教育部高等教育司推荐
国外优秀信息科学与技术系列教学用书

# 计算机科学导论
## ——伟大思想与Java程序设计

（影印版）

# GREAT IDEAS IN COMPUTER SCIENCE WITH JAVA

■ Alan W. Biermann
Dietolf Ramm

高等教育出版社
Higher Education Press
The MIT Press

教育部高等教育司推荐
国外优秀信息科学与技术系列教学用书

# 计算机科学导论

## ——伟大思想与 Java 程序设计

（影印版）

# GREAT IDEAS IN COMPUTER SCIENCE WITH JAVA

Alan W. Biermann

Dietolf Ramm

高等教育出版社

The MIT Press

图字：01-2002-2113 号

**Great Ideas in Computer Science with Java**

Alan W. Biermann, Dietolf Ramm

©2002 by **The Massachusetts Institute of Technology**

All rights reserved.

No part of this book may be reproduced in any form or by any electronic or mechanical means (including photocopying, recording, or information storage and retrieval) without permission in writing from the publisher.

This reprint is for sale in the People's Republic of China only and exclude Hong Kong and Macau.

**图书在版编目(CIP)数据**

计算机科学导论：伟大思想与 Java 程序设计 /（美）比尔曼（Biermann, A.W.），（美）拉姆（Ramm, D.）著. 影印本. —北京：高等教育出版社，2002．7
计算机系本科教材
ISBN 7-04-011258-2

Ⅰ.计… Ⅱ.①比…②拉… Ⅲ.①电子计算机—基础理论—高等学校—教材—英文②JAVA 语言—程序设计—高等学校—教材—英文 Ⅳ.TP3

中国版本图书馆 CIP 数据核字（2002）第 041609 号

计算机科学导论——伟大思想与 Java 程序设计(影印版)

Alan W. Biermann, Dietolf Ramm

| | | | |
|---|---|---|---|
| **出版发行** | 高等教育出版社 | **购书热线** | 010-64054588 |
| **社　　址** | 北京市东城区沙滩后街 55 号 | **免费咨询** | 800-810-0598 |
| **邮政编码** | 100009 | **网　　址** | http://www.hep.edu.cn |
| **传　　真** | 010-64014048 | | http://www.hep.com.cn |
| **经　　销** | 新华书店北京发行所 | | |
| **印　　刷** | 北京民族印刷厂 | | |
| **开　　本** | 787×1092　1/16 | **版　　次** | 2002 年 7 月第 1 版 |
| **印　　张** | 34.5 | **印　　次** | 2002 年 7 月第 1 次印刷 |
| **字　　数** | 800 000 | **定　　价** | 36.00 元 |

本书如有缺页、倒页、脱页等质量问题，请到所购图书销售部门联系调换。

**版权所有　侵权必究**

# 前　言

20 世纪末，以计算机和通信技术为代表的信息科学和技术对世界经济、科技、军事、教育和文化等产生了深刻影响。信息科学技术的迅速普及和应用，带动了世界范围信息产业的蓬勃发展，为许多国家带来了丰厚的回报。

进入 21 世纪，尤其随着我国加入 WTO，信息产业的国际竞争将更加激烈。我国信息产业虽然在 20 世纪末取得了迅猛发展，但与发达国家相比，甚至与印度、爱尔兰等国家相比，还有很大差距。国家信息化的发展速度和信息产业的国际竞争能力，最终都将取决于信息科学技术人才的质量和数量。引进国外信息科学和技术优秀教材，在有条件的学校推动开展英语授课或双语教学，是教育部为加快培养大批高质量的信息技术人才采取的一项重要举措。

为此，教育部要求由高等教育出版社首先开展信息科学和技术教材的引进试点工作。同时提出了两点要求，一是要高水平，二是要低价格。在高等教育出版社和信息科学技术引进教材专家组的努力下，经过比较短的时间，第一批引进的 20 多种教材已经陆续出版。这套教材出版后受到了广泛的好评，其中有不少是世界信息科学技术领域著名专家、教授的经典之作和反映信息科学技术最新进展的优秀作品，代表了目前世界信息科学技术教育的一流水平，而且价格也是最优惠的，与国内同类自编教材相当。

