D: Recursos Software

El JDK de *java.sun.com*. Incluso si se elige un entorno de desarrollo de un tercero, siempre es buena idea tener el JDK a mano por si se presenta algo que pudiera ser un error del compilador. El JDK es la referencia y si hay un fallo en él, seguro que éste es bien conocido.

La documentación HTML de Java de *java.sun.com*. Nunca he encontrado un libro de referencias sobre bibliotecas estándar de Java que no estuviera anticuado o al que le faltara algo. Aunque la documentación HTML de Sun está salpicada de pequeños fallos y en ocasiones es tan tersa que no se puede usar, todas las clases y métodos, al menos *están* ahí. La gente suele mostrarse poco cómoda al principio al usar un recurso en línea en vez de un libro impreso, pero al menos se puede disponer de información a grandes rasgos. Si no es suficiente, entonces deberá acudirse a libros impresos.

Libros

Thinking in Java, 1st Edition. Disponible como HTML completamente indexado, ,y con la sintaxis reemplazada en el CD ROM en este libro, o como descarga gratuita de *www.BruceEckel.com.* Incluye material viejo y material que no se tuvo lo suficientemente en consideración como para incluirlo en la segunda edición.

Core Java 2, de Horstmann & Cornell, Volume I —Fundamentals (Prentice-Hall, 1999). Volume II —Advanced Features, 2000. Libro muy comprensivo y el primer recurso al que acudo yo cuando necesito respuestas. Es el libro que yo recomiendo cuando se ha acabado *Piensa en Java*, y se precisa pasar a un nivel superior.

Java in a Nutshell: A Desktop Quick Reference, 2nd Edition, de David Flanagan *(O'Reilly, 1997)*. Un resumen compacto de la documentación en línea de Java. Prefiero navegar por los documentos de *java.sun.com*, especialmente porque varían a menudo. Sin embargo, tengo muchos colegas que prefieren documentación impresa, y ésta satisface los requisitos; también proporciona mayores discusiones que la documentación en línea.

The Java Class Libraries: An Annotated Reference, de Patrick Chan y Rosanna Lee (Addison-Wesley, 1997). Es lo que *debería* haber sido la documentación en línea: bastante descripción con el propósito de hacer algo usable. Uno de los revisores técnicos de *Piensa en Java* dijo, "si sólo tuviera un libro técnico de Java, sería éste (bien, además del tuyo, claro)". Yo no estoy tan enamorado del libro como él. Es grande, caro y la calidad de sus ejemplos no me satisface. *Pero* es un lugar al que acudir cuando no se queda bloqueado y parece tener más profundidad (y tamaño) que *Java in a Nutshell.*

Java Network Programming, de Elliotte Rusty Harold (O'Reilly, 1997). No empecé a entender la red en Java hasta que encontré este libro. También encuentro su sitio web, Café au Lait, estimulante, digna de mención y una perspectiva actualizada de los desarrollos de Java, y a la que todos los

vendedores veneran. Sus actualizaciones regulares permiten mantenerse al día de las noticias relativas a Java tan rápidamente cambiante. Ver *http://metalab.unc.edu/javafaq/*.

JDBC Database Access with Java, de Hamilton, Catell & Fisher (Addison-Wesley-1997). Si no se sabe nada de SQL y bases de datos, es una introducción buena y gentil. También contiene algunos detalles además de una "referencia anotada" a la API (de nuevo, lo que debería haber sido la documentación en línea). Su pega, como con todos los libros de *The Java Series* ("Los ÚNICOS libros autorizados por JavaSoft") es que ha sido censurado de forma que sólo dice cosas maravillosas sobre Java —nadie puede encontrar pegas en ningún libro de esta serie.

Java Programming with CORBA, de Andreas Vogel & Keith Duddy (John Wiley & Sons, 1997). Un tratamiento serio del tema con ejemplos de código para tres ORBs Java (Visibroker, Orbix, Joe).

Design Patterns, de Gamma, Helm, Jonson & Vlissides (Addison-Wesley, 1995). El primer libro que gestó el movimiento de los patrones en el mundo de la programación.

Practical Algorithms for Programmers, de Binstock & Rex (Addison-Wesley, 1995). Los algoritmos están en C, por lo que son bastante fáciles de traducir a Java. Cada algoritmo incluye una explicación muy detallada.

