Scala Programming Introduction

March 10, 2016

Agenda

- Functional Programming Vs Imperative Programming
- Why Functional Programming?
- Scala Programming Fundamentals
  - REPL
  - Scala Runnable Program
  - Scala classes and objects
  - Tuples
- Scala Collections Vs Java Collections
- Apache Spark Scala API Intro

Imperative Versus Functional Programming

In imperative approach, a developer writes code that describes in exacting detail the steps that the computer must take to accomplish the goal.

Functional approach involves composing the problem as a set of functions to be executed. “Pure Functions” makes the order of execution irrelevant - since no side-effect can change the value of an expression, it can be evaluated at any time.

Characteristic	Imperative approach	Functional approach
Programmer focus	How to perform tasks (algorithms) and how to track changes in state.	What information is desired and what transformations are required.
State changes	Important.	Non-existent.
Order of execution	Important.	Low importance.
Primary flow control	Loops, conditionals, and function (method) calls.	Function calls, including recursion.
Primary manipulation unit	Instances of structures or classes.	Functions as first-class objects and data collections.

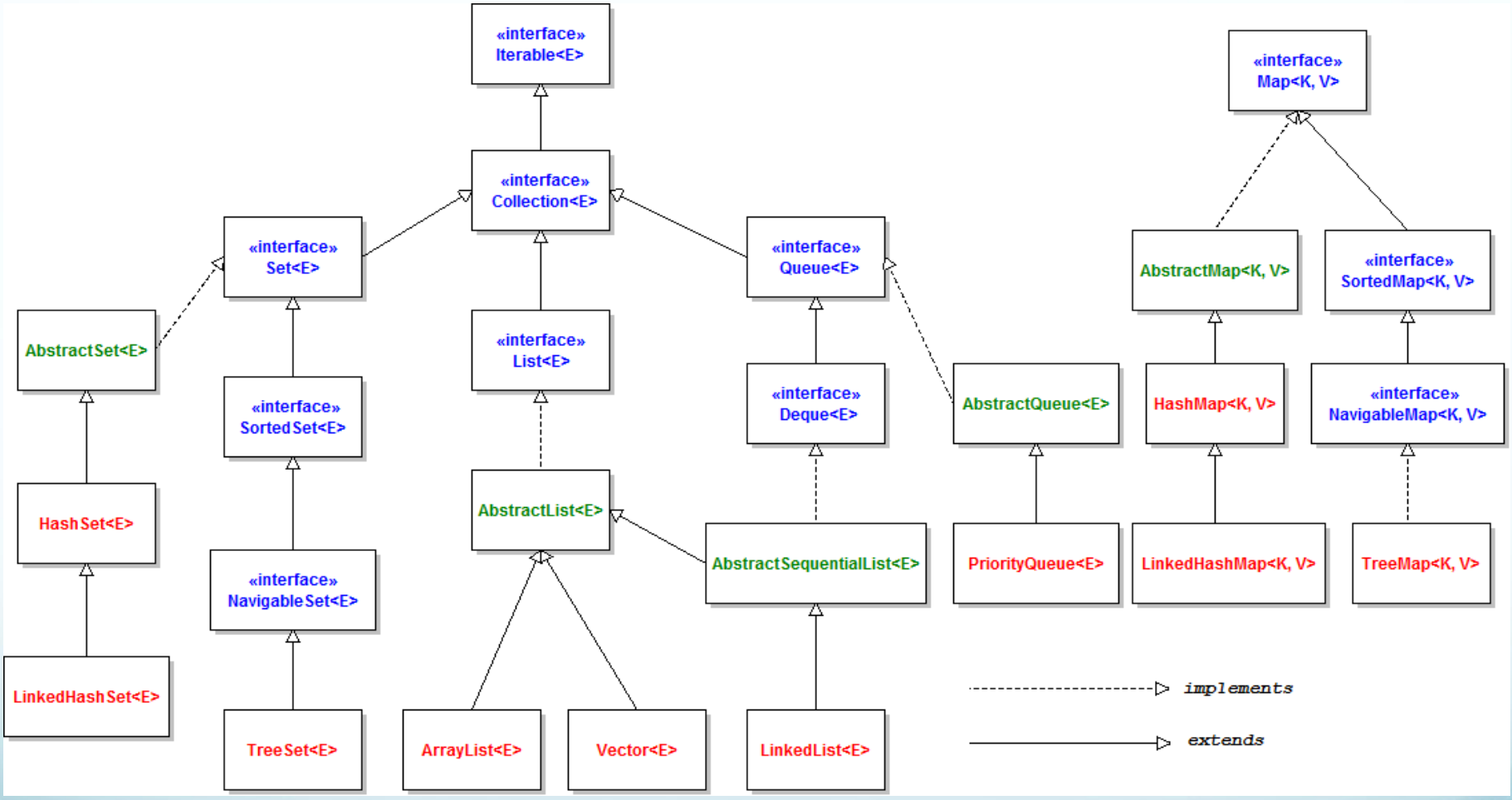
Why Functional Programming?

