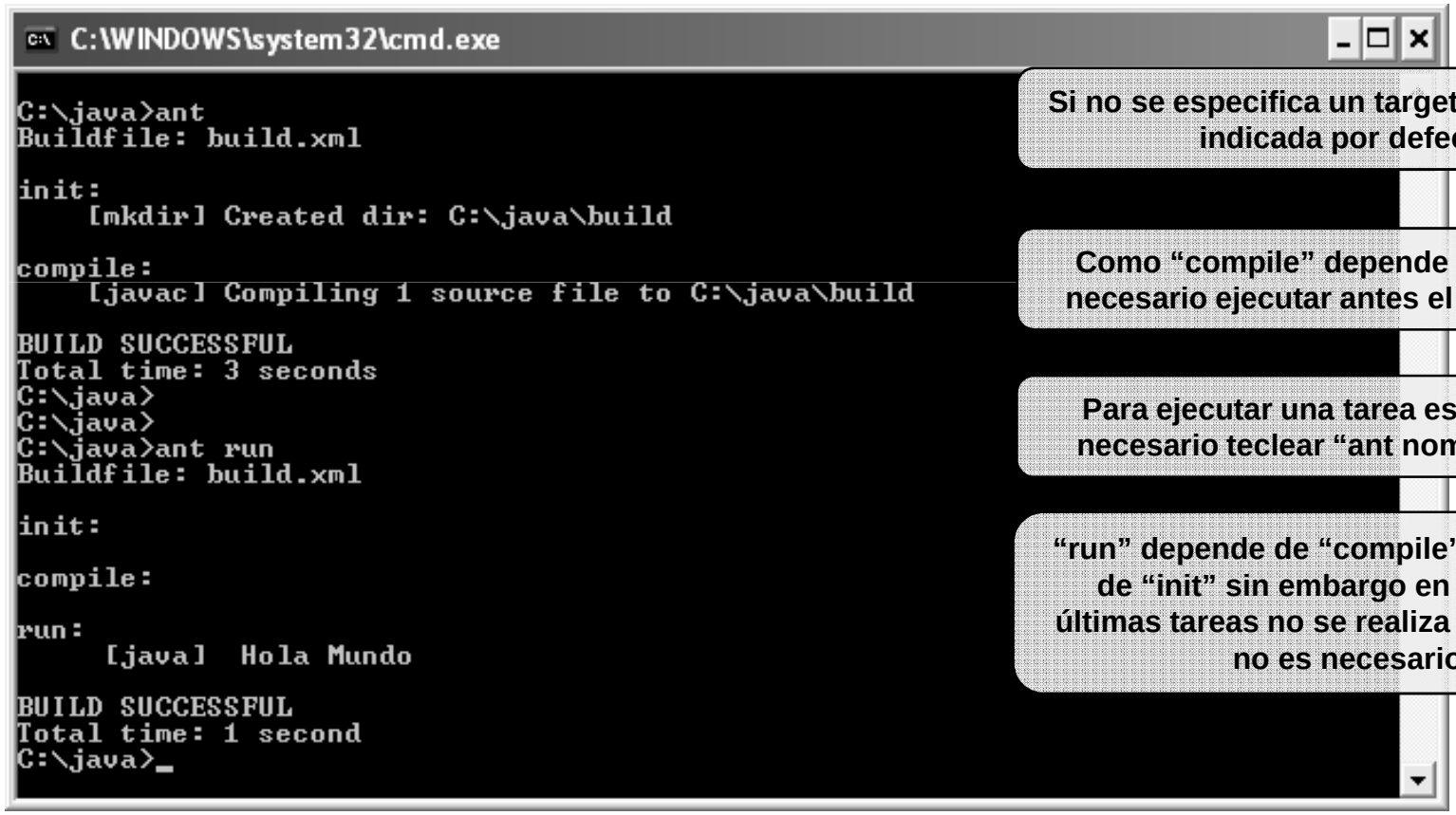
La tarea init crea en directorio build si no existe

La tarea clean elimina el directorio build

La tarea compile depende de init y compila todos los fuentes que están en ${scr.dir} enviando los ficheros .class al ${build.dir}

La tarea run depende de compile ejecuta el fichero HolaMundo

Compilación en línea
Compilación compleja



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
- Edición
- Compilación
- Ejecución
- Depuración
- Ejemplo

