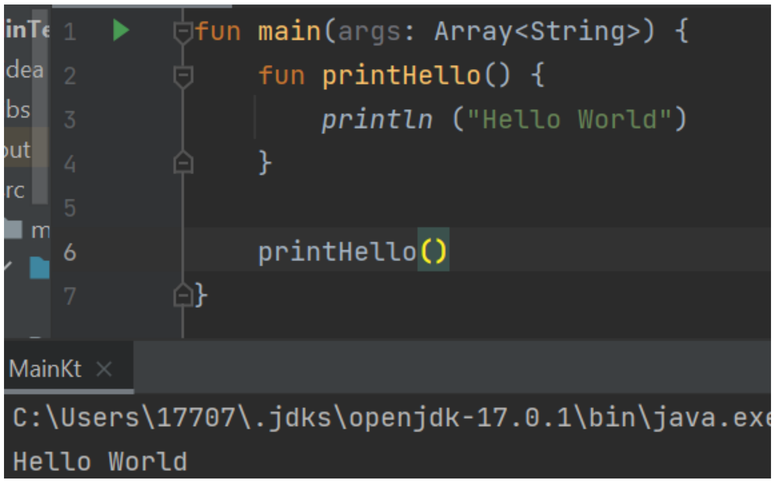
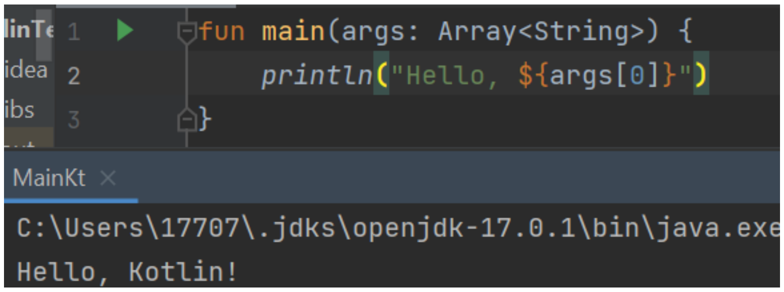


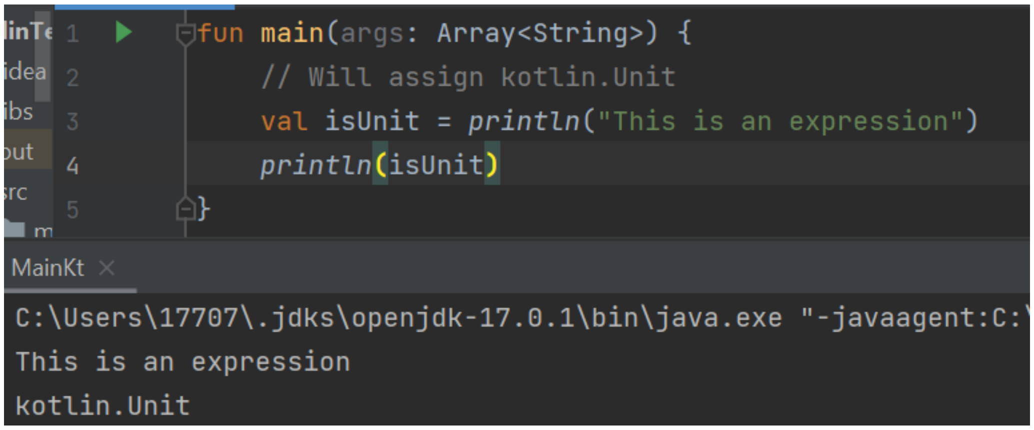
```
fun main(args: Array<String>) {
    fun printHello() {
        println ("Hello World")
    }

    printHello()
}
```

```
C:\Users\17707\.jdks\openjdk-17.0.1\bin\java.exe
Hello World
```



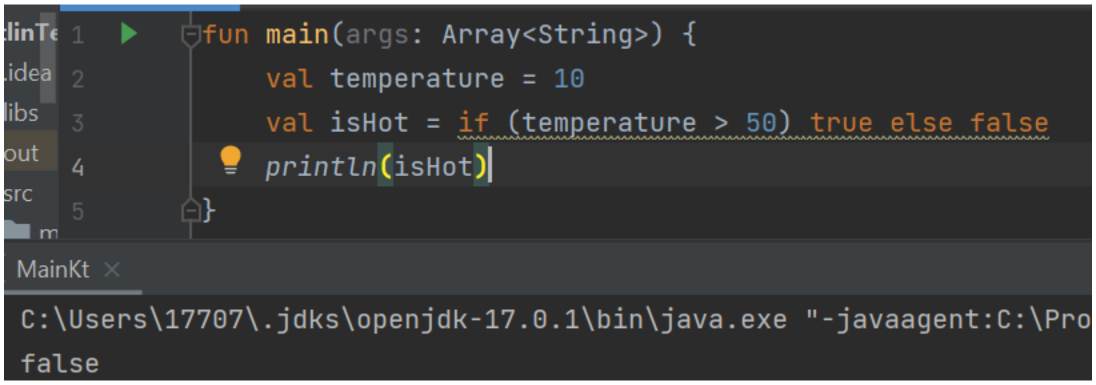
```
fun main(args: Array<String>) {
    println("Hello, ${args[0]}")
}
```

```
C:\Users\17707\.jdks\openjdk-17.0.1\bin\java.exe
Hello, Kotlin!
```



