Piensa en Java

Piensa en Java

SEGUNDA EDICIÓN

Bruce Eckel

Traducción:
Jorge González Barturen
Facultad de Ingeniería
Universidad de Deusto

Revisión técnica:
Javier Parra Fuente
Ricardo Lozano Quesada
Departamento de Lenguajes y Sistemas Informáticos e Ingeniería de Software
Universidad Pontificia de Salamanca en Madrid

Coordinación general y revisión técnica:
Luis Joyanes Aguilar
Departamento de Lenguajes y Sistemas Informáticos e Ingeniería de Software
Universidad Pontificia de Salamanca en Madrid

Madrid • México • Santafé de Bogotá • Buenos Aires • Caracas • Lima • Montevideo
San Juan • San José • Santiago • Sao Paulo • White Plains

Datos de catalogación bibliográfica

Bruce Eckel
PIENSA EN JAVA
Segunda edición

PEARSON EDUCACIÓN, S.A. Madrid, 2002

ISBN: 84-205-3192-8
Materia: Informática 681.3

Formato 195 x 250 Páginas: 960

Bruce Eckel
PIENSA EN JAVA, segunda edición.

ISBN: 84-205-3192-8
Depósito Legal: M.4.162-2003
Última reimpresión, 2003

PRENTICE HALL es un sello editorial autorizado de PEARSON EDUCACIÓN, S.A.

Traducido de:
Thinking in JAVA, Second Edition by Bruce Eckel.

ISBN: 0-13-027363-5

Edición en español:
Equipo editorial:
Editor: Andrés Otero
Asistente editorial: Ana Isabel García
Equipo de producción:
Director: José A. Clares
Técnico: Diego Marín
Diseño de cubierta: Mario Guindel, Lía Sáenz y Begoña Pérez
Composición: COMPOMAR. S.L.
Impreso por: LAVEL, S. A.

IMPRESO EN ESPAÑA - PRINTED IN SPAIN

Este libro ha sido impreso con papel y tintas ecológicos

A la persona que, incluso en este momento, está creando el próximo gran lenguaje de programación.

Índice de contenido

Prólogo

Sugerí a mi hermano Todd, que está dando el salto del hardware a la programación, que la siguiente gran revolución será en ingeniería genética.

Tendremos microbios diseñados para hacer comida, combustible y plástico; limpiarán la polución y en general, nos permitirán dominar la manipulación del mundo físico por una fracción de lo que cuesta ahora. De hecho yo afirmé que la revolución de los computadores parecería pequeña en comparación.

Después, me di cuenta de que estaba cometiendo un error frecuente en los escritores de ciencia ficción: perderme en la tecnología (lo que por supuesto es fácil de hacer en ciencia ficción). Un escritor experimentado sabe que la historia nunca tiene que ver con los elementos, sino con la gente. La genética tendrá un gran impacto en nuestras vidas, pero no estoy seguro de que haga sombra a la revolución de los computadores (que hace posible la revolución genética) —o al menos la revolución de la información. La información hace referencia a comunicarse con otros: sí, los coches, los zapatos y especialmente la terapia genética son importantes, pero al final, ésto no son más que adornos. Lo que verdaderamente importa es cómo nos relacionamos con el mundo. Y cuánto de eso es comunicación.

Este libro es un caso. La mayoría de colegas pensaban que estaba un poco loco al poner todo en la Web. "¿Por qué lo compraría alguien?", se preguntaban. Si hubiera sido de naturaleza más conservadora no lo habría hecho, pero lo que verdaderamente no quería era escribir más libros de computación al estilo tradicional. No sabía qué pasaría pero resultó que fue una de las cosas más inteligentes que he hecho con un libro.

Por algún motivo, la gente empezó a mandar correcciones. Éste ha sido un proceso divertido, porque todo el mundo ha recorrido el libro y ha detectado tanto los errores técnicos como los gramaticales, y he podido eliminar fallos de todos los tipos que de otra forma se habrían quedado ahí. La gente ha sido bastante amable con ésto, diciendo a menudo " yo no quiero decir esto por criticar...", y tras darme una colección de errores estoy seguro de que de otra forma nunca los hubiera encontrado. Siento que éste ha sido un tipo de grupo de procesos que ha convertido el libro en algo especial.

Pero cuando empecé a oír: "De acuerdo, bien, está bien que hayas puesto una versión electrónica, pero quiero una copia impresa proveniente de una auténtica editorial", puse mi mayor empeño en facilitar que todo se imprimiera con formato adecuado, pero eso no frenó la demanda de una versión publicada. La mayoría de la gente no quiere leer todo el libro en pantalla, y merodear por un conjunto de papeles, sin que importe cuán bien impresos estén, simplemente no era suficiente. (Además, tampoco creo que resulte tan barato en términos de tóner para impresora láser.) Parece que a fin de cuentas, la revolución de los computadores no conseguirá dejar sin trabajo a las editoriales. Sin embargo, un alumno me sugirió que éste podría ser un modelo para publicaciones finales: los libros se publicarán primero en la Web, y sólo si hay el suficiente interés, merecerá la pena pasar el libro a papel. Actualmente, la gran mayoría de libros conllevan problemas financieros, y quizás este nuevo enfoque pueda hacer que el negocio de la publicación sea más beneficioso. Este li-

bro se convirtió en una experiencia reveladora para mí de otra forma. Originalmente me acerqué a Java como "simplemente a otro lenguaje de programación", lo que en cierto sentido es verdad. Pero a medida que pasaba el tiempo y lo estudiaba más en profundidad, empecé a ver que la intención fundamental de este lenguaje es distinta de la de otros lenguajes que he visto.

La programación está relacionada con gestionar la complejidad: la complejidad del problema que se quiere solucionar, que yace sobre la complejidad de la máquina en que se soluciona. Debido a esta complejidad, la mayoría de nuestros proyectos fallan. Y lo que es más, de todos los lenguajes de programación de los que soy consciente, ninguno se ha lanzado completamente decidiendo que la meta de diseño principal fuera conquistar la complejidad del desarrollo y mantenimiento de programas[1]. Por supuesto, muchas decisiones de diseño de lenguajes se hicieron sin tener en mente la complejidad, pero en algún punto había siempre algún otro enfoque que se consideraba esencial añadirlo al conjunto. Inevitablemente, estos otros aspectos son los que hacen que generalmente los programadores "se den con la pared" contra ese lenguaje. Por ejemplo, C++ tenía que ser compatible con C (para permitir la migración fácil a los programadores de C), además de eficiente. Estas metas son ambas muy útiles y aportan mucho al éxito de C++, pero también exponen complejidad extra que evita que los proyectos se acaben (ciertamente, se puede echar la culpa a los programadores y la gestión, pero si un lenguaje puede ayudar a capturar los errores, ¿por qué no hacer uso de ello?). Como otro ejemplo, Visual Basic (VB) estaba atado a BASIC, que no estaba diseñado verdaderamente para ser un lenguaje ampliable, por lo que todas las aplicaciones que se apilaban sobre VB producían sintaxis verdaderamente horribles e inmantenibles. Perl es retrocompatible con Awk, Sed, Grep y otras herramientas Unix a las que iba a reemplazar, y como resultado se le acusa a menudo, de producir "código de sólo escritura" (es decir, código que tras unos pocos meses no hay quien lea). Por otro lado, C++, VB, Perl, y otros lenguajes como Smalltalk han visto cómo algunos de sus esfuerzos de diseño se centraban en el aspecto de la complejidad y como resultado son remarcadamente exitosos para solucionar ciertos tipos de problemas.

Lo que más me impresionó es que he llegado a entender que Java parece tener el objetivo de reducir la complejidad *para el programador*. Como si se dijera "no nos importa nada más que reducir el tiempo y la dificultad para producir un código robusto". En los primeros tiempos, esta meta llevaba a un código que no se ejecutaba muy rápido (aunque se habían hecho promesas sobre lo rápido que se ejecutaría Java algún día), pero sin duda ha producido reducciones sorprendentes de tiempo de desarrollo; la mitad o menos del tiempo que lleva crear un programa C++ equivalente. Este resultado sólo puede ahorrar cantidades increíbles de tiempo y dinero, pero Java no se detiene ahí. Envuelve todas las tareas complejas que se han convertido en importantes, como el multihilo y la programación en red, en bibliotecas o aspectos del lenguaje que en ocasiones pueden convertir esas tareas en triviales. Y finalmente, asume muchos problemas de complejidad grandes: programas multiplataforma, cambios dinámicos de código, e incluso seguridad, cada uno de los cuales pueden encajar dentro de un espectro de complejidades que oscila en el rango de "impedimento" a "motivos de cancelación". Por tanto, a pesar de los problemas de rendimiento que se han visto, la promesa de Java es tremenda: puede convertirnos en programadores significativamente más productivos.

Uno de los sitios en los que veo el mayor impacto de esto es en la Web. La programación en red siempre ha sido complicada, y Java la convierte en fácil (y los diseñadores el lenguaje Java están

[1] Esto lo retomo de la 2.ª edición: creo que el lenguaje Python se acerca aún más a esto. Ver http://www.Python.org.

trabajando en facilitarla aún más). La programación en red es como hablar simultáneamente de forma efectiva y de forma más barata de lo que nunca se logró con teléfonos (sólo el correo electrónico ya ha revolucionado muchos negocios). Al intercomunicarnos más, empiezan a pasar cosas divertidas, probablemente mucho más interesantes que las que pasarán con la ingeniería genética.

De todas formas —al crear los programas, trabajar para crear programas, construir interfaces para los programas, de forma que éstos se puedan comunicar con el usuario, ejecutar los programas en distintos tipos de máquinas, y escribir de forma sencilla programas que pueden comunicarse a través de Internet— Java incrementa el ancho de banda de comunicación *entre la gente*. Creo que quizás los resultados de la revolución de la comunicación no se contemplarán por lo que conlleva el transporte de grandes cantidades de bits; veremos la auténtica revolución porque podremos comunicarnos con mayor facilidad: de uno en uno, pero también en grupos y, como planeta. He oído la sugerencia de que la próxima revolución es la formación de cierto tipo de mente global para suficiente gente y suficiente nivel de interconectividad. Puede decirse que Java puede fomentar o no esa revolución, pero al menos la mera posibilidad me ha hecho sentir como si estuviera haciendo algo lleno de sentido al intentar enseñar ese lenguaje.

