Preface

John M. Morrison

July 31, 2015

Contents

This is an introduction to the creation of applications in the object-oriented language Java. It assumes basic knowledge of procedural programming. The predecessor volume *Computing in Python* provides that knowledge; various programming constructs in Java are explained in terms of Python and appropriate comparisions are drawn.

The change in language is deliberate. It is important for the student to see the language-invariant features of computing. It is interesting to see that different languages are suited to solving problems in different domains.

Why Java? One obvious reason is that as of now, Java is the language for the Advanced Placement exam. This is a small but convenient reason. The topics in this book far transcend the AP syllabus and the book addresses vital matters such as file manipulation, exception handling, and graphical programming that are ommitted from the AP syllabus.

More importantly, it is important for programmers to be able to deal with staticially typed languages, as these can yield object code that is both small and very efficient. Java, like Python is strongly typed. This forces a different set of habits of mind when creating functions and when building programs. Java is more forgiving than but in many ways similar to C, which is a language that every serious student of computing must learn. It is also a very widely used application development language used in the software development world.

Java has a clean and very pleasing object-oriented programming model. Mastering Java's model makes it easy to go on and learn any other object-oriented langage such as C++.

Java has a highly developed set of classes for creating event-driven programs that operate with a graphical user interface. The Java Swing libary provides a wealth of widgets to place in applications and makes it easy to attach code to the widgets so that they become functional.

What this book is not Just like its older sibling, *Computing in Python*, this book is not an encyclopaedia. It is not a reference on the Java language. It is not an exhaustive technical reference of any ilk. It is unabashedly designed as a learning tool.

It will not magically transmogrify you into a master Java Ninja without a lot of work on your part. You will need to spend a lot of time designing, creating, debugging and deploying programs.

For whom this book is written Are you open-minded? Are you creative? Do you know how to program in a Turing-complete programming language? Do you have a sense of adventure? Are you unafraid of failing? If you can answer yes to these question, then course is set. Tell the engine room, "Full steam ahead!" Plenty of peril, adventure, and fun lie ahead.

This book is unabashedly designed as a learning tool. It will blaze a path for you through the exciting worlds of object-oriented and event-driven programming. It will enable you to write modern, professional-looking programs that withstand the abuse of the unholy end-user.

You will learn how to read the Java API guide and learn how to use it to integrate new features into your programs. This is an essential skill for every software developer. Reading and understanding this book will enable you to understand and profitably read more advanced treatments of the language.

Approach We recommend the freely available Dr. Java Integrated Development Environment to get started. At some time, you will may want to use a more sophisticated tool, such as Eclipse. Dr. Java enjoys some features that make programming for beginners more fun and easier.

The Getting Started chapter tells you how to prepare your lappy or compy for Java programming. If you need more detail, the Dr. Java manual gives very detailed instructions, but what we show here is enough to get your machine properly set up.

Chapters 1-3 introduce the Java programming model and the fundamentals of the language. Having this under control, we turn to a case study in Chapter 4 that creates a new data type for extended-precision rational arithmetic. This shows a full-blown example of a useful API being designed and created from scratch. This is the first and most important step in becoming a Java programmer: you must be able to create and API and document it so that another programmer can use it to solve problems. Chapters 1-4 emphasize programming in the small. You create a class and then use it to solve other problems.

With Chapter 5, we begin programming in the large. This chapter discusses the various relationships between classes. It uses the creation of GUI programs as a means of motivating the ideas of inheritance and interface. Then, in Chapter

6, we turn to another case study, `Tricolor.java`, that creates a very simple GUI application. Again, this is created from scratch and the student gets to see the idea evolve from a loose set of ideas to a working application.

Chapter 7 discusses inner classes and listeners. The Java 8 construct of lambdas is used to implement listeners and to shorten and simplify code. We then turn to exception handling in Chapter 8 and do yet another case study that builds an app for mixing colors in RGB.

Finally, in chapters in 9, 10 and 11, we discuss fileIO, and then do two case studies, `Nitpad.java`, which is a re-creation of Notepad, and `UniDraw.jav`, which is a simple draw program. Both of these apps are full-featured. They can save work in a file, retrieve prior sessions from files and they do not crash in the face of user abuse. Each has a full complement of appropriate GUI features.