Prólogo

Sugerí a mi hermano Todd, que hace el salto desde el hardware a la programación, que la siguiente gran revolución estará en la ingeniería genética.

Diseñaremos a los microbios para hacer comida, combustible, y plástico; barrerán la contaminación del medio ambiente y en general nos permitirán dominar con maestría la manipulación del mundo físico por un fragmento de lo que cuesta ahora. Afirmé que eso haría a la revolución de las computadoras verse pequeña en contraste.

Luego me di cuenta de que cometía un error común a los escritores de ciencia ficción: perdiéndome en la tecnología (que es por supuesto fácil de hacer en la ciencia ficción). Un escritor experimentado sabe que la historia no se trata nunca de las cosas; se trata de la gente. La genética tendrá un impacto muy grande en nuestras vidas, pero no estoy tan seguro que dejará empequeñecida a la revolución de las computadoras (la cual posibilita la revolución genética) - o al menos la revolución de la información. La información se trata de hablar los unos con los otros: sí, los coches, los zapatos y las curas especialmente genéticas son importantes, pero al fin y al cabo son simplemente atavíos. Lo que verdaderamente tiene importancia es cómo guardamos relación con el mundo. Y muchas de estas cosas se tratan de comunicación.

Este libro es un caso en concreto. La mayoría de las personas pensó que fui muy atrevido o un poco loco para construir un libro completamente en la Web. ¿"Porqué lo compraría alguien"? preguntaron. Si yo hubiera sido de un carácter más conservador, entonces no lo habría hecho, pero realmente no quise escribir otro libro por computadora de la misma antigua forma. No supe qué ocurriría pero resultó ser la cosa más inteligente que alguna vez he hecho con un libro.

En primer lugar, las personas empezaron a enviar correcciones. Éste ha sido un proceso asombroso, porque las personas han investigado cada rincón y cada grieta y han percibido ambos, los errores especializados y gramaticales, ha sido posible eliminar errores de todos los tipos que conozco que de otra manera habrían pasado sin ser vistos. Las personas han sido simplemente fantásticas por eso, muy a menudo diciendo "Ahora Bien, no trato de decir esto en una forma crítica..." Y luego dándome una colección de errores que estoy seguro nunca habría encontrado. Siento como esto ha sido una clase de proceso grupal y eso realmente ha convertido al libro en algo especial. Por el valor de esta retroalimentación, he creado varias encarnaciones de un sistema llamado "BackTalk" para coleccionar y clasificar en categorías los comentarios.

Pero entonces empecé a oír "bien, estupendo, es bonito, usted ha adelantado una versión electrónica, pero quiero una copia impresa y comprometida de un editor real" Hice un intento muy duro para hacerlo fácil para todo el mundo de imprimir en un formato agradable pero eso no contuvo la demanda por el libro

publicado. La mayoría de la gente no quiere dar lectura al libro entero en pantalla, y transportar alrededor de una gavilla de papeles, no importa cuán agradablemente impresas, no les gustó tampoco. (Es más, pienso que no es tan barato en términos del toner de la impresora láser.) Parece que la revolución de la computadora no expulsará a los editores de negocio, después de todo. Sin embargo, un estudiante sugirió que esto puede volverse un modelo para la publicación de futuro: Los libros serán publicados en la Web primero, y sólo cuando existan suficientes garantías de interés el libro será editado. Actualmente, el grueso del pueblo de todos los libros son fracasos financieros, y quizá este método nuevo podría hacer la industria editorial más provechosa.

Este libro se convirtió en una experiencia aleccionadora para mí en otra forma. Originalmente me acerqué a Java como "simplemente otro lenguaje de programación", lo cual en muchos sentidos lo es. Pero a medida que el tiempo pasó y le estudié más profundamente, comencé a ver que la intención fundamental de este lenguaje fue diferente a otros lenguajes que había visto hasta el momento.

