

More Scala

Data Structures

Lecture Objective

- This is Lecture Objective 3 from the Pale Blue Dot project -
 - In the `pbd.PaleBlueDot` object, write a method named "cityPopulations" which:
 - Takes three Strings as parameters representing:
 - The name of a file containing city data. ex. "data/cities.csv"
 - A two character country code. You may assume that the input is in all lowercase letters
 - A region code in the same format as they appear in the cities file
 - Returns a Map of Strings to Ints mapping city names to their population
 - The city names should appear exactly as they do in the cities file
 - Note: All the Strings will be in the same format as they appear in the cities file. You don't have to do anything with upper/lower case for this objective

Data Structures

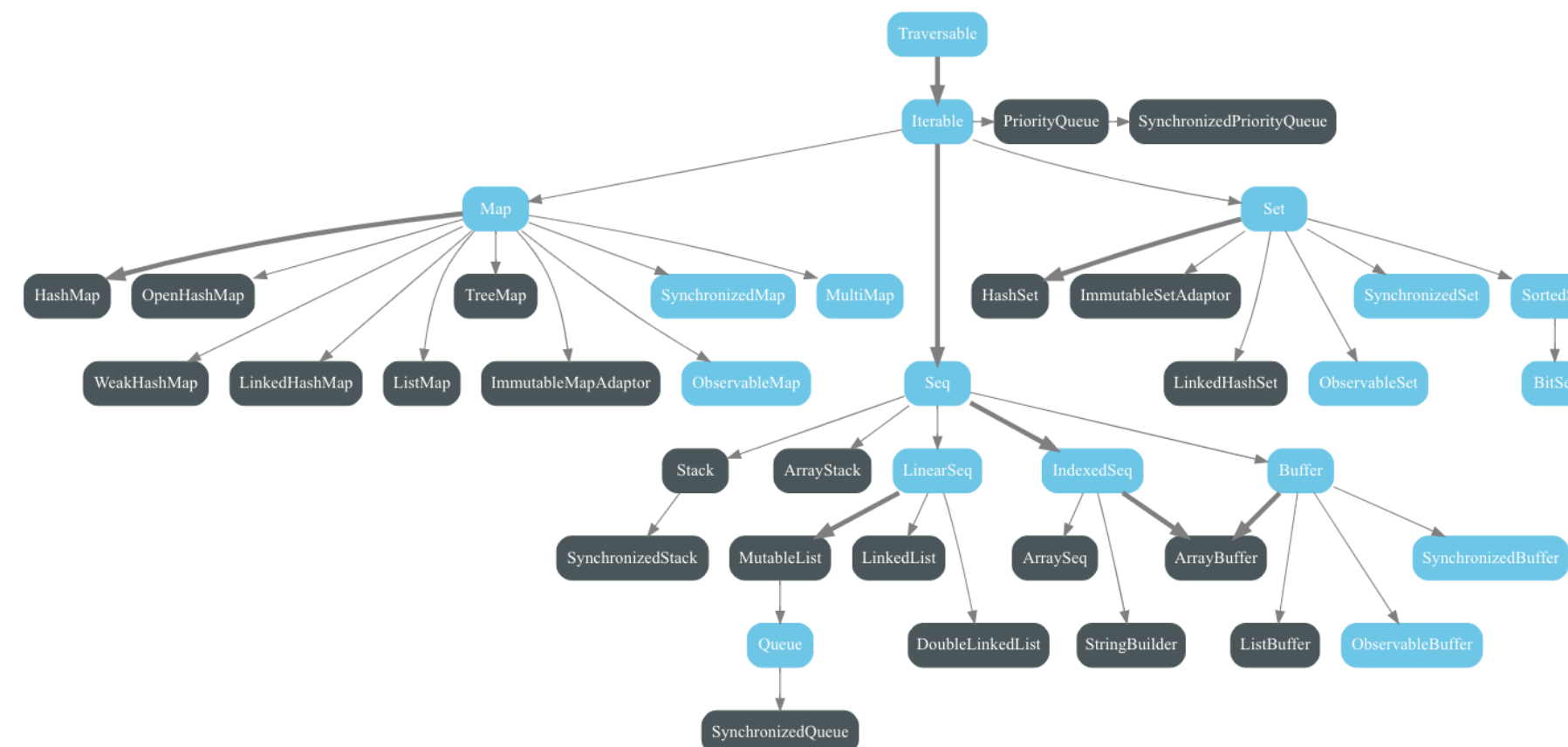
Data Structures

- Wide variety of Data Structures in Scala

Graph for immutable hierarchy:



Graph for mutable hierarchy:



<https://docs.scala-lang.org/tutorials/FAQ/collections.html>

Data Structures

- Let's keep it simple and focus on 3 for now
- Array
 - Sequential
 - Fixed Size
- List
 - Sequential
- Map
 - Key-Value Store

Array

- Sequential
 - One continuous block of memory
 - Random access based on memory address
 - $\text{address} = \text{first_address} + (\text{element_size} * \text{index})$
- Fixed Size
 - Since memory adjacent to the block may be used
 - Efficient when you know how many elements you'll need to store

Array

```
def arrayExample(): Unit = {  
  // Create new Array of Int  
  val arr: Array[Int] = Array(2, 3, 4)  
  
  // Change a value by index  
  arr(1) = 20  
  
  // Access a value by index  
  val x: Int = arr(1)  
  
  // Iterate over elements  
  for (element <- arr) {  
    println(element)  
  }  
  
  // Iterate over indices  
  for (index <- 0 to (arr.length - 1)) {  
    println(index)  
  }  
  
  // Iterate over indices - alternate  
  for (index <- arr.indices) {  
    println(index)  
  }  
}
```

List

- Sequential
 - Spread across memory
 - Each element knows the memory address of the next element
 - Follow the addresses to find each element
- Variable Size
 - Store new element anywhere in memory
 - Add the new memory address to the last element
 - Or new element stores address of first element
- Values cannot change [In Scala]

List

```
def listExample(): Unit = {  
  // Create new Array of Int  
  var list: List[Int] = List(2, 3, 4)  
  
  // Access the first element  
  val x: Int = list.head  
  
  // Access a value by position  
  val y: Int = list.apply(1)  
  
  // Add an element to the end of the list (append)  
  list = list :+ 50  
  
  // Add an element to the beginning of the list (prepend)  
  list = 70 :: list  
  
  // Iteration  
  for(element <- list){  
    println(element)  
  }  
}
```

Map

- Key-Value Store
 - Values stored at keys instead of indices
 - Multiple different implementations
 - Default is HashMap (CSE250 topic)
- Variable Size
- Variable Values

Map

```
def mapExample(): Unit = {  
  // Create new Map of Int to Int  
  var myMap: Map[Int, Int] = Map(2 -> 4, 3 -> 9, 4 -> 16)  
  
  // Add a key-value pair  
  myMap = myMap + (5 -> 25)  
  
  // Access a value by key (Crashes if key not in map)  
  val x: Int = myMap(3)  
  
  // Access a value by key with default value if key not in map  
  val y: Int = myMap.getOrElse(100, -1)  
  
  // Iteration  
  for((key, value) <- myMap){  
    println("value " + value + " stored at key " + key)  
  }  
}
```

Data Structures

Example in IntelliJ

Lecture Objective

- This is Lecture Objective 3 from the Pale Blue Dot project -
 - In the `pbd.PaleBlueDot` object, write a method named "cityPopulations" which:
 - Takes three Strings as parameters representing:
 - The name of a file containing city data. ex. "data/cities.csv"
 - A two character country code. You may assume that the input is in all lowercase letters
 - A region code in the same format as they appear in the cities file
 - Returns a Map of Strings to Ints mapping city names to their population
 - The city names should appear exactly as they do in the cities file
 - Note: All the Strings will be in the same format as they appear in the cities file. You don't have to do anything with upper/lower case for this objective