Quinta edición

JAVA™ CÓMO PROGRAMAR

DEITEL • DEITEL

Java™ ha revolucionado el desarrollo del software mediante el código orientado a objetos independiente de la plataforma, con uso intensivo de multimedia, para aplicaciones basadas en Internet, intranets o extranets. Esta *quinta edición* del libro de texto sobre Java más utilizado a nivel mundial, explica cómo utilizar las extraordinarias herramientas de este software.

La obra contiene una vasta colección de ejemplos, ejercicios y proyectos que proporcionan la oportunidad de resolver problemas reales. El contenido se concentra en los principios de la buena ingeniería de software, haciendo hincapié en la claridad de los programas. Todos los ejemplos de código han sido probados.
Entre los temas clave que se tratan aquí, se encuentran:

- Aplicaciones/Applets
- GUI Swing/Manejo de eventos
- Clases/Objetos/Interfaces
- POO/Herencia/Polimorfismo
- Gráficos/Java 2D™/Imágenes/Animación/Audio
- Excepciones/Subprocesamiento múltiple
- DOO/UML/Patrones de diseño

Cómo programar en Java, quinta edición incluye extensas características pedagógicas:

- Cientos de programas en código activo (LIVE-CODE™) con capturas de pantallas que muestran los resultados obtenidos
- Una gran cantidad de recursos en Internet y Web para impulsar al lector a que investigue más sobre los temas de su interés
- Cientos de tips, prácticas recomendadas y precauciones. Todos identificados con su respectivo icono:

En el sitio Web `www.pearsoneducacion.net/deitel` encontrará los ejemplos de código del libro e información para maestros, estudiantes y profesionales.
Sin duda, éste es el libro más completo para aprender Java.

El CD incluye:

- Archivos de texto en *español* no incluidos en el libro
- Todo el código fuente utilizado en los ejercicios del libro
- Programas para ejecutar Java, como Java™ 2 Platform Software Development Kit y Apache Tomcat v.4.1.12, y mucho más

ISBN 970-26-0518-0

Visítenos en:
www.pearsoneducacion.net

DEITEL

JAVA™ CÓMO PROGRAMAR

Quinta edición

Introducción al DOO con UML y los PATRONES DE DISEÑO JDBC™, SERVLETS, JSP™

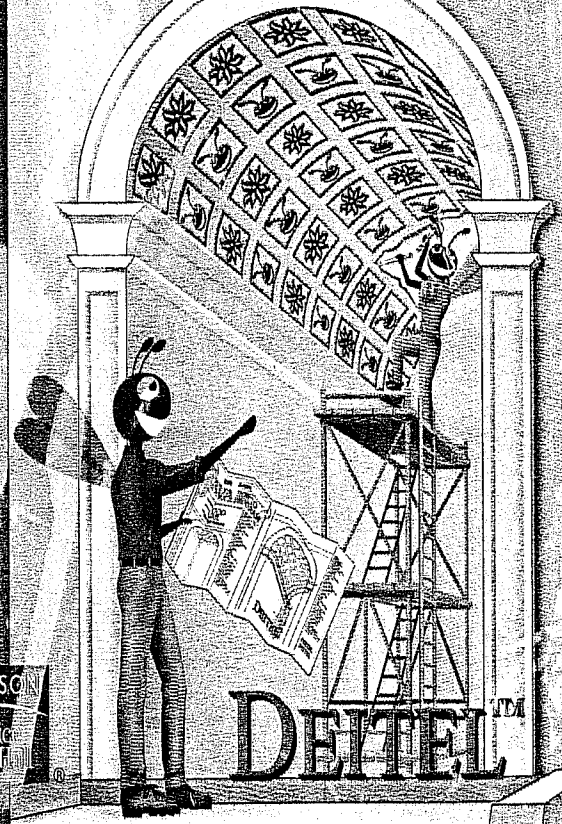
INCLUYE CD-ROM

Quinta edición

Incluye CD con el Kit de desarrollo de software Java™ 2, Standard Edition, Apache Tomcat v.4.1.12, ¡y mucho más!

JAVA™ CÓMO PROGRAMAR

Introducción al DOO con UML y los PATRONES DE DISEÑO JDBC™, SERVLETS, JSP™



DEITEL

DEITEL
DEITEL

QUINTA EDICIÓN

CÓMO PROGRAMAR EN JAVA

QUINTA EDICIÓN

CÓMO PROGRAMAR EN JAVA

Jorge Alberto Barrios García

Harvey M. Deitel
Deitel & Associates, Inc.

Paul J. Deitel
Deitel & Associates, Inc.

TRADUCCIÓN

Alfonso Vidal Romero Elizondo
Ingeniero en Sistemas Electrónicos
Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Monterrey

REVISIÓN TÉCNICA

M. en C. Gabriela Azucena Campos García
Profesora de tiempo completo
Departamento de Sistemas de Información
División de Profesional y Graduados
Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Estado de México

M. en C. Sergio Fuenlabrada Velázquez
Ing. Mario Alberto Sesma Martínez
M.C.I. Edna Martha Miranda Chávez
Profesores investigadores
Academia de computación
Unidad Profesional Interdisciplinaria de Ingeniería y Ciencias Sociales y Administrativas
Instituto Politécnico Nacional

M. en C. Jesús Muñoz Bauza
Profesor investigador
Departamento de Computación
División de Profesional y Graduados
Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Ciudad de México

PEARSON Educación

México • Argentina • Brasil • Colombia • Costa Rica • Chile • Ecuador
España • Guatemala • Panamá • Perú • Puerto Rico • Uruguay • Venezuela

Datos de catalogación bibliográfica

DEITEL, HARVEY M. y DEITEL, PAUL J.
Quinta edición

Cómo programar en Java

PEARSON EDUCACIÓN, México, 2004

ISBN: 970-26-0518-0
Área: Universitarios

Formato: 20 × 25.5 cm Páginas: 1268

ISBN 0-13-101621-0

Edición en español:
Editor: Guillermo Trujano Mendoza
e-mail: guillermo.trujano@pearsoned.com
Editor de desarrollo: Miguel B. Gutiérrez Hernández
Supervisor de producción: Enrique Trejo Hernández

Edición en inglés:
Vice President and Editorial Director, ECS: ***Marcia J. Horton***
Acquisitions Editor: ***Petra J. Recter***
Assistant Editor: ***Sarah Parker***
Project Manager: ***Jennifer Cappello***
Vice President and Director of Production and Manufacturing, ESM: ***David W. Riccardi***
Executive Managing Editor: ***Vince O'Brien***
Managing Editor: ***Tom Manshreck***
Production Editor: ***John F. Lovell***
Director of Creative Services: ***Paul Belfanti***
Creative Director: ***Carole Anson***
Chapter Opener and Cover Designer: ***Tamara L. Newnam, Dr. Harvey Deitel***
Interior Design Assistance: ***Geoffrey Cassar***
Manufacturing Manager: ***Trudy Pisciotti***
Manufacturing Buyer: ***Lisa McDowell***
Marketing Manager: ***Pamela Shaffer***
Marketing Assistant: ***Barrie Reinhold***

QUINTA EDICIÓN, 2004

D.R. © 2004 por Pearson Educación de México, S.A. de C.V.
Atlacomulco 500-5o. piso
Col. Industrial Atoto
53519, Naucalpan de Juárez, Edo. de México
e-mail: editorial.universidades@pearsoned.com

Cámara Nacional de la Industria Editorial Mexicana.
Reg. Núm. 1031.

Prentice Hall es una marca registrada de Pearson Educación de México, S.A. de C.V.

ISBN 970-26-0518-0

Impreso en México. *Printed in Mexico.*
1 2 3 4 5 6 7 8 9 0 - 05 04 03 02

PARA:

Terell Hull y James Huddleston:

Por su constante dedicación a la excelencia en la enseñanza y al escribir sobre Java y la tecnología de objetos.

Por sus extraordinarias contribuciones como revisores y por insistir en que "lo hiciéramos bien".

Gracias por ser nuestros mentores, colegas y amigos.

Es un privilegio trabajar con profesionales consumados en el software.

Harvey y Paul Deitel

Contenido

Prefacio

¡Bienvenido a Java! En Deitel & Associates escribimos libros de texto sobre lenguajes de programación, tanto de nivel universitario como profesional, y nos esforzamos por mantener nuestros libros actualizados. Escribir esta quinta edición fue un placer. Este libro y el material de apoyo incluyen todo lo que necesitan los profesores y estudiantes para obtener una experiencia educativa interesante, informativa, retadora y entretenida con Java. Esta publicación concuerda con la última versión de Java [*plataforma Java 2, Standard Edition (J2SE), versión 1.4.1*] y con el diseño orientado a objetos, utilizando la versión más reciente de *UML (Lenguaje Unificado de Modelado)* desarrollado por el Object Management Group (OMG). Pusimos a tono la escritura, la pedagogía, nuestro estilo de codificación, el paquete de accesorios del libro, y además incluimos un análisis importante del desarrollo de aplicaciones de bases de datos basadas en Internet y Web. Pasamos el *Paseo por el libro* al prefacio. Este paseo ayudará a los profesores, estudiantes y profesionales a tener una idea más clara de la extensa cobertura que proporciona este libro, en relación con la programación orientada a objetos en Java, el diseño orientado a objetos con UML y el desarrollo de aplicaciones basadas en Internet y Web.

Ya sea usted profesor, estudiante, profesional experimentado o programador novato, este libro tiene mucho que ofrecerle. Java es un lenguaje de programación de clase mundial que se utiliza para desarrollar aplicaciones computacionales de uso industrial, para dispositivos que van desde teléfonos celulares y PDAs, hasta los servidores empresariales más grandes. Nosotros auditamos cuidadosamente el manuscrito, comparándolo con la *Especificación del lenguaje Java*[1], que define a este lenguaje. Como resultado, los programas que usted cree al estudiar este texto deberán funcionar con cualquier plataforma Java que sea compatible con la J2SE 1.4.1.

En este prefacio veremos la extensa suite de materiales educativos de *Cómo programar en Java, 5ª edición*, los cuales ayudarán a los profesores a maximizar la experiencia de aprendizaje de sus estudiantes en cuanto a Java. Explicamos las convenciones que utilizamos, como la presentación de la sintaxis del código de los ejemplos, para "lavar código" y para resaltar segmentos importantes de éste, lo que ayuda a los estudiantes a enfocar su atención en los conceptos clave que se introducen en cada capítulo. Daremos un vistazo general a las nuevas características de *Cómo programar en Java, 5ª edición*, incluyendo nuestra adaptación mejorada sobre la programación orientada a objetos, el desarrollo de aplicaciones Web con servlets y JSP, el ejemplo práctico opcional mejorado de diseño orientado a objetos (DOO) con UML sobre la simulación de un elevador, el vistazo general a los patrones de diseño y el uso extensivo de diagramas de UML que se han actualizado a los estándares de la versión 1.4 de UML.

El libro incluye un CD con el texto del *Kit de desarrollo de software J2SE 1.4.1* (J2SDK) de Sun Microsystems, junto con *Sun ONE Studio 4 (Community Edition)*, el entorno integrado de desarrollo (IDE) de Sun. Para un mayor apoyo a los programadores novatos, ofrecemos varias publicaciones gratuitas de la *Serie DIVE-INTO™*, las cuales explican cómo compilar, ejecutar y depurar programas en Java utilizando el Kit J2SDK, Sun

1. Existen copias electrónicas en formato HTML y PDF de la *Especificación del lenguaje Java*, las cuales están disponibles en forma gratuita en el sitio Web sobre Java de Sun Microsystems en `java.sun.com/docs/books/jls/index.html`.

ONE Studio (Community Edition) y el software *JBuilder* de Borland, *Personal edition*. Todo este material se encuentra en inglés y está disponible en la página `www.deitel.com/books/downloads.html` o en `www.pearsoneducacion.net/deitel`, junto con otros recursos para este libro.

Además, en este sitio ofrecemos un panorama general del paquete completo de materiales adicionales para profesores y estudiantes que utilicen este libro como texto en un curso. Entre otros materiales, también en inglés, se incluye un *CD* llamado *Instructor's Resource*, con soluciones a la mayoría de los ejercicios planteados en los capítulos del libro, así como un archivo llamado *Test-Item* que contiene cientos de preguntas de opción múltiple y sus respuestas correspondientes. Los profesores que deseen mantener sesiones cerradas de laboratorio (o asignar tareas altamente estructuradas), pueden adquirir un manual de laboratorio, *Java in the Lab*. Esta publicación incluye actividades previas al laboratorio, ejercicios de laboratorio y actividades posteriores, que han sido cuidadosamente elaboradas para este fin.

Este prefacio también analiza la quinta edición de *The Java 2 Multimedia Cyber Classroom 5ª ed.*, que es la versión multimedia, en inglés, de la versión original de este libro. Esta herramienta de aprendizaje proporciona extensas características de interactividad, incluyendo hipervínculos, búsqueda de texto, "recorridos" con audio a través de los programas, animaciones en Flash® y cientos de ejercicios y sus soluciones.

Analizaremos también varias herramientas DEITEL™ de aprendizaje en línea, incluyendo una explicación del contenido de *Blackboard*, *CourseCompass* y *WebCT* de la serie *Course Management Systems*, cada uno de los cuales apoya el material de este libro.

La cuarta edición de este libro fue revisada por 35 distinguidos académicos y profesionales del ramo. Después de aplicar sus comentarios, el manuscrito para *Cómo programar en Java, 5ª edición*, fue revisado nuevamente por 44 distinguidos académicos y profesionales. En los reconocimientos mencionamos los nombres de los revisores y en dónde laboran. El prefacio concluye con información acerca de los autores y de Deitel & Associates, Inc. Si al leer este libro surge alguna pregunta, por favor envíe un correo electrónico a `deitel@deitel.com`; le responderemos inmediatamente. Visite con regularidad nuestro sitio Web, `www.deitel.com`, e inscríbase en el boletín de correo electrónico *DEITEL™ BUZZ ONLINE* en `www.deitel.com/newsletter/subscribe.html`. Utilizamos el sitio Web y el boletín para mantener a nuestros lectores actualizados en cuanto a las publicaciones y servicios de DEITEL™.