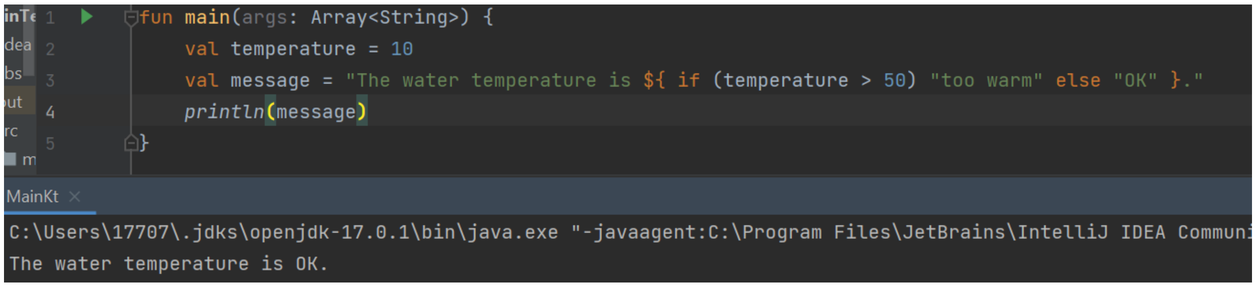
```
fun main(args: Array<String>) {
    // Will assign kotlin.Unit
    val isUnit = println("This is an expression")
    println(isUnit)
}
```

```
C:\Users\17707\.jdks\openjdk-17.0.1\bin\java.exe "-javaagent:C:
This is an expression
kotlin.Unit
```



```
fun main(args: Array<String>) {
    val temperature = 10
    val isHot = if (temperature > 50) true else false
    println(isHot)
}
```

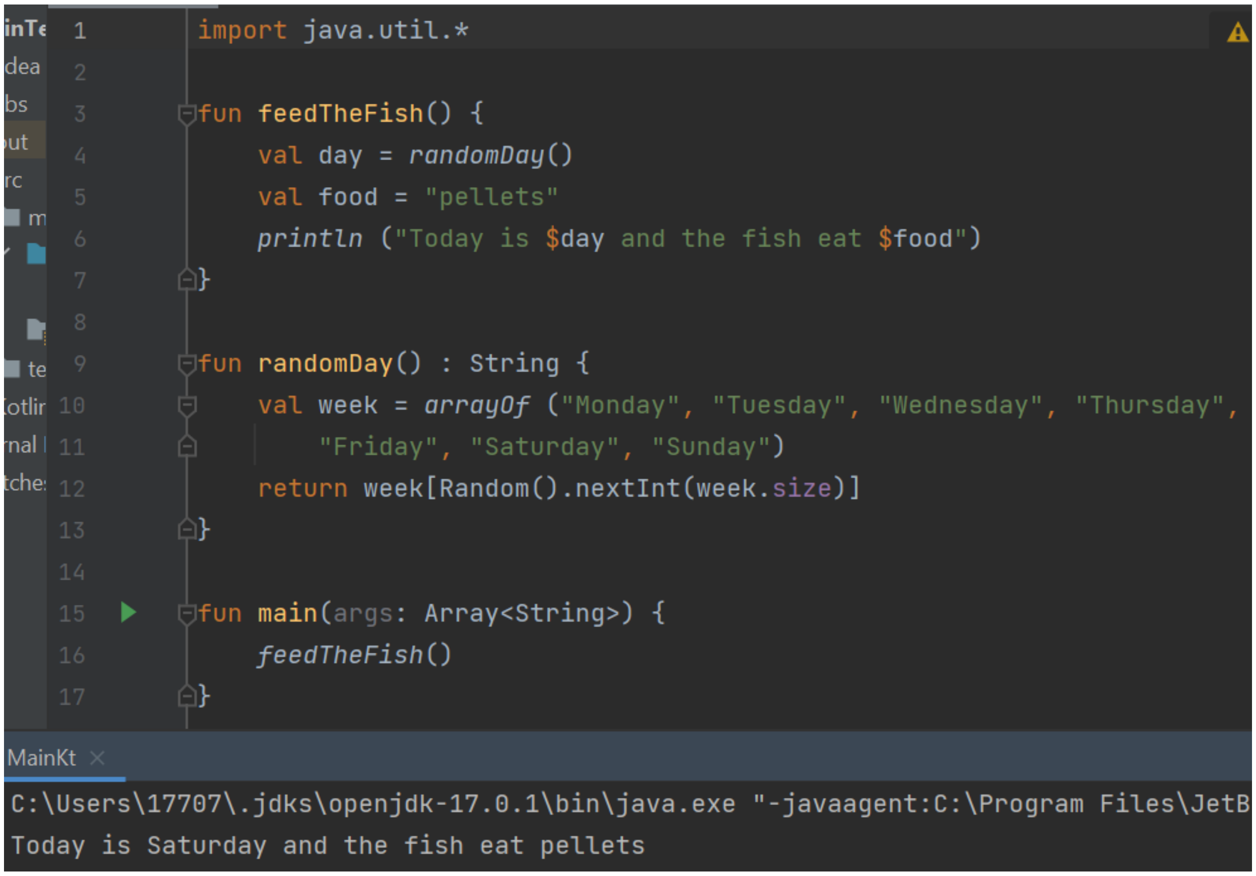
```
C:\Users\17707\.jdks\openjdk-17.0.1\bin\java.exe "-javaagent:C:\Pro
false
```



```
fun main(args: Array<String>) {
    val temperature = 10
    val message = "The water temperature is ${ if (temperature > 50) "too warm" else "OK" }."
    println(message)
}
```

