Chapter 9: Collections

Exercises:

1. There are many ways you can use the amazing map function. Here are some example exercises to strengthen your grip on this function:
    - Create a list of numbers and return true if an element is even; otherwise, return false.
    - Create a list of strings and extract the first and last character of each string.
    - Load a file in Scala and load its content in a list. Then iterate through each line, one by one.
2. Try using ArrayBuffer as a mutable collection. Make sure you understand the difference between ListBuffer and ArrayBuffer.
3. Understand what Array is in Scala. How is it different from the other collections that you studied?
4. Understand what vector is in Scala and how is it different from the other collections that you studied.

Answer :

- **ArrayBuffer**

As per the Scala Documentation, an is a mutable data structure which allows you to access and modify elements at specific index.

How to initialize an ArrayBuffer with 3 elements

```
scala> import scala.collection.mutable.ArrayBuffer
import scala.collection.mutable.ArrayBuffer

scala> println("Step 1: How to initialize an ArrayBuffer with 3 elements")
Step 1: How to initialize an ArrayBuffer with 3 elements

scala> val arrayBuffer1: ArrayBuffer[String] = ArrayBuffer("Plain Donut","Strawberry Donut","Chocolate Donut")
arrayBuffer1: scala.collection.mutable.ArrayBuffer[String] = ArrayBuffer(Plain Donut, Strawberry Donut, Chocolate Donut)

scala> println(s"Elements of arrayBuffer1 = $arrayBuffer1")
Elements of arrayBuffer1 = ArrayBuffer(Plain Donut, Strawberry Donut, Chocolate Donut)
```

How to access elements of an ArrayBuffer at specific index

```
scala> println("\nStep 2: How to access elements of an ArrayBuffer at specific index")

Step 2: How to access elements of an ArrayBuffer at specific index

scala> println(s"Element at index 0 = ${arrayBuffer1(0)}")
Element at index 0 = Plain Donut

scala> println(s"Element at index 1 = ${arrayBuffer1(1)}")
Element at index 1 = Strawberry Donut

scala> println(s"Element at index 2 = ${arrayBuffer1(2)}")
Element at index 2 = Chocolate Donut
```

How to add elements to an ArrayBuffer using +=

```
scala> println("\nStep 3: How to add elements to an ArrayBuffer using +=")

Step 3: How to add elements to an ArrayBuffer using +=

scala> arrayBuffer1 += "Vanilla Donut"
res19: arrayBuffer1.type = ArrayBuffer(Plain Donut, Strawberry Donut, Chocolate Donut, Vanilla Donut)

scala> println(s"Elements of arrayBuffer1 = $arrayBuffer1")
Elements of arrayBuffer1 = ArrayBuffer(Plain Donut, Strawberry Donut, Chocolate Donut, Vanilla Donut)
```

How to add elements from a List to an ArrayBuffer using ++=

```
scala> println("\nStep 4: How to add elements from a List to an ArrayBuffer using ++=")

Step 4: How to add elements from a List to an ArrayBuffer using ++=

scala> arrayBuffer1 ++= List[String]("Glazed Donut", "Krispy creme")
res22: arrayBuffer1.type = ArrayBuffer(Plain Donut, Strawberry Donut, Chocolate Donut, Vanilla Donut, Glazed Donut, Krispy creme)

scala> println(s"Elements of arrayBuffer1 = $arrayBuffer1")
Elements of arrayBuffer1 = ArrayBuffer(Plain Donut, Strawberry Donut, Chocolate Donut, Vanilla Donut, Glazed Donut, Krispy creme)
```

How to remove elements from an ArrayBuffer

```
cala> println("\nStep 5: How to remove elements from an ArrayBuffer")

tep 5: How to remove elements from an ArrayBuffer

cala> arrayBuffer1 -= "Plain Donut"
es25: arrayBuffer1.type = ArrayBuffer(Strawberry Donut, Chocolate Donut, Vanilla Donut, Glazed Donut, Krispy creme)

cala> println(s"Elements of arrayBuffer1 = $arrayBuffer1")
lements of arrayBuffer1 = ArrayBuffer(Strawberry Donut, Chocolate Donut, Vanilla Donut, Glazed Donut, Krispy creme)
```