Nuevas características en Cómo programar en Java, 5ª edición

Esta edición contiene muchas nuevas características y mejoras, incluyendo:

Resaltado de código
En este libro hemos agregado mucho código resaltado (con pantalla). Al revisar nuestro código eliminamos la mayoría de los fragmentos "redundantes" que aparecían en ediciones anteriores. Dichos fragmentos los mantuvimos en la primera parte del libro como una herramienta pedagógica que ayudara a los principiantes. Queremos que el lector vea, en contexto, las nuevas características del código, por lo cual, a partir del capítulo 4, nuestras revisiones al código simplemente harán referencia a los números de línea de los nuevos segmentos de código en los programas fuente. Para facilitar a los lectores el detectar los segmentos presentados, los hemos resaltado. Esto ayuda a los estudiantes a revisar el material rápidamente cuando se preparen para algún examen.

"Lavado de código"
Lavado de código es el término que utilizamos para aplicar comentarios extensos, para utilizar identificadores importantes, para aplicar sangría y usar espaciado vertical para separar las unidades importantes de un programa. Este proceso da como resultado programas que son mucho más fáciles de leer y que sirven como autodocumentación. Hemos realizado un amplio "lavado" a todo el código de los programas de fuente de este texto, del manual de laboratorio, del material auxiliar y del *Cyber Classroom*.

El proceso para poner a punto la programación orientada a objetos en los capítulos 9 y 10
Ésta es una de las mejoras más importantes de esta nueva edición. Hemos realizado una actualización muy precisa del capítulo 9 de *Cómo programar en Java, 4ª edición*, y lo dividimos en dos capítulos. Las mejoras hacen el material más claro y accesible para los estudiantes y profesionales, especialmente para los que estudian la programación orientada a objetos por primera vez.

Capítulo 9 "Programación orientada a objetos: herencia". El capítulo 9 lleva cuidadosamente al lector a través de una secuencia de cinco ejemplos que muestran los datos privados (`private`), protegidos (`protected`) y la reutilización de software por medio de la herencia. Comenzamos mostrando una clase con variables de instancia privada y métodos públicos para manipular esos datos. Luego implementamos una segunda clase con varias capacidades adicionales. Para esto, duplicamos gran parte del código del primer ejemplo. En nuestro tercer ejemplo empezamos nuestra explicación sobre la herencia y la reutilización de software; utilizamos la clase del primer ejemplo como una superclase y heredamos sus datos y su funcionalidad en una nueva subclase. Este ejemplo introduce el mecanismo de la herencia y demuestra que una subclase no puede acceder a los miembros privados de su superclase directamente. Esto motiva nuestro cuarto ejemplo, en el cual introdujimos datos protegidos en la superclase y demostramos que la subclase puede acceder a los datos protegidos que hereda de la superclase. El último ejemplo en la secuencia demuestra la ingeniería de software adecuada al definir los datos de la superclase como privados y al utilizar los métodos públicos de la superclase (que fueron heredados por la subclase) para manipular, desde la subclase, los datos privados de la superclase. Seguimos la introducción de cinco partes con una jerarquía de clases de tres niveles, la cual emplea las técnicas de ingeniería de software que se introducen en la primera parte de este capítulo. El capítulo cierra con una discusión de la ingeniería de software con la herencia.

Capítulo 10 "Programación orientada a objetos: polimorfismo". El nuevo capítulo 10 se basa en los conceptos de herencia que presentamos en el capítulo 9, y se enfoca en las relaciones existentes entre las clases de una jerarquía de clases. El capítulo 10 utiliza una secuencia de tres ejemplos para presentar las poderosas capacidades de procesamiento que se permiten en este tipo de relaciones. Comenzamos con un ejemplo que ilustra la relación "es un" entre un objeto de una subclase y su tipo de superclase. Esta relación permite que el objeto de la subclase sea tratado como un objeto de su superclase. Mostramos que podemos asignar una referencia de un objeto de la subclase a una variable de la superclase, e invocar los métodos de la superclase en ese objeto. Este ejemplo utiliza el polimorfismo, que permite a un programa procesar objetos de clases relacionadas por una jerarquía de clases, como objetos de su tipo de superclase. Cuando un método se invoca por medio de una variable de la superclase, se invoca la versión de ese método que es específica de la subclase. En nuestro segundo ejemplo demostramos que lo inverso no se aplica (el objeto de una superclase no se considera como un objeto de su tipo de subclase) y también mostramos que se producen errores de compilación si un programa intenta manipular un objeto de la superclase en esta forma. Nuestro tercer ejemplo demuestra que los únicos métodos que pueden invocarse a través de la variable de una superclase son los que están definidos por el tipo de superclase. El ejemplo muestra que los intentos de invocar métodos que sólo pertenecen a la subclase ocasionan errores de compilación. El capítulo continúa con un ejemplo práctico sobre el polimorfismo, en el cual procesamos un arreglo de variables que contienen referencias a objetos. Todos los objetos referenciados por los elementos del arreglo tienen una superclase abstracta común, la cual contiene el conjunto de métodos comunes para todas las clases de la jerarquía. Concluimos con un ejemplo práctico que demuestra cómo un programa que procesa objetos en forma polimórfica puede aún realizar un procesamiento específico a cada tipo, determinando el tipo del objeto que se esté procesando en ese momento.

Nuevas APIs de E/S (NIO) de Java
Las nuevas APIs de E/S de Java son un agregado considerable a la J2SE 1.4. Analizamos partes de estas APIs en algunas secciones de tres capítulos. La sección 11.8 demuestra las capacidades de las nuevas APIs de E/S en cuanto al uso de expresiones regulares, lo cual permite a los programas buscar patrones de caracteres en las cadenas. La sección 17.13 introduce las clases de E/S de alto rendimiento de las nuevas APIs de E/S, las cuales permiten a los desarrolladores aprovechar los búferes, canales, conjuntos de caracteres y demás. Esta sección también presenta un ejemplo del uso de canales y búferes para escribir datos hacia, y leer datos desde, un archivo. La sección 18.11 continúa nuestra discusión sobre las nuevas APIs de E/S con una introducción a los selectores y a la E/S sin bloqueo para implementar servidores de red de alto rendimiento. Después implementamos un programa de conversación distribuida que demuestra esas capacidades. Las secciones 11.8 y 17.13 también proporcionan vínculos Web para un estudio más detallado de las nuevas APIs de E/S.

Desarrollo de aplicaciones Web y de bases de datos con JDBC, servlets y JSP
Debido a la demanda popular, hemos regresado varios temas a *Cómo programar en Java, 5ª edición*. El capítulo 23, Conectividad de bases de datos en Java mediante JDBC, demuestra cómo crear aplicaciones controladas por

bases de datos con la API JDBC™. El capítulo 24, Servlets y el capítulo 25, JavaServer Pages™ (JSP), expanden nuestro análisis de los temas de programación relacionados con Internet y Web, y tienen todo lo que los lectores necesitan para empezar a desarrollar sus propias aplicaciones basadas en Web ¡que funcionarán en Internet! Los lectores aprenderán a crear las denominadas aplicaciones de *n* niveles, en las cuales la funcionalidad que proporciona cada nivel puede distribuirse para separar computadoras a través de Internet, o pueden ejecutarse en la misma computadora. En especial, crearemos una aplicación de encuesta de tres niveles basada en Web, y una aplicación de libro de visitantes de tres niveles basada en Web. La información de cada aplicación se almacena en el nivel de datos de la aplicación; en este libro se utiliza una base de datos implementada con el producto de bases de datos Cloudscape de IBM, basado en Java (en el CD del libro se incluye una versión de prueba). El usuario introduce las peticiones y recibe las respuestas en el nivel de cliente de cada aplicación, que por lo general es una computadora que ejecuta un navegador Web, como Microsoft Internet Explorer o Netscape. Por supuesto, los navegadores Web saben cómo comunicarse con los sitios Web a través de Internet. El nivel medio contiene un servidor Web y uno o más servlets específicos a la aplicación (en el caso de nuestra aplicación de encuesta) o JavaServer Pages (en el caso de nuestra aplicación de libro de visitantes). Utilizamos el servidor Web Tomcat de Apache como nuestro servidor de aplicaciones para estos ejemplos. Tomcat, que es la implementación de referencia para las tecnologías de servlets y JavaServer Pages, se incluye en el CD que acompaña a este libro y está disponible para descargarse en forma gratuita de `www.apache.org`. Tomcat se comunica con el nivel de cliente a través de Internet, utilizando el Protocolo de transferencia de hipertexto (HTTP). Hablamos sobre el papel imprescindible del servidor Web en la programación Web y proporcionamos muchos ejemplos que demuestran las interacciones entre un navegador Web y un servidor Web.

Lenguaje Unificado de Modelado™ (UML)

El Lenguaje Unificado de Modelado™ (UML) se ha convertido en el lenguaje de modelado gráfico preferido para diseñar sistemas orientados a objetos. En la edición anterior de este libro solamente utilizamos UML en secciones opcionales, y empleamos segmentos de diagramas de flujo convencionales y diagramas de herencia para reforzar las explicaciones. Ahora hemos transformado completamente los diagramas del libro para que sean compatibles con UML 1.4. En especial, actualizamos todas las figuras del ejemplo práctico de simulación del elevador con UML/DOO; convertimos todos los diagramas de flujo en los capítulos 4 y 5 sobre las instrucciones de control, a diagramas de actividad de UML; y convertimos todos los diagramas de herencia de los capítulos 9, 10, 12, 13 y 15 a diagramas de clases de UML.

Esta quinta edición pone a punto cuidadosamente el ejemplo práctico opcional (pero ampliamente recomendado) que presentamos sobre el diseño orientado a objetos, utilizando UML. El ejemplo práctico se envió a un equipo distinguido de revisores de DOO/UML, incluyendo líderes en el campo por parte de Rational (los creadores de UML) y del Grupo de administración de objetos (responsable del mantenimiento y la evolución de UML). En el ejemplo práctico implementamos completamente la simulación de un elevador. En las secciones "Acerca de los objetos" al final de los capítulos 1 a 8, 10 a 14, 16 y 19, presentamos una introducción detallada al diseño orientado a objetos, utilizando UML. Mostramos un subconjunto conciso y simplificado de UML, y después guiamos al lector a través de su primera experiencia con el diseño, la cual está dirigida a diseñadores y programadores orientados a objetos que apenas inician. El ejemplo práctico está completamente resuelto. No es un ejercicio, sino una experiencia completamente educativa, que concluye con un recorrido detallado por el código de Java. En cada uno de los primeros cinco capítulos nos concentramos en la metodología "convencional" de la programación estructurada, ya que los objetos que creamos utilizan estas piezas de programa estructurado. Concluimos cada capítulo con una sección "Acerca de los objetos", en la que presentamos una introducción al diseño orientado a objetos (DOO) mediante el uso de UML. Estas secciones ayudarán a los estudiantes a desarrollar un diseño orientado a objetos, de manera que puedan utilizar inmediatamente los conceptos de programación orientada a objetos que comienzan a aprender en el capítulo 8. En la primera de estas secciones al final del capítulo 1, presentamos los conceptos básicos y la terminología del DOO. En las secciones "Acerca de los objetos" opcionales al final de los capítulos 2 a 5 consideramos cuestiones más importantes mientras hacemos frente a problemas que suponen un reto con las técnicas del DOO. Analizamos la declaración de un problema típico en el que se requiere la construcción de un sistema, determinamos los objetos necesarios para implementarlo, los atributos que necesitan tener estos objetos, y los comportamientos que necesitan exhibir dichos objetos y especificamos la manera en que los objetos deberán interactuar entre sí para cumplir con los requerimientos del sistema. Logramos esto incluso antes de comenzar a discutir cómo escribir programas en Java. En los apéndices D a F, incluimos una implementación en Java del sistema orientado a objetos que diseñamos en los capítulos anteriores. Este ejemplo práctico ayudará a los estudiantes a prepararse para los tipos de proyectos que encontrarán en la industria. Empleamos un proceso de diseño orientado a objetos incremental y cuidadosamente desarrollado para producir un modelo en UML para nuestra simulación de elevador. A partir de este diseño producimos una verdadera y funcional implementación en Java utilizando las nociones claves de programación, incluyendo clases, objetos, encapsulamiento, visibilidad, composición y herencia.

Descubrimiento de patrones de diseño

Estas secciones opcionales introducen patrones de diseño orientado a objetos populares. Durante la década pasada, la industria de la ingeniería de software progresó considerablemente en el campo de los *patrones de diseño*: arquitecturas comprobadas para construir software orientado a objetos que sea flexible y pueda mantenerse[2]. El uso de patrones de diseño puede reducir considerablemente la complejidad del proceso de diseño. Presentamos varios patrones de diseño en Java, pero éstos pueden implementarse en cualquier lenguaje orientado a objetos tal como C++, C# o Visual Basic .NET. Describimos varios patrones de diseño utilizados por Sun Microsystems en la API de Java. Además, se utilizaron en muchos programas de este libro, los cuales identificaremos en las secciones "Descubrimiento de patrones de diseño". Estos programas proporcionan ejemplos acerca del uso de patrones de diseño para construir software orientado a objetos que sea confiable y robusto.