MainKt

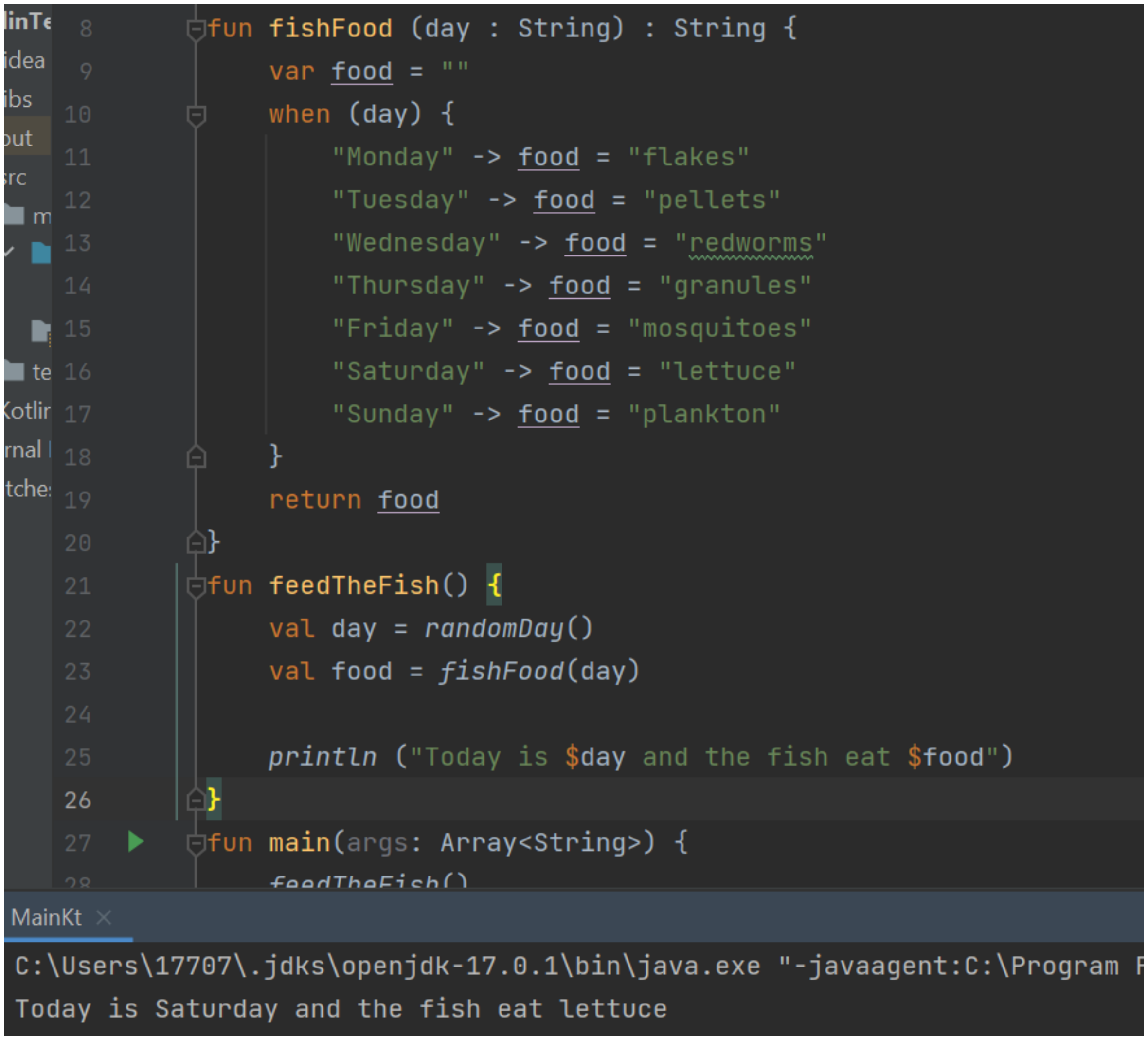
```
C:\Users\17707\.jdks\openjdk-17.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Communi
The water temperature is OK.
```



```
import java.util.*

fun feedTheFish() {
    val day = randomDay()
    val food = "pellets"
    println ("Today is $day and the fish eat $food")
}

fun randomDay() : String {
    val week = arrayOf ("Monday", "Tuesday", "Wednesday", "Thursday",
        "Friday", "Saturday", "Sunday")
    return week[Random().nextInt(week.size)]
}

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

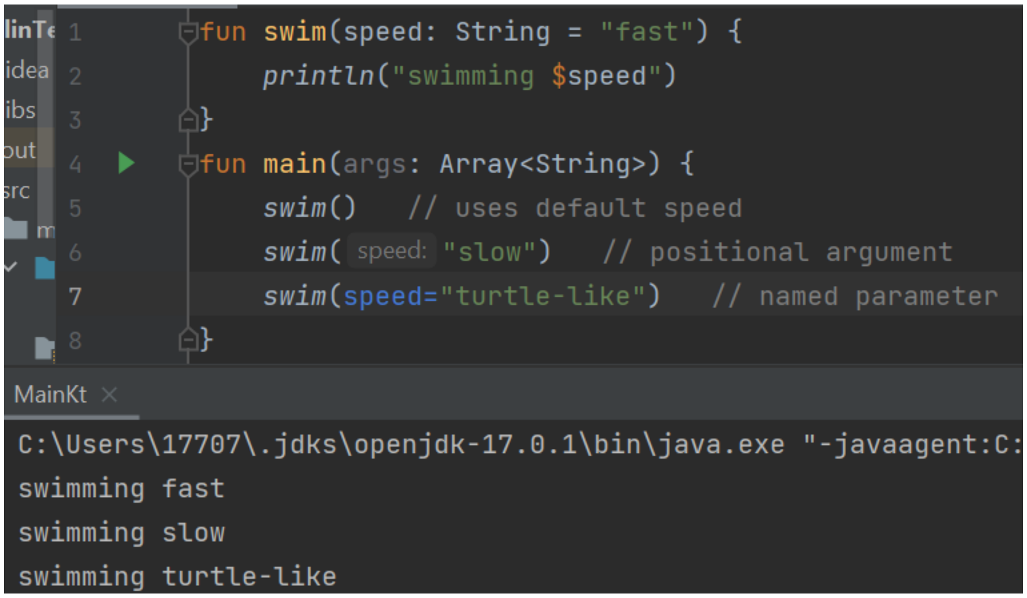
MainKt

