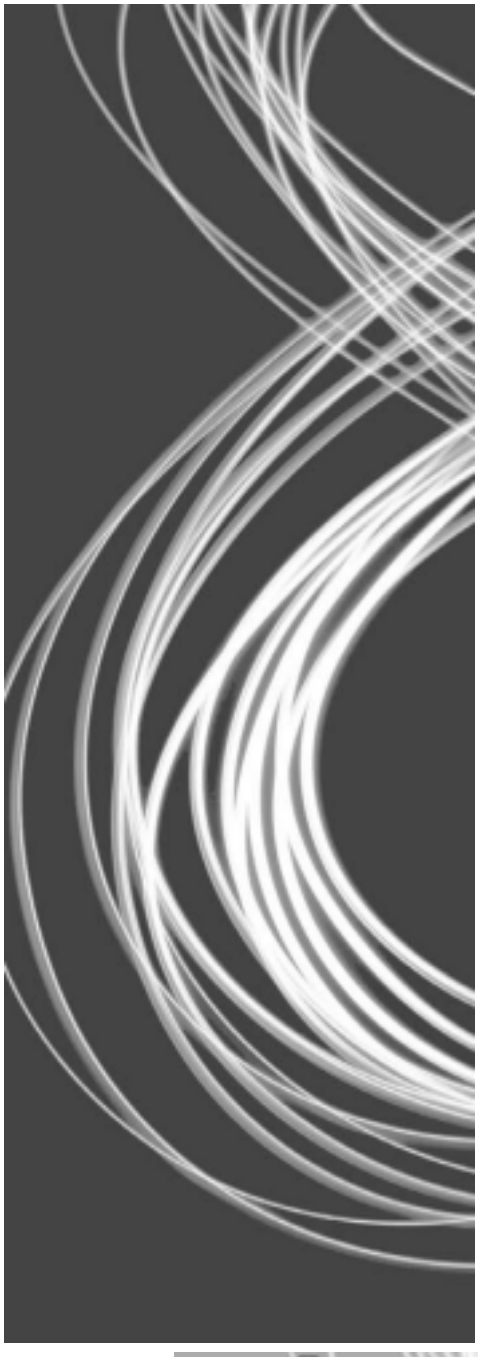
Java



CÓMO PROGRAMAR

CÓMO PR
CÓMO PR

Deitel & Associates, Inc.

Deitel & Associates, Inc.

Ingeniero en Sistemas Electrónicos
Instituto Tecnológico y de Estudios Superiores de Monterrey - Campus Monterrey

Sergio Fuenlabrada Velázquez

Judith Sonck Ledezma

José Luis López Goytia
Departamento de Sistemas
Unidad Profesional Interdisciplinaria de Ingeniería y Ciencias Sociales y Administrativas, Instituto Politécnico Nacional, México

Datos de catalogación bibliográfica

DEITEL, PAUL y DEITEL, HARVEY

Cómo programar en Java
Décima edición

PEARSON EDUCACIÓN, México, 2016

ISBN: 978-607-32-3802-1
Área: Computación

Formato: 20 × 25.5 cm Páginas: 560

Authorized translation from the English language edition, entitled *Java How to program 10th* Edition, by *Paul Deitel and Harvey Deitel,* published by Pearson Education, Inc., publishing as Prentice Hall, Copyright © 2016. All rights reserved. ISBN 9780133807806

Edición en español
Director General: Sergio Fonseca Garza
Director de innovación y servicios educativos: Alan David Palau
Gerencia de contenidos y servicios editoriales: Jorge Luis Íñiguez Caso
Coordinador de Contenidos, Educación Superior: Guillermo Domínguez Chávez
guillermo.dominguez@pearson.com
Especialista en Contenidos de Aprendizaje: Luis Miguel Cruz Castillo
Especialista en Desarrollo de Contenidos: Bernardino Gutiérrez Hernández
Supervisor de Arte y Diseño: José Dolores Hernández Garduño

DÉCIMA EDICIÓN, 2016

ISBN VERSIÓN IMPRESA: 978-607-32-3802-1
ISBN VERSIÓN E-BOOK: 978-607-32-3803-8

Impreso en México. *Printed in Mexico.*
1 2 3 4 5 6 7 8 9 0 - 19 18 17 16

www.pearsonenespañol.com

Para Brian Goetz,
Arquitecto del lenguaje Java de Oracle y jefe de especificaciones para el proyecto Lambda de Java SE 8:

Su tutoría nos ayudó a hacer un mejor libro. Gracias por insistir en que lo hiciéramos bien.

Paul y Harvey Deitel

LOS CAPÍTULOS 12 A 19 SE ENCUENTRAN DISPONIBLES, EN ESPAÑOL, EN EL SITIO WEB DE ESTE LIBRO

Los capítulos 20 a 34 y los apéndices F a N se encuentran disponibles, en idioma inglés, en el sitio web de este libro

He estado enamorado de Java incluso antes de que se publicara su versión 1.0 en 1995. Desde entonces he sido desarrollador de Java, autor, ponente, maestro y embajador de tecnología Java (Oracle Java Technology Ambassador). En esta aventura he tenido el privilegio de llamar colega a Paul Deitel, además de aprovechar y recomendar con frecuencia su libro *Cómo programar en Java*. En sus muchas ediciones, este libro ha demostrado ser un excelente texto para cursos universitarios y profesionales que otros y yo hemos desarrollado para enseñar el lenguaje de programación Java.

Una de las cualidades que hacen de este libro un excelente recurso es su cobertura detallada y perspicaz de los conceptos de Java, incluyendo los que se introdujeron recientemente en Java SE 8. Otra cualidad útil es su tratamiento de los conceptos y las prácticas esenciales para un desarrollo de software efectivo.

Como admirador incondicional de este libro, me gustaría señalar algunas de las características que

más me han impresionado de esta décima edición:

- Un nuevo y ambicioso capítulo sobre las expresiones lambda y los fl ujos en Java. Este capítulo comienza con una introducción básica sobre la programación funcional, donde presenta las expresiones lambda de Java y cómo usar los fl ujos para realizar tareas de programación funcio nales en las colecciones.
- Aunque desde la primera edición de este libro se ha lidiado con la concurrencia, es cada vez más importante debido a las arquitecturas multinúcleo. En el capítulo de concurrencia hay ejemplos de sincronización (donde se usan las nuevas clases de la API de fecha/hora que se introdujo en Java SE 8) que muestran las mejoras de rendimiento de la tecnología multinúcleo en comparación con el uso de un solo núcleo.
- JavaFX es la tecnología de GUI/gráfi cos/multimedia de Java en progreso, por lo que es agradable ver un tratamiento de tres capítulos de JavaFX en el estilo pedagógico de código activo de Deitel. Uno de estos capítulos es parte del libro impreso y los otros dos están en el sitio Web.

Les pido que se unan conmigo para felicitar a Paul y Harvey Deitel por su más reciente edición de un maravilloso recurso, tanto para los estudiantes de ciencias computacionales como para los desarrolladores de software.

James L. Weaver
Java Technology Ambassador
Oracle Corporation

"El principal mérito del lenguaje es la claridad…"
—Galen

Bienvenido al lenguaje de programación Java y al libro *Cómo programar en Java*, *décima edición*. Estelibro presenta las tecnologías computacionales de vanguardia para estudiantes, instructores y desarrolladores de software. Es apropiado para cursos académicos introductorios y para cursos profesionales basados en las recomendaciones curriculares de ACM/IEEE, y sirve como preparación para el examen de Colocación avanzada (AP) de Ciencias computacionales.

Nos enfocamos en las mejores prácticas de ingeniería de software. La base del libro es nuestro reconocido "método de código activo", en el cual los conceptos se presentan en el contexto de programas funcionales completos que se ejecutan en las versiones recientes de Windows®, OS X® y Linux®, en vez de hacerlo a través de fragmentos separados de código. Cada ejemplo de código completo viene acompañado de ejemplos de ejecuciones reales.

Cómo contactar a los autores

Si surge alguna duda o pregunta a medida que lea este libro, envíe un correo electrónico a

deitel@deitel.com

y le responderemos a la brevedad. Para obtener actualizaciones sobre este libro, visite

http://www.deitel.com/books/jhtp10

también puede suscribirse al boletín de correo electrónico *Deitel® Buzz Online*

en http://www.deitel.com/newsletter/subscribe.html

y puede unirse a las comunidades de redes sociales de Deitel en

- Facebook® (http://www.deitel.com/deitelfan)
- Twitter® (@deitel)
- Google+™ (http://google.com/+DeitelFan)
- YouTube® (http://youtube.com/DeitelTV)
- LinkedIn® (http://linkedin.com/company/deitel-&-associates)

Código fuente

Encontrará todo el código fuente utilizado en esta edición en español en:

www.pearsonenespañol.com/deitel

Notas de video (VideoNotes)

Encontrará los videos indicados en el libro (en inglés) en:

http://www.deitel.com/books/jhtp10

lo invitamos a que visite el sitio Web del libro en:

http://www.pearsonenespañol.com/deitel

Organización modular

Cómo programar en Java, *10 edición* es apropiado para cursos de programación en diversos niveles, en especial los de ciencias computacionales CS 1 y CS 2, además de aquellos introductorios en disciplinas relacionadas. La estructura modular del libro ayuda a los instructores a organizar sus planes de estudios:

Introducción

- Capítulo 1, Introducción a las computadoras, Internet y Java
- Capítulo 2, Introducción a las aplicaciones en Java: entrada/salida y operadores
- Capítulo 3, Introducción a las clases, los objetos, los métodos y las cadenas

Fundamentos adicionales de programación

- Capítulo 4, Instrucciones de control: parte 1: operadores de asignación, ++ y --
- Capítulo 5, Instrucciones de control: parte 2: operadores lógicos
- Capítulo 6, Métodos: un análisis más detallado
- Capítulo 7, Arreglos y objetos ArrayList
- Capítulo 14 (en línea), Cadenas, caracteres y expresiones regulares
- Capítulo 15 (en línea), Archivos, flujos y serialización de objetos

Programación orientada a objetos y diseño orientado a objetos

- Capítulo 8, Clases y objetos: un análisis más detallado
- Capítulo 9, Programación orientada a objetos: herencia

- Capítulo 10, Programación orientada a objetos: polimorfismo e interfaces
- Capítulo 11, Manejo de excepciones: un análisis más detallado
- Capítulo 33 (en línea), Ejemplo práctico opcional del cajero automático (ATM), parte 1: Diseño orientado a objetos con el UML (en inglés)
- Capítulo 34 (en línea), Ejemplo práctico opcional del cajero automático (ATM), parte 2: Implementación de un diseño orientado a objetos (en inglés)

Interfaces gráficas de usuario de Swing y gráficos de Java 2D

- Capítulo 12 (en línea), Componentes de la GUI: parte 1
- Capítulo 13 (en línea), Gráficos y Java 2D
- Capítulo 22 (en línea), Componentes de GUI: parte 2 (en inglés)

Estructuras de datos, colecciones, lambdas y flujos

- Capítulo 16 (en línea), Colecciones de genéricos
- Capítulo 17 (en línea), Lambdas y flujos de Java SE 8
- Capítulo 18 (en línea), Recursividad
- Capítulo 19 (en línea), Búsqueda, ordenamiento y Big O
- Capítulo 20 (en línea), Clases y métodos genéricos (en inglés)
- Capítulo 21 (en línea), Estructuras de datos genéricas personalizadas (en inglés)

Concurrencia: redes

- Capítulo 23 (en línea), Concurrencia (en inglés)
- Capítulo 28 (en línea), Redes (en inglés)

Interfaces gráficas de usuario, gráficos y multimedia de JavaFX

- Capítulo 25 (en línea), GUI JavaFX: parte 1
- Capítulo 26 (en línea), GUI JavaFX: parte 2
- Capítulo 27 (en línea), Gráficos y multimedia de JavaFX

Desarrollo Web y de escritorio orientado a bases de datos

- Capítulo 24 (en línea), Acceso a bases de datos con JDBC
- Capítulo 29 (en línea), API de persistencia de Java (JPA)
- Capítulo 30 (en línea), Aplicaciones Web de JavaServer™ Faces: parte 1
- Capítulo 31 (en línea), Aplicaciones Web de JavaServer™ Faces: parte 2
- Capítulo 32 (en línea), Servicios Web basados en REST

Características nuevas y mejoradas del libro

He aquí las actualizaciones que realizamos a *Cómo programar en Java, décima edición*:

Java Standard Edition: Java SE 7 y el nuevo Java SE 8

- ***Fácil de usar con Java SE 7 o Java SE 8.*** Para satisfacer las necesidades de nuestros usuarios, diseñamos el libro para cursos tanto universitarios como profesionales que tengan como base Java SE 7, Java SE 8 o una mezcla de ambos. Las características de Java SE 8 se cubren en secciones opcionales que pueden incluirse u omitirse con facilidad. Las nuevas herramientas de Java SE 8 pueden mejorar notablemente el proceso de programación. La figura 1 muestra algunas de las nuevas características que cubrimos de Java SE 8.

Expresiones lambda

Mejoras en la inferencia de tipos

Anotación `@FunctionalInterface`

Ordenamiento de arreglos en paralelo

Operaciones de datos a granel para colecciones de Java: `filter`, `map` y `reduce`

Mejoras en la biblioteca para el soporte de lambdas (por ejemplo, `java.util.stream`, `java.util.function`) API de fecha y hora (`java.time`)

Mejoras en la API de concurrencia de Java

Métodos `static` y `default` en interfaces

Interfaces funcionales: interfaces que definen sólo un método `abstract` y pueden incluir métodos `static` y `default` Mejoras en JavaFX

Fig. 1 Algunas características nuevas de Java SE 8.

- ***Lambdas, flujos e interfaces de Java SE 8 con métodos `default` y `static`.*** Las características nuevas más importantes en Java SE 8 son las lambdas y las tecnologías complementarias, que cubrimos con detalle en el capítulo 17 opcional y en las secciones opcionales identificadas como "Java SE 8" en capítulos posteriores. En el capítulo 17 (en línea) verá que la programación fun cional con lambdas y flujos puede ayudarle a escribir programas de manera más rápida, concisa y simple, con menos errores y que sean más fáciles de paralelizar (para obtener mejoras de rendi miento en sistemas multinúcleo) que los programas escritos con las técnicas anteriores. Descubri rá que la programación funcional complementa a la programación orientada a objetos. Después de que lea el capítulo 17, podrá reimplementar de manera inteligente muchos de los ejemplos de Java SE 7 del libro (figura 2).

Capítulo 7, Arreglos y objetos `ArrayList`	Las secciones 17.3 a 17.4 presentan herramientas básicas de lambdas y fl ujos que procesan arreglos unidimensionales.
Capítulo 10, Programación orientada a objetos: polimorfi smo e interfaces	La sección 10.10 presenta las nuevas características de interfaces de Java SE 8 (métodos `default`, métodos `static` y el concepto de interfaces funcionales) que dan soporte a la programación funcional con lambdas y fl ujos.
Capítulos 12 y 22, Componentes de GUI: parte 1 y parte 2, respectivamente	La sección 17.9 muestra cómo usar una expresión lambda para implementar una interfaz funcional de componente de escucha de eventos de Swing.
Capítulo 14, Cadenas, caracteres y expresiones regulares	La sección 17.5 muestra cómo usar lambdas y fl ujos para procesar colecciones de objetos `String`.
Capítulo 15, Archivos, fl ujos y serialización de objetos	La sección 17.7 muestra cómo usar lambdas y fl ujos para procesar líneas de texto de un archivo.
Capítulo 23, Concurrencia	Muestra que los programas funcionales son más fáciles de paralelizar para que puedan aprovechar las arquitecturas multinúcleo y mejorar el rendimiento. Demuestra el procesamiento de fl ujos en paralelo. Muestra que el método `parallelSort` de `Arrays` mejora el rendimiento en arquitecturas

multinúcleo al ordenar arreglos de gran tamaño.

Capítulo 25, GUI JavaFX: parte 1 La sección 25.5.5 muestra cómo usar una expresión lambda para implementar una interfaz funcional para un componente de escucha de eventos de JavaFX.

Fig. 2 Explicaciones y ejemplos de lambdas y flujos de Java SE 8.

- ***La instrucción try con recursos y la interfaz AutoClosable de Java SE 7.*** Los objetos AutoClosable reducen la probabilidad de que haya fugas de recursos si los usa con la instrucción try con recursos, la cual cierra de manera automática los objetos AutoClosable. En esta edición usamos try con recursos y objetos AutoClosable según sea apropiado a partir del capítulo 15, Archivos, flujos y serialización de objetos.
- ***Seguridad de Java.*** Auditamos nuestro libro conforme al estándar CERT de codificación segura de Oracle para Java según lo apropiado para un libro de texto introductorio. Consulte la sección Programación segura en Java de este prefacio para obtener más información sobre CERT.
- ***API NIO de Java.*** Actualizamos los ejemplos de procesamiento de archivos en el capítulo 15 para usar las características de la API NIO (nueva IO) de Java.

- ***Documentación de Java.*** A lo largo del libro le proporcionamos vínculos hacia la documenta ción de Java, donde aprenderá más sobre los diversos temas que presentamos. Para la documen tación de Java SE 7, los vínculos empiezan con

  http://docs.oracle.com/javase/7/

  y para la documentación de Java SE 8, los vínculos comienzan con

  http://download.java.net/jdk8/

  Estos vínculos pueden cambiar cuando Oracle libere Java SE 8; *es posible* que los vínculos comien cen con

  http://docs.oracle.com/javase/8/

  Si hay vínculos que cambien después de publicar el libro, las actualizaciones estarán disponibles en

  http://www.deitel.com/books/jhtp10

GUI Swing y JavaFX, gráficos y multimedia

- ***GUI Swing y gráficos de Java 2D.*** Hablaremos sobre la GUI Swing de Java en las secciones op cionales de GUI y gráficos de los capítulos 3 al 10, y en los capítulos 12 y 22. Ahora Swing se encuentra en modo de mantenimiento: Oracle detuvo el desarrollo y proporcionará sólo correc ción de errores de aquí en adelante, pero seguirá siendo parte de Java y aún se utiliza ampliamente. El capítulo 13 habla de los gráficos de Java 2D.
- ***GUI JavaFX, gráficos y multimedia.*** La API de GUI, gráficos y multimedia de Java que conti nuará progresando es JavaFX. En el capítulo 25 usamos JavaFX 2.2 (que se liberó en 2012) con Java SE 7. Nuestros capítulos 26 y 27 en línea, que se encuentran en el sitio Web complementario del libro (vea la portada interior del libro), presentan características adicionales de la GUI JavaFX e introducen los gráficos y multimedia de JavaFX en el contexto de Java FX 8 y Java SE 8. En los capítulos 25 a 27 usamos Scene Builder, una herramienta de arrastrar y soltar para crear interfaces GUI de JavaFX en forma rápida y conveniente. Es una herramienta independiente que puede usar por separado o con cualquiera de los IDE de Java.
- ***Presentación escalable de GUI y gráficos.*** Los profesores que dictan cursos introductorios tienen

una gran variedad de opciones en cuanto a la cantidad por cubrir de GUI, gráficos y multimedia: desde nada en lo absoluto, pasando por las secciones de introducción opcionales en los primeros capítulos, hasta un análisis detallado de la GUI Swing y los gráficos de Java 2D en los capítulos 12, 13 y 22, o incluso un análisis detallado de la GUI Java FX, gráficos y multimedia en el capí tulo 25 y los capítulos en línea 26 y 27.

Concurrencia

- ***Concurrencia para un óptimo rendimiento con multinúcleo.*** En esta edición tenemos el pri vilegio de tener como revisor a Brian Goetz, coautor de *Java Concurrency in Practice* (Addison Wesley). Actualizamos el capítulo 23 (en línea) con la tecnología y el idioma de Java SE 8. Agre gamos un ejemplo de comparación entre `parallelSort` y `sort` que usa la API de hora/fecha de Java SE 8 para sincronizar cada operación y demostrar que `parallelSort` tiene un mejor rendi miento en un sistema multinúcleo. Incluimos un ejemplo de procesamiento de comparación entre flujos paralelos y secuenciales de Java SE 8, usando de nuevo la API de hora/fecha para mostrar las mejoras de rendimiento. Por último, agregamos un ejemplo de `CompletableFuture` de Java SE 8 que demuestra la ejecución secuencial y paralela de cálculos extensos.

• ***Clase SwingWorker.*** Usamos la clase `SwingWorker` para crear interfaces de usuario multihilo. En el capítulo 26 (en línea), mostramos cómo es que JavaFX maneja la concurrencia. • ***La concurrencia es desafiante.*** La programación de aplicaciones concurrentes es difícil y pro pensa a errores. Hay una gran variedad de características de concurrencia. Señalamos las que de berían usarse más y mencionamos las que deben dejarse a los expertos.