Método de enseñanza

Esta quinta edición contiene una vasta colección de ejemplos, ejercicios y proyectos sustraídos de diversos campos para proporcionar al estudiante la oportunidad de resolver problemas interesantes del mundo real. El libro se concentra en los principios de la buena ingeniería de software, haciendo hincapié en la claridad de los programas. Evitamos la tecnología arcana y las especificaciones sintácticas, para favorecer la enseñanza mediante ejemplos. Nuestros ejemplos de código se han probado en las plataformas populares de Java. Somos educadores que enseñamos temas de vanguardia en salones de clases de la industria alrededor del mundo. Este texto pone énfasis en la buena pedagogía.

Cómo aprender Java mediante el método LIVE-CODE™

Este libro contiene numerosos ejemplos "reales". Cada nuevo concepto se presenta en el contexto de un ejemplo completo y funcional, que es seguido inmediatamente por una o más ejecuciones que muestran la entrada/salida del programa. Este estilo ejemplifica la manera en que enseñamos y escribimos acerca de la programación, y es el enfoque de nuestros cursos multimedia *Cyber Classrooms* y de capacitación basados en Web. A este método de enseñar y escribir le llamamos **Método del código activo** (o ***Método LIVE-CODE™***). *Utilizamos lenguajes de programación para enseñar lenguajes de programación.* Leer los ejemplos en el texto es muy parecido a escribirlos y ejecutarlos en la computadora. Tanto en el CD que acompaña al libro como en `www.deitel.com` encontrará todo el código fuente para los ejemplos que aparecen en el libro. Le recomendamos ejecutar todos los ejemplos.

¡Programación en Java con aplicaciones y Swing desde el capítulo dos!

Este libro lo pasa directamente a la programación de aplicaciones Java con los componentes de la GUI Swing del capítulo 2. Hay mucho que hacer en Java, así que ¡manos a la obra! Definitivamente Java no es algo trivial, pero es divertido programar en este lenguaje y los estudiantes pueden ver resultados inmediatos. Además, pueden ejecutar rápidamente programas gráficos, animados, basados en multimedia, con uso intensivo de audio, con subprocesamiento múltiple, con uso intensivo de bases de datos, o programas basados en red, a través de las extensas bibliotecas de clases de Java que cuentan con componentes reutilizables. Los alumnos pueden implementar proyectos impresionantes. Por lo general son más creativos y productivos en un curso de uno o dos semestres que en cursos de introducción a C y C++.

2. Gamma, Erich, Richard Helm, Ralph Johnson y John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. (Massachusetts: Addison-Wesley, 1995.)

Acceso a World Wide Web

Todos los ejemplos de código fuente (en inglés) para *esta quinta edición* (y demás publicaciones de los autores) están disponibles en Internet, de manera que pueden descargarse del siguiente sitio Web:

```
www.deitel.com
www.pearsoneducacion.net/deitel
```

Puede registrarse en forma rápida y sencilla, y las descargas son gratuitas. Hacer cambios a los ejemplos e inmediatamente ver los efectos de esos cambios es una excelente manera de mejorar su aprendizaje de Java.

Objetivos

Cada capítulo comienza con objetivos que informan a los estudiantes lo que deben esperar, y les brinda la oportunidad, después de leer el capítulo, de determinar si cumplieron los objetivos planteados. Los objetivos sirven para crear confianza.

Frases

Los objetivos de los capítulos van seguidos de una o más frases. Algunas son graciosas, otras, filosóficas y algunas más ofrecen ideas interesantes. Hemos descubierto que los estudiantes disfrutan relacionando las frases con el material del capítulo. Muchas de éstas merecen leerse otra vez, *después* de leer los capítulos.

Plan general

El plan general de cada capítulo permite a los estudiantes abordar el material de manera ordenada. Junto con los objetivos de cada capítulo, el plan general ofrece a los estudiantes un panorama de los temas a tratar para establecer un ritmo cómodo y efectivo de aprendizaje.

23,341 líneas de código en 219 programas de ejemplo (con la salida de los programas)

Presentamos las características de Java en un contexto de programas completos y funcionales en Java. Estos programas LIVE-CODE ™ varían en longitud: desde unas cuantas líneas de código hasta ejemplos importantes con cientos de líneas de código. Cada programa va seguido por una ventana que contiene los resultados que se producen al ejecutar dicho programa; de manera que los estudiantes puedan confirmar que los programas funcionen como se esperaba. El proceso de comparar las operaciones de salida con las instrucciones del programa que producen esas operaciones es una excelente manera de aprender y reforzar los conceptos. Nuestros programas están diseñados para ejercitar las diversas características de Java.

615 Ilustraciones/Figuras

Incluimos una gran cantidad de gráficas, dibujos lineales, programas y salidas de programa. Hemos convertido todos los diagramas de flujo en diagramas de actividad en UML. También utilizamos diagramas de clase en UML para modelar las relaciones existentes entre las clases a lo largo de este texto.

534 Tips de programación

Hemos incluido tips de programación para ayudar a los estudiantes a enfocarse en los aspectos importantes del desarrollo de programas. Resaltamos cientos de estos tips como *Buenas prácticas de programación*, *Errores comunes de programación*, *Tips para prevenir errores*, *Observaciones de apariencia visual*, *Tips de rendimiento*, *Tips de portabilidad* y *Observaciones de ingeniería de software*. Estos tips y prácticas representan lo mejor que hemos podido recabar a lo largo de seis décadas (combinadas) de experiencia en la programación y la enseñanza. Uno de nuestros estudiantes, especialista en matemáticas, recientemente nos comentó que siente que este método es similar al de resaltar axiomas, teoremas y corolarios en los libros de matemáticas, ya que proporciona una base sólida sobre la cual se puede construir buen software.

82 Buenas prácticas de programación

Las Buenas prácticas de programación son tips que llaman la atención hacia técnicas para escribir programas claros. Además, estas técnicas ayudan a los estudiantes a producir programas más legibles, más fáciles de entender y mantener.

156 Errores comunes de programación

Los estudiantes que están aprendiendo un lenguaje (especialmente en su primer curso de programación) tienden con frecuencia a cometer ciertos tipos de errores. Al poner atención en estos Errores comunes de programación se reduce la probabilidad de que los estudiantes cometan los mismos errores. ¡También hace las líneas más cortas afuera de las oficinas de los profesores!

50 Tips para prevenir errores

Cuando diseñamos por primera vez este "tipo de tip", pensamos que lo usaríamos estrictamente para decir a las personas cómo probar y depurar programas en Java. De hecho, muchos de los tips describen aspectos de Java que reducen la probabilidad de "errores", lo que por consecuencia simplifica los procesos de prueba y depuración.

36 Observaciones de apariencia visual

Le ofrecemos Observaciones de apariencia visual para resaltar las convenciones de la interfaz gráfica de usuario. Estas observaciones ayudan a los estudiantes a diseñar sus propias interfaces gráficas de usuario en conformidad con las normas de la industria.

52 Tips de rendimiento

A lo largo de nuestra experiencia, enseñar a los estudiantes a escribir programas claros y comprensibles es, sin duda, la meta más importante para el primer curso de programación. Pero los estudiantes quieren escribir los programas que sean más rápidos, que usen la menor cantidad de memoria, requieran el menor número de pulsaciones de tecla, o que destaquen de alguna otra manera. Los estudiantes realmente se preocupan por el rendimiento. Quieren saber qué es lo que pueden hacer para mejorar el rendimiento de sus programas. Es por ello que nosotros resaltamos las oportunidades para mejorar el rendimiento de los programas; hacer que los programas se ejecuten más rápido o minimizar la cantidad de memoria que ocupan.

23 Tips de portabilidad

Una de las características que supuestamente han llevado a Java a la fama es su portabilidad "universal", por lo que algunos programadores asumen que si implementan una aplicación en Java, ésta será "perfectamente" portable a través de todas las plataformas Java, en forma automática. Desafortunadamente no siempre es el caso. Incluimos Tips de portabilidad para ayudar a los estudiantes a escribir código portable, y para ofrecer ideas acerca de cómo puede Java lograr su alto nivel de portabilidad. Cómo programar en Java tiene menos de estos tips que nuestros libros, Cómo programar en C y Cómo programar en C++ debido a que Java está diseñado para ser portable de arriba abajo; y para lograr esta portabilidad, se requiere de un menor esfuerzo por parte del programador de Java que del que trabaja con C o C++.

135 Observaciones de ingeniería de software

El paradigma de la programación orientada a objetos requiere de una completa reconsideración sobre la manera en que construimos los sistemas de software. Java es un lenguaje efectivo para lograr una buena ingeniería de software. Las Observaciones de ingeniería de software resaltan los asuntos de arquitectura y diseño, lo cual afecta la construcción de los sistemas de software, especialmente los de gran escala. Mucho de lo que los estudiantes aprendan aquí les será útil tanto en cursos de nivel superior como en la industria, a medida que empiecen a trabajar con sistemas grandes y complejos del mundo real.

Resumen (954 puntos de resumen)

Cada capítulo termina con elementos pedagógicos adicionales. Presentamos un resumen completo del capítulo, en forma de lista. Cada capítulo contiene un promedio de 38 puntos de resumen. Esto ayuda a los estudiantes a revisar y reforzar los conceptos clave.

Terminología (2166 términos)

En la sección *Terminología* incluimos una lista en orden alfabético de los términos importantes definidos en el capítulo; de nuevo, esto sirve como un refuerzo adicional. Cada capítulo contiene un promedio de 87 términos.

437 ejercicios de autoevaluación y sus respuestas (la cuenta incluye partes separadas)

En esta sección incluimos diversos ejercicios de autoevaluación así como sus respuestas, para que los estudiantes practiquen el autoestudio. Esto brinda la oportunidad de familiarizarse con el material y prepararse para intentar resolver los ejercicios regulares. Se debe alentar a los estudiantes para que hagan todos los ejercicios de autoevaluación y comprueben sus respuestas.

858 ejercicios (la cuenta incluye partes separadas)

Cada capítulo concluye con un conjunto de ejercicios que incluyen un sencillo recordatorio de terminología y conceptos importantes; escritura de instrucciones individuales en Java; escritura de pequeñas porciones de métodos y clases en Java; escritura de métodos, clases, aplicaciones y subprogramas completos en Java; y escritura de proyectos finales. El gran número de ejercicios, a lo largo de una extensa variedad de áreas, permite a los profesores diseñar sus cursos de acuerdo con las necesidades específicas de sus audiencias, y modificar las asig-

naturas del curso cada semestre. Los profesores pueden usar estos ejercicios para asignar tareas, realizar exámenes cortos y los exámenes principales. Las soluciones para la mayoría de los ejercicios se incluyen en el *CD de recursos para el instructor*, el cual *está disponible sólo para profesores*, a través de los representantes de Pearson Educación. [**NOTA: No nos escriba para solicitarnos el CD del instructor. La distribución de este accesorio está limitada estrictamente a profesores universitarios que tienen este libro como texto de sus cursos. Los profesores sólo pueden obtener el manual de soluciones a través de los representantes de esta empresa.**] Los estudiantes y lectores pueden obtener la solución de aproximadamente la mitad de los ejercicios si compran la quinta edición del *Java 2 Multimedia Cyber Classroom*, 5ª edición, producto en inglés que ofrece además muchas otras opciones muy valiosas, y es ideal ya sea como autoestudio o referencia. También encontrará disponible en nuestro sitio Web (`www.deitel.com`) otros productos de apoyo a la quinta edición de este libro, los cuales están disponibles en tiendas de libros en línea como `www.informIT.com`. Si usted ya tiene el libro de texto, puede comprar el producto *Java 2 Multimedia Cyber Classroom, 5ª edición* (ISBN# 0-13-101769-1) por separado en `www.InformIT.com/cyberclassrooms`.

Bibliografía

Se incluye en este libro una extensa bibliografía de libros, artículos y documentación de Java 2 por parte de Sun Microsystems, para que el lector pueda profundizar en los temas aquí descritos.

Software incluido con este libro

Hay una variedad de herramientas de desarrollo en Java, disponibles para su venta. Aunque no son necesarias para empezar a trabajar con Java. Escribimos este libro utilizando solamente el *Kit de desarrollo de software Java 2 Standard Edition (J2SDK), versión 1.4.1*. Para su conveniencia, este kit se incluye en el CD del libro. Siempre hay una versión actualizada del J2SDK que puede descargarse del sitio Web Java de Sun en `java.sun.com/j2se`. De este sitio también se puede descargar la documentación del J2SDK.

Gracias a la cooperación de Sun, también pudimos incluir en el CD un poderoso entorno de desarrollo integrado (IDE) de Java: *SunONE™ Studio 4, Community Edition*. Éste es un IDE profesional escrito en Java, que incluye un diseñador de interfaz gráfica de usuario, editor de código, compilador, depurador visual y mucho más. Hay que instalar el J2SDK primero, antes de instalar *Sun ONE™ Studio 4 Community Edition*. Si tiene preguntas acerca del uso de este software puede leer la documentación que viene en el CD, o puede también leer la publicación, en inglés, *Dive Into Sun ONE Studio 4, Community Edition* de la serie *DIVE-INTO™*. Este documento está disponible en la página `www.deitel.com/books/downloads.html`, junto con los recursos para esta quinta edición.

El CD contiene los ejemplos del libro (incluyendo la implementación del ejemplo práctico del elevador) y una página Web HTML con vínculos a los sitios Web de Deitel & Associats, Inc.. Si usted cuenta con acceso a Internet, puede cargar esta página Web en su navegador para que obtenga un rápido acceso a todos los recursos. Además incluimos varios capítulos y apéndices de otras publicaciones Deitel. Se incluye material sobre XHTML y Hojas de estilo en cascada (para usarse con el capítulo 24, Servlets y con el capítulo 25, JavaServer Pages™). Además se incluye material sobre el Lenguaje de marcado extendible (XML) y las APIs de Java para procesar XML, que ahora forman parte del J2SE 1.4.