Prólogo a la 2.ª edición

La gente ha hecho muchos, muchos comentarios maravillosos sobre la primera edición de este libro, cosa que ha sido para mí muy, pero que muy, placentero. Sin embargo, en todo momento habrá quien tenga quejas, y por alguna razón una queja que suele aparecer periódicamente es que "el libro es demasiado grande". Para mí, esto no es verdaderamente una queja, si se reduce a que "tiene demasiadas páginas". (Uno se acuerda de las quejas del Emperador de Austria sobre el trabajo de Mozart: "¡Demasiadas páginas!", y no es que me esté intentando comparar con Mozart de ninguna forma). Además, sólo puedo asumir que semejante queja puede provenir de gente que no tiene aún una idea clara de la vasta extensión del propio lenguaje Java en sí, y que no ha visto el resto de libros sobre la materia —por ejemplo, mi referencia favorita es el *Core Java* de Cay Horstmann & Cary Cornell (Prentice-Hall), que creció tanto que hubo que dividirlo en dos tomos. A pesar de esto, una de las cosas que he intentado hacer en esta edición es eliminar las portes que se han vuelto obsoletas o al menos no esenciales. Me siento a gusto haciendo esto porque el material original sigue en la Web y en el CD ROM que acompaña al libro, en la misma forma de descarga gratuita que la primera edición del libro (en http://www.BruceEckel.com). Si se desea el material antiguo, sigue ahí, y esto es algo maravilloso para un autor. Por ejemplo, puede verse que el último capítulo original, "Proyectos", ya no está aquí; dos de los proyectos se han integrado en los otros capítulos, y el resto ya no son adecuados. También el capítulo de "Patrones de diseño" se volvió demasiado extenso y ha sido trasladado a un libro que versa sobre ellos (descargable también en el sitio web). Por tanto, el libro debería ser más fino.

Pero no lo es.

El aspecto mayor es el continuo desarrollo del lenguaje Java en sí, y en particular las API que se expanden, y prometen proporcionar interfaces estándar para casi todo lo que se desee hacer (y no me sorprendería ver aparecer la API "JTostadora"). Cubrir todas estas API se escapa por supuesto del ámbito de este libro, y es una tarea relegada a otros autores, pero algunos aspectos no pueden ig-

norarse. El mayor de éstos incluye el Java de lado servidor (principalmente Servlets & Java Server Pages o *JSP*), que es verdaderamente una solución excelente al problema de la World Wide Web, donde se descubrió que las distintas plataformas de navegadores web no son lo suficientemente consistentes como para soportar programación en el lado cliente. Además, está todo el problema de crear de forma sencilla aplicaciones que interactúen de forma sencilla con bases de datos, transacciones, seguridad y semejante, cubiertos gracias a los Enterprise Java Beans (EJB). Estos temas están desarrollados en el capítulo que antes se llamaba "Programación de red" y ahora "Computación distribuida", un tema que se está convirtiendo en esencial para todo el mundo. También se verá que se ha compilado este capítulo para incluir un repaso de Jini (pronunciado "yeni", y que no es un acrónimo, sino sólo un nombre), que es una tecnología emergente que permite que cambiemos la forma de pensar sobre las aplicaciones interconectadas. Y por supuesto, el libro se ha cambiado para usar la biblioteca IGU Swing a lo largo de todo el mismo. De nuevo, si se desea el material Java 1.0/1.1 antiguo, es posible conseguirlo gratuitamente del libro de descarga gratuita de *http://www.BruceEckel.com* (también está incluido en el nuevo CD ROM de esta edición, que se adjunta al mismo; hablaré más de él un poco más adelante).

Aparte de nuevas características del lenguaje añadidas a Java 2, y varias correcciones hechas a lo largo de todo el libro, el otro cambio principal está en el capítulo de colecciones que ahora se centra en las colecciones de Java 2, que se usan a lo largo de todo el libro. También he mejorado ese capítulo para que entre más en profundidad en algunos aspectos importantes de las colecciones, en particular, en cómo funcionan las funciones de ***hashing*** (de forma que se puede saber cómo crear una adecuadamente). Ha habido otros movimientos y cambios, incluida la reescritura del Capítulo 1, y la eliminación de algunos apéndices y de otros materiales que ya no consideraba necesarios para el libro impreso, que son un montón de ellos. En general, he intentado recorrer todo, eliminar de la 2.ª edición lo que ya no es necesario (pero que sigue existiendo en la primera edición electrónica), incluir cambios y mejorar todo lo que he podido. A medida que el lenguaje continúa cambiando —aunque no a un ritmo tan frenético como antiguamente— no cabe duda de que habrá más ediciones de este libro.

Para aquellos de vosotros que siguen sin poder soportar el tamaño del libro, pido perdón. Lo creáis o no, he trabajado duro para que se mantenga lo menos posible. A pesar de todo, creo que hay bastantes alternativas que pueden satisfacer a todo el mundo. Además, el libro está disponible electrónicamente (en idioma inglés desde el sitio web, y desde el CD ROM que acompaña al libro), por lo que si se dispone de un ordenador de bolsillo, se puede disponer del libro sin tener que cargar un gran peso. Si sigue interesado en tamaños menores, ya existen de hecho versiones del libro para Palm Pilot. (Alguien me dijo en una ocasión que leería el libro en la cama en su Palm, con la luz encendida a la espalda para no molestar a su mujer. Sólo espero que le ayude a entrar en el mundo de los sueños.) Si se necesita en papel, sé de gente que lo va imprimiendo capítulo a capítulo y se lo lee en el tren.

Java 2

En el momento de escribir el libro, es inminente el lanzamiento del *Java Development Kit* (JDK) 1.3 de Sun, y ya se ha publicado los cambios propuestos para JDK 1.4. Aunque estos números de versión se corresponden aún con los "unos", la forma estándar de referenciar a las versiones posterio-

res a la JDK 1.2 es llamarles "Java 2". Esto indica que hubo cambios muy significativos entre el "viejo Java" —que tenía muchas pegas de las que ya me quejé en la primera edición de este libro— y esta nueva versión más moderna y mejorada del lenguaje, que tiene menos pegas y más adiciones y buenos diseños.

Este libro está escrito para Java 2. Tengo la gran ventaja de librarme de todo el material y escribir sólo para el nuevo lenguaje ya mejorado porque la información vieja sigue existiendo en la 1.ª versión electrónica disponible en la Web y en el CD-ROM (que es a donde se puede ir si se desea obcecarse en el uso de versiones pre-Java 2 del lenguaje). También, y dado que cualquiera puede descargarse gratuitamente el JDK de http://java.sun.com, se supone que por escribir para Java 2, no estoy imponiendo ningún criterio financiero o forzando a nadie a hacer una actualización del software.

Hay, sin embargo, algo que reseñar. JDK 1.3 tiene algunas mejoras que verdaderamente me gustaría usar, pero la versión de Java que está siendo actualmente distribuida para Linux es la JDK 1.2.2 (ver http://www.Linux.org). Linux es un desarrollo importante en conjunción con Java, porque es rápido, robusto, seguro, está bien mantenido y es gratuito; una auténtica revolución en la historia de la computación (no creo que se hayan visto todas estas características unidas en una única herramienta anteriormente). Y Java ha encontrado un nicho muy importante en la programación en el lado servidor en forma de *Servlets*, una tecnología que es una grandísima mejora sobre la programación tradicional basada en CGI (todo ello cubierto en el capítulo "Computación Distribuida").

Por tanto, aunque me gustaría usar sólo las nuevas características, es crítico que todo se compile bajo Linux, y por tanto, cuando se desempaquete el código fuente y se compile bajo ese SO (con el último JDK) se verá que todo compila. Sin embargo, se verá que he puesto notas sobre características de JDK 1.3 en muchos lugares.

El CD ROM

Otro bonus con esta edición es el CD ROM empaquetado al final del libro. En el pasado me he resistido a poner CD ROM al final de mis libros porque pensaba que no estaba justificada una carga de unos pocos Kbytes de código fuente en un soporte tan grande, prefiriendo en su lugar permitir a la gente descargar los elementos desde el sitio web. Sin embargo, pronto se verá que este CD ROM es diferente.

El CD contiene el código fuente del libro, pero también contiene el libro en su integridad, en varios formatos electrónicos. Para mí, el preferido es el formato HTML porque es rápido y está completamente indexado —simplemente se hace clic en una entrada del índice o tabla de contenidos y se estará inmediatamente en esa parte del libro.

La carga de más de 300 Megabytes del CD, sin embargo, es un curso multimedia denominado *Thinking in C: Foundations for C++ & Java*. Originalmente encargué este seminario en CD ROM a Chuck Allison, como un producto independiente, pero decidí incluirlo con la segunda edición tanto de *Thinking in C++* como de *Piensa en Java*, gracias a la consistente experiencia de haber tenido gente viniendo a los seminarios sin la requerida experiencia en C. El pensamiento parece aparentemente ser: "Soy un programador inteligente y no *deseo* aprender C, y sí C++ o Java, por lo que me saltaré C e iré directamente a C++/Java." Tras llegar al seminario, todo el mundo va comprendiendo que el

prerrequisito de aprender C está ahí por algo. Incluyendo el CD ROM con el libro, se puede asegurar que todo el mundo atienda al seminario con la preparación adecuada.

El CD también permite que el libro se presente para una audiencia mayor. Incluso aunque el Capítulo 3 («Controlando el flujo del programa») cubre los aspectos fundamentales de las partes de Java que provienen de C, el CD es una introducción más gentil, y asume incluso un trasfondo de C menor que el que supone el libro. Espero que al introducir el CD será más la gente que se acerque a la programación en Java.

Prólogo a la edición en español

Java se convierte día a día en un *lenguaje de programación universal*; es decir, ya no sólo sirve como lenguaje para programar en entornos de Internet, sino que se está utilizando cada vez más como herramienta de programación orientada a objetos y también como herramienta para cursos específicos de programación o de estructuras de datos, aprovechando sus características de lenguaje *"multiparadigma"*. Por estas razones, los libros que afronten temarios completos y amplios sobre los temas anteriores siempre serán bienvenidos. Si, además de reunir estos requisitos, el autor es uno de los más galardonados por sus obras anteriores, nos enfrentamos ante un reto considerable: "la posibilidad de encontrarnos" ante un gran libro, de esos que hacen "historia". Éste, pensamos, es el caso del libro que tenemos entre las manos. ¿Por qué pensamos así?

