Chapter 0

Introduction

Concepts:
- ▷ Approaches to this material
- ▷ Principles

This is an important notice.
Please have it translated.
—The Phone Company

YOUR MOTHER probably provided you with constructive toys, like blocks or Tinkertoys[1] or Lego bricks. These toys are educational: they teach us to think spatially and to build increasingly complex structures. You develop modules that can be stuck together and rules that guide the building process.

If you are reading this book, you probably enjoyed playing with constructive toys. You consider writing programs an artistic process. You have grown from playing with blocks to writing programs. The same guidelines for building structures apply to writing programs, save one thing: there is, seemingly, no limit to the complexity of the programs you can write. *I lie.*

Well, almost. When writing large programs, the *data structures* that maintain the data in your program govern the space and time consumed by your running program. In addition, large programs take time to write. Using different structures can actually have an impact on how long it takes to *write* your program. Choosing the wrong structures can cause your program to run poorly or be difficult or impossible to implement effectively.

Thus, part of the program-writing process is choosing between different structures. Ideally you arrive at solutions by analyzing and comparing their various merits. This book focuses on the creation and analysis of traditional data structures in a modern programming environment, The Java Programming Language, or Java for short.

0.1 Read Me

As might be expected, each chapter is dedicated to a specific topic. Many of the topics are concerned with specific data structures. The structures we will investigate are abstracted from working implementations in Java that are available to you if you have access to the Internet.[2] Other topics concern the "tools of the

[1] All trademarks are recognized.
[2] For more information, see `http://www.cs.williams.edu/JavaStructures`.

trade." Some are mathematical and others are philosophical, but all consider the process of programming well.

The topics we cover are not all-inclusive. Some useful structures have been left out. Instead, we will opt to learn the *principles of programming data structures*, so that, down the road, you can design newer and better structures yourself.

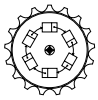
Perhaps the most important aspect of this book is the set of problems at the end of each section. *All are important for you to consider*. For some problems I have attempted to place a reasonable hint or answer in the back of the book. Why should you do problems? Practice makes perfect. I could show you how to ride a unicycle, but if you never practiced, you would never learn. If you study and understand these problems, you will find your design and analytical skills are improved. As for your mother, she'll be proud of you.

Unicycles: the ultimate riding structure.

Sometimes we will introduce problems in the middle of the running text—these problems do not have answers (sometimes they are repeated as formal problems in the back of the chapter, where they *do* have answers)—they should be thought about carefully as you are reading along. You may find it useful to have a pencil and paper handy to help you "think" about these problems on the fly.

Exercise 0.1 *Call*[3] *your Mom and tell her you're completing your first exercise. If you don't have a phone handy, drop her a postcard. Ask her to verify that she's proud of you.*

This text is brief and to the point. Most of us are interested in experimenting. We will save as much time as possible for solving problems, perusing code, and practicing writing programs. As you read through each of the chapters, you might find it useful to read through the source code online. As we first consider the text of files online, the file name will appear in the margin, as you see here. The top icon refers to files in the `structure` package, while the bottom icon refers to files supporting examples.



`Structure`

`Example`

One more point—this book, like most projects, is an ongoing effort, and the latest thoughts are unlikely to have made it to the printed page. If you are in doubt, turn to the website for the latest comments. You will also find online documentation for each of the structures, generated from the code using `javadoc`. It is best to read the online version of the documentation for the most up-to-date details, as well as the documentation of several structures not formally presented within this text.

0.2 He Can't Say That, Can He?

Sure! Throughout this book are little political comments. These remarks may seem trivial at first blush. Skip them! If, however, you are interested in ways

[3] Don't e-mail her. Call her. Computers aren't everything, and they're a poor medium for a mother's pride.

to improve your skills as a programmer and a computer scientist, I invite you to read on. Sometimes these comments are so important that they appear as *principles*:

Principle 1 *The principled programmer understands a principle well enough to form an opinion about it.*

Self Check Problems

Solutions to these problems begin on page 441.

0.1 Where are the answers for "self check" problems found?

0.2 What are features of large programs?

0.3 Should you read the entire text?

0.4 Are *principles* statements of truth?

Problems

Solutions to the odd-numbered problems begin on page 451.

0.1 All odd problems have answers. Where do you find answers to problems? (Hint: See page 451.)

0.2 You are an experienced programmer. What five serious pieces of advice would you give a new programmer?

0.3 Surf to the website associated with this text and review the resources available to you.

0.4 Which of the following structures are described in this text (see Appendix D): `BinarySearchTree`, `BinaryTree`, `BitSet`, `Map`, `Hashtable`, `List`?

0.5 Surf to `http://www.javasoft.com` and review the Java resources available from Sun, the developers of Java.

0.6 Review documentation for Sun's `java.util` package. (See the Core API Documentation at `http://www.javasoft.com`.) Which of the following data structures are available in this package: `BinarySearchTree`, `BinaryTree`, `BitSet`, `Dictionary`, `Hashtable`, `List`?

0.7 Check your local library or bookstore for Java reference texts.

0.8 If you haven't done so already, learn how to use your local Java programming environment by writing a Java application to write a line of text. (Hint: Read Appendix B.)

0.9 Find the local documentation for the `structure` package. If none is to be found, remember that the same documentation is available over the Internet from `http://www.cs.williams.edu/JavaStructures`.

0.10 Find the examples electronically distributed with the `structure` package. Many of these examples are discussed later in this text.