Accesorios para la quinta edición de Cómo programar en Java

Este libro cuenta con diversos accesorios, todo en inglés, para los profesores. El CD llamado *Instructor's Resource CD (IRCD)* contiene el *Manual del instructor* con las soluciones a la gran mayoría de los ejercicios que se encuentran al final de cada capítulo, y un archivo llamado *Test Item File* con preguntas de opción múltiple (aproximadamente dos por cada sección del libro). Además incluye diapositivas de PowerPoint con todo el código y las figuras del libro, así como una lista de los elementos que resumen los puntos clave del texto. Los profesores pueden adaptar las diapositivas según sus requerimientos. Estas diapositivas de PowerPoint pueden descargarse del sitio `www.deitel.com`, donde se ofrecen recursos tanto para profesores como para alumnos. Para los profesores, el *Sitio Web Companion* ofrece material (*Syllabus Manager*) que les ayudará a planear los cursos en forma interactiva, y a crear programas de estudios en línea.

Los estudiantes también pueden beneficiarse de la funcionalidad del *Sitio Web Companion*. Los recursos que se incluye para este libro son:

- Diapositivas adaptables de PowerPoint®.
- Ejemplos de código fuente.
- Materiales de referencia de los apéndices del libro (como una tabla de precedencia de operadores, conjunto de caracteres y recursos Web).

Los recursos de cada capítulo del libro para los estudiantes son:

- Objetivos de cada capítulo.
- Elementos destacados (por ejemplo, resumen del capítulo).
- Plan general.
- Tips (por ejemplo, *Errores comunes de programación*, *Tips para prevenir errores*, *Buenas prácticas de programación*, *Observaciones de apariencia visual*, *Tips de portabilidad*, *Tips de rendimiento* y *Observaciones de ingeniería de software*).
- Guía de estudio en línea, la cual contiene ejercicios adicionales de autoevaluación que incluye las respuestas cortas (por ejemplo, verdadero/falso) y proporciona una retroalimentación inmediata al estudiante.

El *Sitio Web Companion* con todo el material anterior, en idioma inglés, se encuentra en `www.pearsoneducacion.net/deitel`.

Java in the Lab

Este manual de laboratorio (cuyo título completo es *Java in the Lab, Lab Manual to Accompany Java How to Program, 5ª edición*; ISBN 0-13-101631-8[3]) complementa a este libro (*Cómo programar en Java*), y al producto opcional *Java 2 Multimedia Cyber Classroom, 5ª edición*, mediante una serie de tareas de laboratorio diseñadas para reforzar la comprensión del estudiante en cuanto al material de lectura. Este manual de laboratorio está diseñado para laboratorios cerrados, que consisten en clases programadas en forma regular y supervisadas por un profesor. Los laboratorios cerrados ofrecen un excelente ambiente para el aprendizaje, ya que los estudiantes pueden utilizar los conceptos presentados en clase para resolver problemas de laboratorio cuidadosamente diseñados. Así, los profesores tienen más posibilidades de evaluar la comprensión de sus estudiantes con respecto al material, supervisando el progreso de éstos en el laboratorio. Este manual de laboratorio puede usarse también para laboratorios abiertos, tareas y autoestudio.

Java in the Lab se enfoca en los capítulos 1 a 12, 15 y 17 de este libro. Cada capítulo del manual se divide en 3 partes: *Actividades previas al laboratorio (Prelab Activities)*, *Ejercicios de laboratorio (Lab Exercises)* y *Actividades posteriores al laboratorio (Postlab Activities)*.[4] Cada capítulo contiene objetivos que introducen los temas clave del laboratorio, junto con una lista de asignaciones que permite a los estudiantes marcar cuáles ejercicios ha asignado el profesor. Cada página del manual del laboratorio está perforada, de forma que los estudiantes puedan entregar sus respuestas (en caso de ser necesario). Nuevamente, es prudente mencionar que todo el material complementario mencionado en este prefacio se encuentra en idioma inglés.

Las soluciones a los ejercicios de las secciones antes mencionadas se encuentran en formato electrónico. Los profesores pueden obtener estos materiales si los solicitan a los representantes de esta editorial; las soluciones no están disponibles para los estudiantes.

---

3. *Cómo programar en Java, 5ª edición*, y el manual de laboratorio también están disponibles en conjunto a través de un paquete especial (ISBN# 0-13-102719-0).
4. Las prácticas de los capítulos 12, 15 y 17 no contienen los extensos conjuntos de actividades disponibles en los capítulos anteriores. Sin embargo, los profesores podrán llevar a cabo las prácticas de laboratorios en forma efectiva, utilizando los ejercicios que hemos incluido sobre estos temas más complejos. Los profesores con requerimientos especiales pueden escribir a `deitel@deitel.com`.

Actividades previas al laboratorio

Se pretende que los estudiantes resuelvan las actividades de la sección *Prelab Activities* después de estudiar cada capítulo del presente libro. Estas actividades evalúan la comprensión de los estudiantes en cuanto al material presentado en el libro de texto, y los prepara para los ejercicios de programación de la sesión de laboratorio. Los ejercicios se enfocan en terminología y conceptos de programación importantes, y son efectivos para autoevaluación. Las actividades previas al laboratorio incluyen *Ejercicios de asociación*, *ejercicios de llenar espacios en blanco*, *preguntas de respuestas cortas*, *ejercicios de programación* (éstos solicitan a los estudiantes que determinen qué es lo que hacen pequeños segmentos de código, sin necesidad de ejecutar el programa) y *ejercicios de corrección de código* (éstos piden a los estudiantes que identifiquen y corrijan todos los errores en pequeños segmentos de código).

Ejercicios de laboratorio

La sección más importante de cada capítulo es la que corresponde a *Lab Exercises*. Estos ejercicios enseñan a los estudiantes cómo aplicar lo aprendido en *Cómo programar en Java, 5ª edición*, y los preparan para escribir programas en Java. Cada laboratorio contiene uno o más ejercicios y un problema de depuración. Los ejercicios de laboratorio contienen lo siguiente:

- Los *objetivos del laboratorio* (Lab Objectives) resaltan los conceptos específicos sobre los cuales se enfoca el ejercicio de laboratorio.
- Las *descripciones del problema* (Problem Descriptions) proporcionan los detalles del ejercicio y sugerencias que ayudan a los estudiantes a implementar el programa.
- Las *salidas de muestra* (Sample Outputs) ilustran el comportamiento esperado del programa, el cual favorece a la descripción de los problemas y ayuda a los estudiantes a escribir sus programas.
- Las *plantillas de programa* (Program Templates) toman los programas completos de Java y reemplazan líneas clave de código con comentarios que describen el código faltante.
- Los *tips para la solución del problema* (Problem-Solving Tips) resaltan los puntos clave que los estudiantes necesitan considerar cuando resuelven los ejercicios del laboratorio.
- Las *preguntas y actividades de seguimiento* (Follow-Up Questions and Activities) piden a los estudiantes que modifiquen las soluciones a los ejercicios, que escriban nuevos programas que sean similares a las soluciones de sus ejercicios de laboratorio o que expliquen las opciones de implementación que se eligieron cuando resolvieron los ejercicios de laboratorio.
- Los *problemas de depuración* (Debugging Problems) consisten de bloques de código que contienen errores de sintaxis y/o errores lógicos. Esto alerta a los estudiantes sobre los posibles tipos de errores que pueden encontrar durante el proceso de programación.

Actividades posteriores al laboratorio

Por lo general, los profesores asignan actividades posteriores al laboratorio para reforzar los conceptos importantes o para que los estudiantes adquieran más experiencia en programación fuera del laboratorio. La sección *Postlab Activities* evalúa el conocimiento del alumno sobre el material previo al laboratorio y los ejercicios del mismo, y pide al alumno que aplique el conocimiento adquirido para crear programas desde cero. Esta sección proporciona dos tipos de actividades de programación: ejercicios de código y retos de programación. Los ejercicios de código son cortos y sirven como repaso después de haber cumplido las *actividades previas al laboratorio* y los *ejercicios de laboratorio*. Estos ejercicios piden a los estudiantes que escriban programas o segmentos de programas que utilicen los conceptos importantes del libro. Los *retos de programación* permiten a los estudiantes aplicar el conocimiento que han adquirido en clase en ejercicios sustanciales de programación. Las sugerencias, las salidas de muestra y/o el pseudocódigo se proporcionan para ayudar a los estudiantes con estos problemas. Los estudiantes que completan de manera exitosa los *retos de programación* de un capítulo ciertamente han dominado el material del mismo. Las respuestas a los retos de programación están disponibles para su descarga en `www.deitel.com/books/downloads.html`.

Java 2 Multimedia Cyber Classroom, 5ª edición, y The Complete Java 2 Training Course, 5ª edición

Hemos actualizado nuestra versión multimedia interactiva opcional del libro —*The Java 2 Multimedia Cyber Classroom, 5ª edición* (CD para Windows®)— con bastante audio adicional, incluyendo el nuevo material sobre desarrollo de bases de datos con JDBC y desarrollo de aplicaciones Web con servlets y JavaServer Pages. Este recurso está cargado con características electrónicas que son ideales tanto para el aprendizaje como de referencia. El número ISBN del *Cyber Classroom* es 0-13-101769-1. Los *Cyber Classrooms* de Deitel™ por lo general están disponibles en CD y en distintos tipos de formatos de capacitación en Web.

El CD que viene con este manual proporciona una introducción en la cual los autores revisan las características del *Cyber Classroom*. Los 219 programas LIVE-CODE™ de ejemplo que vienen en el libro de texto, verdaderamente "cobran vida" en el *Cyber Classroom*. Si desea modificar un programa y ver el efecto de sus cambios, al hacer clic en el icono del disco flexible el código fuente "despega" del CD y "cae dentro" de uno de sus propios directorios, de manera que pueda editarlo, recompilar el programa y ejecutar su nueva versión. Además, el *Cyber Classroom* contiene el texto completo (en inglés) de *Cómo programar en Java, 5ª edición*, en un formato especial para búsqueda de texto.

Asimismo, el *Cyber Classroom* proporciona exámenes de autoevaluación (con respuestas) para cada capítulo del libro. Estos exámenes son poderosas características que permiten a los usuarios medir su comprensión de los conceptos de programación que presentan los capítulos. Cada pregunta del examen tiene un hipervínculo a la sección, dentro del libro, de donde se deriva la pregunta. Esto permite a los usuarios revisar el material apropiado del capítulo antes o después de responder a la pregunta. Además se proporciona un cuadro que presenta los resultados, por capítulo, de cada examen del usuario.

El *Cyber Classroom* se basa en formato estilo navegador, de manera que recuerda las secciones que usted visitó recientemente y le permite moverse hacia delante o hacia atrás entre ellas. Las miles de entradas del índice están hipervinculadas al texto correspondiente. Las entradas de la tabla de contenido son vínculos dinámicos, de manera que al hacer clic en el nombre de un capítulo o sección lo llevará de inmediato al texto de referencia.

A los estudiantes les fascinará el hecho de que el *Cyber Classroom* ofrece las soluciones de cerca de la mitad de los ejercicios del libro. Estudiar y ejecutar estos programas adicionales es una excelente manera de que los estudiantes mejoren su experiencia de aprendizaje con código activo.

Para una lista completa de los *Cyber Classrooms* y *Complete Training Courses* disponibles y futuros, visite el sitio Web de los autores en: `www.deitel.com` o `www.InformIT.com/deitel`.

Advanced Java™ 2 Platform How to Program

Nuestro libro complementario, en inglés, ***Advanced Java 2 Platform How to Program*** se enfoca en la ***Plataforma Java 2, Enterprise Edition (J2EE)***, y presenta características avanzadas de esta plataforma de Java e introduce la ***Plataforma Java 2, Micro Edition (J2ME)***. Este libro está destinado a los desarrolladores y estudiantes universitarios de nivel superior en cursos avanzados, que ya conocen Java y desean profundizar más sobre este lenguaje. Además presenta nuestro libro distintivo LIVE-CODE™, con programas funcionales completos y más de 37,000 líneas de código. Los programas son más complejos que los presentados en *Cómo programar en Java, 5ª edición*. Este libro expande la cobertura de la Conectividad de bases de datos en Java (JDBC), los servlets y las JavaServer Pages (JSP) expuestas en *Cómo programar en Java, 5ª edición*. También cubre las tecnologías emergentes y más avanzadas de Java, que son de interés para los desarrolladores de aplicaciones empresariales, incluyendo el Modelo-Vista-Controlador; Java 2D y Java 3D, Modelo de componentes JavaBeans; Seguridad; Java 2 Micro Edition (J2ME) e Internet inalámbrica; Llamada a métodos remotos (RMI); Enterprise JavaBeans (EJBs); Servicio de mensajes de Java (JMS); Jini; JavaSpaces; Jiro; Extensiones de gestión de Java (JMX); Arquitectura común del agente de petición de objetos (CORBA); Redes de igual a igual; Servicios Web; XML e Interfaz nativa de Java (JNI).

Java Web Services for Experienced Programmers

Este libro, que forma parte de la nueva serie *Deitel Developer* para profesionales de computación, utiliza nuestro conocido método LIVE-CODE™ para evaluar las más recientes tecnologías de XML y Java para crear e

integrar servicios Web, incluyendo el *Paquete para desarrolladores de Servicios Web en Java (Java Web Services Developer Pack)*. El libro está dirigido a los profesionales de la industria que requieren de una cobertura detallada sobre las tecnologías de servicios Web en Java. También es adecuado para los cursos de ciencias computacionales de nivel superior que tratan sobre los servicios Web, o como suplemento para los cursos en computación distribuida y programación avanzada en Java. Los temas que se cubren en este libro son: XML; Definiciones de tipos de documentos (DTDs); Modelo de objetos de documentos (DOM™) y la API de Java para procesamiento de XML (JAXP); Transformaciones de hojas de estilo extensibles (XSLT™) y las API de transformación para XML (TrAX); Protocolo simple de acceso a objetos (SOAP); Lenguaje de descripción de servicios Web (WSDL); Integramiento, descubrimiento y descripción universal (UDDI) y la API de Java para Registros de XML (JAXR); API de Java para Llamadas a procedimientos remotos basadas en XML (JAX-RPC); API de Java para Mensajería XML (JAXM) y la API SOAP con adjuntos para Java (SAAJ); Seguridad de los servicios Web con el Nivel de sockets seguro (SSL); Firma digital XML (XML Signature), Cifrado con XML (XML Encryption), Especificación de gestión de claves con XML (XKMS), Lenguaje de marcado de aserciones de seguridad (SAML) y Lenguaje de marcado de control de acceso extendible (XACML); y los Servicios Web inalámbricos con Java 2 Micro Edition (J2ME™).

