致读者：

我从 2002 年 7 月开始翻译这本书，当时还是第二版。但是翻完前言和介绍部分后，chinapub 就登出广告，说要出版侯捷的译本。于是我中止了翻译，等着侯先生的作品。

我是第一时间买的 这本书，但是我失望了。比起第一版，我终于能看懂这本书了，但是相比我的预期，它还是差一点。所以当 Bruce Eckel 在他的网站上公开本书的第三版的时候，我决定把它翻译出来。

说说容易，做做难。一本 1000 多页的书不是那么容易翻的。期间我也曾打过退堂鼓，但最终还是全部翻译出来了。从今年的两月初起，到 7 月底，我几乎放弃了所有的业余时间，全身心地投入本书的翻译之中。应该说，这项工作的难度超出了我的想像。

首先，读一本书和翻译一本书完全是两码事。英语与中文是两种不同的语言，用英语说得很畅的句子，翻成中文之后就完全破了相。有时我得花好几分钟，用中文重述一句我能用几秒钟读懂的句子。更何况作为读者，一两句话没搞懂，并不影响你理解整本书，但对译者来说，这就不一样了。

其次，这是一本讲英语的人写给讲英语的人的书，所以同很多要照顾非英语读者的技术文档不同，它在用词，句式方面非常随意。英语读者会很欣赏这一点，但是对外国读者来说，这就是负担了。

再有，Bruce Eckel 这样的大牛人，写了 1000 多页，如果都让你读懂，他岂不是太没面子？所以，书里还有一些很有“禅意”的句子。比如那句著名的“The genesis of the computer revolution was in a machine. The genesis of our programming languages thus tends to look like that machine.”我就一直没吃准该怎么翻译。我想大概没人能吃准，说不定 Bruce 要的就是这个效果。

这是一本公认的名著，作者在技术上的造诣无可挑剔。而作为译者，我的编程能力差了很多。再加上上面讲的这些原因，使得我不得不格外的谨慎。当我重读初稿的时候，我发现需要修改的地方实在太多了。因此，我不能现在就公开全部译稿，我只能公开已经修改过的部分。不过这不是最终的版本，我还会继续修订的。

本来，我准备到 10 月份，等我修改完前 7 章之后再公开。但是，我发现我又有点要放弃了，因此我决定给自己一点压力，现在就公开。以后，我将修改完一章就公开一章，请关注 www.wgqqh.com/shhgs/tij.html。

如果你觉得好，请给告诉我，你的鼓励是我工作的动力；如果你觉得不好，那就更应该告诉我了，我会参考你的意见作修改的。我希望能通过这种方法，译出一本配得上原著的书。

shhgs

2003 年 9 月 8 日

前言

我兄弟 Todd 正从硬件领域转向编程。我提醒他，下一次重大的革命将是基因工程。

我们会设计出一些能生产食品，燃料以及塑料的微生物。它们还能清除污染。总之我们可以用它们来操控物质世界，而成本只是现在的一个零头。我敢打赌，相形之下，计算机革命将会显得微不足道。

然而我发觉，我犯了一个科幻小说作家常犯的错误：迷失在技术中了(当然写科幻小说很容易就会这样)。一个有经验的作家应该知道小说的主题决不能是某样东西，而应该是人。基因技术会极大的影响我们的生活，但我不能肯定它是否会矮化计算机革命(没有计算机革命，哪来的基因革命)，甚至是信息革命。信息牵涉到人和人之间的交流；而轿车，鞋子，特别是基因治疗技术，尽管十分重要，但说到底只不过是装饰。真正事关重大的是我们是如何同这个世界交流。在这个方面通讯就显得非常重要了。

本书就是一个例子。大多数人都认为我忒大胆，甚至是有些疯了，竟然会把整部书都放到了网上。他们问我“这样一来，还有什么理由要去买它呢？”。如果我的性格稍微保守一些，我就不会这么做了。但我实在是不想再用老方法写一本计算机的书了。我也不知道会发生些什么，但结果居然是，这成了我在写作方面做的最聪明的一件事。

