Scala Set

It is used to store unique elements in the set. It does not maintain any order for storing elements. You can apply various operations on them. It is defined in the Scala.collection.immutable package.

Scala Set Syntax

val variableName:Set[Type] = Set(element1, element2,... elementN) or

val variableName = Set(element1, element2,... elementN)

Scala Set Example

In this example, we have created a set. You can create an empty set also. Let's see how to create a set.

```
import scala.collection.immutable._
object MainObject{
   def main(args:Array[String]){
      val set1 = Set()                        // An empty set
      val games = Set("Cricket","Football","Hocky","Golf")    // Creating a set with elements
      println(set1)
      println(games)
   }
}
```

Output:

```
Set()     // an empty set
Set(Cricket,Football,Hocky,Golf)
```

Scala Set Example 2

In Scala, Set provides some predefined properties to get information about set. You can get

first or last element of Set and many more. Let's see an example.

```
import scala.collection.immutable._
object MainObject{
      def main(args:Array[String]){
         val games = Set("Cricket","Football","Hocky","Golf")
         println(games.head)           // Returns first element present in the set
         println(games.tail)        // Returns all elements except first element.
         println(games.isEmpty)         // Returns either true or false
      }
   }
```

Output:

```
Cricket
Set(Football, Hocky, Golf)
false
```

Scala Set Example: Merge two Set

You can merge two sets into a single set. Scala provides a predefined method to merge sets. In this example, ++ method is used to merge two sets.

```
import scala.collection.immutable._
object MainObject{
      def main(args:Array[String]){
         val games = Set("Cricket","Football","Hocky","Golf")
         val alphabet = Set("A","B","C","D","E")
         val mergeSet = games ++ alphabet           // Merging two sets
         println("Elements in games set: "+games.size)   // Return size of collection
         println("Elements in alphabet set: "+alphabet.size)
```

```
        println("Elements in mergeSet: "+mergeSet.size)
        println(mergeSet)
    }
  }
```

Output:

```
Elements in games set: 4
Elements in alphabet set: 5
Elements in mergeSet: 9
Set(E, Football, Golf, Hocky, A, B, C, Cricket, D)
```

This example also proves that the merge set does not maintain order to store elements.

Scala Set Example 2

You can check whether element is present in the set or not. The following example describe the use of contains() method.

```
import scala.collection.immutable._
object MainObject{
    def main(args:Array[String]){
        val games = Set("Cricket","Football","Hocky","Golf")
        println(games)
        println("Elements in set: "+games.size)
        println("Golf exists in the set : "+games.contains("Golf"))
        println("Racing exists in the set : "+games.contains("Racing"))

    }
  }
```

Output:

Set(Cricket, Football, Hocky, Golf)

Elements in set: 4

Golf exists in the set : true

Racing exists in the set : false

Scala Set Example: Adding and Removing Elements

You can add or remove elements from the set. You can add only when your code is mutable. In this example, we are adding and removing elements of the set.

```
import scala.collection.immutable._

object MainObject{

    def main(args:Array[String]){

      var games = Set("Cricket","Football","Hocky","Golf")

      println(games)

      games += "Racing"          // Adding new element

      println(games)

      games += "Cricket"         // Adding new element, it does not allow duplicacy.

      println(games)

      games -= "Golf"          // Removing element

      println(games)

    }

  }
```

Output:

Set(Cricket, Football, Hocky, Golf)

Set(Football, Golf, Hocky, Cricket, Racing)

Set(Football, Golf, Hocky, Cricket, Racing)

Set(Football, Hocky, Cricket, Racing)

Scala Set Example: Iterating Set Elements using for loop

You can iterate set elements either by using for loop or foreach loop. You can also filter elements during iteration. In this example have used for loop to iterate set elements.

```
import scala.collection.immutable._

object MainObject{

    def main(args:Array[String]){

      var games = Set("Cricket","Football","Hocky","Golf")

      for(game <- games){

        println(game)

      }

    }

  }
```

Output:

```
Cricket

Football

Hocky

Golf
```

Scala Set Example Iterating Elements using foreach loop

In this example, we are using foreach loop to iterate set elements.

```
import scala.collection.immutable._

  object MainObject{

    def main(args:Array[String]){

      var games = Set("Cricket","Football","Hocky","Golf")
```

```
        games.foreach((element:String)=> println(element))
    }
}
```

Output:

Cricket

Football

Hocky

Golf

Scala Set Example: Set Operations

In scala Set, you can also use typical math operations like: intersection and union. In the following example we have used predefined methods to perform set operations.

```
import scala.collection.immutable._
object MainObject{
  def main(args:Array[String]){
    var games = Set("Cricket","Football","Hocky","Golf","C")
    var alphabet = Set("A","B","C","D","E","Golf")
    var setIntersection = games.intersect(alphabet)
    println("Intersection by using intersect method: "+setIntersection)
    println("Intersection by using & operator: "+(games & alphabet))
    var setUnion = games.union(alphabet)
    println(setUnion)
  }
}
```

Output:

Intersection by using intersect method: Set(Golf, C)

Intersection by using & operator: Set(Golf, C)

Set(E, Football, Golf, Hocky, A, B, C, Cricket, D)

Scala SortedSet

In scala, SortedSet extends Set trait and provides sorted set elements. It is useful when you want sorted elements in the Set collection. You can sort integer values and string as well.

It is a trait and you can apply all the methods defined in the traversable trait and Set trait.

Scala SortedSet Example

In the following example, we have used SortedSet to store integer elements. It returns a Set after sorting elements.

```
import scala.collection.immutable.SortedSet

object MainObject{

   def main(args:Array[String]){

      var numbers: SortedSet[Int] = SortedSet(5,8,1,2,9,6,4,7,2)

      numbers.foreach((element:Int)=> println(element))

   }

}
```

Output:

1

2

4

5

6

7

8

9