Scala List

List is used to store ordered elements. It extends LinearSeq trait. It is a class for immutable linked lists. This class is good for last-in-first-out (LIFO), stack-like access patterns.

It maintains order of elements and can contain duplicates elements also.

Scala List Example

In this example, we have created two lists. Here, both lists have different syntax to create list.

```
import scala.collection.immutable._

object MainObject{

   def main(args:Array[String]){

      var list = List(1,8,5,6,9,58,23,15,4)

       var list2:List[Int] = List(1,8,5,6,9,58,23,15,4)

       println(list)

       println(list2)

   }

}
```

Output:

```
List(1, 8, 5, 6, 9, 58, 23, 15, 4)

List(1, 8, 5, 6, 9, 58, 23, 15, 4)
```

Scala List Example: Applying Predefined Methods

```
import scala.collection.immutable._

object MainObject{

   def main(args:Array[String]){

      var list = List(1,8,5,6,9,58,23,15,4)

      var list2 = List(88,100)
```

```
        print("Elements: ")
        list.foreach((element:Int) => print(element+" "))       // Iterating using foreach loop
        print("\nElement at 2 index: "+list(2))          // Accessing element of 2 index
        var list3 = list ++ list2                        // Merging two list
        print("\nElement after merging list and list2: ")
        list3.foreach((element:Int)=>print(element+" "))
        var list4 = list3.sorted                        // Sorting list
        print("\nElement after sorting list3: ")
        list4.foreach((element:Int)=>print(element+" "))
        var list5 = list3.reverse                        // Reversing list elements
        print("\nElements in reverse order of list5: ")
        list5.foreach((element:Int)=>print(element+" "))


    }
}
```

Output:

```
Elements: 1 8 5 6 9 58 23 15 4
Element at 2 index: 5
Element after merging list and list2: 1 8 5 6 9 58 23 15 4 88 100
Element after sorting list3: 1 4 5 6 8 9 15 23 58 88 100
Elements in reverse order of list5: 100 88 4 15 23 58 9 6 5 8 1
```