Análisis y Diseño

Extreme Programming Explained, de Kent Beck (Addison-Wesley, 2000) (próxima publicación en castellano, 2002). *Me encanta* este libro. Sí, tiendo a tomar un enfoque radical de las cosas, pero siempre he pensado que podría haber un proceso de desarrollo de programas diferente y mucho mejor, y creo que XP está bastante próximo. El único libro que ha tenido un impacto comparable en mí fue *PeopleWare* (descrito más abajo), que habla principalmente del entorno y de cómo tratar la cultura corporativa. *Extreme Programming Explained* habla sobre la programación, e incluye todo, incluso los "hallazgos" más recientes. Incluso llegan tan lejos que dicen que los dibujos estén bien siempre y cuando no se gaste demasiado tiempo en ellos, teniendo en mente que acabarán en la basura. (Se verá que este libro *no* tiene el "sello UML de aprobación" en su cubierta). Llegaría a elegir si trabajar en una compañía o no sólo en función de si usa XP. Es un libro pequeño, con capítulos pequeños, cuya lectura no supone un gran esfuerzo, y que presenta temas excitantes. Uno empieza a pensar que trabaja en una atmósfera así e incluso acaba teniendo visiones sobre todo el mundo.

UML Distilled, 2nd Edition, de Margin Fowler (Addison-Wesley, 2000). Cuando se encuentra UML por primera vez, asusta ver tantos diagramas y detalles. De acuerdo con Fowler, la mayoría de esto es innecesario y él corta por lo sano dejando sólo la esencia. Para la mayoría de proyectos, sólo hay que conocer unas pocas herramientas de generación de diagramas, y la meta de Fowler es llegar a un buen diseño en vez de preocuparse por todos los artefactos para llegar a él. Un libro delgado, bueno y legible; el primero que debería consultarse si hubiera que entender UML.

UML Toolkit, de Hans-Erik Eriksson & Magnus Penker (John Wiley & Sons, 1997). Explica UML y cómo usarlo,y tiene un caso de estudio en Java. El CD ROM que adjunta contiene el código Java y una versión limitada de Rational Rose. Una introducción excelente a UML y a cómo usarlo para construir un sistema real.

The Unified Software Development Process, de Ivar Jacobson, Grady Booch y James Rumbaugh (Addison-Wesley, 1999) (disponible en castellano). Mi predisposición hacia este libro era que no me gustara. Parecía tener todos los elementos de un libro de texto aburrido. Me llevé una grata sorpresa —sólo algunas pequeñas partes del libro contienen explicaciones que parece como si los conceptos ni siquiera estuvieran claros para los autores. La gran mayoría del libro no sólo es clara, sino digna de disfrutarla. Y lo mejor de todo, el proceso tiene mucho sentido. No es Programación Extrema (y no tiene su claridad a la hora de las pruebas) pero también es parte de UML —incluso si no se pudiera adoptar XP, la mayoría de gente está de acuerdo en que "UML es bueno" (independientemente del nivel de experiencia que tengan con el mismo) por lo que se podría adoptar. Creo que este libro debería ser el abanderado del UML, y el que debería leerse tras *UML Distilled* de Fowler si se desea más nivel de detalle.

Antes de elegir un método, ayuda a tener perspectivas de aquéllos que no intentan vender ninguno. Es fácil adoptar un método sin entender verdaderamente qué se desea de él o para qué sirve exactamente. Otros lo usan, y eso parece una razón de peso. Sin embargo, las personas tienen un comportamiento psicológico algo extraño en ocasiones: cuando desean creer que algo soluciona sus problemas, lo prueban. (Esto es experimentar, que es bueno). Pero si no soluciona sus problemas, pueden redoblar sus esfuerzos y anunciar a voz en grito lo maravilloso que es lo que han descubierto. Se presupone, en este caso, que si se logra que más gente monte en el mismo barco, uno ya no está sólo, incluso aunque el barco no lleve a ninguna parte (o se esté hundiendo).

No quiero decir que ninguna metodología conduzca a ninguna parte, sino que uno debería pensar en la técnica experimental ("No funciona; vamos a buscar alguna otra cosa") en vez del modo negación ("No, eso no es verdaderamente un problema. Todo va bien, no tenemos por qué cambiar"). Creo que los libros siguientes, leídos *antes* de elegir un método, le proporcionarán bastante ayuda.

Software Creativity, de Robert Glass (Prentice-Hall, 1995). Éste es el mayor libro que he visto que discuta la *perspectiva* de todos los aspectos metodológicos. Es una colección de pequeños ensayos y artículos escritos por Glass, o bien recopilados por él (P. J. Plauger es uno de sus colaboradores), reflejando muchos años de pensamiento y estudio en la materia. Son entretenidos sólo lo suficientemente largos como para decir aquello que es necesario; no se extiende ni aburre. No es simplemente humo, sino que incluye cientos de referencias a otros artículos y estudios. Todo programador y gestor debería leer este libro antes de introducirse en el mar de la metodología.

