Kotlin Collections

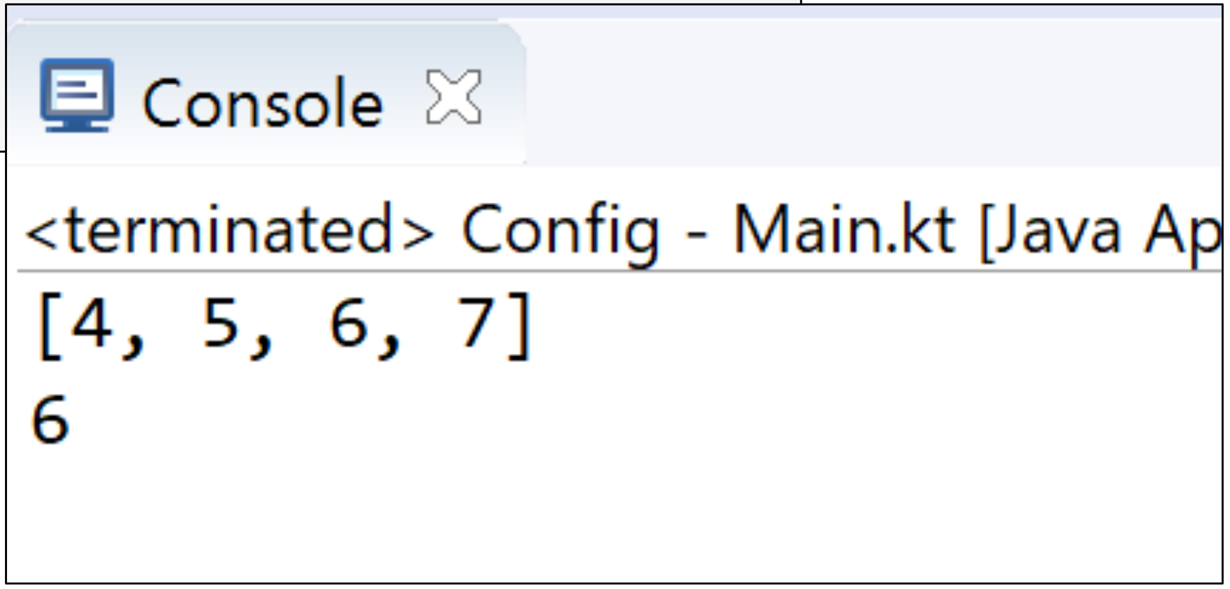
Sources:

http://kotlinlang.org/docs/reference/basic-syntax.html
http://petersommerhoff.com/dev/kotlin/kotlin-for-java-devs/
https://www.programiz.com/kotlin-programming
https://medium.com/@napperley/kotlin-tutorial-5-basic-collections-3f114996692b

Arrays (using arrayOf)

Arrays in Kotlin can be created using **arrayOf()** or the **Array()** constructor.

```kotlin
fun main(args: Array<String>) {

  val myArray = arrayOf(4, 5, 6, 7)
  println(myArray.asList())
  print(myArray[2])

}
```

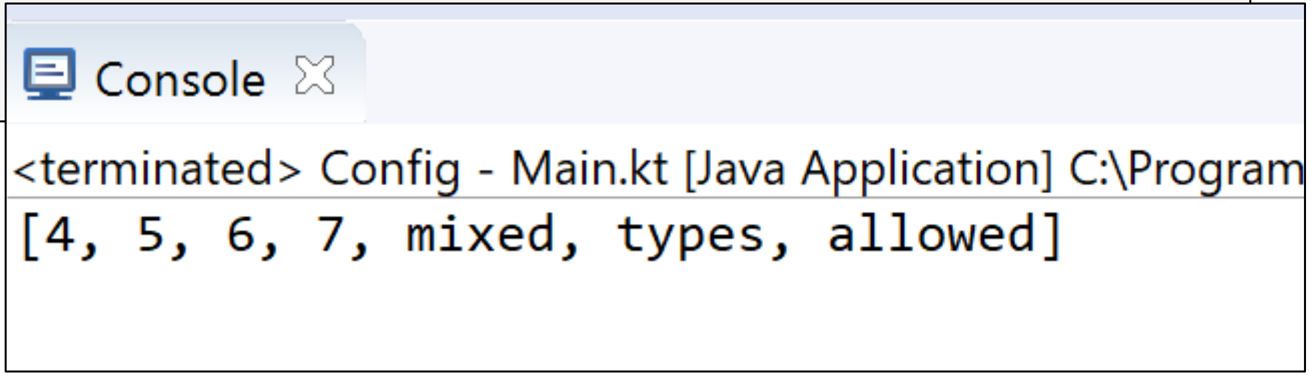


Console

```
<terminated> Config - Main.kt [Java Ap
[4, 5, 6, 7]
6
```

Arrays (using arrayOf)

```
fun main(args: Array<String>) {

    val myArray = arrayOf(4, 5, 6, 7, "mixed", “types", "allowed")
   print(myArray.asList())
}
```



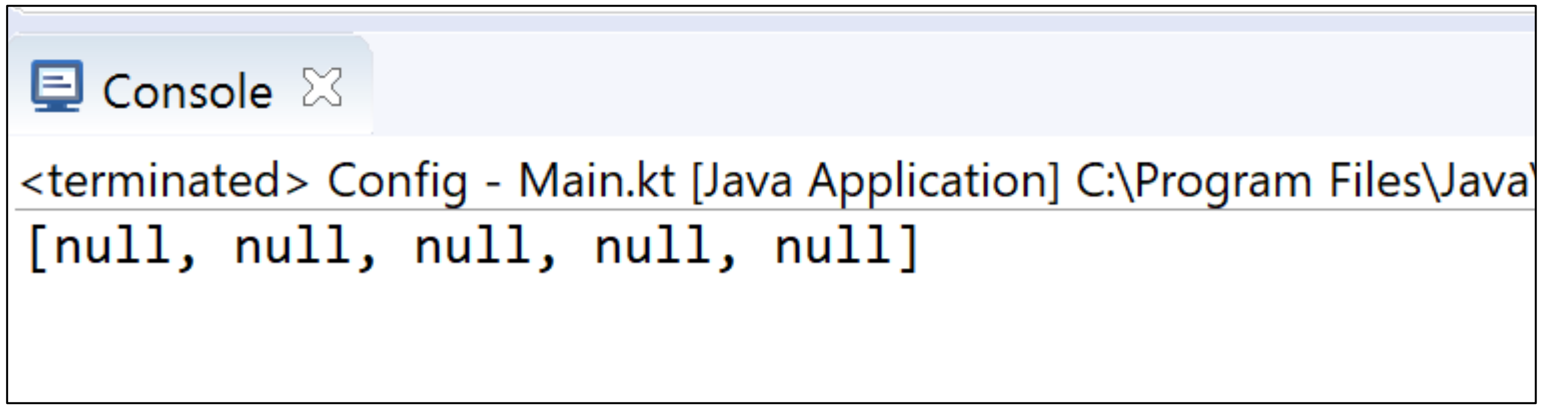
Console

```
<terminated> Config - Main.kt [Java Application] C:\Program
[4, 5, 6, 7, mixed, types, allowed]
```

```
fun main(args: Array<String>) {

  val intArray1     = intArrayOf(4, 5, 6, 7)
  val intArray2     = arrayOf<Int>(4, 5, 6, 7)
  val charArray     = charArrayOf('a', 'b', 'c', 'd’)
  val booleanArray = booleanArrayOf(true, false, true)

  val mixedArray1 = intArrayOf(4, 5, 6, 7, "will","not","compile")
  val mixedArray2 = arrayOf<Int>(4, 5, 6, 7,"will","not","compile")

}
```