这项教材引进工作是在教育部高等教育司和高教社的共同组织下，由国内信息科学技术领域的专家、教授广泛参与，在对大量国外教材进行多次遴选的基础上，参考了国内和国外著名大学相关专业的课程设置进行系统引进的。其中，John Wiley 公司出版的贝尔实验室信息科学研究中心副总裁 Silberschatz 教授的经典著作《操作系统概念》，是我们经过反复谈判，做了很多努力才得以引进的。William Stallings 先生曾编写了在美国深受欢迎的信息科学技术系列教材，其中有多种教材获得过美国教材和学术著作者协会颁发的计算机科学与工程教材奖，这批引进教材中就有他的两本著作。留美中国学者 Jiawei Han 先生的《数据挖掘》是该领域中具有里程碑意义的著作。由达特茅斯学院 Thomas Cormen 和麻省理工学院、哥伦比亚大学的几

位学者共同编著的经典著作《算法导论》，在经历了 11 年的锤炼之后于 2001 年出版了第二版。目前任教于美国 Massachusetts 大学的 James Kurose 教授，曾在美国三所高校先后 10 次获得杰出教师或杰出教学奖，由他主编的《计算机网络》出版后，以其体系新颖、内容先进而倍受欢迎。在努力降低引进教材售价方面，高等教育出版社做了大量和细致的工作。这套引进的教材体现了权威性、系统性、先进性和经济性等特点。

教育部也希望国内和国外的出版商积极参与此项工作，共同促进中国信息技术教育和信息产业的发展。我们在与外商的谈判工作中，不仅要坚定不移地引进国外最优秀的教材，而且还要千方百计地将版权转让费降下来，要让引进教材的价格与国内自编教材相当，让广大教师和学生负担得起。中国的教育市场巨大，外国出版公司和国内出版社要通过扩大发行数量取得效益。

在引进教材的同时，我们还应做好消化吸收，注意学习国外先进的教学思想和教学方法，提高自编教材的水平，使我们的教学和教材在内容体系上，在理论与实践的结合上，在培养学生的动手能力上能有较大的突破和创新。

目前，教育部正在全国 35 所高校推动示范性软件学院的建设和实施，这也是加快培养信息科学技术人才的重要举措之一。示范性软件学院要立足于培养具有国际竞争力的实用性软件人才，与国外知名高校或著名企业合作办学，以国内外著名 IT 企业为实践教学基地，聘请国内外知名教授和软件专家授课，还要率先使用引进教材开展教学。

我们希望通过这些举措，能在较短的时间，为我国培养一大批高质量的信息技术人才，提高我国软件人才的国际竞争力，促进我国信息产业的快速发展，加快推动国家信息化进程，进而带动整个国民经济的跨越式发展。

教育部高等教育司

二〇〇二年三月

此为试读，需要完整PDF请访问：www.ertongbook.com

*To our parents,*
*David J. and Ruth Biermann*
*Wolfgang and Dora Ramm*

# Preface

## A Survival Kit for the Third Millennium

This book is written for people who find themselves on the outside of a high-technology world and who want to find a way in. If you have ever wondered what is meant by a Java class, a file server, an ethernet, the World Wide Web, or Pretty Good Privacy, this book shows you a path to the answers. You have been told you need a compiler, but why do you need it? What does it actually do? Watch out for certain types of problems: they are NP-complete, and you may not be able to compute the answer. You want to have a neat graphics picture jump onto your Web page when you push the button labeled "surprise." Can you make it happen? Your computer runs at 1 gigahertz: that is one billion *what* per second? By the way, what actually *is* a computer?

Most people are beginning to realize that "we are not in Kansas anymore" and that operating a spreadsheet program or clicking onto the Web is not good enough. With millions of people in the world hooking in to the high-tech network and every organization and computer connected to every other one, we need to catch up with what is going on. In our jobs, in our recreation, in our personal lives, we are going to have to live with these machines, and we need some skills and understanding to do it.

## More Than a List of Facts

This book presents the story of computer science organized around the theme of the "great ideas" of the field. These are the discoveries, the paradigms, the simplifying equations that attract the attention of everyone who comes near, and they provide the best-known paths to understanding. The book begins with an introduction to the World Wide Web and then moves to programming in Java. The theory is that if you learn to format your own Web pages with HTML, and if you can program your machine to do some interesting tasks, you will truly understand them.

Once you understand programming, you will have a notation and a vocabulary to talk about computing and will be able to dig into the mechanisms behind the facade. This is the second area of study: the hardware and software that deliver the many services to you. What are the operating system, the compiler, the browser, and the applications programs actually doing when you ask for service? Why do they act in the peculiar ways they do? What can we expect in the future? This book has chapters on computer architecture, compilation, operating systems, security mechanisms, and networks, with detailed descriptions of what they do and how they work.