La programación trata de manejar complejidad: La complejidad del problema que usted quiere solucionar, situado en la complejidad de la máquina en la cual es solucionado. Por esta complejidad, la mayor parte de nuestros proyectos de programación fallan. Y todavía, de todos los lenguajes de programación de los cuales soy consciente, ninguno de ellos ha dado lo mejor y se ha decidido a que su meta principal de diseño debería conquistar la complejidad de desarrollar y el mantenimiento de los programas. Por supuesto, muchas decisiones del diseño de lenguajes fueron hechas con la complejidad en mente, pero en algún punto hubo siempre algunos otros asuntos que fueron considerados esenciales para estar añadidos en la mezcla. Inevitablemente, esos otros asuntos son lo que causan a los programadores a eventualmente "golpear la pared" con ese lenguaje. Por ejemplo, C++ tuvo que ser compatible hacia atrás con C, como así también eficiente (para permitir la migración fácil a los programadores de C). Esas son ambas metas muy útiles y dan mucha explicación del éxito de C ++, pero también exponen complejidad adicional que impide a algunos proyectos de ser finalizados. (Ciertamente, usted puede culpar a los programadores y a la gestión, pero si un lenguaje puede ayudar atrapando sus errores, ¿por qué no lo hace?). Como otro ejemplo, el Visual Basic (VB) estaba atado al BASIC, el cual no fue realmente diseñado para ser un lenguaje extensible, de tal manera todas las extensiones apiladas en VB han producido alguna sintaxis verdaderamente horrible e insostenible. Perl es compatible hacia atrás con Awk, Sed, Grep, y otras herramientas Unix que estaba supuesto a reemplazar, y como resultado es acusado de producir "código que solo se escribe" (esto es, después de unos pocos meses usted no lo puede leer). Por otra parte, C++, VB, Perl, y otros lenguajes como Smalltalk enfocaron una parte de sus esfuerzos de diseño en el tema de la complejidad y como consecuencia tienen un éxito notable en solucionar ciertos tipos de problemas.

Lo que, me impresionó más a medida que he llegado a entender Java es eso, en alguna parte en la mezcla de objetivos de diseño de Sun, parece que estaba la meta de reducir la complejidad para el programador. Como quien dice "nos

preocupamos por reducir el tiempo y la dificultad de producir código robusto" En los primeros días, esta meta resultó en código que no anduvo muy rápido pero efectivamente ha producido reducciones asombrosas en el tiempo de desarrollo (aunque ha habido muchas promesas hechas sobre qué tan rápidamente Java algún día ejecutará). La mitad o menos del tiempo que toma crear un programa equivalente en C++. Este resultado a solas puede ahorrar cantidades increíbles de tiempo y de dinero, pero Java no se detiene allí. Procede a envolver muchas de las tareas complicadas que han cobrado importancia, como el multihilado (multithreading) y la programación en red, en características del lenguaje o en bibliotecas que pueden a veces hacer esas tareas fáciles. Y finalmente, afronta algunos problemas de complejidad realmente grandes: Los programas interplataformas, los cambios dinámicos de código, y aun la seguridad, cada uno de los cuales puede ajustar su espectro de complejidad desde "el impedimento" al “show-stopper” (?). Así a pesar de los problemas de desempeño que hemos visto, la promesa de Java es tremenda: Nos puede hacer programadores significativamente más productivos.

Uno de los lugares en que veo el impacto más grande para esto es en la Web. El programar redes siempre ha sido duro, y Java lo hace fácil (y los diseñadores del lenguaje Java están trabajando en hacerlo aún más fácil). La programación de redes es cómo hablamos con los demás más eficazmente y más barato de lo que alguna vez hicimos con los teléfonos (el correo electrónico a solas ha revolucionado muchos negocios). Como hablamos mas entre nosotros, cosas asombrosas comienzan a ocurrir, posiblemente más asombrosas aun que la promesa de la ingeniería genética.

En todas las formas -creando programas, trabajando en equipos para crear programas, construyendo interfaces de usuario con las que programas puedan comunicarse con el usuario, ejecutando programas en tipos diferentes de máquinas, y escribiendo fácilmente programas que comunican a través de Internet- Java incrementa el ancho de banda de comunicación entre las personas. Pienso que los resultados de la revolución de la comunicación no pueden ser vistos a partir de los efectos de mover cantidades grandes de bits por allí; veremos la verdadera revolución porque todos nosotros podremos hablar con los demás más fácilmente: Unos con otros, pero también en grupos y, como un planeta. Me han sugerido que la siguiente revolución será la formación de una especie de mente global que resulte de una cantidad adecuada de personas y una cantidad adecuada de interconexiones. Java puede o no ser la herramienta que fomente esa revolución, pero al menos la posibilidad me ha hecho sentir que estoy haciendo algo significativo intentando enseñar el lenguaje.

