Scala Collections API

Sr. No	**Array Methods with Description**
1	**def apply(x: T, xs: T*): Array[T]** Creates an array of T objects, where T can be Unit, Double, Float, Long, Int, Char, Short, Byte, Boolean.
2	**def concat[T](xss: Array[T]*): Array[T]** Concatenates all arrays into a single array.
3	**def copy(src: AnyRef, srcPos: Int, dest: AnyRef, destPos: Int, length: Int): Unit** Copy one array to another. Equivalent to Java's System.arraycopy(src, srcPos, dest, destPos, length).
4	**def empty[T]: Array[T]** Returns an array of length 0
5	**def iterate[T](start: T, len: Int)(f: (T) => T): Array[T]** Returns an array containing repeated applications of a function to a start value.
6	**def fill[T](n: Int)(elem: => T): Array[T]** Returns an array that contains the results of some element computation a number of times.
7	**def fill[T](n1: Int, n2: Int)(elem: => T): Array[Array[T]]** Returns a two-dimensional array that contains the results of some element computation a number of times.
8	**def iterate[T](start: T, len: Int)(f: (T) => T): Array[T]** Returns an array containing repeated applications of a function to a start value.
9	**def ofDim[T](n1: Int): Array[T]** Creates array with given dimensions.
10	**def ofDim[T](n1: Int, n2: Int): Array[Array[T]]** Creates a 2-dimensional array
11	**def ofDim[T](n1: Int, n2: Int, n3: Int): Array[Array[Array[T]]]** Creates a 3-dimensional array
12	**def range(start: Int, end: Int, step: Int): Array[Int]** Returns an array containing equally spaced values in some integer interval.
13	**def range(start: Int, end: Int): Array[Int]** Returns an array containing a sequence of increasing integers in a range.
14	**def tabulate[T](n: Int)(f: (Int)=> T): Array[T]** Returns an array containing values of a given function over a range of integer values starting from 0.
15	**def tabulate[T](n1: Int, n2: Int)(f: (Int, Int) => T): Array[Array[T]]** Returns a two-dimensional array containing values of a given function over ranges of integer values starting from 0.

List Methods with Description

1	**def +(elem: A): List[A]** Prepends an element to this list
2	**def ::(x: A): List[A]** Adds an element at the beginning of this list.
3	**def :::(prefix: List[A]): List[A]** Adds the elements of a given list in front of this list.
4	**def ::(x: A): List[A]** Adds an element x at the beginning of the list
5	**def addString(b: StringBuilder): StringBuilder** Appends all elements of the list to a string builder.
6	**def addString(b: StringBuilder, sep: String): StringBuilder** Appends all elements of the list to a string builder using a separator string.
7	**def apply(n: Int): A** Selects an element by its index in the list.
8	**def contains(elem: Any): Boolean** Tests whether the list contains a given value as an element.
9	**def copyToArray(xs: Array[A], start: Int, len: Int): Unit** Copies elements of the list to an array. Fills the given array xs with at most length (len) elements of this list, beginning at position start.
10	**def distinct: List[A]** Builds a new list from the list without any duplicate elements.
11	**def drop(n: Int): List[A]** Returns all elements except first n ones.
12	**def dropRight(n: Int): List[A]** Returns all elements except last n ones.
13	**def dropWhile(p: (A) => Boolean): List[A]** Drops longest prefix of elements that satisfy a predicate.
14	**def endsWith[B](that: Seq[B]): Boolean** Tests whether the list ends with the given sequence.
15	**def equals(that: Any): Boolean** The equals method for arbitrary sequences. Compares this sequence to some other object.
16	**def exists(p: (A) => Boolean): Boolean** Tests whether a predicate holds for some of the elements of the list.
17	**def filter(p: (A) => Boolean): List[A]** Returns all elements of the list which satisfy a predicate.
18	**def forall(p: (A) => Boolean): Boolean** Tests whether a predicate holds for all elements of the list.
19	**def foreach(f: (A) => Unit): Unit** Applies a function f to all elements of the list.
20	**def head: A** Selects the first element of the list.

21	**def indexOf(elem: A, from: Int): Int** Finds index of first occurrence value in the list, after the index position.
22	**def init: List[A]** Returns all elements except the last.
23	**def intersect(that: Seq[A]): List[A]** Computes the multiset intersection between the list and another sequence.
24	**def isEmpty: Boolean** Tests whether the list is empty.
25	**def iterator: Iterator[A]** Creates a new iterator over all elements contained in the iterable object.
26	**def last: A** Returns the last element.
27	**def lastIndexOf(elem: A, end: Int): Int** Finds index of last occurrence of some value in the list; before or at a given end index.
28	**def length: Int** Returns the length of the list.
29	**def map[B](f: (A) => B): List[B]** Builds a new collection by applying a function to all elements of this list.
30	**def max: A** Finds the largest element.
31	**def min: A** Finds the smallest element.
32	**def mkString: String** Displays all elements of the list in a string.
33	**def mkString(sep: String): String** Displays all elements of the list in a string using a separator string.
34	**def reverse: List[A]** Returns new list with elements in reversed order.
35	**def sorted[B >: A]: List[A]** Sorts the list according to an Ordering.
36	**def startsWith[B](that: Seq[B], offset: Int): Boolean** Tests whether the list contains the given sequence at a given index.
37	**def sum: A** Sums up the elements of this collection.
38	**def tail: List[A]** Returns all elements except the first.
39	**def take(n: Int): List[A]** Returns first "n" elements.