Arrays (using arrayOfNulls)

```
fun main(args: Array<String>) {

    val nullArray = arrayOfNulls<Int>(5);
    println (nullArray.asList())

}
```



Console

```
<terminated> Config - Main.kt [Java Application] C:\Program Files\Java
[null, null, null, null, null]
```

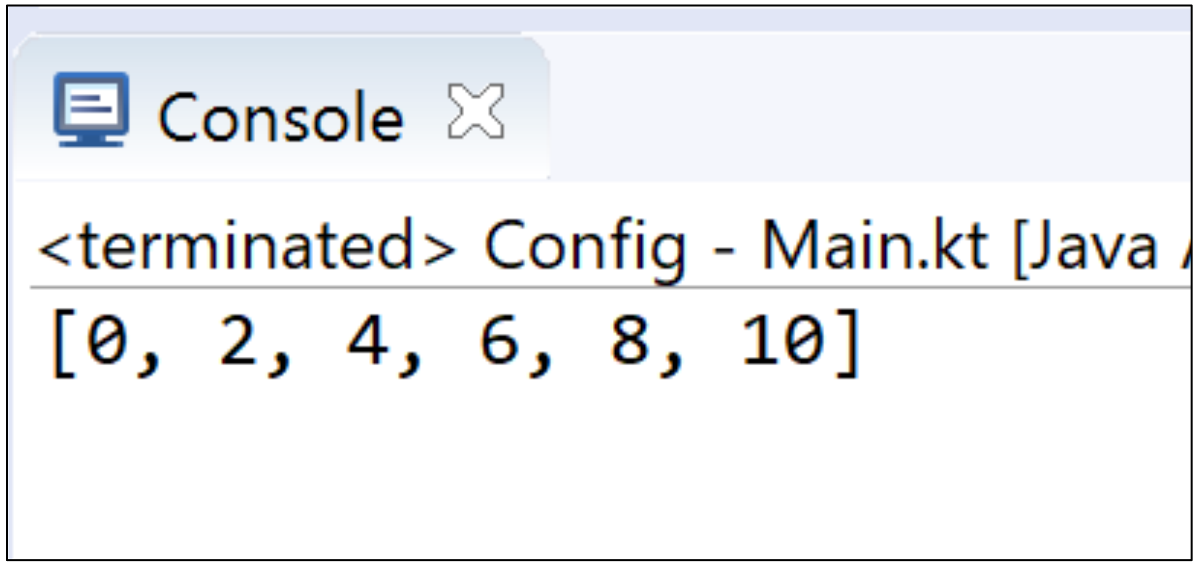
Arrays (using constructor)

The **Array()** constructor requires a size and a lambda function.

```
fun main(args: Array<String>) {

   val intArray = Array(6, { i -> i * 2 })
   print (intArray.asList())

}
```

Index of the array element

Value to be inserted into the index



```
<terminated> Config - Main.kt [Java
[0, 2, 4, 6, 8, 10]
```

Collections

Unlike many languages, Kotlin distinguishes between **mutable** and **immutable** collections (lists, sets, maps, etc).

Precise control over exactly when collections can be edited is useful for eliminating bugs, and for designing good APIs.

Collections – mutable vs immutable

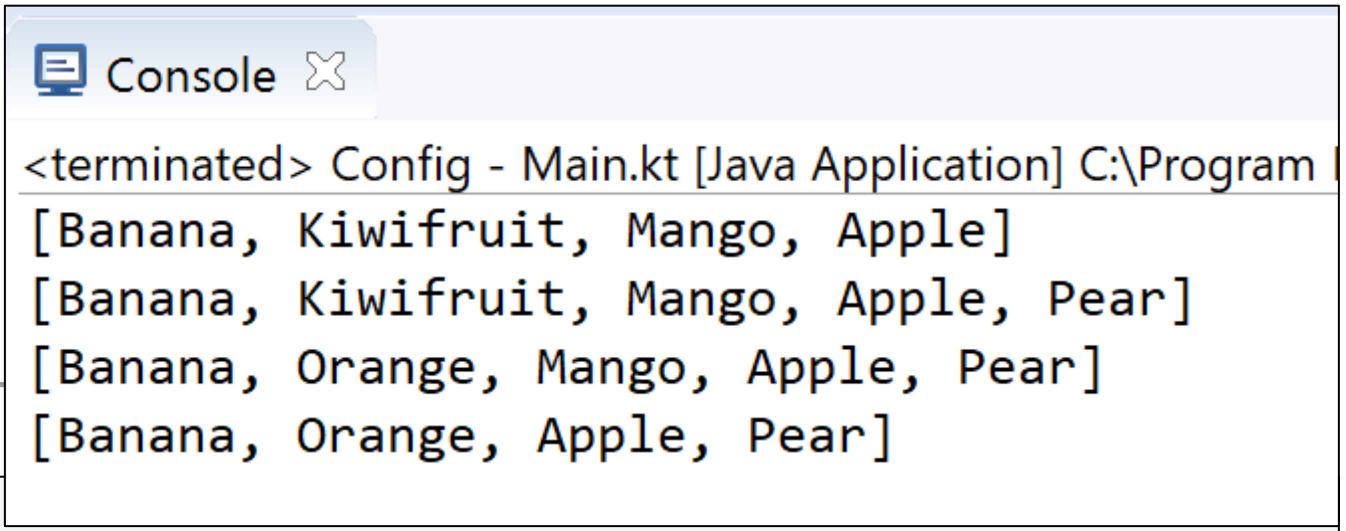
The Kotlin List<out T> type is an interface that provides read-only operations like size, get and so on.

Like in Java, it inherits from Collection<T> and that in turn inherits from Iterable<T>.

Methods that change the list are added by the MutableList<T> interface.

This pattern holds also for Set<out T>/MutableSet<T> and Map<K, out V>/MutableMap<K, V>.

Collections – mutable List


Console

```
<terminated> Config - Main.kt [Java Application] C:\Program
[Banana, Kiwifruit, Mango, Apple]
[Banana, Kiwifruit, Mango, Apple, Pear]
[Banana, Orange, Mango, Apple, Pear]
[Banana, Orange, Apple, Pear]
```

```
fun main(args: Array<String>) {

  // Create a mutable list (MutableList).
  val fruit = mutableListOf("Banana", "Kiwifruit", "Mango", "Apple")
  println(fruit)

  // Add a element to the list.
  fruit.add("Pear")
  println(fruit)

  // Change an element in the list.
  fruit[1] = "Orange"
  println(fruit)

  // Remove a existing element from the list.
  fruit.removeAt(2)
  println(fruit)

}
```

Collections – immutable List – example 1

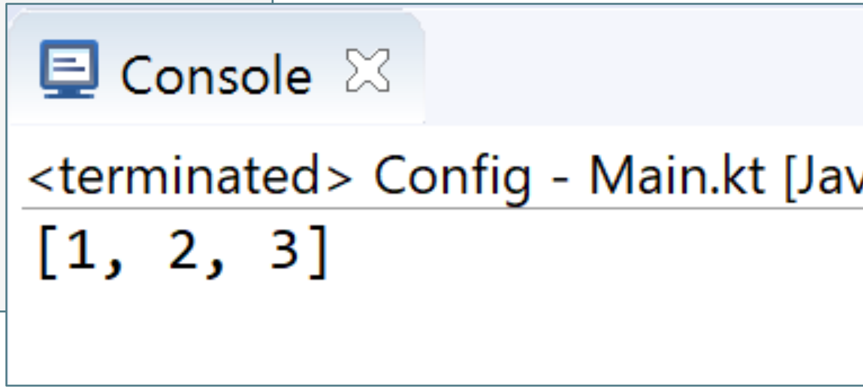
```kotlin
fun main(args: Array<String>) {

   val numbers: MutableList<Int> = mutableListOf(1, 2, 3)
   val readOnlyView: List<Int> = numbers

   println(numbers)          // prints "[1, 2, 3]"
   numbers.add(4)

   println(readOnlyView)     // prints "[1, 2, 3, 4]"
   readOnlyView.clear()      // -> does not compile

}
```

Collections – immutable List – example 2

```
class Person(  _firstName: String = "UNKNOWN",
               _lastName: String = "UNKNOWN") {

   private val _items = mutableListOf<String>("1", "2", "3")
   val items: List<String> get() = _items.toList()
}
```

items returns a snapshot of a collection at a particular point in time (that's guaranteed to not change). **toList()** just duplicates the items.