一方面，人们开始给我发勘误信。这个过程真是令人叫绝。他们搜遍了每个角落，连犄角旮旯都不放过，把技术错误和文法错误全部纠了出来。这样我就能消除各种五花八门，原本可能会被漏掉的 bug。他们发信通常都很有趣。经常这么说，“好吧，我想这东西可能算不上非常严重...”,然后给了我一大串错误，都是那些我打破头也找不到的。我觉得这有点象是集体参与，这点使得本书有些与众不同。

但是，接着我就听到有人说，“嘿，不错！你把电子版的放到了网上，但是我要一本真正从出版社出的，印好订好的纸版书。” 为了让每个人都能漂漂亮亮地把它打印出来，在排版方面我可没少花心思。但是这阻止不了人们对印刷书籍的需求。绝大多数人不愿意在屏幕上读完整部书；同样，不论打印得多漂亮，拖着一大捆纸终究不是那么回事。(此外，激光打印机的硒鼓也不便宜。)看来，计算机革命最终还是革不了出版社的命。不过有个学员倒是提出，这或许是未来出版业的方向。书首先在网上出版，只有有了足够多的感兴趣读者，它才会被印到纸上。如今出的书，绝大多数都亏钱，或许这个新方法会提高出版业的利润。

从另一个角度来说，这本书的写作过程对我本人也有一番启迪。起初我只是将 Java 当作一种新的编程语言。尽管这么理解从很多方面来讲并没有

错，但是随着时间的推移以及我本人研究的深入，我开始认识到，这个语言的与我以前接触过的其他语言都不同。

大体上说编程就是控制复杂性。要解决的问题的复杂程度，是由解决它的工具的复杂程度所决定的。正因为如此，大多数项目都以失败而告终。然而就我所知，没有一种编程语言，在其设计之初，就全力以赴地打算帮助程序员以及维护人员，去征服这种复杂性。[1] 当然，许多语言的设计考虑到了这一点，但是最后总会因为这个或那个原因，掺进一些他们觉得“更重要”的东西。结果是，掺进去的总是那些会让程序员撞得头破血流的玩意儿。比如说，C++必须提供 C 的向后兼容(其目的是为了更方便的让 C 的程序员掌握 C++)，以及相同的运行效率。这两者都是很实际的设计目标，而且对 C++的成功贡献良多。然而它们也带来了某些额外的复杂性，使得某些项目不能顺利完成。(当然你也可以将它归咎与程序员和项目管理。但是如果别的语言可以帮你捕捉错误，为什么它就不行呢？)再举个例子，Visual Basic (VB)源于 BASIC，而 BASIC 从来就没打算成为一个可扩展的语言。所以堆在 VB 上那些扩展，造就了那些看起来吓人，而且根本没法维护的语法。Perl 对 Awk，Sed，Grep 以及其他 Unix 工具提供向后兼容，结果是它成了制造“只写代码”(也就是说，过几个月，你自己也看不懂了)的元凶。从另一个方面来看，C++，VB，Perl 以及其他诸如 Smalltalk 之类的语言，在解决复杂性的问题方面也都花了一番功夫，就某些特定的问题而言，它们功效卓著。

等我理解 Java 之后，最让我印象最深刻的是，我发现 Sun 在订立总体设计目标的时候，有一个专门的降低 Java 程序员难度的目标。就好像听到他们说“我们非常关心如何提高编写健壮的代码的效率，以及降低开发难度。”在最初的日子里，达成这个目标的代价就是，程序跑起来不够快，(不过听到很多承诺说将来 Java 会跑得有多快)，但是开发时间确实是大幅度的降了下来。比起用 C++来，开发一个项目只要用一半甚至更少的时间。单这一点就可以省下很多时间和金钱，但是 Java 还不止于此。它继续涉足那些重要领域，将诸如多线程和网络编程之类的复杂问题，归化到语言的特性或类库中，从而简化了任务。最后，它还解决了一些真正的大的复杂性问题：跨平台的程序，动态代码变化，以及安全性。如果你做演示的时候有人问到这些问题，那么它们中的任何一个，都会让你“语塞”甚至是“下不来台”。因此虽然 Java 的性能欠佳，它的前景还是非常看好的：它能让我们成为更加高产的程序员。