La serie Course Management Systems: Blackboard™, WebCT™, CourseCompass℠ y Premium CourseCompass℠

Parte del contenido de la serie introductoria para lenguajes de programación *Cómo programar* de Deitel, incluyendo este libro, se ha integrado en varios de los populares textos de la serie Course Management Systems, como en Blackboard, CourseCompass y WebCT. Una versión mejorada de CourseCompass, llamada Premium Course Compass, estará disponible para la quinta edición de *Cómo programar en Java*. Estos textos ayudan en la creación, el manejo y el uso de sofisticadas herramientas y programas de educación basados en Web. Con ese material, creado por y para profesores, los profesores pueden ahorrarse horas de introducción de datos. Blackboard, CourseCompass and WebCT ofrecen:

- **Características para crear y personalizar cursos en línea**, como zonas para colocar información del curso (por ejemplo, políticas, planes de estudio, anuncios, asignaturas, grados, evaluaciones de rendimiento y seguimiento del proceso), herramientas de administración de clases y estudiantes, un libro de calificaciones, herramientas para reportes, rastreo de páginas, un calendario y asignaturas.
- **Herramientas de comunicaciones** para ayudar a crear y mantener las relaciones interpersonales entre los estudiantes y los profesores, incluyendo salones de conversación, pizarras electrónicas, documentación compartida, foros de discusión y correo electrónico privado.
- **Herramientas de evaluación flexibles** que permiten al profesor crear cuestionarios en línea y exámenes vinculados de manera directa al texto, y que califican y dan seguimiento a los resultados con efectividad. Todas las evaluaciones pueden introducirse en el libro de calificaciones para una eficiente administración del curso. Además, WebCT permite a los instructores administrar el tiempo de los cuestionarios en línea.
- El **Material de apoyo** para los instructores se encuentra disponible en formato para impresión y para consulta en línea.

Además de las herramientas que se encuentran en Blackboard y WebCT, CourseCompass incluye:

- **La página de inicio de CourseCompass**, la cual hace el curso tan fácil como navegar en este libro. Una tabla de contenido expandible que permite a los instructores ver el contenido del curso al instante y vincularse con cualquier sección.
- **Secciones de ayuda en línea "How Do I"**, están disponibles para los usuarios que requieren sitios con ayuda personalizada para los cursos, incluyendo las instrucciones paso a paso para agregar diapositivas de Power-Point®, vídeo y más.
- **Guía del instructor Quick Start**, ofrece a los profesores la opción de crear cursos en línea mediante un sencillo proceso, paso a paso.

Para demostraciones gratuitas en línea y aprender más acerca de la serie Course Management Systems, que cuenta con material de apoyo para este libro, visite las páginas Web:

- Blackboard: `www.blackboard.com`
- WebCT: `www.webct.com`
- CourseCompass: `www.coursecompass.com`

La nueva serie para desarrolladores (DEITEL™ Developer)

Deitel & Associates, Inc., está realizando un compromiso considerable por cubrir las tecnologías de punta para los profesionales de software de la industria, a través del lanzamiento de nuestra serie para desarrolladores *DEITEL™ Developer Series*. *Web Services A Technical Introduction* y *Java Web Services for Experienced Programmers* se encuentran entre los primeros libros en la serie. A estos libros le seguirán *Java 2 Enterprise Edition*, *Java 2 Micro Edition*, *.NET A Technical Introduction*, *ASP .NET with Visual Basic .NET for Experienced Programmers*, *ASP .NET with C# for Experienced Programmers* y muchos más. Por favor visite el sitio `www.deitel.com` para ver las actualizaciones en todos los títulos publicados y futuros de esta serie.

Paseo por el libro

Está a punto de estudiar uno de los lenguajes de programación más excitantes y con un rápido desarrollo en la actualidad. El dominio de Java le ayudará a desarrollar poderoso software de aplicaciones comercial y personal. En esta sección daremos un rápido vistazo a muchos de los temas y características de Java que veremos a lo largo de todo el libro.

Capítulo 1 "Introducción a las computadoras, Internet y Web": Habla sobre lo que son las computadoras, cómo trabajan y cómo se programan. Este capítulo presenta una breve historia sobre el desarrollo de los lenguajes de programación, partiendo desde los lenguajes máquina, pasando por los lenguajes ensambladores y terminando con los lenguajes de alto nivel. Se habla también sobre los orígenes del lenguaje de programación Java. El capítulo incluye una introducción a un entorno de programación común en Java. También presenta la tecnología de objetos, el Lenguaje de modelado unificado y los patrones de diseño.

Capítulo 2 "Introducción a las aplicaciones en Java": Ofrece una ligera introducción a la programación de *aplicaciones* en el lenguaje de programación Java. Este capítulo introduce los conceptos y construcciones básicas de programación a los no programadores. Los programas de este capítulo ilustran cómo mostrar datos en la pantalla para que los vea el usuario, y cómo obtener datos del usuario mediante el teclado. Parte de los procesos de entrada y salida se lleva a cabo mediante el uso de componentes de *interfaz gráfica de usuario (GUI)*. El capítulo 2 también ofrece un análisis detallado de la *toma de decisiones* y las *operaciones aritméticas*.

Capítulo 3 "Introducción a los subprogramas de Java": Presenta los *subprogramas* Java, que son programas en Java diseñados para transportarse a través de Internet y ejecutarse en navegadores Web (como Netscape Navigator y Microsoft Internet Explorer). Este capítulo muestra varios de los subprogramas de demostración que se incluyen con el J2SDK. Después escribiremos subprogramas de Java que llevan a cabo tareas similares a los programas del capítulo 2, y explicaremos las similitudes y diferencias entre los subprogramas y las aplicaciones.

Capítulo 4 "Instrucciones de control: Parte 1": Se enfoca en el proceso de desarrollo de programas. Este capítulo habla sobre cómo tomar un *enunciado del problema* y, a partir de éste, desarrollar un programa funcional en Java, incluyendo la realización de los pasos intermedios con *pseudocódigo*. También explica ciertos tipos primitivos de instrucciones simples de control para la toma de decisiones (`if` e `if...else`) y la repetición (`while`). También analizamos la repetición controlada por contador y la repetición controlada por centinela, y se introducen los operadores de incremento, decremento y asignación de Java. Este capítulo utiliza diagramas de actividad simples en UML para mostrar el flujo de control a través de cada una de las instrucciones de control.

Capítulo 5 "Instrucciones de control: Parte 2": Continúa hablando sobre las instrucciones de control de Java, con ejemplos de la instrucción de repetición `for`, la instrucción de repetición `do...while`, la instrucción de selección switch, las instrucciones break y continue. También se incluye una discusión sobre los operadores lógicos de Java.

Capítulo 6 "Métodos": Ofrece un análisis más detallado de los objetos. Los objetos contienen datos llamados campos y unidades ejecutables llamadas métodos. Aquí hablamos sobre los métodos de la biblioteca de

clases y también creamos nuestros propios métodos. Para los cursos de ciencias computacionales, este capítulo presenta también una discusión sobre la recursividad. Las técnicas que se presentan en el capítulo 6 son esenciales para producir programas correctamente estructurados, especialmente los programas más grandes que desarrollan los programadores de sistemas y de aplicaciones. El tema de la sobrecarga de métodos (es decir, permitir que varios métodos tengan el mismo nombre, siempre y cuando tengan distintas "firmas") se motiva y se explica claramente. También presentamos los *eventos* y el *manejo de eventos*.

Capítulo 7 "Arreglos": Explora las listas de procesamiento y las tablas de valores. Los arreglos en Java son objetos, una evidencia más del compromiso de Java con la orientación a objetos. En este capítulo hablamos sobre la estructuración de datos en arreglos de elementos de datos del mismo tipo. Aquí también se presentan numerosos ejemplos, tanto de los arreglos unidimensionales como de los multidimensionales. Los ejemplos en este capítulo investigan las manipulaciones comunes de los arreglos, la impresión de histogramas, el proceso de pasar arreglos a los métodos y una introducción al campo del análisis de datos de encuestas (con estadísticas simples). Una característica de este capítulo es la discusión de las técnicas elementales para ordenar y realizar búsquedas, junto con la presentación de la búsqueda binaria como una mejora dramática sobre la búsqueda lineal.

Capítulo 8 "Programación basada en objetos": Empieza nuestra discusión más detallada acerca de las clases. Este capítulo representa una maravillosa oportunidad para enseñar la abstracción de datos de la "forma correcta"; a través de un lenguaje (Java) dedicado específicamente a la implementación de nuevos tipos. Se enfoca en la esencia y la terminología de las clases y los objetos. Habla sobre la implementación de clases en Java, el acceso a los miembros de una clase, cómo implementar la ocultación de la información con modificadores de acceso, separar la interfaz de la implementación, usar métodos de acceso y métodos utilitarios e inicializar objetos con constructores. El capítulo habla también sobre cómo declarar y utilizar constantes, la *composición*, la referencia con `this`, los miembros de clase `static` y ejemplos de tipos de datos abstractos populares, como las pilas y las colas. También introduce la instrucción package y habla sobre cómo crear paquetes reutilizables.

Capítulo 9 "Programación orientada a objetos: Herencia": En este capítulo se explica la herencia, una de las herramientas fundamentales de los lenguajes de programación orientados a objetos, la cual es una forma de reutilización de software en la que nuevas clases se desarrollan rápida y fácilmente al absorber las características de las clases existentes y al agregar nuevas características que sean apropiadas. También habla sobre las nociones de superclases y subclases, el modificador de acceso `protected`, las superclases directas e indirectas, el uso de constructores en superclases y subclases, y la ingeniería de software con la herencia. El capítulo compara la herencia (relaciones "es un") con la composición (relaciones "tiene un").

Capítulo 10 "Programación orientada a objetos: Polimorfismo": Este capítulo habla sobre otra herramienta fundamental de la programación orientada a objetos, mejor conocida como comportamiento polimórfico. Este estilo de programación se utiliza generalmente para implementar los populares marcos de trabajo de GUI, como Swing de Java. Este capítulo hace la distinción entre clases abstractas (`abstract`) y clases concretas, además de presentar las interfaces: el reemplazo de Java para la peligrosa (aunque poderosa) característica de C++ conocida como herencia múltiple. También presenta el poderoso concepto de *clases anidadas*, las cuales ayudan a ocultar los detalles de implementación. Luego, el capítulo demuestra nuestras primeras aplicaciones basadas en GUI, como parte de una introducción más completa al manejo de eventos. En esta sección utilizamos clases anidadas para definir los manejadores de eventos para varios componentes de GUI. Una característica notable de este capítulo es que cuenta con tres ejemplos prácticos de polimorfismo: un sistema de nóminas, una jerarquía de figuras encabezada por una clase `abstract` y una jerarquía de clases encabezada por una interfaz.

Capítulo 11 "Cadenas y caracteres": Este capítulo habla sobre el procesamiento de palabras, oraciones, caracteres y grupos de caracteres. Presentamos las clases String, `StringBuffer`, `Character` y `StringTokenizer`. También presentamos la API de Java para expresiones regulares (nueva en la J2SE 1.4), la cual permite a los programas buscar en las cadenas secuencias de caracteres que concuerden con ciertos patrones especificados.

Capítulo 12 "Gráficos y Java2D": Éste es el primero de varios capítulos en los que se presentan las herramientas gráficas y de multimedia de Java. Hablamos sobre los contextos y los objetos gráficos; el dibujo de cadenas, caracteres y bytes; el control de color y tipo de letra; la manipulación de la pantalla, los modos de pantalla y el dibujo de líneas, rectángulos, rectángulos redondeados, rectángulos tridimensionales, óvalos, arcos y polígonos. Presentamos la API Java2D, la cual ofrece poderosas herramientas para gráficos. La figura 12.22 es un ejemplo de lo fácil que es utilizar la API Java2D para crear efectos con gráficos complejos, como texturas y degradados.[5]

Capítulo 13 "Componentes de la interfaz gráfica de usuario, Parte 1": Este capítulo presenta varios de los *componentes Swing* de Java para crear programas con interfaces gráficas de usuario amigables (GUIs). Estos componentes de GUI *independientes de la plataforma* están escritos completamente en Java, lo que les proporciona una gran flexibilidad. Los componentes Swing pueden personalizarse para asemejarse a la apariencia de la plataforma computacional en la que se ejecute el programa, o pueden usar la apariencia visual estándar de Java para brindar una interfaz de usuario idéntica en todas las plataformas computacionales. El desarrollo de GUIs es un tema extenso, por lo que lo dividimos en dos capítulos. Estos capítulos cubren el material con suficiente detalle como para permitirle a usted crear interfaces de usuario atractivas y funcionales. Este capítulo ilustra los principios de las GUIs, la jerarquía `javax.swing`, las etiquetas, botones, listas, campos de texto, cuadros combinados, casillas de verificación, botones de opción, paneles, el manejo de eventos de ratón y de teclado, y los administradores de esquemas para posicionar componentes. El capítulo también hace énfasis en el manejo de eventos.