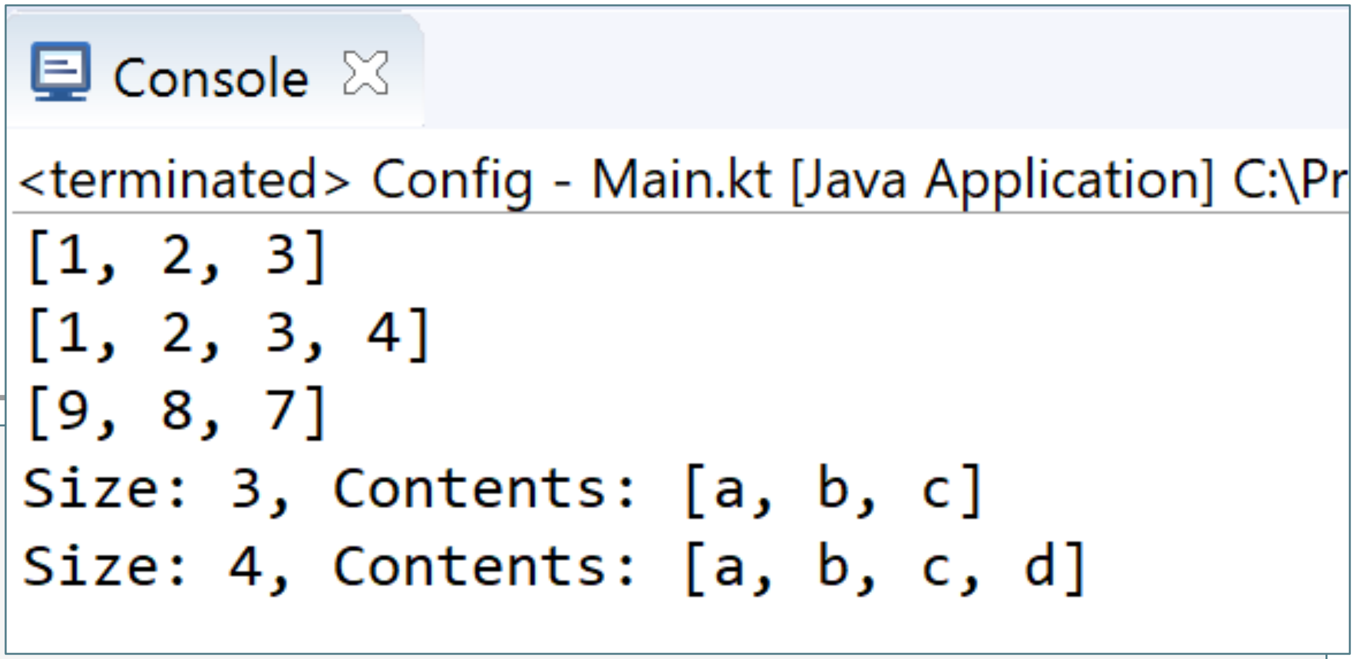
```
fun main(args: Array<String>) {

    val person = Person()
    println(person.items)

    //person.items.clear()  //doesn't compile

}
```



```
<terminated> Config - Main.kt [Jav
[1, 2, 3]
```

Collections – Set and hashSet

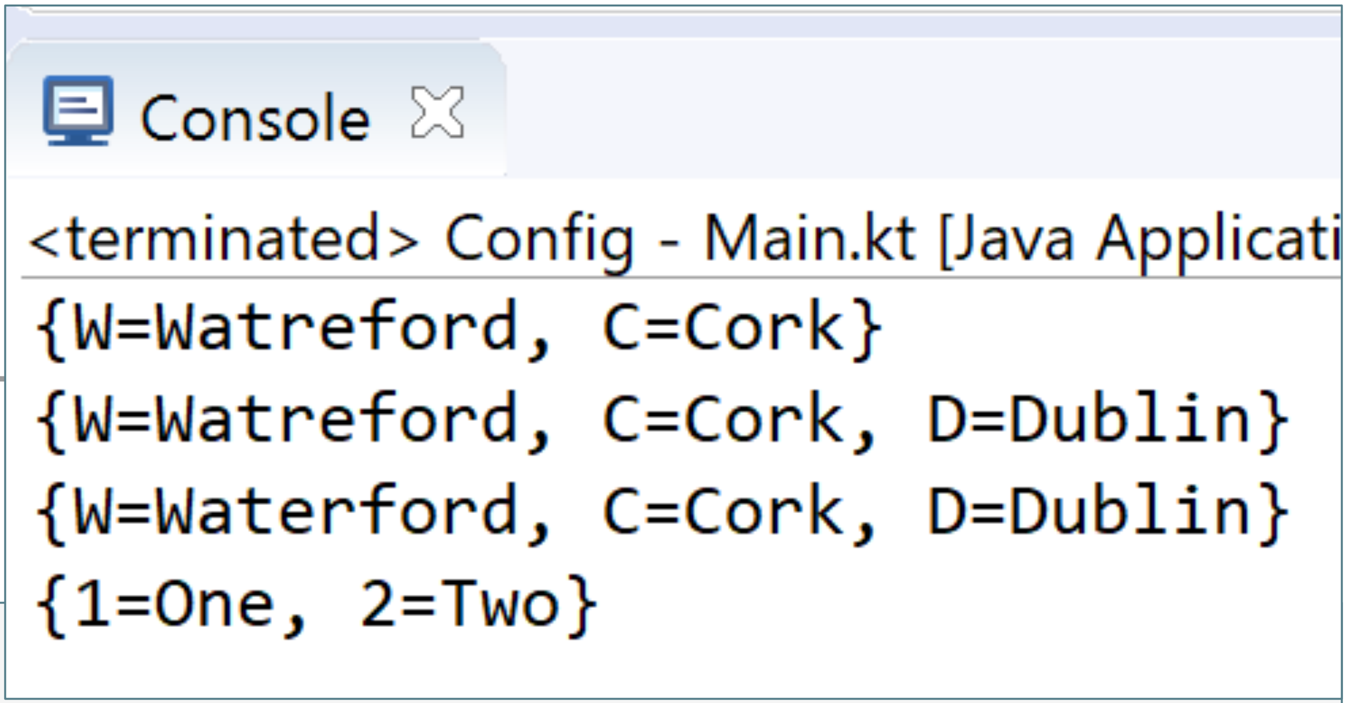
```
fun main(args: Array<String>) {

    // mutatble set
    val mutableSet : MutableSet<Int> = mutableSetOf(1,2,3)
    println(mutableSet)
    mutableSet.add(4)
    println(mutableSet)

    // immutatble set
    val immutableSet : Set<Int> = setOf(9,8,7)
    println(immutableSet)
    //immutableSet.add(6)  //won't compile

    //note: ignores duplicate items
    val strings = hashSetOf("a", "b", "c", "c")
    println("Size: ${strings.size}, Contents: " + strings)
    strings.add("d")
    println("Size: ${strings.size}, Contents: " + strings)

}
```