Cómo calcular bien las cantidades monetarias

- ***Cantidades monetarias.*** En los primeros capítulos usamos por conveniencia el tipo `double` para representar cantidades monetarias. Debido a la posibilidad de realizar cálculos monetarios inco rrectos con el tipo `double`, hay que usar la clase `BigDecimal` (que es un poco más compleja) para representar las cantidades monetarias. Demostramos el uso de `BigDecimal` en los capítulos 8 y 25.

Tecnología de objetos

• ***Programación y diseño orientados a objetos.*** Usamos la metodología de tratar los *objetos en los primeros capítulos*, donde presentamos la terminología y los conceptos básicos de la tecnología de objetos en el capítulo 1. Los estudiantes desarrollan sus primeras clases y objetos personalizados en el capítulo 3. Al presentar los objetos y las clases en los primeros capítulos, hacemos que los estudiantes "piensen en objetos" de inmediato y que dominen estos conceptos con más profundi dad [para los cursos que requieren una metodología en la que se presenten los objetos en capítulos posteriores, le recomendamos el libro *Java How to Program, Late Objects Version, 10ª edición*].

- ***Ejemplos prácticos reales sobre el uso de objetos en los primeros capítulos.*** La presentación de clases y objetos en los primeros capítulos incluye ejemplos prácticos con las clases `Cuenta`, `Estu diante`, `AutoPolicy`, `Tiempo`, `Empleado`, `LibroCalificaciones` y un ejemplo práctico sobre barajar y repartir cartas con la clase `Carta`, introduciendo de manera gradual los conceptos de OO más complejos.
- ***Herencia, interfaces, polimorfismo y composición.*** Usamos una serie de ejemplos prácticos reales para ilustrar cada uno de estos conceptos de OO y explicamos situaciones en las que sea conveniente usar cada uno de ellos para crear aplicaciones industriales.
- ***Manejo de excepciones.*** Integramos el manejo básico de excepciones en los primeros capítulos del libro y luego presentamos un análisis más detallado en el capítulo 11. El manejo de excep ciones es importante para crear aplicaciones de "misión crítica" e "imprescindibles para los negocios". Los programadores necesitan lidiar con las siguientes dudas: "¿Qué ocurre cuando el componente que invoco para realizar un trabajo experimenta dificultades? ¿Cómo indicará ese componente que tuvo un problema?". Para usar un componente de Java no sólo hay que saber cómo se comporta ese componente cuando "las cosas salen bien", sino también las

excep ciones que ese componente "lanza" cuando "las cosas salen mal".

- ***Las clases Arrays y ArrayList.*** El capítulo 7 cubre la clase Arrays (que contiene métodos para realizar manipulaciones comunes de arreglos) y la clase ArrayList (que implementa una estruc tura de datos tipo arreglo, cuyo tamaño se puede ajustar en forma dinámica). Esto va de acuerdo con nuestra filosofía de obtener mucha práctica al utilizar las clases existentes, al tiempo que el estudiante aprende a definir sus propias clases. La extensa selección de ejercicios del capítulo in cluye un proyecto importante sobre cómo crear su propia computadora mediante la técnica de simulación por software. El capítulo 21 (en línea) incluye un proyecto de seguimiento sobre la creación de su propio compilador que puede compilar programas en lenguaje de alto nivel a có digo de lenguaje máquina, y que se ejecutará en su simulador de computadora.
- ***Ejemplo práctico opcional: desarrollo de un diseño orientado a objetos y una implementación en Java de un cajero automático (ATM).*** Los capítulos 33 y 34 (en línea) incluyen un ejemplo práctico *opcional* sobre el diseño orientado a objetos mediante el uso del UML (Lenguaje Unifi cado de Modelado$_{\textbf{TM}}$): el lenguaje gráfico estándar en la industria para modelar sistemas orientados a objetos. Diseñamos e implementamos el software para un cajero automático (ATM) simple.

Analizamos un documento de requerimientos típico, el cual especifica el sistema que se va a cons truir. Determinamos las clases necesarias para implementar ese sistema, los atributos que deben tener esas clases, los comportamientos que necesitan exhibir y especificamos cómo deben interac tuar las clases entre sí para cumplir con los requerimientos del sistema. A partir del diseño, pro ducimos una implementación completa en Java. A menudo los estudiantes informan que pasan por un «momento de revelación»: el ejemplo práctico les ayuda a «atar cabos» y comprender en verdad la orientación a objetos.

Estructuras de datos y colecciones genéricas

- ***Presentación de estructuras de datos.*** Empezamos con la clase genérica ArrayList en el capítulo 7. Nuestros análisis posteriores sobre las estructuras de datos (capítulos 16 al 21) ofrecen un trata miento más detallado de las colecciones de genéricos, ya que enseñan a utilizar las colecciones integradas de la API de Java. Hablamos sobre la recursividad, que es importante para implemen tar las clases de estructuras de datos tipo árbol. Hablamos sobre los algoritmos populares de bús queda y ordenamiento para manipular el contenido de las colecciones y proporcionamos una introducción amigable a Big O: un medio para describir qué tan duro tendría que trabajar un algoritmo para resolver un problema. Luego mostramos cómo implementar los métodos y las clases genéricas, además de las estructuras de datos genéricas *personalizadas* (esto es un tema para las materias de especialidad en ciencias computacionales; la mayoría de los programadores deben usar las colecciones de genéricos prefabricadas). Las lambdas y los flujos (que se presentan en el capítulo 17) son especialmente útiles para trabajar con colecciones de genéricos.

Base de datos

- ***JDBC.*** El capítulo 24 (en línea) trata sobre JDBC y usa el sistema de administración de bases de datos Java DB. El capítulo presenta el Lenguaje estructurado de consulta (SQL) y un ejemplo práctico de OO sobre cómo desarrollar una libreta de direcciones controlada por una base de datos en donde se demuestran las instrucciones preparadas.
- ***API de persistencia de Java.*** El nuevo capítulo 29 (en línea) cubre la API de persistencia de Java (JPA): un estándar para el mapeo objeto-relacional (ORM) que usa JDBC "tras bambalinas". Las herramientas de ORM pueden analizar el esquema de una base de datos y generar un

conjunto de clases que nos permitan interactuar con esta base de datos sin tener que usar JDBC y SQL de manera directa. Esto agiliza el desarrollo de las aplicaciones de bases de datos, reduce los errores y produce código más portable.

Desarrollo de aplicaciones Web

- ***Java Server Faces (JSF).*** Los capítulos 30 y 31 (en línea) se actualizaron para introducir la tecnología JavaServer Faces (JSF) más reciente, que simplifica en gran medida la creación de aplicaciones Web con JSF. El capítulo 30 incluye ejemplos sobre la creación de interfaces GUI para aplicaciones Web, la validación de formularios y el rastreo de sesiones. El capítulo 31 habla sobre las aplicaciones JSF controladas por datos y habilitadas para Ajax. El capítulo incluye una libreta de direcciones Web multinivel controlada por una base de datos, la cual permite a los usuarios agregar y buscar contactos.
- ***Servicios Web.*** El capítulo 32 (en línea) se concentra ahora en la creación y el consumo de servicios Web basados en REST. Ahora la gran mayoría de los servicios Web de la actualidad usan REST.

Programación segura en Java

Es difícil crear sistemas industriales que resistan a los ataques de virus, gusanos y otras formas de "malware". En la actualidad, debido al uso de Internet, esos ataques pueden ser instantáneos y de un alcance global. Al integrar la seguridad en el software desde el principio del ciclo del desarrollo es posible reducir, en gran

medida, las vulnerabilidades. En nuestros análisis y ejemplos de código incorporamos varias prácticas de codificación seguras en Java (en la medida apropiada para un libro de texto introductorio). El Centro de coordinación del CERT® (www.cert.org) se creó para analizar y responder de manera oportuna a los ataques. El CERT (Equipo de respuesta ante emergencias informáticas) es una organización financiada por el gobierno de los Estados Unidos dentro de Carnegie Mellon University Software Engineering Institute™. CERT publica y promueve estándares de codificación segura para diversos lenguajes de programación populares con el fin de ayudar a los desarrolladores de software a implementar sistemas industriales que eviten las prácticas de programación que dejan los sistemas abiertos a los ataques.

Nos gustaría agradecer a Robert C. Seacord, gerente de codificación segura en CERT y profesor adjunto en Carnegie Mellon University School of Computer Science. El Sr. Seacord fue revisor técnico de nuestro libro *Cómo programar en C, séptima edición,* en donde escudriñó nuestros programas en C desde el punto de vista de la seguridad y nos recomendó adherirnos al *estándar de codificación segura en C del CERT*. Esta experiencia también influyó en nuestras prácticas de codificación en *Cómo programar en C++, novena edición,* y en *Cómo programar en Java, décima edición*.

Ejemplo práctico opcional de GUI y gráficos

A los estudiantes les gusta crear aplicaciones de GUI y gráficos. Para los cursos en los que se presentan los temas de GUI y gráficos en las primeras clases, integramos una introducción opcional de 10 segmentos para crear gráficos e interfaces gráficas de usuario (GUI) basadas en Swing. El objetivo de este ejemplo práctico es crear una aplicación polimórfica simple de dibujo en la que el usuario pueda elegir una forma para dibujar, seleccionar las características de esa forma (como su color) y usar el ratón para dibujarla. El ejemplo práctico se desarrolla en forma gradual para llegar a ese objetivo, en donde el lector implementa el dibujo polimórfico en el capítulo 10, agrega una GUI orientada a eventos en el capítulo 12 y mejora las capacidades de dibujo en el capítulo 13 con Java 2D.

- Sección 3.6: Uso de cuadros de diálogo

- Sección 4.15: Creación de dibujos simples
- Sección 5.11: Dibujo de rectángulos y óvalos
- Sección 6.13: Colores y figuras rellenas
- Sección 7.17: Cómo dibujar arcos
- Sección 8.16: Uso de objetos con gráficos
- Sección 9.7: Mostrar texto e imágenes usando etiquetas
- Sección 10.11: Realización de dibujos mediante el polimorfismo
- Ejercicio 12.17: Expansión de la interfaz
- Ejercicio 13.31: Incorporación de Java2D

Métodos de enseñanza

Cómo programar en Java, 10ª edición contiene cientos de ejemplos funcionales completos. Hacemos hincapié en la claridad de los programas y nos concentramos en crear software bien diseñado.

Notas de video (VideoNotes). El sitio Web complementario incluye muchas notas en inglés de video en las que el coautor Paul Deitel explica con detalle la mayoría de los programas de los capítulos básicos del libro. A los estudiantes les gusta ver las notas de video para reforzar los conceptos básicos y para obtener información adicional.

Resaltado de código. Colocamos rectángulos de color gris alrededor de los segmentos de código clave.

Uso de fuentes para dar énfasis. Para facilitar su identificación, ponemos en **negritas** los términos clave y la referencia de la página del índice para cada definición. Enfatizamos los componentes en pantalla en la

fuente Helvetica en negritas (por ejemplo, el menú Archivo) y enfatizamos el texto del programa en la fuen te Lucida (por ejemplo, `int x = 5;`).

Acceso Web. Todos los ejemplos de código fuente se pueden descargar de:

http://www.deitel.com/books/jhtp10
http://www.pearsonenespañol.com/deitel

Objetivos. Las citas de apertura van seguidas de una lista de objetivos del capítulo.

Ilustraciones y figuras. Incluimos una gran cantidad de tablas, dibujos lineales, diagramas de UML, programas y salidas de programa.

Tips de programación. Incluimos tips de programación para ayudarle a enfocarse en los aspectos importantes del desarrollo de programas. Estos tips y prácticas representan lo mejor que hemos podido recabar a lo largo de siete décadas combinadas de experiencia en la programación y la enseñanza.

Buena práctica de programación

Las buenas prácticas de programación *llaman la atención hacia técnicas que le ayudarán a producir programas más claros, comprensibles y fáciles de mantener.*

Error común de programación

Al poner atención en estos Errores comunes de programación *se reduce la probabilidad de que usted pueda cometer los mismos errores.*

Tip para prevenir errores

Estos cuadros contienen sugerencias para exponer los errores y eliminarlos de sus programas; muchos de ellos describen aspectos de Java que evitan que los errores entren siquiera a los programas.

Tip de rendimiento

Estos cuadros resaltan las oportunidades para hacer que sus programas se ejecuten más rápido, o para minimizar la cantidad de memoria que ocupan.

Tip de portabilidad

Los tips de portabilidad *le ayudan a escribir código que pueda ejecutarse en una variedad de plataformas.*

Observación de ingeniería de software

Las observaciones de ingeniería de software *resaltan los aspectos de arquitectura y diseño, lo cual afecta la construcción de los sistemas de software, especialmente los de gran escala.*

Observación de apariencia visual

Las observaciones de apariencia visual *resaltan las convenciones de la interfaz gráfica de usuario. Estas observaciones le ayudan a diseñar interfaces gráficas de usuario atractivas y amigables para el usuario, en conformidad con las normas de la industria.*

Viñetas de resumen. Presentamos un resumen detallado de cada sección del capítulo en un estilo de lista con viñetas. Para facilitar la referencia, incluimos el número de página de cada defi nición de los términos clave del libro.

Ejercicios de autoevaluación y respuestas. Se incluyen diversos ejercicios de autoevaluación *y sus* respuestas, para que los estudiantes practiquen por su cuenta. Todos los ejercicios en el ejemplo práctico opcional sobre el ATM están totalmente resueltos.

Ejercicios. Los ejercicios de los capítulos incluyen:

- recordatorio simple de la terminología y los conceptos importantes
- ¿cuál es el error en este código?
- ¿qué hace este código?
- escritura de instrucciones individuales y pequeñas porciones de métodos y clases
- escritura de métodos, clases y programas completos
- proyectos importantes
- en muchos capítulos, hay ejercicios "Marcando la diferencia" que animan al lector a usar las compu tadoras e Internet para investigar y resolver problemas sociales importantes.

Los ejercicios que son únicamente para la versión SE 8 de Java se identifi can como tales. Consulte nuestro Centro de recursos de proyectos de programación, donde encontrará muchos ejercicios adicionales y posibilidades de proyectos (www.deitel.com/ProgrammingProjects/).

Índice. Incluimos un índice extenso. Las entradas que corresponden a las definiciones de los términos clave están resaltadas **en negritas** junto con su número de página. El índice del libro incluye todos los tér minos del libro impreso y de los capítulos en español disponibles en el sitio web; es decir, incluye los términos de los capítulos 1 a 19.

Software utilizado en *Cómo programar en Java 10ª edición*

Podrá descargar a través de Internet, y sin costo, todo el software necesario para este libro . En la sección Antes de empezar que se encuentra después de este Prefacio, encontrará los vínculos para cada

descarga. Para escribir la mayoría de los ejemplos de este libro utilizamos el kit de desarrollo gratuito Java Standard Edition Development Kit (JDK) 7. Para los módulos opcionales de Java SE 8, utilizamos la versión JDK 8 de prueba de OpenJDK. En el capítulo 25 (en línea) y en varios capítulos también utilizamos el IDE Netbeans. Encontrará recursos y descargas de software adicionales en nuestros Centros de recursos de Java, ubicados en:

www.deitel.com/ResourceCenters.html

Suplementos para el profesor (en inglés)

Los siguientes suplementos están disponibles sólo para profesores registrados en el Centro de recursos para el profesor de Pearson (www.pearsonenespañol.com/deitel):

- ***Diapositivas de PowerPoint®*** que contienen todo el código y las figuras del texto, además de elementos en viñetas que sintetizan los puntos clave.
- ***Banco de exámenes*** con preguntas de opción múltiple (aproximadamente dos por cada sección del libro).
- ***Manual de soluciones*** con soluciones para la gran mayoría de los ejercicios que aparecen al final de capítulo. **Antes de asignar un ejercicio de tarea, los profesores deberán consultar el IRC para asegurarse de que incluya la solución.**

Por favor, no escriba a los autores para solicitar acceso al Centro de recursos para el profesor de Pearson. El acceso está limitado estrictamente a profesores que utilicen este libro como texto en sus cursos y estén registrados en nuestro sistema. Los profesores sólo pueden obtener acceso a través de los representantes de Pearson. No se proveen soluciones para los ejercicios de «proyectos».

Si no es un profesor registrado, póngase en contacto con su representante de Pearson o visite www.pearsonhighered.com/educator/replocator/.

Contenido de este libro

Este libro contiene los primeros 11 capítulos mencionados (impresos); adicionalmente, dentro de la página Web www.pearsonenespañol.com/deitel, encontrará, en español, los capítulos 12 a 19; y en inglés los capítulos 20 a 34, así como los apéndices F a N.

Reconocimientos

Queremos agradecer a Abbey Deitel y a Barbara Deitel por las largas horas que dedicaron a este proyecto. Somos afortunados al haber trabajado en este proyecto con un dedicado equipo de editores profesionales de Pearson. Apreciamos la orientación, inteligencia y energía de Tracy Johnson, editora en jefe de ciencias computacionales. Tracy y su equipo se encargan de todos nuestros libros de texto académicos. Carole Snyder reclutó a los revisores técnicos del libro y se hizo cargo del proceso de revisión. Bob Engelhardt se hizo cargo de la publicación del libro. Nosotros seleccionamos la imagen de portada y Laura Gardner diseñó la portada.

Revisores

Queremos agradecer los esfuerzos de nuestros revisores más recientes: un grupo distinguido de profesores, miembros del equipo de Oracle Java, Oracle Java Champions y otros profesionales de la industria. Ellos examinaron minuciosamente el texto y los programas, proporcionando innumerables sugerencias para mejorar la presentación.

Apreciamos la asesoría tanto de Jim Weaver y de Johan Vos (coautores de *Pro JavaFX 2*), así como de Simon Ritter en los tres capítulos sobre JavaFX.

Revisores de la décima edición: Lance Andersen (Oracle Corporation), Dr. Danny Coward (Oracle Corporation), Brian Goetz (Oracle Corporation), Evan Golub (University of Maryland), Dr. Huiwei Guan (profesor del departamento de ciencias computacionales y de la información en North Shore

Community College), Manfred Riem (Java Champion), Simon Ritter (Oracle Corporation), Robert C. Seacord (CERT, Software Engineering Institute, Carnegie Mellon University), Khallai Taylor (profesora asistente, Triton College y profesora adjunta, Lonestar College: Kinwood), Jorge Vargas (Yumbling y Java Champion), Johan Vos (LodgON y Oracle Java Champion) y James L. Weaver (Oracle Corporation y autor de *Pro JavaFX 2*).

Revisores de ediciones anteriores: Soundararajan Angusamy (Sun Microsystems), Joseph Bowbeer (Consultor), William E. Duncan (Louisiana State University), Diana Franklin (University of California, Santa Barbara), Edward F. Gehringer (North Carolina State University), Ric Heishman (George Mason University), Dr. Heinz Kabutz (JavaSpecialists.eu), Patty Kraft (San Diego State University), Lawrence Premkumar (Sun Microsystems), Tim Margush (University of Akron), Sue McFarland Metzger (Villanova University), Shyamal Mitra (Th e University of Texas en Austin), Peter Pilgrim (Consultor), Manjeet Rege, Ph.D. (Rochester Institute of Technology), Susan Rodger (Duke University), Amr Sabry (Indiana

University), José Antonio González Seco (Parlamento de Andalucía), Sang Shin (Sun Microsystems), S. Sivakumar (Astra Infotech Private Limited), Raghavan "Rags" Srinivas (Intuit), Monica Sweat (Georgia Tech), Vinod Varma (Astra Infotech Private Limited) y Alexander Zuev (Sun Microsystems).

Un agradecimiento especial a Brian Goetz

Tuvimos el privilegio de que Brian Goetz (arquitecto del lenguaje Java de Oracle, líder de especifi caciones para el proyecto Lambda de Java SE 8 y coautor de *Java Concurrency in Practice*) realizara una revisión detallada de todo el libro. Examinó minuciosamente cada capítulo y proporcionó muchas ideas y comentarios constructivos bastante útiles. Cualquier otra falla en el libro es culpa nuestra.

Bueno, ¡ahí lo tiene! A medida que lea el libro, apreciaremos con sinceridad sus comentarios, críticas, correcciones y sugerencias para mejorar el texto. Dirija toda su correspondencia a:

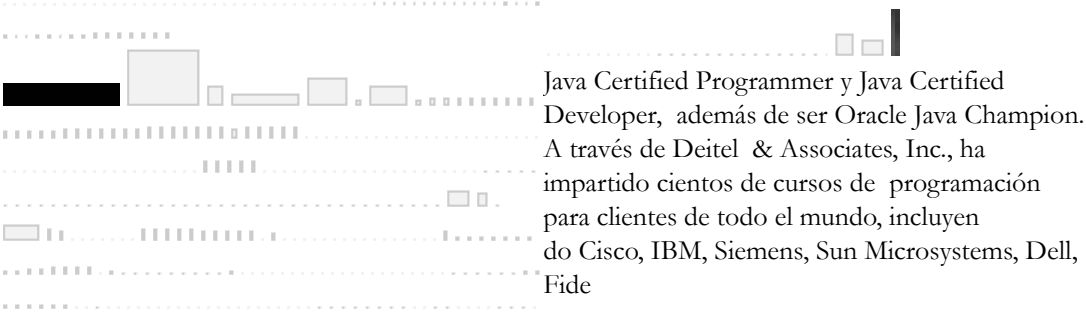
deitel@deitel.com

Le responderemos oportunamente. Esperamos que disfrute trabajando con este libro tanto como nosotros al escribirlo.

