EPFL

Sequences and Maps

Effective Programming in Scala

Sequences and Maps

In this section we’ll learn about two broad groups of collections—sequences and maps—and features that are specific to each group.

Sequences

Some collections are *sequences*, which means the collection's elements have a defined ordering, usually the order in which they are inserted into the collection.

`List` and `ArrayBuffer` are sequences. `Map` is not.

Sequences: Head and Tail

With a sequence we can get the first element of the sequence—the `head`—and the rest of the sequence—the `tail`.

```scala
val data = mutable.ArrayBuffer(1, 2, 3)
data.head // 1
data.tail // ArrayBuffer(2, 3)
```

Sequences: Head and Tail

With a sequence we can get the first element of the sequence—the `head`—and the rest of the sequence—the `tail`.

```
val data = mutable.ArrayBuffer(1, 2, 3)
data.head // 1
data.tail // ArrayBuffer(2, 3)
```

Note: if the sequence is empty we'll get an exception when we use `head` or `tail`.

Sequences: Linear and Indexed

Lists are *linear* sequences: accessing the nth element requires iterating through the first n - 1 elements. So, accessing the element at index n is an O(n) operation.

Array buffers are *indexed* sequences: accessing an element at any index takes the same time and therefore is an O(1) operation.

Both data structures have pros and cons. It is *your* responsibility to choose the right data structure according to your needs!

Sequences: Sorting

Use the method `sortBy` to sort sequences.

```
val data = List(
  ”Alice” -> 42,
  ”Bob” -> 30,
  ”Werner” -> 77,
  ”Owl” -> 6
)

data.sortBy((_, age) => age)
// List[(String, Int)] = List((Owl,6), (Bob,30), (Alice,42), (Werner,77))

data.sortBy((name, _) => name)
// List[(String, Int)] = List((Alice,42), (Bob,30), (Owl,6), (Werner,77))
```

Maps

`get` is the most important method on `Map` that we have not yet explored. It allows us to get the element associated with a key.

```
val data = Map("a" -> 0, "b" -> 1, "c" -> 2)
data.get("a") // Some(0)
data.get("z") // None
```

Maps

`get` is the most important method on `Map` that we have not yet explored. It allows us to get the element associated with a key.

```
val data = Map(”a” -> 0, ”b” -> 1, ”c” -> 2)
data.get(”a”) // Some(0)
data.get(”z”) // None
```

Note: `get` returns an `Option` as there may not be an element associated with the key.

Sequences and Maps: Summary

We have learned that sequences are collections whose elements have a defined order. We saw that:

- we can access the first and remainder of the sequence with `head` and `tail`;
- we can sort a sequence using `sortBy`;
- some sequences are linear, meaning the time to access an element is proportional to how far it is from the start, and some are indexed, meaning all elements take the same time to access.

We also learned that maps have a method `get` to access values by key.