Scala Programming II

Closures

- Clouse는 함수이다

```
val multiplier = (i:Int) => i * 10
val multiplier = (i:Int) => i * factor
var factor = 3
val multiplier = (i:Int) => i * factor

object Demo {
   def main(args: Array[String]) {
      println( "multiplier(1) value = " +  multiplier(1) )
      println( "multiplier(2) value = " +  multiplier(2) )
   }
   var factor = 3
   val multiplier = (i:Int) => i * factor
}
```

Strings

- Java.lang.String class

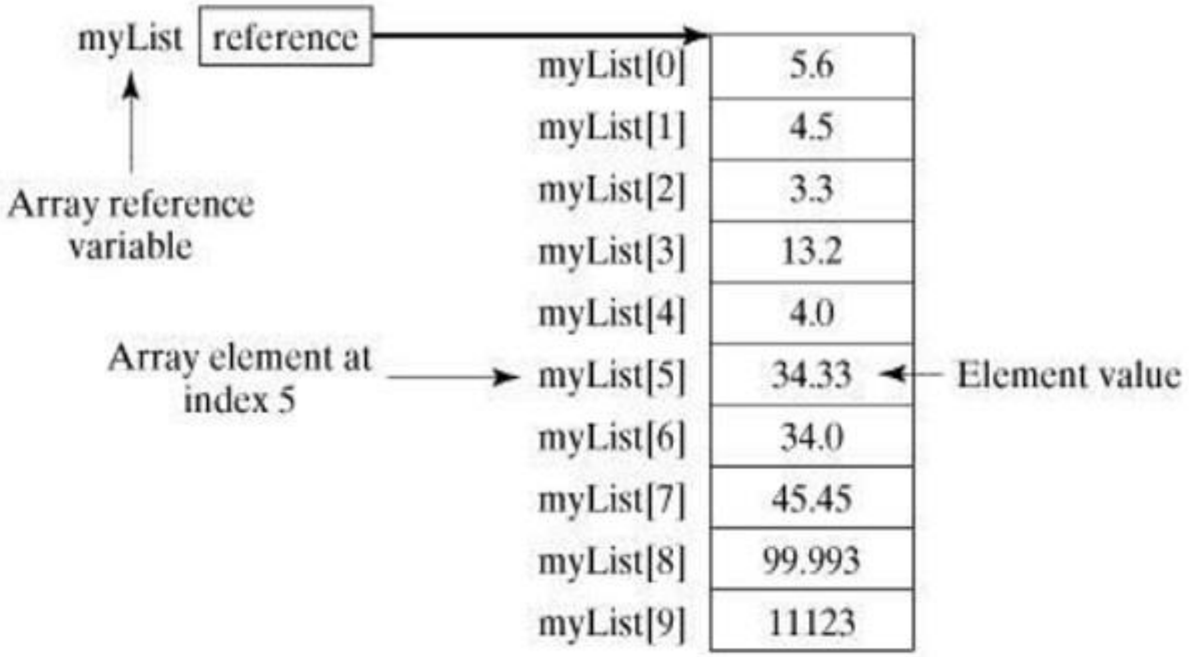
```
object Demo {
   val greeting: String = "Hello, world!"

   def main(args: Array[String]) {
      println( greeting )
   }
}
object Demo {
   def main(args: Array[String]) {
      var palindrome = "Dot saw I was Tod";
      var len = palindrome.length();

      println( "String Length is : " + len );
   }
}
```

Arrays

- var z:Array[String] = new Array[String](3)
- var z = new Array[String](3)
- z(0) = "Zara"; z(1) = "Nuha"; z(4/2) = "Ayan"
- var z = Array("Zara", "Nuha", "Ayan")



Arrays

```
object Demo {
   def main(args: Array[String]) {
      var myList = Array(1.9, 2.9, 3.4, 3.5)
      for ( x <- myList ) {
         println( x )
      }
      var total = 0.0;
      for ( i <- 0 to (myList.length - 1)) {
         total += myList(i);
      }
      println("Total is " + total);
       var max = myList(0);
      for ( i <- 1 to (myList.length - 1) ) {
         if (myList(i) > max) max = myList(i);
      }
      println("Max is " + max);
   }
}
```

Collections

- Lists
  - linked list of type T
- Sets
  - collection of pairwise different elements of the same type
- Maps
  - collection of key/value pairs
- Tuples
  - tuple can hold objects with different types
- Options
  - container for zero or one element of a given type
- Iterators
  - not a collection, way to access the elements of a collection one by one

Traits

```
trait Cards {
    def details(d:String):String
}

class Cardet extends Cards {
  import scala.io.Source
  override def details(source:String) = {
    Source.fromString(source).mkString
  }
}

object Demo {
  def main(args:Array[String]){
    val c1 = new Cardet
    println(c1.details("Car are being displayed"))
    println(c1.isInstanceOf[Cards])
  }
}
```

패턴 매칭(Pattern Matching)

```
object Demo {
  def main(args: Array[String]) {
    println(matchTest(3))
  }

  def matchTest(x: Int): String = x match {
    case 1 => "one"
    case 2 => "two"
    case _ => "many"
  }
}
```

정규 표현식(Regular Express)

```
object Demo {
  def main(args: Array[String]) {
    println(matchTest("two"))
    println(matchTest("test"))
    println(matchTest(1))
  }

  def matchTest(x: Any): Any = x match {
    case 1 => "one"
    case "two" => 2
    case y: Int => "scala.Int"
    case _ => "many"
  }
}
```

예외처리(Exception Handling)

```
import java.io.FileReader
import java.io.FileNotFoundException
import java.io.IOException

object Demo {
  def main(args: Array[String]) {
    try {
      val f = new FileReader("input.txt")
    } catch {
      case ex: FileNotFoundException =>{
        println("Missing file exception")
      }
      case ex: IOException => {
        println("IO Exception")
      }
    }
  }
```