Collections

Harit Himanshu
@harittweets

Overview

- Benefits of Scala collections
- Mutable and Immutable
- Scala collections Hierarchy
- List with examples
- Set with examples
- Map with examples
- Methods on numeric collections
- Filtering, size and conversion operations on collections
- Transforming a collection using map
- Transforming a collection using flatMap
- Working with Option and flatMap

Scala collection properties

Easy to use

Concise

Safe

Fast

Easy to use

Small vocabulary

Persistent Collections

No Update/Iterator interference

Concise

Compact

Light-weight Syntax

Safe

Heavily Tested

Explicit Input/Output

Static Type checking

Fast

Tuned and Optimized

Parallel Execution support

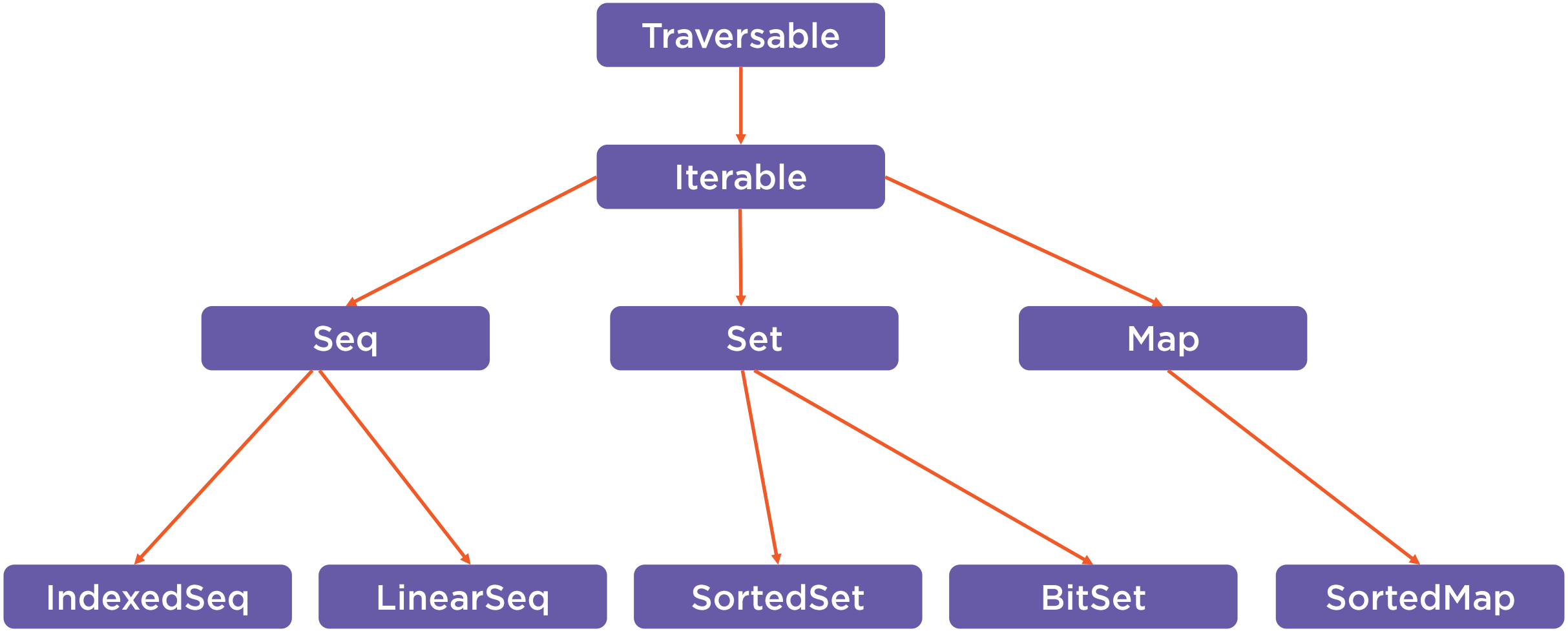
Types of Scala collections

Sequences

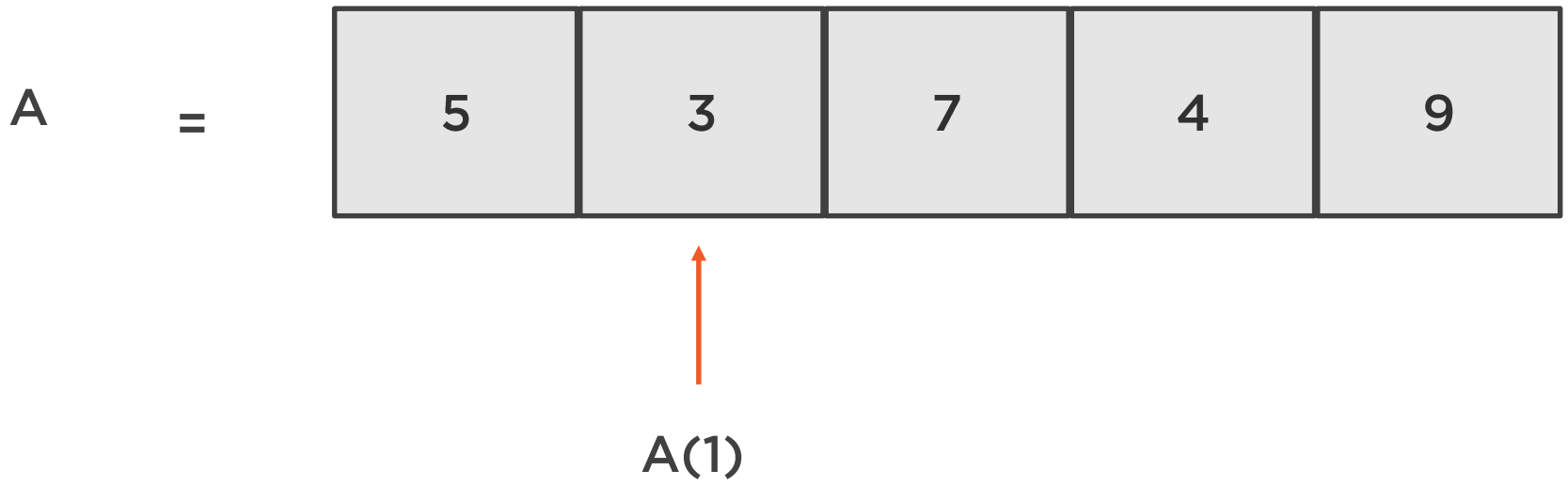
IndexedSeq

LinearSeq

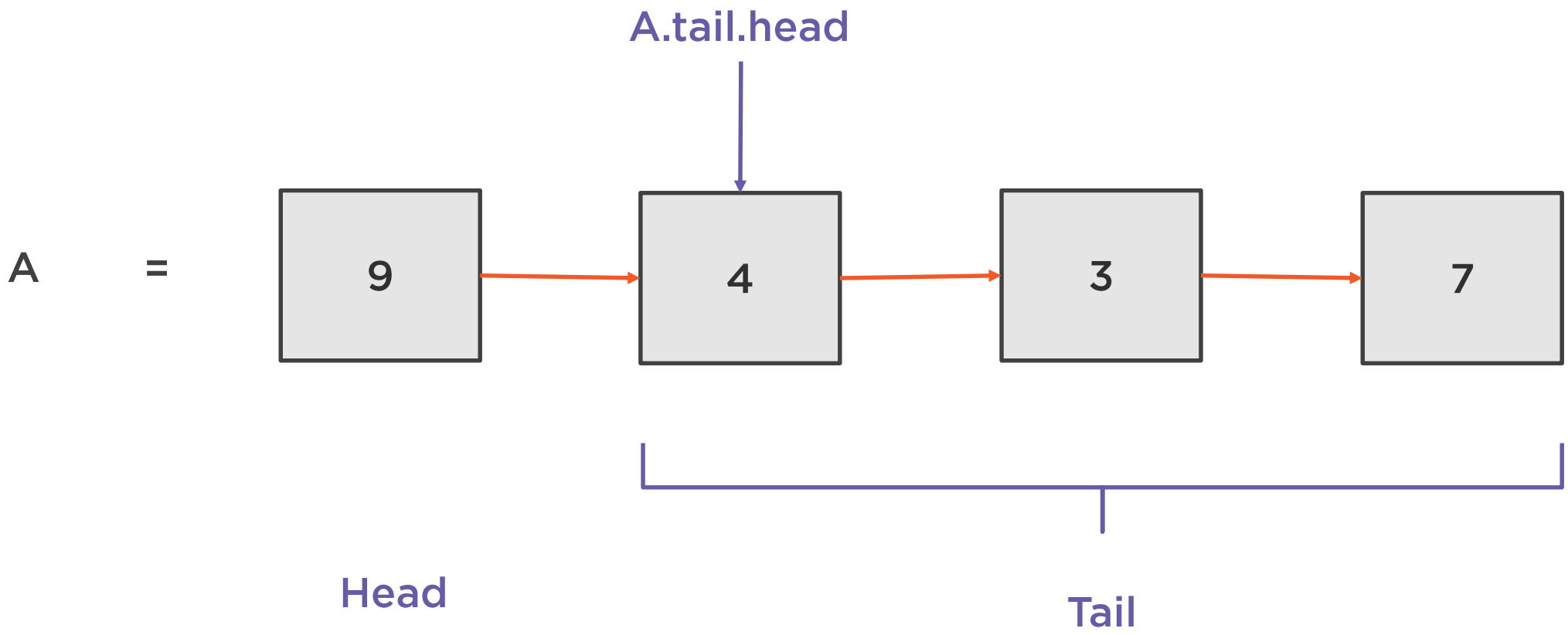
Scala collection hierarchy



IndexedSeq



LinearSeq



Set

SortedSet

BitSet

SortedSet

BitSet

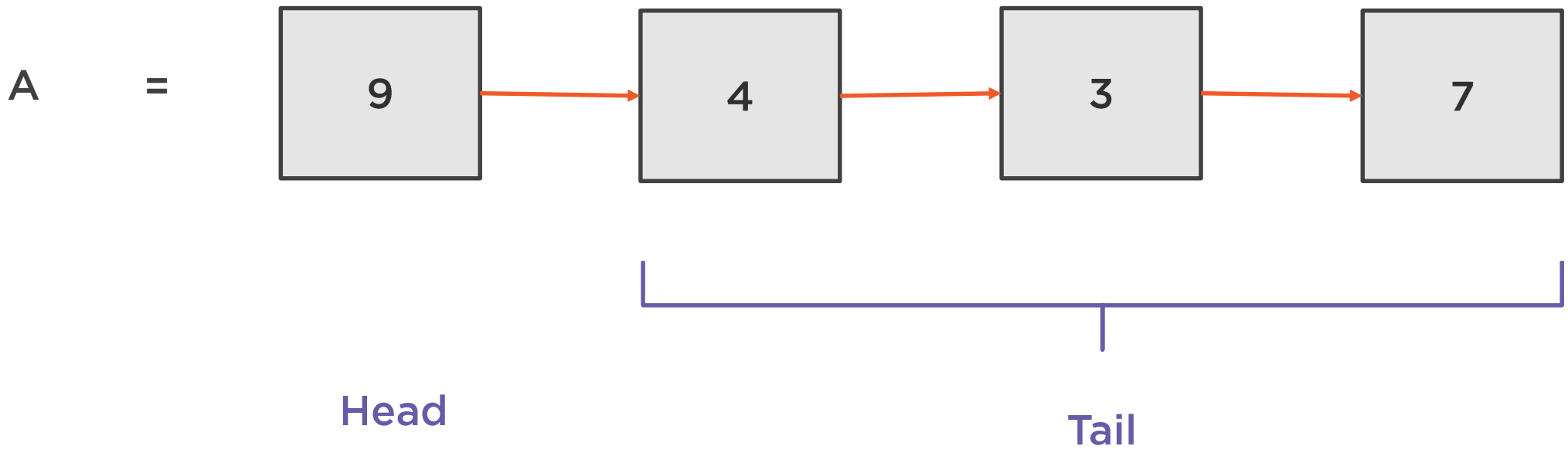
A concrete class

Non-negative integers as variable-size arrays

Map

SortedMap

Scala List



```
map[B](f: (A) ⇒ B): List[B]
```

Scala map interface

Our programming challenge

Given the list containing lists as its element

Increment every number in the nested list

Return the result as one list *instead of nested list*

```
nestedList.map((aList:List[Int]) => {/* something with the list */})
```

Iterate on the outer list

flatMap key points

Provide map implementation

flatMap as *flattenAfterMap*

Our programming challenge

- Given command-line program to accept user inputs
- Transform program inputs from String to Integer
- Add the numbers and return the result

Examples

Input	Output
“4”, “2”, “1”	7
“4”, “2”, “one”	6

Summary

- Benefits of Scala collections
- Mutable and Immutable
- Scala collections Hierarchy
- List with examples
- Set with examples
- Map with examples
- Methods on numeric collections
- Filtering, size and conversion operations on collections
- Transforming a collection using map
- Transforming a collection using flatMap
- Working with Option and flatMap