Console

```
<terminated> Config - Main.kt [Java Application] C:\Pr
[1, 2, 3]
[1, 2, 3, 4]
[9, 8, 7]
Size: 3, Contents: [a, b, c]
Size: 4, Contents: [a, b, c, d]
```

Collections – Map and hashMap



Console

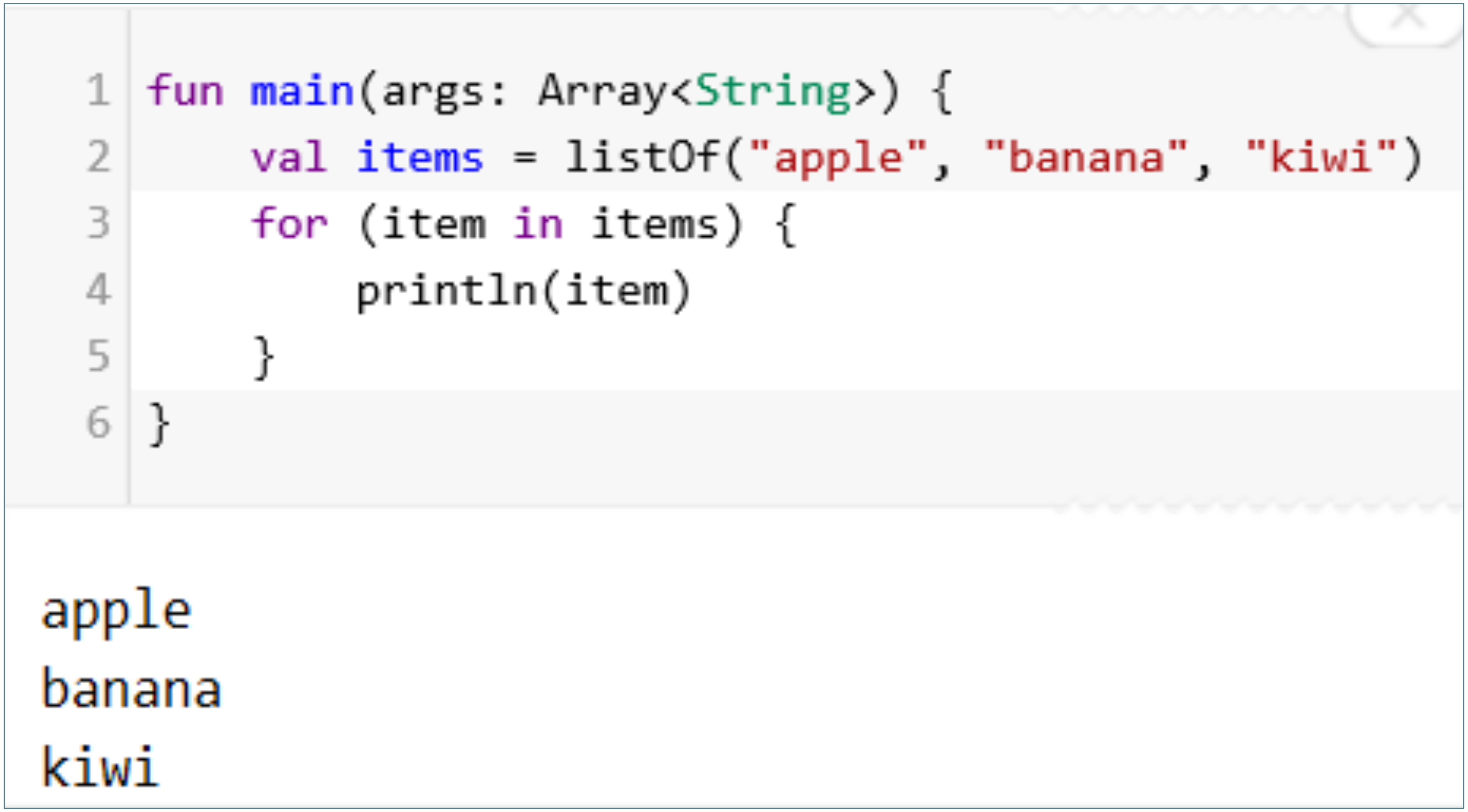
```
<terminated> Config - Main.kt [Java Applicati
{W=Watreford, C=Cork}
{W=Watreford, C=Cork, D=Dublin}
{W=Waterford, C=Cork, D=Dublin}
{1=One, 2=Two}
```

```kotlin
fun main(args: Array<String>) {

   // mutatble map
   val mutableMap = mutableMapOf("W" to "Watreford", "C" to "Cork")
   println(mutableMap)
   mutableMap.put("D", "Dublin")
   println(mutableMap)
   mutableMap["W"] = "Waterford"
   println(mutableMap)

   // immutatble map
   val immutableMap : Map<Int, String> = mapOf(1 to "One", 2 to "Two")
   println(immutableMap)
   //immutableMap.put(3, "Three")  //won't compile

}
```

Collections

The in operator and using lambdas

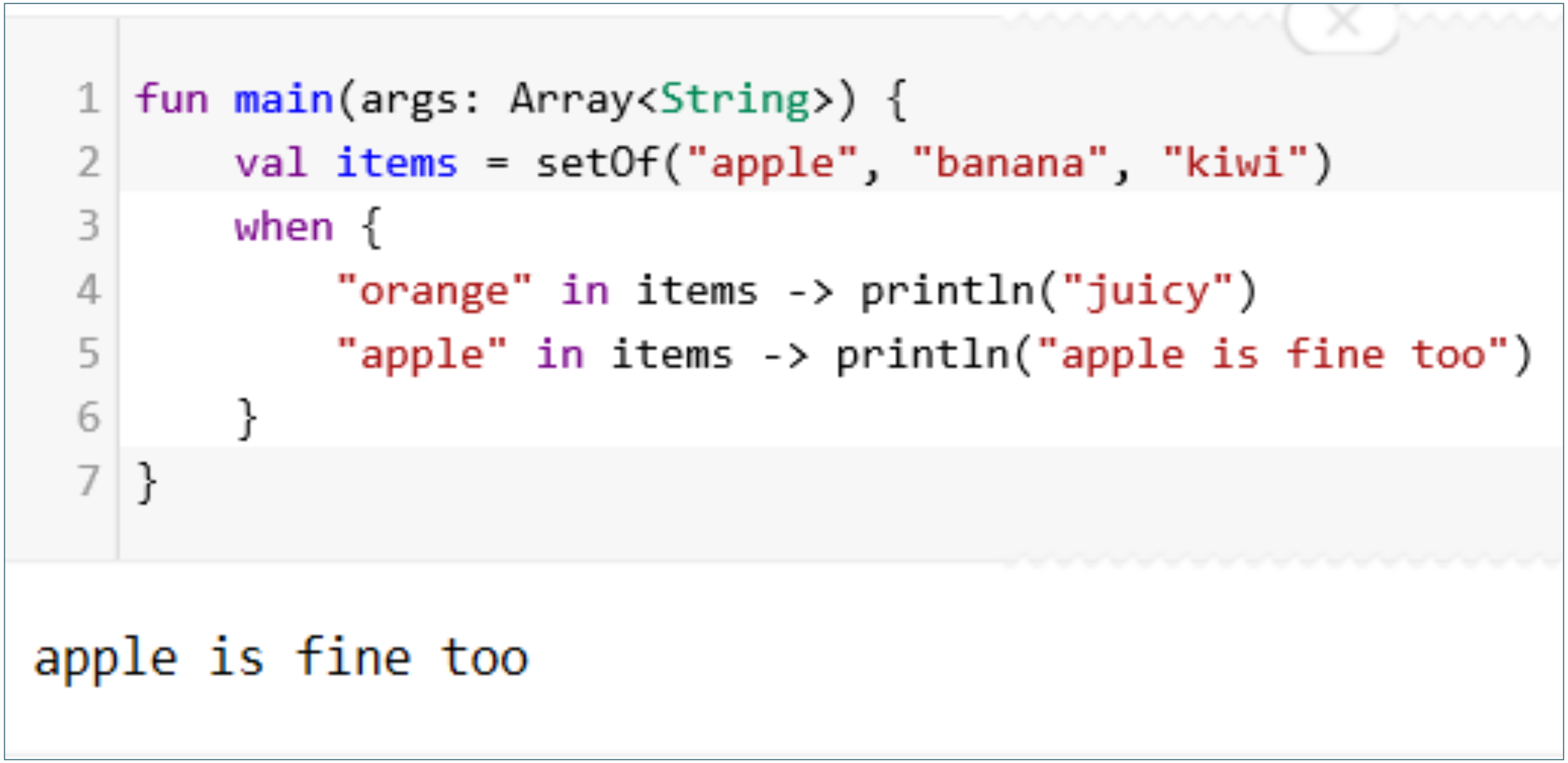
Collections – iterating using the in operator



```
fun main(args: Array<String>) {
    val items = listOf("apple", "banana", "kiwi")
    for (item in items) {
        println(item)
    }
}
```

```
apple
banana
kiwi
```

