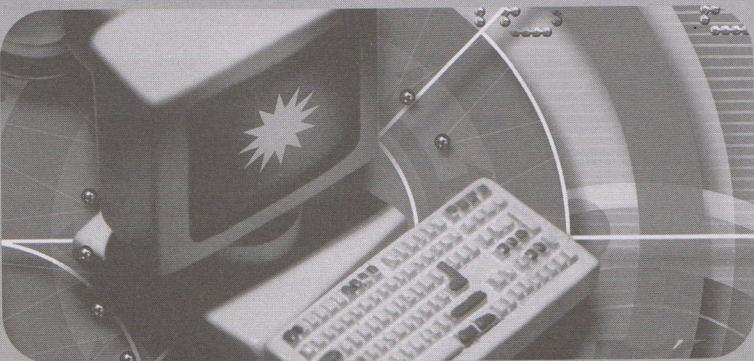


## CAPÍTULO 2



# Los primeros programas

En este capítulo conoceremos:

- Cómo crear, compilar y ejecutar programas de Java.
- El uso de un entorno de desarrollo integrado.
- Los conceptos de clases, objetos y métodos.
- Cómo mostrar un cuadro de diálogo.
- Cómo colocar texto en un campo de texto.

### ● Introducción

Para aprender a programar en Java necesitará una computadora con herramientas de Java, y por fortuna el lenguaje Java está diseñado para ejecutarse en cualquier sistema operativo. En la actualidad los sistemas operativos más utilizados son los sistemas Windows de Microsoft en las PC, aunque también se utilizan los sistemas operativos Linux de GNU en estos equipos, y OS X en las Mac de Apple. Java se puede ejecutar en cualquiera de estos sistemas. Éste es un beneficio importante, pero significa que hay una variación en las instrucciones detalladas para usar Java. Aquí le proporcionaremos la información general solamente. En el apéndice I se incluyen más detalles sobre cómo obtener sistemas de Java sin costo.

Una vez instalado Java, tendrá que atravesar cuatro etapas para generar un programa:

- Crear un nuevo archivo o proyecto.
- Introducir/modificar el programa con un editor.
- Compilar el programa.
- Ejecutar el programa.

## ● Entornos de desarrollo integrados

Hay dos formas principales de crear y ejecutar sus programas. En primer lugar, podríamos elegir un entorno de desarrollo integrado (IDE). Éste es un paquete de software diseñado para ayudar en todo el proceso de crear y ejecutar un programa de Java. Si utiliza un IDE, de todas formas es conveniente que se familiarice con los conceptos de archivos, edición, compilación y ejecución, como veremos más adelante.

Hay varios IDE en la actualidad. Uno de los más populares, además de gratuito, es Eclipse (que también está escrito en Java). En el apéndice I encontrará más información al respecto. Como alternativa puede usar un editor de texto (en vez de un procesador de palabras simple) para crear sus programas. Algunos de los más poderosos (como Textpad) se pueden configurar para enlazar software de Java, para que usted pueda iniciar los procesos de compilación y ejecución haciendo clic en una opción de menú.

## ● Archivos y carpetas

Los programas que se cargan y ejecutan de manera automática al encender la computadora se conocen en forma colectiva como sistema operativo. Una parte importante de un sistema operativo se encarga del almacenamiento de los archivos; a continuación veremos una breve introducción.

La información que se almacena en un disco de computadora se organiza en forma de archivos, al igual que la información guardada en los archiveros de una oficina se almacena en archivos.

Por lo general, un archivo contiene información relacionada. Por ejemplo:

- Una carta para su mamá.
- Una lista de los estudiantes de un curso específico.
- Una lista de amigos con nombres, direcciones y números telefónicos.