就我的观察，Web 是受它影响最大的一个领域。网络编程一直是个难点，但是 Java 让它变得方便了(Java 语言的开发者还在让它变得更容易)。网络编程就是要让我们，用比电话更有效更廉价的方式交流。(单单是 email 就使许多行业发生了革命性的变化)。随着人们交流越来越多，一些神奇的东西就会发生，甚至会比基因工程的前景更为神奇。

编程，团队开发，创建用户界面使得用户能同程序互动，在不同种类的机器上运行同一个程序，方便的编写 Internet 通讯程序。通过上述的途径，Java 扩展了人际交流的带宽。我想信息革命的成就，不应该以数据

流量来衡量，而是要看我们是否能以一种更为便捷的方式来相互交流。不仅是一对一，而且是一群人之间，甚至是在整个地球范围。我曾听说，由于有了足够多的人和足够多的通道，下一场革命将会是全球思想的重组。不论 Java 是不是促成这一革命的工具，但至少这种可能性让我觉得教这门语言是一件有意义的工作。

第三版前言

写这一版的目的，以及工作的重点，很大程度上是为了让本书能跟上 Java JDK 1.4 版的发布。同时我也明白，绝大多数读者学这本书的目的，是为了在 Java 方面打下扎实的基础，为学习更复杂内容的作准备。由于 Java 还在不断发展，所以重新评估什么是“基础知识”就变得很必要了，这也是本书所涉及的范围不再扩张的部分原因。这就意味着，举例来说，为了让读者能够了解线程的核心思想的基础知识，我重写了“并发 Concurrency”这一章(过去叫“多线程”Multithreading)。如果不知道它的核心思想，你很难理解多线程方面的更复杂的问题。

我也慢慢理解代码测试的重要性了。如果没有能运行测试程序的内置测试框架，那每次系统编译之后，你就无法知道代码是否可靠。在本书中，为了做到这一点，我创建了一个特殊的，用来显示和验证程序输出的单元测试的框架。这个内容放在新的第 15 章。在这一章中，还介绍 ant(事实上的标准 Java 编译系统，类似于 *make*)，JUnit(事实上 Java 单元测试框架的标准)，以及 JDK 1.4 中新增加的日志(logging)和断言(assertion)功能，以及调试(debugging)和性能测试 (profiling)。为了能概括这么多概念，新的一章被命名为“发现问题 Discovering Problems”。同时这一章还介绍了我所认为的当今 Java 程序员都该掌握的一些基本技能。

此外，我复查了本书的每个例程，并且问自己，“为什么要这么做？”大多数情况下，为了让代码风格更为一致，同时示范一下我所理解的良好的 Java 编程习惯(至少在比较基础的范围内)，我对代码作了一些改进。此外，我还去除了不再有意义的例子，加进了一些新的例子，重新设计和实现了很多已有的例子。

本书的第 16 章提出了我所理解的 Java 语言的基本介绍。这本书可以用作介绍课程的教材，但是怎样才能得到那些更高级的材料呢？

本书原计划加入一个新的章节，专门讨论“Java 2 Enterprise Edition” (J2EE)的基础知识。这些章节由我的朋友，以及一同授课或开发项目的同事，比如 Andrea Provaglio, Bill Venners, Chuck Allison, Dave Bartlett, 和 Jeremy Meyer 等人来写。但是当我察看这些新章节的进度，再对照出版的最终日期的时候，我就开始有些担心了。然后我发现，前 16 章的篇幅已经同本书的第二版一样大了。就是这个篇幅，读者们也会经常抱怨太大了。

