Preface

You will find this book to be somewhat unusual. Most computer science texts will begin with a section on the history of computers and then with a flurry of definitions that are just "so many words" to the average student. My approach with Blue Pelican Java is to first give the student some experience upon which to hang the definitions that come later, and consequently, make them more meaningful.

This book does have a history section in Appendix S and plenty of definitions later when the student is ready for them. If you will look at Lesson 1, you will see that we go right to work and write a program the very first day. The student will not understand several things about that first program, yet he can immediately make the computer do something useful. This work ethic is typical of the remainder of the book. Rest assured that full understanding comes in time. Abraham Lincoln himself subscribed to this philosophy when he said, "Stop petting the mule, and load the wagon."

The usual practice in most Java textbooks of introducing classes and objects alongside the fundamental concepts of primitive variable types, loops, decision structures, etc. is deferred until the student has a firm grasp of the fundamentals. Thus, the student is not overwhelmed by **simultaneous** introduction of OOPs (Object Oriented Programming) and the fundamentals. Once introduced, (Lesson 15), OOPs is heavily emphasized for the remainder of the book.

I fully realize that there are those who disagree with this idea of deferring the introduction of OOPs, and from their own point of view, they are right. In most cases they teach only the very highest achieving, mature students. In those cases, I agree that it is acceptable to begin with OOPs; however, for the average student and especially for younger high school students, I feel that they need to understand the fundamentals first.

Upon first examination of this book it may not appear to be very "robust" in that there is not great depth for some of the topics. Actually the depth **is** there,… in the Appendix. The Appendix for this book is unusually large. Here is why the book is organized this way:

- The lessons are kept purposely short so as to hold down the intimidation factor. As a result, the lessons should look "doable" to the students.
- The in-depth material is placed in the Appendices, and references to the Appendices are made in the lessons. As an example, in Lesson 18 the *split* method is introduced. The *split* method uses regular expressions that are briefly discussed there; however, the in-depth presentation of regular expressions is placed in Appendix AC.

Unfortunately, this book does not introduce any graphics or windows programming. The 57 lessons in this book can be covered in one school year, but just barely. To prepare students for the AP test (and contests) there is only time to cover the essentials presented in this book. Check http://www.bluepelicanjava.com for the availability of study materials for the current **AP case study**, updates on this book, **videos** for each lesson, and an inexpensive way to purchase hard-cover books.

I am often asked **how to use this book**. "Which lessons are really important and which can be skipped?" The answer is simple:

- **Start on Lesson 1.**
- **Proceed at a reasonable rate.** (See Appendix P for a time-line.)
- **Don't skip anything** (except for, perhaps Lesson 23, Lesson 48 and Lesson 54)
- **Give a simple, confidence-building quiz on each lesson.** Quizzes and keys are provided in the Answer Book (available at www.bluepelicanjava.com).
- **Make sure the students do the provided exercises and projects.**
- **Give tests at regular intervals.** Tests and keys are provided in the Answer Book.

In this book you will also notice another part of my philosophy of teaching and educational material in general…**Keep it simple**… I try to keep things as simple and uncluttered as possible. For example, you will find specific examples in greater numbers than long-winded explanations in this book. You won't find many pictures and sidebars and lots of little colored side notes scattered about. Some of that type format does contain some useful information; however, I feel that it is largely distracting. Apparently more and more people are coming around to my way of thinking on this, and here is why I think so. Recall that just a few years ago that nearly all web pages looked like cobbled together ransom notes with just a profusion of colors, links, and tidbits scattered all over the page. Take a look at professional web pages today. They typically have a very neat, clean appearance…often with just a plain white background and with plenty of space between the various elements. This is good. Simple is better.

Since this textbook has a strong emphasis on preparation for the AP test and competition (computer science contests), special "contest type" problems are provided at the end of most lessons. I realize that most students will not compete and some may not even take the AP exam; however, the material is not wasted on them. Those "contest type" problems are good for the average student too, as long as they are not overwhelmed with too many problems at one sitting. Hopefully, I have just the optimum number of these type problems on each lesson and students won't be burned-out by too much of a good thing.

Finally, we come to the reason for the choice of Blue Pelican Java as a name for this book. One of the early (and free) java IDE's available for students was BlueJ and it was the first my students used. I always thought BlueJ was an elegant name and had expressed a desire to a colleague to continue the tradition by naming the book after some other blue-colored bird. He jokingly suggested Blue Pelican, not really being serious about naming a book after this rather ungainly, clunky bird. For the lack of an existing name for the book during development, it continued to be called Blue Pelican. If you call something by a particular name long enough, that's its name, and so the name stuck.

I truly hope Blue Pelican Java is useful to you and that you find the experience of learning to program a rewarding one. Just remember, few things worthwhile are acquired without some sacrifice. The "sacrifice" here will be the time you invest in creating programs and trying the code suggested in these pages.

Charles E. Cook