40	**def takeRight(n: Int): List[A]** Returns last "n" elements.
41	**def toArray: Array[A]** Converts the list to an array.
42	**def toBuffer[B >: A]: Buffer[B]** Converts the list to a mutable buffer.
43	**def toMap[T, U]: Map[T, U]** Converts this list to a map.
44	**def toSeq: Seq[A]** Converts the list to a sequence.
45	**def toSet[B >: A]: Set[B]** Converts the list to a set.
46	**def toString(): String** Converts the list to a string.

Sr.No	Map Methods with Description
1	**def ++(xs: Map[(A, B)]): Map[A, B]** Returns a new map containing mappings of this map and those provided by xs.
2	**def -(elem1: A, elem2: A, elems: A*): Map[A, B]** Returns a new map containing all the mappings of this map except mappings with a key equal to elem1, elem2 or any of elems.
3	**def --(xs: GTO[A]): Map[A, B]** Returns a new map with all the key/value mappings of this map except mappings with a key equal to a key from the traversable object xs.
4	**def get(key: A): Option[B]** Optionally returns the value associated with a key.
5	**def iterator: Iterator[(A, B)]** Creates a new iterator over all key/value pairs of this map
6	**def addString(b: StringBuilder): StringBuilder** Appends all elements of this shrinkable collection to a string builder.
7	**def addString(b: StringBuilder, sep: String): StringBuilder** Appends all elements of this shrinkable collection to a string builder using a separator string.
8	**def apply(key: A): B** Returns the value associated with the given key, or the result of the map's default method, if none exists.
9	**def clear(): Unit** Removes all bindings from the map. After this operation has completed, the map will be empty.
10	**def clone(): Map[A, B]** Creates a copy of the receiver object.
11	**def contains(key: A): Boolean** Returns true if there is a binding for key in this map, false otherwise.
12	**def copyToArray(xs: Array[(A, B)]): Unit** Copies values of this shrinkable collection to an array. Fills the given array xs with values of this shrinkable collection.
13	**def count(p: ((A, B)) => Boolean): Int** Counts the number of elements in the shrinkable collection which satisfy a predicate.
14	**def default(key: A): B** Defines the default value computation for the map, returned when a key is not found.
15	**def drop(n: Int): Map[A, B]** Returns all elements except first n ones.
16	**def dropRight(n: Int): Map[A, B]** Returns all elements except last n ones
17	**def dropWhile(p: ((A, B)) => Boolean): Map[A, B]** Drops longest prefix of elements that satisfy a predicate.

18	**def empty: Map[A, B]** Returns the empty map of the same type.
19	**def equals(that: Any): Boolean** Returns true if both maps contain exactly the same keys/values, false otherwise.
20	**def exists(p: ((A, B)) => Boolean): Boolean** Returns true if the given predicate p holds for some of the elements of this shrinkable collection, otherwise false.
21	**def filter(p: ((A, B))=> Boolean): Map[A, B]** Returns all elements of this shrinkable collection which satisfy a predicate.
22	**def filterKeys(p: (A) => Boolean): Map[A, B]** Returns an immutable map consisting only of those key value pairs of this map where the key satisfies the predicate p.
23	**def find(p: ((A, B)) => Boolean): Option[(A, B)]** Finds the first element of the shrinkable collection satisfying a predicate, if any.
24	**def foreach(f: ((A, B)) => Unit): Unit** Applies a function f to all elements of this shrinkable collection.
25	**def init: Map[A, B]** Returns all elements except the last.
26	**def isEmpty: Boolean** Tests whether the map is empty.
27	**def keys: Iterable[A]** Returns an iterator over all keys.
28	**def last: (A, B)** Returns the last element.
29	**def max: (A, B)** Finds the largest element.
30	**def min: (A, B)** Finds the smallest element.
31	**def mkString: String** Displays all elements of this shrinkable collection in a string.
32	**def product: (A, B)** Returns the product of all elements of this shrinkable collection with respect to the * operator in num.
33	**def remove(key: A): Option[B]** Removes a key from this map, returning the value associated previously with that key as an option.
34	**def retain(p: (A, B) => Boolean): Map.this.type** Retains only those mappings for which the predicate p returns true.
35	**def size: Int** Return the number of elements in this map.
36	**def sum: (A, B)** Returns the sum of all elements of this shrinkable collection with respect to the +

	operator in num.
37	**def tail: Map[A, B]** Returns all elements except the first.
38	**def take(n: Int): Map[A, B]** Returns first n elements.
39	**def takeRight(n: Int): Map[A, B]** Returns last n elements.
40	**def takeWhile(p: ((A, B)) => Boolean): Map[A, B]** Takes longest prefix of elements that satisfy a predicate.
41	**def toArray: Array[(A, B)]** Converts this shrinkable collection to an array.
42	**def toBuffer[B >: A]: Buffer[B]** Returns a buffer containing all elements of this map.
43	**def toList: List[A]** Returns a list containing all elements of this map.
44	**def toSeq: Seq[A]** Returns a seq containing all elements of this map.
45	**def toSet: Set[A]** Returns a set containing all elements of this map.
46	**def toString(): String** Returns a String representation of the object.

