Unit 3e: Scala Control & Data Structures

1 Selection 1
2 Iteration 1
3 Arrays and collections 2

1 Selection

In Java the *if* statement allows a path of execution to be selected on the basis of some condition.

In Scala the *if* expression allows evaluation to be selected on the basis of some condition. The difference is subtle, but important for functional programming as you will see shortly; it *Returns a value* which is the result of the selected evaluation.

```
scala> val x = 3
x: Int = 3

scala> val result = if(x<0){
     | "negative"} else {
     | "positive"}
result: String = positive
```

In Java the switch statement selects the execution path based on a case value.

In Scala a match expression allows evaluation to be selected on the basis of case; this is much more powerful than switch and is the basis of pattern matching which is important in functional programming. It *Returns a value* which is the result of the selected evaluation.

```
scala> var month = "Feb"
month: String = Feb
scala> var season = month match{
    | case "Dec"|"Jan"|"Feb" => "winter"
    | case "Mar"|"Apr"|"May" => "spring"
    | case "Jun"|"Jul"|"Aug" => "summer"
    | case "Sep"|"Oct"|"Nov" => "autumn"
    |}
season: String = winter
```

2 Iteration

Java has a while loop, a do while loop, a for loop, and an enhanced for loop.

Scala has a while loop and a for loop but, you will see later that, loops are used much less commonly in functional programming than in imperative programming. *For expressions* iterate through a collection and evaluate an expression for each member – *yield* each and collect results. Many data structure classes have a *foreach* method that applies a function to each member in turn.

```
scala> for (i <- 1 to 10 if i % 2 == 0)
    | yield i
res0:  scala.collection.immutable.
IndexedSeq[Int]0 = Vector(2, 4, 6, 8, 10)

scala> val x = List(1,2,3)
x: List[Int] = List(1, 2, 3)
scala> x.foreach { println }
1
2
3
```

3 Arrays and collections

Java Arrays hold multiple values or objects of the same type. The Collections framework contains many library classes for collections of values or objects, each with methods appropriate to the data structure it implements, e.g. ArrayList.

Scala Arrays hold multiple objects of the same type; Collections exist as in Java such as List, Vector, and Map. You should distinguish between *immutable* (never change) and *mutable*; the default is usually *immutable* in functional programming.

```
scala> var z:Array[String] = new Array[String](3)
z: Array[String] = Array(null, null, null)

scala> z(0) = "Hello"; z(1) = "Bonjour"; z(2) = "Hei"

scala> var z = Array("Hello","Bonjour","Hei")
z: Array[String] = Array(Hello, Bonjour, Hei)

scala> val z:List[String] = List("Hello","Bonjour","Hei")
z: List[String] = List(Hello, Bonjour, Hei)
```