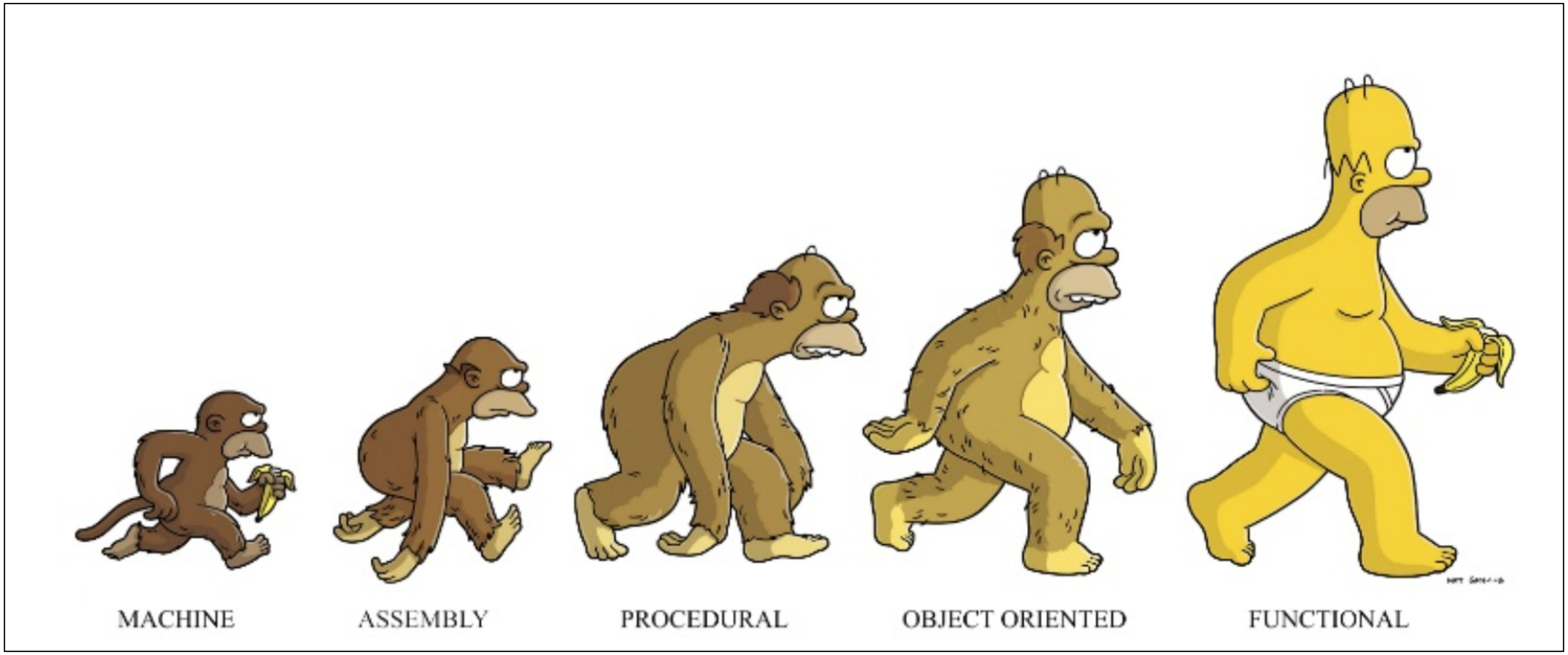
Collections – checking if collection contains an object



```
fun main(args: Array<String>) {
    val items = setOf("apple", "banana", "kiwi")
    when {
        "orange" in items -> println("juicy")
        "apple" in items -> println("apple is fine too")
    }
}
```

```
apple is fine too
```

Kotlin….functional programming is prevalent!



Collections – lambdas

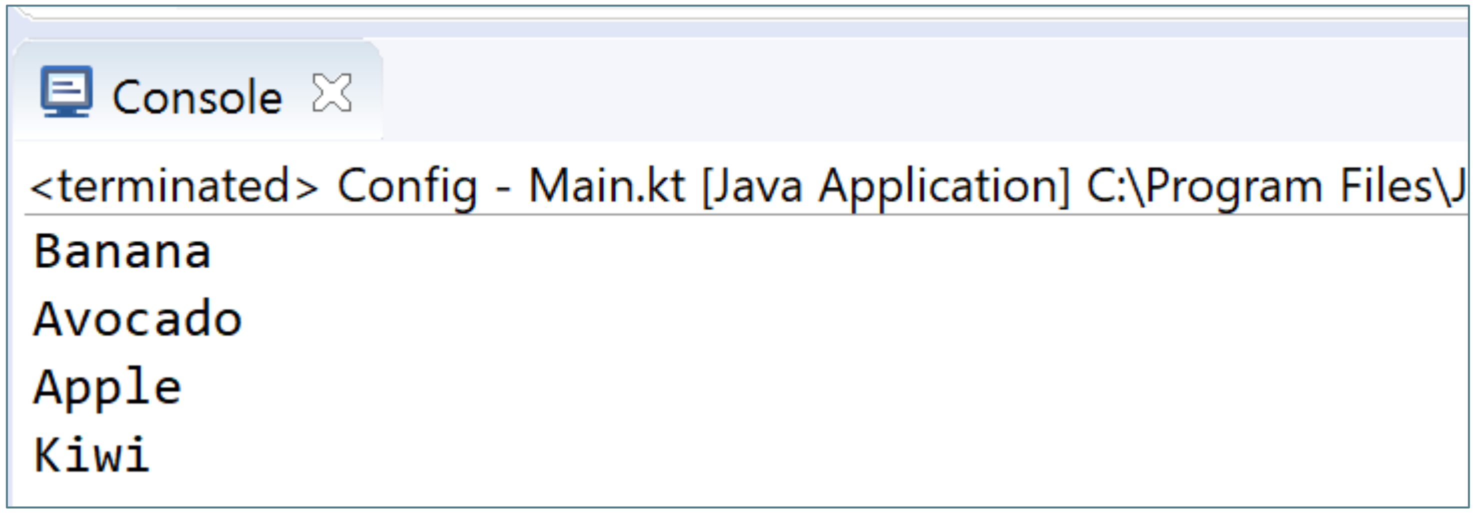
- You can pass an anonymous function (a lambda) as a parameter of a function.
- A lambda expression is always surrounded by curly braces.
- Its parameters (if any) are declared before **->** (parameter types may be omitted),
- The body goes after **->** (when present).
- An implicit variable called “it” is created and refers to the lambda expression’s only argument.