Paul y Harvey Deitel

Acerca de los autores

Paul J. Deitel, CEO y Director Técnico de Deitel & Associates, Inc., es egresado del MIT, donde estudió Tecnologías de la Información. Posee las designaciones



Java Certified Programmer y Java Certified Developer, además de ser Oracle Java Champion. A través de Deitel & Associates, Inc., ha impartido cientos de cursos de programación para clientes de todo el mundo, incluyendo Cisco, IBM, Siemens, Sun Microsystems, Dell, Fidelity, NASA en el Centro Espacial Kennedy, el National Severe Storm Laboratory, White Sands Missile Range, Rogue Wave Software, Boeing, SunGard Higher Education, Nortel Networks, Puma, iRobot, Invensys y muchos más. Él y su coautor, el Dr. Harvey M. Deitel, son autores de los libros de texto, libros profesionales y videos sobre lenguajes de programación más vendidos en el mundo.

Dr. Harvey M. Deitel, Presidente y Consejero de Estrategia de Deitel & Associates, Inc., tiene más de

50 años de experiencia en el campo de la computación. El Dr. Deitel obtuvo una licenciatura y una maestría en ingeniería eléctrica del MIT y un doctorado en Matemáticas de la Universidad de Boston. Tiene muchos años de experiencia como profesor universitario, lo cual incluye un puesto vitalicio y haber sido presidente del departamento de Ciencias de la computación en Boston College antes de fundar, Deitel & Associates, Inc. en 1991 junto con su hijo Paul. Los libros de los Deitel se han ganado el reconocimiento internacional
y han sido traducidos al japonés, alemán, ruso, español, coreano, francés, polaco, italiano, chino simplifi cado, chino tradicional, coreano, portugués, griego, urdú y turco. El Dr. Deitel ha impartido cientos de seminarios profesionales para grandes empresas, instituciones académicas, organizaciones gubernamenta les y diversos sectores del ejército.

Acerca de Deitel & Associates, Inc.

Deitel & Associates, Inc., fundada por Paul Deitel y Harvey Deitel, es una empresa reconocida a nivel mundial dedicada a la capacitación corporativa y la creación de contenidos que se especializa en lenguajes de programación computacionales, tecnología de objetos, desarrollo de aplicaciones móviles y tecnología de software de Internet y Web. Sus clientes de capacitación incluyen muchas de las empresas más grandes del mundo, agencias gubernamentales, sectores del ejército e instituciones académicas. La empresa proporciona cursos presenciales en todo el mundo, sobre la mayoría de los lenguajes y plataformas

de programación, como Java™, desarrollo de aplicaciones para Android™, desarrollo de aplicaciones de Objective-C e iOS, C++, C, Visual C#®, Visual Basic®, Visual C++®, Python®, tecnología de objetos, programación en Internet y Web, y una lista cada vez mayor de cursos adicionales de programación y desarrollo de software.

A lo largo de su sociedad editorial de más de 39 años con Pearson/Prentice Hall, Deitel & Associates, Inc. ha publicado libros de texto de vanguardia sobre programación y libros profesionales tanto impresos como en un amplio rango de formatos de libros electrónicos, además de cursos de video *LiveLessons*. Puede contactarse con Deitel & Associates, Inc. y con los autores por medio de correo electrónico:

deitel@deitel.com

Para conocer más acerca de la oferta de la Serie de Capacitación Corporativa *Dive Into*® de Deitel,

visite: http://www.deitel.com/training/

Para solicitar una cotización de capacitación presencial en su organización, envíe un correo a deitel@deitel.com.

Las personas que deseen comprar libros de Deitel y cursos de capacitación *LiveLessons* pueden hacerlo a través de www.deitel.com. Las empresas, gobiernos, instituciones militares y académicas que deseen realizar pedidos al mayoreo deben hacerlo directamente con Pearson. Para obtener más información, visite

http://www.informit.com/store/sales.aspx

Esta sección contiene información que debe revisar antes de usar este libro. Cualquier actualización a la información que se presenta aquí, será publicada en:

http://www.deitel.com/books/jhtp10

Convenciones de tipos de letra y nomenclatura

Utilizamos tipos de letra distintos para diferenciar entre los componentes de la pantalla (como los nombres de menús y los elementos de los mismos) y el código o los comandos en Java. Nuestra convención es hacer hincapié en los componentes en pantalla en una fuente Helvetica sans-serif en negritas (por ejemplo, el menú Archivo) y enfatizar el código y los comandos de Java en una fuente Lucida sans-serif (por ejemplo, System.out.println()). Hemos omitido intencionalmente el uso de acentos y caracteres especiales en los segmentos de código donde estos caracteres podrían presentar alguna difi cultad para la correcta ejecución del programa.

Software a utilizar en el libro

Podrá descargar sin costo todo el software necesario para este libro a través de la Web. Con excepción de los ejemplos específi cos para Java SE 8, todos los ejemplos se probaron con los kits de desarrollo Java Standard Edition Development Kits (JDK) de las ediciones Java SE 7 y Java SE 8.

Kit de desarrollo de software Java Standard Edition 7 (JDK 7)
Las plataformas de JDK 7 para Windows, OS X y Linux están disponibles en:

http://www.oracle.com/technetwork/java/javase/downloads/index.html

Kit de desarrollo de software Java Standard Edition 8 (JDK 8)
Al momento de publicar este libro, la versión más reciente de JDK 8 para las plataformas Windows, OS X y Linux estaba disponible en:

http://jdk8.java.net/download.html

Una vez que se libere la versión fi nal de JDK 8, estará disponible en:

http://www.oracle.com/technetwork/java/javase/downloads/index.html

Instrucciones de instalación del JDK

Después de descargar el instalador del JDK, asegúrese de seguir con cuidado las instrucciones de instalación para su plataforma, las cuales están disponibles en:

http://docs.oracle.com/javase/7/docs/webnotes/install/index.html

Aunque estas instrucciones son para el JDK 7, también se aplican al JDK 8; sólo tendrá que actualizar el número de versión del JDK en las instrucciones específi cas de cualquier otra versión.

Cómo establecer la variable de entorno PATH

La variable de entorno PATH en su computadora determina los directorios en los que la computadora debe buscar aplicaciones, como las que le permiten compilar y ejecutar sus aplicaciones de Java (llamadas javac y java, respectivamente). *Para lograr establecer correctamente la variable de entorno PATH, siga con cuidado las instrucciones de instalación de Java en su plataforma.* Los pasos para establecer las variables de entorno difi eren según el sistema operativo, y algunas veces dependen también de la versión del mismo (por ejemplo, Windows 7 o Windows 8). Las instrucciones para diversas plataformas se

encuentran en:

http://www.java.com/en/download/help/path.xml

Si no establece la variable PATH de manera correcta en Windows, y en algunas instalaciones de Linux, recibirá un mensaje como éste cuando utilice las herramientas del JDK:

'java' no se reconoce como un comando interno o externo, un programa ejecutable ni un archivo por lotes.

En este caso, regrese a las instrucciones de instalación para establecer la variable PATH y vuelva a comprobar sus pasos. Si descargó una versión más reciente del JDK, tal vez tenga que cambiar el nombre del directorio de instalación del JDK en la variable PATH.

Directorio de instalación del JDK y el subdirectorio bin

El directorio de instalación del JDK varía según la plataforma que se utilice. Los directorios que se listan a continuación son para el JDK 7 actualización 51 de Oracle:

- JDK de 32 bits en Windows:
  C:\Program Files (x86)\Java\jdk1.7.0_51
- JDK de 64 bits en Windows:
  C:\Program Files\Java\jdk1.7.0_51
- Mac OS X:
  /Library/Java/JavaVirtualMachines/jdk1.7.0_51.jdk/Contents/Home
- Linux Ubuntu:
  /usr/lib/jvm/java-7-oracle

Dependiendo de la plataforma que utilice y del idioma que tenga configurado, el nombre de la carpeta de instalación del JDK podría diferir si usa una actualización distinta del JDK 7 o si usa el JDK 8. En el caso de Linux, la ubicación de instalación depende del instalador que utilice y posiblemente de su versión de Linux. Nosotros usamos Linux Ubuntu. La variable de entorno PATH debe apuntar al subdirectorio ***bin*** del directorio de instalación JDK.

Al establecer PATH, asegúrese de usar el nombre del directorio de instalación del JDK correcto para la versión específi ca del JDK que instaló; a medida que haya versiones disponibles de JDK más recientes, el nombre del directorio de instalación del JDK cambia para incluir un *número de versión de actualización*. Por ejemplo, al momento de escribir este libro la versión del JDK 7 más reciente era la actualización 51. Para esta versión, el nombre del directorio de instalación del JDK termina con "_51".

Cómo establecer la variable de entorno CLASSPATH

Si intenta ejecutar un programa de Java y recibe un mensaje como éste:

Exception in thread "main" java.lang.NoClassDefFoundError: *SuClase*

entonces su sistema tiene una variable de entorno CLASSPATH que debe modifi carse. Para corregir el error anterior, siga los pasos para establecer la variable de entorno PATH, localice la variable CLASSPATH y modifi que su valor para que incluya el directorio local, que por lo general se representa como un punto (.). En Windows agregue

.;

al principio del valor de CLASSPATH (sin espacios antes o después de esos caracteres). En otras plataformas, sustituya el punto y coma con los caracteres separadores de ruta apropiados; por lo general, el signo de dos puntos (:).

Cómo establecer la variable de entorno JAVA_HOME

El software de bases de datos Java DB que usará en el capítulo 24 y en varios de los capítulos en línea,

requiere que establezca la variable de entorno JAVA_HOME en su directorio de instalación del JDK. Puede usar los mismos pasos que utilizó al establecer la variable PATH para establecer las demás variables de entorno, como JAVA_HOME.

Entornos integrados de desarrollo (IDE) de Java

Hay muchos entornos integrados de desarrollo de Java que usted puede usar para programar en este lenguaje. Por esta razón, para la mayoría de los ejemplos del libro usaremos sólo las herramientas de línea de comandos del JDK. Proporcionamos videos Dive-Into® que muestran cómo descargar, instalar y usar tres IDE populares: NetBeans, Eclipse e IntelliJ IDEA. Usaremos NetBeans en el capítulo 25 y en varios de los capítulos en línea del libro.

Descargas de NetBeans
Puede descargar el paquete JDK/NetBeans en:

http://www.oracle.com/technetwork/java/javase/downloads/index.html

La versión de NetBeans incluida con el JDK es para el desarrollo con Java SE. Los capítulos en línea de JavaServer Faces (JSF) y de los servicios Web, usan la versión Java Enterprise Edition (Java EE) de NetBeans, que puede descargar en:

https://netbeans.org/downloads/

Esta versión soporta el desarrollo con Java SE y Java EE.

Descargas de Eclipse
Puede descargar el IDE Eclipse en:

https://www.eclipse.org/downloads/

Para el desarrollo con Java SE, seleccione el IDE Eclipse para desarrolladores de Java. Para el desarrollo con Java Enterprise Edition (Java EE) (como JSF y los servicios Web), seleccione el IDE Eclipse para desarrolladores de Java EE; esta versión soporta el desarrollo con Java SE y Java EE.

Descargas de IntelliJ IDEA Community Edition
Puede descargar el entorno IntelliJ IDEA Community Edition gratuito en:

http://www.jetbrains.com/idea/

La versión gratuita soporta solamente el desarrollo con Java SE.

Cómo obtener los ejemplos de código

Los ejemplos de este libro están disponibles para descargarlos sin costo en

http://www.deitel.com/books/jhtp10/

bajo el encabezado Download Code Examples and Other Premium Content (Descargar ejemplos de códi go y contenido especial adicional). Los ejemplos también están disponibles en

http://www.pearsonhighered.com/deitel

Cuando descargue el archivo ZIP en su computadora, anote la ubicación en donde eligió guardarlo. Extraiga el contenido del archivo examples.zip; utilice una herramienta como 7-Sip (www.7-zip-org), WinZip (www.winzip.com) o las herramientas integradas de su sistema operativo. Las instrucciones en el libro asumen que los ejemplos se encuentran en:

- C:\ejemplos en Windows
- la subcarpeta ejemplos de la carpeta de inicio de su cuenta de usuario en Linux •

la subcarpeta ejemplos de la carpeta Documentos en Mac OS X

Apariencia visual Nimbus de Java

Java incluye una apariencia visual multiplataforma conocida como Nimbus. En los programas con interfaces gráfi cas de usuario Swing (como en los capítulos 12 y 22), hemos configurado nuestros equipos de prueba para usar Nimbus como la apariencia visual predeterminada.

Para establecer Nimbus como la opción predeterminada para todas las aplicaciones de Java, debe crear un archivo de texto llamado swing.properties en la carpeta lib de las carpetas de instalación del JDK y del JRE. Coloque la siguiente línea de código en el archivo:

```
swing defaultlaf=com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel
```

Para obtener más información sobre cómo localizar estas carpetas, visite http://docs.oracle.com/javase/7/docs/webnotes/install/index.html [*nota:* además del JRE individual, hay un JRE anidado en su carpeta de instalación del JDK. Si utiliza un IDE que depende del JDK (como NetBeans), tal vez también tenga que colocar el archivo swing.properties en la carpeta lib de la carpeta jre anidada].

Ahora está listo para empezar sus estudios de Java con el libro *Cómo programar en Java, 10ª edición*. ¡Esperamos que disfrute el libro!

Internet y Java

Introducción a las computadoras,

El hombre sigue siendo la computadora más extraordinaria de todas.
—John F. Kennedy

Un buen diseño es un buen negocio.
—Thomas J. Watson, fundador de IBM

Objetivos

En este capítulo aprenderá sobre:

- Los excitantes y recientes desarrollos en el campo de las computadoras.
- Los conceptos básicos de hardware, software y redes.
- La jerarquía de datos.

- Los distintos tipos de lenguajes de programación.
- La importancia de Java y los otros lenguajes de programación líderes.
- Los fundamentos de la programación orientada a objetos.
- La importancia de Internet y Web.
- Un entorno de desarrollo típico de programas en Java.
- Cómo probar una aplicación en Java.
- Algunas de las recientes tecnologías de software clave.
- Cómo mantenerse actualizado con las tecnologías de la información.

modelado)

Ejercicios de autoevaluación | *Respuestas a los ejercicios de autoevaluación* | *Ejercicios* | *Marcando la diferencia*

1.1 Introducción

Bienvenido a Java, uno de los lenguajes de programación de computadoras más utilizados en el mundo. Usted ya está familiarizado con las poderosas tareas que realizan las computadoras. Mediante este libro de texto, podrá escribir instrucciones que ordenen a las computadoras que realicen esos tipos de tareas. El **software** (es decir, las instrucciones que usted escribe) controla el **hardware** (es decir, las computadoras). Usted aprenderá sobre la *programación orientada a objetos*: la principal metodología de programación en la actualidad. En este texto creará y trabajará con muchos *objetos de software*.

Java es el lenguaje preferido para satisfacer las necesidades de programación empresariales de muchas organizaciones. También se ha convertido en el lenguaje de elección para implementar aplicaciones basadas en Internet y software para dispositivos que se comunican a través de una red.

Forrester Research pronostica que habrá más de dos mil millones de computadoras personales en uso para 2015.[1] De acuerdo con Oracle, el 97% de los equipos de escritorio empresariales, el 89% de las PC de escritorio, tres mil millones de dispositivos (figura 1.1) y el 100% de todos los reproductores Blu-Ray Disc™ ejecutan Java, y hay más de 9 millones de desarrolladores de este lenguaje.[2]

De acuerdo con un estudio realizado por Gartner, los dispositivos móviles seguirán superando a las PC como los dispositivos primarios de los usuarios: se estima que se entregarán alrededor de 2 mil millones de teléfonos inteligentes y 388 millones de tabletas en 2015; 8.7 veces el número de equipos PC.[3] Para 2018,

1 http://www.worldometers.info/computers.
2 http://www.oracle.com/technetwork/articles/java/javaone12review-1863742.html. 3 http://www.gartner.com/newsroom/id/2645115.

Sistemas de aeroplanos	Cajeros automáticos (ATM)	Sistemas de infoentretenimiento de automóviles
Reproductores Blu-Ray Disc™	Decodificadores de TV por cable	Copiadoras
Tarjetas de crédito	Escáneres para TC	Computadoras de escritorio
Lectores de libros electrónicos	Consolas de juegos	Sistemas de navegación GPS
	Sistemas de seguridad para el hogar	Interruptores de luz
Electrodomésticos	Dispositivos médicos	Teléfonos móviles
Terminales de lotería	Estaciones de pago de estacionamiento	Impresoras
MRI	Robots	Enrutadores
Pases de transporte	Medidores inteligentes	Plumas inteligentes
Tarjetas inteligentes	Tabletas	Televisiones
Teléfonos inteligentes	Termostatos	Sistemas de diagnóstico vehicular
Decodificadores para televisores		

Fig. 1.1 Algunos dispositivos que utilizan Java.

se espera que el mercado de las aplicaciones móviles (apps) llegue a $92 mil millones.[4] Esto genera oportunidades profesionales importantes para las personas que programan aplicaciones móviles, muchas de las cuales se programan en Java (vea la sección 1.6.3).

Java Standard Edition

Java ha evolucionado con tanta rapidez que esta décima edición de *Cómo programar en Java* —basada en

Java Standard Edition 7 (Java SE 7) y **Java Standard Edition 8 (Java SE 8)**— se publicó tan sólo 17 años después de la primera edición. Java Standard Edition contiene las herramientas necesarias para desarrollar aplicaciones de escritorio y de servidor. El libro puede usarse *ya sea* con Java SE 7 *o* Java SE 8. Todas las características de Java SE 8 se analizan en secciones modulares que son fáciles de incluir u omi tir a lo largo del libro.

Antes de Java SE 8, Java soportaba tres paradigmas de programación: *programación por procedimientos*, *programación orientada a objetos* y *programación genérica*. Java SE 8 agrega la *programación funcional*. En el capítulo 17 le mostraremos cómo usar la programación funcional para escribir programas en forma más rápida y concisa, con menos errores y que sean fáciles de *paralelizar* (es decir, que puedan realizar varios cálculos al mismo tiempo) para aprovechar las arquitecturas de hardware multinúcleo actuales que mejoran el rendimiento de una aplicación.

Java Enterprise Edition

Java se utiliza en un espectro tan amplio de aplicaciones que tiene otras dos ediciones. **Java Enterprise Edition (Java EE)** está orientada hacia el desarrollo de aplicaciones de red distribuidas, de gran escala, y aplicaciones basadas en Web. En el pasado, la mayoría de las aplicaciones de computadora se ejecutaban en computadoras "independientes" (que no estaban conectadas en red). En la actualidad se pueden escribir aplicaciones que se comuniquen con computadoras en todo el mundo por medio de Internet y Web. Más adelante hablaremos sobre cómo crear dichas aplicaciones basadas en Web con Java.

4 https://www.abiresearch.com/press/tablets-will-generate-35-of-this-years-25-billion-.

Java Micro Edition

Java Micro Edition (Java ME) (un subconjunto de Java SE) está orientada hacia el desarrollo de aplica ciones para pequeños dispositivos incrustados con una capacidad limitada de memoria, como los relojes inteligentes, los reproductores MP3, los decodificadores para televisión, los medidores inteligentes (para monitorear el uso de la energía eléctrica) y más.

1.2 Hardware y software

Las computadoras pueden realizar cálculos y tomar decisiones lógicas con una rapidez increíblemente mayor que los humanos. Muchas de las computadoras personales actuales pueden realizar miles de millones de cálculos en un segundo; más de lo que un humano podría realizar en toda su vida. ¡Las *supercomputado ras* ya pueden realizar *miles de millones* de instrucciones por segundo! ¡La supercomputadora Tianhe-2 de la National University of Defense Technology de China puede realizar más de 33 mil billones de cálculos por segundo (33.86 *petaflops*)![5] Para ilustrar este ejemplo, *¡la supercomputadora Tianhe-2 puede ejecutar en un segundo el equivalente a cerca de 3 millones de cálculos por cada habitante del planeta!* Y estos "límites máximos" están aumentando con rapidez.