El libro como referencia obligada a Java

Piensa en Java introduce todos los fundamentos teóricos y prácticos del lenguaje Java, tratando de explicar con claridad y rigor *no sólo lo que hace* el lenguaje *sino también el porqué.* Eckel introduce los fundamentos de objetos y cómo los utiliza Java. Éste es el caso del estudio que hace de la ocultación de las implementaciones, reutilización de clases y polimorfismo. Además, estudia en profundidad propiedades y características tan importantes como AWT, programación concurrente (multihilo, *multithreading2*), programación en red, e incluso diseño de patrones.

Es un libro que puede servir para iniciarse en Java y llegar hasta un nivel avanzado. Pero, en realidad se sacará el máximo partido al libro si se conoce otro lenguaje de programación, o al menos técnicas de programación (como haber seguido un curso de Fundamentos de Programación, Metodología de la Programación, Algoritmos, o cursos similares) y ya se puede apostar por un alto y eficiente rendimiento si la migración a Java se hace desde un lenguaje orientado a objetos, como C++.

El libro como formación integral de programador

Una de las fortalezas más notables del libro es su contenido y la gran cantidad de temas importantes cubiertos con toda claridad y rigor, y con gran profundidad. El contenido es muy amplio y sobre todo completo. Eckel prácticamente ha tocado casi todas las técnicas existentes y utilizadas hoy día en el mundo de la programación y de la ingeniería del software. Algunos de los temas más sobresalientes analizados en el libro son: fundamentos de diseño orientado a objetos, implementación de herencia y polimorfismo, manejo de excepciones, multihilo y persistencia, Java en Internet, recolección de basura, paquetes Java, diseño por reutilización: composición, herencia, interfaces y clases internas, arrays y contenedores de clases, clases de E/S Java, programación de redes con *sockets*, JDBC para bases de datos, JSPs *(JavaServer Pages)*, RMI, CORBA, EJBs *(Enterprise JavaBeans)* y Jini, JNI *(Java Native Interface)*.

El excelente y extenso contenido hacen al libro *idóneo* para la preparación de cursos de nivel medio y avanzado de programación, tanto a nivel universitario como profesional. Asimismo, por el enfoque masivamente profesional que el autor da al libro, puede ser una herramienta muy útil como referencia básica o complementaria para preparar los exámenes de certificación Java que la casa Sun Microsystems otorga tras la superación de las correspondientes pruebas. Esta característica es un valor añadido muy importante, al facilitar considerablemente al lector interesado las directrices técnicas necesarias para la preparación de la citada certificación.

Recursos gratuitos en línea

Si las características citadas anteriormente son de por sí lo suficientemente atractivas para la lectura del libro, es sin duda el excelente sitio en Internet del autor otro valor añadido difícil de medir, por no decir inmedible y valiosísimo. La generosidad del autor —y, naturalmente, de Pearson—, que ofrece a cualquier lector, sin necesidad de compra previa, todo el contenido en línea, junto a las frecuentes revisiones de la obra y soluciones a ejercicios seleccionados, con la posibilidad de descargarse gratuitamente todo este inmenso conocimiento incluido en el libro, junto al conocimiento complementario ofertado (ejercicios, revisiones, actualizaciones...), hacen a esta experiencia innovadora del autor digna de los mayores agradecimientos por parte del cuerpo de programadores noveles o profesionales de cualquier lugar del mundo donde se utilice Java (que hoy es prácticamente "todo el mundo mundial", que dirían algunos periodistas).

De igual forma es de agradecer el CD ROM que acompaña al libro y la oferta de un segundo CD gratuito que se puede conseguir siguiendo las instrucciones incluidas en el libro con el texto completo de la versión original en inglés y un gran número de ejercicios seleccionados resueltos y recursos Java de todo tipo.

Para facilitar al lector el uso del CD ROM incluido en el libro, el equipo de revisión técnica ha realizado el Apéndice E: *Correspondencias español-inglés de clases, bases de datos, tablas y campos del CD ROM que acompaña al libro*, a fin de identificar el nombre asignado en la traducción al español, con el nombre original en inglés de dichas clases.

Unas palabras todavía más elogiosas

Para las personas que, como el autor de este prólogo, llevamos muchos años (ya décadas) dedicándonos a programar computadores, enseñar a programar y escribir sobre programación, un libro como éste sólo nos trae elevados y elogiosos pensamientos. Consideramos que es un libro magnífico, maduro, consistente, intelectualmente honesto, bien escrito y preciso. Sin duda, como lo demuestra su larga lista de premios y sus numerosas y magníficas cualidades, *Piensa en Java,* no sólo es una excelente obra para aprender y llegar a dominar el lenguaje Java y su programación, sino también una excelente obra para aprender y dominar las técnicas modernas de programación.

Luis Joyanes Aguilar
Director del Departamento de Lenguajes y Sistemas Informáticos e Ingeniería de Software
Universidad Pontificia de Salamanca campus Madrid

Comentarios de los lectores

Mucho mejor que cualquier otro libro de Java que haya visto. Esto se entiende "en orden de magnitud" ... muy completo, con ejemplos directos y al grano, excelentes e inteligentes, sin embarullarse, lleno de explicaciones....En contraste con muchos otros libros de Java lo he encontrado inusualmente maduro, consistente, intelectualmente honesto, bien escrito y preciso. En mi honesta opinión, un libro ideal para estudiar Java. **Anatoly Vorobey, Technion University, Haifa, Israel.**

Uno de los mejores tutoriales de programación, que he visto de cualquier lenguaje. **Joakim Ziegler, FIX sysop.**

Gracias por ese libro maravilloso, maravilloso en Java. **Dr. Gavin Pillary, Registrar, King Eduard VIII Hospital, Suráfrica.**

Gracias de nuevo por este maravilloso libro. Yo estaba completamente perdido (soy un programador que no viene de C) pero tu libro me ha permitido avanzar con la misma velocidad con la que lo he leído. Es verdaderamente fácil entender los principios subyacentes y los conceptos desde el principio, en vez de tener que intentar construir todo el modelo conceptual mediante prueba y error. Afortunadamente podré acudir a su seminario en un futuro no demasiado lejano. **Randall R. Hawley, Automation Technician, Eli Lilly & Co.**

El mejor libro escrito de computadores que haya visto jamás. **Tom Holland.**

Éste es uno de los mejores libros que he leído sobre un lenguaje de programación... El mejor libro sobre Java escrito jamás. **Revindra Pai, Oracle Corporation, línea de productos SUNOS.**

¡Éste es el mejor libro sobre Java que haya visto nunca! Has hecho un gran trabajo. Tu profundidad es sorprendente. Compraré el libro en cuanto se publique. He estado aprendiendo Java desde octubre del 96. He leído unos pocos libros y considero el tuyo uno que "SE DEBE LEER". En estos últimos meses nos hemos centrado en un producto escrito totalmente en Java. Tu libro ha ayudado a consolidar algunos temas en los que andábamos confusos y ha expandido mi conocimiento base. Incluso he usado algunos de tus ejemplos y explicaciones como información en mis entrevistas para ayudar al equipo. He averiguado el conocimiento de Java que tienen preguntándoles por algunas de las cosas que he aprendido a partir de la lectura de tu libro (por ejemplo, la diferencia entre arrays y Vectores). ¡El libro es genial! **Steve Wilkinson, Senior Staff Specialist, MCI Telecommunications.**

Gran libro. El mejor libro de Java que he visto hasta la fecha. **Jeff Sinlair, ingeniero de Software, Kestral Computing.**

Gracias por *Piensa en Java*. Ya era hora de que alguien fuera más allá de una mera descripción del lenguaje para lograr un tutorial completo, penetrante, impactante y que no se centra en los fabricante. He leído casi todos los demás —y sólo el tuyo y el de Patrick Winston han encontrado un lugar en mi corazón. Se lo estoy recomendando ya a los clientes. Gracias de nuevo. **Richard Brooks, consultor de Java, Sun Professional Services, Dallas.**

Otros libros contemplan o abarcan el QUÉ de Java (describiendo la sintaxis y las bibliotecas) o el CÓMO de Java (ejemplos de programación prácticos). *Piensa en Java*[1] es el único libro que conoz-

[1] Thinking in Java (título original de la obra en inglés).

co que explica el PORQUÉ de Java; por qué se diseñó de la manera que se hizo, por qué funciona como lo hace, por qué en ocasiones no funciona, por qué es mejor que C++, por qué no lo es. Aunque hace un buen trabajo de enseñanza sobre el qué y el cómo del lenguaje, *Piensa en Java* es la elección definitiva que toda persona interesada en Java ha de hacer. **Robert S. Stephenson.**

Gracias por escribir un gran libro. Cuanto más lo leo más me gusta. A mis estudiantes también les gusta. **Chuck Iverson.**

Sólo quiero comentarte tu trabajo en *Piensa en Java*. Es la gente como tú la que dignifica el futuro de Internet y simplemente quiero agradecerte el esfuerzo. **Patrick Barrell, Network Officer Mamco, QAF Mgf. Inc.**

La mayoría de libros de Java que existen están bien para empezar, y la mayoría tienen material para principiantes y muchos los mismos ejemplos. El tuyo es sin duda el mejor libro y más avanzado para pensar que he visto nunca. ¡Por favor, publícalo rápido!... También compré *Thinking in C++* simplemente por lo impresionado que me dejó *Piensa en Java*. **George Laframboise, LightWorx Technology Consulting Inc.**

Te escribí anteriormente con mis impresiones favorables relativas a *Piensa en Java* (un libro que empieza prominentemente donde hay que empezar). Y hoy que he podido meterme con Java con tu libro electrónico en mi mano virtual, debo decir (en mi mejor Chevy Chase de *Modern Problems*) "¡Me gusta!". Muy informativo y explicativo, sin que parezca que se lee un texto sin sustancia. Cubres los aspectos más importantes y menos tratados del desarrollo de Java: los porqués. **Sean Brady.**

Tus ejemplos son claros y fáciles de entender. Tuviste cuidado con la mayoría de detalles importantes de Java que no pueden encontrarse fácilmente en la débil documentación de Java. Y no malgastas el tiempo del lector con los hechos básicos que todo programador conoce. **Kai Engert, Innovative Software, Alemania.**

Soy un gran *fan* de *Piensa en Java* y lo he recomendado a mis asociados. A medida que avanzo por la versión electrónica de tu libro de Java, estoy averiguando que has retenido el mismo alto nivel de escritura. **Peter R. Neuvald.**