Cada archivo tiene su propio nombre, elegido por la persona que lo creó. Como podría esperarse, es común elegir un nombre que describa con claridad lo que hay en el archivo. El nombre del archivo tiene una *extensión* (una parte al final) que describe el tipo de información que contiene. Por ejemplo, un archivo llamado **carta1** que contiene una carta y se edita por lo general con un procesador de texto podría tener la extensión **.doc** (abreviación de documento), de manera que su nombre completo sería **carta1.doc**. Un archivo que contiene un programa de Java tiene la extensión **.java**, por lo que un nombre de archivo común podría ser **Juego.java**.

Un grupo de archivos relacionados se guarda en una carpeta (algunas veces se le conoce también como directorio). Por lo tanto, en una carpeta específica podría guardar todas las cartas enviadas al banco. En otra carpeta podría almacenar todas las cifras de ventas de un año. En definitiva, es conveniente mantener todos los archivos que se utilizan en un programa de Java en la misma carpeta. Puede asignar un nombre a cada carpeta (por lo general un nombre representativo) que le ayude a encontrarla.

En general, las carpetas también se pueden agrupar en otra carpeta. Por ejemplo, podría tener una carpeta llamada **Toms** que contenga a las carpetas **misprogs** y **cartas**.

Tal vez piense que esto puede continuar de manera indefinida, y así es, podemos establecer carpetas de carpetas *ad infinitum*. Comúnmente su sistema de cómputo contendrá cientos de carpetas y miles de archivos. Algunos de éstos serán tuyos (puede crearlos y después modificarlos) y otros pertenecerán al sistema operativo (*¡no los modifique!*).

Entonces, un archivo es una colección de información con un nombre. Los archivos relacionados se guardan dentro de una carpeta, la cual también tiene un nombre.

Para poder ver las listas de carpetas y los archivos que contienen, debemos utilizar un programa conocido como *Explorador de Windows* en los sistemas operativos Windows de Microsoft. Al hacer clic en una carpeta, ésta muestra los archivos que contiene. Los sistemas Linux de GNU y Mac de Apple tienen herramientas similares.

## PRÁCTICA DE AUTOEVALUACIÓN

- 2.1** (a) ¿Cuál es la diferencia entre una carpeta y un directorio?  
(b) ¿Qué es una carpeta?  
(c) ¿Es posible crear dos carpetas con el mismo nombre?

## ● Creación de un programa de Java

Ya sea que utilice un poderoso IDE o un editor de texto más sencillo, hay varios pasos necesarios para crear un programa de Java.

### Creación de un nuevo archivo

Si utiliza un editor de texto tiene que crear un nuevo archivo, el cual debe tener la extensión `.java`. En un IDE debe crear un proyecto, el cual consta de varios archivos. El único archivo que debe modificar es el `.java`.

### Edición del archivo

Para ello tiene que escribir y modificar el programa (más adelante le mostraremos un programa de ejemplo). Puede utilizar un editor de texto, pero todos los IDEs traen un componente editor de texto. En el editor pasará mucho tiempo, por lo tanto es importante que explore sus herramientas avanzadas, como las que se utilizan para buscar y reemplazar texto.

### Compilación del programa

Un compilador es un programa que convierte otro programa escrito en un lenguaje como Java en un lenguaje que la computadora pueda entender. Por lo tanto, un compilador es como un traductor automático capaz de traducir un lenguaje (de computadora) a otro. Los programas de Java se convierten a código de bytes. El código de bytes no es exactamente el lenguaje que una computadora comprende (código de máquina). Por el contrario, es un lenguaje de máquina idealizado, lo cual quiere decir que su programa de Java se ejecutará en cualquier tipo de computadora. Al ejecutar su programa, el código de bytes es interpretado por un programa conocido como Máquina Virtual de Java (JVM).

Para iniciar el proceso de compilación debemos hacer clic en un botón o seleccionar una opción de menú.