How to remove elements of a List from ArrayBuffer using --=

```
scala> println("\nStep 6: How to remove elements of a List from ArrayBuffer using --=")

Step 6: How to remove elements of a List from ArrayBuffer using --=

scala> arrayBuffer1 --= List[String]("Glazed Donut", "Krispy creme")
res28: arrayBuffer1.type = ArrayBuffer(Strawberry Donut, Chocolate Donut, Vanilla Donut)

scala> println(s"Elements of arrayBuffer1 = $arrayBuffer1")
Elements of arrayBuffer1 = ArrayBuffer(Strawberry Donut, Chocolate Donut, Vanilla Donut)
```

How to initialize an empty ArrayBuffer

```
scala> println("\nStep 7: How to initialize an empty ArrayBuffer")

Step 7: How to initialize an empty ArrayBuffer

scala> val emptyArrayBuffer: ArrayBuffer[String] = ArrayBuffer.empty[String]
emptyArrayBuffer: scala.collection.mutable.ArrayBuffer[String] = ArrayBuffer()

scala> println(s"Empty array buffer = $emptyArrayBuffer")
Empty array buffer = ArrayBuffer()
```

- Scala ArrayBuffer vs ListBuffer

A ListBuffer is like an array buffer except that it uses a linked list internally instead of an array. If you plan to convert the buffer to a list once it is built up, use a list buffer instead of an array buffer. scala> val buf = scala.collection.mutable. ListBuffer .empty [Int] buf: scala.collection.mutable.

If you're an OOP developer coming to Scala from Java, the ArrayBuffer class will probably be most comfortable for you, so we'll demonstrate it first. It's a mutable sequence, so you can use its methods to modify its contents, and those methods are similar to methods on Java sequences.

Just as the Vector class is the recommended "go to" class for immutable, indexed sequential collections, the ArrayBuffer class is recommended as the general-purpose collections class for mutable, indexed sequential collections. ArrayBuffer is an indexed sequential collection. Use ListBuffer if you prefer a linear sequential collection that is mutable.

As per the Scala Documentation, a ListBuffer is resizable similar to an ArrayBuffer, except that it uses a Linked List as its internal data structure.

To use, ArrayBuffer, scala.collection.mutable.ArrayBuffer class is imported, an instance of ArrayBuffer is created. Internally, an ArrayBuffer is an Array of elements, as well as the store's current size of the array. When an element is added to an ArrayBuffer, this size is checked.

What is an ArrayBuffer? As per the Scala Documentation, an ArrayBuffer is a mutable data structure which allows you to access and modify elements at specific index. Compared to the previous tutorial on Array, an ArrayBuffer is resizable while an Array is fixed in size.

List vs ListBuffer. A list is another data structure which maintains the order of the elements, allows for duplication and is immutable. Similarly to the ArrayBuffer, the ListBuffer is the mutable form of a list. List -- A list is an immutable data structure which can contain duplicates and maintains the order if its elements.

- Arrays

Array is a special kind of collection in scala. it is a fixed size data structure that stores elements of the same data type. The index of the first element of an array is zero and the last element is the total number of elements minus one. It is a collection of mutable values. It

corresponds to arrays(in terms of syntax) in java but at the same time it's different(in terms of functionalities) from java.

Some Important Points:
•Scala arrays can be generic. which mean we can have an Array[T], where T is a type parameter or abstract type.

•Scala arrays are compatible with Scala sequences – we can pass an Array[T] where a Seq[T] is required.

•Scala arrays also support all sequence operations

```
scala> val numbers = Array(1, 2, 3, 4)
numbers: Array[Int] = Array(1, 2, 3, 4)

scala>
```

- Vector

is an indexed, immutable sequence. The "indexed" part of the description means that you can access vector elements very rapidly by their index value, such as accessing listOfPeople(999999).

In general, except for the difference that Vector is indexed and List is not, the two classes work the same, so we'll run through these examples quickly.

Here are a few ways you can create a Vector:

```
scala> val nums = Vector(1, 2, 3, 4, 5)
nums: scala.collection.immutable.Vector[Int] = Vector(1, 2, 3, 4, 5)

scala>

scala> val strings = Vector("one", "two")
strings: scala.collection.immutable.Vector[String] = Vector(one, two)
```