- Describe what you want, rather than how to get it.
  - Instead of creating a for-loop with an iterator variable and marching through an collection doing something to each cell, pass a function and it will be executed for all the elements in the collection which can be optimized
  - More concise and expressive code
- Functional programming moves more basic programming ideas into the compiler.
  - Syntactic Sugar – case classes, pattern matching
  - Type Inference
  - List comprehension
  - conversion to primitives, tail recursion
  - Implicit conversions
- Concurrency
  - No Shared mutable state – so no locking required
- Pure functional languages support lazy evaluation.
  - Can create infinitely large collections
  - Value will be calculated only when required and function call be substituted in place of functional value (Referential integrity)

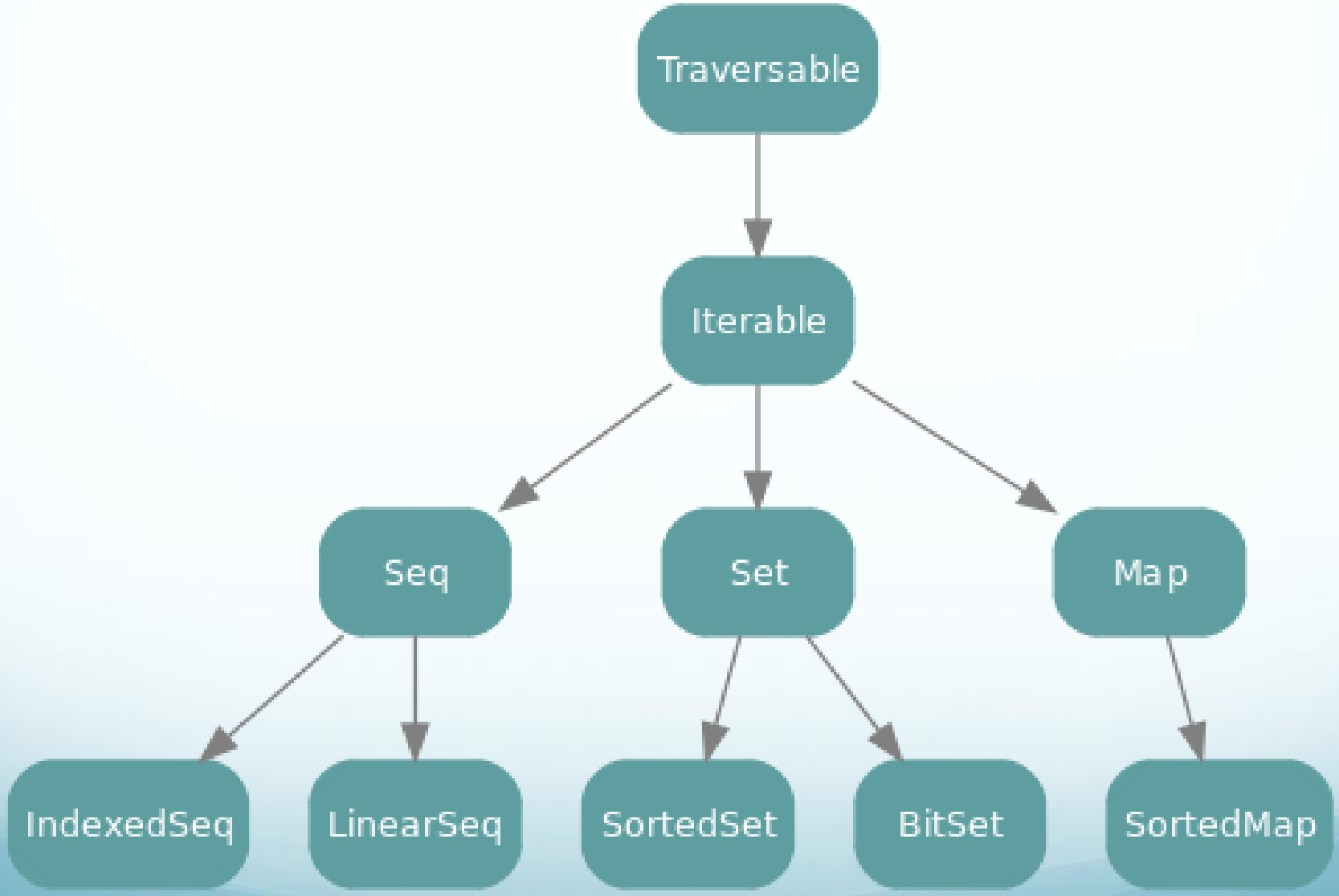
Scala Programming Fundamentals

- Scala brings together concepts from both Functional Programming and Objected Oriented Programming.
- Provides a more comprehensive API, a superior compiler but stills runs on the JVM.
- Demos:
  - REPL
  - Runnable Program in Scala
  - Scala classes and Objects
  - Tuples