Collections – lambdas

Using lambda expressions to filter and map collections

```
fun main(args: Array<String>) {

  val fruits = listOf ("Banana", "Avocado", "Apple", "Kiwi")
  fruits.forEach  {it -> println(it)}
}
```

it -> is optional

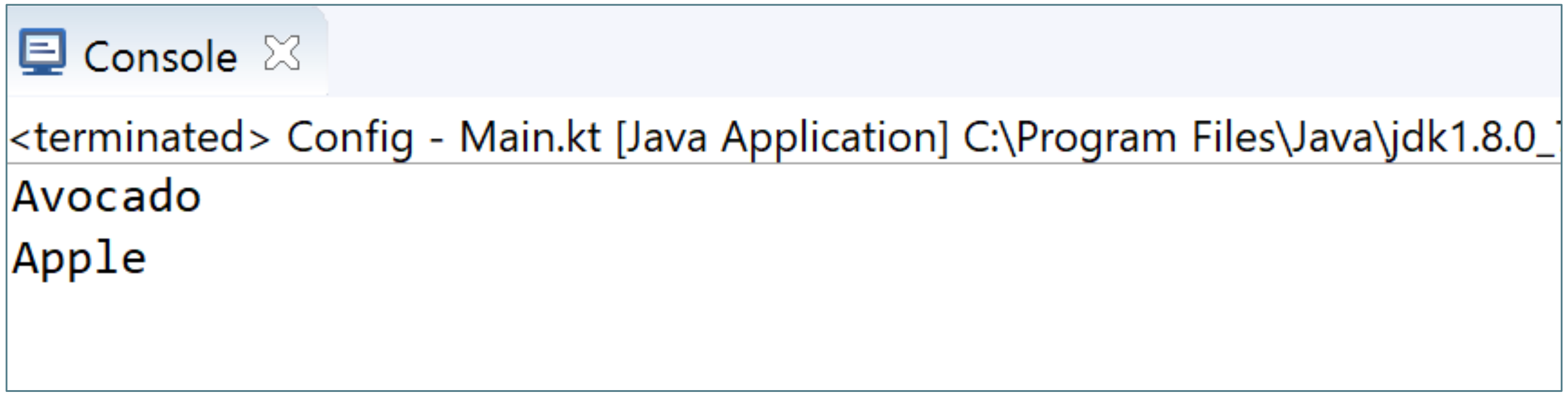

Console

```
<terminated> Config - Main.kt [Java Application] C:\Program Files\J
Banana
Avocado
Apple
Kiwi
```

Collections – lambdas

Using lambda expressions to filter and map collections

```
fun main(args: Array<String>) {

  val fruits = listOf ("Banana", "Avocado", "Apple", "Kiwi")
  fruits.filter   {it.startsWith("A")
        .forEach  {println(it)}
}
```

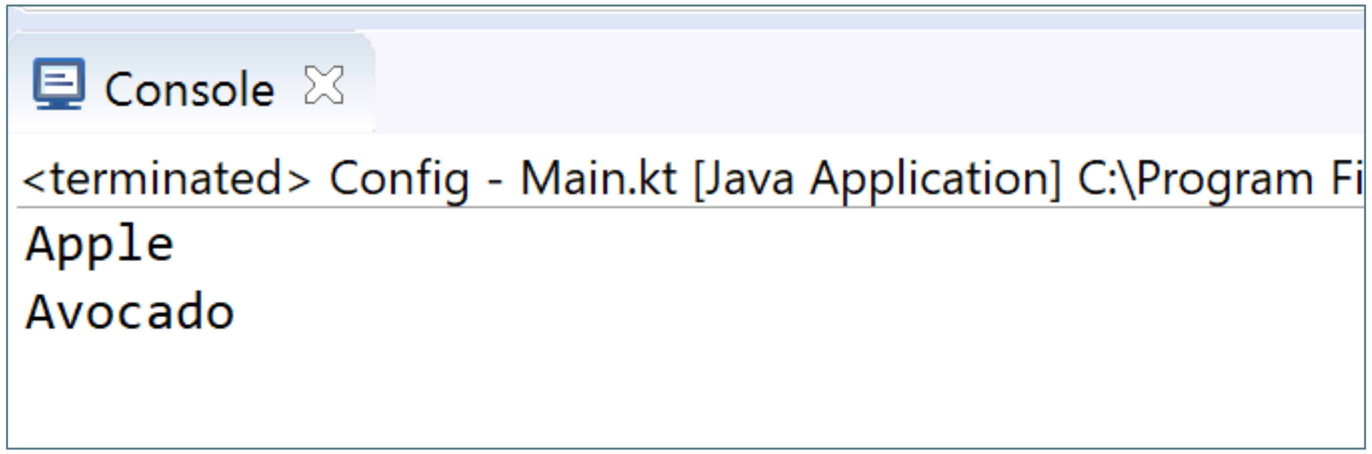


```
<terminated> Config - Main.kt [Java Application] C:\Program Files\Java\jdk1.8.0_
Avocado
Apple
```

Collections – lambdas

Using lambda expressions to filter and map collections

```
fun main(args: Array<String>) {

  val fruits = listOf ("Banana", "Avocado", "Apple", "Kiwi")
  fruits.filter   {it.startsWith("A")
        .sortedBy { it }
        .forEach  {println(it)}
}
```

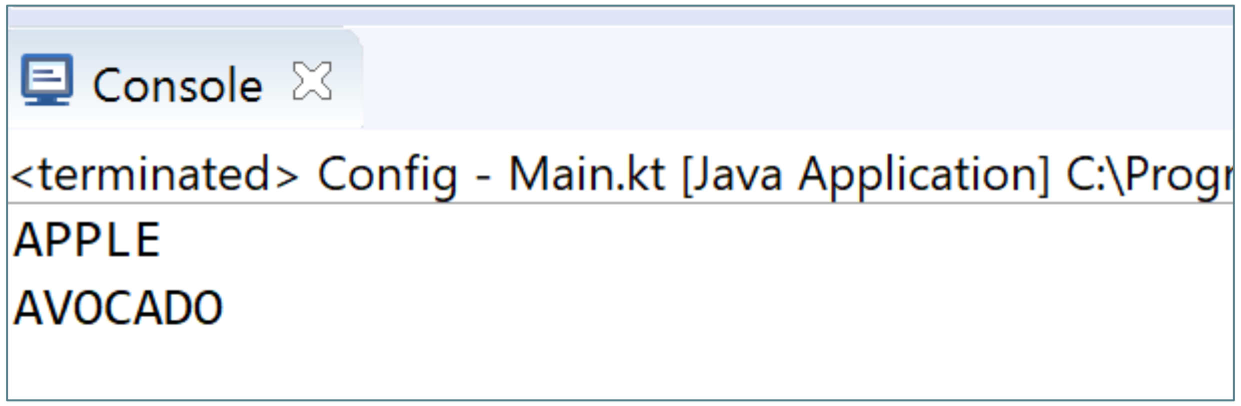


```
<terminated> Config - Main.kt [Java Application] C:\Program Fi
Apple
Avocado
```