La Herramienta NetBeans

Introducción

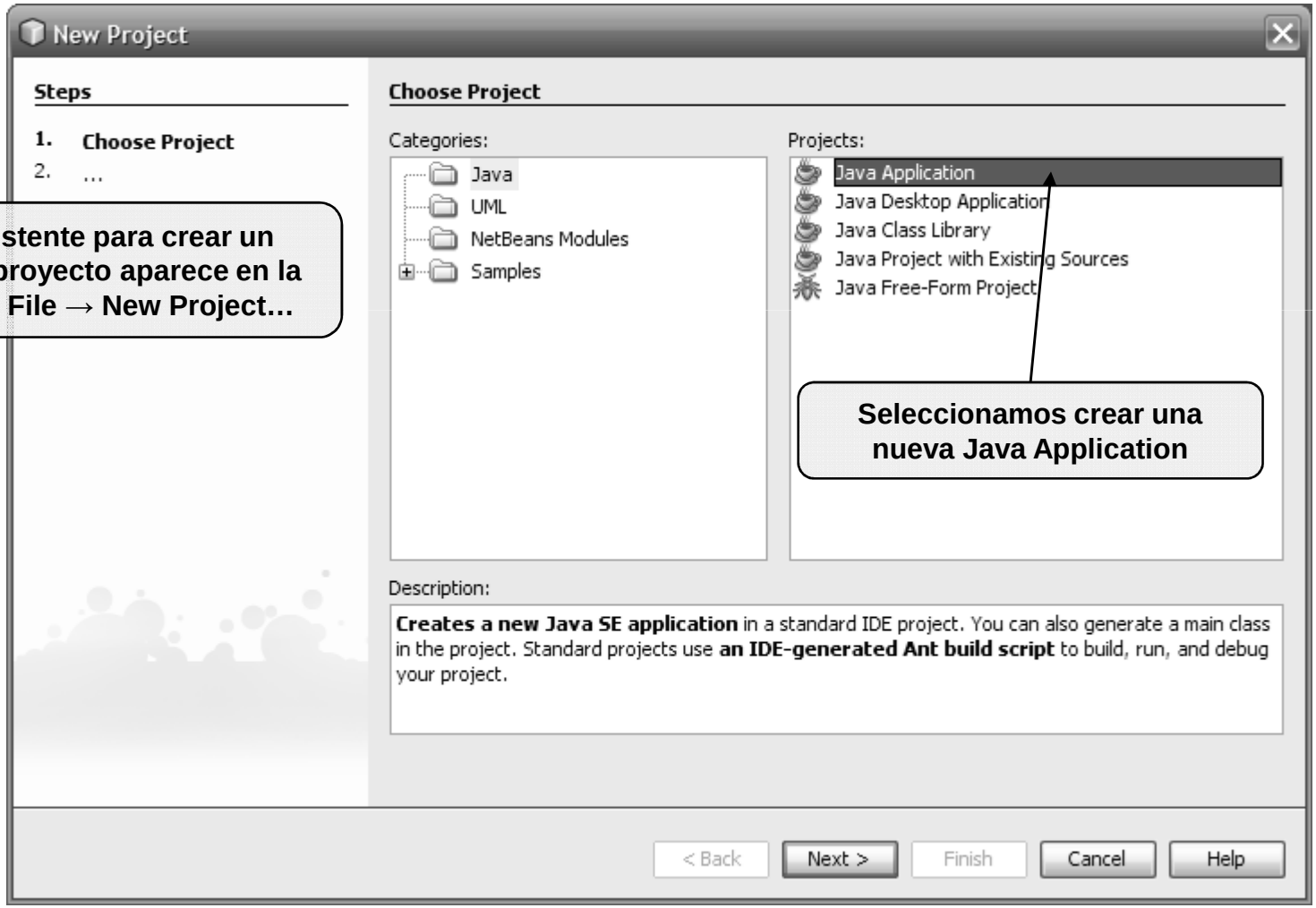
- IDE OpenSource para el desarrollo de código Java mantenido por Sun (http://www.netbeans.org)
- Muy completo permitiendo el desarrollo en las plataformas Micro, Standard y Enterprise. No tiene nada que envidiar a otros IDEs comerciales
- El IDE Eclipse le ha robado una buena cuota de mercado basandose 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



