Wrapping Up

External iterators are common in imperative-style programming, whereas internal iterators are the way of life in functional programming. Internal iterators are less complex, more fluent, expressive, and concise when compared to external iterators. Kotlin provides internal iterators directly on collections. However, the execution of these methods on collections is eager. This isn't an issue for small collections, but for larger collections, it may result in poor performance due to evaluations that may not be necessary. For such situations, Kotlin provides sequences as wrappers that will postpone evaluations. By lazily evaluating operations, you may eliminate computations that are otherwise not needed, resulting in better performance while retaining all the other benefits of internal iterators.

Internal iterators are concise, expressive, and elegant, but Kotlin takes the characteristics to a whole new level with its capabilities to create fluent code. That's the topic we'll explore in the next chapter.