Unit 3b: Scala Basics

1 Expressions and statements

In Java an expression is something that can be *evaluated*, e.g. 2 + 3; while a statement is something that can be *executed*, e.g. y = 2 + 3; .

Scala has expressions and statements, but many things that would be statements in Java are expressions – the result is shown when executed in a REPL.

```
scala> val y = 2+3
y: Int = 5
```

Note the colour coding is used here to differentiate between input to and output from the REPL, you don't actually see colours in the REPL.

Examples include: Assignment ; *if*; Loops; Function definitions etc.

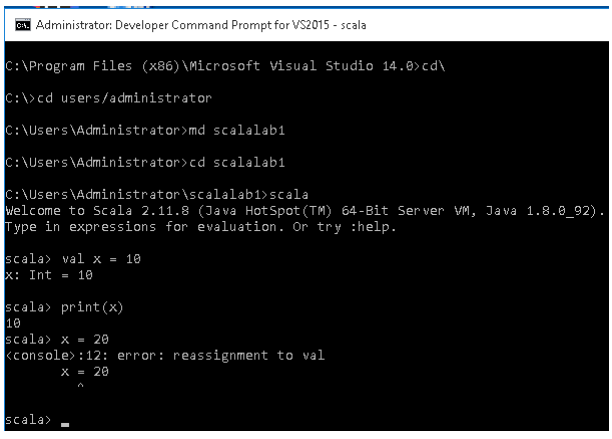
2 Immutable values

In Java variables can be declared with an additional key word **final** ensuring the value can't be changed. This can be useful, e.g. to define named constants, but we work with *mutable* variables mostly i.e. the value can be changed after it is initially assigned.

Scala variables are declared with either ***val*** or ***var*** keywords.

```
val name = "Jim": String
var age = 21: Int
```
(may omit type if it can be inferred)

val makes a variable *immutable*, *var* makes it *mutable*.



As you will see, functional programming style prefers immutability, so it is usual to use *val* unless there is a good reason to have a mutable variable. Scala supports (and encourages) functional style, but doesn't enforce it.

3 Operators

Java has primitive types and object types. Operators are *special symbols that perform specific operations* on *operands* and then return a result e.g. *+* is a binary operator that (among other things) performs addition on two numeric operands; it is not valid if the operands are objects unless they are instances of the type wrapper classes (e.g. *Integer*), in which case a compiler ensures these are converted to the corresponding primitive at runtime.

Scala has no primitive types - everything is an object (Scala has classes and objects, like Java). Scala has value types, e.g. Int, but these are classes not primitives. Operators are nothing special, they are *just methods*: *+* is just a method of the class *Int* e.g. 3.+(4) adds two *Ints* by called the + method of one and passing the other as a parameter - the *infix notation* allows this to be written as 3 + 4. Any method that takes one parameter can be used as an "operator" using this notation.

```
scala> var y = 5
y: Int = 5

scala> x.+(y)
res1: Int = 15

scala>
```

4 Some special types

Java	Scala	Comment on Scala type
Object	**Any, ScalaObject**	Any is base for all types, Scala has subclasses *AnyVal* and *AnyRef* as base classes for value and reference types. *ScalaObject* derives from *AnyRef* and is base class for all classes
void	**Unit**	*Unit* is a common concept in FP as functions should evaluate to some value Has value **()**, represents a value that contains no information, rather than no value
	Nothing	Base type of all others, has no value, only use as return type if function will never return
null	**Option – *Some[val] or None***	Useful in situations where a function may need to return a value or no value depending on the input,