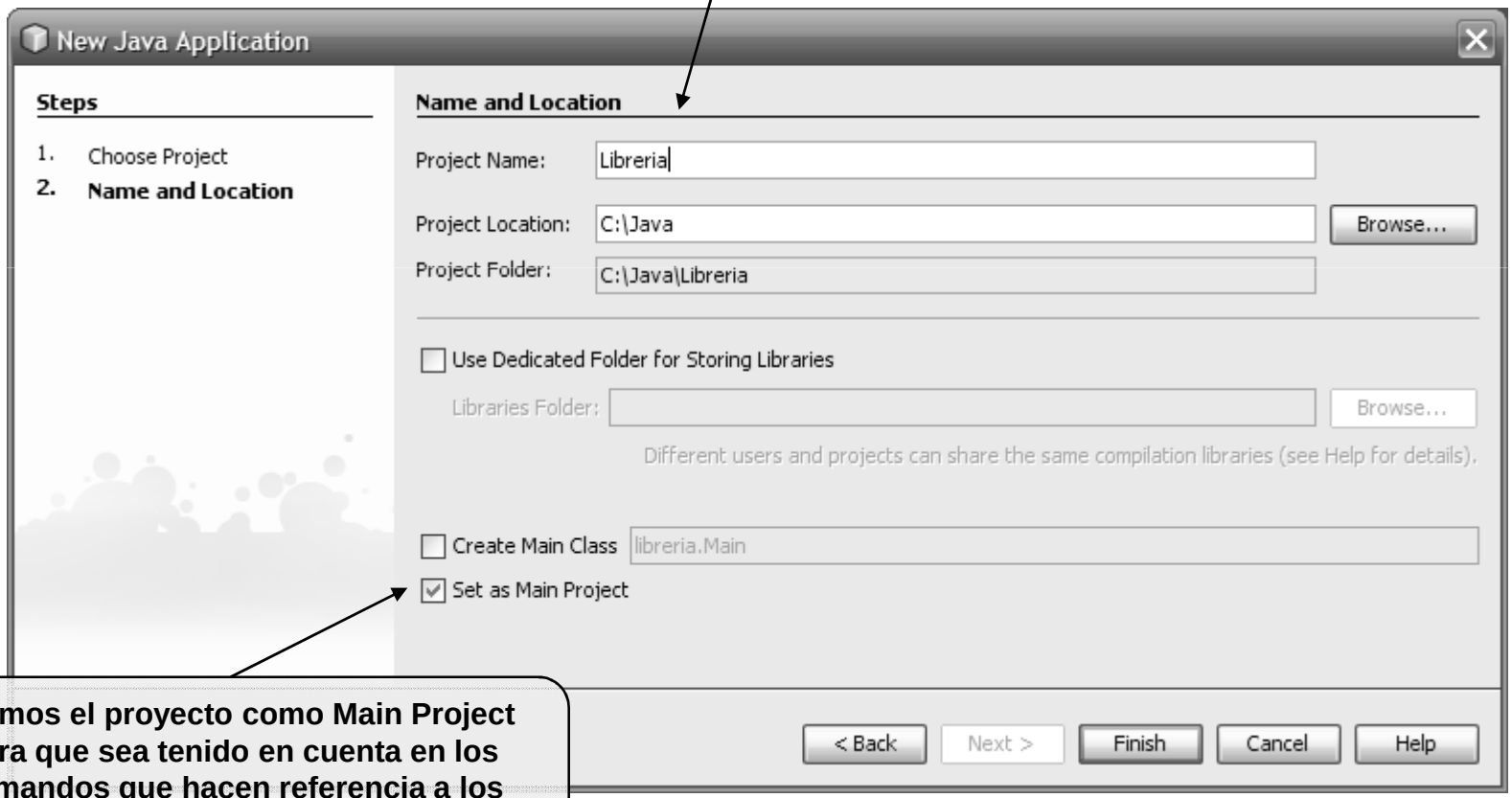
El asistente para crear un nuevo proyecto aparece en la opción File → New Project…

Seleccionamos crear una nueva Java Application

La Herramienta NetBeans

Proyectos

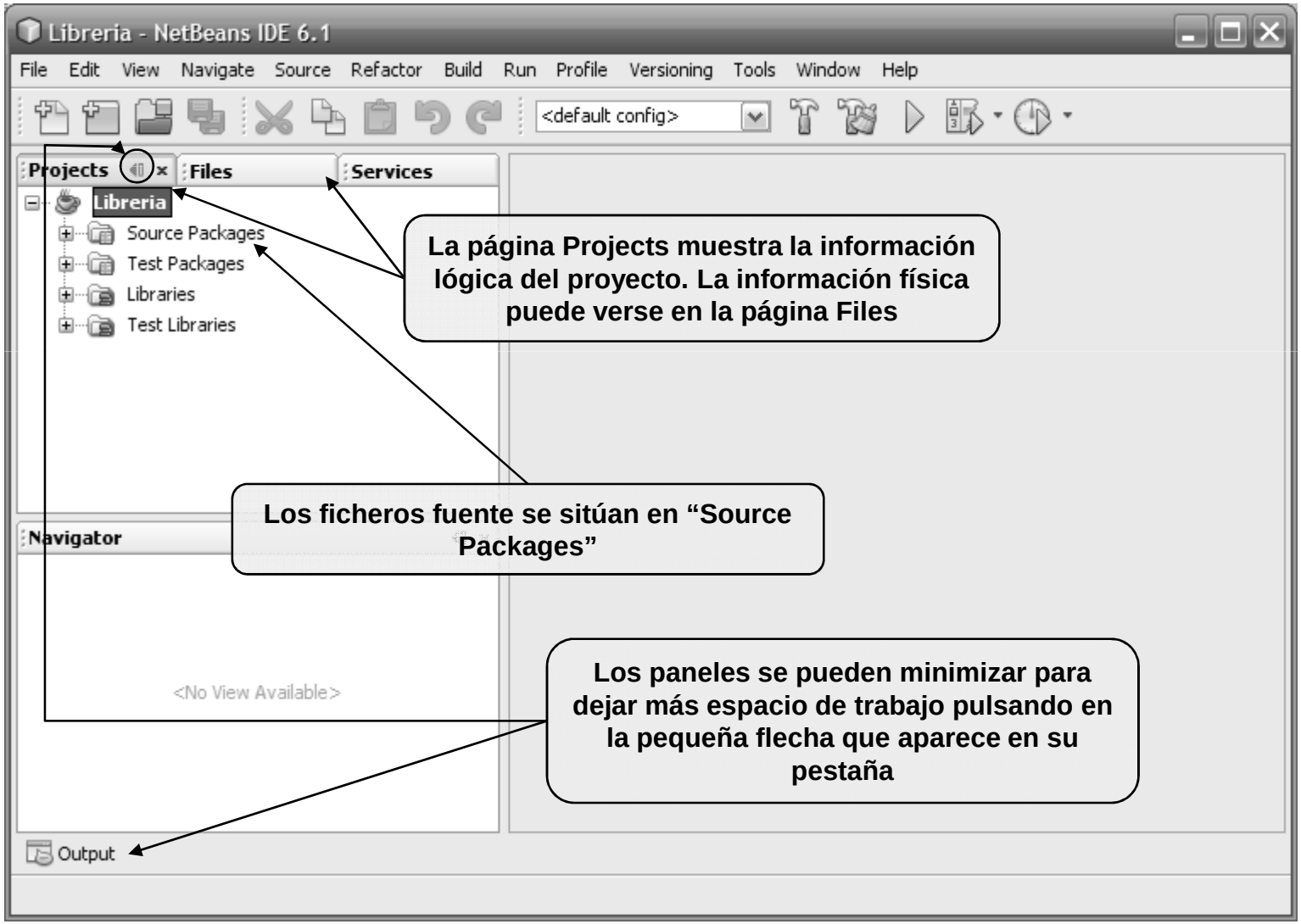
Elegimos la localización del proyecto y su nombre