Al compilar su programa, el compilador de Java verifica que éste se apegue a las reglas de programación de Java y, si algo sale mal, muestra los mensajes de error apropiados. También verifica que los programas de cualquier biblioteca que usted utilice se empleen en forma correcta. Es raro (incluso para los programadores experimentados) que un programa se compile correctamente al primer intento, así que no se desanime si recibe algunos mensajes de error. He aquí un ejemplo de un mensaje de error:

```
Hola.java:9: ';' expected
```

Este mensaje indica el nombre del archivo, el número de línea del error (9 en este caso) y una descripción del mismo. Necesitamos comprender el error (esto no es siempre obvio) y después regresar al editor para corregir el texto.

## Ejecución del programa

Una vez que eliminamos todos los errores de compilación, el programa se puede ejecutar (interpretar). Para iniciar la JVM tiene que hacer clic en un botón o usar una opción de menú. Ahora podemos ver el efecto del programa a medida que hace su trabajo.

Tome en cuenta que en algunos IDE, al hacer clic para iniciar el proceso de compilación también se iniciará la ejecución del programa si no se encuentran errores de compilación.

## PRÁCTICA DE AUTOEVALUACIÓN

- 2.2 (a) Averigüe cómo iniciar y usar su editor o su IDE.
- (b) Escriba en su editor texto que contenga la palabra “el” varias veces. Averigüe cómo puede reemplazar en su editor cada ocurrencia de “el” por “ella” con un solo comando.

## Su primer programa

Use el editor o el IDE para crear un nuevo archivo (llamado `Hola.java`) o proyecto y después introduzca el pequeño programa de Java que mostraremos a continuación.

Un archivo que contenga un programa de Java debe tener la extensión `java`. El nombre del archivo debe coincidir con el nombre de la clase, que va después de las palabras `public class` en el código de Java. El programador puede elegir este nombre a su gusto, aunque se sugiere que sea un nombre representativo de la clase.

No se preocupe por lo que significa en esta etapa. Como puede ver, el programa contiene ciertos caracteres inusuales y tres tipos distintos de paréntesis. Tal vez tenga que buscarlos en su teclado. El texto que introduzca en su editor o IDE se conoce como el *código de Java*.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Hello extends JFrame {