The third area of study examines the limitations and challenges of the field as it exists today. What are the paradigms for understanding the hardest problems that we face? What do we currently not know how to do but have some chance of solving in the years to come? What kinds of problems are probably not solvable by computers, now or in the future?

The book as a whole is designed to give a broad overview of academic computer science with an emphasis on understanding the deep issues of the field.

## In Your Bones as well as in Your Intellect

The method of teaching is by doing rather than just by reading and talking. The theory is that if you personally go through the steps of creating a program to do something or if you hand-simulate a process, that mechanism and the related ideas will get built into your body as well as your mind. They will become instinctive as well as intellectually known. You learn about database systems by coding one in Java. You learn about system architecture by reading and writing programs in assembly language. You learn the mechanisms of a compiler by hand-compiling Java statements into assembly language. You learn to understand noncomputability by working with the proof on noncomputability and by learning to classify problems yourself as either computable or noncomputable. This is the problems-oriented approach to teaching that has been so successful in the traditional quantitative sciences.

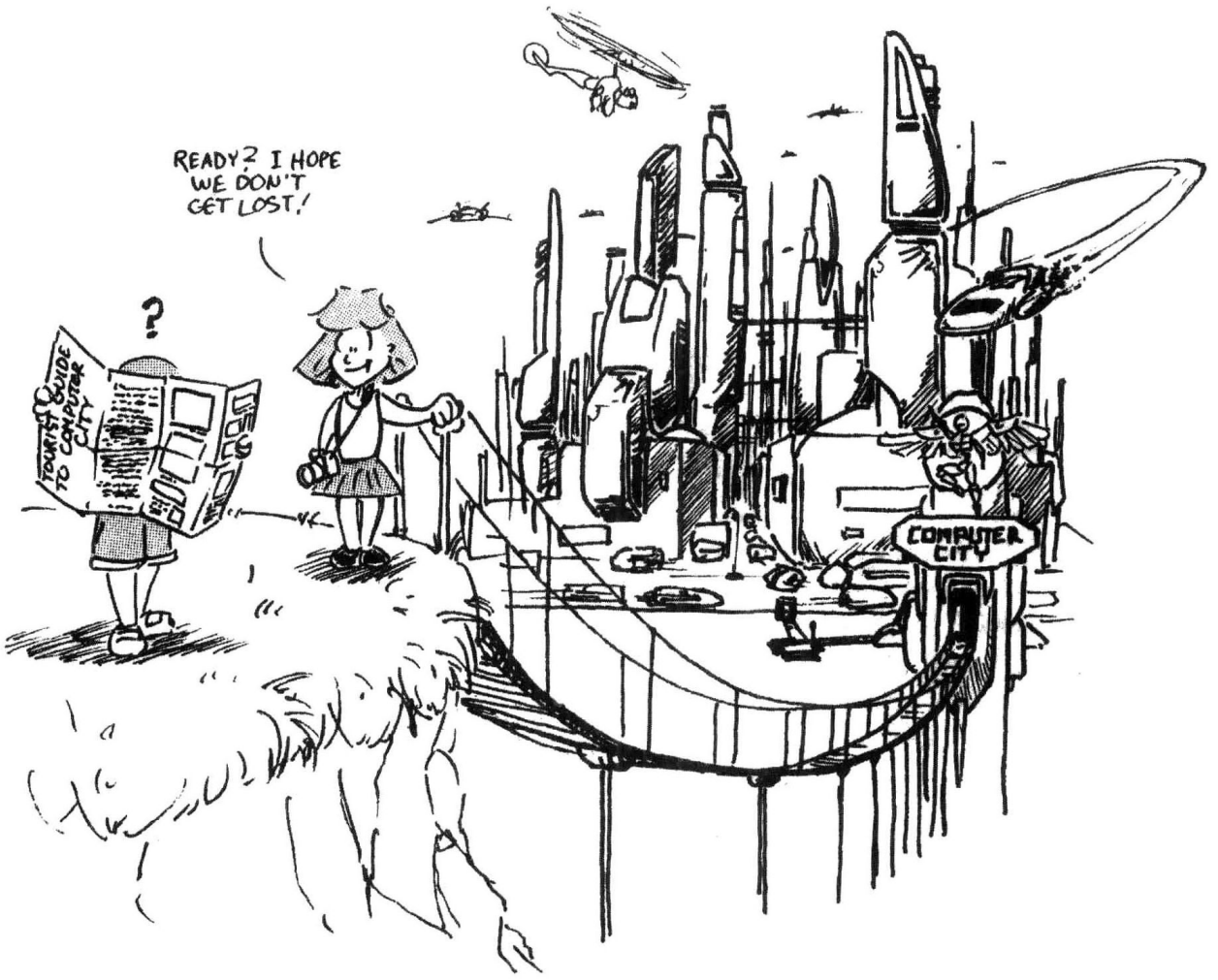
## A Course in Computer Science

This book is unusual in two ways. It covers a very broad set of topics (from HTML text formatting to program complexity and noncomputability), and it covers these topics to a relatively deep level (to the extent that students can solve problems in these domains). As such, it can be used in an introductory computer science course for students who will major in the field. These students will begin their studies with a broad view and will fit each succeeding course into a slot that has been prepared by the earlier studies. But the

book can also serve as a text in the only computer science course that some students will ever take. It provides a conceptual structure of computing and information technology that well-informed lay people should have. It supports the model of FITness (Fluency in Information Technology) described in a recent National Research Council study (Snyder et al. 1999) by covering most of the information technology concepts that the study specified for current-day fluency.

## A Thousand Heroes

This book is the product of fifteen years of experience in teaching "great ideas in computer science" at Duke University and many other institutions. The list of contributors includes many faculty and student assistants who have taught the "great ideas" approach. (See, for example, a description by Biermann of this type of course at several universities, in "Computer Science for the Many," *Computer* 27 (1994): 62–73.) Our teaching assistants have contributed extensively by helping us develop an approach to introducing Java, by writing many of the notes that eventually evolved into this book, and by developing the laboratory exercises and software for our classes. The primary contributors were Steve Myers, Eric Jewart, Steve Ruby, and Greg Keim. We owe special