```
C:\Users\17707\.jdks\openjdk-17.0.1\bin\java.exe "-javaagent:C:\Program Files\JetB
Today is Saturday and the fish eat pellets
```



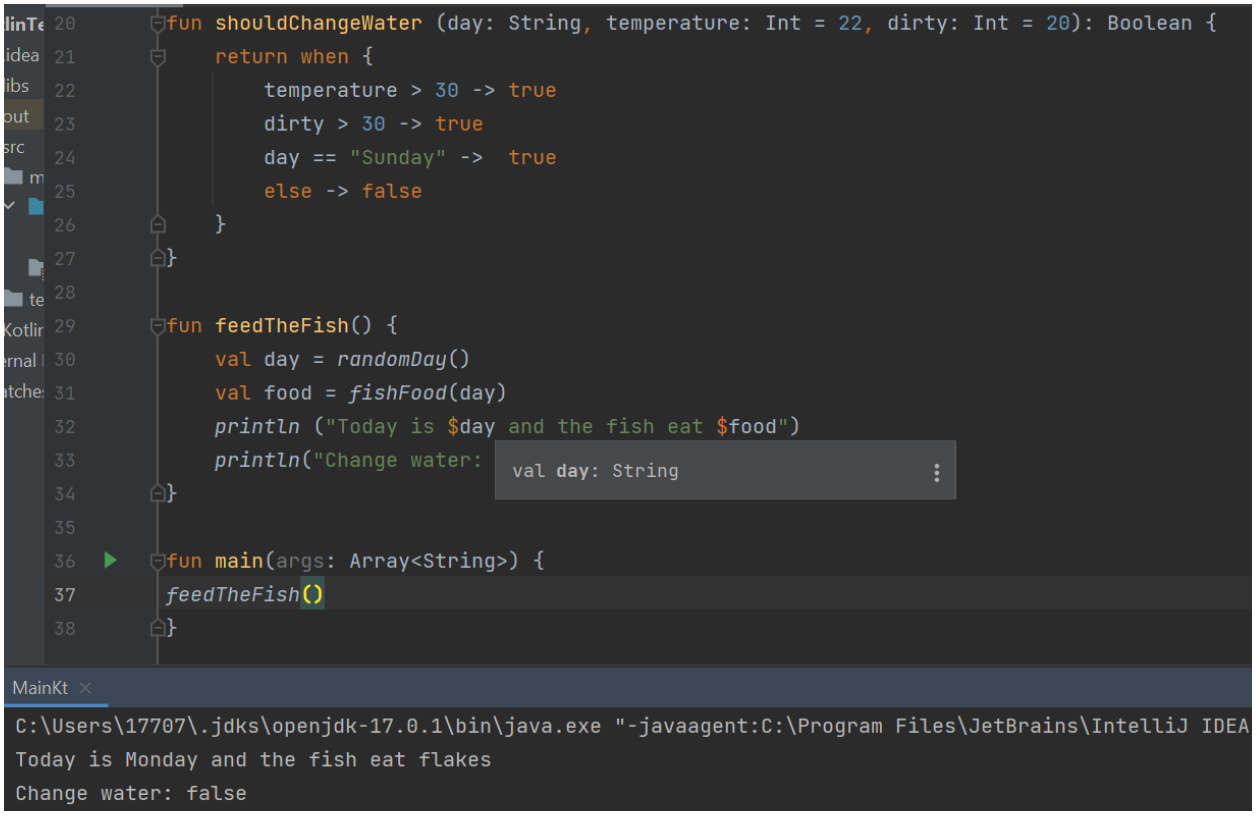
```kotlin
fun fishFood (day : String) : String {
    var food = ""
    when (day) {
        "Monday" -> food = "flakes"
        "Tuesday" -> food = "pellets"
        "Wednesday" -> food = "redworms"
        "Thursday" -> food = "granules"
        "Friday" -> food = "mosquitoes"
        "Saturday" -> food = "lettuce"
        "Sunday" -> food = "plankton"
    }
    return food
}
fun feedTheFish() {
    val day = randomDay()
    val food = fishFood(day)

    println ("Today is $day and the fish eat $food")
}
fun main(args: Array<String>) {
    feedTheFish()
```

```
C:\Users\17707\.jdks\openjdk-17.0.1\bin\java.exe "-javaagent:C:\Program F
Today is Saturday and the fish eat lettuce
```



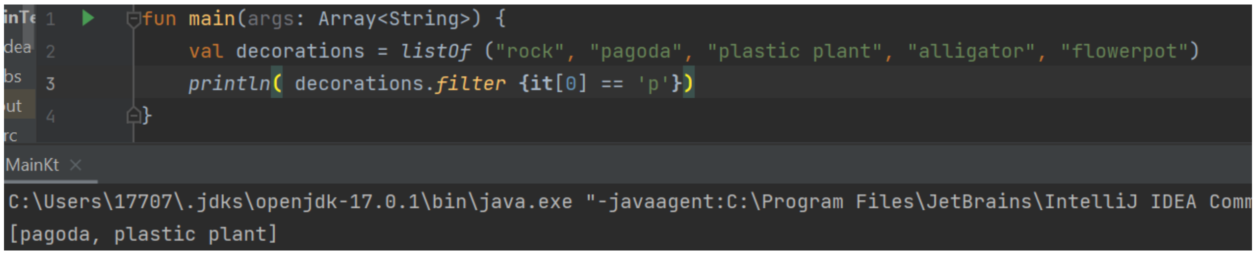
```kotlin
fun swim(speed: String = "fast") {
    println("swimming $speed")
}
fun main(args: Array<String>) {
    swim()   // uses default speed
    swim( speed: "slow")   // positional argument
    swim(speed="turtle-like")   // named parameter
}
```

