Scala Higher Order Functions

Higher order function is a function that either takes a function as argument or returns a function. In other words we can say a function which works with function is called higher order function.

Higher order function allows you to create function composition, lambda function or anonymous function etc.

Let's see an example.

Scala Example: Passing a Function as Parameter in a Function

```
object MainObject {
  def main(args: Array[String]) = {
    functionExample(25, multiplyBy2)                 // Passing a function as parameter
   }
   def functionExample(a:Int, f:Int=>AnyVal):Unit = {
      println(f(a))                              // Calling that function
   }
   def multiplyBy2(a:Int):Int = {
      a*2
   }
}
```

Scala Example: Function Composition

In scala, functions can be composed from other functions. It is a process of composing in which a function represents the application of two composed functions.

Let's see an example.

```
object MainObject {
  def main(args: Array[String]) = {
    var result = multiplyBy2(add2(10))     // Function composition
    println(result)
   }
   def add2(a:Int):Int = {
      a+2
   }

   def multiplyBy2(a:Int):Int = {
      a*2
   }
}
```

Scala Anonymous (lambda) Function

Anonymous function is a function that has no name but works as a function. It is good to create an anonymous function when you don't want to reuse it latter.

You can create anonymous function either by using => (rocket) or _ (underscore) wild card in scala.

Let's see an example.

Scala Anonymous function Example

```
object MainObject {
  def main(args: Array[String]) = {
    var result1 = (a:Int, b:Int) => a+b       // Anonymous function by using => (rocket)
    var result2 = (_:Int)+(_:Int)          // Anonymous function by using _ (underscore) wild card

    println(result1(10,10))

    println(result2(10,10))

   }

}
```

Output:

```
20

20
```

Scala Multiline Expression

Expressions those are written in multiple lines are called multiline expression. In scala, be carefull while using multiline expressions.

The following program explains about if we break an expression into multiline, the scala compiler throw a warning message.

Scala Multiline Expression Example

```
def add1(a:Int, b:Int) = {

       a

       +b

   }
```

The above program does not evaluate complete expression and just return b here. So, be careful while using multiline expressions.

Output:

```
MainObject.scala:3: warning: a pure expression does nothing in statement

position; you may be omitting necessary parentheses
```

```
        a
        ^
one warning found

10
```

You can apply following ways to avoid above problem.

Scala Example Multiline Expression

```
object MainObject {
    def add2(a:Int, b:Int) = {
        a+
        b
    }
    def add3(a:Int, b:Int) = {
        (a
        +b)
    }
    def main(args: Array[String]) = {
        var result2 = add2(10,10)
        var result3 = add3(10,10)
        println(result2+"\n"+result3)
    }
}
```

Output:

```
20
20
```

Scala Function Currying

In scala, method may have multiple parameter lists. When a method is called with a fewer number of parameter lists, then this will yield a function taking the missing parameter lists as its arguments.

In other words it is a technique of transforming a function that takes multiple arguments into a function that takes a single argument.

Scala Function Currying Example

```
object MainObject {
   def add(a:Int)(b:Int) = {
      a+b
   }
   def main(args: Array[String]) = {
      var result = add(10)(10)
      println("10 + 10 = "+result)
      var addIt = add(10)_
      var result2 = addIt(3)
      println("10 + 3 = "+result2)
   }
}
```

Scala Nested Functions

Scala is a first class function language which means it allows you to passing function, returning function, composing function, nested function etc. An example below explain about how to define and call nested functions.

Scala Nested Functions Example

```
object MainObject {
   def add(a:Int, b:Int, c:Int) = {
      def add2(x:Int,y:Int) = {
         x+y
      }
      add2(a,add2(b,c))
   }
   def main(args: Array[String]) = {
      var result = add(10,10,10)
      println(result)
   }
}
```

Output:

```
30
```

Scala Function with Variable Length Parameters

In scala, you can define function of variable length parameters. It allows you to pass any number of arguments at the time of calling the function.

Let's see an example.

Scala Example: Function with Variable Length Parameters

```
def add(args: Int*) = {
   var sum = 0;
   for(a <- args) sum+=a
   sum
}
```

```
var sum = add(1,2,3,4,5,6,7,8,9);

println(sum);
```