Un libro de Java MUY BIEN escrito... Pienso que has hecho un GRAN trabajo con él. Como líder de un grupo de interés especial en Java del área de Chicago, he mencionado de forma favorable tu libro y sitio web muy frecuentemente en mis últimas reuniones. Me gustaría usar *Piensa en Java* como la base de cada reunión mensual del grupo, para poder ir repasando y discutiendo sucesivamente cada capítulo. **Mark Ertes.**

Verdaderamente aprecio tu trabajo, y tu libro es bueno. Lo recomiendo aquí a nuestros usuarios y estudiantes de doctorado. **Hughes Leroy // Irisa-Inria Rennes France, jefe de Computación Científica y Transferencia Industrial.**

De acuerdo, sólo he leído unas 40 páginas de *Piensa en Java*, pero ya he averiguado que es el libro de programación mejor escrito y más claro que haya visto jamás... Yo también soy escritor, por lo que probablemente soy un poco crítico. Tengo *Piensa en Java* encargado y ya no puedo esperar más —soy bastante nuevo en temas de programación y no hago más que enfrentarme a curvas de

aprendizaje en todas partes. Por tanto, esto no es más que un comentario rápido para agradecerte este trabajo tan excelente. Ya me había empezado a quemar de tanto navegar por tanta y tanta prosa de tantos y tantos libros de computadores —incluso muchos que venían con magníficas recomendaciones. Me siento muchísimo mejor ahora. **Glenn Becker, Educational Theatre ssociation.**

Gracias por permitirme disponer de este libro tan maravilloso. Lo he encontrado inmensamente útil en el entendimiento final de lo que he experimentado —algo confuso anteriormente— con Java y C++. Leer tu libro ha sido muy gratificante. **Felix Bizaoui, Twin Oaks Industries, Luisa, Va.**

Debo felicitarte por tu excelente libro. He decidido echar un vistazo a *Piensa en Java* guiado por mi experiencia en *Thinking in C++*, y no me ha defraudado. **Jaco van der Merwe, Software Specialist, DataFusion Systems Ltd., Stellenbosch, Suráfrica.**

Este libro hace que todos los demás libros de Java que he leído parezcan un insulto o sin duda inútiles. **Brett g Porter, Senior Programmer, Art & Logic.**

He estado leyendo tu libro durante una semana o dos y lo he comparado con otros libros de Java que he leído anteriormente. Tu libro parece tener un gran comienzo. He recomendado este libro a muchos de mis amigos y todos ellos lo han calificado de excelente. Por favor, acepta mis felicitaciones por escribir un libro tan excelente. **Rama Krishna Bhupathi, Ingeniera de Software, TCSI Corporation, San José.**

Simplemente quería decir lo "brillante" que es tu libro. Lo he estado usando como referencia principal durante mi trabajo de Java hecho en casa. He visto que la tabla de contenidos es justo la más adecuada para localizar rápidamente la sección que se requiere en cada momento. También es genial ver un libro que no es simplemente una compilación de las API o que no trata al programador como a un monigote. **Grant Sayer, Java Components Group Leader, Ceedata Systems Pty Ltd., Australia.**

¡Guau! Un libro de Java profundo y legible. Hay muchos libros pobres (y debo admitir también que un par de ellos buenos) de Java en el mercado, pero por lo que he visto, el tuyo es sin duda uno de los mejores. **John Root, desarrollador Web, Departamento de la Seguridad Social, Londres.**

Acabo de empezar *Piensa en Java*. Espero que sea bueno porque me gustó mucho *Thinking in C++* (que leí como programador ya experimentado en C++, intentado adelantarme a la curva de aprendizaje). En parte estoy menos habituado a Java, pero espero que el libro me satisfaga igualmente. Eres un autor maravilloso. **Kevin K. Lewis, Tecnólogo, ObjectSpace Inc.**

Creo que es un gran libro. He aprendido todo lo que sé de Java a partir de él. Gracias por hacerlo disponible gratuitamente a través de Internet. Si no lo hubieras hecho no sabría nada de Java. Pero lo mejor es que tu libro no es un mero folleto publicitario de Java. También muestra sus lados negativos. TÚ has hecho aquí un gran trabajo. **FrederikFix, Bélgica.**

Siempre me han enganchado tus libros. Hace un par de años, cuando quería empezar con C++, fue *C++ Inside & Out* el que me introdujo en el fascinante mundo de C++. Me ayudó a disponer de mejores oportunidades en la vida. Ahora, persiguiendo más conocimiento y cuando quería aprender Java, me introduje en *Piensa en Java* —sin dudar de que gracias a él ya no necesitaría ningún otro libro. Simplemente fantástico. Es casi como volver a descubrirme a mí mismo a medida que avanzo

en el libro. Apenas hace un mes que he empezado con Java y mi corazón late gracias a ti. Ahora lo entiendo todo mucho mejor. **Anand Kumar S., ingeniero de Software Computervision, India.**

Tu libro es una introducción general excelente. **Peter Robinson, Universidad de Cambridge, Computar Laboratory.**

Es con mucho el mejor material al que he tenido acceso al aprender Java y simplemente quería que supieras la suerte que he tenido de poder encontrarlo. ¡GRACIAS! **Chuck Peterson, Product Leader, Internet Product Line, IVIS International.**

Este libro es genial. Es el tercer libro de Java que he empezado y ya he recorrido prácticamente dos tercios. Espero acabar éste. Me he enterado de su existencia porque se usa en algunas clases internas de Lucen Technologies y un amigo me ha dicho que el libro estaba en la Red. Buen trabajo. **Jerry Nowlin, MTS, Lucent Technologies.**

De los aproximadamente seis libros de Java que he acumulado hasta la fecha, tu *Piensa en Java* es sin duda el mejor y el más claro. **Michael Van Waas, doctor, presidente, TMR Associates.**

Simplemente quiero darte las gracias por *Piensa en Java*. ¡Qué libro tan maravilloso has hecho! ¡Y para qué mencionar el poder bajárselo gratis! Como estudiante creo que tus libros son de valor incalculable, tengo una copia de *C++ Inside & Out*, otro gran libro sobre C++), porque no sólo me enseñan el cómo hacerlo, sino que también los porqués, que sin duda son muy importantes a la hora de sentar unas buenas bases en lenguajes como C++ y Java. Tengo aquí bastantes amigos a los que les encanta programar como a mí, y les he hablado de tus libros. ¡Todos piensan que son geniales! Por cierto, soy indonesio y vivo en Java. **Ray Frederick Djajadinata, estudiante en Trisakti University, Jakarta.**

El mero hecho de que hayas hecho que este trabajo esté disponible gratuitamente en la Red me deja conmocionado. Pensé que debía decirte cuánto aprecio y respeto lo que estás haciendo. **Shane KeBouthillier, estudiante de Ingeniería en Informática, Universidad de Alberta, Canadá.**

Tengo que decirte cuánto ansío leer tu columna mensual. Como novato en el mundo de la programación orientada a objetos, aprecio el tiempo y el grado de conocimiento que aportas en casi todos los temas elementales. He descargado tu libro, pero puedes apostar a que compraré una copia en papel en cuanto se publique. Gracias por toda tu ayuda. **Dan Cashmer, D. C. Ziegler & Co.**

Simplemente quería felicitarte por el buen trabajo que has hecho. Primero me recorrí la versión PDF de *Piensa en Java*. Incluso antes de acabar de leerla, corrí a la tienda y compré *Thinking in C++*. Ahora que llevo en el negocio de la informática ocho años, como consultor, ingeniero de software, profesor/formador, y últimamente autónomo, creo que puedo decir que he visto suficiente (fíjate que no digo haber visto "todo" sino suficiente). Sin embargo, estos libros hacen que mi novia me llame "geek". No es que tenga nada contra el concepto en sí —simplemente pensaba que ya había dejado atrás esta fase. Pero me veo a mí mismo disfrutando sinceramente de ambos libros, de una forma que no había sentido con ningún otro libro que haya tocado o comprado hasta la fecha. Un estilo de escritura excelente, una introducción genial de todos los temas y mucha sabiduría en ambos textos. Bien hecho. **Simon Goland, simonsez@smartt.com, Simon Says Consulting, Inc.**

¡Debo decir que tu *Piensa en Java* es genial! Es exactamente el tipo de documentación que buscaba. Especialmente las secciones sobre los buenos y malos diseños basados en Java. **Dirk Duehr, Lexikon Verlag, Bertelsmann AG, Alemania.**

Gracias por escribir dos grandes libros (*Thinking in C++, Piensa en Java*). Me has ayudado inmensamente en mi progresión en la programación orientada a objetos. **Donald Lawon, DCL Enterprises.**

Gracias por tomarte el tiempo de escribir un libro de Java que ayuda verdaderamente. Si enseñar hace que aprendas algo, tú ya debes estar más que satisfecho. **Dominic Turner, GEAC Support.**

Es el mejor libro de Java que he leído jamás —y he leído varios. **Jean-Yves MENGANT, Chief Software Architect NAT-SYSTEM, París, Francia.**

Piensa en Java proporciona la mejor cobertura y explicación. Muy fácil de leer, y quiero decir que esto se extiende también a los fragmentos de código. **Ron Chan, Ph. D., Expert Choice Ind., Pittsburg PA.**

Tu libro es genial. He leído muchos libros de programación y el tuyo sigue añadiendo luz a la programación en mi mente. **Ningjian Wang, Information System Engineer, The Vanguard Group.**

Piensa en Java es un libro excelente y legible. Se lo recomiendo a todos mis alumnos. **Dr. Paul Gorman, Department of Computer Sciente, Universidad de Otago, Dunedin, Nueva Zelanda.**

Haces posible que exista el proverbial almuerzo gratuito, y no simplemente una comida basada en sopa de pollo, sino una delicia de *gourmet* para aquéllos que aprecian el buen software y los libros sobre él mismo. **Jose Suriol, Scylax Corporation.**

¡Gracias por la oportunidad de ver cómo este libro se convierte en una obra maestra! ES EL MEJOR libro de la materia que he leído o recorrido. **Jeff Lapchinsky, programador, Net Result Tecnologies.**

Tu libro es conciso, accesible y gozoso de leer. **Keith Ritchie, Java Research & Develpment Team, KL Group Inc.**

¡Es sin duda el mejor libro de Java que he leído! **Daniel Eng.**

¡Es el mejor libro de Java que he visto! **Rich Hoffarth, Arquitecto Senior, West Group.**

Gracias por un libro tan magnífico. Estoy disfrutando mucho a medida que leo capítulos. **Fred Trimble, Actium Corporation.**

Has llegado a la maestría en el arte de hacernos ver los detalles, despacio y con éxito. Haces que la lectura sea MUY fácil y satisfactoria. Gracias por un tutorial tan verdaderamente maravilloso. **Rajesh Rau, Software Consultant.**

Piensa en Java es un rock para el mundo libre! **Miko O'Sullivan, Presidente, Idocs Inc.**

Sobre Thinking in C++:

Best Book! Ganador en 1995 del Software Development Magazine Jolt Award!