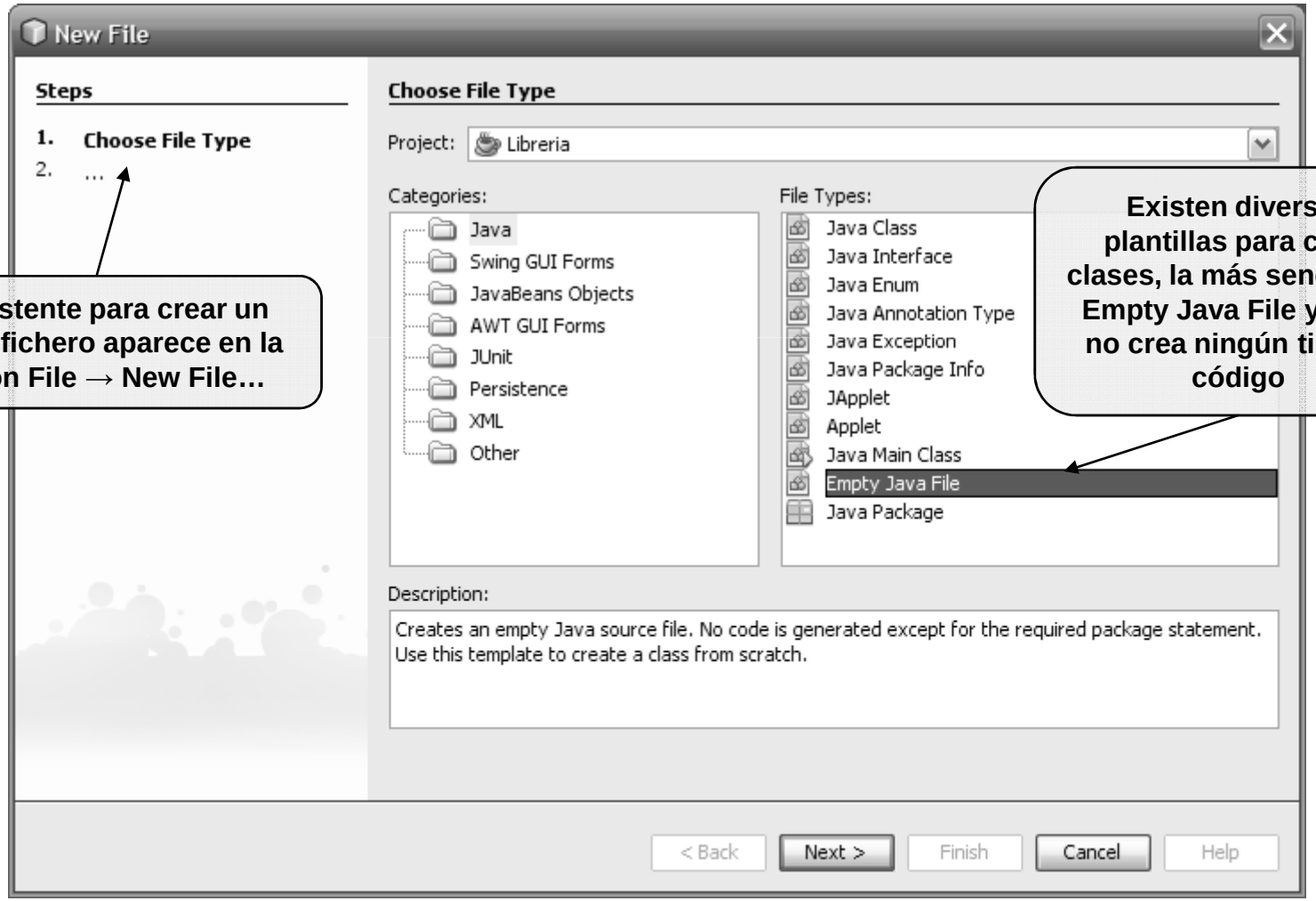
Fijamos el proyecto como Main Project para que sea tenido en cuenta en los comandos que hacen referencia a los proyectos (compilar, ejecutar, etc.)

