Introducing Collections

Effective Programming in Scala

Introducing Collections

We've already met the `List` collection. We'll now learn about other collection types. In particular we will see how to

- construct;
- query; and
- transform

collections.

We will then learn about specific features of sequences and maps.

We will also encounter tuples and `Option`, which are not part of the collection classes but are very useful.

Introducing Collections

We've already met the `List` collection. We'll now learn about other collection types. In particular we will see how to

- construct;
- query; and
- transform

collections.

We will then learn about specific features of sequences and maps.

We will also encounter tuples and `Option`, which are not part of the collection classes but are very useful.

Note: the collection library is large. We'll only cover the most common cases.

Three Examples of Collections

We'll use three different collection types to illustrate the commonality and differences between collections:

- `List`, which is an immutable sequence;
- `ArrayBuffer`, which is a mutable sequence; and
- `Map`, which is an immutable dictionary.

Three Examples of Collections

We'll use three different collection types to illustrate the commonality and differences between collections:

- `List`, which is an immutable sequence;
- `ArrayBuffer`, which is a mutable sequence; and
- `Map`, which is an immutable dictionary.

A dictionary is also known as a *map*, *hash table*, or *associative array*. It is a datastructure that allows us to store and retrieve values by key. For example, we could store users keyed by their user id, and then quickly retrieve a user given an id.

Collection Packages

Collections live in the `scala.collection` package:

- `scala.collection.immutable` for the immutable collections; and
- `scala.collection.mutable` for the mutable ones.

By default most of the immutable collections are already available to use. However we need to import the mutable collections before we can use them.

Importing Mutable Collections

A common practice is to import the `mutable` package directly.

```
import scala.collection.mutable
```

Then we refer to definitions by prefixing them with the package.

```
val buffer = mutable.ArrayBuffer()
```

This convention makes it clear when we're creating a mutable collection.

We will learn more about imports in week 3.

Introduction: Summary

We started our introduction to collection by looking at three examples of collection types: `List`, `ArrayBuffer`, and `Map` and learned about the packages in which they are found.