    public static void main(String[] args) {
        JOptionPane.showMessageDialog(null, "¡Hola mundo!");
        JOptionPane.showMessageDialog(null, "Adiós");
        System.exit(0);
    }
}
```

Sin duda cometerá errores al teclear este programa. Puede usar el editor para corregir el programa.

Cuando el programa se vea correcto, trate de compilarlo y observe si aparecen mensajes de error. Para corregirlos, compare su código con nuestra versión.

Una de las paradojas en la programación es que con frecuencia los mensajes de error de los compiladores son enigmáticos y de poca ayuda. El compilador le indicará (nota: no señalará con precisión) la posición de los errores. Analice lo que introdujo y trate de encontrar lo que está mal. Los errores comunes son:

- Faltan puntos y comas o están en el lugar incorrecto.
- Faltan llaves.
- Utilizó comillas sencillas (') en vez de comillas dobles (").

Identifique su error, edite el programa y vuelva a compilarlo. ¡Aquí es en donde se pone a prueba su paciencia! Repita el proceso hasta que elimine los errores.

## PRÁCTICA DE AUTOEVALUACIÓN

- 2.3 Cometa un error intencional en su código, borrando un punto y coma. Observe el mensaje de error que produce el compilador. Después vuelva a poner el punto y coma en su lugar.

Después de editar y compilar el programa sin errores, podemos ejecutarlo. El compilador crea un archivo en el disco con la extensión **class**. La primera parte del nombre coincide con el nombre del programa de Java; en este ejemplo el nombre de archivo sería:

Hola.class

Este es el archivo que la JVM ejecutará. Para iniciar el proceso tiene que hacer clic en un botón o seleccionar una opción de menú.

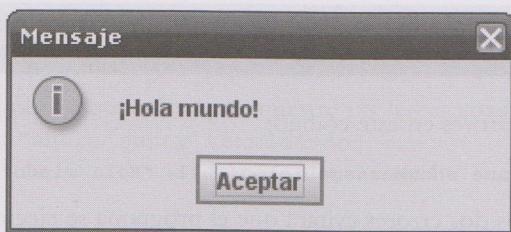


Figura 2.1 La pantalla de `Hola.java`.

Ahora se ejecutará el programa. Primero mostrará el siguiente mensaje:

**¡Hola mundo!**

Haga clic en **Aceptar** para cerrar este mensaje y que aparezca el siguiente mensaje:

**Adiós**

Haga clic en **Aceptar** de nuevo. El programa terminará. En la figura 2.1 se muestra la pantalla del primer mensaje.

## ● Las bibliotecas

Como vimos, la salida que produce el compilador es un archivo `.class`, el cual podemos ejecutar con la JVM. Sin embargo, el archivo de clase no contiene el programa completo. De hecho, todo programa de Java necesita cierta ayuda de parte de una o más piezas de programas contenidos en bibliotecas. En términos computacionales, una biblioteca es una colección de piezas útiles de programas ya escritos con anterioridad, los cuales se guardan en archivos. Su pequeño programa de ejemplo necesita utilizar una de esas piezas de programas para mostrar información en la pantalla. Para poder lograr esto, la pieza de programa requerida se tiene que enlazar con su programa al momento de ejecutarlo.

Las bibliotecas son colecciones de piezas útiles. Suponga que desea diseñar un nuevo automóvil. Probablemente quiera diseñar la carrocería y los interiores. Pero tal vez quiera usar un motor que alguien más haya diseñado y construido con anterioridad. De manera similar, podría también usar las ruedas que algún otro fabricante haya producido. Así, algunos elementos del automóvil serían nuevos y otros se obtendrían de terceros. Los componentes de terceros son como las piezas de programas en la biblioteca de Java. Por ejemplo, nuestro programa de ejemplo usa un cuadro de diálogo que está contenido en una biblioteca.

Las cosas pueden salir mal a la hora de que el compilador verifique los vínculos con el software de la biblioteca y tal vez emita un mensaje de error críptico. Los errores comunes son:

- Falta la biblioteca o no se encuentra.
- Usted escribió mal el nombre de algo en la biblioteca.

Las bibliotecas se incorporan a su programa cuando éste se ejecuta.

## PRÁCTICA DE AUTOEVALUACIÓN

- 2.4 (a) Busque dos errores en este código:

```
JOptionPane.showMessageDialog(null, "Hola mundo");
```

- (b) ¿Cuál de estos dos errores evitara que el programa se ejecute?

## ● Desmitificación del programa

Ahora veamos las generalidades sobre el programa de Java. Aun cuando es bastante pequeño, podemos ver que el programa tiene muchos componentes. Esto se debe a que Java es un verdadero lenguaje de uso industrial, e incluso el programa más pequeño necesita algunos ingredientes importantes. Tenga en cuenta que en esta primera etapa no cubriremos todos los detalles. En los siguientes capítulos nosaremos cargo de ello.

En la figura 2.2 le mostraremos de nuevo el código del programa. Esta vez tiene números de línea para ayudar con la explicación (los números de línea no deben formar parte de un programa real). Las líneas 8 y 9 son las piezas más importantes de este programa. Instruyen a la computadora para que muestre texto en un rectángulo desplegable, conocido como cuadro de diálogo, de la clase `JOptionPane`. La línea 8 muestra el texto `;Hola mundo!`, que debe ir encerrado entre comillas dobles. El texto entre comillas de este tipo se denomina cadena de texto, o cadena. La línea termina con un punto y coma, como lo hacen muchas líneas en Java. De manera similar, la línea 9 muestra `"Adiós"`. En Java, la letra `s` (que representa a Java, desde luego) se antepone a muchos nombres (como `JOptionPane`).

En la figura 2.1 se muestra el efecto de la línea 8. Al hacer clic en Aceptar, el programa continúa con la línea 9. Esta vez se muestra la cadena `"Adiós"`. Observe que los cuadros de diálogo se muestran en secuencia, recorriendo el programa hacia abajo.