Las computadoras procesan datos bajo el control de secuencias de instrucciones conocidas como **pro gramas de computadora**. Estos programas guían a la computadora a través de acciones ordenadas especi ficadas por gente conocida como **programadores** de computadoras. En este libro aprenderá una metodo logía de programación clave que está mejorando la productividad del programador, con lo cual se reducen
los costos de desarrollo del software: la *programación orientada a objetos*.

Una computadora consiste en varios dispositivos conocidos como hardware (teclado, pantalla, ratón, discos duros, memoria, unidades de DVD y unidades de procesamiento). Los costos de las computadoras *han disminuido en forma espectacular*, debido a los rápidos desarrollos en las tecnologías de hardware y software. Las computadoras que ocupaban grandes espacios y que costaban millones de dólares hace algu
nas décadas, ahora pueden grabarse en superficies de chips de silicio más pequeños que una uña, y con

un costo de quizá unos cuantos dólares cada uno. Aunque suene irónico, el silicio es uno de los materiales más abundantes en el planeta (es un ingrediente de la arena común). La tecnología de los chips de silicio se ha vuelto tan económica que las computadoras se han convertido en un producto básico.

1.2.1 Ley de Moore

Es probable que cada año espere pagar al menos un poco más por la mayoría de los productos y servicios que utiliza. En los campos de las computadoras y las comunicaciones se ha dado lo opuesto, en especial con relación a los costos del hardware que da soporte a estas tecnologías. Los costos del hardware han disminui do con rapidez durante varias décadas.

Cada uno o dos años, las capacidades de las computadoras se *duplican* aproximadamente sin que el precio se incremente. Esta notable observación se conoce en el ámbito común como la **Ley de Moore**, y debe su nombre a la persona que identificó esta tendencia en 1960: Gordon Moore, cofundador de Intel (uno de los principales fabricantes de procesadores para las computadoras y los sistemas incrustados (em bebidos) de la actualidad). La Ley de Moore y las observaciones relacionadas son especialmente ciertas para la cantidad de memoria que tienen destinada las computadoras para programas, para la cantidad de alma cenamiento secundario (como el almacenamiento en disco) que tienen para guardar los programas y datos
durante periodos extendidos de tiempo, y para las velocidades de sus procesadores (las velocidades con que las computadoras *ejecutan* sus programas y realizan su trabajo).

Se ha producido un crecimiento similar en el campo de las comunicaciones, en donde los costos se han desplomado a medida que la enorme demanda por el *ancho de banda* de las comunicaciones (es decir, la

5 http://www.top500.org/.

capacidad de transmisión de información) atrae una competencia intensa. No conocemos otros campos en los que la tecnología mejore con tanta rapidez y los costos disminuyan de una manera tan drástica. Dicha mejora fenomenal está fomentando sin duda la *Revolución de la información*.

1.2.2 Organización de la computadora

Sin importar las diferencias en la apariencia física, es posible percibir una segmentación de las computadoras en varias **unidades lógicas** o secciones (figura 1.2).

Unidad de entrada

Unidad

de salida

Esta sección de "embarque" toma información que ya ha sido procesada por la compu tadora y la coloca en los diferentes **dispositivos de salida**, para que esté disponible fuera de la computadora. En la actualidad, la mayor parte de la información de salida de las computadoras se muestra en pantallas (incluyendo pantallas táctiles), se imprime en papel (lo cual no es muy bueno para el medio ambiente), se reproduce como audio o video en equipos PC y reproductores de medios (como los iPod de Apple) y pantallas gigantes en estadios deportivos, se transmite a través de Internet, o se utiliza para controlar otros dispositivos, como robots y aparatos "inteligentes". La información también se envía por lo general a dispositivos de almacenamiento secundarios, como discos duros, unidades DVD y unidades Flash USB. Una forma popular reciente de salida es la vibración de los teléfonos inteligentes.

Unidad de memoria

Esta sección de "recepción" obtiene información (datos y programas de cómputo) de los **dispositivos de entrada** y la pone a disposición de las otras unidades para que pueda procesarse. La mayor parte de la información se introduce a través de los teclados, las pantallas táctiles y los ratones. La información también puede introducirse de muchas otras formas, como a través de comandos de voz, la digitalización de imágenes y códigos de barras, por medio de dispositivos de almacenamiento secundario (como discos duros, unidades de DVD, Blu-ray Disc™ y memorias Flash USB —también conocidas como "*thumb drives*" o "*memory sticks*"), mediante la recepción de video de una cámara Web y al recibir información en su computadora a través de Internet (como cuando descarga videos de YouTube™ o libros electrónicos de Amazon). Las formas más recientes de entrada son: la lectura de datos de geolocalización a través de un dispositivo GPS, y la información sobre el movimiento y la orientación mediante un *acelerómetro* (un dispositivo que responde a la aceleración hacia arriba o abajo, a la derecha o izquierda y hacia delante o atrás) en un teléfono inteligente o un controlador de juegos (Como Microsoft® Kinect® y Xbox®, Wii™ Remote y Sony® PlayStation® Move).

Esta sección de "almacén" de acceso rápido, pero con relativa baja capacidad, retiene la información que se introduce a través de la unidad de entrada para que pueda estar disponible de manera inmediata y procesarla cuando sea necesario. La unidad de memoria también retiene la información procesada hasta que la unidad de salida pueda colocarla en los dispositivos de salida. La información en la unidad de memoria es *volátil*, ya que por lo general se pierde cuando se apaga la computadora. Con frecuencia, a la unidad de memoria se le conoce como **memoria**, **memoria principal** o **RAM** (memoria de acceso aleatorio). Las típicas memorias principales en las computadoras de escritorio y portátiles pueden contener hasta 128 GB de RAM. GB se refiere a gigabytes; un gigabyte equivale aproximadamente a mil millones de bytes. Un **byte** equivale a ocho bits. Un bit puede ser un `0` o un `1`.

Fig. 1.2 Unidades lógicas de una computadora (parte 1 de 2).

Java

Unidad aritmética y lógica (ALU)

Unidad central de procesamiento (CPU)

Unidad de

almacenamiento secundario
Esta sección de "manufactura" realiza *cálculos* como suma, resta, multiplicación y división. También contiene los mecanismos de *decisión* que permiten a la computadora hacer cosas como, por ejemplo, comparar dos elementos de la unidad de memoria para determinar si son iguales o no. En los sistemas actuales, la ALU se implementa por lo general como parte de la siguiente unidad lógica, la CPU.

Esta sección "administrativa" coordina y supervisa la operación de las demás secciones. La CPU le indica a la unidad de entrada cuándo debe colocarse la información dentro de la unidad de memoria, a la ALU cuándo debe utilizarse la información de la unidad de memoria para los cálculos, y a la unidad de salida cuándo enviar la información desde la unidad de memoria hasta ciertos dispositivos de salida. Muchas de las computadoras actuales contienen múltiples CPU y, por lo tanto, pueden realizar muchas operaciones de manera simultánea. Un **procesador multinúcleo** implementa varios procesadores en un solo chip de circuitos integrados; un *procesador de doble núcleo (dual-core)* tiene dos CPU y un *procesador de cuádruple núcleo (quad-core)* tiene cuatro CPU. Las computadoras de escritorio de la actualidad tienen procesadores que pueden ejecutar miles de millones de instrucciones por segundo.

Ésta es la sección de "almacén" de alta capacidad y de larga duración. Los programas o datos que no utilizan con frecuencia las demás unidades se colocan por lo general en dispositivos de almacenamiento secundario (por ejemplo, el *disco duro*) hasta que se requieran de nuevo, lo cual puede llegar a ser horas, días, meses o incluso años después. La información en los dispositivos de almacenamiento secundario es *persistente*, lo que significa que se mantiene aun y cuando se apaga la computadora. El tiempo para acceder a la información en almacenamiento secundario es mucho mayor que el necesario para acceder a la de la memoria principal, pero el costo por unidad de memoria secundaria es mucho menor que el correspondiente a la unidad de memoria principal. Las unidades de disco duro, DVD y Flash USB son ejemplos de dispositivos de almacenamiento secundario, los cuales pueden contener hasta 2 TB (TB se refiere a terabytes; un terabyte equivale aproximadamente a un billón de bytes). Los discos duros típicos en las compu tadoras de escritorio y portátiles pueden contener hasta 2 TB y algunas unidades de disco duro de escritorio pueden contener hasta 4 TB.

Fig. 1.2 Unidades lógicas de una computadora (parte 2 de 2).

1.3 Jerarquía de datos

Los elementos de datos que procesan las computadoras forman una **jerarquía de datos** cuya estructura se vuelve cada vez más grande y compleja, a medida que pasamos de los elementos de datos más simples (co nocidos como "bits") a los más complejos, como los caracteres y los campos. La figura 1.3 ilustra una porción de la jerarquía de datos.

Bits

El elemento de datos más pequeño en una computadora puede asumir el valor `0` o el valor `1`. A dicho ele mento de datos se le denomina **bit** (abreviación de "dígito binario": un dígito que puede asumir uno de *dos* valores). Es increíble que todas las impresionantes funciones que realizan las computadoras impliquen sólo las manipulaciones más simples de los dígitos `0` y `1`, como *examinar el valor de un bit, establecer el valor de un bit* e *invertir el valor de un bit* (de `1` a `0` o de `0` a `1`).

Tom Azul

Sally Negro

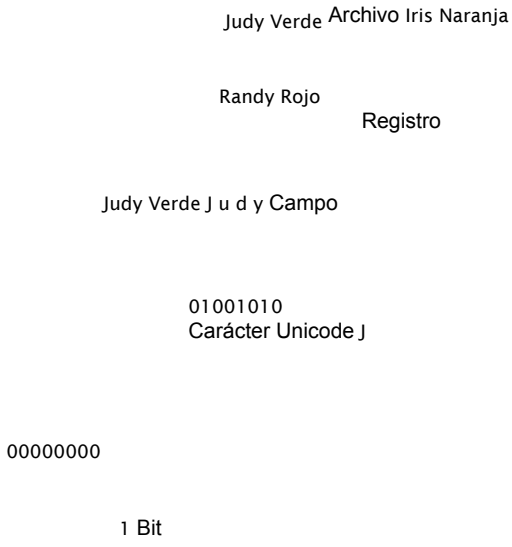


Fig. 1.3 Jerarquía de datos.

Caracteres

Es tedioso para las personas trabajar con datos en el formato de bajo nivel de los bits. En cambio, prefieren trabajar con *dígitos decimales* (0-9), *letras* (A-Z y a-z) y *símbolos especiales* (por ejemplo, $, @, %, &, *, (,), –, +, “, :, ? y /). Los dígitos, letras y símbolos especiales se conocen como **caracteres**. El **conjunto de ca racteres** de la computadora es el conjunto de todos los caracteres que se utilizan para escribir programas y representar elementos de datos. Las computadoras sólo procesan unos y ceros, por lo que el conjunto de caracteres de una computadora representa a cada carácter como un patrón de unos y ceros. Java usa carac teres **Unicode**® que están compuestos de uno, dos o cuatro bytes (8, 16 o 32 bits). Unicode contiene caracteres para muchos de los idiomas que se usan en el mundo. En el apéndice H obtendrá más informa ción sobre Unicode. En el apéndice B obtendrá más información sobre el conjunto de caracteres **ASCII (Código estándar estadounidense para el intercambio de información)**, que es el popular subconjun to de Unicode que representa las letras mayúsculas y minúsculas, los dígitos y algunos caracteres especia les comunes.

Campos

Así como los caracteres están compuestos de bits, los **campos** están compuestos de caracteres o bytes. Un campo es un grupo de caracteres o bytes que transmiten un significado. Por ejemplo, un campo compues to de letras mayúsculas y minúsculas se puede usar para representar el nombre de una persona, y un campo compuesto de dígitos decimales podría representar la edad de esa persona.

Registros

Se pueden usar varios campos relacionados para componer un **registro** (el cual se implementa como una clase [class] en Java). Por ejemplo, en un sistema de nómina, el registro de un empleado podría consistir en los siguientes campos (los posibles tipos para estos campos se muestran entre paréntesis):

- Número de identificación del empleado (un número entero)
- Nombre (una cadena de caracteres)
- Dirección (una cadena de caracteres)
- Salario por horas (un número con punto decimal)
- Ingresos del año a la fecha (un número con punto decimal)
- Monto de impuestos retenidos (un número con punto decimal)

Por lo tanto, un registro es un grupo de campos relacionados. En el ejemplo anterior, todos los campos pertenecen al *mismo* empleado. Una compañía podría tener muchos empleados y un registro de nómina para cada uno.

Archivos

Un **archivo** es un grupo de registros relacionados. [*Nota:* dicho en forma más general, un archivo contiene datos arbitrarios en formatos arbitrarios. En algunos sistemas operativos, un archivo se ve tan sólo como una *secuencia de byte* y cualquier organización de esos bytes en un archivo, como cuando se organizan los datos en registros, es una vista creada por el programador de la aplicación. En el capítulo 15 verá cómo se hace eso]. Es muy común que una organización tenga muchos archivos, algunos de los cuales pueden contener miles de millones, o incluso billones de caracteres de información.

Base de datos

Una **base de datos** es una colección de datos organizados para facilitar su acceso y manipulación. El mo delo más popular es la *base de datos relacional*, en la que los datos se almacenan en simples *tablas*. Una tabla incluye *registros* y *campos*. Por ejemplo, una tabla de estudiantes podría incluir los campos nombre, apellido, especialidad, año, número de identificación (ID) del estudiante y promedio de calificaciones. Los datos para cada estudiante constituyen un registro y las piezas individuales de información en cada registro son los campos. Puede *buscar*, *ordenar* y manipular de otras formas los datos con base en la relación que tienen con otras tablas o bases de datos. Por ejemplo, una universidad podría utilizar datos de la base de datos de los estudiantes en combinación con los de bases de datos de cursos, alojamiento en el campus, planes ali menticios, etc. En el capítulo 24 hablaremos sobre las bases de datos.

Big Data

La cantidad de datos que se produce a nivel mundial es enorme y aumenta con rapidez. De acuerdo con IBM, cada día se generan alrededor de 2.5 trillones (2.5 *exabytes*) de datos y el 90% de los datos en el mundo se crearon ¡tan sólo en los últimos dos años![6] De acuerdo con un estudio de Digital Universe, en 2012 el suministro de datos globales llegó a 2.8 *zettabytes* (lo que equivale a 2.8 billones de giga bytes).[7] La figura 1.4 muestra algunas mediciones comunes de bytes. Las aplicaciones de **Big Data** li dian con dichas cantidades masivas de datos y este campo crece con rapidez, lo que genera muchas oportunidades para los desarrolladores de software. De acuerdo con un estudio por parte de Gartner Group, para 2015 más de 4 millones de empleos de TI a nivel mundial darán soporte a los grandes volúmenes de datos (Big Data).[8]

6 http://www-01.ibm.com/software/data/bigdata/.
7 http://www.guardian.co.uk/news/datablog/2012/dec/19/big-data-study-digital-universe-global volume.
8 http://tech.fortune.cnn.com/2013/09/04/big-data-employment-boom/.

1 kilobyte (KB) 1 megabyte	megabytes 1024 gigabytes	10^{15} (1,000,000,000,000,000
(MB) 1 gigabyte (GB) 1	1024 terabytes 1024 petabytes	bytes) 10^{18}
terabyte (TB) 1 petabyte (PB)	1024 exabytes	(1,000,000,000,000,000,000
1 exabyte (EB) 1 zettabyte	10^{3} (1024 bytes exactamente)	bytes) 10^{21}
(ZB)	10^{6} (1,000,000 bytes)	(1,000,000,000,000,000,000,00
1024 bytes	10^{9} (1,000,000,000 bytes)	0 bytes)
1024 kilobytes 1024	10^{12} (1,000,000,000,000 bytes)	

Fig. 1.4 Mediciones de bytes.

1.4 Lenguajes máquina, lenguajes ensambladores y lenguajes de alto nivel

Los programadores escriben instrucciones en diversos lenguajes de programación, algunos de los cuales los comprende directamente la computadora, mientras que otros requieren pasos intermedios de *traducción*. En la actualidad se utilizan cientos de lenguajes de computación. Éstos se dividen en tres tipos generales:

1. Lenguajes máquina
2. Lenguajes ensambladores
3. Lenguajes de alto nivel

Lenguajes máquina

Cualquier computadora sólo puede entender de manera directa su propio **lenguaje máquina**, el cual se defi ne según su diseño de hardware. Por lo general, los lenguajes máquina consisten en cadenas de números (que finalmente se reducen a unos y ceros) que instruyen a las computadoras para realizar sus operaciones más elementales, una a la vez. Los lenguajes máquina son *dependientes de la máquina* (es decir, un lenguaje máqui na en particular puede usarse sólo en un tipo de computadora). Dichos lenguajes son difíciles de comprender para los humanos. Por ejemplo, he aquí una sección de uno de los primeros programas de nómina en lengua je máquina, el cual suma el pago de las horas extras al sueldo base y almacena el resultado en el sueldo bruto:

```
+1300042774
+1400593419
+1200274027
```

Lenguajes ensambladores y ensambladores

La programación en lenguaje máquina era demasiado lenta y tediosa para la mayoría de los programadores. En vez de utilizar las cadenas de números que las computadoras podían entender de manera directa, los programadores empezaron a utilizar abreviaturas del inglés para representar las operaciones elementales. Estas abreviaturas formaron la base de los **lenguajes ensambladores**. Se desarrollaron *programas traductores* conocidos como **ensambladores** para convertir los primeros programas en lenguaje ensamblador a lengua
je máquina, a la velocidad de la computadora. La siguiente sección de un programa en lenguaje ensamblador también suma el pago de las horas extras al sueldo base y almacena el resultado en el sueldo bruto:

```
load sueldobase
add sueldoextra
store sueldobruto
```

Aunque este código es más claro para los humanos, las computadoras no lo pueden entender sino hasta que se traduce en lenguaje máquina.

Lenguajes de alto nivel y compiladores

El uso de las computadoras se incrementó rápidamente con la llegada de los lenguajes ensambladores, pero los programadores aún requerían de muchas instrucciones para llevar a cabo incluso hasta las tareas más simples. Para agilizar el proceso de programación se desarrollaron los **lenguajes de alto nivel**, en los que podían escribirse instrucciones individuales para realizar tareas importantes. Los programas traductores, denominados **compiladores**, convierten programas en lenguaje de alto nivel a lenguaje máquina. Los lenguajes de alto nivel permiten a los programadores escribir instrucciones que son muy similares al inglés común, y contienen la notación matemática común. Un programa de nómina escrito en un lenguaje de alto nivel podría contener *una* instrucción como la siguiente:

```
sueldoBruto = sueldoBase + sueldoExtra
```

Desde el punto de vista del programador, los lenguajes de alto nivel son mucho más recomendables que los lenguajes máquina o ensamblador. Java es uno de los lenguajes de alto nivel más utilizados.

Intérpretes

El proceso de compilación de un programa extenso escrito en lenguaje de alto nivel a un lenguaje máquina, puede tardar un tiempo considerable en la computadora. Los programas*intérpretes*, que se desarrollaron para ejecutar de manera directa programas en lenguaje de alto nivel, evitan el retraso de la compilación, aunque se ejecutan con más lentitud que los programas compilados. Hablaremos más sobre la forma en que trabajan los intérpretes en la sección 1.9, en donde aprenderá que Java utiliza una astuta mezcla de compilación e in terpretación, optimizada con base en el rendimiento, para ejecutar los programas.

1.5 Introducción a la tecnología de los objetos

Ahora que la demanda de software nuevo y más poderoso va en aumento, crear software en forma rápida, correcta y económica sigue siendo un objetivo difícil de alcanzar. Los *objetos*, o dicho en forma más precisa, las *clases* de las que provienen los objetos, son en esencia componentes de software *reutilizables*. Existen objetos de fecha, objetos de hora, objetos de audio, objetos de video, objetos de automóviles, objetos
de personas, etc. Casi cualquier *sustantivo* se puede representar de manera razonable como un objeto de software en términos de sus *atributos* (como el nombre, color y tamaño) y *comportamientos* (por ejemplo, calcular, moverse y comunicarse). Los grupos de desarrollado de software pueden usar una metodología de diseño e implementación orientada a objetos y modular para ser mucho más productivos de lo que era
posible con las técnicas populares anteriores, como la "programación estructurada". Por lo general los programas orientados a objetos son más fáciles de comprender, corregir y modificar.

1.5.1 El automóvil como un objeto

Para ayudarle a comprender los objetos y su contenido, empecemos con una analogía simple. Suponga que desea *conducir un auto y hacer que vaya más rápido al presionar el pedal del acelerador*. ¿Qué debe ocurrir para que usted pueda hacer esto? Bueno, antes de que pueda conducir un auto, alguien tiene que *diseñarlo*. Por lo general, un auto empieza en forma de dibujos de ingeniería, similares a los *planos de construcción* que describen el diseño de una casa. Estos dibujos de ingeniería incluyen el diseño del pedal del acelerador. El pedal *oculta* al conductor los complejos mecanismos que se encargan de que el auto

aumente su velocidad, de igual forma que el pedal del freno "oculta" los mecanismos que disminuyen la velocidad del auto y el volante "oculta" los mecanismos que hacen que el auto de vuelta. Esto permite que las personas con poco o ningún conocimiento sobre cómo funcionan los motores, los frenos y los mecanismos de la dirección puedan conducir un auto con facilidad.