La Herramienta NetBeans

Proyectos



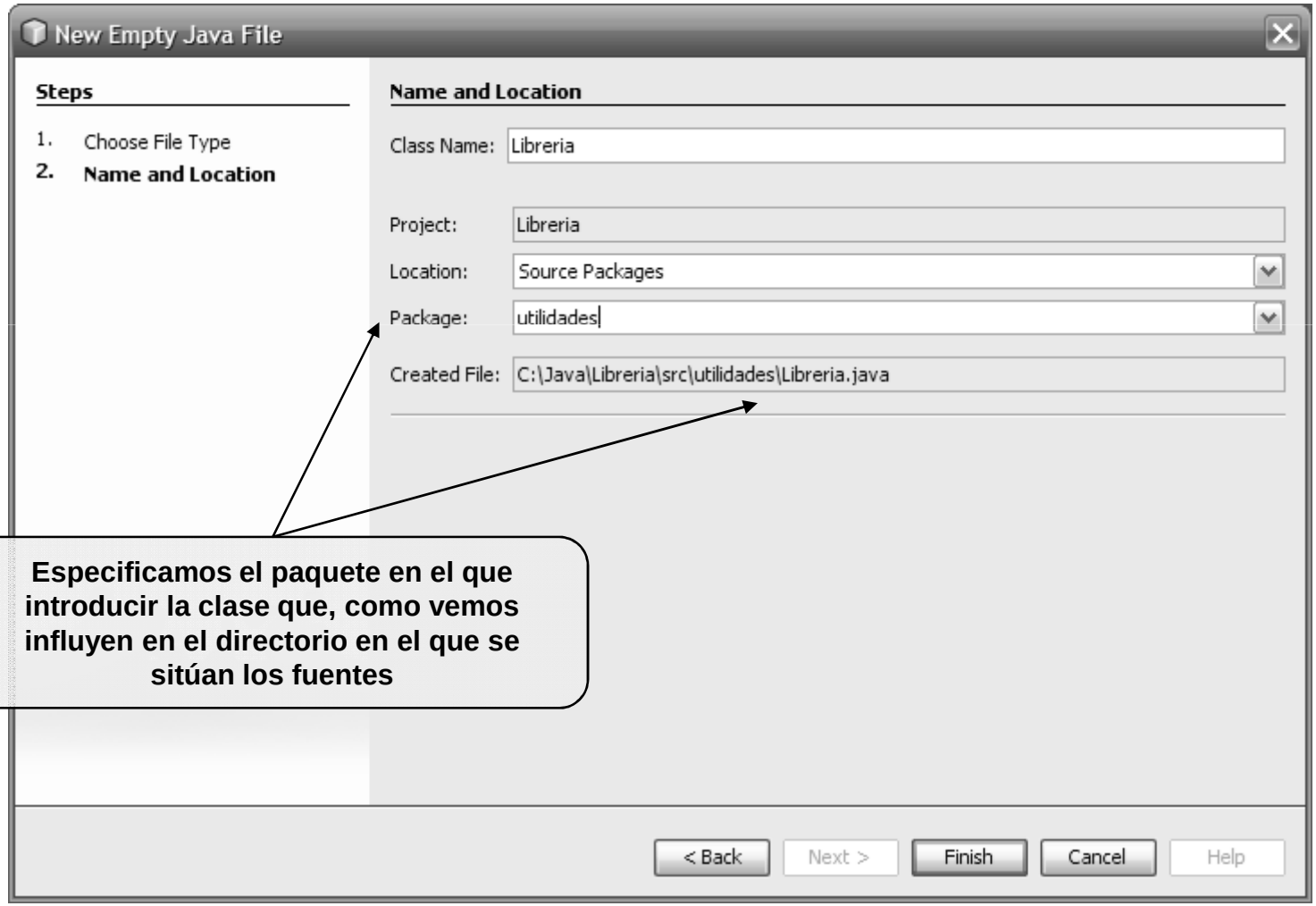
La Herramienta NetBeans
Edición



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

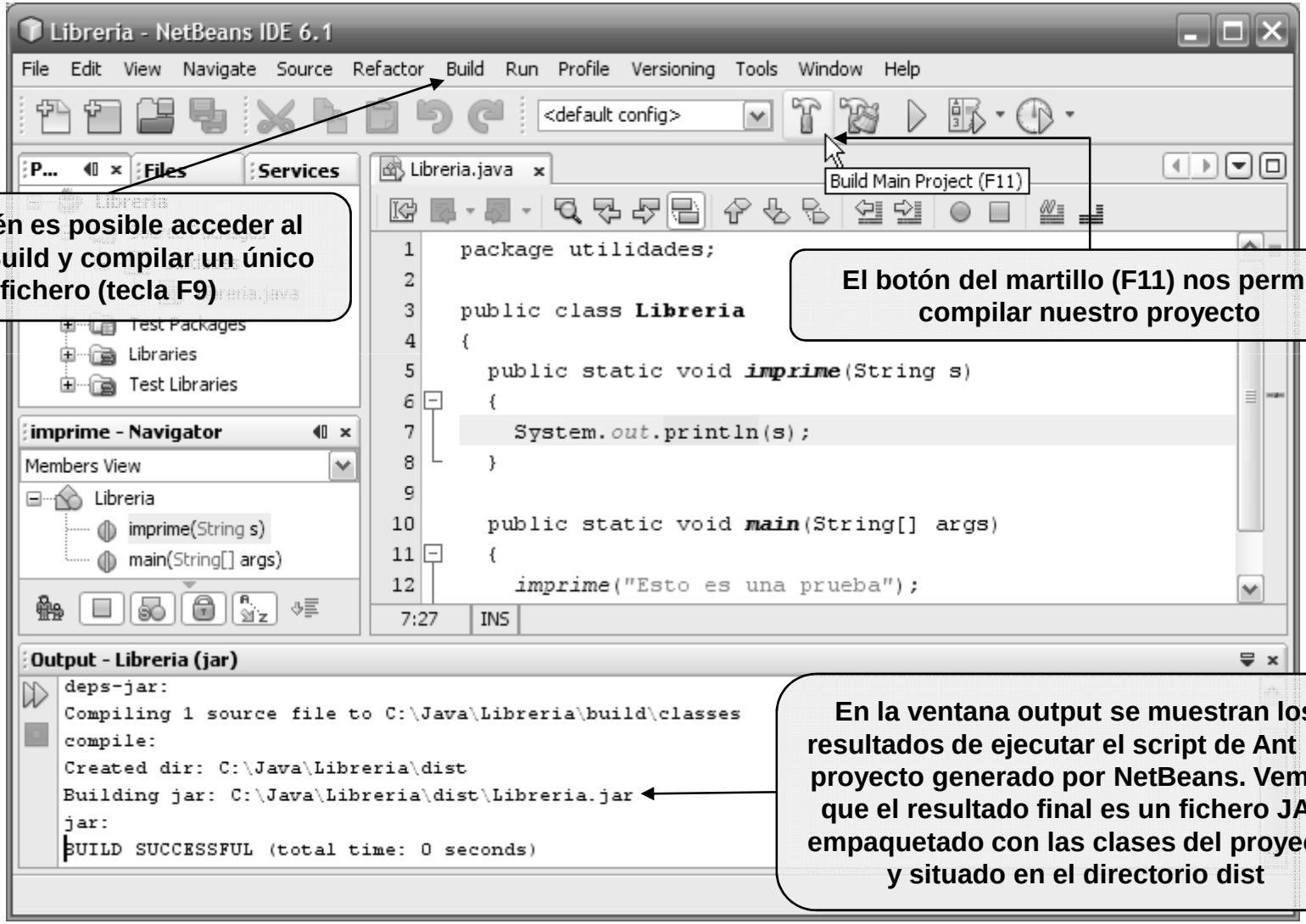
Existen diversas plantillas para crear clases, la más sencilla es Empty Java File ya que no crea ningún tipo de código