```
1 import java.awt.*;
2 import java.awt.event.*;
3 import javax.swing.*;
4
5 public class Hello extends JFrame {
6
7     public static void main(String[] args) {
8         JOptionPane.showMessageDialog(null, ";Hola mundo!");
9         JOptionPane.showMessageDialog(null, "Adiós");
10        System.exit(0);
11    }
12}
```

Figura 2.2 El programa `Hola`.

## PRÁCTICA DE AUTOEVALUACIÓN

- 2.5 Modifique el programa de manera que muestre un tercer cuadro de diálogo en el que aparezca la cadena “Ahora estoy terminando”.

En la parte superior, las líneas 1, 2 y 3 especifican información sobre los programas de biblioteca que nuestro programa utiliza. La palabra `import` va seguida del nombre de una biblioteca que se utilizará en el programa. Este programa utiliza las bibliotecas AWT (Kit de herramientas de Ventanas Abstractas) y Swing para mostrar un cuadro de diálogo.

La línea 4 está en blanco. Podemos usar líneas en blanco en cualquier parte, para mejorar la legibilidad de un programa.

La línea 5 es un encabezado que anuncia que el código es un programa llamado `Hola`.

El programa en sí va encerrado dentro de las llaves. La { de apertura en la línea 5 coincide con la } de cierre en la línea 12. Dentro de estas líneas hay más llaves. La { en la línea 7 coincide con la } en la línea 11.

La línea 10 hace que el programa se deje de ejecutar.

Más adelante le presentaremos un programa más extenso que muestra texto en la pantalla de una manera distinta. Pero antes de hacerlo, veamos el uso de los “objetos” en Java.

## ● Objetos, métodos: una introducción

Una de las razones de la popularidad de Java es que se trata de un lenguaje *orientado a objetos*. Éste es el tema principal del libro, por lo cual lo veremos con detalle en capítulos posteriores. Pero aquí le presentaremos el concepto de los objetos por medio de una analogía.

En primer lugar imagine una casa con un reproductor de CD en la cocina y uno idéntico en la recámara. En la jerga de Java, los denominamos “objetos”. Ahora considere las herramientas que cada reproductor de CD nos ofrece. En otras palabras, ¿qué botones tiene un reproductor de CD? En la jerga de Java, a cada herramienta se le denomina “método” (por ejemplo, podríamos tener los métodos reproducir y detener, y un método para saltar pistas con base en su número).

El término *método* es algo extraño; proviene de la historia de los lenguajes de programación. Imagine que significa “función” o “herramienta”, como en “Este reproductor de CD tiene un método reproducir y un método detener”.

Ahora considere la tarea de identificar cada botón en cada uno de los reproductores. No basta con decir:

`detener`

ya que hay dos reproductores. ¿A cuál de ellos nos referimos? Lo que debemos hacer es identificar el reproductor también. Si nos basamos en algo más parecido a Java, podríamos usar:

`cdCocina.detener`

Observe el uso del punto. Se utiliza de una forma similar en todos los lenguajes orientados a objetos. Tenemos dos elementos:

- El nombre de un objeto; por lo general corresponde a un sustantivo.
- Un método que provee el objeto; por lo general es un verbo.

Más adelante, cuando hablemos sobre los métodos con más detalle, veremos que la versión exacta de Java para el anterior ejemplo es:

```
cdCocina.detener();
```

Observe el punto y coma y los paréntesis que no contienen nada. Para algunos otros métodos tal vez tengamos que suministrar información adicional con la que el método pueda trabajar, como seleccionar una pista numerada:

```
cdRecamara.seleccionar(4);
```

Al elemento entre los paréntesis se le conoce como “parámetro” (de nuevo tenemos un término de programación tradicional, en vez de uno que tenga significado de inmediato).

En términos generales, la forma en que usamos los métodos es:

