

# Scala Cheat Sheet

## 1 Scala Class Hierarchy

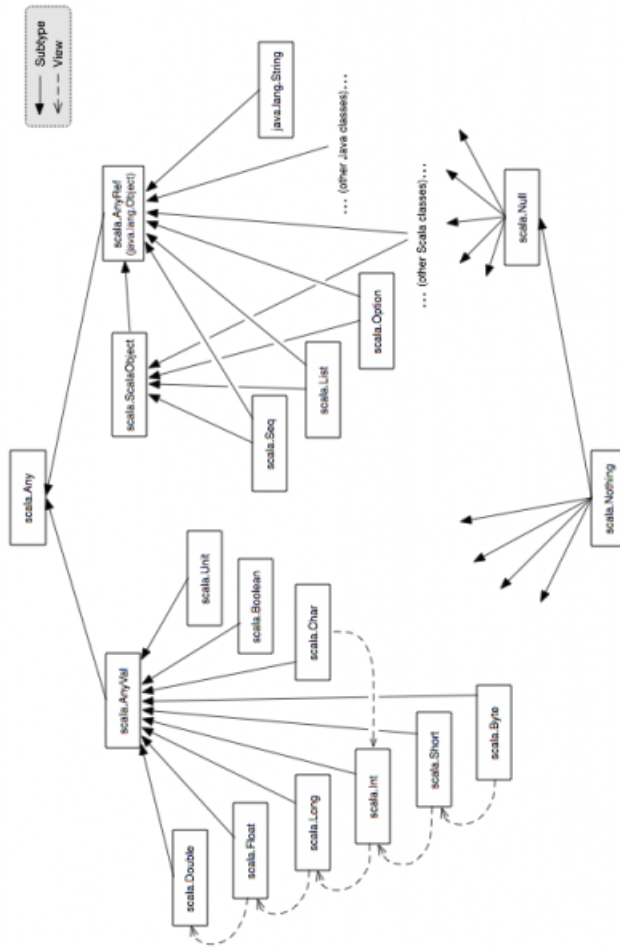


Figure 1: Scala class hierarchy, source: <http://docs.scala-lang.org/tutorials/tour/unified-types.html>

## 2 Scala Collections

### 2.1 Scala Collections Hierarchy

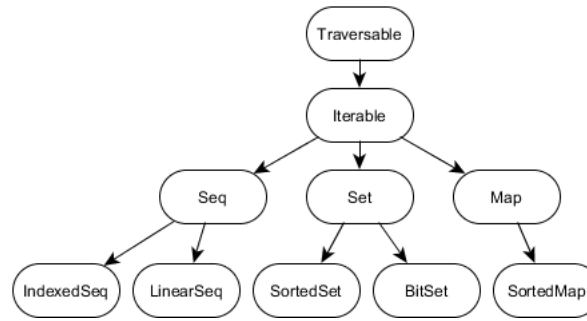
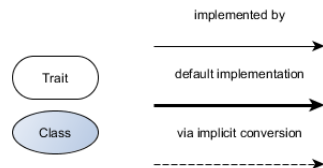