La Herramienta NetBeans
Edición



La Herramienta NetBeans
Edición

La Herramienta NetBeans
Compilación

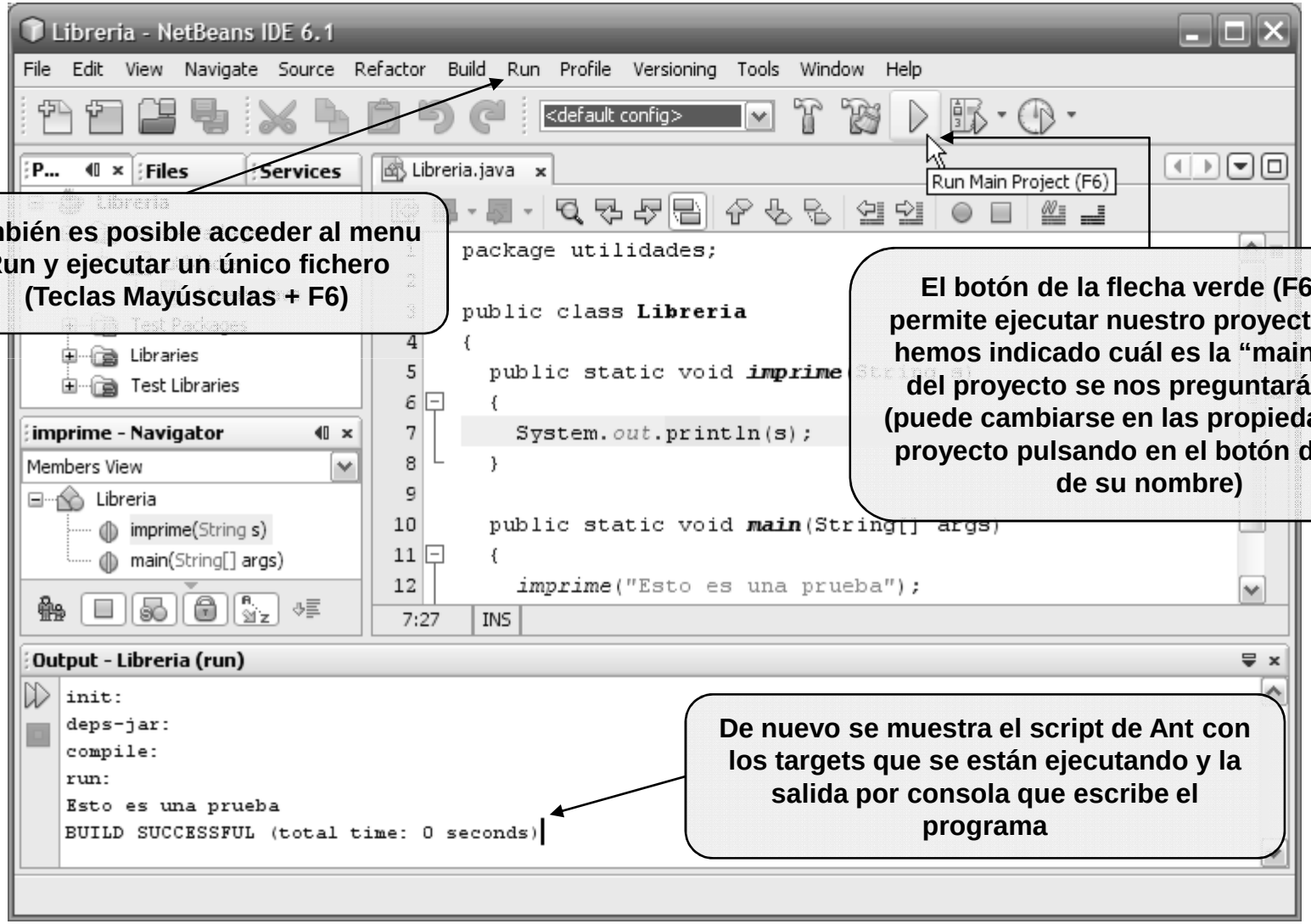


También es posible acceder al menu Build y compilar un único fichero (tecla F9)

El botón del martillo (F11) nos permite compilar nuestro proyecto

En la ventana output se muestran los resultados de