```
objeto.método(parámetros);
```

Si el método específico no necesita parámetros, de todas formas debemos usar los paréntesis. En el capítulo 5 veremos los parámetros y los métodos.

## PRÁCTICA DE AUTOEVALUACIÓN

- 2.6** Suponga que nuestros reproductores de CD de la recámara y de la cocina tienen herramientas (métodos) para detener, iniciar y seleccionar una pista numerada. He aquí un ejemplo de cómo usar uno de los métodos:

```
cdCocina.seleccionar(6);
```

Proporcione cinco ejemplos sobre el uso de los métodos.

### ● Clases: una analogía

El concepto de clase es sumamente importante en la programación orientada a objetos. Recuerde nuestra analogía: tenemos dos objetos reproductores de CD idénticos en nuestra casa. En la jerga de lenguajes orientados a objetos, decimos que tenemos dos “instancias” de la “clase” reproductor de CD. Una clase es como una línea de producción que puede fabricar nuevos reproductores de CD.

Vamos a diferenciar entre una clase y las instancias de una clase. La casa tiene dos instancias de la clase reproductor de CD, las cuales existen de verdad: realmente podemos usarlas. Una clase es un concepto más abstracto. Aunque la línea de producción de reproductores de CD posee el diseño (de una forma u otra) de un reproductor de CD, no existe una instancia real sino hasta que la

máquina fabrica una. En Java utilizamos la palabra “new” para indicarle a una clase que fabrique una nueva instancia. Para resumir:

- Los objetos son instancias de una clase.
- Una clase puede producir todas las instancias que necesitemos.

Vale la pena repetir que estos conceptos son los principales de este libro, por lo cual los cubriremos con mucho más detalle en capítulos posteriores. No esperamos que usted pueda crear programas de Java con objetos y clases en este capítulo.

## ● Uso de un campo de texto

El primer programa que vimos utilizaba un cuadro de diálogo. El segundo programa que presentaremos es más grande, pero constituye la base para muchos de los programas de este libro. Utiliza un *campo de texto* para mostrar una sola línea de texto, la cadena “¡Hola!” en este caso. En la figura 2.3 se muestra la pantalla y en la figura 2.4 aparece el código con números de línea.

En esta etapa necesitamos recordarle de nuevo que empezaremos a ver los detalles sobre Java en el siguiente capítulo. Por ahora le vamos a mostrar algunos programas con una explicación general sobre lo que hacen. No esperamos (todavía) que usted pueda analizar una línea de código y decir con precisión lo que hace.

Como antes, incluimos los números de línea para que ayuden en nuestras explicaciones, pero no debe introducirlos junto con el programa. El nombre que sigue a las palabras `public class` es **Saludo**, por lo que debe guardar el programa en un archivo llamado:

**Saludo.java**

Compile y ejecute el programa. Para detenerlo, haga clic en la cruz en la esquina superior derecha de la ventana o haga clic en el ícono de Java en la esquina superior izquierda y después seleccione **Cerrar** en el menú.

Ahora veremos algunos de los usos de instancias, métodos y parámetros.

Recuerde nuestras analogías. Anteriormente mencionamos el uso de la notación “punto” para los objetos y sus métodos asociados. Localice la línea 11:

```
marco.setSize(300, 200);
```

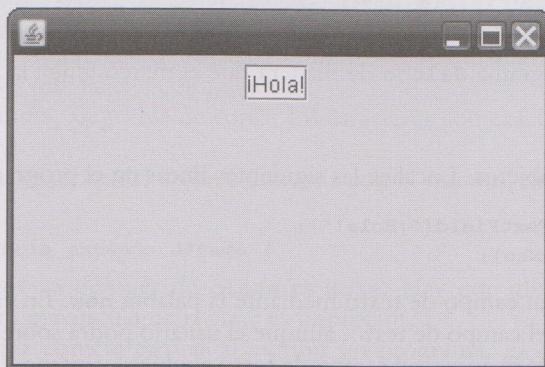


Figura 2.3 La pantalla de `Saludo.java`.