thanks to our faculty colleagues Owen Astrachan, Robert Duvall, Jeff Forbes, and Gershon Kedem for providing constructive critique, stimulating conversation, and technical advice. Ben Allen prepared some of the Java programs that are presented in the simulation chapter. Carrie Liken created some of the graphics in chapter 5. Matt Evans was the artist for most of the cartoons. Charlene Gordon contributed cartoons for chapters five and eleven. Other contributors have been David and Jennifer Biermann, Alice M. Gordon, Karl, Lenore, and M. K. Ramm, Jeifu Shi, Michael Fulkerson, Elina Kaplan, Denita Thomas, our several thousand students in courses at Duke, long lists of people who helped us with our earlier editions, our manuscript editors Deborah Cantor-Adams and Alice Cheyer, and, as always, our kind executive editor Robert Prior.


READY? I HOPE WE DON'T GET LOST!
?
TOURIST GUIDE TO COMPUTER CITY
COMPUTER CITY

# Studying Academic Computer Science: An Introduction

## Rumors

Computers are the subject of many rumors, and we wonder what to believe. People say that computers in the future will do all clerical jobs and even replace some well-trained experts. They say computers are beginning to simulate the human mind, to create art, to prove theorems, to learn, and to make careful judgments. They say that computers will permeate every aspect of our jobs and private lives by managing communication, manipulating information, and providing entertainment. They say that even our political systems will be altered—that in previously closed societies computers will bring universal communication that will threaten the existing order, and in free societies they will bring increased monitoring and control. On the other hand, there are skeptics who say that computer science has many limitations and that the impact of machines has been overstated.

Some of these rumors are correct and give us fair warning of things to come. Others may be somewhat fanciful, leading us to worry about the future more than is necessary. Still others point out questions that we may argue about for years without finding answers. Whatever the case, we can be sure that there are many important issues related to computers that are of vital importance, and they are worth trying to understand.

We should study computer science and address these concerns. We should get our hands on a machine and try to make it go. We should control the machine; we should play with it; we should harness it; and most important, we should try to understand how it works. We should try to build insights from our limited experiences that will illuminate answers to our questions. We should try to arm ourselves with understanding because the Computer Age is upon us.

This book is designed to help people understand computers and computer science. It begins with a study of programming in the belief that using, controlling, and manipulating machines is an essential avenue to understanding them. Then it takes readers on a guided tour of the internals of a machine, exploring all of its essential functioning from the internal

此为试读,需要完整PDF请访问: www.ertongbook.com

registers and memory to the software that controls them. Finally, the book explores the limitations of computing, the frontiers of the science as they are currently understood.

