APÉNDICE

E

Ejecutar Java fuera del entorno BlueJ

A lo largo de este libro hemos usado BlueJ para desarrollar y ejecutar nuestras aplicaciones Java. Hay una buena razón para esto: BlueJ nos ofrece algunas herramientas para que resulten más fáciles algunas de las tareas de desarrollo. En particular, nos permite ejecutar fácilmente métodos individuales de clases y de objetos, lo que resulta muy útil si queremos probar rápidamente un fragmento de código.

Dividimos la discusión sobre cómo trabajar fuera del entorno BlueJ en dos categorías: ejecutar una aplicación y desarrollarla fuera del entorno BlueJ.

E.1 Ejecutar fuera del entorno BlueJ

Generalmente, cuando se entregan las aplicaciones a los usuarios finales, son ejecutadas de diferentes maneras. Las aplicaciones tienen un solo punto de comienzo que define el lugar en que empieza la ejecución cuando un usuario inicia la aplicación.

El mecanismo exacto que se usa para iniciar una aplicación depende del sistema operativo; generalmente, se hace doble clic sobre el icono de la aplicación o se ingresa el nombre de la misma en una línea de comando. Luego, el sistema operativo necesita saber qué método o qué clase debe invocar para ejecutar el programa completo.

En Java, este problema se resuelve usando una convención: cuando se inicia un programa Java, el nombre de la clase se especifica como un parámetro del comando de inicio y el nombre del método es siempre el mismo, el nombre de este método es «main». Por ejemplo, considere el siguiente comando ingresado en una línea de comando, como si fuera un comando de Windows o de una terminal Unix:

```
java Juego
```

El comando `java` inicia la máquina virtual de Java, que forma parte del kit de desarrollo de Java (SDK) y que debe estar instalado en su sistema. `Juego` es el nombre de la clase que queremos iniciar.

Luego, el sistema Java buscará un método en la clase `Juego` cuya signatura coincida exactamente con la siguiente:

```
public static void main(String[] args)
```

El método debe ser público para que pueda ser invocado desde el exterior de la clase. Debe ser estático porque no existe ningún objeto cuando se inicia el programa; inicialmente, tenemos sólo clases, motivo por el cual sólo podemos invocar métodos estáticos. Este método estático crea el primer objeto. El tipo de retorno es `void` ya que este método no retorna ningún valor. Aunque el nombre «`main`» fue seleccionado arbitrariamente por los desarrolladores de Java, es fijo: el método debe tener siempre este

nombre. (La elección de «main» como nombre del método inicial en realidad proviene del lenguaje C, del que Java hereda gran parte de su sintaxis.)

El parámetro es un arreglo de String, que permite a los usuarios pasar argumentos adicionales. En nuestro ejemplo, el valor del parámetro args será un arreglo de longitud cero. Sin embargo, la línea de comandos que inicia el programa puede definir argumentos:

```
java Juego 2 Fred
```

En esta línea de comando, cada palabra ubicada a continuación del nombre de la clase será leído como un String independiente y pasado al método main como un elemento del arreglo de String. En este caso, el arreglo args contendrá dos elementos que son las cadenas «2» y «Fred». Los parámetros en la línea de comandos no son muy usados en Java.

En teoría, el cuerpo del método main puede contener el número de sentencias que se deseen. Sin embargo, un buen estilo indica que el método main debiera mantenerse lo más corto posible; específicamente, no debiera contener nada que forme parte de la lógica de la aplicación.

En general, el método main debe hacer exactamente lo que se hizo interactivamente para iniciar la misma aplicación en BlueJ. Por ejemplo, si para iniciar la aplicación en BlueJ se creó un objeto de la clase Juego y se invocó el método de nombre start, en el método main de la clase Juego deberían agregarse las siguientes sentencias:

```
public static void main (String[] args)
{
  Juego juego = new Juego();
  juego.start();
}
```

Ahora, al ejecutar el método main se imitará la invocación interactiva del juego.