Sr.No	Set Methods with Description
1	**def +(elem: A): Set[A]** Creates a new set with an additional element, unless the element is already present.
2	**def -(elem: A): Set[A]** Creates a new set with a given element removed from this set.
3	**def contains(elem: A): Boolean** Returns true if elem is contained in this set, false otherwise.
4	**def &(that: Set[A]): Set[A]** Returns a new set consisting of all elements that are both in this set and in the given set.
5	**def &~(that: Set[A]): Set[A]** Returns the difference of this set and another set.
6	**def +(elem1: A, elem2: A, elems: A*): Set[A]** Creates a new immutable set with additional elements from the passed sets
7	**def ++(elems: A): Set[A]** Concatenates this immutable set with the elements of another collection to this immutable set.
8	**def -(elem1: A, elem2: A, elems: A*): Set[A]** Returns a new immutable set that contains all elements of the current immutable set except one less occurrence of each of the given argument elements.
9	**def addString(b: StringBuilder): StringBuilder** Appends all elements of this immutable set to a string builder.
10	**def addString(b: StringBuilder, sep: String): StringBuilder** Appends all elements of this immutable set to a string builder using a separator string.
11	**def apply(elem: A)** Tests if some element is contained in this set.
12	**def count(p: (A) => Boolean): Int** Counts the number of elements in the immutable set which satisfy a predicate.
13	**def copyToArray(xs: Array[A], start: Int, len: Int): Unit** Copies elements of this immutable set to an array.
14	**def diff(that: Set[A]): Set[A]** Computes the difference of this set and another set.
15	**def drop(n: Int): Set[A]]** Returns all elements except first n ones.
16	**def dropRight(n: Int): Set[A]** Returns all elements except last n ones.
17	**def dropWhile(p: (A) => Boolean): Set[A]** Drops longest prefix of elements that satisfy a predicate.

18	**def equals(that: Any): Boolean** The equals method for arbitrary sequences. Compares this sequence to some other object.
19	**def exists(p: (A) => Boolean): Boolean** Tests whether a predicate holds for some of the elements of this immutable set.
20	**def filter(p: (A) => Boolean): Set[A]** Returns all elements of this immutable set which satisfy a predicate.
21	**def find(p: (A) => Boolean): Option[A]** Finds the first element of the immutable set satisfying a predicate, if any.
22	**def forall(p: (A) => Boolean): Boolean** Tests whether a predicate holds for all elements of this immutable set.
23	**def foreach(f: (A) => Unit): Unit** Applies a function f to all elements of this immutable set.
24	**def head: A** Returns the first element of this immutable set.
25	**def init: Set[A]** Returns all elements except the last.
26	**def intersect(that: Set[A]): Set[A]** Computes the intersection between this set and another set.
27	**def isEmpty: Boolean** Tests if this set is empty.
28	**def iterator: Iterator[A]** Creates a new iterator over all elements contained in the iterable object.
29	**def last: A** Returns the last element.
30	**def map[B](f: (A) => B): immutable.Set[B]** Builds a new collection by applying a function to all elements of this immutable set.
31	**def max: A** Finds the largest element.
32	**def min: A** Finds the smallest element.
33	**def mkString: String** Displays all elements of this immutable set in a string.
34	**def mkString(sep: String): String** Displays all elements of this immutable set in a string using a separator string.
35	**def product: A** Returns the product of all elements of this immutable set with respect to the * operator in num.
36	**def size: Int** Returns the number of elements in this immutable set.

37	**def splitAt(n: Int): (Set[A], Set[A])** Returns a pair of immutable sets consisting of the first n elements of this immutable set, and the other elements.
38	**def subsetOf(that: Set[A]): Boolean** Returns true if this set is a subset of that, i.e. if every element of this set is also an element of that.
39	**def sum: A** Returns the sum of all elements of this immutable set with respect to the + operator in num.
40	**def tail: Set[A]** Returns a immutable set consisting of all elements of this immutable set except the first one.
41	**def take(n: Int): Set[A]** Returns first n elements.
42	**def takeRight(n: Int):Set[A]** Returns last n elements.
43	**def toArray: Array[A]** Returns an array containing all elements of this immutable set.
44	**def toBuffer[B >: A]: Buffer[B]** Returns a buffer containing all elements of this immutable set.
45	**def toList: List[A]** Returns a list containing all elements of this immutable set.
46	**def toMap[T, U]: Map[T, U]** Converts this immutable set to a map
47	**def toSeq: Seq[A]** Returns a seq containing all elements of this immutable set.
48	**def toString(): String** Returns a String representation of the object.