Así como no es posible cocinar en la cocina que está en un plano de construcción, tampoco es posible conducir los dibujos de ingeniería de un auto. Antes de poder conducir un auto, éste debe *construirse* a partir de los dibujos de ingeniería que lo describen. Un auto completo tendrá un pedal acelerador *verdade ro* para hacer que aumente su velocidad, pero aun así no es suficiente; el auto no acelerará por su propia cuenta (¡esperemos que así sea!), así que el conductor debe *presionar* el pedal para acelerar el auto.

1.5.2 Métodos y clases

Ahora vamos a utilizar nuestro ejemplo del auto para presentar algunos conceptos clave de la progra mación orientada a objetos. Para realizar una tarea en una aplicación se requiere un **método**. Ese método aloja las instrucciones del programa que se encargan de realizar sus tareas. El método oculta al usuario estas instrucciones, de la misma forma que el pedal del acelerador de un auto oculta al conductor los mecanismos para hacer que el auto vaya más rápido. En Java, creamos una unidad de programa llamada **clase** para alojar el conjunto de métodos que realizan las tareas de esa clase. Por ejemplo, una clase que representa a una cuenta bancaria podría contener un método para *depositar* dinero en una cuenta, otro para *retirar* dinero de una cuenta y un tercero para *solicitar* el saldo actual de la cuenta. Una clase es similar en concepto a los dibujos de ingeniería de un auto, que contienen el diseño de un pedal acelerador, volante de dirección, etcétera.

1.5.3 Instanciación

Así como alguien tiene que *construir un auto* a partir de sus dibujos de ingeniería para que alguien más pueda conducirlo después, también es necesario *crear un objeto* de una clase para que un programa pueda realizar las tareas definidas por los métodos de esa clase. Al proceso de hacer esto se le denomina *instancia ción*. Por lo tanto, un objeto viene siendo una **instancia** de su clase.

1.5.4 Reutilización

Así como los dibujos de ingeniería de un auto se pueden *reutilizar* muchas veces para construir muchos autos, también es posible *reutilizar* una clase muchas veces para crear muchos objetos. Al reutilizar las clases existentes para crear nuevas clases y programas, ahorramos tiempo y esfuerzo. La reutilización tam bién nos ayuda a crear sistemas más confiables y efectivos, ya que con frecuencia las clases y los componen tes existentes pasan por un extenso proceso de *prueba*, *depuración* y optimización del *desempeño*. De la misma manera en que la noción de *piezas intercambiables* fue crucial para la Revolución Industrial, las clases reutilizables son cruciales para la revolución del software incitada por la tecnología de objetos.

Observación de ingeniería de software 1.1

Use un método de construcción en bloques para crear sus programas. Evite reinventar la rueda: use piezas existentes siempre que sea posible. Esta reutilización de software *es un beneficio clave de la programación orientada a objetos.*

1.5.5 Mensajes y llamadas a métodos

Cuando usted conduce un auto, al presionar el pedal del acelerador envía un *mensaje* al auto para que realice una tarea: aumentar la velocidad. De manera similar, es posible *enviar mensajes a un objeto*. Cada mensaje se implementa como **llamada a método**, para indicar a un método del objeto que realice su tarea. Por ejemplo, un programa podría llamar al método *depositar* de un objeto cuenta de banco para aumentar el saldo de esa cuenta.

1.5.6 Atributos y variables de instancia

Además de tener capacidades para realizar tareas, un auto también tiene *atributos*: color, número de puer tas, capacidad de gasolina en el tanque, velocidad actual y registro del total de kilómetros recorridos (es

decir, la lectura de su odómetro). Al igual que sus capacidades, los atributos del auto se representan como parte de su diseño en sus diagramas de ingeniería (que, por ejemplo, incluyen un velocímetro y un indi cador de combustible). Al conducir un auto real, estos atributos se llevan junto con el auto. Cada auto mantiene sus *propios* atributos. Por ejemplo, cada uno sabe cuánta gasolina hay en su tanque, pero *no* cuánta hay en los tanques de *otros* autos.

De manera similar, un objeto tiene atributos que lleva consigo a medida que se utiliza en un programa. Estos atributos se especifican como parte del objeto de esa clase. Por ejemplo, un objeto *cuenta bancaria* tiene un *atributo saldo* que representa la cantidad de dinero en la cuenta. Cada objeto cuenta bancaria co noce el saldo de la cuenta que representa, pero *no* los saldos de las *otras* cuentas en el banco. Los atributos se especifican mediante las **variables de instancia** de la clase.

1.5.7 Encapsulamiento y ocultamiento de información

Las clases y sus objetos **encapsulan** (envuelven) sus atributos y métodos. Los atributos y métodos de una clase (y sus objetos) están muy relacionados entre sí. Los objetos se pueden comunicar entre sí, pero por lo general no se les permite saber cómo están implementados otros objetos; los detalles de implementa ción están *ocultos* dentro de los mismos objetos. Este **ocultamiento de información**, como veremos más adelante, es crucial para la buena ingeniería de software.

1.5.8 Herencia

Mediante la **herencia** es posible crear con rapidez y de manera conveniente una nueva clase de objetos. La nueva clase (conocida como **subclase**) comienza con las características de una clase existente (cono cida como **superclase**), con la posibilidad de personalizarlas y agregar características únicas propias. En nuestra analogía del auto, sin duda un objeto de la clase "convertible" es *un* objeto de la clase más *gene ral* llamada "automóvil" pero, de manera más *específica*, el toldo puede ponerse o quitarse.

1.5.9 Interfaces

Java también soporta las **interfaces**: colecciones de métodos relacionados que por lo general nos permi ten indicar a los objetos *qué* hacer, pero no *cómo* hacerlo (en Java SE 8 veremos una excepción a esto). En la analogía del auto, una interfaz con "capacidades básicas de conducción" que consista de un volan te de dirección, un pedal acelerador y un pedal del freno, permitiría a un conductor indicar al auto *qué* debe hacer. Una vez que sepa cómo usar esta interfaz para dar vuelta, acelerar y frenar, podrá conducir muchos tipos de autos, incluso aunque los fabricantes puedan *implementar* estos sistemas de manera *diferente*.

Una clase **implementa** cero o más interfaces, cada una de las cuales puede tener uno o más métodos, al igual que un auto implementa interfaces separadas para las funciones básicas de conducción, para el control del radio, el control de los sistemas de calefacción y aire acondicionado, etcétera. Así como los fabricantes de automóviles implementan las capacidades de manera *diferente*, las clases pueden imple

mentar los métodos de una interfaz de manera *diferente*. Por ejemplo, un sistema de software puede incluir una interfaz de "respaldo" que ofrezca los métodos *guardar* y *restaurar*. Las clases pueden imple mentar esos métodos de manera distinta, dependiendo de los tipos de cosas que se vayan a respaldar, como programas, textos, archivos de audio, videos, etc., y los tipos de dispositivos en donde se vayan a almacenar estos elementos.

1.5.10 Análisis y diseño orientado a objetos (A/DOO)

Pronto escribirá programas en Java. ¿Cómo creará el **código** (es decir, las instrucciones) para sus

programas? Tal vez, al igual que muchos programadores, sólo encenderá su computadora y empezará a escribir. Quizás

este método funcione para pequeños programas (como los que presentamos en los primeros capítulos del libro), pero ¿qué tal si le pidieran crear un sistema de software para controlar miles de cajeros automáticos para un banco importante? O ¿qué tal si le piden que trabaje con un equipo de 1,000 desarrolladores de software para crear el nuevo sistema de control de tráfico aéreo en Estados Unidos? Para proyectos tan
grandes y complejos, no es conveniente tan sólo sentarse y empezar a escribir programas. Para crear las mejores soluciones, debe seguir un proceso de **análisis** detallado para determinar los **requerimientos** de su proyecto (definir *qué* se supone que debe hacer el sistema) y desarrollar un **diseño** que los satisfaga (decidir *cómo* debe hacerlo el sistema). Lo ideal sería pasar por este proceso y revisar el diseño con cuidado (además de pedir a otros profesionales de software que revisen su diseño) antes de escribir cualquier código. Si este proceso implica analizar y diseñar su sistema desde un punto de vista orientado a objetos, se denomina **proceso de análisis y diseño orientado a objetos (A/DOO)**. Los lenguajes como Java son orientados a objetos. La programación en un lenguaje de este tipo, conocida como **programación orientada a objetos (POO)**, le permite implementar un diseño orientado a obje tos como un sistema funcional.

1.5.11 El UML (Lenguaje unificado de modelado)

Aunque existen muchos procesos de A/DOO distintos, hay un solo lenguaje gráfico para comunicar los resultados de *cualquier* proceso de A/DOO que se utiliza en la mayoría de los casos. Este lenguaje, co nocido como Lenguaje unificado de modelado (UML), es en la actualidad el esquema gráfico más uti lizado para modelar sistemas orientados a objetos. Presentamos nuestros primeros diagramas de UML en los capítulos 3 y 4; después los utilizamos en nuestro análisis más detallado de la programación orientada a objetos en el capítulo 11. En nuestro ejemplo práctico *opcional* de ingeniería de software del ATM en los capítulos 33 y 34 presentamos un subconjunto simple de las características del UML, mien tras lo guiamos por una experiencia de diseño orientada a objetos.

1.6 Sistemas operativos

Los **sistemas operativos** son sistemas de software que se encargan de hacer más conveniente el uso de las computadoras para los usuarios, desarrolladores de aplicaciones y administradores de sistemas. Los sistemas operativos proveen servicios que permiten a cada aplicación ejecutarse en forma segura, eficien te y *concurrente* (es decir, en paralelo) con otras aplicaciones. Al software que contiene los componentes básicos del sistema operativo se denomina **kernel**. Los sistemas operativos de escritorio populares son: Linux, Windows y Mac OS X. Los sistemas operativos móviles populares que se utilizan en teléfonos inteligentes y tabletas son: Android de Google, iOS de Apple (para sus dispositivos iPhone, iPad e iPod Touch), Windows Phone 8 y BlackBerry OS.

1.6.1 Windows: un sistema operativo propietario

A mediados de la década de 1980 Microsoft desarrolló el **sistema operativo Windows**, el cual con siste en una interfaz gráfica de usuario creada sobre DOS: un sistema operativo de computadora personal muy popular con el que los usuarios interactuaban tecleando comandos. Windows tomó prestados muchos conceptos (como los iconos, menús y ventanas) que se hicieron populares gracias a los primeros sistemas operativos Apple Macintosh, desarrollados en un principio por Xerox PARC. Windows 10 es el sistema operativo más reciente de Microsoft; sus características incluyen soporte para equipos PC y tabletas, una interfaz de usuario basada en mosaicos, mejoras en la seguridad, soporte para pantallas táctiles y multitáctiles, entre otras cosas más. Windows es un sistema operativo *propietario*; está bajo el control exclusivo de Microsoft. Es por mucho el sistema operativo más uti lizado

en el mundo.

1.6.2 Linux: un sistema operativo de código fuente abierto

El sistema operativo **Linux** (muy popular en servidores, computadoras personales y sistemas incrusta dos) es tal vez el más grande éxito del movimiento de *código fuente abierto*. El **código fuente abierto** es un estilo de desarrollo de software que se desvía del desarrollo *propietario* (que se utiliza, por ejemplo, con Windows de Microsoft y Mac OS X de Apple). Con el desarrollo de código fuente abierto, indivi duos y compañías (por lo general a nivel mundial) suman sus esfuerzos para desarrollar, mantener y evolucionar el software. Cualquiera lo puede usar y personalizar para sus propios fines, por lo general sin costo. Ahora el Kit de desarrollo de Java y muchas de las tecnologías de Java relacionadas son de código fuente abierto.

Algunas organizaciones en la comunidad de código fuente abierto son: la *fundación Eclipse* (el *En torno integrado de desarrollo Eclipse* ayuda a los programadores de Java a desarrollar software de manera conveniente), la *fundación Mozilla* (creadores del *navegador Web Firefox*), la *fundación de software Apache* (creadores del *servidor Web Apache que entrega páginas Web a través de Internet* en respuesta a las solicitu des de los navegadores Web) y *GitHub* y *SourceForge*(que proporcionan las *herramientas para administrar proyectos de código fuente abierto*).

Las rápidas mejoras en la computación y las comunicaciones, la reducción en costos y el software de código fuente abierto han logrado que en la actualidad sea mucho más fácil y económico crear un nego cio basado en software de lo que era hace unas cuantas décadas. Facebook, que se inició desde un dor mitorio universitario, se creó con software de código fuente abierto.[9]

Son varias cuestiones —el poder de mercado de Microsoft, el relativamente pequeño número de aplicaciones Linux amigables para los usuarios y la diversidad de distribuciones de Linux, tales como Linux Red Hat, Linux Ubuntu y muchas más— las que han impedido que se popularice el uso de Linux en las computadoras de escritorio. Pero este sistema operativo se ha vuelto muy popular en servidores y sistemas incrustados, como los teléfonos inteligentes basados en Android.

1.6.3 Android

Android —el sistema operativo con mayor crecimiento a la fecha para dispositivos móviles y teléfonos inteligentes— está basado en el kernel de Linux y en Java. Los programadores experimentados de Java no tienen problemas para entrar y participar en el desarrollo de aplicaciones para Android. Un beneficio de desarrollar este tipo de aplicaciones es el grado de apertura de la plataforma. El sistema operativo es gratui
to y de código fuente abierto.

El sistema operativo Android fue desarrollado por Android, Inc., compañía que adquirió Google en 2005. En 2007 se formó la Alianza para los dispositivos móviles abiertos **TM** (OHA) —que ahora cuenta con 87 compañías miembro a nivel mundial (http://www.openhandsetalliance.com/oha_members.html)—, para continuar con el desarrollo, mantenimiento y evolución de Android, impulsando la innova ción en la tecnología móvil y mejorando la experiencia del usuario, reduciendo al mismo tiempo los costos. Al mes de abril de 2013, se activaban a *diario* más de 1.5 millones de dispositivos con Android (teléfonos inteligentes, tabletas, etc.).[10] Para octubre de 2013, un informe de Strategy Analytics mostró que Android tenía el 81.3% de la participación global en el mercado de *teléfonos inteligentes*, en comparación con el 13.4% de Apple, el 4.1% de Microsoft y el 1% de BlackBerry.[11] Ahora los dispositivos Android incluyen teléfonos inteligentes, tabletas, lectores electrónicos, robots, motores de jet, satélites de la NASA, consolas de juegos, refrigeradores, televisiones, cámaras, dispositivos para el cuidado de la salud, relojes inteligentes, sistemas de infoentretenimiento en vehículos (para controlar el radio, GPS, llamadas telefónicas, termos tato, etc.) y más.[12]

9 http://developers.facebook.com/opensource.
10 http://www.technobuffalo.com/2013/04/16/google-daily-android-activations-1-5-million. 11 http://www.cnet.com/news/android-shipments-exceed-1-billion-for-first-time-in-2014/. 12 http://www.businessweek.com/articles/2013-05-29/behind-the-internet-of-things-is-android and-its-everywhere.

Los teléfonos inteligentes Android incluyen la funcionalidad de un teléfono móvil, un cliente de In ternet (para navegar en Web y comunicarse a través de Internet), un reproductor de MP3, una consola de juegos, una cámara digital y demás. Estos dispositivos portátiles cuentan con *pantallas multitáctiles* a todo color, que le permiten controlar el dispositivo con *ademanes* en los que se requieren uno o varios toques simultáneos. Puede descargar aplicaciones de manera directa a su dispositivo Android, a través de Google Play y de otros mercados de aplicaciones. Al momento de escribir este libro había más de un millón de aplicaciones en **Google Play**; esta cifra aumenta con rapidez.[13]

En nuestro libro de texto *Android How to Program*, *segunda edición*, y en nuestro libro profesional, *Android for Programmers: An App-Driven Approach, segunda edición*, presentamos una introducción al de sarrollo de aplicaciones para Android. Después de que aprenda Java, descubrirá que no es tan complicado empezar a desarrollar y ejecutar aplicaciones Android. Puede colocar sus aplicaciones en Google Play (play.google.com) y, si se vuelven populares, tal vez hasta pueda iniciar su propio negocio. Sólo recuerde:

Facebook, Microsoft y Dell se iniciaron desde un dormitorio.

1.7 Lenguajes de programación

En esta sección comentaremos brevemente algunos lenguajes de programación populares (figura 1.5). En la siguiente sección veremos una introducción a Java.

Fortran Fortran (FORmula TRANslator, traductor de fórmulas) fue desarrollado por IBM Corpora tion a mediados de la década de 1950 para utilizarse en aplicaciones científicas y de ingeniería que requerían cálculos matemáticos complejos. Aún se utiliza mucho y sus versiones más recientes soportan la programación orientada a objetos.

COBOL COBOL (COmmon Business Oriented Language, lenguaje común orientado a negocios) fue desarrollado a finales de la década de 1950 por fabricantes de computadoras, el gobierno estadounidense y usuarios de computadoras de la industria, con base en un lenguaje desarrollado por Grace Hopper, un oficial de la Marina de Estados Unidos y científico informático, que también abogó por la estandarización internacional de los lenguajes de programación. COBOL aún se utiliza mucho en aplicaciones comerciales que requieren de una manipulación precisa y eficiente de grandes volúmenes de datos. Su versión más reciente soporta la programación orientada a objetos.

Pascal Las actividades de investigación en la década de 1960 dieron como resultado la *programación estructurada*: un método disciplinado para escribir programas que sean más claros, fáciles de probar y depurar, y más fáciles de modificar que los programas extensos producidos con técnicas anteriores. Un resultado de esta investigación fue el desarrollo del lenguaje de programación Pascal en 1971, el cual se diseñó para la enseñanza de la programación estructurada y fue popular en los cursos universitarios durante varias décadas.

Ada Ada, un lenguaje basado en Pascal, se desarrolló bajo el patrocinio del Departamento de Defensa (DOD) de Estados Unidos durante la década de 1970 y a principios de la década de 1980. El DOD quería un solo lenguaje que pudiera satisfacer la mayoría de sus necesidades. El nombre de este lenguaje es en honor de Lady Ada Lovelace, hija del poeta Lord Byron. A ella se le atribuye el haber escrito el primer programa para computadoras en el mundo, a principios de la década de 1800 (para la Máquina Analítica, un dispositivo de cómputo mecánico diseñado por Charles Babbage). Ada también soporta la programación orientada a objetos.

Fig. 1.5 Otros lenguajes de programación (parte 1 de 3).

13 http://en.wikipedia.org/wiki/Google_Play.

Basic Basic se desarrolló en la década de 1960 en el Dartmouth College, para familiarizar a los principiantes con las técnicas de programación. Muchas de sus versiones más recientes son orientadas a objetos.

C C fue desarrollado a principios de la década de 1970 por Dennis Ritchie en los Laboratorios Bell. En un principio se hizo muy popular como el lenguaje de desarrollo del sistema operativo UNIX. En la actualidad, la mayoría del código para los sistemas operativos de propósito general se escribe en C o C++.

C++ C++, una extensión de C, fue desarrollado por Bjarne Stroustrup a principios de la década de 1980 en los Laboratorios Bell. C++ proporciona varias características que "pulen" al lenguaje C, pero lo más importante es que proporciona las capacidades de una programación orientada a objetos.

Objective-C Objective-C es un lenguaje orientado a objetos basado en C. Se desarrolló a principios de la década de 1980 y después fue adquirido por la empresa Next, que a su vez fue adquirida por Apple. Se ha convertido en el lenguaje de programación clave para el sistema operativo OS X y todos los dispositivos operados por el iOS (como los dispositivos iPod, iPhone e iPad).

Visual Basic El lenguaje Visual Basic de Microsoft se introdujo a principios de la década de 1990 para simplificar el desarrollo de aplicaciones para Microsoft Windows. Sus versiones más recientes soportan la programación orientada a objetos.

Visual C# Los tres principales lenguajes de programación orientados a objetos de Microsoft son Visual Basic (basado en el Basic original), Visual C++ (basado en C++) y Visual C# (basado en C++ y Java; desarrollado para integrar Internet y Web en las aplicaciones de compu tadora).

PHP PHP es un lenguaje orientado a objetos de secuencias de comandos y código fuente abierto, el cual recibe soporte por medio de una comunidad de usuarios y desarrolladores; se utiliza en millones de sitios Web. PHP es independiente de la plataforma —existen implementacio nes para todos los principales sistemas operativos UNIX, Linux, Mac y Windows. PHP también soporta muchas bases de datos, incluyendo la popular MySQL de código fuente abierto.

