Wrapping Up

Coroutines not only provide a great way to program asynchronous execution, but they also offer a sensible and easy way to deal with exceptions. Coroutines nicely help keep the code structure of synchronous, sequential code similar to concurrent and asynchronous code. This reduces the cost of making code change when concurrency and asynchronous operations are necessary, and also makes it easier to reason, debug, and maintain code. In addition to paving the way for efficient execution, mapping a complex network of tasks onto a hierarchy of coroutines makes it easier to manage the lifecycle of execution. You can control the duration of execution using timeouts and also set up supervisory jobs to control the interaction between the coroutines in the hierarchy.

In this chapter, in addition to learning these nuances of coroutines, you've also seen how to asynchronously fetch data from a remote service.

In the next chapter, we'll focus on integrating Kotlin with Java and apply the language capabilities to build Spring and Android applications.