In short, the book attempts to give a thorough introduction to the field with an emphasis on the fundamental mechanisms that enable computers to work. It presents many of the "great ideas" of computer science, the intellectual paradigms that scientists use to understand the field. These ideas provide the tools to help readers comprehend and live with machines.

## Studying Computer Science

*Computer science* is the study of recipes and ways to carry them out. A recipe is a procedure or method for doing something. The science studies kinds of recipes, the properties of recipes, languages for writing them down, methods for creating them, and the construction of machines that will carry them out. Of course, computer scientists want to distinguish themselves from chefs, so they have their own name for recipes: they call them *algorithms*. But we will save most of the technical jargon for later.

If we wish to understand computer science, then we must study recipes, or algorithms. The first problem relates to how to conceive of them and how to write them down. For example, one might want a recipe for treating a disease, for classifying birds on the basis of their characteristics, or for organizing a financial savings program. We need to study some example recipes to see how they are constructed, and then we need to practice writing our own. We need experience in abstracting the essence of real-world situations and in organizing this knowledge into a sequence of steps for getting our tasks done.

Once we have devised a method for doing something, we wish to *code* it in a computer language in order to communicate our desires to the machine. Thus, it is necessary to learn a computer language and to learn to translate the steps of a recipe into commands that can be carried out by a machine. This book presents a language called *Java*, which is easy to learn and quite satisfactory for our example programs.

The combination of creating the recipe and coding it into a computer language is called *programming*, and this is the subject of the first part of the book (chapters 1–6). These chapters give a variety of examples of problem types, their associated solution methods, and the Java code, the *program*, required to solve them. Chapter 7 discusses problems related to scaling up the lessons learned here to industrial-sized programming projects.

While the completion of the programming chapters leads to an ability to create useful code, the resulting level of understanding will still fall short of our deeper goals. The programmer's view of a computer is that it is a magic box that efficiently executes commands; the internal mechanisms may remain a mystery. However, as scholars of computer science, we must know something of these mechanisms so that we can comprehend why a machine acts as it does, what its limitations are, and what improvements can be expected.

The second part of the book addresses the issue of how and why computers are able to compute.

Chapter 8 describes machine architecture and the organization of typical computers. It presents the basic hardware at the core of a computer system. Chapter 9 addresses the problem of translating a high-level computer language like Java into a lower-level language so that a program written in a high-level language can be run on a given architecture. Chapter 10 introduces concepts related to *operating systems*; these are the programs that bridge the gap between the user and the many hardware and software facilities on the machine. They make it easy for users to obtain the computing services that they want. Chapter 11 examines a topic of great concern in our networked world, computer security. As more and more of our lives become documented on machines and the connectivity of every machine to every other increases, we wonder if our lives will be secure in the new millennium. The final chapter of this section (12) introduces computer networks and the many concepts related to machines' talking to each other.

The final chapters of the book examine the limitations of computers and the frontiers of the science as it currently stands. Chapter 13 discusses problems related to program execution time and computations that require long processing times. Chapter 14 describes an attempt to speed up computers to take on larger problems, the introduction of parallel architectures. Chapter 15 discusses the existence of so-called noncomputable functions, and chapter 16 gives an introduction to the field of artificial intelligence.

A great many programs have been developed to illustrate ideas in this book, and you can obtain them via the Internet. They can be found at Biermann's World Wide Web page at the Department of Computer Science, Duke University (http://www.cs.duke.edu/~awb).

## An Approach for Nonmathematical Readers

A problem that arises in the teaching of computer science is that many instructors who know the field tend to speak in technical language and use too much mathematical notation for lay people to understand. Then the difficulty in communication leads them to conclude that ordinary people are not able to understand computer science. Thus, books and university courses often skirt the central issues of computer science and instead teach the operation of software packages or the history and sociology of computing.