对于本书的前两版，读者给了很高的评价，当然我也十分欣慰。但有时，他们也会抱怨。其中常被提起的就是“这本书实在太厚了”。在我看来，如果这是你抓到的唯一把柄，那你也是在太次了。(人们会想起奥地利国王对莫扎特作品的批评：“音符太多了”。当然，我可没有把自己比作莫扎特。)此外，我只能设想提出这种批评的人还未能了解 Java 语言的众多特性，还未读过其它的书籍。尽管如此，在这一版中，我还会尽量删减那些已经过时的，或是不那么关键的部分。这么做我很安心，因为本书的第一和第二版还可以从网站上(在 www.BruceEckel.com)免费下载，另外附在书后的 CD ROM 上也有。如果你还需要老的资料，它还在那里，对于作者来说这么做可以减轻很多负担。比方说，“设计模式(Design Patterns)”这一章太大了，已经独立成册了：Thinking in Patterns (Java 语言)(也可以在网站上下载)。

我已经决定，等到 Sun 发布下一版 Java(JDK 1.5)，我就把本书分成两册。估计那一版中有一个重大的改进，会(效法 C++的模板“templates”)引入范型(generics)。分成两册可以增加这些内容的章节。但是，有个声音悄悄在问“为什么要等到下一版？”。所以我在这一版中就这么做了，于是一切都明朗了。我往一本普及性的书里塞了太多东西。

新的书不是第二卷，而只是一些高级的课题。它会叫 Thinking in Enterprise Java。这本书现在可以在 www.BruceEckel.com(以某种形式)免费下载。由于是一本独立的书，因此它的篇幅可以随课题所需而扩展。就像 Thinking in Java 一样，它的目标是给读者做一个简单易懂的 J2EE 技术的基础知识的介绍。为读者能探讨更高级的课题做准备。你能在附录 C 中能找到更多细节。

对于那些仍旧不能忍受本书厚度的读者来说，我只能说抱歉了。不管你信不信，为了让它尽可能的薄，我是费尽心机。别去管书有多厚，我觉得你还可以用很多别的方法。比方说你可以得到本书的电子版(从网站或是随书的 CD ROM 上)，所以如果你平常带着笔记本的话，你可以把它放到你的笔记本里，这么做不会增添额外的重量。如果你想让它更轻，还有本书的 Palm Pilot 版，很容易找到。(有人跟我说，他喜欢躺在床上，打开屏幕的背光灯读这本书。还说这样就不会妨碍他妻子了。但愿他睡得香。)如果一定要在纸上读，据我所知有人一次打印一章，然后放到公文包里，在火车上读。

Java 2, JDK 1.4

Java JDK 发布的序号是 1.0, 1.1, 1.2, 1.3 以及本书讨论的 1.4。尽管这些版本号仍然停留在 1，但是对于 JDK 1.2 以及其后版本的正式称呼是 Java 2。它展示了新旧版本之间的巨大差异。旧版本中有许多瑕疵，对此我在本书的第一版中颇有非议；而新版本中的缺陷大大减少，并且增加了许多特性和精彩的设计。

本书是为 Java 2,特别是 JDK 1.4 所写的(很多代码不能在旧版本上编译，如果你试一下就会看到，编译系统会提示并且停下来)。由于有网上和 CD ROM 上的本书的前两版垫底，我在删旧迎新方面作得真是很奢侈。此外，因为任何人都能从 java.sun.com 自由地下载 JDK，所以我也不会因为为升了级的 JDK 1.4 写书，而跟大家要钱。

以前 Java 在 Linux(参见 www.Linux.org)版的发布方面总是慢一拍，但现在看来这个问题已经解决。新版 Java 同时发布了其 Linux 和其他平台的版本，现在甚至 Macintosh 版也跟上了。Linux 与 Java 关系密切。由于它快速，稳定，健壮，安全，易维护并且还是自由软件，因此正在以极快的速度成为一个最重要的服务器平台。这是计算机史上一场真正的革命(从来还没有见过，在一个工具上集中了所有这些特性)。而 Java 也以 Servlets 的方式在服务器端编程方面找到了用武之地(这些以及相关的课题在 Thinking in Enterprise Java 中讨论)。这项技术比传统的 CGI 编程有了巨大的改进。

---

[1] 在第二版中我收回这句话：我认为 Python 语言已经非常接近实现这一目标了。参见 *www.Python.org*。