Prefacio de la 3ra edición

Mucha de la motivación y el esfuerzo de esta edición es poner al día el libro con la liberación del lenguaje Java JDK 1.4. Sin embargo, también ha quedado claro que la mayoría de lectores usan el libro para obtener una comprensión sólida de los fundamentos de manera que puedan seguir adelante hacia temas

más complicados. Debido a que el lenguaje continúa creciendo, se volvió necesario (en parte a fin de que el libro no sobre estirara sus ataduras) reevaluar el significado de los "fundamentos". Esto significó, por ejemplo, reescribir completamente el capítulo de Concurrencia (Llamado antiguamente "Multithreading" (Multihilado)) a fin de dar los fundamentos básicos en las ideas centrales sobre hilos (threading). Sin ese núcleo, es difícil entender los asuntos más complejos sobre hilos.

También he tenido en cuenta la importancia del testeo de código. Sin un esqueleto (framework) de testeo integrado con tests que sean ejecutados cada vez que uno hace una construcción de su sistema, no hay forma de saber si el código es confiable o no. Para lograr esto en el libro, un esqueleto especial de testeo de unidades fue creado para mostrar y validar la salida de cada programa. Esto fue puesto en el Capítulo 15, un nuevo capítulo, junto con explicaciones de "ant", (el sistema estándar para la constitución de sistemas en Java de facto, parecido a "make") JUnit, (la unidad del esqueleto de testeo estándar de facto) y cobertura sobre el registro de actividades y afirmaciones, (nuevo en JDK 1.4) junto con una introducción para depurar y el trazado de perfiles. Para abarcar todos estos conceptos, el nuevo capítulo es denominado "Descubriendo Problemas," e introduce lo que yo ahora creo son habilidades fundamentales que todos los programadores de Java deberían tener en su caja de herramientas básica.

¿Además, he repasado cada ejemplo del libro y me he preguntado, "por qué lo hice así"? En la mayoría de los casos he hecho algunas modificaciones y mejoras, ambos para hacer los ejemplos más coherentes dentro de ellos mismos y también para demostrar lo que considero son las mejores prácticas en la codificación Java (al menos, dentro de las limitaciones de un texto introductorio). Los ejemplos que ya no tuvieron sentido para mí fueron removidos, y ejemplos nuevos se han agregado. Un número de los ejemplos existentes ha tenido un muy significativo rediseño y reimplementación.

Los 16 capítulos en este libro producen lo que pienso es una introducción fundamental para el lenguaje Java. El libro factiblemente puede ser utilizado como un curso introductorio. ¿Pero qué acerca del más material avanzado?

El plan original para el libro fue agregar un capítulo nuevo cubriendo los fundamentos de "Java 2 Edición Enterprise" (J2EE). Muchos de estos capítulos serían creados por amigos y colegas que trabajan conmigo en seminarios y otros proyectos, como Andrea Provaglio, Bill Venners, Chuck Allison, Dave Bartlett, y Jeremy Meyer. Cuando miré el progreso de estos nuevos capítulos, y la fecha límite del libro comencé a ponerme un poco nervioso. Luego observé que el tamaño de los primeros 16 capítulos era efectivamente igual que el tamaño de la segunda edición del libro. Y las personas algunas veces se quejan que ese es ya demasiado grande.

Los lectores han hecho muchos, muchos comentarios maravillosos acerca de las primeras dos ediciones de este libro, lo cual naturalmente ha sido muy agradable para mí. Sin embargo, de vez en cuando, alguien tiene quejas, y por

alguna razón una queja que surge periódicamente es "el libro es demasiado grande". En mi mente está esa débil condena con más razón si "demasiadas páginas" es la única queja. (Uno recuerda las quejas del Emperador de Austria sobre el trabajo de Mozart: ¡"demasiadas notas"! No de que estoy de cualquier manera tratando de compararme a Mozart.) Además, sólo puedo suponer que tal queja viene de alguien quién está aún tomando conocimiento de la inmensidad del mismo lenguaje Java y no ha visto el resto de los libros del tema. Debido a esto, una de las cosas que he tratado de hacer en esta edición es recortar las porciones que han quedado obsoletas, o al menos no son esenciales. En general, he tratado de pasar por encima de todo, quitar de la tercera edición lo que ya no es necesario, incluyendo cambios, y mejorando todas las cosas que podría. Me siento cómodo quitando porciones porque el material original permanece en el sitio Web (*www.BruceEckel.com*) y en el CD-ROM que acompaña este libro, en forma de primera y segunda ediciones libremente descargables del libro. Si usted quiere las cosas viejas, entonces están todavía disponibles, y éste es un alivio maravilloso para un autor. Por ejemplo, el capítulo "Diseño de Patrones" se tornó demasiado grande y ha sido movido a un propio libro: Thinking in Patterns (con Java) (también descargable en el sitio Web).