Software Runaways: Monumental Software Disasters, de Robert Glass (Prentice-Hall, 1997). Lo mejor de este libro es que presenta aquello de lo que no se suele hablar: proyectos no sólo que fallaron, sino que lo hicieron estrepitosamente. Creo que la mayoría de nosotros pensamos "Eso no me puede pasar a mí" (o "Eso no me puede *volver a* ocurrir"), y creo que esto te coloca en clara desventaja. Tener en mente que todo puede fallar te permite estar en una posición mucho mejor como para hacer todo bien.

Peopleware, 2nd Edition, de Tom Demarco y Timothy Lister (Dorset House, 1999). Aunque tienen experiencia en el desarrollo de software, este libro es sobre proyectos y equipos en general. Se centran en la *gente* y sus necesidades, en vez de en la tecnología y sus necesidades. Hablan de la creación de un entorno en el que todo el mundo sea feliz y productivo, en vez de decidir qué reglas deberían seguir estas personas para ser los componentes adecuados de una máquina. Esta última

actitud, creo, es la mayor contribución a la sonrisa y disfrute de los programadores cuando se adopta el método XYZ y simplemente se hace lo que se ha hecho siempre.

Complexity, de M. Mitchell Waldrop (Simon & Schuster, 1992). Se trata de una crónica de cómo varios científicos de varias disciplinas se juntan en Santa Fe, Nuevo Méjico, para discutir problemas reales que no podrían solucionar sus disciplinas individuales (la bolsa en económicas, la formación inicial de la vida en biología, por qué la gente hace lo que hace en sociología, etc.) Atravesando la física, economía, química, matemáticas, computación, sociología, etc. se logra un enfoque multidisciplinar. Pero lo que es más importante, emerge una forma distinta de *pensar* en estos problemas ultra-complejos: lejos del determinismo matemático y de la ilusión con la que se puede escribir una ecuación que prediga todo comportamiento, y más dirigida a *observar* y buscar un posible patrón e intentar emular ese patrón por cualquier medio. (El libro relata, por ejemplo, la emergencia de algoritmos genéticos). Este tipo de pensamiento, creo, es útil puesto que observamos formas de gestionar proyectos software más y más complejos.

Python

Learning Python, de Mark Lutz y David Ascher (O'Reilly, 1999). Una introducción genial para programadores a lo que se está convirtiendo rápidamente en mi lenguaje favorito, un compañero excelente para Java. El libro incluye una introducción a JPython, lo que permite combinar Java y Python en un mismo programa (el intérprete JPython se compila a *bytecodes* Java, así que no hay que añadir nada especial para que todo esto sea posible). Esta unción de lenguajes promete unas posibilidades enormes.

Mi propia lista de libros

Listados en orden de publicación, no todos ellos siguen estando disponibles.

Computer Interfacing with Pascal & C, (Autopublicado por la imprenta Eisys, 1988. Disponible únicamente vía *http://www.BruceEckel.com*). Una introducción a la electrónica desde cuando CP/M seguía siendo el rey y DOS un principiante. Usé lenguajes de programación de alto nivel y muy a menudo el puerto paralelo del ordenador para manejar varios proyectos electrónicos. Adaptación de mis columnas en la primera y mejor revista para la que he escrito, *Micro Cornucopia* (Para parafrasear a Larry O'Brien, editor durante mucho tiempo de *Software Development Magazine*: la mejor revista de informática jamás publicada —incluso tenían ¡planes para fabricar un robot en un florero!.).Micro C se perdió mucho antes de que apareciera Internet. Crear este libro fue una experiencia de publicación muy gratificante.

Using C++, (Osborne/McGraw-Hill, 1993). Uno de los primeros libros de C++. Está fuera de edición y reemplazada por su segunda edición, de nombre *C++ Inside & Out*.

C+ Inside & Out, (Osborne / McGraw-Hill, 1993). Como se ha dicho, es de hecho la 2ª edición de **Using C++**. El C++ de este libro es razonablemente exacto, pero data de 1992, así que *Thinking in*

C++ trata de reemplazarlo. Puede encontrarse más información sobre este libro y descargar el código fuente en *http://www.BruceEckel.com*.

Thinking in C++, 1st edition, (Prentice-Hall, 1995).

Thinking in C++, 2nd Edition, Volume I, (Prentice-Hall, 2000). Descargable de *http://www.BruceEckel.com*.

Black Belt C++, the Master's Collection, Bruce Eckel, editor (M&T Books, 1994). Fuera de tirada. Una colección de capítulos de varias reflexiones sobre C++ basadas en sus presentaciones en la serie de C++ de la *Software Development Conference*, que presidí. La cubierta de este libro me estimuló a intentar controlar futuros diseños de las portadas.

Thinking in Java, 1st Edition, (Prentice-Hall, 1998). La primera edición de este libro ganó el premio a la productividad de la *Software Development Magazine*, el premio *Java Developer's Editor's Choice*, y el premio al mejor libro de *JavaWorld Reader's Choice*. Descargable de *http://www.BruceEckel.com*.