"Este libro es un tremendo logro. Deberías tener una copia en la estantería. El capítulo sobre flujos de E/S presenta el tratamiento más comprensible y fácil de entender sobre ese tema que jamás haya visto."

Al Stevens
Editor, *Doctor Dobbs Journal*

"El libro de Eckel es el único que explica claramente cómo replantearse la construcción de programas para la orientación a objetos. Que el libro es también un tutorial excelente en las entradas y en las salidas de C++ es un valor añadido."

Andrew Binstock
Editor, *Unix Review*

"Bruce continúa deleitándome con esta introspección de C++, y *Thinking in C++* es la mejor colección de ideas hasta la fecha. Si se desean respuestas rápidas a preguntas difíciles sobre C++, compre este libro tan sobresaliente."

Gary Entsminger
Autor, *The Tao of Objects*

"*Thinking in C++*" explora paciente y metódicamente los aspectos de cuándo y cómo usar los interlineados, referencias, sobrecargas de operadores, herencia, y objetos dinámicos, además de temas avanzados como el uso adecuado de plantillas, excepciones y la herencia múltiple. Todo el esfuerzo se centra en un producto que engloba la propia filosofía de Eckel del diseño de objetos y programas. Un libro que no debe faltar en la librería de un desarrollador de C++, *Piensa en Java* es el libro de C++ que hay que tener si se están haciendo desarrollos serios con C++."

Richard Hale Shaw
Ayudante del Editor, *PC Magazine*

Introducción

Como cualquier lenguaje humano, Java proporciona una forma de expresar conceptos. Si tiene éxito, la expresión media será significativamente más sencilla y más flexible que las alternativas, a medida que los problemas crecen en tamaño y complejidad.

No podemos ver Java como una simple colección de características —algunas de las características no tienen sentido aisladas. Se puede usar la suma de partes sólo si se está pensando en *diseño*, y no simplemente en codificación. Y para entender Java así, hay que entender los problemas del lenguaje y de la programación en general. Este libro habla acerca de problemas de programación, por qué son problemas y el enfoque que Java sigue para solucionarlos. Por consiguiente, algunas características que explico en cada capítulo se basan en cómo yo veo que se ha solucionado algún problema en particular con el lenguaje. Así, espero conducir poco a poco al lector, hasta el punto en que Java se convierta en lengua casi materna.

Durante todo el tiempo, estaré tomando la actitud de que el lector construya un modelo mental que le permita desarrollar un entendimiento profundo del lenguaje; si se encuentra un puzzle se podrá alimentar de éste al modelo para tratar de deducir la respuesta.

Prerrequisitos

Este libro asume que se tiene algo de familiaridad con la programación: se entiende que un programa es una colección de sentencias, la idea de una subrutina/función/macro, sentencias de control como "if" y bucles estilo "while", etc. Sin embargo, se podría haber aprendido esto en muchos sitios, como, por ejemplo, la programación con un lenguaje de macros o el trabajo con una herramienta como Perl. A medida que se programa hasta el punto en que uno se siente cómodo con las ideas básicas de programación, se podrá ir trabajando a través de este libro. Por supuesto, el libro será más fácil para los programadores de C y aún más para los de C++, pero tampoco hay por qué excluirse a sí mismo cuando se desconocen estos lenguajes (aunque en este caso es necesario tener la voluntad de trabajar duro; además, el CD multimedia que acompaña a este texto te permitirá conocer rápidamente los conceptos de la sintaxis de C necesarios para aprender Java). Presentaré los conceptos de la programación orientada a objetos (POO) y los mecanismos de control básicos de Java, para tener conocimiento de ellos, y los primeros ejercicios implicarán las secuencias de flujo de control básicas.

Aunque a menudo aparecerán referencias a aspectos de los lenguajes C y C++, no deben tomarse como comentarios profundos, sino que tratan de ayudar a los programadores a poner Java en perspectiva con esos lenguajes, de los que, después de todo, es de los que desciende Java. Intentaré hacer que estas referencias sean lo más simples posibles, y explicar cualquier cosa que crea que una persona que no haya programado nunca en C o C++ pueda desconocer.

Aprendiendo Java

Casi a la vez que mi primer libro *Using C++* (Osborne/McGraw-Hill, 1989)apareció, empecé a enseñar ese lenguaje. Enseñar lenguajes de programación se ha convertido en mi profesión; he visto cabezas dudosas, caras en blanco y expresiones de puzzle en audiencias de todo el mundo desde 1989. A medida que empecé con formación *in situ* a grupos de gente más pequeños, descubrí algo en los ejercicios. Incluso aquéllos que sonreían tenían pegas con muchos aspectos. Al dirigir la sesión de C++ en la *Software Development Conference* durante muchos años (y después la sesión de Java), descubrí que tanto yo como otros oradores tendíamos a ofrecer a la audiencia, en general, muchos temas demasiado rápido. Por tanto, a través, tanto de la variedad del nivel de audiencia como de la forma de presentar el material, siempre se acababa perdiendo parte de la audiencia. Quizás es pedir demasiado, pero dado que soy uno de ésos que se resisten a las conferencias tradicionales (y en la mayoría de casos, creo que esta resistencia proviene del aburrimiento), quería intentar algo que permitiera tener a todo el mundo enganchado.

Durante algún tiempo, creé varias presentaciones diferentes en poco tiempo. Por consiguiente, acabé aprendiendo a base de experimentación e iteración (una técnica que también funciona bien en un diseño de un programa en Java). Eventualmente, desarrollé un curso usando todo lo que había aprendido de mi experiencia en la enseñanza —algo que me gustaría hacer durante bastante tiempo. Descompone el problema de aprendizaje en pasos discretos, fáciles de digerir, y en un seminario en máquina (la situación ideal de aprendizaje) hay ejercicios seguidos cada uno de pequeñas lecciones. Ahora doy cursos en seminarios públicos de Java, que pueden encontrarse en http://www.BruceEckel.com. (El seminario introductorio también está disponible como un CD ROM. En el sitio web se puede encontrar más información al respecto.)

La respuesta que voy obteniendo de cada seminario me ayuda a cambiar y reenfocar el material hasta que creo que funciona bien como medio docente. Pero este libro no es simplemente un conjunto de notas de los seminarios —intenté empaquetar tanta información como pude en este conjunto de páginas, estructurándola de forma que cada tema te vaya conduciendo al siguiente. Más que otra cosa, el libro está diseñado para servir al lector solitario que se está enfrentando y dando golpes con un nuevo lenguaje de programación.

Objetivos

Como en mi libro anterior *Thinking in C++*, este libro pretende estar estructurado en torno al proceso de enseñanza de un lenguaje. En particular, mi motivación es crear algo que me proporcione una forma de enseñar el lenguaje en mis propios seminarios. Cuando pienso en un capítulo del libro, lo pienso en términos de lo que constituiría una buena lección en un seminario. Mi objetivo es lograr fragmentos que puedan enseñarse en un tiempo razonable, seguidos de ejercicios que sean fáciles de llevar a cabo en clase.

Mis objetivos en este libro son:

1. Presentar el material paso a paso de forma que se pueda digerir fácilmente cada concepto antes de avanzar.

2. Utilizar ejemplos que sean tan simples y cortos como se pueda. Esto evita en ocasiones acometer problemas del "mundo real", pero he descubierto que los principiantes suelen estar más contentos cuando pueden entender todos los detalles de un ejemplo que cuando se ven impresionados por el gran rango del problema que solucionan. Además, hay una limitación severa de cara a la cantidad de código que se puede absorber en una clase. Por ello, no dudaré en recibir críticas por usar "ejemplos de juguete", sino que estoy deseoso de aceptarlas en aras de lograr algo pedagógicamente útil.

3. Secuenciar cuidadosamente la presentación de características de forma que no se esté viendo algo que aún no se ha expuesto. Por supuesto, esto no es siempre posible; en esas situaciones se dan breves descripciones introductorias.

4. Dar lo que yo considero que es importante que se entienda del lenguaje, en lugar de todo lo que sé. Creo que hay una jerarquía de importancia de la información, y que hay hechos que el 95% de los programadores nunca necesitarán saber y que simplemente confunden a la gente y añaden su percepción de la complejidad del lenguaje. Por tomar un ejemplo de C, si se memoriza la tabla de precedencia de los operadores (algo que yo nunca hice) se puede escribir un código más inteligente. Pero si se piensa en ello, también confundirá la legibilidad y mantenibilidad de ese código. Por tanto, hay que olvidarse de la precedencia, y usar paréntesis cuando las cosas no estén claras.

5. Mantener cada sección lo suficientemente enfocada de forma que el tiempo de exposición —el tiempo entre periodos de ejercicios— sea pequeño. Esto no sólo mantiene más activas las mentes de la audiencia, que están en un seminario en máquina, sino que también transmite más sensación de avanzar.

6. Proporcionar una base sólida que permita entender los aspectos lo suficientemente bien como para avanzar a cursos y libros más difíciles.

Documentación en línea

El lenguaje Java y las bibliotecas de Sun Microsystems (de descarga gratuita) vienen con su documentación en forma electrónica, legible utilizando un navegador web, y casi toda implementación de Java de un tercero tiene éste u otro sistema de documentación equivalente. Casi todos los libros publicados de Java, incorporan esta documentación. Por tanto, o ya se tiene, o se puede descargar, y a menos que sea necesario, este libro no repetirá esa documentación pues es más rápido encontrar las descripciones de las clases en el navegador web que buscarlas en un libro (y la documentación en línea estará probablemente más actualizada). Este libro proporcionará alguna descripción extra de las clases sólo cuando sea necesario para complementar la documentación, de forma que se pueda entender algún ejemplo particular.

Capítulos

Este libro se diseñó con una idea en la cabeza: la forma que tiene la gente de aprender Java. La realimentación de la audiencia de mis seminarios me ayudó a ver las partes difíciles que necesitaban aclaraciones. En las áreas en las que me volvía ambiguo e incluía varias características a la vez, descubrí —a través del proceso de presentar el material— que si se incluyen muchas características de golpe, hay que explicarlas todas, y esto suele conducir fácilmente a la confusión por parte del alumno. Como resultado, he tenido bastantes problemas para presentar las características agrupadas de tan pocas en pocas como me ha sido posible.