Perl Perl (Lenguaje práctico de extracción y reporte), uno de los lenguajes de secuencia de comandos orientados a objetos más utilizados para la programación Web, fue desarrollado en 1987 por Larry Wall. Cuenta con capacidades complejas de procesamiento de textos.

Python Python, otro lenguaje orientado a objetos de secuencias de comandos, se liberó al público en 1991. Fue desarrollado por Guido van Rossum del Instituto Nacional de Investigación para las Matemáticas y Ciencias Computacionales en Amsterdam (CWI); la mayor parte de Python se basa en Modula-3, un lenguaje de programación de sistemas. Python es "extensible", lo que significa que puede extenderse a través de clases e interfaces de programación.

JavaScript JavaScript es el lenguaje de secuencias de comandos más utilizado en el mundo. Su principal uso es para agregar comportamiento dinámico a las páginas Web; por ejemplo, animaciones e interactividad mejorada con el usuario. Se incluye en todos los principales navegadores Web.

Fig. 1.5 Otros lenguajes de programación (parte 2 de 3).

Ruby on Rails Ruby, que se creó a mediados de la década de 1990, es un lenguaje de programación orientado a objetos de código fuente abierto, con una sintaxis simple que es similar a Python. Ruby on Rails combina el lenguaje de secuencias de comandos Ruby con el marco de trabajo de aplicaciones Web Rails, desarrollado por 37Signals. Su libro, *Getting Real* (gettingreal.37signals.com/toc.php), es una lectura obligatoria para los desarrolladores Web. Muchos desarrolladores de Ruby on Rails han reportado ganancias de productividad superiores a las de otros lenguajes, al desarrollar aplicaciones Web que trabajan de manera intensiva con bases de datos.

Fig. 1.5 Otros lenguajes de programación (parte 3 de 3).

1.8 Java

La contribución más importante a la fecha de la revolución del microprocesador es que hizo posible el desarrollo de las computadoras personales. Los microprocesadores han tenido un profundo impacto en los dispositivos electrónicos inteligentes para uso doméstico. Al reconocer esto, Sun Microsystems patrocinó en 1991 un proyecto interno de investigación corporativa dirigido por James Gosling, que dio como resultado un lenguaje de programación orientado a objetos y basado en C++, al que Sun llamó Java.

Un objetivo clave de Java es poder escribir programas que se ejecuten en una gran variedad de sistemas computacionales y dispositivos controlados por computadora. A esto se le conoce algunas veces como "escribir una vez, ejecutar en cualquier parte".

La popularidad del servicio Web explotó en 1993; en ese entonces Sun vio el potencial de usar Java para agregar *contenido dinámico*, como interactividad y animaciones, a las páginas Web. Java atrajo la atención de la comunidad de negocios debido al fenomenal interés en el servicio Web. En la actualidad, Java se utiliza para desarrollar aplicaciones empresariales a gran escala, para mejorar la funcionalidad de los servidores Web (las computadoras que proporcionan el contenido que vemos en nuestros navegadores Web), para proporcionar aplicaciones para los dispositivos de uso doméstico (como teléfonos celulares, teléfonos inteligentes, receptores de televisión por Internet y mucho más) y para muchos otros propósitos. Java también es el lenguaje clave para desarrollar aplicaciones para teléfonos inteligentes y tabletas de Android. En 2010, Oracle adquirió Sun Microsystems.

Bibliotecas de clases de Java

Usted puede crear cada clase y método que necesite para formar sus programas de Java. Sin embargo, la mayoría de los programadores en Java aprovechan las ricas colecciones de clases y métodos existentes en las **bibliotecas de clases de Java**, que también se conocen como **API (Interfaces de programación de apli caciones) de Java**.

Tip de rendimiento 1.1

Si utiliza las clases y métodos de las API de Java en vez de escribir sus propias versiones, puede mejorar el rendimiento de sus programas, ya que estas clases y métodos están escritos de manera cuidadosa para funcionar con efi ciencia. Esta técnica también reduce el tiempo de desarrollo de los programas.

1.9 Un típico entorno de desarrollo en Java

Ahora explicaremos los pasos típicos utilizados para crear y ejecutar una aplicación en Java. Por lo general hay cinco fases: edición, compilación, carga, verificación y ejecución. Hablaremos sobre estos conceptos

en el contexto del Kit de desarrollo de Java (JDK) SE 8. Lea la sección *Antes de empezar de este libro para obtener información acerca de cómo descargar e instalar el JDK en Windows, Linux y OS X.*

Fase 1: Creación de un programa
La fase 1 consiste en editar un archivo con un *programa de edición*, conocido comúnmente como *editor* (figura 1.6). A través del editor, usted escribe un programa en Java (a lo cual se le conoce por lo general como **código fuente**), realiza las correcciones necesarias y guarda el programa en un dispositivo de alma cenamiento secundario, como su disco duro. Los archivos de código fuente de Java reciben un nombre que termina con la **extensión .java**, lo que indica que éste contiene código fuente en Java.

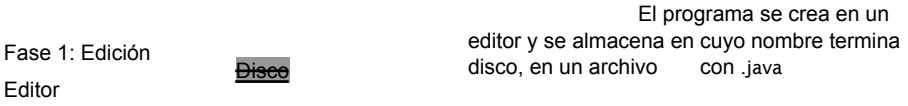


Fig. 1.6 Entorno de desarrollo típico de Java: fase de edición.

Dos de los editores muy utilizados en sistemas Linux son vi y emacs. Windows cuenta con el **Bloc de Notas**. OS X ofrece **TextEdit**. También hay muchos editores de freeware y shareware disponibles en línea, como Notepad++ (notepad-plus-plus.org), Edit-Plus (www.editplus.com), TextPad (www.textpad.com) y jEdit (www.jedit.org).

Los **entornos de desarrollo integrados (IDE)** proporcionan herramientas que dan soporte al proce so de desarrollo del software, entre las que se incluyen editores, depuradores para localizar **errores lógicos** (errores que provocan que los programas se ejecuten en forma incorrecta) y más. Hay muchos IDE de Java populares, como

- Eclipse (www.eclipse.org)
- NetBeans (www.netbeans.org)
- IntelliJ IDEA (www.jetbrains.com)

En el sitio Web del libro en

www.deitel.com/books/jhtp10

proporcionamos videos Dive-Into® que le muestran cómo ejecutar las aplicaciones de Java de este libro y cómo desarrollar nuevas aplicaciones de Java con Eclipse, NetBeans e IntelliJ IDEA.

Fase 2: Compilación de un programa en Java para convertirlo en códigos de bytes En la fase 2, el programador utiliza el comando **javac** (el **compilador de Java**) para **compilar** un programa (figura 1.7). Por ejemplo, para compilar un programa llamado Bienvenido.java, escriba

```
javac Bienvenido.java
```

en la ventana de comandos de su sistema (es decir, el Símbolo del sistema de Windows, o la aplicación Terminal en OS X) o en un shell de Linux (que también se conoce como Terminal en algunas versiones de Linux). Si el programa se compila, el compilador produce un archivo **.class** llamado Bienvenido.class que contiene la versión compilada del programa. Por lo general los IDE proveen un elemento de menú,
como Build (Generar) o Make (Crear), que invoca el comando javac por usted. Si el compilador detecta errores, tendrá que ir a la fase 1 y corregirlos. En el capítulo 2 hablaremos más sobre los tipos de errores que el compilador puede detectar.

Fig. 1.7 Entorno de desarrollo típico de Java: fase de compilación.

El compilador de Java traduce el código fuente de Java en **códigos de bytes** que representan las tareas a ejecutar en la fase de ejecución (fase 5). La **Máquina Virtual de Java (JVM),** que forma parte del JDK y es la base de la plataforma Java, ejecuta los códigos de bytes. Una **máquina virtual (VM)** es una aplicación de software que simula a una computadora, pero oculta el sistema operativo y el hardware subyacentes de los programas que interactúan con ésta. Si se implementa la misma VM en muchas plataformas compu
tacionales, las aplicaciones escritas para ese tipo de VM se podrán utilizar en todas esas plataformas. La JVM es una de las máquinas virtuales más utilizadas en la actualidad. La plataforma .NET de Microsoft utiliza una arquitectura de máquina virtual similar.

A diferencia de las instrucciones de lenguaje máquina, que *dependen de la plataforma* (es decir, depen den del hardware de una computadora específica), los códigos de bytes son instrucciones *independientes de la plataforma*. Por lo tanto, los códigos de bytes de Java son **portables**; es decir, se pueden ejecutar los mismos códigos de bytes en cualquier plataforma que contenga una JVM que incluya la versión de Java en la que se compilaron los códigos de bytes sin necesidad de volver a compilar el código fuente. La JVM se invoca mediante el comando **java**. Por ejemplo, para ejecutar una aplicación en Java llamada Bienvenido, debe escribir el comando

```
java Bienvenido
```

en una ventana de comandos para invocar la JVM, que a su vez inicia los pasos necesarios para ejecutar la aplicación. Esto comienza la fase 3. Por lo general los IDE proporcionan un elemento de menú, como **Run** (**Ejecutar**), que invoca el comando java por usted.

Fase 3: Carga de un programa en memoria

En la fase 3, la JVM coloca el programa en memoria para ejecutarlo; a esto se le conoce como **carga** (figura 1.8). El **cargador de clases** de la JVM toma los archivos .class que contienen los códigos de bytes del programa y los transfiere a la memoria principal. El cargador de clases también carga cualquiera de los archivos .class que su programa utilice, y que sean proporcionados por Java. Puede cargar los archi vos .class desde un disco en su sistema o a través de una red (como la de su universidad local o la red de la empresa, o incluso desde Internet).

Fig. 1.8 Entorno de desarrollo típico de Java: fase de carga.

Fase 4: Verificación del código de bytes

En la fase 4, a medida que se cargan las clases, el **verificador de códigos de bytes** examina sus códigos de bytes para asegurar que sean válidos y que no violen las restricciones de seguridad de Java (figura 1.9). Java implementa una estrecha seguridad para asegurar que los programas en Java que llegan a través de la red no dañen sus archivos o su sistema (como podrían hacerlo los virus de computadora y los gusanos).

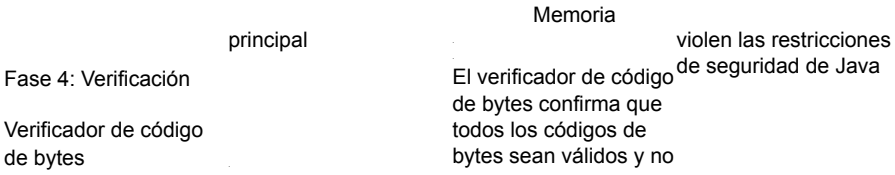


Fig. 1.9 Entorno de desarrollo típico de Java: fase de verificación.

Fase 5: Ejecución

En la fase 5, la JVM **ejecuta** los códigos de bytes del programa, realizando así las acciones especificadas por el mismo (figura 1.10). En las primeras versiones de Java, la JVM era tan sólo un *intérprete* de códigos de bytes de Java. Esto hacía que la mayoría de los programas se ejecutaran con lentitud, ya que la JVM tenía que inter pretar y ejecutar un código de bytes a la vez. Algunas arquitecturas de computadoras modernas pueden ejecu tar varias instrucciones en paralelo. Por lo general, las JVM actuales ejecutan códigos de bytes mediante una combinación de la interpretación y la denominada **compilación justo a tiempo (JIT)**. En este proceso, la JVM analiza los códigos de bytes a medida que se interpretan, en busca de *puntos activos*(partes de los códigos de bytes que se ejecutan con frecuencia). Para estas partes, un **compilador justo a tiempo (JIT)**, como el **compilador HotSpot™ de Java** de Oracle, traduce los códigos de bytes al lenguaje máquina correspondiente de la computadora. Cuando la JVM vuelve a encontrar estas partes compiladas, se ejecuta el código en len guaje máquina, que es más rápido. Por ende, los programas en Java en realidad pasan por *dos* fases de compi lación: una en la cual el código fuente se traduce a código de bytes (para tener portabilidad a través de las JVM en distintas plataformas computacionales) y otra en la que, durante la ejecución los *códigos de bytes* se tradu cen en *lenguaje máquina* para la computadora actual en la que se ejecuta el programa.

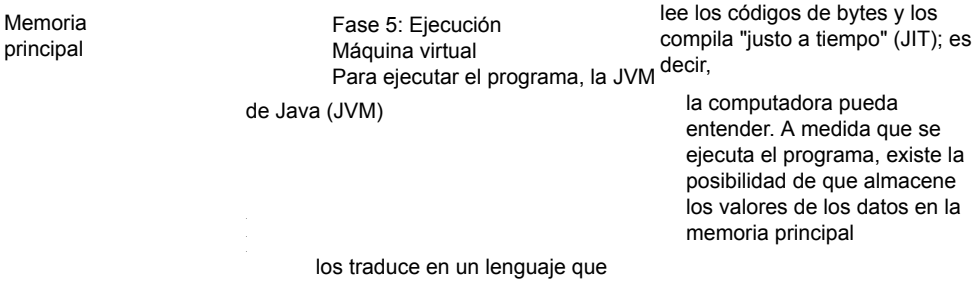


Fig. 1.10 Entorno de desarrollo típico de Java: fase de ejecución.

Problemas que pueden ocurrir en tiempo de ejecución

Es probable que los programas no funcionen la primera vez. Cada una de las fases anteriores puede fallar, debido a diversos errores que describiremos en este libro. Por ejemplo, un programa en ejecución podría

intentar una división entre cero (una operación ilegal para la aritmética con números enteros en Java). Esto haría que el programa de Java mostrara un mensaje de error. Si esto ocurre, tendría que regresar a la fase de edición, hacer las correcciones necesarias y proseguir de nuevo con las fases restantes, para deter minar que las correcciones hayan resuelto el o los problemas [*nota:* la mayoría de los programas en Java reciben o producen datos. Cuando decimos que un programa muestra un mensaje, por lo general quere mos decir que muestra ese mensaje en la pantalla de su computadora. Los mensajes y otros datos pueden enviarse a otros dispositivos, como los discos y las impresoras, o incluso a una red para transmitirlos a otras computadoras].

Error común de programación 1.1

Los errores, como la división entre cero, ocurren a medida que se ejecuta un programa, de manera que a estos errores se les llama ***errores en tiempo de ejecución****. Los* ***errores fatales en tiempo de ejecución*** *hacen que los programas terminen de inmediato, sin haber realizado bien su trabajo. Los* ***errores no fatales en tiempo de ejecución*** *permiten a los programas ejecutarse hasta terminar su trabajo, lo que a menudo produce resultados incorrectos.*

1.10 Prueba de una aplicación en Java

En esta sección ejecutará su primera aplicación en Java e interactuará con ella. La aplicación Painter, que creará en el transcurso de varios ejercicios, le permite arrastrar el ratón para "dibujar". Los ele mentos y la funcionalidad que podemos ver en esta aplicación son típicos de lo que aprenderá a pro gramar en este libro. Mediante el uso de la interfaz gráficos de usuario (GUI) de Painter, usted puede controlar el color de dibujo, la forma a dibujar (línea, rectángulo u óvalo) y si la forma se debe llenar o no con un color. También puede deshacer la última forma que agregó al dibujo o borrarlo todo [*nota:* utilizamos fuentes para diferenciar las diversas características. Nuestra convención es enfatizar las características de la pantalla como los títulos y menús (por ejemplo, el menú Archivo) en una fuente Helvetica sans-serif en negritas, y enfatizar los elementos que no son de la pantalla, como los nombres de archivo, código del programa o los datos de entrada (como NombrePrograma.java) en una fuente Lucida sans-serif].

Los pasos en esta sección le muestran cómo ejecutar la aplicación Painter desde una ventana Símbo lo del sistema (Windows), Terminal (OS X) o shell (Linux) de su sistema. A lo largo del libro nos referi remos a estas ventanas simplemente como *ventanas de comandos*. Realice los siguientes pasos para usar la aplicación Painter para dibujar una cara sonriente:

1. ***Revise su configuración.*** Lea la sección Antes de empezar este libro para confirmar que haya instalado Java de manera apropiada en su computadora, que haya copiado los ejemplos del libro en su disco duro y que sepa cómo abrir una ventana de comandos en su sistema.

2. ***Cambie al directorio de la aplicación completa.*** Abra una ventana de comandos y use el comando cd para cambiar al directorio (también conocido como *carpeta*) de la aplicación Painter. Vamos a suponer que los ejemplos del libro se encuentran en C:\ejemplos en Windows o en la carpeta Documents/ejemplos en Linux o en OS X. En Windows escriba cd C:\ejemplos\cap01\painter y después oprima *Intro*. En Linux u OS X escriba cd ~/Documents/ejemplos/cap01/painter y después oprima *Intro*.

3. ***Ejecute la aplicación Painter.*** Recuerde que el comando java, seguido del nombre del archivo .class de la aplicación (en este caso, Painter), ejecuta la aplicación. Escriba el comando java Painter y oprima *Intro* para ejecutar la aplicación. La figura 1.11 muestra la aplicación en ejecu ción en Windows, Linux y OS X, respectivamente; redujimos el tamaño de las ventanas para

ahorrar espacio.

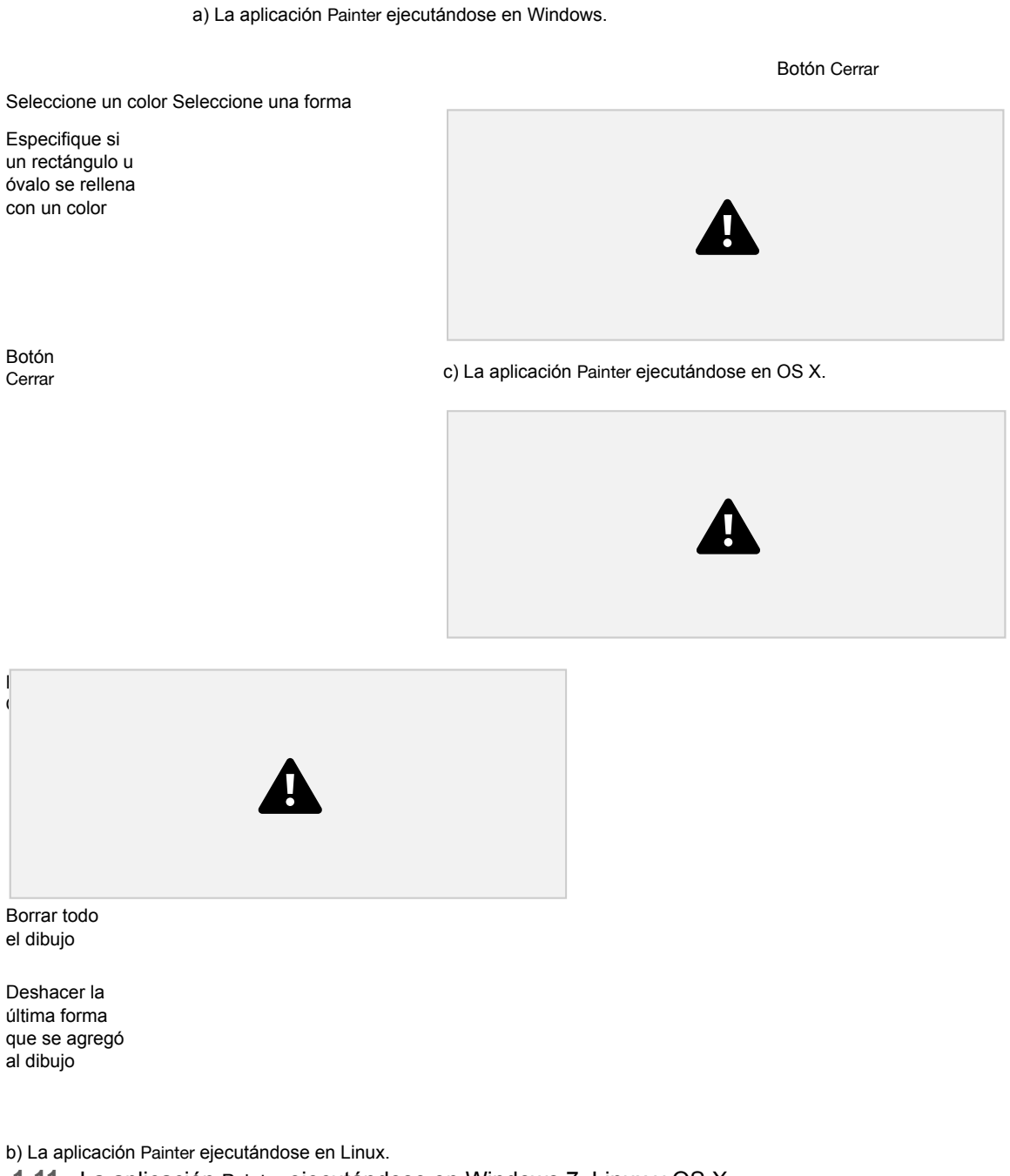


Fig. 1.11 La aplicación Painter ejecutándose en Windows 7, Linux y OS X.