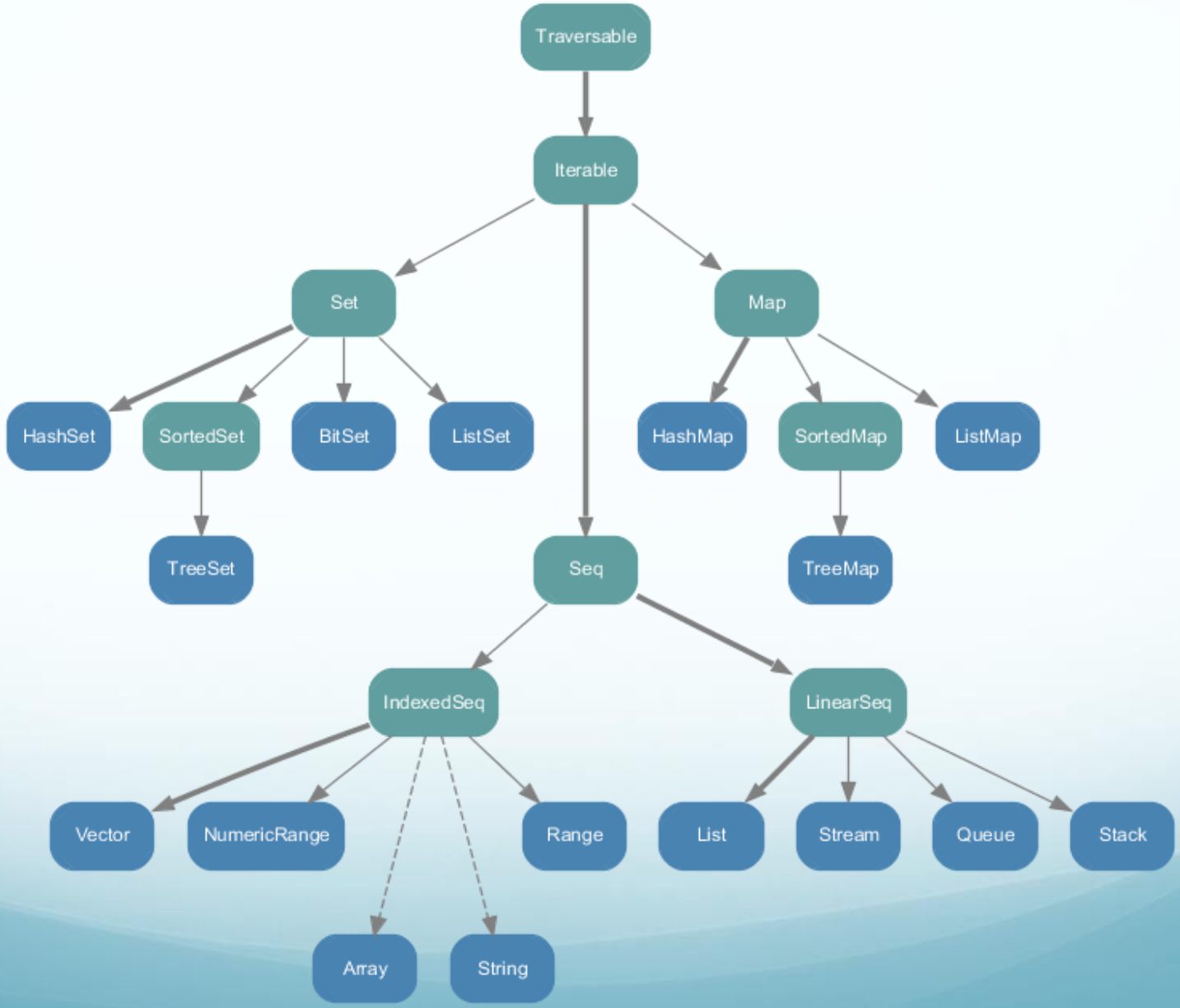
Java Collections Hierarchy



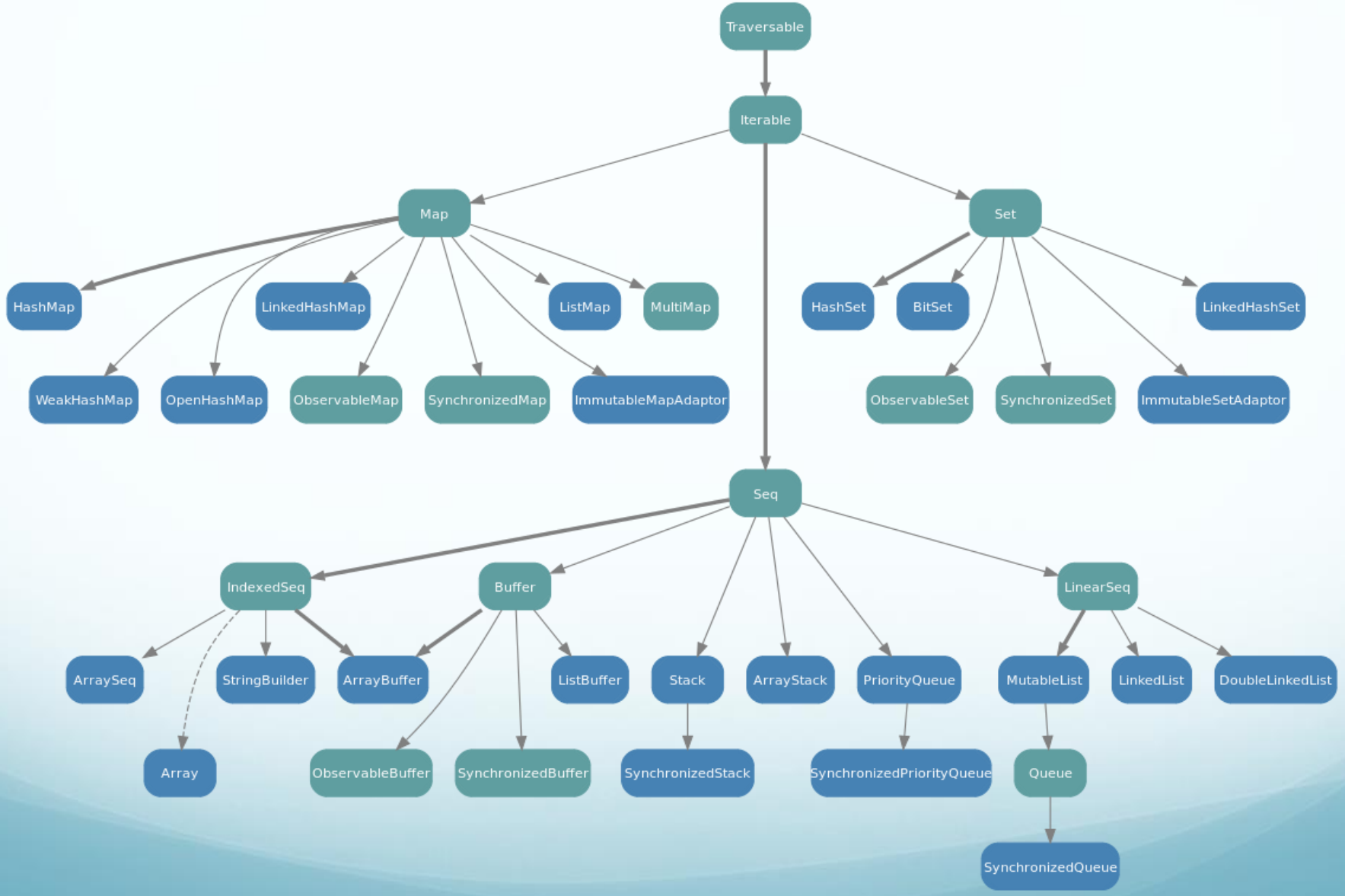