El objetivo, por tanto, es que cada capítulo enseñe una única característica, o un pequeño grupo de características asociadas, sin pasar a características adicionales. De esa forma se puede digerir cada fragmento en el contexto del conocimiento actual antes de continuar.

He aquí una breve descripción de los capítulos que contiene el libro, que corresponde a las conferencias y periodos de ejercicio en mis seminarios en máquina.

Capítulo 1: Introducción a los objetos

Este capítulo presenta un repaso de lo que es la programación orientada a objetos, incluyendo la respuesta a la cuestión básica "¿Qué es un objeto?", interfaz frente a implementación, abstracción y encapsulación, mensajes y funciones, herencia y composición, y la importancia del polimorfismo. También se obtendrá un repaso a los aspectos de la creación de objetos como los constructores, en los que residen los objetos, dónde ponerlos una vez creados, y el mágico recolector de basura que limpia los objetos cuando dejan de ser necesarios. Se presentarán otros aspectos, incluyendo el manejo de errores con excepciones, el multihilo para interfaces de usuario con buen grado de respuesta, y las redes e Internet. Se aprenderá qué convierte a Java en especial, por qué ha tenido tanto éxito, y también algo sobre análisis y diseño orientado a objetos.

Capítulo 2: Todo es un objeto

Este capítulo te lleva al punto donde tú puedas crear el primer programa en Java, por lo que debe dar un repaso a lo esencial, incluyendo el concepto de *referencia* a un objeto; cómo crear un objeto; una introducción de los tipos primitivos y arrays; el alcance y la forma en que destruye los objetos el recolector de basura; cómo en Java todo es un nuevo tipo de datos (clase) y cómo crear cada uno sus propias clases; funciones, argumentos y valores de retorno; visibilidad de nombres y el uso de componentes de otras bibliotecas; la palabra clave **static**; y los comentarios y documentación embebida.

Capítulo 3: Controlando el flujo de los programas

Este capítulo comienza con todos los operadores que provienen de C y C++. Además, se descubrirán los fallos de los operadores comunes, la conversión de tipos, la promoción y la precedencia. Des-

pués se presentan las operaciones básicas de control de flujo y selección existentes en casi todos los lenguajes de programación: la opción con *if-else*; los bucles con *while* y *for*; cómo salir de un bucle con *break* y *continue*, además de sus versiones etiquetadas en Java (que vienen a sustituir al "goto perdido" en Java); la selección con *switch*. Aunque gran parte de este material tiene puntos comunes con el código de C y C++, hay algunas diferencias. Además, todos los ejemplos estarán hechos completamente en Java por lo que el lector podrá estar más a gusto con la apariencia de Java.

Capítulo 4: Inicialización y limpieza

Este capítulo comienza presentando el constructor, que garantiza una inicialización adecuada. La definición de constructor conduce al concepto de sobrecarga de funciones (puesto que puede haber varios constructores). Éste viene seguido de una discusión del proceso de limpieza, que no siempre es tan simple como parece. Normalmente, simplemente se desecha un objeto cuando se ha acabado con él y el recolector de basura suele aparecer para liberar la memoria. Este apartado explora el recolector de basura y algunas de sus idiosincrasias. El capítulo concluye con un vistazo más cercano a cómo se inicializan las cosas: inicialización automática de miembros, especificación de inicialización de miembros, el orden de inicialización, la inicialización **static** y la inicialización de arrays.

Capítulo 5: Ocultando la implementación

Este capítulo cubre la forma de empaquetar junto el código, y por qué algunas partes de una biblioteca están expuestas a la vez que otras partes están ocultas. Comienza repasando las palabras clave **package** e **import**, que llevan a cabo empaquetado a nivel de archivo y permiten construir bibliotecas de clases. Después examina el tema de las rutas de directorios y nombres de fichero. El resto del capítulo echa un vistazo a las palabras clave **public**, **private** y **protected**, el concepto de acceso "friendly", y qué significan los distintos niveles de control de acceso cuando se usan en los distintos conceptos.

Capítulo 6: Reutilizando clases

El concepto de herencia es estándar en casi todos los lenguajes de POO. Es una forma de tomar una clase existente y añadirla a su funcionalidad (además de cambiarla, que será tema del Capítulo 7). La herencia es a menudo una forma de reutilizar código dejando igual la "clase base", y simplemente parcheando los elementos aquí y allí hasta obtener lo deseado. Sin embargo, la herencia no es la única forma de construir clases nuevas a partir de las existentes. También se puede empotrar un objeto dentro de una clase nueva con la *composición*. En este capítulo, se aprenderán estas dos formas de reutilizar código en Java, y cómo aplicarlas.

Capítulo 7: Polimorfismo

Cada uno por su cuenta, podría invertir varios meses para descubrir y entender el polimorfismo, claves en POO. A través de pequeños ejemplos simples, se verá cómo crear una familia de tipos con

herencia y manipular objetos de esa familia a través de su clase base común. El polimorfismo de Java permite tratar los objetos de una misma familia de forma genérica, lo que significa que la mayoría del código no tiene por qué depender de un tipo de información específico. Esto hace que los programas sean extensibles, por lo que se facilita y simplifica la construcción de programas y el mantenimiento de código.

Capítulo 8: Interfaces y clases internas

Java proporciona una tercera forma de establecer una relación de reutilización a través de la *interfaz*, que es una abstracción pura del interfaz de un objeto. La **interfaz** es más que una simple clase abstracta llevada al extremo, puesto que te permite hacer variaciones de la "herencia múltiple" de C++, creando una clase sobre la que se puede hacer una conversión hacia arriba a más de una clase base.

A primera vista, las clases parecen un simple mecanismo de ocultación de código: se colocan clases dentro de otras clases. Se aprenderá, sin embargo, que la clase interna hace más que eso —conoce y puede comunicarse con la clase contenedora— y que el tipo de código que se puede escribir con clases internas es más elegante y limpio, aunque es un concepto nuevo para la mayoría de la gente y lleva tiempo llegar a estar cómodo utilizando el diseño clases internas.

Capítulo 9: Guardando tus objetos

Es un programa bastante simple que sólo tiene una cantidad fija de objetos de tiempo de vida conocido. En general, todos los programas irán creando objetos nuevos en distintos momentos, conocidos sólo cuando se está ejecutando el programa. Además, no se sabrá hasta tiempo de ejecución la cantidad o incluso el tipo exacto de objetos que se necesitan. Para solucionar el problema de programación general, es necesario crear cualquier número de objetos, en cualquier momento y en cualquier lugar. Este capítulo explora en profundidad la biblioteca de contenedores que proporciona Java 2 para almacenar objetos mientras se está trabajando con ellos: los simples arrays y contenedores más sofisticados (estructuras de datos) como **ArrayList** y **HashMap**.

Capítulo 10: Manejo de errores con excepciones

La filosofía básica de Java es que el código mal formado no se ejecutará. En la medida en que sea posible, el compilador detecta problemas, pero en ocasiones los problemas —debidos a errores del programador o a condiciones de error naturales que ocurren como parte de la ejecución normal del programa— pueden detectarse y ser gestionados sólo en tiempo de ejecución. Java tiene el *manejo de excepciones* para tratar todos los problemas que puedan surgir al ejecutar el programa. Este capítulo muestra cómo funcionan en Java las palabras clave **try**, **catch**, **throw**, **throws** y **finally**; cuándo se deberían lanzar excepciones y qué hacer al capturarlas. Además, se verán las excepciones estándar de Java, cómo crear las tuyas propias, qué ocurre con las excepciones en los constructores y cómo se ubican los gestores de excepciones.

Capítulo 11: El sistema de E/S de Java

Teóricamente, se puede dividir cualquier programa en tres partes: entrada, proceso y salida. Esto implica que la E/S (entrada/salida) es una parte importante de la ecuación. En este capítulo se aprenderá las distintas clases que proporciona Java para leer y escribir ficheros, bloques de memoria y la consola. También se mostrará la distinción entre E/S "antigua" y "nueva". Además, este capítulo examina el proceso de tomar un objeto, pasarlo a una secuencia de bytes (de forma que pueda ser ubicado en el disco o enviado a través de una red) y reconstruirlo, lo que realiza automáticamente la *serialización de objetos de Java.* Además, se examinan las bibliotecas de compresión de Java, que se usan en el formato de archivos de Java (JAR).

Capítulo 12: Identificación de tipos en tiempo de ejecución

La identificación de tipos en tiempo de ejecución (RTTI) te permite averiguar el tipo exacto de un objeto cuando se tiene sólo una referencia al tipo base. Normalmente, se deseará ignorar intencionadamente el tipo exacto de un objeto y dejar que sea el mecanismo de asignación dinámico de Java (polimorfismo) el que implemente el comportamiento correcto para ese tipo. A menudo, esta información te permite llevar a cabo operaciones de casos especiales, más eficientemente. Este capítulo explica para qué existe la RTTI, cómo usarlo, y cómo librarse de él cuando sobra. Además, este capítulo presenta el mecanismo de *reflectividad* de Java.

Capítulo 13: Creación de ventanas y applets

Java viene con la biblioteca IGU Swing, que es un conjunto de clases que manejan las ventanas de forma portable. Estos programas con ventanas pueden o bien ser applets o bien aplicaciones independientes. Este capítulo es una introducción a Swing y a la creación de applets de World Wide Web. Se presenta la importante tecnología de los "JavaBeans", fundamental para la creación de herramientas de construcción de programas de Desarrollo Rápido de Aplicaciones (RAD).

Capítulo 14: Hilos múltiples

Java proporciona una utilidad preconstruida para el soporte de múltiples subtareas concurrentes denominadas *hilos*, que se ejecutan en un único programa. (A menos que se disponga de múltiples procesadores en la máquina, los múltiples hilos sólo son *aparentes.*) Aunque éstas pueden usarse en todas partes, los hilos son más lucidos cuando se intenta crear una interfaz de usuario con alto grado de respuesta, de forma que, por ejemplo, no se evita que un usuario pueda presionar un botón o introducir datos mientras se está llevando a cabo algún procesamiento. Este capítulo echa un vistazo a la sintaxis y la semántica del multihilo en Java.