[*Nota:* los comandos en Java son *sensibles a mayúsculas/minúsculas*. Es importante escribir el nombre de esta aplicación como `Painter` con `P` mayúscula. De lo contrario, la aplicación *no* se ejecutará. Si se especifica la extensión `.class` al usar el comando `java` se produce un error. Ademas, si recibe el mensaje de error "`Exception in thread "main" java.lang.NoClassDefFoundError: Painter`", entonces su

sistema tiene un problema con CLASSPATH. Consulte la sección Antes de empezar del libro para obtener instrucciones sobre cómo corregir este problema].

4. ***Dibuje un óvalo relleno de color amarillo para el rostro.*** Seleccione Yellow (Amarillo) como el color de dibujo, Oval (Óvalo) como la forma y marque la casilla de verificación Filled (Relleno); luego arrastre el ratón para dibujar un óvalo más grande (figura 1.12).

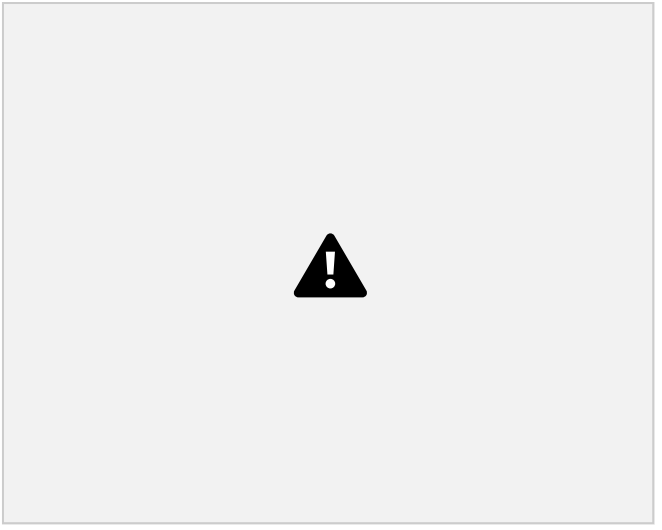


Fig. 1.12 Dibuje un óvalo relleno de color amarillo para el rostro.

5. ***Dibuje los ojos azules.*** Seleccione Blue (Azul) como el color de dibujo y luego dibuje dos óvalos pequeños como los ojos (figura 1.13).

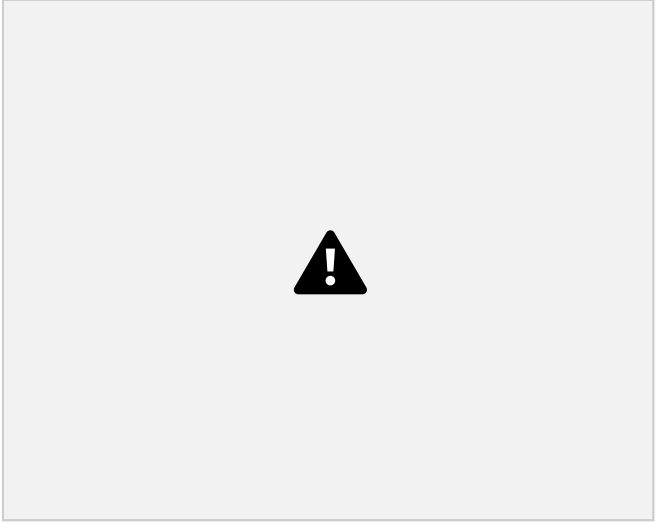


Fig. 1.13 Dibuje los ojos azules.

6. ***Dibuje cejas negras y una nariz.*** Seleccione Black (Negro) como el color de dibujo y Line (Línea) como la forma; después dibuje cejas y una nariz (figura 1.14). Las líneas no tienen relleno, por lo que si se deja la casilla de verificación Filled (Relleno) marcada, esto no tendrá efecto cuando se dibujen las líneas.

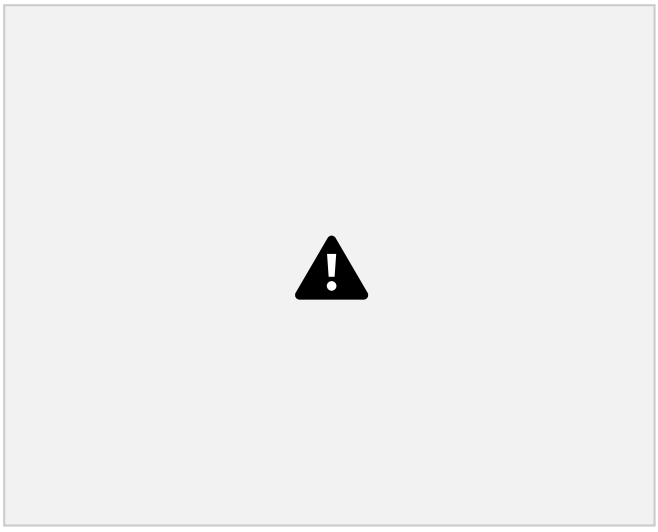


Fig. 1.14 Dibuje cejas negras y una nariz.

7. ***Dibuje una boca color magenta.*** Seleccione Magenta como el color de dibujo y Oval (Óvalo) como la forma; después dibuje una boca (figura 1.15).

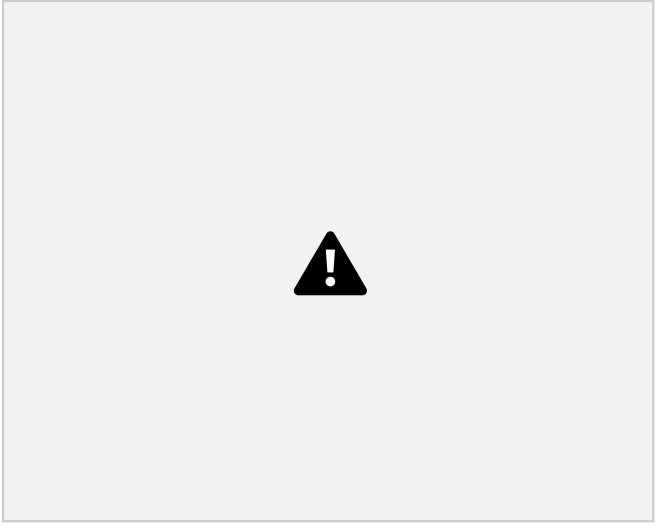


Fig. 1.15 Dibuje una boca color magenta.

8. ***Dibuje un óvalo amarillo en la boca para crear una sonrisa.*** Seleccione Yellow (Amarillo) como el color de dibujo y luego dibuje un óvalo para convertir el óvalo color magenta en una sonrisa (figura 1.16).

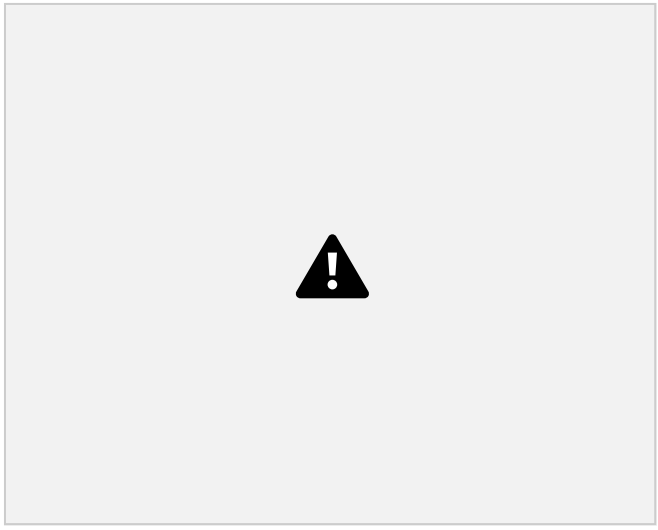


Fig. 1.16 Dibuje un óvalo amarillo en la boca para crear una sonrisa.

9. ***Salga de la aplicación*** *Painter.* Para salir de la aplicación Painter, haga clic en el botón Cerrar (en la esquina superior derecha de la ventana en Windows y en la esquina superior izquierda en Linux y OS X). Al cerrar la ventana terminará la ejecución de la aplicación Painter.

1.11 Internet y World Wide Web

A finales de la década de 1960, la ARPA (Agencia de proyectos avanzados de investigación del Departa mento de Defensa de Estados Unidos) implementó los planes para conectar en red los principales sistemas de cómputo de aproximadamente una docena de universidades e instituciones de investigación financiadas por la ARPA. Las computadoras se iban a conectar con líneas de comunicación que operaban a velocida des en el orden de 50,000 bits por segundo, una tasa impresionante en una época en que la mayoría de las personas (de los pocos que incluso tenían acceso a redes) se conectaban a través de líneas telefónicas a las computadoras a una tasa de 110 bits por segundo. La investigación académica estaba a punto de dar un gran paso hacia delante. La ARPA procedió a implementar lo que rápidamente se conoció como ARPANET, precursora de la red Internet actual. Las velocidades de Internet más rápidas de la actualidad están en el orden de miles de millones de bits por segundo y pronto estarán disponibles las velocidades de billones de bits por segundo.

Las cosas funcionaron de manera distinta al plan original. Aunque la ARPANET permitió que los investigadores conectaran en red sus computadoras, su principal beneficio demostró ser la capacidad de comunicarse con rapidez y facilidad a través de lo que se denominó correo electrónico (e-mail). Esto es cierto incluso en la red Internet actual, en donde el correo electrónico, la mensajería instantánea, la trans
ferencia de archivos y los medios sociales como Facebook y Twitter permiten que miles de millones de personas en todo el mundo se comuniquen de una manera rápida y sencilla.

El protocolo (conjunto de reglas) para comunicarse a través de la ARPANET se denominó **Protocolo de control de transmisión (TCP)**. Este protocolo se aseguraba de que los mensajes, que consistían en piezas numeradas en forma secuencial conocidas como *paquetes*, se enrutaran correctamente del emisor al receptor, que llegaran intactos y se ensamblaran en el orden correcto.

1.11.1 Internet: una red de redes

En paralelo con la evolución de Internet en sus primeras etapas, las organizaciones de todo el mundo estaban implementando sus propias redes para comunicarse tanto dentro de la organización como entre varias organizaciones. En esta época apareció una enorme variedad de hardware y software de red. Un desafío era permitir que esas distintas redes se comunicaran entre sí. La ARPA logró esto al desarrollar el Protocolo Internet (IP), que creó una verdadera "red de redes", la arquitectura actual de Internet. Al
conjunto combinado de protocolos se le conoce ahora como **TCP/IP**.

Las empresas descubrieron rápidamente que al usar Internet podían mejorar sus operaciones ade más de ofrecer nuevos y mejores servicios a sus clientes. Las compañías comenzaron a invertir grandes cantidades de dinero para desarrollar y mejorar su presencia en Internet. Esto generó una feroz compe tencia entre las operadoras de comunicaciones y los proveedores de hardware y software para satisfacer la mayor demanda de infraestructura. Como resultado, el **ancho de banda** (la capacidad que tiene las líneas de comunicación para transportar información) en Internet se incrementó de manera considera ble, mientras que los costos del hardware se desplomaron.

1.11.2 World Wide Web: cómo facilitar el uso de Internet

World Wide Web (conocida simplemente como "Web") es una colección de hardware y software asociado con Internet que permite a los usuarios de computadora localizar y ver documentos basados en multimedia (documentos con diversas combinaciones de texto, gráficos, animaciones, sonido y videos) sobre casi cualquier tema. La introducción de Web fue un evento relativamente reciente. En 1989, Tim Berners-Lee de CERN (la Organización europea de investigación nuclear) comenzó a desarrollar una tecnología para compartir información a través de documentos de texto con "hiper vínculos". Berners-Lee llamó a su invención el **Lenguaje de marcado de hipertexto (HTML)**. Tam bién escribió protocolos de comunicaciones como el **Protocolo de transferencia de hipertexto (HTTP)** para formar la espina dorsal de su nuevo sistema de información de hipertexto, al cual de nominó World Wide Web.

En 1994, Berners-Lee fundó una organización conocida como **Consorcio World Wide Web** (**W3C**, www.w3.org), dedicada al desarrollo de tecnologías Web. Uno de los principales objetivos del W3C es que Web sea accesible en forma universal para todos, sin importar las discapacidades, el idioma o la cultura. En este libro usted usará Java para crear aplicaciones basadas en Web.

1.11.3 Servicios Web y *mashups*

En el capítulo 32 incluimos un tratamiento detallado sobre los servicios Web (figura 1.17). La meto dología de desarrollo de aplicaciones conocida como *mashups* le permite desarrollar rápidamente po derosas y asombrosas aplicaciones de software, al combinar servicios Web complementarios (a menudo gratuitos) y otras fuentes de información. Uno de los primeros mashups combinaba los listados de bienes raíces proporcionados por www.craigslist.org con las capacidades de generación de mapas de *Google Maps* para ofrecer mapas que mostraran las ubicaciones de las casas en venta o renta den tro de cierta área.

Google Maps Servicios de mapas

Twitter Microblogs

Fig. 1.17 Algunos servicios Web populares (http://www.programmableweb.com/ category/all/apis) (parte 1 de 2).

YouTube Búsquedas de videos Facebook Redes sociales
Instagram Compartir fotografías
Foursquare Registros de visitas (check-ins) móviles
LinkedIn Redes sociales para negocios
Groupon Comercio social
Netflix Renta de películas
eBay Subastas en Internet
Wikipedia Enciclopedia colaborativa
PayPal Pagos
Last.fm Radio por Internet
Amazon eCommerce Compra de libros y otros artículos
Salesforce.com Administración de la relación con los clientes (CRM)
Skype Telefonía por Internet
Microsoft Bing Búsqueda
Flickr Compartir fotografías
Zillow Precios de bienes raíces
Yahoo Search Búsqueda
WeatherBug Clima

Fig. 1.17 Algunos servicios Web populares (http://www.programmableweb.com/ category/all/apis) (parte 2 de 2).

1.11.4 Ajax

Ajax ayuda a las aplicaciones basadas en Internet a funcionar como las aplicaciones de escritorio; una tarea difícil, dado que dichas aplicaciones sufren de retrasos en la transmisión, a medida que los datos se intercambian entre su computadora y las computadoras servidores en Internet. Mediante el uso de Ajax, las aplicaciones como Google Maps han logrado un desempeño excelente, además de que se su apariencia visual se asemeja a las aplicaciones de escritorio. Aunque en este libro no hablaremos sobre la programación "pura" con Ajax (que es bastante compleja), en el capítulo 31 le mostraremos cómo crear aplicaciones habilitadas para Ajax mediante el uso de los componentes de JavaServer Faces (JSF) habilitados para Ajax.

1.11.5 Internet de las cosas

Internet ya no sólo es una red de computadoras: es una **Internet de las cosas**. Una *cosa* es cualquier objeto con una dirección IP y la habilidad de enviar datos de manera automática a través de una red (por ejemplo, un auto con un transpondedor para pagar peaje, un monitor cardiaco implantado en un hu mano, un medidor inteligente que reporta el uso de energía, aplicaciones móviles que pueden rastrear su movimiento y ubicación, y termostatos inteligentes que ajustan las temperaturas del cuarto con base en los pronósticos del clima y la actividad en el hogar). En el capítulo 28 en línea usted usará direcciones IP para crear aplicaciones en red.

1.12 Tecnologías de software

La figura 1.18 muestra una lista de palabras de moda que escuchará en la comunidad de desarrollo de software. Creamos Centros de recursos sobre la mayoría de estos temas, y hay muchos por venir.

Desarrollo ágil de software
El **desarrollo ágil de software** es un conjunto de metodologías que tratan de implementar software con más rapidez y mediante el uso de menos recursos. Visite los sitios de Agile Alliance (www.agilealliance.org) y Agile Manifesto (www.agilemanifesto.org). También puede visitar el sitio en español www.agile-spain.com.

Refactorización La **refactorización** implica reformular programas para hacerlos más claros y fáciles de mantener, al tiempo que se conserva su funcionalidad e integridad. Es muy utilizado con las metodologías de desarrollo ágil. Muchos IDE contienen *herramientas de refactorización* integradas para realizar de manera automática la mayor parte del proceso de refactorización.

Patrones de diseño
Los **patrones de diseño** son arquitecturas comprobadas para construir software orientado a objetos flexible y que pueda mantenerse. El campo de los patrones de diseño trata de enumerar esos patrones recurrentes, y de alentar a los diseñado res de software para que los *reutilicen* y puedan desarrollar un software de mejor calidad con menos tiempo, dinero y esfuerzo. En el apéndice N analizaremos los patrones de diseño de Java.

LAMP **LAMP** es un acrónimo para las tecnologías de código fuente abierto que muchos desarrolladores usan en la creación de aplicaciones Web: se refiere a *Linux*, *Apache*, *MySQL* y *PHP* (o *Perl*, o *Python*; otros dos lenguajes de secuencias de comandos). MySQL es un sistema manejador de base de datos de código abierto. PHP es el lenguaje servidor de "secuencias de comandos" de código fuente abierto más popular para el desarrollo de aplicaciones Web. Apache es el software servidor Web más popular. El equivalente para el desarrollo en Windows es WAMP: *Windows*, Apache, MySQL y PHP.

Software como un servicio (SaaS)
A medida que aparecen nuevas versiones, debe actualizar el software, lo cual genera con frecuencia un costo considerable en tiempo y dinero. Este proceso puede ser incómodo para las organizaciones con decenas de miles de sistemas, a los que se debe dar mantenimiento en una diversa selección de equipo de cómputo. En el **Software como un servicio (SaaS)**, el software se ejecuta en servidores ubicados en cualquier parte de Internet. Cuando se actualizan esos servidores, los clientes en todo el mundo ven las nuevas capacidades; no se requiere una instala ción local. Podemos acceder al servicio a través de un navegador. Los navegadores son bastante portables, por lo que podemos ver las mismas aplicaciones en una amplia variedad de computadoras desde cualquier parte del mundo. Salesforce. com, Google, Microsoft Office Live y Windows Live ofrecen SaaS.

Plataforma como un servicio (PaaS)
Por lo general, el software siempre se ha visto como un producto; la mayoría del software aún se ofrece de esta forma. Si desea ejecutar una aplicación, tiene que comprarla a un distribuidor de software: a menudo un CD, DVD o descarga Web. Después instala ese software en la computadora y lo ejecuta cuando sea necesario.
La **Plataforma como un servicio (PaaS)** provee una plataforma de cómputo para desarrollar y ejecutar aplicaciones como un servicio a través de Web, en vez de instalar las herramientas en su computadora. Algunos

proveedores de PaaS son: Google App Engine, Amazon EC2 y Windows Azure™.

Fig. 1.18 Tecnologías de software (parte 1 de 2).

Computación en la nube	almacenados en su computadora de escritorio, notebook o dispositivo móvil. Esto le permite aumentar o disminuir los recursos de cómputo para satisfacer sus necesidades en cualquier momento dado, lo cual es más efectivo en costo que comprar hardware para ofrecer suficiente almacenamiento y poder de procesamiento para satisfacer las demandas pico ocasionales. La computación en la nube también ahorra dinero al transferir al proveedor de servicios la carga de administrar estas aplicaciones.
Kit de desarrollo de software (SDK)	Los **Kits de desarrollo de software (SDK)** incluyen tanto las herramientas como la documentación que utilizan los desarrolladores para programar aplicaciones. Por ejemplo, usted usará el Kit de desarrollo de Java (JDK) para crear y ejecutar aplicaciones de Java.

SaaS y PaaS son ejemplos de **computación en la nube**. Puede usar el software y los datos almacenados en la "nube" (es decir, se accede a éstos mediante compu tadoras remotas o servidores, a través de Internet y están disponibles bajo demanda) en vez de tenerlos

Fig. 1.18 Tecnologías de software (parte 2 de 2).

El software es complejo. Las aplicaciones de software extensas que se usan en el mundo real pueden tardar muchos meses, o incluso años, en diseñarse e implementarse. Cuando hay grandes productos de software en desarrollo, por lo general se ponen a disponibilidad de las comunidades de usuarios como una serie de versiones, cada una más completa y pulida que la anterior (figura 1.19).