ejecutar el script de Ant del proyecto generado por NetBeans. Vemos que el resultado final es un fichero JAR empaquetado con las clases del proyecto y situado en el directorio dist

La Herramienta NetBeans
Ejecución



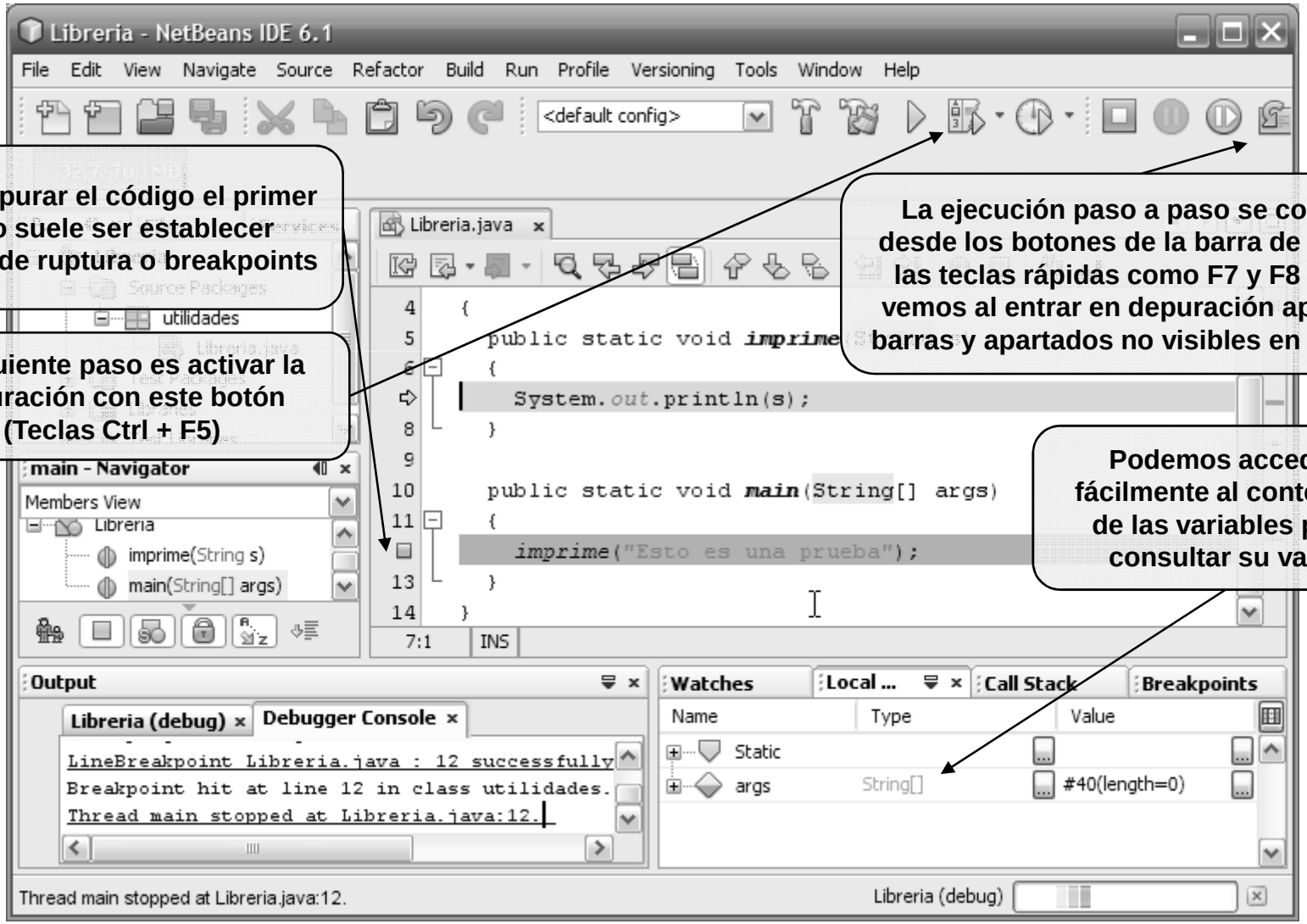
También es posible acceder al menu Run y ejecutar un único fichero (Teclas Mayúsculas + F6)