Figure 2: scala.collection

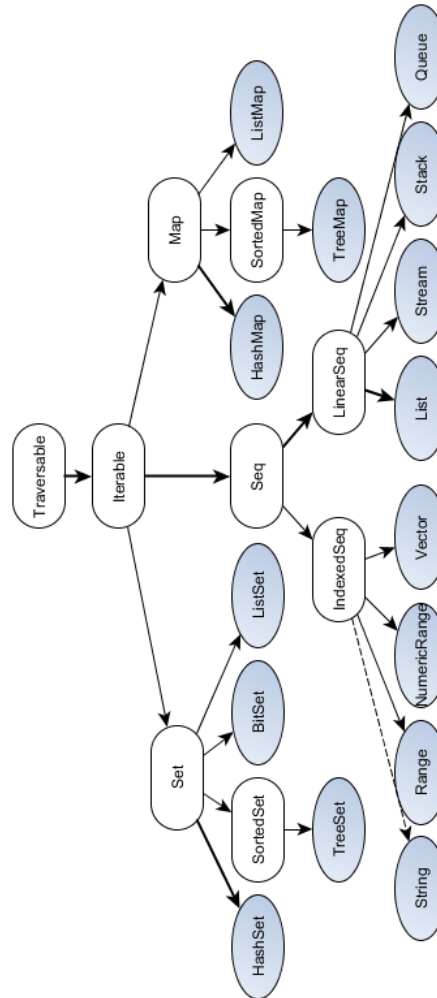


Figure 3: scala.collection.immutable

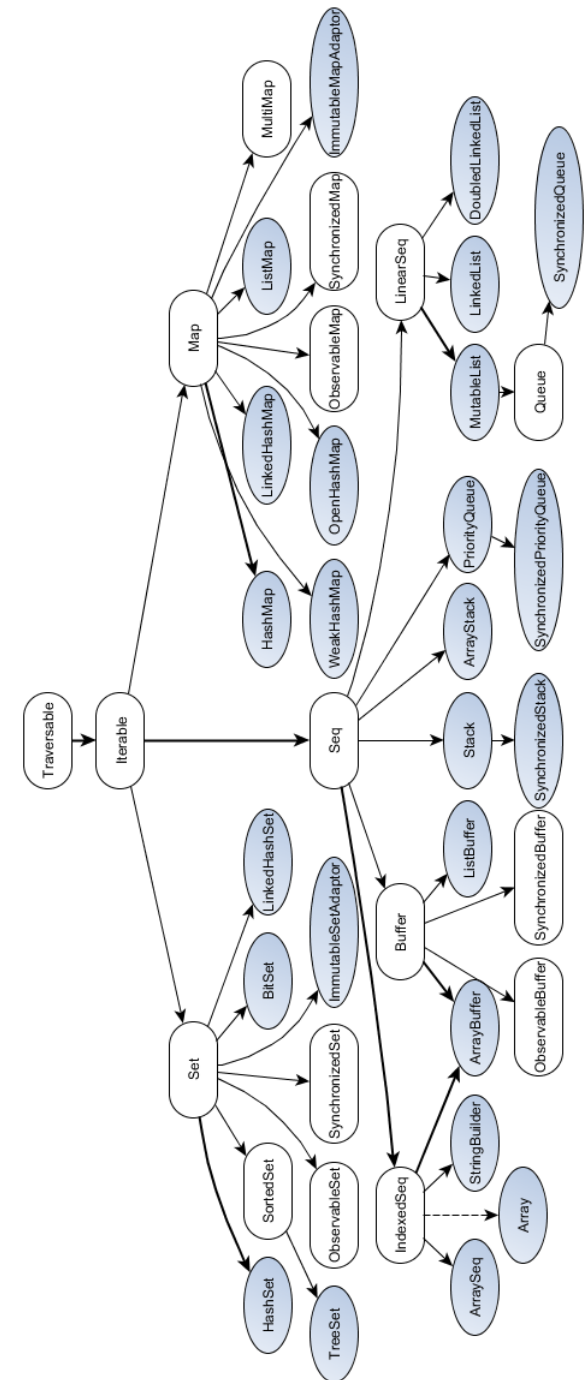


Figure 4: scala.collection.mutable

2.2 Trait Traversable

Table 1: Methods in Traversable

Category	Methods
Abstract	xs foreach f
Addition	xs ++ ys
Maps	xs map f, xs flatMap f, xs collect f
Conversions	toArray, toList, toIterable, toSeq, toIndexedSeq, toStream, toSet, toMap
Size info	isEmpty, nonEmpty, size, hasDefiniteSize
Element	head, headOption, last, lastOption,
Retrieval	xs find p
Sub-collection	xs.tail, xs.init, xs slice (from, to), xs take n, xs drop n, xs takeWhile p, xs dropWhile p, xs filter p, xs withFilter p, xs filterNot p
Subdivision	xs splitAt n, xs span p, xs partition p, xs groupBy f
Element Condition	xs forall p, xs exists p, xs count p
Fold	(z /: xs)(op), (xs : z)(op), xs.foldLeft(z)(op), xs.foldRight(z)(op), xs.reduceLeft op, xs.reduceRight op
Specific Fold String	xs.sum, xs.product, xs.min, xs.max xs.addString(b, start, sep, end), xs.mkString(start, sep, end), xs.stringPrefix
View	xs.view, xs.view(from, to)

Reference: <http://docs.scala-lang.org/overviews/collections/trait-traversable.html>

2.3 Trait Iterable

All methods in this trait are defined in terms of an abstract method, `iterator`, which yields the collections elements one by one.