```
C:\Users\17707\.jdks\openjdk-17.0.1\bin\java.exe "-javaagent:C:
swimming fast
swimming slow
swimming turtle-like
```



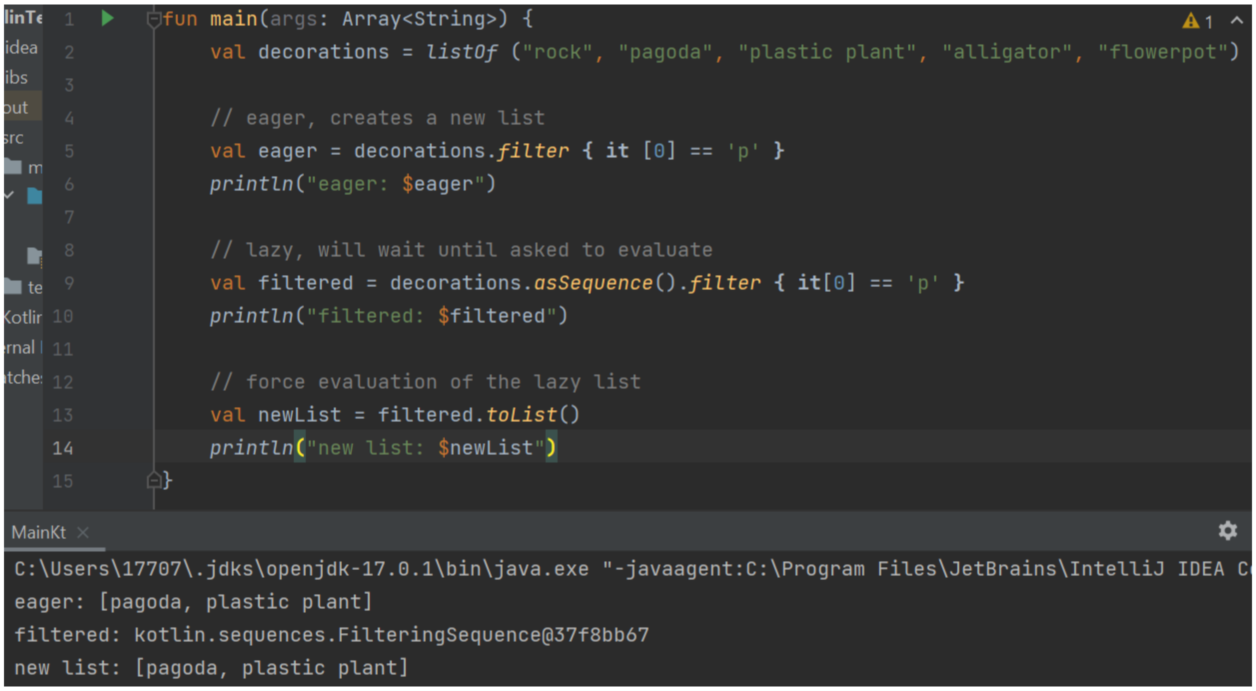
```
fun shouldChangeWater (day: String, temperature: Int = 22, dirty: Int = 20): Boolean {
    return when {
        temperature > 30 -> true
        dirty > 30 -> true
        day == "Sunday" ->  true
        else -> false
    }
}

fun feedTheFish() {
    val day = randomDay()
    val food = fishFood(day)
    println ("Today is $day and the fish eat $food")
    println("Change water:
}

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

```
C:\Users\17707\.jdks\openjdk-17.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA
Today is Monday and the fish eat flakes
Change water: false
```



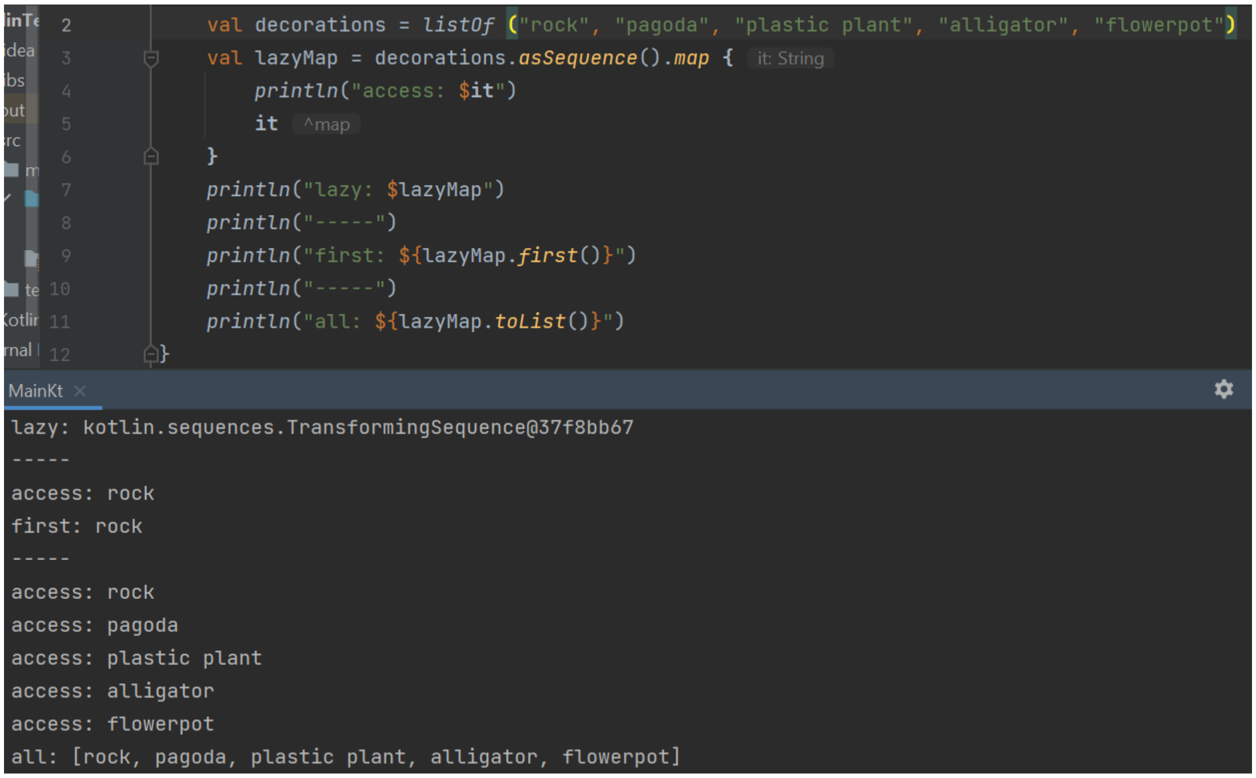
```
fun main(args: Array<String>) {
    val decorations = listOf ("rock", "pagoda", "plastic plant", "alligator", "flowerpot")
    println( decorations.filter {it[0] == 'p'})
}
```

```
C:\Users\17707\.jdks\openjdk-17.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Comm
[pagoda, plastic plant]
```