Capítulo 15: Computación distribuida

Todas las características y bibliotecas de Java aparecen realmente cuando se empieza a escribir programas que funcionen en red. Este capítulo explora la comunicación a través de redes e Internet, y las clases que proporciona Java para facilitar esta labor. Presenta los tan importantes conceptos de *Servlets* y *JSP* (para programación en el lado servidor), junto con *Java DataBase Connectivity* (JDBC) y el *Remote Method Invocation* (RMI). Finalmente, hay una introducción a las nuevas tecnologías de *JINI, JavaSpaces*, y *Enterprise JavaBeans (EJBS)*.

Apéndice A: Paso y retorno de objetos

Puesto que la única forma de hablar con los objetos en Java es mediante referencias, los conceptos de paso de objetos a una función y de devolución de un objeto de una función tienen algunas consecuencias interesantes. Este apéndice explica lo que es necesario saber para gestionar objetos cuando se está entrando y saliendo de funciones, y también muestra la clase **String**, que usa un enfoque distinto al problema.

Apéndice B: La Interfaz Nativa de Java (JNI)

Un programa Java totalmente portable tiene importantes pegas: la velocidad y la incapacidad para acceder a servicios específicos de la plataforma. Cuando se conoce la plataforma sobre la que está ejecutando, es posible incrementar dramáticamente la velocidad de ciertas operaciones construyéndolas como *métodos nativos*, que son funciones escritas en otro lenguaje de programación (actualmente, sólo están soportados C/C++). Este apéndice da una introducción más que satisfactoria que debería ser capaz de crear ejemplos simples que sirvan de interfaz con código no Java.

Apéndice C: Guías de programación Java

Este apéndice contiene sugerencias para guiarle durante la realización del diseño de programas de bajo nivel y la escritura de código.

Apéndice D: Lecturas recomendadas

Una lista de algunos libros sobre Java que he encontrado particularmente útil.

Ejercicios

He descubierto que los ejercicios simples son excepcionalmente útiles para completar el entendimiento de los estudiantes durante un seminario, por lo que se encontrará un conjunto de ellos al final de cada capítulo.

La mayoría de ejercicios están diseñados para ser lo suficientemente sencillos como para poder ser resueltos en un tiempo razonable en una situación de clase mientras que observa el profesor, asegurándose de que todos los alumnos asimilen el material. Algunos ejercicios son más avanzados para evitar que los alumnos experimentados se aburran. La mayoría están diseñados para ser resueltos en poco tiempo y probar y pulir el conocimiento. Algunos suponen un reto, pero ninguno presenta excesivas dificultades. (Presumiblemente, cada uno podrá encontrarlos —o más probablemente te encontrarán ellos a ti.)

En el documento electrónico *The Thinking in Java Annotated Solution Guide* pueden encontrarse soluciones a ejercicios seleccionados, disponibles por una pequeña tasa en http://www.BruceEckel.com.

CD ROM Multimedia

Hay dos CD multimedia asociados con este libro. El primero está en el propio libro: *Thinking in C*, descritos al final del prefacio, que te preparan para el libro aportando velocidad en la sintaxis de C necesaria para poder entender Java.

Hay disponible un segundo CD ROM multimedia, basado en los contenidos del libro. Este CD ROM es un producto separado y contiene los contenidos **enteros** del seminario de formación "Hands-On Java" de una semana de duración. Esto son grabaciones de conferencias de más de 15 horas que he grabado, y sincronizado con cientos de diapositivas de información. Dado que el seminario se basa en este libro, es el acompañamiento ideal.

El CD ROM contiene todas las conferencias (¡con la importante excepción de la atención personalizada!) de los seminarios de cinco días de inmersión total. Creemos que establece un nuevo estándar de calidad.

El CD ROM "Hands-On Java" está disponible sólo bajo pedido, que se cursa directamente del sitio web *http://www.BruceEckel.com*.

Código fuente

Todo el código fuente de este libro está disponible de modo gratuito sometido a *copyright*, distribuido como un paquete único, visitando el sitio web *http://www.BruceEckel.com*. Para asegurarse de obtener la versión más actual, éste es el lugar oficial para distribución del código y de la versión electrónica del libro. Se pueden encontrar versiones espejo del código y del libro en otros sitios (algunos de éstos están referenciados en *http://www.BruceEckel.com)*, pero habría que comprobar el sitio oficial para asegurarse de obtener la edición más reciente. El código puede distribuirse en clases y en otras situaciones con fines educativos.

La meta principal del *copyright* es asegurar que el código fuente se cite adecuadamente, y prevenir que el código se vuelva a publicar en medios impresos sin permiso. (Mientras se cite la fuente, utilizando los ejemplos del libro, no habrá problema en la mayoría de los medios.)

En cada fichero de código fuente, se encontrará una referencia a la siguiente nota de *copyright*:

```
//:! :CopyRght.txt
Copyright (c)2000 Bruce Eckel
Source code file from the 2nd edition of the book
"Thinking in Java." All rights reserved EXCEPT as
allowed by the following statements:
You can freely use this file
for your own work (personal or commercial),
including modifications and distribution in
executable form only. Permission is granted to use
this file in classroom situations, including its
use in presentation materials, as long as the book
"Thinking in Java" is cited as the source.
Except in classroom situations, you cannot copy
and distribute this code; instead, the sole
distribution point is http://www.BruceEckel.com
(and official mirror sites) where it is
freely available. You cannot remove this
copyright and notice. You cannot distribute
modified versions of the source code in this
package. You cannot use this file in printed
media without the express permission of the
author. Bruce Eckel makes no representation about
the suitability of this software for any purpose.
It is provided "as is" without express or implied
warranty of any kind, including any implied
warranty of merchantability, fitness for a
particular purpose or non-infringement. The entire
risk as to the quality and performance of the
software is with you. Bruce Eckel and the
publisher shall not be liable for any damages
suffered by you or any third party as a result of
using or distributing software. In no event will
Bruce Eckel or the publisher be liable for any
lost revenue, profit, or data, or for direct,
indirect, special, consequential, incidental, or
punitive damages, however caused and regardless of
the theory of liability, arising out of the use of
or inability to use software, even if Bruce Eckel
and the publisher have been advised of the
possibility of such damages. Should the software
prove defective, you assume the cost of all
necessary servicing, repair, or correction. If you
think you've found an error, please submit the
correction using the form you will find at
www.BruceEckel.com. (Please use the same
form for non-code errors found in the book.)
///:~
```

El código puede usarse en proyectos y en clases (incluyendo materiales de presentación) mientras se mantenga la observación de *copyright* que aparece en cada archivo fuente.

Estándares de codificación

En el texto de este libro, los identificadores (nombres de funciones, variables y clases) están en **negrita**. La mayoría de palabras clave también están en negrita, excepto en aquellos casos en que las palabras se usan tanto que ponerlas en negrita podría volverse tedioso, como es el caso de la palabra "clase".

Para los ejemplos de este libro, uso un estilo de codificación bastante particular. Este estilo sigue al estilo que la propia Sun usa en prácticamente todo el código de sitio web (véase *http://java.sun.com/docs/codeconv/index.html*), y parece que está soportado por la mayoría de entornos de desarrollo Java. Si ha leído el resto de mis trabajos, también verá que el estilo de codificación de Sun coincide con el mío —esto me alegra, aunque no tenía nada que hacer con él. El aspecto del estilo de formato es bueno para lograr horas de tenso debate, por lo que simplemente diré que no pretendo dictar un estilo correcto mediante mis ejemplos; tengo mi propia motivación para usar el estilo que uso. Java es un lenguaje de programación de forma libre, se puede seguir usando cualquier estilo con el que uno esté a gusto.

Los programas de este libro son archivos incluidos por el procesador de textos, directamente sacados de archivos compilados. Por tanto, los archivos de código impresos en este libro deberían funcionar sin errores de compilador. Los errores que *deberían* causar mensajes de error en tiempo de compilación están comentados o marcados mediante //!, por lo que pueden ser descubiertos fácilmente, y probados utilizando medios automáticos. Los errores descubiertos de los que ya se haya informado al autor, aparecerán primero en el código fuente distribuido y posteriormente en actualizaciones del libro (que también aparecerán en el sitio web *http://www.BruceEckel.com*).

Versiones de Java

Generalmente confío en la implementación que Sun hace de Java como referencia para definir si un determinado comportamiento es o no correcto.

Con el tiempo, Sun ha lanzado tres versiones principales de Java: la 1.0, la 1.1 y la 2 (que se llama versión 2, incluso aunque las versiones del JDK de Sun siguen usando el esquema de numeración de 1.2, 1.3, 1.4, etc.). La versión 2 parece llevar finalmente a Java a la gloria, especialmente en lo que concierne a las herramientas de interfaces. Este libro se centra en, y está probado con, Java 2, aunque en ocasiones hago concesiones a las características anteriores de Java 2, de forma que el código pueda compilarse bajo Linux (vía el JDK de Linux que estaba disponible en el momento de escribir el libro).

Si se necesita aprender versiones anteriores del lenguaje no cubiertas en esta edición, la primera edición del libro puede descargarse gratuitamente de *http://www.BruceEckel.com*, y también está en el CD adjunto a este libro.

Algo de lo que uno se dará cuenta es de que, cuando menciono versiones anteriores del lenguaje, no uso los números de sub-revisión. En este libro me referiré sólo a Java 1.0, 1.1 y 2, para protegerme de errores tipográficos producidos por sub-revisiones posteriores de estos productos.

Seminarios y mi papel como mentor

Mi empresa proporciona seminarios de formación de cinco días, en máquina, públicos e *in situ*, basados en el material de este libro. Determinado material de cada capítulo representa una lección, seguida de un periodo de ejercicios guiados de forma que cada alumno recibe atención personal. Las conferencias y las diapositivas del seminario introductorio también están en el CD ROM para proporcional al menos alguna de la experiencia del seminario sin el viaje y el coste que conllevaría. Para más información, visitar *http://www.BruceEckel.com*.

Mi compañía también proporciona consultoría, servicios de orientación y acompañamiento para ayudar a guiar un proyecto a lo largo de su ciclo de desarrollo —especialmente indicado para el primer proyecto en Java de una empresa.

Errores

Sin que importe cuántos trucos utiliza un escritor para detectar errores, siempre hay alguno que se queda ahí y que algún lector encontrará.