Collections – lambdas

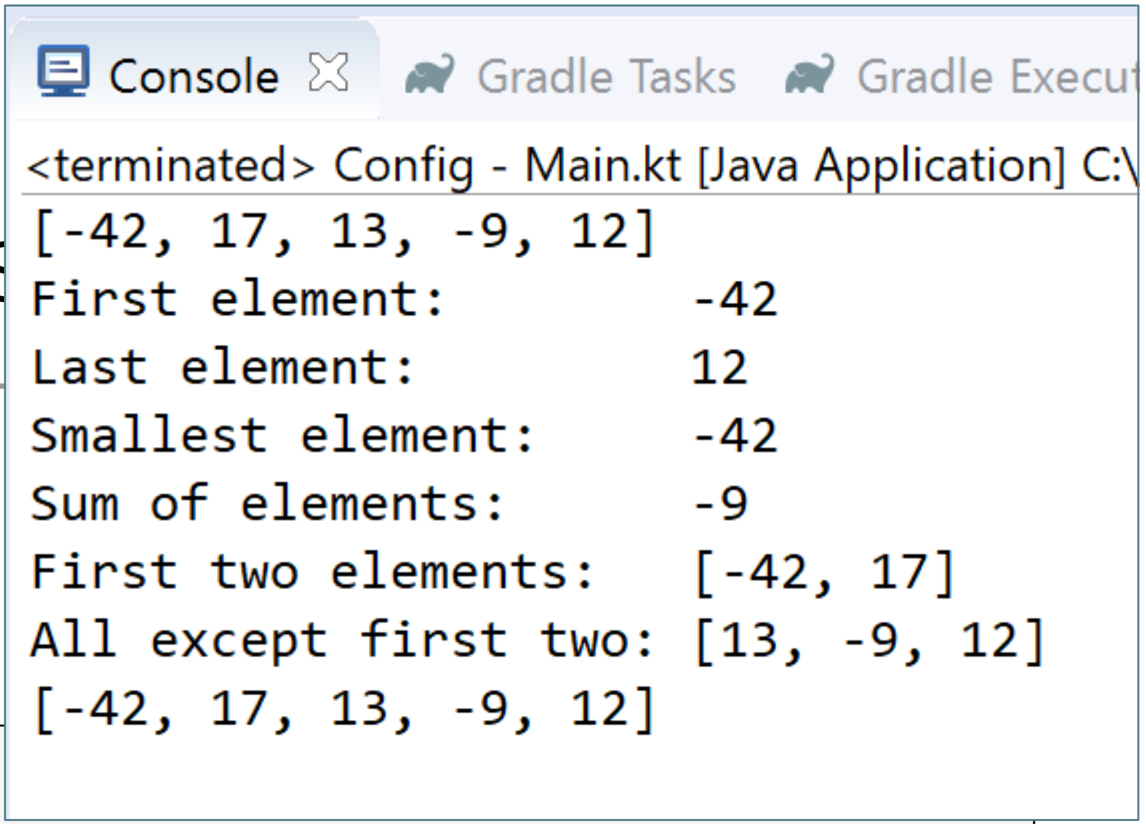
Using lambda expressions to filter and map collections

```
fun main(args: Array<String>) {

  val fruits = listOf ("Banana", "Avocado", "Apple", "Kiwi")
  fruits.filter   {it.startsWith("A")}
        .sortedBy { it }
        .map      {it.toUpperCase()}
        .forEach  {println(it)}
}
```



```
<terminated> Config - Main.kt [Java Application] C:\Progr
APPLE
AVOCADO
```

Collections – sample functions

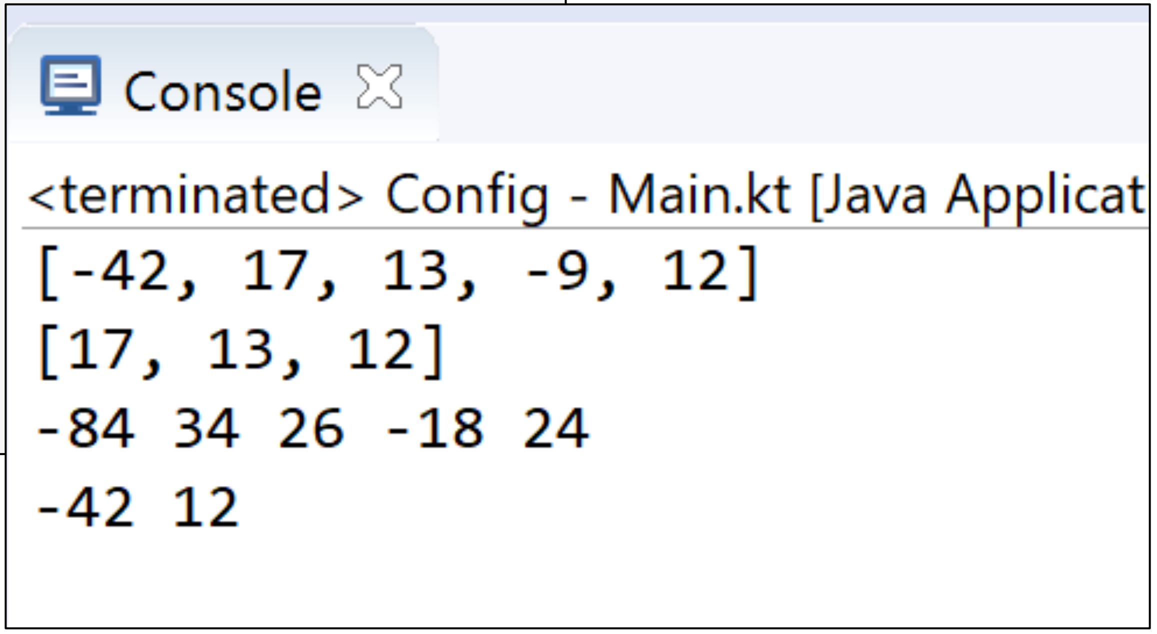

Console | Gradle Tasks | Gradle Execut

```
<terminated> Config - Main.kt [Java Application] C:\
[-42, 17, 13, -9, 12]
First element:          -42
Last element:           12
Smallest element:       -42
Sum of elements:        -9
First two elements:     [-42, 17]
All except first two:  [13, -9, 12]
[-42, 17, 13, -9, 12]
```

```kotlin
fun main(args: Array<String>) {

  val numbers = listOf(-42, 17, 13, -9, 12)
  println(numbers)

  println("First element:          " + numbers.first())
  println("Last element:           " + numbers.last())
  println("Smallest element:       " + numbers.min())
  println("Sum of elements:        " + numbers.foldRight
                                       (0, { a, b -> a + b }))
  println("First two elements:     " + numbers.take(2))
  println("All except first two: " + numbers.drop(2))

  println(numbers)
}
```

