Chapter 7 continues to explore Java's basic APIs by focusing on reflection, string management, the `System` class, and threading.

Chapter 8 focuses exclusively on Java's collections framework, which provides you with a solution for organizing objects in lists, sets, queues, and maps.

Chapter 9 continues to explore Java's utility APIs by introducing you to the concurrency utilities, internationalization, preferences, random number generation, and regular expressions.

Chapter 10 is all about input/output (I/O). In this chapter, you explore Java's classic I/O support in terms of its `File` class, `RandomAccessFile` class, various stream classes, and various writer/reader classes. My discussion of stream I/O includes coverage of Java's object serialization and deserialization mechanisms.

Note: This book largely discusses APIs that are common to Java SE and Android. However, it diverges from this practice in Chapter 9, where I use the Swing toolkit to provide a graphical user interface for one of this chapter's internationalization examples. (Android does not support Swing.)

After you complete this book, I recommend that you obtain a copy of *Beginning Android 2* by Mark L Murphy (Apress, 2010; ISBN: 1430226293) and start learning how to develop Android apps. In that book, "you'll learn how to develop applications for Android 2.x mobile devices, using simple examples that are ready to run with your copy of the JDK."

Note: Over the next few months, I will make available at my `javajeff.mb.ca` website six additional PDF-based chapters. These chapters will introduce you to more Java APIs (such as networking and database APIs) that I could not discuss in this book because the book has greatly exceeded its initial 400-page estimate (and the good folks at Apress have been gracious enough to let me do so, but there are limits). I present more information about these PDF files at the end of Chapter 10's "Summary" section.

Thanks for purchasing my book. I hope you find it a helpful preparation for, and I wish you lots of success in achieving, a satisfying and lucrative career as an Android app developer.

Jeff "JavaJeff" Friesen, August 2010

Note: You can download this book's source code by pointing your web browser to `www.apress.com/book/view/1430231564` and clicking the Source Code link under Book Resources. Although most of this code is compilable with Java version 6, you will need Java version 7 to compile one of the applications.

Chapter 1 introduces you to Java by first focusing on Java's dual nature (language and platform). It then briefly introduces you to Sun's/Oracle's Java SE, Java EE, and Java ME editions of the Java development software, as well as Google's Android edition. You next learn how to download and install the Java SE Development Kit (JDK), and learn some Java basics by developing and playing with a pair of simple Java applications. After receiving a brief introduction to the NetBeans and Eclipse IDEs, you learn about application development in the context of *Four of a Kind*, a console-based card game.

Chapter 2 starts you on an in-depth journey of the Java language by focusing on language fundamentals (such as types, expressions, variables, and statements) in the contexts of classes and objects. Because applications are largely based on classes, it is important to learn how to architect classes correctly, and this chapter shows you how to do so.

Chapter 3 adds to Chapter 2's pool of object-based knowledge by introducing you to those language features that take you from object-based applications to object-oriented applications. Specifically, you learn about features related to inheritance, polymorphism, and interfaces. While exploring inheritance, you learn about Java's ultimate superclass. Also, while exploring interfaces, you discover the real reason for their inclusion in the Java language; interfaces are not merely a workaround for Java's lack of support for multiple implementation inheritance, but serve a higher purpose.

Chapter 4 introduces you to four categories of advanced language features: nested types, packages, static imports, and exceptions. While discussing nested types, I briefly introduce the concept of a closure and state that closures will appear in Java version 7 (which many expect to arrive later this year).

Note: I wrote this book several months before Java version 7's expected arrival in the fall of 2010. Although I have tried to present an accurate portrayal of version 7–specific language features, it is possible that feature syntax may differ somewhat from what is presented in this book. Also, I only discuss closures briefly because this feature was still in a state of flux while writing this book. For more information about closures and other functional programming concepts (such as lambdas) being considered for Java version 7, I recommend that you check out articles such as "Functional Programming Concepts in JDK 7" by Alex Collins (`http://java.dzone.com/articles/lambdas-closures-jdk-7`).

Chapter 5 continues to explore advanced language features by focusing on assertions, annotations, generics, and enums. Although the topic of generics has brought confusion to many developers, I believe that my discussion of this topic will clear up much of the murkiness. Among other items, you learn how to interpret type declarations such as `Enum<E extends Enum<E>>`.

Chapter 6 begins a trend that focuses more on APIs than language features. This chapter first introduces you to many of Java's math-oriented types (such as `Math`, `StrictMath`, `BigDecimal`, and `BigInteger`), and also introduces you to Java's `strictfp` reserved word. It then looks at the `Package` class, primitive wrapper classes, and the References API.

Introduction

Smartphones and other touch-based mobile devices are all the rage these days. Their popularity is largely due to their ability to run *apps*. Although the iPhone and iPad with their growing collection of Objective-C-based apps are the leaders of the pack, Android-based smartphones with their growing collection of Java-based apps are proving to be a strong competitor.

Not only are many iPhone/iPad developers making money by selling their apps, many Android developers are also making money by selling similar apps. According to tech websites such as *The Register* (`www.theregister.co.uk/`), some Android developers are making lots of money (`www.theregister.co.uk/2010/03/02/android_app_profit/`).

In today's tough economic climate, perhaps you would like to try your hand at becoming an Android developer and make some money. If you have good ideas, perseverance, and some artistic talent (or perhaps know some talented individuals), you are already part of the way toward achieving this goal.

Tip: A good reason to consider Android app development over iPhone/iPad app development is the lower startup costs that you will incur with Android. For example, you do not need to purchase a Mac on which to develop Android apps (a Mac is required for developing iPhone/iPad apps); your existing Windows, Linux, or Unix machine will do nicely.

Most importantly, you will need to possess a solid understanding of the Java language and foundational application programming interfaces (APIs) before jumping into Android. After all, Android apps are written in Java and interact with many of the standard Java APIs (such as threading and input/output APIs).

I wrote *Learn Java for Android Development* to give you a solid Java foundation that you can later extend with knowledge of Android architecture, API, and tool specifics. This book will give you a strong grasp of the Java language and many important APIs that are fundamental to Android apps and other Java applications. It will also introduce you to key development tools.

Learn Java for Android Development is organized into ten chapters and one appendix. Each chapter focuses on a collection of related topics and presents a set of exercises that you should complete to get the most benefit from the chapter's content. The appendix provides the solutions to each chapter's exercises.