Kotlin Variables & Functions

Sources: http://kotlinlang.org/docs/reference/basic-syntax.html
http://petersommerhoff.com/dev/kotlin/kotlin-for-java-devs/

Local Variables – val (read-only)

Assign-once (read-only) local variable:

```
val a: Int = 1  // immediate assignment
val b = 2       // `Int` type is inferred
val c: Int      // Type required when no initializer is provided
c = 3           // deferred assignment
```

```
a = 1, b = 2, c = 3
```

Target platform: JVM Running on kotlin v. 1.1.50

Local Variables – val (read-only)

Assign-once (read-only) local variable:

```
val a: Int = 1  // immediate assignment
val b = 2       // `Int` type is inferred
val c: Int      // Type required when no initializer is provided
c = 3           // deferred assignment
c = 4           // Syntax error: cannot reassign value
```

Target platform: JVM Running on kotlin v. 1.1.50

Local Variables – var (mutable)

Mutable variable:

```
var x = 5 // `Int` type is inferred
x += 1    // Mathematical operations permitted on Int
```

```
x = 6
```

Target platform: JVM Running on kotlin v. 1.1.50

Local Variables – var (mutable)

Mutable variable:

```
var x = 5 // `Int` type is inferred
x += 1    // Mathematical operations permitted on Int
x = 5     // Allowed to reassign value
```

```
x = 5
```

Target platform: JVM Running on kotlin v. 1.1.50

Functions

Parameters, return types, expression body, inferred return type

Functions – parameters and return types

Function having two `Int` parameters with `Int` return type:

```
1 fun sum(a: Int, b: Int): Int {
2     return a + b
3 }
4
5 fun main(args: Array<String>) {
6     print("sum of 3 and 5 is ")
7     println(sum(3, 5))
8 }
```

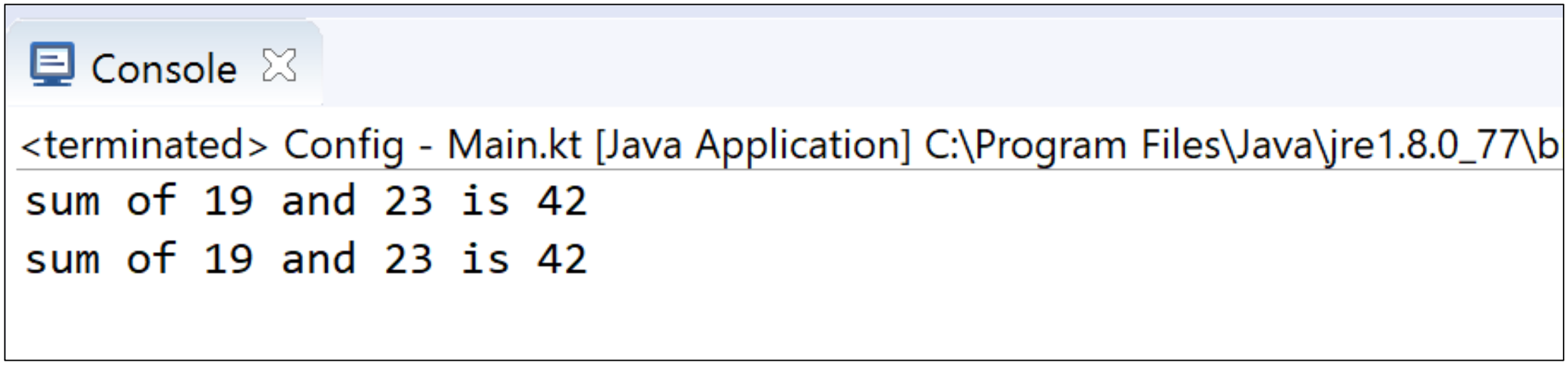
```
sum of 3 and 5 is 8
```

Functions – expression body, inferred return type

Function “sum” with an expression body and inferred return type

```
fun sum(a: Int, b: Int) = a + b

fun main(args: Array<String>) {
    println("sum of 19 and 23 is ${sum(19, 23)}")
    println("sum of 19 and 23 is " + sum(19, 23))
}
```



Functions – no return data

Function returning no meaningful value:

```
1  fun printSum(a: Int, b: Int): Unit {
2      println("sum of $a and $b is ${a + b}")
3  }
4
5  fun main(args: Array<String>) {
6      printSum(-1, 8)
7  }
```

```
sum of -1 and 8 is 7
```

`Unit` return type can be omitted:

```
1  fun printSum(a: Int, b: Int) {
2      println("sum of $a and $b is ${a + b}")
3  }
4
5  fun main(args: Array<String>) {
6      printSum(-1, 8)
7  }
```

```
sum of -1 and 8 is 7
```

Arguments

default and named

Default Arguments (optional)

In Java, you often have to duplicate code in order **define different variants of a method or constructor (i.e. overloading)**.

Kotlin simplifies this by using default values for arguments (i.e. makes them optional arguments).

Default Arguments (optional)

```
   class NutritionFacts(     val foodName:
String,
                     val calories: Int,
                     val protein: Int = 0,
                     val carbohydrates: Int = 0,
                     val fat: Int = 0,
                     val description: String = "")
   {
   }
```

Primary Constructor

Optional Parameters

```
 val pizza  = NutritionFacts("Pizza",  442, 12, 27, 24, "Deep Pan Pizza")
 val pasta = NutritionFacts("Pasta", 371, 14, 25, 11)
 val soup  = NutritionFacts("Soup",  210)
```

Some possible constructor calls

Named Arguments

```
   class NutritionFacts(  val
foodName: String,
                  val calories: Int,
                  val protein: Int = 0,
                  val carbohydrates:
Int = 0,
                  val fat: Int = 0,
                  val description:
String = "")
   {
```

```
val pasta   = NutritionFacts("Pasta", 371, 14, 25, 11)
val burger = NutritionFacts("Hamburger", calories = 541, f
= 33, protein = 14)
val rice      = NutritionFacts("Rice", 312, carbohydrates
23, description = “Grains")
```

Naming arguments make your code more readible