Collections – sample functions

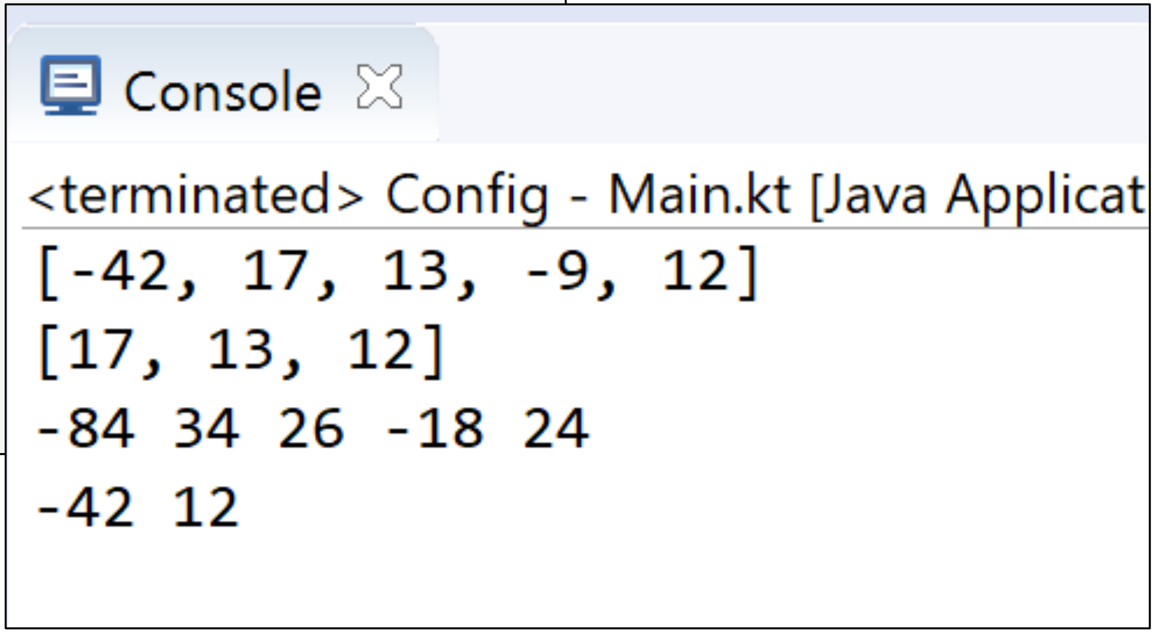
```kotlin
fun main(args: Array<String>) {

   val numbers = listOf(-42, 17, 13, -9, 12)
   println(numbers)

   // New list only containing non-negative numbers
   val nonNegative = numbers.filter { it >= 0 }
   println(nonNegative)

   // Double each element
   numbers.forEach { print("${it * 2} ") }
   println();

   // Output Even elements only
   numbers.filter {it % 2 == 0}
          .forEach {print ("$it " )}
   println();

}
```


Console

```
<terminated> Config - Main.kt [Java Applicat
[-42, 17, 13, -9, 12]
[17, 13, 12]
-84 34 26 -18 24
-42 12
```

Sets and Lambdas

```kotlin
fun main(args: Array<String>) {

  val numbers = setOf(-42, 17, 13, -9, 12)
  println(numbers)

  // New list only containing non-negative numbers
  val nonNegative = numbers.filter { it >= 0 }
  println(nonNegative)

  // Double each element
  numbers.forEach { print("${it * 2} ") }
  println();

  // Output Even elements only
  numbers.filter {it % 2 == 0}
         .forEach {print ("$it " )}
  println();

}
```



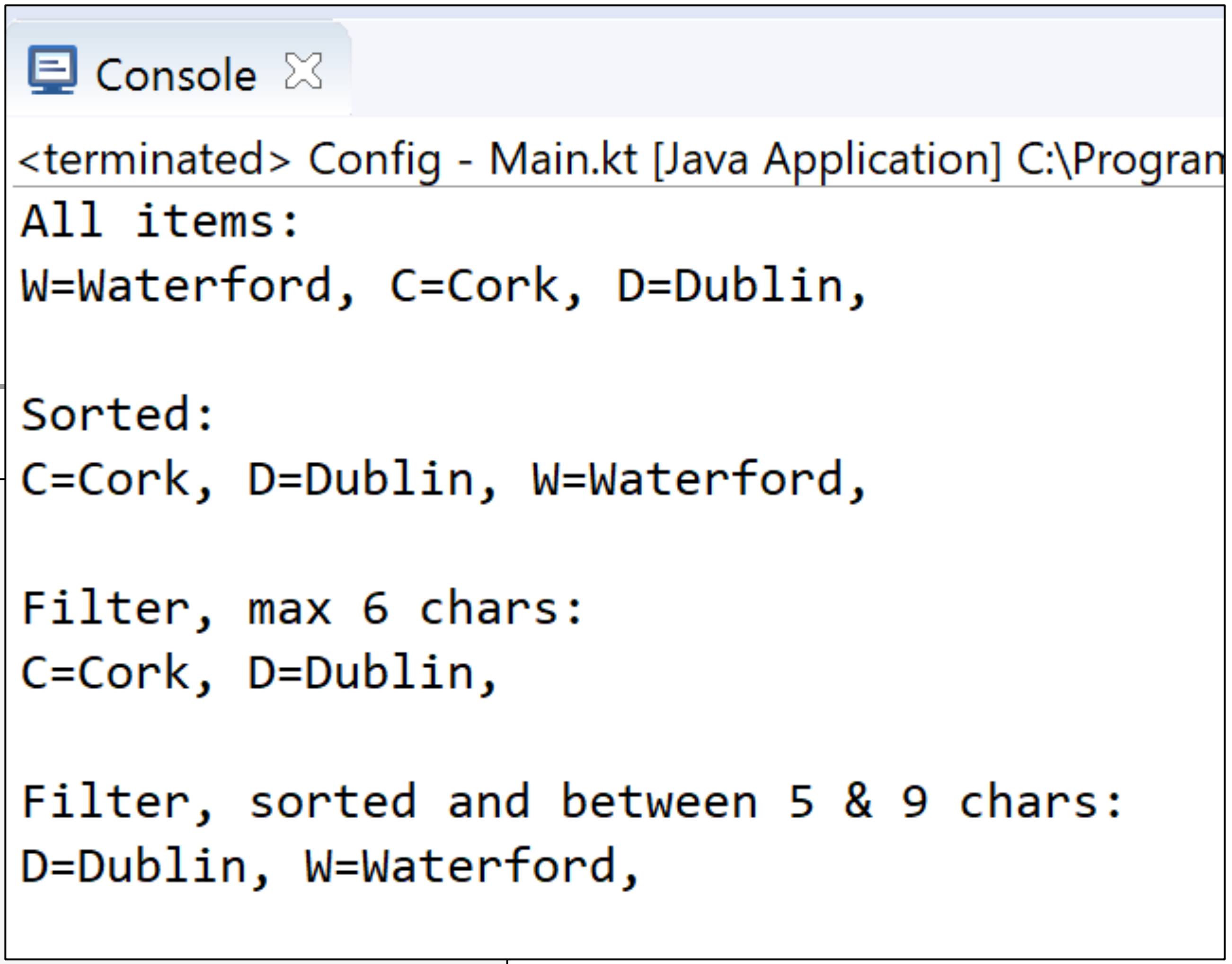
Console

<terminated> Config - Main.kt [Java Applicat

```
[-42, 17, 13, -9, 12]
[17, 13, 12]
-84 34 26 -18 24
-42 12
```

Maps and Lambdas

```
fun main(args: Array<String>) {

    val counties = mapOf(
            Pair("W","Waterford"),
            Pair("C","Cork"),
            Pair ("D","Dublin"))

    println("All items:");
    counties.forEach {print(it); print (", ")}

    println("\n\nSorted:");
    counties.toSortedMap()
            .forEach {print(it); print (", ")}

    println("\n\nFilter, max 6 chars:");
    counties.filter {it.value.length <= 6 }
            .forEach {print(it); print (", ")}

    println("\n\nFilter, sorted and between 5 & 9 chars:");
    counties.filterValues {it.length >= 5 && it.length <=9}
            .toSortedMap()
            .forEach {print(it); print (", ")}
}
```



```
<terminated> Config - Main.kt [Java Application] C:\Progran
All items:
W=Waterford, C=Cork, D=Dublin,

Sorted:
C=Cork, D=Dublin, W=Waterford,

Filter, max 6 chars:
C=Cork, D=Dublin,

Filter, sorted and between 5 & 9 chars:
D=Dublin, W=Waterford,
```