Week 5: Collections

Collections in Scala

- Collections are an important tool in high-level programming.
- Scala provides a large and flexible framework for managing all types of collections and doing comprehensions on them.

Collections in Scala

- All of Scala's collections are found inside the `scala.collection` package; some of them are imported automatically.
- The root collections package has two major subpackages, `mutable` and `immutable`. These house mutable and immutable collections, respectively.

`Traversable`

- `scala.collection.Traversable` is the "root" of all collection types (it has various supertraits, but these are mostly for generifying the framework).
- Traversable contains most of the common comprehension operations, such as `map`, `filter`, and `foreach`.

`Iterable`

- Below `Traversable` the inheritance tree breaks into the mutable and immutable collections.
- `Iterable` is similar to `Traversable`, except it has a method for obtaining an Iterator over its elements.
- Most of the `Traversable` methods are implemented in `Iterable` via the iterator method.

`Seq`, `Set`, `Map`

- On the immutable side, below `Iterator` are `Seq`, `Set`, and `Map`, which represent a list of objects, an unordered set of unique objects, and an associative array.
- The default implementations for these are `Vector`, `HashSet`, and `HashMap`.

Seq

- `Seq`s are ordered collections. `Seq`s can be `IndexedSeq` (which have fast element access and length), like `Vector`, or `LinearSeq` (which have fast head, tail, and isEmpty), like `List`.

- In addition to `Seq`, there is `Array`, which is not a `Seq` but rather sugar for Java's native indexed arrays. There is, however, an implicit `Array`-to-`Seq` conversion.

`Set`

- A `Set` is a collection to which elements can be added or removed (removal returns a new collection), and for which presence of a value can be tested.
- `Set`s can also be unioned and intersected.

Map

- Maps are "associative arrays", or arrays that are indexed by objects rather than numbers.
- Maps are important for many programs that need to keep data connected to something like a `String` name.

Mutable Collections

- While collections in Scala cannot change after being created, they can still have comprehensions done on them, which is the functional style of collections.
- Mutable collections can be changed, and are often used in more object oriented contexts. Most immutable collections have mutable versions, and the two can often be converted from each other.

`Option`

- `scala.Option` is a special kind of "collection", that represents a value that might not exist.
- All `Options` are either a `Some` that contains a value, or the `None` object that represents absence.
- `Option` is used in Scala where `null` would be used in Java, but it avoids many of `null`'s silent errors that crash at runtime.
- `Option` can have comprehensions done on it just like other collections.

Tuples

- Tuples are another special collection, that contain a set number of objects.
- Tuples are different from Lists in that the type of each element is part of the tuple's type: the type of `(1, "foo", new AnyRef)` is `(Int, String, AnyRef)`.
- Tuples are also closely related to function objects.

`Iterator` and `Stream`

- An `Iterator` is a special collection that is used to look at each element of an `Iterable`.
- `Stream` is a special Iterator that evaluates lazily, making it useful for certain kinds of comprehensions. Many `Iterator` companion functions create `Stream`s.