SCALA LOOPS

Scala Pattern Matching

- Pattern matching is a feature of scala. It works same as switch case in other programming languages. It matches best case available in the pattern.
- Example

```
object MainObject {
  def main(args: Array[String]) {
      var a = 1
      a match{
         case 1 => println("One")
         case 2 => println("Two")
         case _ => println("No")
      }
      }
}
```

`Here, match using a variable named a. This variable matches with best available case and prints output. Underscore (_) is used in the last case for making it default case.

Example 2:

- Match expression can return case value also.
- In the next example,define method having a match with cases for any type of data.
- Any is a class in scala which is a super class of all data types and deals with all type of data. Let's see an example.

```
object MainObject {
  def main(args: Array[String]) {
      var result = search ("Hello")
      print(result)
   }
   def search (a:Any):Any = a match{
      case 1  => println("One")
      case "Two" => println("Two")
      case "Hello" => println("Hello")
      case _ => println("No")

      }
}
```

Scala while loop

- In Scala, while loop is used to iterate code till the specified condition. It tests boolean expressions and iterates again and again.
- It is recommended to use while loop if you don't know number of iterations prior.
- **Syntax**

```
while(boolean expression){
   // Statements to be executed
}
```

- Example

```
object MainObject {
  def main(args: Array[String]) {
    var a = 10;                    // Initialization
    while( a<=20 ){                 // Condition
      println(a);
      a = a+2                      // Incrementation
    }
  }
}
```

Scala Infinite While Loop Example

```
object MainObject {
   def main(args: Array[String]) {
      var a = 10;           // Initialization
      while( true ){        // Condition
         println(a);
         a = a+2            // Incrementation
      }
   }
}
```

Scala do-while loop example

```
object MainObject {
   def main(args: Array[String]) {
        var a = 10;         // Initialization
        do {
            println( a );
            a = a + 2;      // Increment
        }
        while( a <= 20 )    // Condition
   }
}
```

Scala Infinite do-while loop

- To create infinite loop just pass true literal in loop condition.

```
object MainObject {
  def main(args: Array[String]) {
      var a = 10;                          // Initialization
      do {
          println( a );
          a = a + 2;                       // Increment
      }
      while( true)                         // Condition
  }
}
```

Scala for loop

- In scala, for loop is known as for-comprehensions.
- It can be used to iterate, filter and return an iterated collection.
- The for-comprehension looks a bit like a for-loop in imperative languages, except that it constructs a list of the results of all iterations.
- **Syntax**

```
for( i <- range){
  // statements to be executed
}
```

range is a value which has *start* and *end* point. can pass range by using **to** or **until** keyword.

Scala for-loop example by using to keyword

```
object MainObject {
   def main(args: Array[String]) {
        for( a <- 1 to 10 ){
         println(a);
      }
   }
}
```

Scala for-loop Example by using until keyword

```
object MainObject {
   def main(args: Array[String]) {
        for( a <- 1 until 10 ){
         println(a);
      }
   }
}
```

The major difference between *until* and *to* is, *to* includes start and end value given in the range, while *until* excludes last value of the range.

Scala for-loop filtering Example

filtering the data by passing a conditional expression.

```
object MainObject {
   def main(args: Array[String]) {
        for( a <- 1 to 10 if a%2==0 ){
         println(a);
      }
   }
}
```

Output:

```
2 4 6 8 10
```

Scala for-loop Example by using yield keyword

- used yield keyword which returns a result after completing of loop iterations.
- The for use buffer internally to store iterated result and after finishing all iterations it yields the final result from that buffer.
- It does not work like imperative loop.

Example

```
object MainObject {
  def main(args: Array[String]) {
       var result = for( a <- 1 to 10) yield a
       for(i<-result){
           println(i)
       }
  }
}
```

Scala for-loop in Collection

- In scala, can iterate collections like list, sequence etc, either by using for each loop or for-comprehensions.

```
object MainObject {
  def main(args: Array[String]) {
      var list = List(1,2,3,4,5,6,7,8,9)        // Creating a list
      for( i <- list){                          // Iterating the list
          println(i)
      }


  }
}
```

Scala for-each loop Example for Iterating Collection

```
object MainObject {
  def main(args: Array[String]) {
        var list = List(1,2,3,4,5,6,7,8,9)   // Creating a list
        list.foreach{
            println     // Print each element
        }
        list.foreach(print)
        println
        list.foreach((element:Int)=>print(element+" "))      // Explicitly mentioning
type of elements
  }
}
```

Scala for-loop Example using by keyword

- The by keyword is used to skip the iteration. When you code like: by 2 it means, this loop will skip all even iterations of loop.

```
object MainObject{
def main(args:Array[String]){
for(i<-1 to 10 by 2){
println(i)
}
}
```

Scala Break

- Break is used to break a loop or program execution.
- It skips the current execution.
- Inside inner loop it breaks the execution of inner loop.
- In scala, there is no break statement but you can do it by using break method and by importing scala.util.control.Breaks._ package.

```
import scala.util.control.Breaks              // Importing  package
object MainObject {
   def main(args: Array[String]) {
        breakable {                                  // Breakable method to avoid exception
            for(i<-1 to 10 by 2){
                if(i==7)
                    break                          // Break used here
                else
                    println(i)
            }
        }
    }
}
```