El botón de la flecha verde (F6) nos permite ejecutar nuestro proyecto. Si no hemos indicado cuál es la "main class" del proyecto se nos preguntará ahora (puede cambiarse en las propiedades del proyecto pulsando en el botón derecho de su nombre)

De nuevo se muestra el script de Ant con los targets que se están ejecutando y la salida por consola que escribe el programa

La Herramienta NetBeans

Depuración



Para depurar el código el primer paso suele ser establecer puntos de ruptura o breakpoints

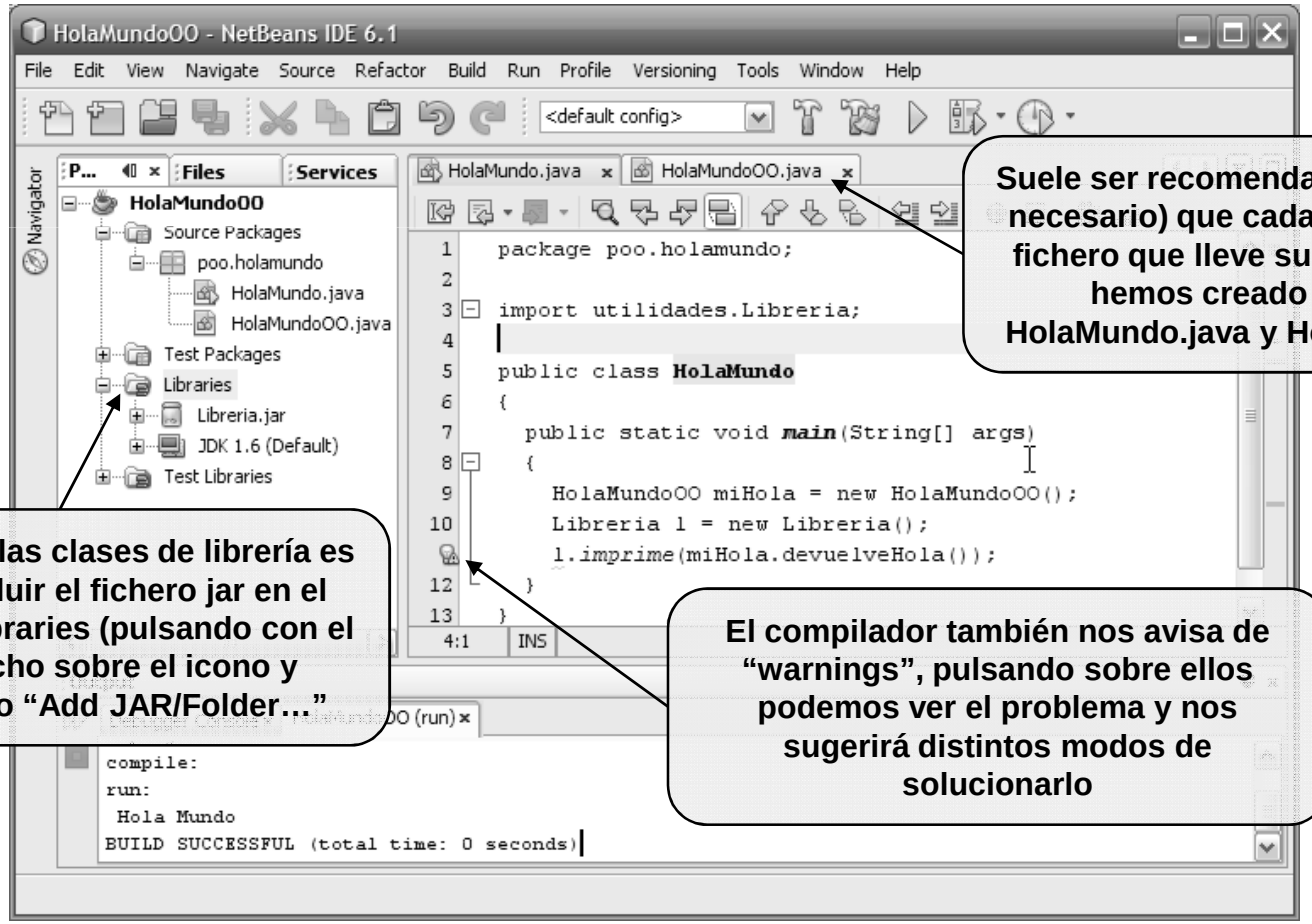
El siguiente paso es activar la depuración con este botón (Teclas Ctrl + F5)

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)

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

La Herramienta NetBeans
Ejercicio

- **Crea un proyecto para el ejemplo HolaMundoOO que se muestra en la transparencia 17 y que hace uso de la clase Librería**



Suele ser recomendable (aunque no es necesario) que cada clase vaya en un fichero que lleve su nombre. Por eso hemos creado los ficheros HolaMundo.java y HolaMundoOO.java

Para acceder a las clases de librería es necesario incluir el fichero jar en el apartado de Libraries (pulsando con el botón derecho sobre el icono y seleccionando "Add JAR/Folder…"

El compilador también nos avisa de "warnings", pulsando sobre ellos podemos ver el problema y nos sugerirá distintos modos de solucionarlo