

# 17.1 Introduction

1. We were privileged to have Brian Goetz, Oracle's Java Language Architect and Specification Lead for Java SE 8's Project Lambda, and co-author of *Java Concurrency in Practice*, do a detailed review of Java How to Program, 10/e. He thoroughly scrutinized the version of this chapter that appeared in that edition and provided many additional suggestions that are reflected in this new edition, Java How to Program, 11/e. Any remaining faults in the book are our own.

8

The way you think about Java programming is about to change profoundly. Prior to Java SE 8, Java supported three programming paradigms—*procedural programming*, *object-oriented programming* and *generic programming*. Java SE 8 added *lambdas* and *streams*<sup>2</sup>—key technologies of *functional programming*.

2. The streams we discuss in this chapter are not the same as the input/output streams we discuss in [Chapter 15](#), Files, Input/Output Streams, NIO and XML Serialization, in which a program reads a stream of bytes from or writes a stream of bytes to a file. [Section 17.13](#) uses lambdas and streams to manipulate the contents of a file.

In this chapter, we'll use lambdas and streams to write certain kinds of programs faster, simpler, more concisely and with fewer bugs than with previous techniques. In [Chapter 23](#), Concurrency, you'll see that such programs can be easier to *parallelize* (i.e., perform multiple operations simultaneously) so that you can take advantage of multi-core architectures to enhance performance—a key goal of lambdas and streams.



# Software Engineering Observation 17.1

*You'll see in [Chapter 23](#), Concurrency that it's hard to create parallel tasks that operate correctly if those tasks modify a program's state (that is, its variables' values). So the techniques that you'll learn in this chapter focus on **immutability**—not modifying the data source being processed or any other program state.*

This chapter presents many examples of lambdas and streams ([Fig. 17.1](#)), beginning with several showing better ways to implement tasks you programmed in [Chapter 5](#). The first several examples are presented in a manner that allows them to be covered in the context of earlier chapters. For this reason, some terminology is discussed later in this chapter. [Figure 17.2](#) shows additional lambdas and streams coverage in later chapters.

Section	May be covered after
<p>Sections 17.2–17.4 introduce basic lambda and streams capabilities that process ranges of integers and eliminate the need for counter-controlled repetition.</p>	<p><a href="#">Chapter 5</a>, Control Statements: Part 2; Logical Operators</p>
<p>Section 17.6 introduces method references and uses them with lambdas and streams to process ranges of integers</p>	<p><a href="#">Chapter 6</a>, Methods: A Deeper Look</p>

<p>Section 17.7 presents lambda and streams capabilities that process one-dimensional arrays.</p>	<p><u>Chapter 7,</u> Arrays and <b>ArrayLists</b></p>
<p>Sections 17.8–17.9 discuss key functional interfaces and additional lambda concepts, and tie these into the chapter’s earlier examples. Section 10.10 introduced Java SE 8’s enhanced interface features (<code>default</code> methods, <code>static</code> methods and the concept of functional interfaces) that support functional-programming techniques in Java.</p>	<p><u>Chapter 10,</u> Object- Oriented Programming: Polymorphism and Interfaces</p>
<p>Section 17.16 shows how to use a lambda to implement a JavaFX event-listener functional interface.</p>	<p><u>Chapter 12,</u> JavaFX Graphical User Interfaces: Part 1,</p>
<p>Section 17.11 shows how to use lambdas and streams to process collections of <code>String</code> objects.</p>	<p><u>Chapter 14,</u> Strings, Characters and Regular Expressions</p>
<p>Section 17.13 shows how to use lambdas and streams to process lines of text from a file—the example in this section also uses some regular expression capabilities from <u>Chapter 14</u>.</p>	<p><u>Chapter 15,</u> Files, Input/Output Streams, NIO and XML Serialization</p>

## Fig. 17.1

This chapter’s lambdas and streams discussions and examples.

Coverage	Chapter
Uses lambdas to implement Swing event-listener functional interfaces.	Chapter 35, Swing GUI Components: Part 2

Shows that functional programs are easier to parallelize so that they can take advantage of multi-core architectures to enhance performance. Demonstrates parallel stream processing. Shows that <code>Arrays</code> method <code>parallelSort</code> can improve performance on multi-core vs. single-core architectures when sorting large arrays.	<u>Chapter 23,</u> Concurrency
Uses lambdas to implement Swing event-listener functional interfaces.	<u>Chapter 26,</u> Swing GUI Components: Part 1
Uses streams to process database query results.	<u>Chapter 29,</u> Java Persistence API (JPA)

## Fig. 17.2

Later lambdas and streams coverage.