Los proyectos Java se guardan generalmente en un directorio independiente para cada uno y todas las clases del proyecto se ubican dentro de este directorio. Cuando se ejecute el comando para iniciar Java y ejecutar su aplicación, se debe asegurar de que el directorio del proyecto sea el directorio activo en la terminal de comandos, lo que asegura que se encontrarán las clases que se usan.

Si no puede encontrar una clase específica, la máquina virtual de Java generará un mensaje de error similar a este:

```
Exception in thread "main" java.lang.NoClassDefFoundError: Juego
```

Si ve un mensaje como éste, asegúrese de que escribió correctamente el nombre de la clase y de que el directorio actual realmente contenga esta clase. La clase se guarda en un archivo de extensión ".class": por ejemplo, el código de la clase Juego se almacena en un archivo de nombre Juego.class.

Si encuentra la clase pero ésta no contiene un método main (o el método main no posee la signatura correcta) verá un mensaje similar a este:

```
Exception in thread "main" java.lang.NoSuchMethodError: main
```

En este caso, asegúrese de que la clase que quiere ejecutar tenga el método main correcto.

E.2 Crear archivos ejecutables .jar

Los proyectos Java se almacenan como una colección de archivos en un directorio (o carpeta). A continuación, hablaremos brevemente sobre los diferentes tipos de archivo.

Generalmente, para distribuir aplicaciones a otros usuarios es más fácil si toda la aplicación se guarda en un único archivo; el mecanismo de Java que realiza esto tiene el formato de archivo Java («.jar»). Todos los archivos de una aplicación se pueden reunir en un único archivo y aun así podrán ser ejecutados. (Si está familiarizado con el formato de compresión «zip», sería interesante saber que, de hecho, el formato es el mismo. Los archivos jar pueden abrirse mediante programas zip y viceversa.)

Para crear un archivo .jar ejecutable es necesario especificar la clase principal en algún lugar. (Recuerde: el método que se ejecuta siempre es el `main`, pero necesitamos especificar la clase que lo contiene.) Esta especificación se hace incluyendo un archivo de texto en el archivo .jar (el archivo explícito) con la información necesaria. Afortunadamente, BlueJ se ocupa por su propia cuenta de esta tarea.

Para crear un archivo ejecutable .jar en BlueJ use la función *Project – Export* y especifique la clase que contiene el método `main` en la caja de diálogo que aparece. (Debe escribir un método `main` exactamente igual al descrito anteriormente.)

Para ver detalles sobre esta función, lea el Tutorial de BlueJ al que puede acceder mediante el menú *Help-Tutorial* de BlueJ o bien visitando el sitio web de BlueJ.

Una vez que se creó el archivo ejecutable .jar, se puede ejecutar haciendo doble clic sobre él. La computadora que ejecuta este archivo .jar debe tener instalado el JDK (Java Development Kit) o el JRE (Java Runtime Environment) y asociado con archivos .jar.

E.3 Desarrollar fuera del entorno BlueJ

Si no quiere solamente ejecutar programas, sino que también quiere desarrollarlos fuera del entorno BlueJ, necesitará editar y compilar las clases. El código de una clase se almacena en un archivo de extensión «`.java`»; por ejemplo, la clase `Juego` se almacena en un archivo de nombre `Juego.java`. Los archivos fuente pueden editarse con cualquier editor de textos. Existen muchos editores de textos libres o muy baratos. Algunos, como el *Notepad* o el *WordPad* se distribuyen con Windows, pero si en realidad quiere usar un editor para hacer algo más que una prueba rápida, querrá obtener uno mejor. Sin embargo, sea cuidadoso con los procesadores de texto: generalmente los procesadores de texto no graban en formato de texto plano y Java no podrá leerlos.

Los archivos fuente pueden compilarse desde una línea de comando usando el compilador Java que se incluye en el JDK y que se invoca mediante el comando `javac`. Para compilar un archivo fuente de nombre `Juego.java` use el comando

```
javac Juego.java
```

Este comando compilará la clase `Juego` y cualquier otra clase que dependa de ella; creará un archivo denominado `Juego.class` que contiene el código que puede ser ejecutado mediante la máquina virtual de Java. Para ejecutar este archivo use el comando

```
java Juego
```

Observe que este comando no incluye la extensión del archivo «.class».