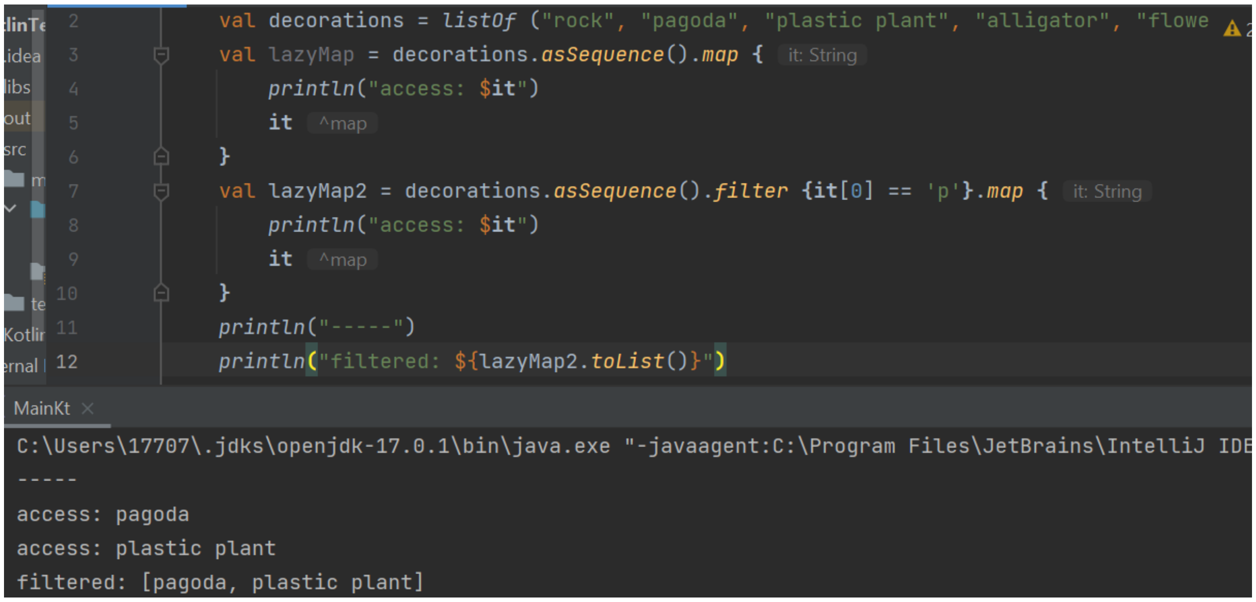
```kotlin
fun main(args: Array<String>) {
    val decorations = listOf ("rock", "pagoda", "plastic plant", "alligator", "flowerpot")

    // eager, creates a new list
    val eager = decorations.filter { it [0] == 'p' }
    println("eager: $eager")

    // lazy, will wait until asked to evaluate
    val filtered = decorations.asSequence().filter { it[0] == 'p' }
    println("filtered: $filtered")

    // force evaluation of the lazy list
    val newList = filtered.toList()
    println("new list: $newList")
}
```

```
C:\Users\17707\.jdks\openjdk-17.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA C
eager: [pagoda, plastic plant]
filtered: kotlin.sequences.FilteringSequence@37f8bb67
new list: [pagoda, plastic plant]
```



```kotlin
    val decorations = listOf ("rock", "pagoda", "plastic plant", "alligator", "flowerpot")
    val lazyMap = decorations.asSequence().map {
        println("access: $it")
        it
    }
    println("lazy: $lazyMap")
    println("-----")
    println("first: ${lazyMap.first()}")
    println("-----")
    println("all: ${lazyMap.toList()}")
}
```

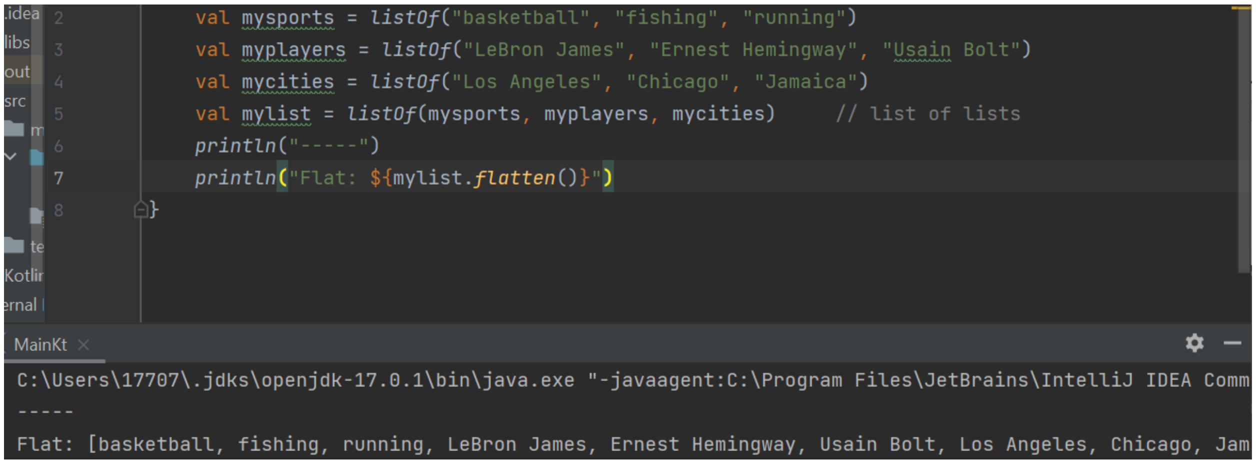
```
lazy: kotlin.sequences.TransformingSequence@37f8bb67
-----
access: rock
first: rock
-----
access: rock
access: pagoda
access: plastic plant
access: alligator
access: flowerpot
all: [rock, pagoda, plastic plant, alligator, flowerpot]
```



```
val decorations = listOf ("rock", "pagoda", "plastic plant", "alligator", "flowe
val lazyMap = decorations.asSequence().map {  it: String
    println("access: $it")
    it  ^map
}
val lazyMap2 = decorations.asSequence().filter {it[0] == 'p'}.map {  it: String
    println("access: $it")
    it  ^map
}
println("-----")
println("filtered: ${lazyMap2.toList()}")
```