```
1 import java.awt.*;
2 import java.awt.event.*;
3 import javax.swing.*;
4
5 public class Saludo extends JFrame {
6
7     private JTextField textField;
8
9     public static void main (String[] args) {
10         Saludo marco = new Saludo();
11         marco.setSize(300, 200);
12         marco.createGUI();
13         marco.setVisible(true);
14     }
15
16     private void createGUI() {
17         setDefaultCloseOperation(EXIT_ON_CLOSE);
18         Container window = getContentPane();
19         ventana.setLayout(new FlowLayout());
20         campoTexto = new JTextField("¡Hola!");
21         ventana.add(campoTexto);
22     }
23 }
```

Figura 2.4 El programa `Saludo`.

Aquí se utiliza la misma notación: objeto, punto, método, parámetros.

Imagine el objeto marco como el borde exterior de la pantalla de la figura 2.3. El método `setSize` recibe dos parámetros: la anchura y altura requeridas del marco en unidades conocidas como píxeles. Aquí utilizamos una metodología orientada a objetos para establecer el tamaño del marco.

### PRÁCTICA DE AUTOEVALUACIÓN

**2.7** Modifique el programa `Saludo` de manera que el marco tenga la mitad de la anchura.

He aquí otro uso de los objetos. Localice las siguientes líneas en el programa:

```
campoTexto = new JTextField("¡Hola!");
ventana.add(campoTexto);
```

En primer lugar, se crea un campo de texto mediante la palabra `new`. En esta etapa podemos elegir el texto que aparecerá en el campo de texto, aunque el usuario podrá sobrescribirlo a la hora de ejecutar el programa. Después se agrega el campo de texto al objeto ventana. Cuando se ejecuta el programa aparece el campo de texto y se centra de manera automática.

## PRÁCTICAS DE AUTOEVALUACIÓN

- 2.8 Modifique el programa para que muestre el siguiente texto:

Un bloque de texto en un campo de texto

- 2.9 Ejecute el programa **Saludo** y trate de cambiar el tamaño del marco arrastrándolo con el ratón. ¿Qué ocurre con la posición del campo de texto?

Por último, una observación general sobre nuestro segundo programa. La mayor parte de las instrucciones tienen que ver con la especificación de las bibliotecas necesarias y cómo establecer la apariencia visual de la pantalla, es decir, la “interfaz gráfica de usuario” o GUI.

Imagine la GUI de su procesador de texto favorito. En la parte superior de su ventana podrá ver una gran cantidad de menús y botones. Al ejecutar el procesador de texto los menús y botones aparecerán en forma instantánea, por lo que tal vez se sorprenda al saber que, tras bambalinas, el procesador de texto empieza con una ventana toda en blanco y laboriosamente agrega cada menú y botón a la ventana, uno por uno. Debido a la velocidad de la computadora el proceso parece instantáneo.

Al escribir programas más grandes, de todas formas se tiene que realizar la configuración inicial de la pantalla, pero esa parte del código se vuelve menos importante que el código encargado de que el programa realice una tarea al momento de que el usuario hace clic en un botón o en un elemento de menú.

## Fundamentos de programación

- Una de las principales características de Java es el uso extendido de las clases.
- La notación “punto” para usar los objetos es:  
`objeto.método(parámetros)`
- Las instrucciones se llevan a cabo en secuencia, desde la parte superior del programa hasta la parte inferior.

## Errores comunes de programación

- Cuando edite un programa, guárdelo cada 10 minutos aproximadamente para evitar perder su trabajo en caso de que falle la computadora.
- Al escribir un programa, asegúrese de copiar los caracteres con exactitud; use las mayúsculas según se muestra.
- Asegúrese de que el nombre del archivo coincida con el nombre de la clase en el mismo. Por ejemplo:

```
public class Hola extends JFrame {
```

Aquí, el nombre que va después de **class** es **Hola**. Hay que guardar el archivo como **Hola.java**. La letra mayúscula de la clase es importante.

- Es muy probable que cometa errores al escribir un programa. El compilador le indicará cuáles son los errores. Procure no frustrarse demasiado por los errores.

## Secretos de codificación

Los programas de Java contienen varias llaves de apertura y cierre. Debe haber el mismo número de llaves de cierre que de llaves de apertura.

## Nuevos elementos del lenguaje

Un cuadro de diálogo puede mostrar una cadena de texto junto con un botón **Aceptar**, como en:

```
JOptionPane.showMessageDialog(null, "¡Hola mundo!");
```

## Resumen

- Los programas de Java se pueden crear y ejecutar en la mayoría de los tipos de computadoras.
- Se utiliza un editor o un IDE para crear y modificar el código fuente de los programas de Java.
- Hay que compilar un programa de Java antes de poder ejecutarlo.
- Es necesario corregir los errores de compilación para poder ejecutar un programa.
- El compilador produce un archivo con el mismo nombre que el archivo original de Java, pero con la extensión **class**. Este archivo contiene instrucciones en código de bytes.
- La JVM (Máquina Virtual de Java) se utiliza para ejecutar programas.
- Gran parte del poder de Java proviene de sus bibliotecas, las cuales se enlazan al momento de ejecutar el programa.
- Java es orientado a objetos. Utiliza los conceptos de clases, instancias y métodos.
- La notación “punto” se utiliza en todos los programas de Java. He aquí un ejemplo:

```
marco.setSize(300, 200);
```

- Los métodos (como **setSize**) realizan tareas sobre el objeto especificado (como **marco**).
- Las cosas pueden salir mal en cualquier etapa; parte del trabajo del programador es identificar y corregir los errores. No lo olvide: es extraño que todo funcione sin problemas la primera vez. Tenga cuidado y relájese.

## Ejercicios

- 2.1 Asegúrese de saber cómo compilar y ejecutar programas de Java en su computadora. Compile y ejecute los dos programas de este capítulo.
- 2.2 En el programa **Hola**, agregue un cuadro de diálogo para mostrar su nombre.
- 2.3 En el programa **Saludo**, haga que el campo de texto muestre su nombre.

## Respuestas a las prácticas de autoevaluación

- 2.1 (a) No hay diferencia. Los términos significan lo mismo.  
(b) Una carpeta contiene varios archivos y/u otras carpetas.  
(c) Sí. Mientras las dos carpetas con nombres idénticos no estén dentro de la misma carpeta (por ejemplo, cada una de las carpetas **Trabajo** y **Casa** podrían contener una carpeta llamada **cartas**).
- 2.2 (a) Esto depende de los editores disponibles en su computadora. Si utiliza un IDE, ya incluye un editor.  
(b) Esto depende de su editor. Muchos editores tienen una herramienta **Buscar...Reemplazar**, la cual explora todo el texto.
- 2.3 El mensaje de error variará, dependiendo del punto y coma que haya omitido.
- 2.4 (a) Hay una "p" incorrecta. Debe ser "P" como en **JOptionPane**. La palabra **mindo** está mal escrita.  
(b) El error de la "p" evitará que el programa se compile, por lo tanto no se podrá ejecutar. El error "mindo" no evitará que el programa se ejecute, pero el resultado no será el esperado.
- 2.5 Inserte la siguiente línea justo después de la línea 9, que muestra el texto "**Adiós**":

```
JOptionPane.showMessageDialog(null, "Ahora estoy terminando");
```

Compile y ejecute el programa modificado.

- 2.6 

```
cdCocina.reproducir();
cdCocina.detener();
cdRecamara.reproducir();
cdRecamara.detener();
cdRecamara.seleccionar(3);
```

- 2.7 Reemplace el **300** por **150** en la línea 11, después vuelva a compilar el programa y ejecútelo.

- 2.8 Reemplace el texto "**¡Hola!**" en la línea 20 por:

```
"Un bloque de texto en un campo de texto"
```

Compile y ejecute el programa.

- 2.9 Permanece centrado cerca de la parte superior del marco.