Hay un formulario para remitir errores al principio de cada capítulo en la versión HTML del libro (y en el CD ROM unido al final de este libro, además de descargable de *http://www.BruceEckel.com*) y también en el propio sitio web, en la página correspondiente a este libro. Si se descubre algo que uno piense que puede ser un error, por favor, utilice el formulario para remitir el error junto con la corrección sugerida. Si es necesario, incluya el archivo de código fuente original y cualquier modificación que se sugiera. Su ayuda será apreciada.

Nota sobre el diseño de la portada

La portada de *Piensa en Java* está inspirada en el *American Arts & Crafts Movement*, que se fundó al cambiar de siglo y alcanzó su cenit entre los años 1900 y 1920. Empezó en Inglaterra como una reacción tanto a la producción de las máquinas de la Revolución Industrial y al estilo victoriano, excesivamente ornamental. *Arts & Crafts* hacía especial énfasis en el mero diseño, en las formas de la naturaleza tal y como se ven en el movimiento del *Art Nouveau*, las manualidades y la importancia del trabajo individual, y sin embargo sin renunciar al uso de herramientas modernas. Hay muchas réplicas con la situación de hoy en día: el cambio de siglo, la evolución de los principios puros de la revolución de los computadores a algo más refinado y más significativo para las personas individuales, y el énfasis en el arte individual que hay en el software, frente a su simple manufactura.

Veo Java de esta misma forma: como un intento de elevar al programador más allá de la mecánica de un sistema operativo y hacia el "arte del software".

Tanto el autor como el diseñador del libro/portada (que han sido amigos desde la infancia) encuentran la inspiración en este movimiento, y ambos poseen muebles, lámparas y otros elementos que o bien son originales, o bien están inspirados en este periodo.

El otro tema de la cubierta sugiere una caja de colecciones que podría usar un naturalista para mostrar los especímenes de insectos que ha guardado. Estos insectos son objetos, ubicados dentro de la caja de objetos. Los objetos caja están a su vez ubicados dentro del "objeto cubierta", que ilustra el concepto fundamental de la agregación en la programación orientada a objetos. Por supuesto, un programador no puede ayudar si no es produciendo "errores" en la asociación, y aquí los errores se han capturado siendo finalmente confinados en una pequeña caja de muestra, como tratando de mostrar la habilidad de Java para encontrar, mostrar y controlar los errores (lo cual es sin duda uno de sus más potentes atributos).

Agradecimientos

En primer lugar, gracias a los asociados que han trabajado conmigo para dar seminarios, proporcionar consultoría y desarrollar productos de aprendizaje: Andrea Provaglio, Dave Bastlett (que también contribuyó significativamente al Capítulo 15), Bill Venners y Larry O'Brien. Aprecio vuestra paciencia a medida que sigo intentando desarrollar el mejor modelo para que tipos tan independientes como nosotros podamos trabajar juntos. Gracias a Rolf André Klaedtke (Suiza); Martin Vleck, Martin Byer, Vlada & Pavel Lahoda, Martin el Oso, y Hanka (Praga); y a Marco Cantu (Italia) por darme alojamiento durante mi primera gira seminario auto organizada por Europa.

Gracias a la *Doyle Street Cohousing Community* por soportarme durante los dos años que me llevó escribir la primera edición de este libro (y por aguantarme en general). Muchas gracias a Kevin y Sonda Donovan por subarrendarme su magnífico lugar en Creste Butte, Colorado, durante el verano mientras trabajaba en la primera edición del libro. Gracias también a los amigables residentes de Crested Butte y al *Rocky Mountain Biologial Laboratory* que me hizo sentir tan acogido.

Gracias a Claudette Moore de la *Moore Literary Agency* por su tremenda paciencia y perseverancia a la hora de lograr que yo hiciera exactamente lo que yo quería hacer.

Mis dos primeros libros se publicaron con Jeff Pepper como editor de Osborne/McGraw-Hill. Jeff apareció en el lugar oportuno y en la hora oportuna en Prentice-Hall y me ha allanado el camino y ha hecho que ocurra todo lo que tenía que ocurrir para que ésta se convirtiera en una experiencia de publicación agradable. Gracias, Jeff —significa mucho para mí.

Estoy especialmente en deuda con Gen Kiyooka y su compañía, Digigami, que me proporcionó gentilmente mi primer servidor web durante los muchos años iniciales de presencia en la Web. Esto constituyó una ayuda de valor incalculable.

Gracias a Cay Hostmann (coautor de *Core Java*, Prentice-Hall, 2000), D'Arcy Smith (Symantec) y Paul Tyma (coautor de *Java Primer Plus*, The Waite Group, 1996), por ayudarme a aclarar conceptos sobre el lenguaje.

Gracias a la gente que ha hablado en mi curso de Java en la *Software Development Conference*, y a los alumnos de mis cursos, que realizan las preguntas que necesito oír para poder hacer un material más claro.

Gracias espaciales a Larry y Tina O'Brien, que me ayudaron a volcar mis seminarios en el CD ROM original *Hands-On Java*. (Puede encontrarse más información en *http://www.BruceEckel.com.*)

Mucha gente me envió correcciones y estoy en deuda con todos ellos, pero envío gracias en particular a (por la primera edición): Kevin Raulerson (encontró cientos de errores enormes), Bob Resendes (simplemente increíble), John Pinto, Joe Dante, Jose Sharp (los tres son fabulosos), David Coms (muchas correcciones gramaticales y aclaraciones), Dr. Robert Stephenson, John Cook, Franklin Chen, Zev Griner, David Karr, Leander A. Stroschein, Steve Clark, Charles A. Lee, Austin Maher, Dennos P. Roth, Roque Oliveira, Douglas Dunn, Dejan Ristic, Neil Galarneau, David B. Malkovsky, Steve Wilkinson, y otros muchos. El profesor Marc Meurrens puso gran cantidad de esfuerzo en publicitar y hacer disponible la versión electrónica de la primera edición del libro en toda Europa.

Ha habido muchísimos técnicos en mi vida que se han convertido en amigos y que también han sido, tanto influyentes, como inusuales por el hecho de que hacen yoga y practican otras formas de ejercicio espiritual, que yo también encuentro muy instructivo e inspirador. Son Karig Borckschmidt, Gen Kiyooka y Andrea Provaglio, (que ayuda en el entendimiento de Java y en la programación general en Italia, y ahora en los Estados Unidos como un asociado del equipo MindView).

No es que me haya sorprendido mucho que entender Delphi me ayudara a entender Java, pues tienen muchas decisiones de diseño del lenguaje en común. Mis amigos de Delphi me proporcionaron ayuda facilitándome a alcanzar profundidad en este entorno de programación tan maravilloso. Son Marco Cantu (otro italiano —¿quizás aprender Latín es una ayuda para entender los lenguajes de programación?), Neil Rubenking (que solía hacer yoga, era vegetariano,... hasta que descubrió los computadores) y por supuesto, Zack Urlocker, un colega de hace tiempo con el que me he movido por todo el mundo.

Las opiniones y el soporte de mi amigo Richard Hale Shaw han sido de mucha ayuda (y la de Kim también). Richard y yo pasamos muchos meses dando seminarios juntos e intentando averiguar cuál era la experiencia de aprendizaje perfecta desde el punto de vista de los asistentes. Gracias a KoAnn Vikoren, Eric Faurot, Marco Pardi, y el resto de equipo y tripulación de MFI. Gracias especialmente a Tara Arrowood, que me volvió a inspirar en las posibilidades de las conferencias.

El diseño del libro, de la portada, y la foto de ésta fueron creadas por mi amigo Daniel Hill-Harris, autor y diseñador de renombre (*http://www.Wil-Harris.com*), que solía jugar con letras de goma en el colegio mientras esperaba a que se inventaran los computadores y los ordenadores personales, y se quejaba de que yo siempre estuviera enmarañado con mis problemas de álgebra. Sin embargo, he producido páginas listas para la cámara por mí mismo, por lo que los errores de tipografía son míos. Para escribir el libro se usó Microsoft ® Word 97 for Windows, y para crear páginas listas para fotografiar en Adobe Acrobat; el libro se creó directamente a partir de los ficheros Acrobat PDF. (Como un tributo a la edad electrónica, estuve fuera en las dos ocasiones en que se produjo la versión final del libro —la primera edición se envío desde Capetown, Sudáfrica, y la segunda edición se

envío desde Praga.) La tipología del cuerpo es *Georgia* y los títulos están en *Verdana*. La tipografía de la portada es *ITC Rennie Mackintosch*.

Gracias a los vendedores que crearon los compiladores. Borland, el Blackdown Group (para Linux), y por supuesto, Sun.

Gracias especiales a todos mis profesores y alumnos (que son a su vez mis profesores). La persona que me enseñó a escribir fue Gabrielle Rico (autora de *Writing the Natural Way*, Putnam, 1985). Siempre guardaré como un tesoro aquella terrorífica semana en Esalen.

El conjunto de amigos que me han ayudado incluyen, sin ser los únicos a: Andrew Binstock, Steve Sinofsky, JD Hildebrandt, Tom Keffer, Brian McElhinney, Brinckely Barr, Hill Gates de *Midnight Engineering Magazine*, Larry Constantine y Lucy Lockwood, Grez Perry, Dan Putterman, Christi Westphal, GeneWang, Dave Mayer, David Intersiomne, Andrea Rosenfield, Claire Sawyers, más italianos (Laura Fallai, Corrado, ILSA, y Cristina Guistozzi). Chris y Laura Strand, los Almquists, Brad Jerbic, Marilyn Cvitanic, los Mabrys, los Haflingers, los Pollocks, Peter Vinci, las familias Robbins, las familias Moelter (y los McMillans), Michael Wilk, Dave Stoner, Laurie Adams, los Cranstons, Larry Fogg, Mike y Karen Sequeiro, Gary Entsminger y Allison Brody, Kevin Donovan y Sonda Eastlack, Chester y Shannon Andersen, Joe Lordy, Dave y Brenda Bartlett, David Lee, los Rentschlers, los Sudeks, Dick, Patty y Lee Eckel, Lynn y Todd y sus familias. Y por supuesto, papá y mamá.

Colaboradores Internet

Gracias a aquellos que me han ayudado a reescribir los ejemplos para usar la biblioteca Swing, y por cualquier otra ayuda: Jon Shvarts, Thomas Kirsch, Rahim Adatia, Rajes Jain, Ravi Manthena, Banu Rajamani, Jens Brandt, Mitin Shivaram, Malcolm Davis y todo el mundo que mostró su apoyo. Verdaderamente, esto me ayudó a dar el primer salto.