MainKt

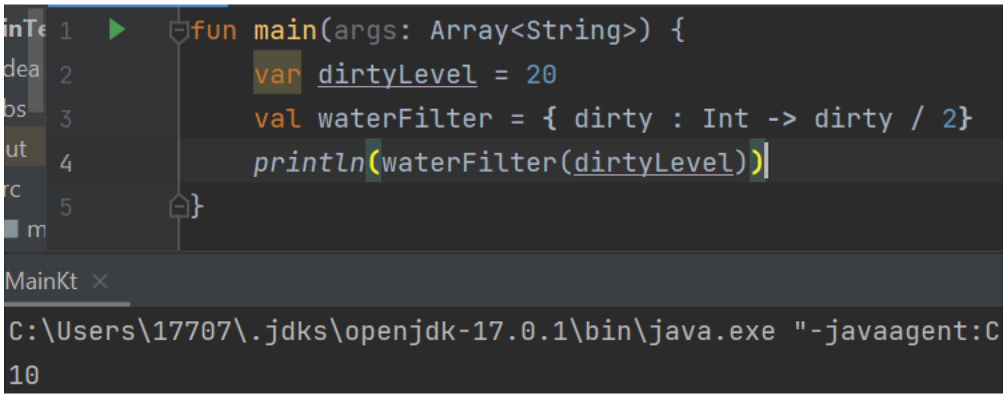
```
C:\Users\17707\.jdks\openjdk-17.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDE
-----
access: pagoda
access: plastic plant
filtered: [pagoda, plastic plant]
```



```
    val mysports = listOf("basketball", "fishing", "running")
    val myplayers = listOf("LeBron James", "Ernest Hemingway", "Usain Bolt")
    val mycities = listOf("Los Angeles", "Chicago", "Jamaica")
    val mylist = listOf(mysports, myplayers, mycities)     // list of lists
    println("-----")
    println("Flat: ${mylist.flatten()}")
}
```

MainKt

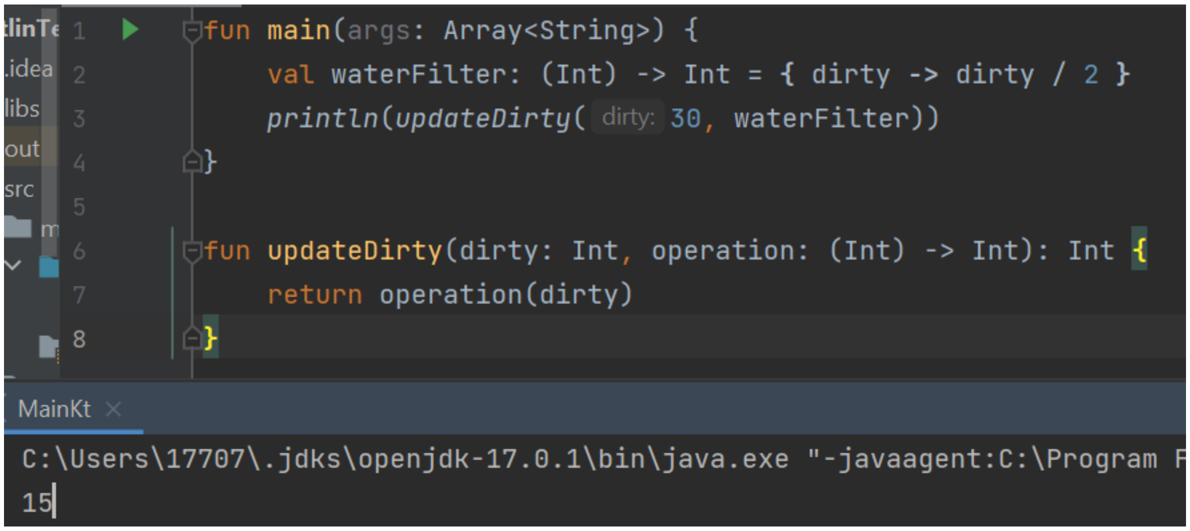
```
C:\Users\17707\.jdks\openjdk-17.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Comm
-----
Flat: [basketball, fishing, running, LeBron James, Ernest Hemingway, Usain Bolt, Los Angeles, Chicago, Jam
```



```
fun main(args: Array<String>) {
    var dirtyLevel = 20
    val waterFilter = { dirty : Int -> dirty / 2}
    println(waterFilter(dirtyLevel))
}
```