Capítulo 14 "Componentes de la interfaz gráfica de usuario: Parte 2": Este capítulo continúa con el tema sobre Swing que se trató en el capítulo 13. A través de sus programas, tablas y dibujos lineales, el capítulo ilustra los principios de diseño de GUIs, las áreas de texto, cómo extender los componentes Swing, los controles deslizables, las ventanas, los menús, menús contextuales, cómo cambiar la apariencia visual, las interfaces con documentos múltiples, los paneles con fichas y el uso de administradores de esquemas avanzados.[6]

Capítulo 15 "Manejo de excepciones": Éste es uno de los capítulos más importantes en el libro, desde el punto de vista de la creación de aplicaciones de "misión crítica" o de "comercio crítico". Los programadores necesitan hacerse cargo de "¿qué ocurre cuando el componente que llamo para hacer un trabajo presenta dificultades? ¿Cómo me indicará ese componente que tuvo un problema?" Para utilizar un componente de Java, es necesario saber no sólo la manera como se comporta ese componente cuando "las cosas salen bien", sino también qué *excepciones lanza* ese componente cuando "las cosas salen mal". Este capítulo hace la distinción entre los `errores` del sistema, que son bastante graves, y las `excepciones`. También habla sobre el vocabulario del manejo de excepciones, incluyendo los *bloques* `try`, *las cláusulas* `catch` y `finally`; además, introduce la nueva capacidad de encadenamiento de excepciones en la J2SE 1.4. El material de este capítulo es imprescindible para muchos de los ejemplos en el resto del libro.

Capítulo 16 "Subprocesamiento múltiple": Este capítulo habla sobre cómo desarrollar programas en Java que puedan llevar a cabo varias actividades en forma concurrente. Las computadoras solían construirse con un solo procesador, bastante costoso. Actualmente, los procesadores son cada vez más baratos, de manera que es posible construir computadoras con varios procesadores que trabajen en paralelo; dichas computadoras se conocen como *multiprocesadores*. La tendencia se inclina claramente hacia las computadoras que pueden realizar estas funciones. Como veremos, el subprocesamiento múltiple es efectivo aún en sistemas con un solo procesador. El capítulo presenta programas con subprocesamiento múltiple, los cuales muestran los problemas que pueden ocurrir en la programación concurrente. Una característica del capítulo es su extenso conjunto de ejemplos que muestran este tipo de problemas y cómo resolverlos. También se abordan los subprocesos y los métodos de los mismos. Avanza a través de los diversos estados de un subproceso y las transiciones de estado con una representación gráfica del ciclo de vida de un subproceso. Asimismo, discurre sobre las prioridades de los subprocesos y cómo programar su ejecución. Analiza una relación productor/consumidor sin sincronización, se observan los problemas que ocurren y se soluciona el problema con la sincronización de los subprocesos. Implementa una relación productor/consumidor con un búfer circular, y una sincronización adecuada con un monitor. Habla sobre los subprocesos tipo "demonio", que se la pasan "esperando" y llevan a cabo tareas cuando hay tiempo disponible en el procesador. Finalmente, otro punto que se trata es sobre la interfaz `Runnable`, que permite a los objetos ejecutarse como subprocesos sin tener que usar la clase `Thread` como subclase.

---

5. Nuestro libro complementario, *Advanced Java 2 Platform How to Program*, presenta la API 3D, útil para crear efectos tridimensionales.
6. Los capítulos 1 a 6 de *Advanced Java 2 Platform How to Program*, ofrece conceptos de la GUI más avanzados.

Capítulo 17 "Archivos y flujos": Este capítulo trata acerca de la entrada/salida que se logra a través de flujos de datos dirigidos hacia, y provenientes desde, los archivos. En este capítulo se traducen los objetos a un formato persistente. El proceso de poder almacenar datos en archivos o desplazarlos a través de redes (capítulo 18) hace que los programas puedan guardar datos y comunicarse entre sí. Este capítulo empieza con una introducción a la jerarquía de datos, comenzando desde los bits, después los bytes, los campos, los registros y los archivos. A continuación se presenta un análisis simple en Java de los archivos y flujos. Mostramos cómo los programas pasan datos a los dispositivos de almacenamiento secundario como los discos, y la forma en que recuperan los datos almacenados en esos dispositivos. También hay una discusión sobre la clase `File`, que es utilizada por los programas para obtener información acerca de archivos y directorios. Se explica el método por el cual los objetos pueden enviarse hacia, y recibirse desde, los dispositivos de almacenamiento secundario. También hay un apartado sobre las Nuevas APIs de E/S de alto rendimiento (introducidas en la J2SE 1.4).

Capítulo 18 "Redes": Este capítulo habla sobre los programas en Java que se comunican a través de redes de computadoras. Presenta las herramientas de red de menor nivel en Java. Los ejemplos de este capítulo muestran a un subprograma interactuando con el navegador en el que se ejecuta, creando un mini navegador Web, comunicándose entre dos programas en Java mediante sockets basados en flujos, y comunicándose entre dos programas en Java mediante paquetes de datos. Una característica clave de este capítulo es la implementación de un juego de Tres en línea que utiliza la arquitectura cliente/servidor en colaboración, en donde dos clientes juegan Tres en línea uno con el otro, y el juego es arbitrado por un servidor con subprocesamiento múltiple. ¡Estupendo! El ejemplo culminante en este capítulo es el del Deitel Messenger, que simula a muchas de las aplicaciones de mensajería instantánea populares hoy en día, las cuales permiten a los usuarios de computadora comunicarse con amigos, parientes y compañeros de trabajo a través de Internet. Este ejemplo práctico de cliente/servidor, con 1,130 líneas y subprocesamiento múltiple, utiliza la mayoría de las técnicas de programación presentadas hasta este punto. Este capítulo también continúa con el tema sobre las nuevas APIs de E/S de la J2SE 1.4, con una introducción a los selectores y la E/S sin bloqueo, para implementar servidores de red con alto rendimiento.[7]

Capítulo 19 "Multimedia: imágenes, animación y audio": Presenta algunas de las herramientas de Java para hacer que los programas cobren vida a través de la multimedia. Este capítulo habla sobre las imágenes y la manipulación de éstas, el audio y la animación. También presenta una aplicación LIVE-CODE de mapa de imágenes, con los iconos de los tips de programación que se mostraron antes en este prefacio, y que aparecen en todo el libro. A medida que el usuario mueve el puntero del ratón por los iconos, se va mostrando el tipo de tip. Una vez que haya leído el capítulo, estará ansioso por probar todas estas técnicas, por lo que se han incluido muchos ejercicios para retarlo y entretenerlo.

Capítulo 20 "Estructuras de datos": Es especialmente valioso en cursos superiores de segundo y tercer nivel. Este capítulo versa sobre las técnicas utilizadas para crear y manipular estructuras dinámicas de datos, como listas enlazadas, pilas, colas y árboles. Para cada tipo de estructura de datos, se presentan ejemplos con muestras de salida. Aunque es valioso saber cómo se implementan estas clases, los programadores de Java descubrirán rápidamente que la mayoría de las estructuras de datos que necesitan están disponibles en las bibliotecas de clases, como la biblioteca `java.util` de Java que describimos en los capítulos 21 y 22.

Capítulo 21 "Paquete de utilerías de Java y manipulación de bits": Presenta varias clases de la biblioteca `java.util` y habla sobre los operadores de manipulación de bits de Java. Una clase especialmente útil es `Vector`: un arreglo dinámico que puede crecer y reducirse según sea necesario. También hablamos sobre las clases `Stack`, `Hashtable`, `Properties`, `Random` y `BitSet`.

Capítulo 22 "Colecciones": Habla sobre las clases de `java.util` que pertenecen a la API Collections, y que ofrecen implementaciones predefinidas de muchas de las estructuras de datos descritas en el capítulo 20. Las colecciones ofrecen a los programadores de Java un conjunto estándar de estructuras de datos para almacenar y recuperar datos, y un conjunto estándar de algoritmos (es decir, procedimientos) que permiten a los programadores manipular esos datos (como buscar ciertos elementos de datos y ordenar los datos en forma ascendente o descendente). Este capítulo demuestra las colecciones tales como listas enlazadas, árboles, mapas y conjuntos, y los algoritmos para buscar, ordenar, encontrar el valor máximo, el mínimo y así, sucesivamente.

7. Nuestro libro complementario, *Advanced Java 2 Platform How to Program*, ofrece un análisis mucho más profundo de las redes y la computación distribuida, con temas que incluyen la invocación a métodos remotos (RMI), Java 2 Enterprise Edition, Java inalámbrico (y Java 2 Micro Edition) y CORBA (Common Object Request Broker Architecture, Arquitectura común de agente de solicitud de objetos).

Capítulo 23 "Conectividad de bases de datos en Java con JDBC™": Habla sobre el soporte que ofrece Java para usar bases de datos. Los sistemas de bases de datos más populares hoy en día son las bases de datos relacionales. Presentamos ejemplos utilizando Cloudscape de IBM, un sistema de administración de bases de datos creado completamente en Java. Este capítulo introduce la JDBC, utilizándola para conectarse con una base de datos Cloudscape y después manipular su contenido. Utilizamos el Lenguaje de consulta estructurado (SQL) para extraer información de, e insertar información en, una base de datos. Los siguientes capítulos sobre servlets y JavaServer Pages utilizan las técnicas que se muestran en este capítulo para crear aplicaciones Web controladas por datos.

Capítulo 24 "Servlets": Habla sobre los servlets, que generalmente extienden la funcionalidad de los servidores Web. Los servlets son efectivos para desarrollar soluciones basadas en Web que interactúan con bases de datos a beneficio de los clientes, generan contenido personalizado en forma dinámica para que lo muestren los navegadores, y mantienen información de sesión única por cada cliente. La API para servlets de Java permite a los desarrolladores aumentar la funcionalidad de los servidores Web para hacerse cargo de las peticiones de los clientes. Los servlets también pueden reutilizarse en la mayoría de los servidores Web y plataformas. Este capítulo muestra el mecanismo de petición/respuesta de Web (principalmente con las peticiones `get` y `post` de HTTP), redireccionando las peticiones hacia otros recursos e interactuando con las bases de datos a través de JDBC. El capítulo presenta una aplicación cliente/servidor de tres niveles que da seguimiento a las respuestas de un usuario en una encuesta.

Capítulo 25 "JavaServer Pages (JSP)": Presenta una extensión de la tecnología de servlets conocida como JavaServer Pages (JSP). Las JSPs permiten el uso de contenido Web generado en forma dinámica, y son utilizadas principalmente por los diseñadores Web y todos los que no están familiarizados con la programación en Java. Las JSPs pueden contener código de Java en forma de scriptlets. Para aumentar el rendimiento, cada JSP se compila en un servlet de Java; esto normalmente ocurre la primera vez que un cliente solicita la JSP. Las peticiones subsecuentes de los clientes son atendidas por el servlet compilado. Este capítulo presenta una aplicación de libro de visitantes, tipo cliente/servidor de tres niveles, que almacena la información de los visitantes en una base de datos.

Apéndice A "Tabla de precedencia de operadores": Enlista cada uno de los operadores de Java e indica su precedencia relativa y su asociatividad.

Apéndice B "Conjunto de caracteres ASCII": Enlista los caracteres del conjunto ASCII (Código estándar estadounidense para el intercambio de información) e indica el valor para cada uno de ellos en código de carácter. Java utiliza el conjunto de caracteres Unicode, con caracteres de 16 bits, para representar todos los caracteres en los lenguajes "comercialmente considerables" en el mundo. Unicode incluye el código ASCII como un subconjunto.

Apéndice C "Sistemas numéricos": Habla sobre los sistemas numéricos binario (base 2), decimal (base 10), octal (base 8) y hexadecimal (base 16). Este material es valioso para los cursos introductorios en ciencias computacionales e ingeniería computacional.

Apéndices D al F: Contienen la implementación de nuestro ejemplo práctico sobre Diseño orientado a objetos con UML. Éste se discute en la descripción general del ejemplo práctico.

Apéndice G "Unicode": Habla sobre el conjunto de caracteres Unicode, el cual permite a Java mostrar la información en muchos lenguajes. Este apéndice incluye un programa Java de ejemplo que muestra "Bienvenido a Unicode" en varios lenguajes.

(Opcional) Un paseo por el ejemplo práctico sobre Diseño orientado a objetos con UML

En esta sección y en la siguiente daremos un paseo a través de las dos principales características opcionales del libro: el ejemplo práctico opcional de diseño orientado a objetos con UML y nuestra introducción a los patrones de diseño. El ejemplo práctico es una adición importante a *Cómo programar en Java, 5ª edición*. Este paseo presenta un avance del contenido de las secciones "Acerca de los objetos" y habla sobre cómo se relacionan con el ejemplo práctico. Después de realizar este ejemplo práctico, usted habrá completado el diseño orientado a objetos y la implementación para una aplicación importante en Java.

Sección 1.15, Acerca de los objetos: Introducción a la tecnología de objetos y el Lenguaje de modelado unificado

Esta sección introduce el ejemplo práctico de diseño orientado a objetos con UML. Aquí presentamos los antecedentes generales de lo que son los objetos y cómo interactúan entre sí. También hablamos brevemente sobre el estado de la industria de la ingeniería de software y la forma en que UML ha influenciado los procesos de análisis y diseño orientado a objetos.

Sección 2.9 (Ejemplo práctico opcional), Acerca de los objetos: Cómo examinar el enunciado de un problema

Nuestro ejemplo práctico comienza con un *enunciado de problema*, el cual especifica los requerimientos para el sistema que vamos a crear. En este ejemplo práctico estudiamos e implementamos la simulación de un sistema de elevador en un edificio de dos plantas. Proporcionamos el diseño de nuestro sistema de elevador después de investigar la estructura y el comportamiento de los sistemas orientados a objetos en general. Hablamos sobre cómo UML facilita el proceso de diseño en las siguientes secciones "Acerca de los objetos", al ofrecernos varios tipos de diagramas para modelar nuestro sistema. Por último, proporcionamos una lista de referencias de URLs y libros que tratan sobre el diseño orientado a objetos con UML. Tal vez le sean útiles estas referencias a medida que progrese a través de la presentación de nuestro ejemplo práctico.