Ya había decidido que cuando la siguiente versión de Java (JDK 1.5) fuera publicada por Sun, la cual probablemente incluirá un importante tema nuevo llamado "generics" (inspirado en las plantillas de la C++), tendría que dividir el libro en dos para añadir ese nuevo capítulo. Una vocecita me dijo "¿por qué esperar?". Entonces, me decidí a hacerlo durante esta edición, y repentinamente todo tuvo sentido.

El nuevo libro no es un segundo volumen, son preferiblemente temas más avanzados. Será llamado "Thinking in Enterprise Java", y está actualmente disponible (en alguna forma) como una descarga gratuita desde www.BruceEckel.com. Debido a que es un libro separado, puede aumentar en tamaño para adecuarse a los temas necesarios. La meta, como "Pensando en Java", es producir una cobertura muy comprensible de los fundamentos de las tecnologías J2EE a fin que el lector se prepare para una cobertura más avanzada de estos temas. Se pueden encontrar más detalles en el Apéndice C.

Para aquéllos de ustedes que todavía no pueden aguantar el tamaño del libro, me disculpo. Aunque parezca mentira, he trabajado duramente para mantenerlo pequeño. A pesar del volumen, tengo la impresión de que pueden haber en el bastantes opciones para complacerlos. En primer lugar, el libro está disponible electrónicamente, así es que si usted llevara su laptop, puede poner el libro en el sin agregar peso adicional a su viaje. Si usted quiere algo realmente más liviano, hay actualmente versiones del libro para Palm Pilot dando vueltas por ahí. (Una persona me contó que leería el libro en su Palm en la cama a medialuz para evitar molestar a su esposa. Sólo espero que lo ayude a llegar al país de los sueños.) Si usted lo necesita en papel, se de personas que imprimen un capítulo a la vez y lo llevan en su portafolio para leer en el tren.

Java 2, JDK 1.4

Las ediciones del Java JDK son numeradas 1.0, 1.1, 1.2, 1.3, y para este libro, 1.4. A pesar de que estos números de versión están todavía en los "unos", la forma estándar para referirse a cualquier versión del lenguaje que sea JDK 1.2 o más grande es llamarle "Java 2". Esto indica los cambios muy significativos entre "el viejo Java" -que tuvo muchos problemas de los que me quejé en la primera edición de este libro- y esta versión más moderna y mejorada del lenguaje, la cual tiene mucha menos cantidad de problemas y muchas adiciones y lindos diseños.

Este libro está escrito para Java 2, en particular JDK 1.4 (mucho del código no compilará con versiones anteriores, y el sistema construido se quejará y se detendrá si usted hace el intento). Tengo el gran lujo de deshacerme de todas las cosas viejas y escribir solo para el lenguaje nuevo y mejorado, porque la información vieja todavía existe en las anteriores ediciones, en la Web, y en el CD-ROM. También, porque cualquiera libremente puede bajar el JDK de java.sun.com, eso significa que escribiendo para JDK 1.4, no impongo una dificultad financiera a todos obligándolos a hacer la actualización.

Las versiones previas de Java fueron lentas en llegar a Linux, pero esto parece haberse arreglado, las nuevas versiones son lanzadas al mercado para Linux al mismo tiempo que para otras plataformas -ahora hasta la Macintosh comienza a mantenerse al día con las más recientes versiones de Java (vea a www.Linux.org). Linux es un desarrollo muy importante en conjunción con Java, porque se está volviendo rápidamente la plataforma de servidores más importante allí fuera -rápido, confiable, robusto, seguro, bien mantenido, y gratis, es una verdadera revolución en la historia de computación (pienso que nunca hemos visto todas esas características antes en alguna herramienta). Y Java ha encontrado un nicho muy importante en la programación del lado del servidor (server-side) en la forma de "Servlets" y "JavaServer Pages" (JSPs), tecnologías que son mejoras enormes sobre la programación tradicional en "Common Gateway Interface" (CGI) (éstos y los temas relatados están cubiertos en "Thinking in Enterprise Java).