MainKt

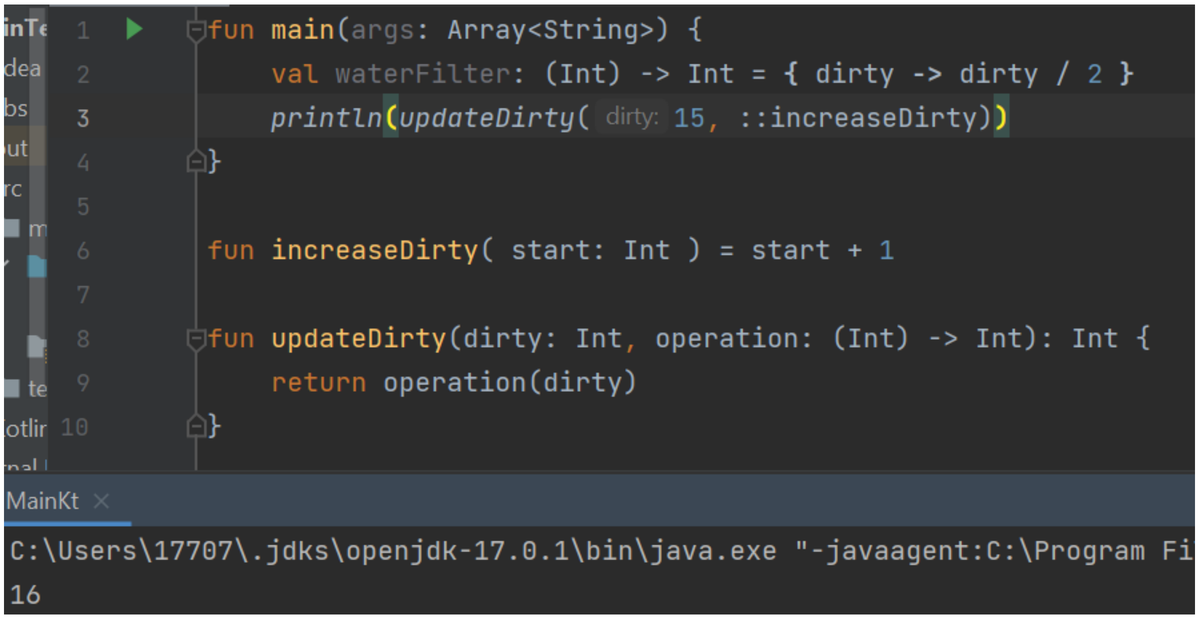
```
C:\Users\17707\.jdks\openjdk-17.0.1\bin\java.exe "-javaagent:C
10
```



```kotlin
fun main(args: Array<String>) {
    val waterFilter: (Int) -> Int = { dirty -> dirty / 2 }
    println(updateDirty( dirty: 30, waterFilter))
}

fun updateDirty(dirty: Int, operation: (Int) -> Int): Int {
    return operation(dirty)
}
```

MainKt

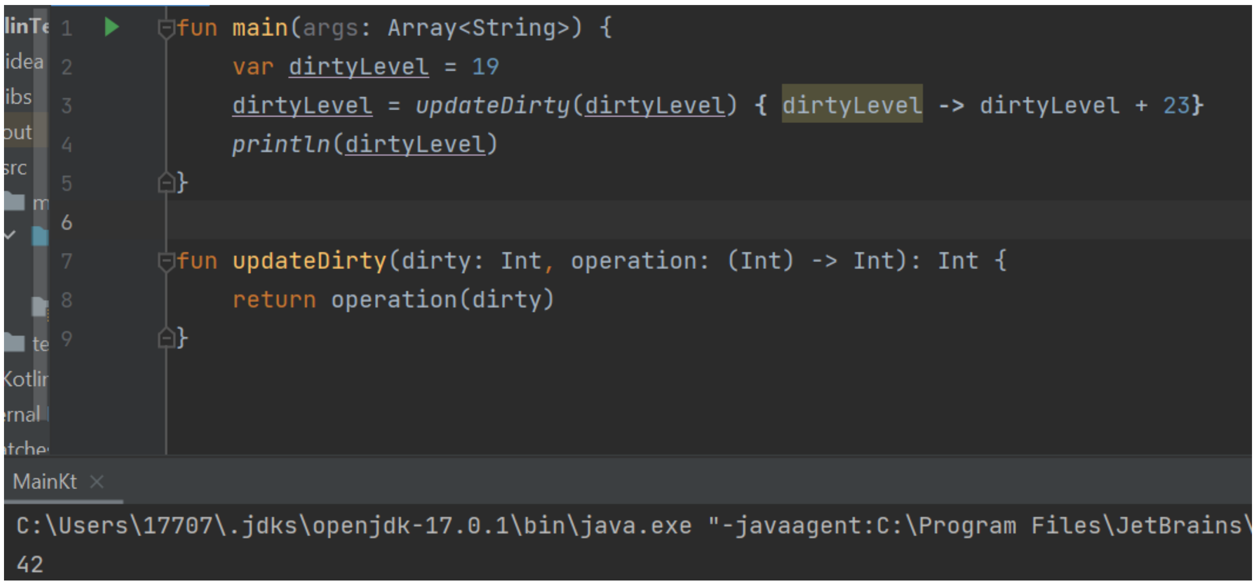
```
C:\Users\17707\.jdks\openjdk-17.0.1\bin\java.exe "-javaagent:C:\Program F
15
```



```kotlin
fun main(args: Array<String>) {
    val waterFilter: (Int) -> Int = { dirty -> dirty / 2 }
    println(updateDirty( dirty: 15, ::increaseDirty))
}

fun increaseDirty( start: Int ) = start + 1

fun updateDirty(dirty: Int, operation: (Int) -> Int): Int {
    return operation(dirty)
}
```

MainKt

```
C:\Users\17707\.jdks\openjdk-17.0.1\bin\java.exe "-javaagent:C:\Program Fi
16
```



```kotlin
fun main(args: Array<String>) {
    var dirtyLevel = 19
    dirtyLevel = updateDirty(dirtyLevel) { dirtyLevel -> dirtyLevel + 23}
    println(dirtyLevel)
}

fun updateDirty(dirty: Int, operation: (Int) -> Int): Int {
    return operation(dirty)
}
```

```
C:\Users\17707\.jdks\openjdk-17.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\
42
```