Sección 3.7 (Ejemplo práctico opcional), Acerca de los objetos: Cómo identificar las clases en el enunciado de un problema

En esta sección empezamos a diseñar la simulación del elevador. Identificamos las clases o "bloques de construcción" de nuestra simulación, extrayendo los sustantivos y las frases nominales del enunciado del problema. Ordenamos estas clases en un diagrama de clases de UML, el cual describe la estructura de clases de nuestra simulación. El diagrama de clases también describe las relaciones, conocidas como *asociaciones*, entre las clases (por ejemplo, una persona tiene una asociación con el elevador, ya que ésta viaja en él).

Sección 4.14 (Ejemplo práctico opcional), Acerca de los objetos: Cómo identificar los atributos de una clase

Una clase contiene tanto *atributos* (datos) como *operadores* (comportamientos). Esta sección se enfoca en los atributos de las clases que se describieron en la sección 3.7. Como veremos en secciones posteriores, los cambios en los atributos de un objeto a menudo afectan al comportamiento de ese objeto. Para determinar los atributos para las clases en nuestro ejemplo práctico, extraemos los adjetivos que describen a los sustantivos y frases nominales (que definen a nuestras clases) del enunciado del problema y después colocaremos los atributos en el diagrama de clases que creamos en la sección 3.7.

Sección 5.11 (Ejemplo práctico opcional), Acerca de los objetos: Cómo identificar los estados y actividades de los objetos

Un objeto, en cualquier momento dado, ocupa una condición específica conocida como *estado*. Una *transición de estado* ocurre cuando ese objeto recibe un mensaje para cambiar de estado. UML cuenta con el *diagrama de estados*, el cual identifica el conjunto de posibles estados que puede ocupar un objeto y modela las transiciones de estado de ese objeto. Un objeto tiene también una *actividad*: el trabajo desempeñado por un objeto durante su vida. UML cuenta con el *diagrama de actividades*: un diagrama de flujo que modela la *actividad* de un objeto. En esta sección utilizamos ambos tipos de diagramas para empezar a modelar ciertos aspectos específicos del comportamiento de nuestra simulación de elevador, como la forma en que una persona viaja en el elevador y cómo éste responde cuando se oprime un botón en un piso dado.

Sección 6.15 (Ejemplo práctico opcional), Acerca de los objetos: Cómo identificar las operaciones de una clase

En esta sección identificamos las operaciones, o servicios, de nuestras clases. Extrajimos del enunciado del problema los verbos y frases verbales que especifican las operaciones para cada clase. Después modificamos el diagrama de clases de la sección 3.7 para incluir cada operación con su clase asociada. En este punto del ejemplo práctico, hemos recopilado toda la información posible del enunciado del problema. Sin embargo, y a medida que los siguientes capítulos introduzcan temas como la herencia, el manejo de eventos y el subprocesamiento múltiple, modificaremos nuestras clases y diagramas.

Sección 7.10 (Ejemplo práctico opcional), Acerca de los objetos: Colaboración entre objetos

En este punto hemos creado un "borrador" del modelo para nuestro sistema de elevador. En esta sección vemos cómo funciona. Investigamos el comportamiento de la simulación al hablar sobre las *colaboraciones* (mensajes que los objetos se envían entre sí para comunicarse). Las operaciones de las clases que hemos descubierto en la sección 6.15 resultan ser las colaboraciones entre los objetos en nuestro sistema. Determinamos las colaboraciones en nuestro sistema y después las reunimos en un *diagrama de colaboración*: el diagrama de UML para modelar colaboraciones. Este diagrama revela qué objetos colaboran y cuándo lo hacen. Presentamos un diagrama de colaboración de las personas que entran y salen del elevador.

Sección 8.17 (Ejemplo práctico opcional), Acerca de los objetos: Cómo empezar a programar las clases para la simulación del elevador

En esta sección tomamos un descanso, dejando a un lado el diseño del comportamiento de nuestro sistema. Comenzamos el proceso de implementación para enfatizar el material descrito en el capítulo 8. Utilizando el diagrama de clases de UML de la sección 3.7 y los atributos y operaciones descritas en las secciones 4.14 y 6.15, mostramos cómo implementar una clase en Java, partiendo de un diseño. No implementamos todas las clases, ya que no hemos completado el proceso de diseño. Basándonos en nuestros diagramas de UML, creamos código para la clase `Elevador`.

Sección 10.12 (Ejemplo práctico opcional), Acerca de los objetos: Cómo incorporar la herencia a la simulación del elevador

Los capítulos 9 y 10 hablan sobre la programación orientada a objetos. Consideramos la herencia: las clases que comparten características similares pueden heredar los atributos y operaciones de una clase "base". En esta sección investigamos cómo nuestra simulación del elevador puede beneficiarse a partir del uso de la herencia. Documentamos nuestros descubrimientos en un diagrama de clases que modela las relaciones de herencia; UML se refiere a estas relaciones como *generalizaciones*. Después modificamos el diagrama de clases de la sección 3.7 mediante el uso de la herencia, para agrupar las clases con características similares.

Sección 11.9 (Ejemplo práctico opcional), Acerca de los objetos: Manejo de eventos

En esta sección incluimos las interfaces necesarias para que los objetos en nuestra simulación del elevador envíen mensajes a otros objetos. En Java, los objetos se comunican generalmente enviando un *evento*: una notificación de que ha ocurrido una acción. El objeto que recibe el evento entonces realiza una acción en respuesta al tipo de evento recibido; esto se conoce como *manejo de eventos*. En la sección 7.10 describimos el paso de mensajes (o las colaboraciones) en nuestro modelo, utilizando un diagrama de colaboraciones. Ahora modificamos este diagrama para incluir el manejo de eventos y, como ejemplo, explicamos detalladamente cómo las puertas en nuestra simulación se abren al momento de la llegada del elevador.

Sección 12.9 (Ejemplo práctico opcional), Acerca de los objetos: Diseño de interfaces con UML

En esta sección diseñamos un diagrama de clases que modela las relaciones entre clases e interfaces en nuestra simulación; UML se refiere a estas relaciones como *realizaciones*. Además, enlistamos todas las operaciones que proporciona cada interfaz a las clases. Por último, mostramos cómo crear clases en Java que implementen estas interfaces.

Sección 13.17 (Ejemplo práctico opcional), Acerca de los objetos: Casos de uso

El capítulo 13 habla sobre las interfaces de usuario que le permiten interactuar con un programa. En esta sección hablamos sobre la interacción entre nuestra simulación del elevador y su usuario. Específicamente, investigamos los casos que pueden ocurrir entre el usuario de la aplicación y la simulación en sí; este conjunto de casos se conoce como *caso de uso*. Modelamos estas interacciones utilizando *diagramas caso de uso* de UML.

Sección 14.13 (Ejemplo práctico opcional), Acerca de los objetos: Modelo-vista-controlador

Diseñamos nuestro sistema para que consista de tres componentes, cada uno de los cuales tiene una responsabilidad distinta. En este punto del ejemplo práctico casi hemos completado el primer componente, conocido como el *modelo*, que contiene datos que representan a la simulación. Diseñaremos la *vista* (el segundo componente, que trata acerca de la manera en que se muestra el modelo) en la sección 19.7. Diseñaremos el *contro-*

lador (el componente que permite al usuario controlar el modelo) en esta sección. Se dice que un sistema como el nuestro, que utiliza los componentes modelo, vista y controlador, se adhiere a la arquitectura *Modelo-vista-controlador* (*MVC*). En esta sección explicamos las ventajas de utilizar esta arquitectura para diseñar software. Utilizamos el *diagrama de componentes* de UML para modelar los tres componentes y luego implementamos este diagrama como código en Java.

Sección 16.11 (Ejemplo práctico opcional), Acerca de los objetos: Subprocesamiento múltiple
En esta sección declaramos ciertos objetos como "subprocesos" para que éstos puedan operar en forma concurrente. Modificamos el diagrama de colaboración que se presentó originalmente en la sección 7.10 (y que se modificó en la sección 11.9) para incorporar el subprocesamiento múltiple. Presentamos el *diagrama de secuencia* de UML para modelar las interacciones en un sistema. Este diagrama enfatiza el orden cronológico de los mensajes. Utilizamos un diagrama de secuencia para modelar la forma en que una persona dentro de la simulación interactúa con el elevador. Esta sección concluye el diseño de la porción de nuestra simulación relacionada con el modelo. Diseñamos la manera en que se muestra este modelo en la sección 19.7 y luego implementamos este modelo como código en Java, en el apéndice E.

Sección 19.7 (Ejemplo práctico opcional), Acerca de los objetos: Animación y sonido en la vista
Esta sección diseña la vista, que especifica cómo se muestra la porción de la simulación relacionada con el modelo. El capítulo 19 presenta varias técnicas para integrar sonido y animación en los programas. Esta sección utiliza algunas de estas técnicas para incorporar sonido y animación en nuestra simulación del elevador. Específicamente, habla sobre cómo animar los movimientos de las personas y de nuestro elevador, cómo generar efectos y tocar "música de elevador" cuando una persona viaja en él. Esta sección concluye el diseño de nuestra simulación de elevador. Los apéndices D, E y F implementan este diseño como un programa en Java de 3,320 líneas, completamente funcional.

Apéndice D: Eventos e interfaces de escucha del elevador
[Nota: Este apéndice se encuentra en el CD que viene con este libro.] Como vimos en la sección 11.9, varios objetos en nuestra simulación interactúan entre sí mediante el envío de mensajes, conocidos como eventos, a otros objetos que desean recibir estos eventos. Los objetos que reciben los eventos se conocen como *objetos de escucha*; éstos deben implementar *interfaces de escucha*. En este apéndice implementamos todas las clases de eventos e interfaces de escucha utilizadas por los objetos en nuestra simulación.

Apéndice E: Modelo del elevador
[Nota: Este apéndice se encuentra en el CD que viene con este libro.] La mayor parte del ejemplo práctico involucra el diseño del modelo (es decir, los datos y la lógica) de la simulación del elevador. En este apéndice implementamos ese modelo en Java. Utilizando todos los diagramas de UML que creamos, presentamos las clases de Java necesarias para implementar el modelo. Aplicamos los conceptos del diseño orientado a objetos con UML y la programación orientada a objetos, y con el nivel de Java que usted aprendió en los capítulos.

Apéndice F: Vista del elevador
[Nota: Este apéndice se encuentra en el CD que viene con este libro.] Este apéndice final implementa la apariencia visual de nuestra simulación del elevador. Utilizamos el mismo enfoque para implementar la vista que para implementar el modelo; creamos todas las clases requeridas para ejecutar la vista, utilizando los diagramas de UML y conceptos clave descritos en los capítulos. Al final de este apéndice, usted habrá completado el diseño y la implementación "a nivel industrial" de un sistema a gran escala. Debe sentirse con la confianza de hacer frente a sistemas más extensos, como el ejemplo práctico empresarial en Java de 10,000 líneas que presentamos en nuestro libro complementario *Advanced Java 2 Platform How to Program* y los tipos de aplicaciones que crean los ingenieros de software profesionales. Esperamos que continúe estudiando aún con más profundidad el diseño orientado a objetos con UML.

(Opcional) Un paseo por las secciones "Descubrimiento de patrones de diseño"

Nuestro análisis de los patrones de diseño se esparce a lo largo de cinco secciones opcionales del libro. Aquí veremos las generalidades acerca de esas secciones.

Sección 10.12 (Opcional), Descubrimiento de los patrones de diseño: Introducción a los patrones de diseño de creación, estructura y comportamiento
Aquí encontrará las secciones en las que hablamos sobre los diversos patrones de diseño. Dividimos la discusión de cada sección en patrones de diseño de creación, estructura y comportamiento. Los patrones de creación proporcionan maneras de crear instancias de objetos, los patrones de estructura tratan con la organización de los objetos y los patrones de comportamiento tratan con las interacciones entre los objetos. El resto de esta sección introduce algunos de estos patrones de diseño como Singular (Singleton), Proxy, Memento y Estado (State). Por último incluimos varios URLs para profundizar más en el estudio de los patrones de diseño.

Sección 14.14 (Opcional), Descubrimiento de los patrones de diseño: Patrones de diseño utilizados en los paquetes `java.awt` y `javax.swing`
Esta sección contiene una buena parte de nuestra discusión sobre los patrones de diseño. Usando el material de los componentes GUI Swing de Java en los capítulos 13 y 14, investigamos algunos ejemplos del uso de patrones en los paquetes `java.awt` y `javax.swing`. Hablamos sobre cómo estas clases utilizan los patrones de diseño Método de fábrica (Factory Method), Adaptador (Adapter), Puente (Bridge), Compuesto (Composite), Cadena de responsabilidad (Chain-of-Responsibility), Comando (Command), Observador (Observer), Estrategia (Strategy) y Método de plantilla (Template Method). Motivamos cada patrón y presentamos ejemplos de cómo aplicarlos.

Sección 16.12 (Opcional), Descubrimiento de los patrones de diseño: Patrones de diseño concurrentes
Los desarrolladores han descubierto varios patrones de diseño desde aquellos que se conocían como la banda de los cuatro. En esta sección hablamos sobre los patrones de diseño de concurrencia, incluyendo: Ejecución sde un solo subproceso (Single-Threaded Execution), Suspensión precavida (Guarded Suspension), Frustración (Balking), Bloqueo de lectura/escritura (Read/Write Lock) y Término en dos fases (Two-Phase Termination); éstos resuelven varios problemas de diseño en los sistemas con subprocesamiento múltiple. Investigamos cómo la clase `java.lang.Thread` utiliza los patrones de concurrencia.