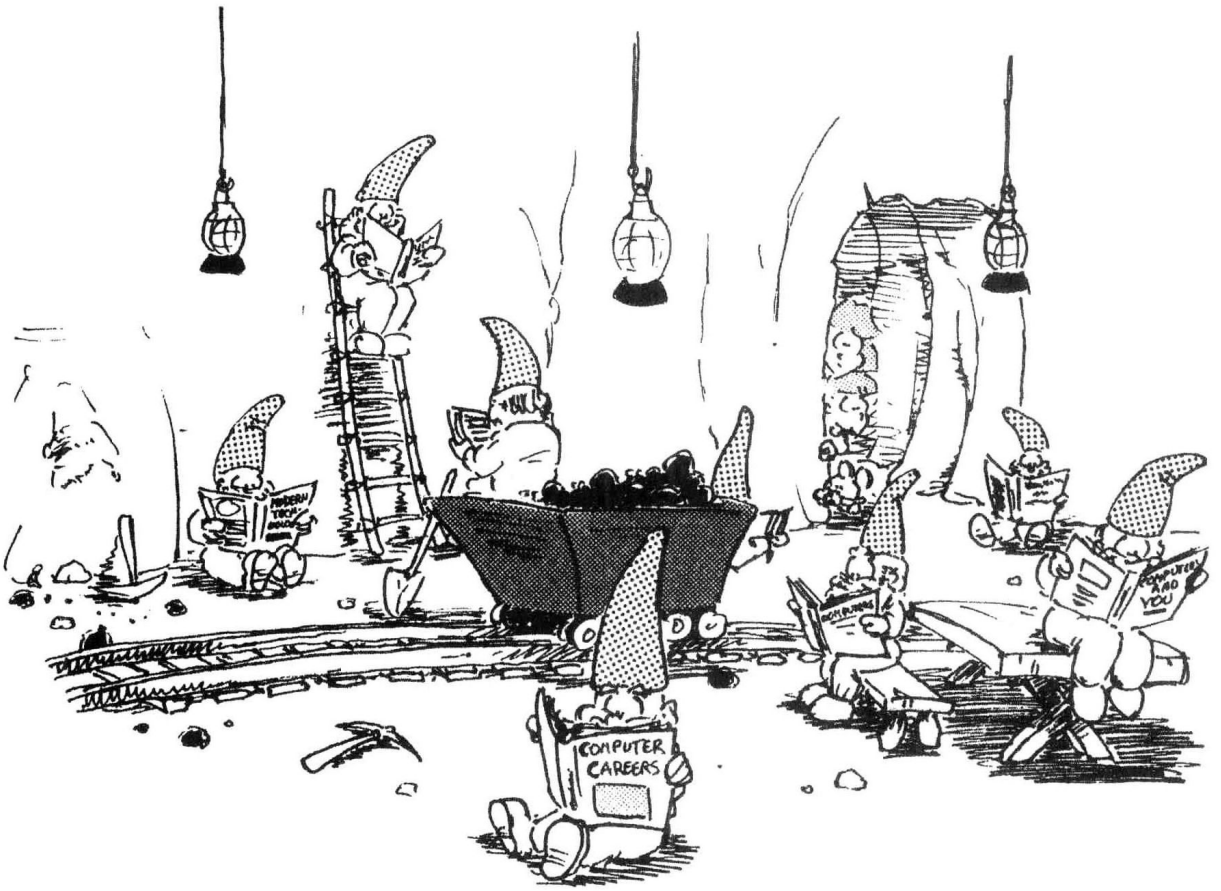
This book was written on the assumption that intelligent people can understand every fundamental issue of computer science if preparation and explanation are adequate. No important topics have been omitted because of "difficulty." However, tremendous efforts were made to prune away unnecessary details from the topics covered and to remove special vocabulary except where careful and complete definitions are given.

Because casual readers may not wish to read every part of every chapter, the book is designed to encourage dabbling. Readers are encouraged to jump to any chapter at

any time and read as much as is of interest. Of course, most chapters use some concepts gathered from earlier pages, and where this occurs, understanding will be reduced by reading chapters selectively. The programming chapters (1–6) are highly dependent on each other, and the architecture chapter (8) should be read before the translation chapter (9). Also, some of the advanced chapters (13–16) use concepts of programming from earlier chapters (1–5). Except for these restrictions, the topics can probably be covered in any order without much sacrifice.

An overview of what the book contains could be obtained in a single evening by reading the introductory (first) and summary (last) sections of each chapter. The intermediate sections contain the primary material of the book and may require substantial time and effort to read. However, sections with an asterisk before the title could be skipped without loss of understanding of the major lessons: they supplement the main chapter text by treating some points in greater detail.

# Great Ideas in Computer Science with Java


COMPUTER
CAREERS
COMPUTERS AND YOU

责任编辑　康兆华
封面设计　张　楠
责任印制　陈伟光

此为试读，需要完整PDF请访问：www.ertongbook.com