Scala Collection Hierarchy



Scala Immutable Collections

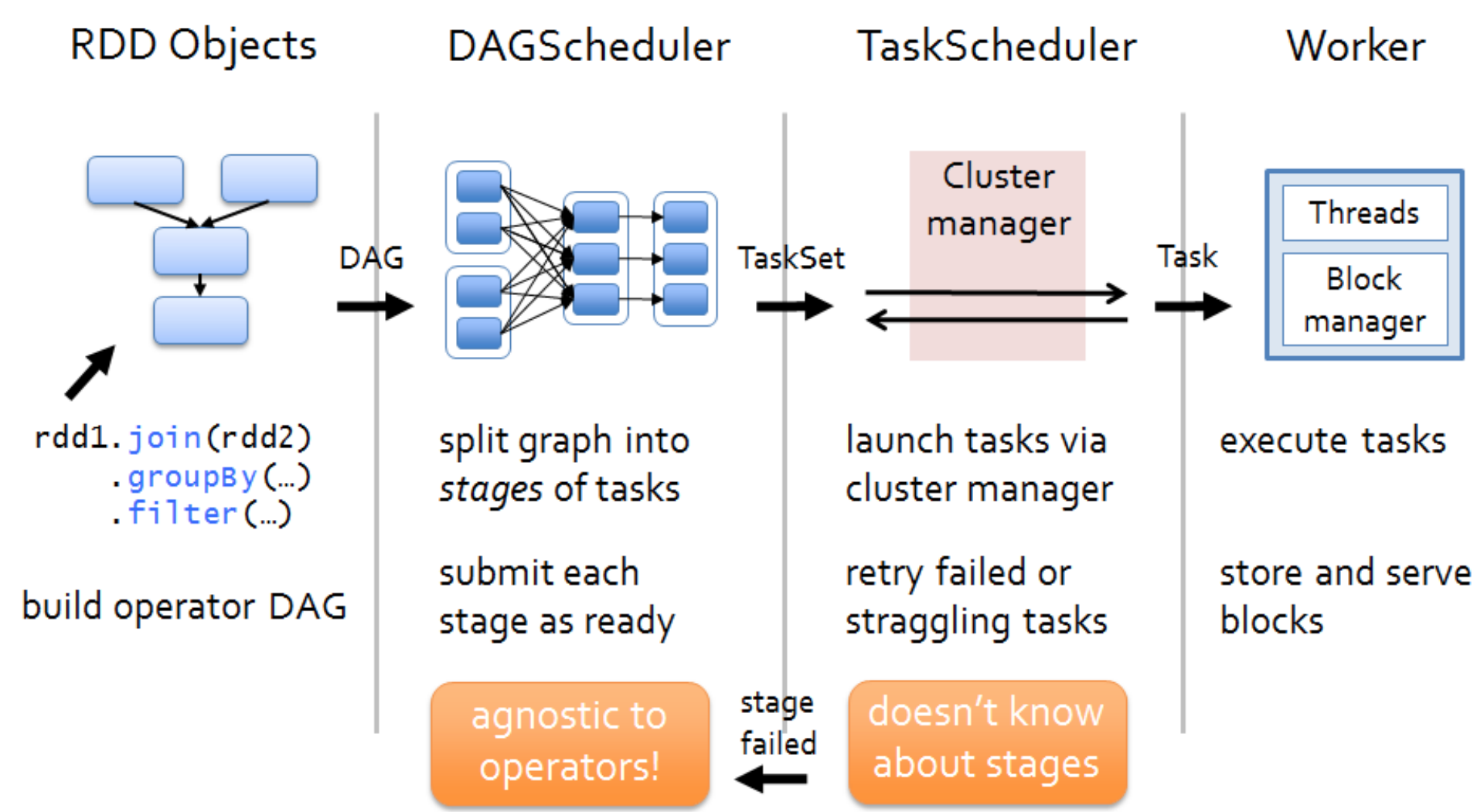


Scala Mutable Collections



Why Spark Uses Scala?

Apache Spark Execution Model



Transformations + lazy evaluation = Optimization