Sección 18.12 (Opcional), Descubrimiento de los patrones de diseño: Patrones de diseño utilizados en los paquetes `java.io` y `java.net`
Usando el material sobre archivos, flujos y redes de los capítulos 17 y 18, investigamos algunos ejemplos del uso de patrones en los paquetes `java.io` y `java.net`. Hablamos sobre cómo estas clases utilizan los patrones de diseño Fábrica abstracta (Abstract Factory), Decorador (Decorator) y Fachada (Facade). También consideramos los patrones de arquitectura, que especifican un conjunto de subsistemas (agregados de objetos que en conjunto abarcan una responsabilidad importante del sistema) y cómo estos subsistemas interactúan unos con otros. Hablamos sobre los populares patrones arquitectónicos Modelo-vista-controlador y Niveles (Layers).

Sección 22.12 (Opcional), Descubrimiento de los patrones de diseño: Patrones de diseño utilizados en el paquete `java.util`
Utilizando el material sobre estructuras de datos y colecciones en los capítulos 20 al 22, investigamos el uso de patrones en el paquete `java.util`. Hablamos sobre cómo estas clases utilizan los patrones de diseño Prototipo (Prototype) e Iterador (Iterator). Esta sección concluye la discusión sobre los patrones de diseño. Después de terminar el material de *Descubrimiento de los patrones de diseño*, usted deberá ser capaz de reconocer y utilizar los patrones de diseño claves y de tener una mejor comprensión sobre el funcionamiento de la API de Java. Después de completar este material, le recomendamos que siga con el libro de la Banda de los cuatro (Gang-of-Four).

Reconocimientos

Uno de los mayores placeres al escribir un libro de texto es el de reconocer el esfuerzo de mucha gente, cuyos nombres quizá no aparezcan en la portada, pero cuyo arduo trabajo, cooperación, amistad y comprensión fue crucial para la elaboración de este libro.

Mucha gente en Deitel & Associates, Inc. dedicó largas horas a este proyecto. Nos gustaría reconocer los esfuerzos de nuestros colegas de tiempo completo en Deitel & Associates, Inc.: Tem Nieto, Sean Santry, Su Zhang y Jeff Listfield.

- Tem Nieto es egresado del Massachusetts Institute of Technology. Imparte seminarios de XML, Java, Internet y Web, C, C++ y Visual Basic, y trabaja con nosotros en la escritura de libros de texto, el desarrollo de cursos y creación de multimedia. Es coautor de varios libros con nosotros, incluyendo *Internet & World Wide Web How to Program (segunda edición), XML How to Program, Visual Basic .NET How to Program* y *C# How to Program*. En este libro, Tem participó como coautor en los capítulos 12 al 14 y 22, y en la sección especial titulada "Construya su propio compilador" en el capítulo 20.
- Sean Santry, egresado del Boston College (Ciencias computacionales y Filosofía) y coautor de *Advanced Java 2 Platform How to Program* y *Java Web Services for Experienced Programmers*, editó todo el manuscrito, diseñó e implementó la aplicación de red Deitel Messenger en el capítulo 18 (Redes), contribuyó en el diseño y actualizó el ejemplo práctico opcional sobre DOO/UML, y actualizó la introducción opcional a los patrones de diseño.
- Su Zhang posee títulos B.Sc y M.Sc en Ciencias computacionales de la Universidad McGill. Su tesis incluyó el modelado y la simulación, los sistemas de tiempo real y la tecnología Java. Ella es coautora con nosotros en *Java Web Services for Experienced Programmers* y ha contribuido en otras publicaciones Deitel, incluyendo *Advanced Java 2 Platform How to Program* y *Python How to Program*. Ayudó a actualizar varios capítulos y creó los ejemplos que presentan nuevas características de Java 1.4, como las Nuevas APIs de E/S (de las cuales se habla en los capítulos 17 y 18) y las excepciones encadenadas (que se mencionan en el capítulo 15).
- Jeff Listfield es egresado en Ciencias computacionales del Harvard College. Su trabajo escolar incluyó clases en gráficos computacionales, redes y teoría computacional, y tiene experiencia de programación en C, C++, C#, Java, Perl y Lisp. Jeff es nuestro coautor en *C# How to Program, C# A Programmer's Introduction* y *C# for Experienced Programmers*, además contribuyó en *Perl How to Program*. Jeff ayudó a actualizar varios capítulos en el libro, escribió nuevos ejemplos y secciones sobre expresiones regulares en el capítulo 11, ordenamiento en el capítulo 22 y partes de las secciones sobre la Nueva API de E/S en los capítulos 17 y 18).

Somos afortunados al haber trabajado en este proyecto con un talentoso y dedicado equipo de editores profesionales. Apreciamos de manera especial el extraordinario esfuerzo de nuestra editora de Ciencias computacionales, Petra Recter y su jefa, nuestra mentora en la edición, Marcia Horton, directora editorial de la división de Ciencias de la computación e ingeniería de Prentice Hall. Tom Manshreck hizo un estupendo trabajo como gerente de producción. Jennifer Cappello hizo un excelente trabajo en el manejo del proceso de revisión.

El *Java 2 Multimedia Cyber Classroom, 5ª edición* se desarrolló en paralelo con *Cómo programar en Java, 5ª edición*. Sinceramente apreciamos la perspicacia, experiencia y habilidad técnica de nuestro editor en jefe de medios electrónicos, mentor y amigo, Mark Taub. Él y nuestra editora de medios electrónicos, Karen Mclean, hicieron un gran trabajo al lograr la publicación de *Java 2 Multimedia Cyber Classroom, 5ª edición* bajo un estrecho itinerario.

Debemos un agradecimiento especial a Tamara Newnam (`smart_art@earthlink.net`), quien creó el trabajo artístico para nuestros iconos de tips de programación y la portada. Ella creó la encantadora criatura que comparte con usted los tips de programación del libro.

Sinceramente apreciamos los esfuerzos de nuestros 70 revisores de la cuarta edición después de su publicación, y de nuestros revisores de la quinta edición:

Revisores de Sun Microsystems
Dibyendu Baksi (Sun Microsystems)
Konstantin Kladko (Sun Microsystems)
Doug Kohlert (Sun Microsystems)
Peter Jones (Sun Microsystems)
Paul Monday (Sun Microsystems)
Tomas Pavek (Sun Microsystems)
Brandon Taylor (Sun Microsystems)

Revisores académicos
Rekha Bhowmik (St. Cloud State University)

Clint Bickmore (Front Range Community College)
Brian Blake (Georgetown University)
Ayad Boudiab (Georgia Perimiter College)
Chadi Boudiab (Georgia Perimiter College)
Michael Buckley (State University of New York-Buffalo)
James Chegwidden (Tarrant County College)
Deborah Coleman (Rochester Institute of Technology)
Don Francis Costello (University of Nebraska)
Balazs Csizmazia (University of Klagenfurt)
Tamara Dinev (Florida Atlantic University)
Sarah Fix (The Career Center High School)
Bill Freitas (The Lawrenceville School)
Thomas Graffte (Strayer University)
Balaji Janamanchi (Texas Tech University)
Charles Lake (Faulkner State Community College)
Brian Larson (Modesto Junior College)
Hong Lin (DeVry University)
David McKain (Lakota East High School)
Andy Novobilski (University of Tennessee-Chattanooga)
Richard Ord (University of California, San Diego)
Gavin Osborne (Saskatchewan Institute of Applied Sci.& Tech.)
Merrill Parker (Chattanooga State Technical Community College)
Donna Reese (Mississippi State University)
Craig Slinkman (University of Texas, Arlington)
Gidget Smith (Arkansas State University)
Ron Sones (James Madison University)
Mahendran Velauthapillai (Georgetown University)
Loran Walker (Lawrence Technological University)
Warren Wiltsie (Fairleigh Dickinson University)

Otros revisores de la industria
Shishir Abhyanker (Accenture)
Sinan Alhir (Consultor independiente)
Richard Bonneau (IONA Technologies)
Columbus Brown (IBM)
Carl Burnham (Southpoint)
Brian Cook (Zurich Insurance)
Jonathan Earl (Consultor independiente)
Ron Felice (Omniware Development)
Karl Frank (TogetherSoft Corporation)
Kyle Gabhart (Consultor independiente)
Johan Galle (E2S)
Mark Grand (ClickBlocks, LLC)
Ajay Gupta (American Airlines, Inc.)
Kevlin Henney (Curbralan, Ltd.)
Ethan Henry (Sitraka)
Anne Horton (AT&T Laboratories)
James Huddleston (Consultor independiente)
Terrell Hull (Sun Certified Java Architect, Rational Qualified Practitioner)
Sachin Korgaonkar (Idealake Technologies, Pvt. Ltd.)
Don Kostuch (You Can C Clearly Now)
Krishna Kunchithapadam (Oracle Corporation)

Paul McLachlan (Compuware Corporation)
Davyd Norris (Rational)
Bill O'Farrell (IBM)
Praveen Sadhu (Infodat Solutions, Inc.)
Cameron Skinner (Embarcadero Technologies/OMG)
Stephen Tockey (Construx Software/OMG)
Kim Topley (Keyboard Edge, Ltd.)
Sudhir Upadhyay (BEA Systems, Inc.)
John Varghese (UBS Warburg)
Bing Xue (Siemens)
Hadar Ziv (eBuilt, Inc.)

A pesar de una calendarización muy apretada, estos revisores revisaron cada aspecto del libro e hicieron incontables sugerencias para mejorar la precisión e integridad del material presentado.

Hemos hecho un gran esfuerzo para crear este libro y su versión opcional de Cyber Classroom. Esta edición está repleta de ejemplos LIVE-CODE™, tips de programación, ejercicios de autoevaluación con sus respuestas, ejercicios y proyectos retadores, y numerosas herramientas de estudio para ayudarlo a dominar el material. Java es un poderoso lenguaje de programación que le ayudará a escribir programas con rapidez y efectividad. Además, es un lenguaje que escala sin problemas en el ámbito del desarrollo de sistemas empresariales, para ayudar a las organizaciones a crear sus sistemas de información claves. Le agradeceremos que, a medida que lea el libro, nos haga saber sus comentarios, críticas, correcciones y sugerencias para mejorar el texto. Por favor dirija toda su correspondencia a:

`deitel@deitel.com`

Le responderemos oportunamente y publicaremos las correcciones y aclaraciones en nuestro sitio Web,

`www.deitel.com`

¡Esperamos que disfrute aprendiendo con este libro tanto como nosotros disfrutamos el escribirlo!

Dr. Harvey M. Deitel
Paul J. Deitel

Acerca de los autores

Dr. Harvey M. Deitel. Presidente de Deitel & Associates, Inc., tiene más de 40 años de experiencia en el campo de la computación, esto incluye un amplio trabajo académico y en la industria. Es uno de los profesores de ciencias computacionales y expositores más reconocidos a nivel mundial. El Dr. Deitel tiene una licenciatura y una maestría por el Massachusetts Institute of Technology y un doctorado de la Boston University. Tiene 20 años de experiencia como profesor universitario, incluyendo un puesto vitalicio y el haber sido presidente del departamento de Ciencias computacionales en el Boston College antes de fundar, con su hijo Paul J. Deitel, Deitel & Associates, Inc. Es autor y coautor de varias docenas de libros y paquetes multimedia. Los textos del Dr. Deitel se han ganado el reconocimiento internacional y han sido traducidas al japonés, ruso, español, italiano, chino tradicional, coreano, francés, polaco, turco, urdú, griego y portugués. El Dr. Deitel ha impartido seminarios profesionales internacionalmente para grandes empresas, organizaciones gubernamentales y diversos sectores del ejército.

Paul J. Deitel. CEO de Deitel & Associates, Inc., es egresado del Sloan School of Management del Massachusetts Institute of Technology, en donde estudió Tecnología de la información. A través de Deitel & Associates, Inc., ha impartido seminarios de lenguajes computacionales a grandes empresas, organizaciones gubernamentales y diversos sectores del ejército. Ha ofrecido conferencias de Java y C++ para la Boston Chapter de la Association for Computing Machinery, y ha impartido cursos vía satélite a través de una empresa de cooperación entre Deitel & Associates, Inc., Prentice Hall y Technology Education Network. Él y su padre, el Dr. Harvey M. Deitel, son autores de los libros de Ciencias de la computación con más ventas en el mundo.

Acerca de Deitel & Associates, Inc.

Deitel & Associates, Inc. es una empresa reconocida a nivel mundial, dedicada al entrenamiento corporativo y la creación de contenido, con especialización en educación para tecnología de software para Internet/World Wide Web, tecnología de software para comercio electrónico (e-business/e-commerce) y lenguajes de programación. La empresa proporciona cursos sobre programación en Internet y World Wide Web, tecnología de objetos y los principales lenguajes de programación. Los fundadores de Deitel & Associates, Inc. son el Dr. Harvey M. Deitel y Paul J. Deitel. Sus clientes incluyen muchas de las empresas de cómputo más grandes del mundo, agencias gubernamentales, sectores del ejército y empresas comerciales. A lo largo de su sociedad editorial con Prentice Hall, Deitel & Associates Inc. ha publicado libros de texto de vanguardia sobre programación, libros profesionales, multimedia interactiva en CD como los *Cyber Classrooms*, cursos vía satélite y cursos de capacitación basados en Web. Deitel & Associates, Inc. y los autores pueden ser contactados mediante correo electrónico en:

`deitel@deitel.com`

Para conocer más acerca de Deitel & Associates, Inc., sus publicaciones y su currículo corporativo mundial en línea, vea las últimas páginas de este libro o visite:

`www.deitel.com`

y suscríbase al boletín gratuito de correo electrónico, *DEITEL BUZZ ONLINE*, en:

`www.deitel.com/newsletter/subscribe.html`