Table 2: Methods in Iterable

Category	Methods
Abstract	xs.iterator
Iterator	xs.grouped n, xs.sliding n
Subcollection	xs.takeRight n, xs.dropRight n
Zipper	xs.zip ys, xs.zipAll(ys, x, y), xs.zipWithIndex
Comparison	xs.sameElements ys

Reference: <http://docs.scala-lang.org/overviews/collections/trait-iterable.html>

In the inheritance hierarchy below `Iterable` you find three traits: `Seq`, `Set`, and `Map`. A common aspect of these three traits is that they all implement the `PartialFunction` trait with its `apply` and `isDefinedAt` methods. However, the way each trait implements `PartialFunction` differs.

2.4 Seq

All methods in this trait are defined in terms of an abstract method, `iterator`, which yields the collections elements one by one.

Table 3: Methods in Seq

Category	Methods
Indexing and Length	xs(i), xs.isDefinedAt i, xs.length, xs.lengthCompare ys, xs.indices
Index Search	xs.indexOf x, xs.lastIndexOf x, xs.indexOfSlice ys, xs.lastIndexOfSlice ys, xs.indexWhere p, xs.segmentLength(p, i), xs.prefixLength p
Addition	x +: xs, xs :=+ x, xs.padTo(len, x)
Update	xs.patch(i, ys, r), xs.updated(i, x), xs(i) = x(only available for mutable.Seq)
Sorting	xs.sorted, xs.sortWith lt, xs.sortBy f
Reversal	xs.reverse, xs.reverseIterator, xs.reverseMap f
Comparison	xs.startsWith ys, xs.endsWith ys, xs.contains x, xs.containsSlice ys, (xs.corresponds ys)(p)
Multiset	xs.intersect ys, xs.union ys, xs.diff ys, xs.distinct

Reference:  
<http://docs.scala-lang.org/overviews/collections/seqs.html>

Trait `Seq` has two subtraits `LinearSeq`, and `IndexedSeq`. These do not add any new operations, but each offers different performance characteristics: A linear sequence has efficient `head` and `tail` operations, whereas an indexed sequence has efficient `apply`, `length`, and (if mutable) `update` operations. Frequently used linear sequences are `immutable.List` and `immutable.Stream`. Frequently used indexed sequences are `scala.Array` and `mutable.ArrayBuffer`. The `Vector` class provides an interesting compromise between indexed and linear access. It has both effectively constant time indexing overhead and constant time linear access overhead. Because of this, vectors are a good foundation for mixed access patterns where both indexed and linear accesses are used.

Table 4: Methods in Buffer

Category	Methods
Addition	buf += x, buf += (x, y, z), buf += xs, x +=: buf, xs +=: buf, buf.insert(i, x), buf.insertAll(i, xs)
Removal	buf -= x, buf.remove i, buf.remove(i, n), buf.trimStart n, buf.trimEnd n, buf.clear()
Cloning	buf.clone

2.5 Set

Category	Methods
Test	xs contains x, xs(x), xs.subsetOf ys
Addition	xs + x, xs + (x, y, z), xs ++ ys
Removal	xs - x, xs - (x, y, z), xs -- ys, xs.empty
Set operation	xs & ys, xs.intersect ys, xs   ys, xs.union ys, xs & ys, xs.diff ys

Reference:  
<http://docs.scala-lang.org/overviews/collections/sets.html>

Mutable sets offer in addition methods to add, remove, or update elements, which are summarized in below.

Table 6: Methods in mutable.Set

Category	Methods
Addition	xs += x, xs += (x, y, z), xs += ys, xs.add x
Removal	xs -= x, xs -= (x, y, z), xs -= ys, xs.remove x, xs.retain p, xs.clear()
Update	xs(x) = b
Cloning	xs.clone

2.6 Map

Table 7: Methods in Map

Category	Methods
Lookup	ms.get k, ms(k), ms.getOrElse(k, d), ms.contains k, ms.isDefinedAt k
Addition	ms + (k -> v), ms + (k -> v, l -> w), ms ++ kvs
Removal	ms - k, ms - (k, l, m), ms -- ks
Update	ms.updated(k, v)
Subcollection	ms.keys, ms.keySet, ms.keyIterator, ms.values, ms.valuesIterator
Transformation	ms.filterKeys p, ms.mapValues f

Reference:  
<http://docs.scala-lang.org/overviews/collections/maps.html>

Table 8: Methods in mutable.Map

Category	Methods
Addition	ms += (k -> v), ms += (k -> v, l -> w), ms += kvs,
Removal	ms -= k, ms -= (k, l, m), ms -= ks, ms.remove k, ms.retain p, ms.clear()
Update	ms(k) = v, ms.put(k, v), ms.getOrElseUpdate(k, d)
Transformation	ms.transform f
Cloning	xs.clone

Table 5: Methods in Set