Introduction to Coroutines in Kotlin

> *Coroutines, a more recent addition to the language, are one of the most exciting features of Kotlin. Coroutines are built on the powerful concept of continuations and form the foundation for concurrent and asynchronous programming. In this part, you'll learn about suspension points and how to alter the thread and sequence of execution of code. Then you'll readily apply the techniques you learn to create high-performant asynchronous applications.*

No one likes to wait for a response to questions like "What's the weather like today?" You fire off that request to your smart device and continue doing your chore; and when the response arrives you take that in and move forward. The code we write should also do the same—that is, be non-blocking, especially when calling tasks that may take some time to run. That's where coroutines come in.

Coroutines are new in Kotlin—they're part of the standard library starting with version 1.3, and they provide a great way to create concurrent non- blocking code. Coroutines go hand in hand with suspendible functions, the execution of which may be suspended and resumed. These features are built in Kotlin using continuations, which are data structures used to preserve the internal state of a function in order to continue the function call later on.

In this chapter we'll focus on the nuts and bolts of coroutines. You'll learn how to configure code to run sequentially or concurrently, understand the relationship between threads and coroutines, control the thread of execution, and debug coroutines.

We'll also take a peek under the hood at continuations, which provide the underpinnings for coroutines, and we'll apply the knowledge to create infinite sequences and iterators that yield unbounded collections of data. The concepts you learn in this chapter will provide the foundations necessary for asynchronous programming, which we'll dig into in the next chapter.