Alfa	El software *alfa* es la primera versión de un producto de software cuyo desarrollo aún se encuentra activo. Por lo general las versiones alfa tienen muchos errores, están incompletas y son inestables; además se liberan a un pequeño número de desarrolladores para que evalúen las nuevas características, para obtener retroalimentación lo más pronto posible, etcétera.
Beta	Las versiones *beta* se liberan a un número mayor de desarrolladores en una etapa posterior del proceso de desarrollo, una vez que se ha corregido la mayoría de los errores importantes y las nuevas características están casi completas. El software beta es más estable, pero todavía puede sufrir muchos cambios.
Candidatos para liberación	En general, los *candidatos para liberación* tienen todas sus *características completas*, están (supuestamente) libres de errores y listos para que la comunidad los utilice, con lo cual se logra un entorno de prueba diverso (el software se utiliza en distintos sistemas, con restricciones variables y para muchos fines diferentes).
Liberación de versión final	Cualquier error que aparezca en el candidato para liberación se corrige y, en un momento dado, el producto final se libera al público en general. A menudo, las compañías de software distribuyen actualizaciones incrementales a través de Internet.
Beta permanente	

El software que se desarrolla mediante este método por lo general no tiene números de versión (por ejemplo, la búsqueda de Google o Gmail). Este software se aloja en la *nube* (no se instala en su computadora) y evoluciona de manera constante, de modo que los usuarios siempre dispongan de la versión más reciente.

Fig. 1.19 Terminología de liberación de versiones de productos de software.

1.13 Cómo estar al día con las tecnologías de información

La figura 1.20 muestra una lista de las publicaciones técnicas y comerciales clave que le ayudarán a mante nerse actualizado con la tecnología, las noticias y las tendencias más recientes. También encontrará una lista cada vez más grande de Centros de recursos relacionados con Internet y Web en www.deitel.com/ ResourceCenters.html.

AllThingsD allthingsd.com

Bloomberg BusinessWeek www.businessweek.com

CNET news.cnet.com

Comunications of the ACM cacm.acm.org

Computerworld www.computerworld.com

Engadget www.engadget.com

eWeek www.eweek.com

Fast Company www.fastcompany.com/

Fortune money.cnn.com/magazines/fortune/

GigaOM gigaom.com

Hacker News news.ycombinator.com

IEEE Computer Magazine www.computer.org/portal/web/computingnow/computer

InfoWorld www.infoworld.com

Mashable mashable.com

PCWorld www.pcworld.com

SD Times www.sdtimes.com

Slashdot slashdot.org/

Technology Review technologyreview.com

Techcrunch techcrunch.com

The Next Web thenextweb.com

The Verge www.theverge.com

Wired www.wired.com

Fig. 1.20 Publicaciones técnicas y comerciales.

Ejercicios de autoevaluación

1.1 Complete las siguientes oraciones:

a) Las computadoras procesan datos bajo el control de conjuntos de instrucciones conocidas como .

b) Las unidades lógicas clave de la computadora son , , , , y .

c) Los tres tipos de lenguajes descritos en este capítulo son , y . d) Los programas que

traducen programas en lenguaje de alto nivel a lenguaje máquina se denominan .

e) es un sistema operativo para dispositivos móviles, basado en el kernel de Linux y en Java. f) En general, el software tiene todas sus características completas, está (supuestamente) libre de errores y listo para que la comunidad lo utilice.

g) Al igual que muchos teléfonos inteligentes, el control remoto del Wii utiliza un que permite al dispositivo responder al movimiento.

1.2 Complete las siguientes oraciones sobre el entorno de Java:

a) El comando del JDK ejecuta una aplicación de Java.

b) El comando del JDK compila un programa de Java.

c) Un archivo de código fuente de Java debe terminar con la extensión de archivo . d) Cuando se compila un programa en Java, el archivo producido por el compilador termina con la extensión de archivo .

1.3 Complete las siguientes oraciones (con base en la sección 1.5):

a) Los objetos permiten la práctica de diseño para ; aunque éstos pueden saber cómo comunicar se con los demás objetos a través de interfaces bien definidas, por lo general no se les permite saber cómo están implementados los otros objetos.

b) Los programadores de Java se concentran en crear , que contienen campos y el conjunto de métodos que manipulan a esos campos y proporcionan servicios a los clientes.

c) El proceso de analizar y diseñar un sistema desde un punto de vista orientado a objetos se denomina .

d) Es posible crear una nueva clase de objetos convenientemente mediante , la nueva clase (sub clase) comienza con las características de una clase existente (la superclase), posiblemente personalizándolas y agregando sus propias características únicas.

e) es un lenguaje gráfico que permite a las personas que diseñan sistemas de software utilizar una notación estándar en la industria para representarlos.

f) El tamaño, forma, color y peso de un objeto se consideran de su clase.

Respuestas a los ejercicios de autoevaluación

1.1 a) programas. b) unidad de entrada, unidad de salida, unidad de memoria, unidad central de procesamiento, unidad aritmética y lógica, unidad de almacenamiento secundario. c) lenguajes máquina, lenguajes ensambladores, lenguajes de alto nivel. d) compiladores. e) Android. f) candidato de liberación. g) acelerómetro.

1.2 a) `java`. b) `javac`. c) `.java`. d) `.class`. e) códigos de bytes.

1.3 a) ocultar información. b) clases. c) análisis y diseño orientados a objetos (A/DOO). d) la herencia. e) El Lenguaje Unificado de Modelado (UML). f) atributos.

Ejercicios

1.4 Complete las siguientes oraciones:

a) La unidad lógica de la computadora que recibe información desde el exterior de la misma para que ésta la utilice se llama .

b) Al proceso de indicar a la computadora cómo resolver un problema se llama . c) es un tipo de lenguaje computacional que utiliza abreviaturas del inglés para las instrucciones de lenguaje máquina.

d) es una unidad lógica de la computadora que envía información que ya ha sido procesada a varios dispositivos, de manera que la información pueda utilizarse fuera de la computadora. e) y son unidades lógicas de la computadora que retienen información. f) es una unidad lógica de la computadora que realiza cálculos.

g) es una unidad lógica de la computadora que toma decisiones lógicas.

h) Los lenguajes son los más convenientes para que el programador pueda escribir programas con rapidez y facilidad.

i) Al único lenguaje que una computadora puede entender directamente se le conoce como el de esa computadora.

j) es una unidad lógica de la computadora que coordina las actividades de todas las demás uni dades lógicas.

1.5 Complete las siguientes oraciones:

a) El lenguaje de programación se utiliza ahora para desarrollar aplicaciones empresariales de gran escala, para mejorar la funcionalidad de los servidores Web, para proporcionar aplicaciones para disposi tivos domésticos y muchos otros fines más.

b) En un principio, se hizo muy popular como lenguaje de desarrollo para el sistema operativo UNIX.

c) El (la) se asegura de que los mensajes, que consisten en piezas numeradas en forma secuencial conocidas como bytes, se enruten de manera apropiada del emisor hasta el receptor, que lleguen intactos y se ensamblen en el orden correcto.

d) El lenguaje de programación fue desarrollado por Bjarne Stroustrup a principios de la década de 1980 en los Laboratorios Bell.

1.6 Complete las siguientes oraciones:

a) Por lo general, los programas de Java pasan a través de cinco fases: , , , y .

b) Un proporciona muchas herramientas que dan soporte al proceso de desarrollo de software, como los editores para escribir y editar programas, los depuradores para localizar los errores lógicos en los programas, y muchas otras características más.

c) El comando `java` invoca al , que ejecuta los programas de Java.

d) Una es una aplicación de software que simula una computadora, pero oculta el sistema opera tivo y el hardware subyacentes de los programas que interactúan con ella.

e) El toma los archivos `.class` que contienen los códigos de bytes del programa y los transfiere a la memoria principal.

f) El examina los códigos de bytes para asegurar que sean válidos.

1.7 Explique las dos fases de compilación de los programas de Java.

1.8 Uno de los objetos más comunes en el mundo es el reloj de pulsera. Analice cómo se aplica cada uno de los siguientes términos y conceptos a la noción de un reloj: objeto, atributos, comportamientos, clase, herencia (por ejemplo, considere un reloj despertador), modelado, mensajes, encapsulamiento, interfaz y ocultamiento de información.

Marcando la diferencia

Hemos incluido en este libro ejercicios Marcando la diferencia, en los que le pediremos que trabaje con problemas que son de verdad importantes para individuos, comunidades, países y para el mundo. Estos ejercicios le serán muy útiles en la práctica profesional.

1.9 ***(Prueba práctica: calculadora de impacto ambiental del carbono)*** Algunos científicos creen que las emi siones de carbono, sobre todo las que se producen al quemar combustibles fósiles, contribuyen de manera considera ble al calentamiento global y que esto se puede combatir si las personas tomamos conciencia y limitamos el uso de los combustibles con base en carbono. Las organizaciones y los individuos se preocupan cada vez más por el "impacto ambiental debido al carbono". Los sitios Web como Terra Pass

`www.terrapass.com/carbon-footprint-calculator/`

y Carbon Footprint

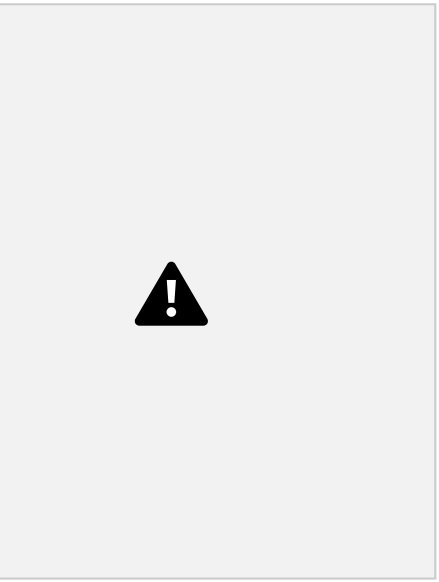
`www.carbonfootprint.com/calculator.aspx`

ofrecen calculadoras de impacto ambiental del carbono. Pruebe estas calculadoras para determinar el impacto que provoca usted en el ambiente debido al carbono. Los ejercicios en capítulos posteriores le pedirán que programe su propia calculadora de impacto ambiental del carbono. Como preparación, le sugerimos investigar las fórmulas para calcular el impacto ambiental del carbono.

1.10 ***(Prueba práctica: calculadora del índice de masa corporal)*** La obesidad provoca aumentos considerables en las enfermedades como la diabetes y las cardiopatías. Para determinar si una persona tiene sobrepeso o padece de obesidad, puede usar una medida conocida como índice de masa corporal (IMC). El Departamento de salud y ser
vicios humanos de Estados Unidos proporciona una calculadora del IMC en http://www.nhlbi.nih.gov/guidelines/obesity/BMI/bmicalc.htm. Úsela para calcular su propio IMC. Un próximo ejercicio le pedirá que programe su propia calculadora del IMC. Como preparación, le sugerimos usar la Web para investigar las fórmulas para calcular el IMC.

1.11 ***(Atributos de los vehículos híbridos)*** En este capítulo aprendió sobre los fundamentos de las clases. Aho ra empezará a describir con detalle los aspectos de una clase conocida como "Vehículo híbrido". Los vehículos hí bridos se están volviendo cada vez más populares, puesto que por lo general pueden ofrecer mucho más kilometra je que los vehículos operados sólo por gasolina. Navegue en Web y estudie las características de cuatro o cinco de los autos híbridos populares en la actualidad; después haga una lista de todos los atributos relacionados con sus características de híbridos que pueda encontrar. Por ejemplo, algunos de los atributos comunes son los kilómetros por litro en ciudad y los kilómetros por litro en carretera. También puede hacer una lista de los atributos de las baterías (tipo, peso, etc.).

1.12 ***(Neutralidad de género)*** Muchas personas desean eliminar el sexismo de todas las formas de comunicación. Usted ha recibido la tarea de crear un programa que pueda procesar un párrafo de texto y reemplazar palabras que tengan un género específico con palabras neutrales en cuanto al género. Suponiendo que recibió una lista de palabras con género específico y sus reemplazos con neutralidad de género (por ejemplo, reemplace "esposo" y "esposa" por "cónyuge", "hombre" y "mujer" por "persona", "hija" e "hijo" por "descendiente"), explique el procedimiento que utilizaría para leer un párrafo de texto y realizar estos reemplazos en forma manual. ¿Cómo podría su procedimiento hacer las adaptaciones de género de los artículos que acompañan a las palabras reemplazadas? Pronto aprenderá que un término más formal para "procedimiento" es "algoritmo", y que un algoritmo especifica los pasos a realizar, ade más del orden en el que se deben llevar a cabo. Le mostraremos cómo desarrollar algoritmos y luego convertirlos en programas de Java que se puedan ejecutar en computadoras.



Introducción a las aplicaciones y operadores en Java: entrada/salida

¿Qué hay en un nombre? A eso a lo que llamamos rosa, si le diéramos q
otro nombre conservaría su misma fragancia dulce.

—William Shakespeare

El mérito principal del lenguaje es la claridad.

—Galen

Una persona puede hacer la diferencia y cada persona debería intentarlo.

—John F. Kennedy

Objetivos

En este capítulo aprenderá a:

- Escribir aplicaciones simples en Java.
- Utilizar las instrucciones de entrada y salida.
- Familiarizarse con los tipos primitivos de Java.
- Comprender los conceptos básicos de la memoria.
- Utilizar los operadores aritméticos.
- Comprender la precedencia de los operadores aritméticos.
- Escribir instrucciones para tomar decisiones.
- Utilizar los operadores relacionales y de igualdad.

2.1 Introducción

En este capítulo le presentaremos la programación de aplicaciones en Java. Empezaremos con ejemplos de programas que muestran (dan salida a) mensajes en la pantalla. Después veremos un programa que obtiene (da entrada a) dos números de un usuario, calcula la suma y muestra el resultado. Usted aprenderá cómo ordenar a la computadora que realice cálculos aritméticos y guardar sus resultados para usarlos más adelante. El último ejemplo demuestra cómo tomar decisiones. La aplicación compara dos números y después muestra mensajes con los resultados de la comparación. Usará las herramientas de la línea de comandos del JDK para compilar y ejecutar los programas de este capítulo. Si prefiere usar un entorno de desarrollo in tegrado (IDE), también publicamos videos Dive Into® en http://www.deitel.com/books/jhtp10/ para

Eclipse, NetBeans e IntelliJ IDEA.

2.2 Su primer programa en Java: impresión de una línea de texto

Una **aplicación** en Java es un programa de computadora que se ejecuta cuando usted utiliza el **comando java** para iniciar la Máquina Virtual de Java (JVM). Más adelante en esta sección hablaremos sobre cómo compilar y ejecutar una aplicación de Java. Primero vamos a considerar una aplicación simple que muestra una línea de texto. En la figura 2.1 se muestra el programa, seguido de un cuadro que muestra su salida.

```
1 // Fig. 2.1: Bienvenido1.java
2 // Programa para imprimir texto.
3
4 public class Bienvenido1
5 {
6 // el método main empieza la ejecución de la aplicación en Java 7 public static
void main(String[] args)
8 {
9 System.out.println("Bienvenido a la programacion en Java!"); 10 } // fin del
método main
11 } // fin de la clase Bienvenido1
```

Fig. 2.1 Programa para imprimir texto (parte 1 de 2).


```
Bienvenido a la programacion en Java!
```

Fig. 2.1 Programa para imprimir texto (parte 2 de 2). *Nota*: En los códigos se han omitido los acentos para evitar la incompatibilidad al momento de ejecutarse en distintos entornos.

El programa incluye números de línea, que incluimos para fines académicos; *no* son parte de un progra ma en Java. Este ejemplo ilustra varias características importantes de Java. Pronto veremos que la línea 9 se encarga del verdadero trabajo: mostrar la frase Bienvenido a la programacion en Java! en la pan talla.

Comentarios en sus programas

Insertamos **comentarios** para **documentar los programas** y mejorar su legibilidad. El compilador

de Java *ignora* los comentarios, de manera que la computadora *no* hace nada cuando el programa se ejecuta. Por convención, comenzamos cada uno de los programas con un comentario, el cual indica el núme ro de figura y el nombre del archivo. El comentario en la línea 1

```
// Fig. 2.1: Bienvenido1.java
```

empieza con //, lo cual indica que es un **comentario de fin de línea**, el cual termina al final de la línea en la que aparecen los caracteres //. Un comentario de fin de línea no necesita empezar una línea; también puede estar en medio de ella y continuar hasta el final (como en las líneas 6, 10 y 11). La línea 2

```
// Programa para imprimir texto.
```

según nuestra convención, es un comentario que describe el propósito del programa. Java también cuenta con **comentarios tradicionales**, que se pueden distribuir en varias líneas, como en

```
/* Éste es un comentario tradicional. Se puede
 dividir en varias líneas */
```

Estos comentarios comienzan y terminan con los delimitadores /* y */. El compilador ignora todo el texto entre estos delimitadores. Java incorporó los comentarios tradicionales y los comentarios de fin de línea de los lenguajes de programación C y C++, respectivamente. Nosotros preferimos usar los comenta rios con //.

Java también cuenta con un tercer tipo de comentarios: los **comentarios Javadoc**, que están delimi tados por /** y */. El compilador ignora todo el texto entre los delimitadores. Estos comentarios nos permiten incrustar la documentación de manera directa en nuestros programas. Dichos comentarios son el formato preferido en la industria. El **programa de utilería javadoc** (parte del JDK) lee esos comentarios y los utiliza para preparar la documentación de su programa, en formato HTML. En el apéndice G en línea, Creación de documentación con javadoc, demostramos el uso de los comentarios Javadoc y la herramien ta javadoc.

Error común de programación 2.1

*Olvidar uno de los delimitadores de un comentario tradicional o Javadoc es un **error de sintaxis**, el cual ocurre cuando el compilador encuentra un código que viola las reglas del lenguaje Java (es decir, su sintaxis). Estas reglas son similares a las reglas gramaticales de un lenguaje natural que especifican la estructura de sus oraciones. Loserrores desintaxis seconocen también como **errores del compilador, errores en tiempo de compilación** o **errores de compilación**, ya que el compilador los detecta durante la compilación del programa. Al encontrar un error de sintaxis, el compilador genera un mensaje de error. Debe eliminar todos los errores de compilación para que su programa se compile de manera correcta.*

Buena práctica de programación 2.1

Ciertas organizaciones requieren que todos los programas comiencen con un comentario que explique su propósito, el autor, la fecha y la hora de la última modificación del mismo.

Tip para prevenir errores 2.1

Cuando escriba nuevos programas o modifique alguno que ya exista, mantenga sus comentarios actualizados con el código. A menudo los programadores tendrán que realizar cambios en el código existente para corregir errores o mejorar capacidades. Al actualizar sus comentarios, ayuda a asegurar que éstos reflejen con precisión lo que el código hace. Esto facilitará la comprensión y edición de su programa en el futuro. Los programadores que usan o actualizan código con comentarios obsoletos podrían realizar suposiciones incorrectas sobre el código, lo cual podría conducir a errores o incluso infracciones de seguridad.

Uso de líneas en blanco

La línea 3 es una línea en blanco. Las líneas en blanco, los espacios y las tabulaciones facilitan la lectura

de los programas. En conjunto se les conoce como **espacio en blanco**. El compilador ignora el espacio en blanco.

Buena práctica de programación 2.2

Utilice líneas en blanco y espacios para mejorar la legibilidad del programa.

Declaración de una clase

La línea 4

```
public class Bienvenido1
```

comienza una **declaración de clase** para la clase Bienvenido1. Todo programa en Java consiste al menos de una clase que usted (el programador) debe definir. La **palabra clave class** introduce una declaración de clase, la cual debe ir seguida de inmediato por el **nombre de la clase** (Bienvenido1). Las **palabras clave** (también conocidas como **palabras reservadas**) se reservan para uso exclusivo de Java y siempre se escriben
en minúscula. En el apéndice C se muestra la lista completa de palabras clave de Java. En los capítulos 2 al 7, todas las clases que definimos comienzan con la palabra clave public. Por ahora, sólo recuerde que debemos usar public. En el capítulo 8 aprenderá más sobre las clases public y las que no son public.

Nombre de archivo para una clase ***public***

Una clase public *debe* colocarse en un archivo que tenga el nombre de la forma *NombreClase*.java. Por lo tanto, la clase Bienvenido1 se almacenará en el archivo Bienvenido1.java.

Error común de programación 2.2

Ocurrirá un error de compilación si el nombre de archivo de una clase public *no es exactamente el mismo nombre que el de la clase (tanto por su escritura como por el uso de mayúsculas y minúsculas) seguido de la extensión* .java.

Nombres de clases e identificadores

Por convención, todos los nombres de clases comienzan con una letra mayúscula, y la primera letra de cada palabra en el nombre de la clase debe ir en mayúscula (por ejemplo, EjemploDeNombreDeClase). El nombre de una clase es un **identificador**, es decir, una serie de caracteres que pueden ser letras, dígitos, guiones bajos (_) y signos de moneda ($), que *no* comiencen con un dígito *ni* tengan espacios. Algunos identifica dores válidos son Bienvenido1, $valor, _valor, m_campoEntrada1 y boton7. El nombre 7boton *no* es un identificador válido, ya que comienza con un dígito, y el nombre campo entrada *tampoco* lo es debido a que contiene un espacio. Por lo general, un identificador que no empieza con una letra mayúscula no es el