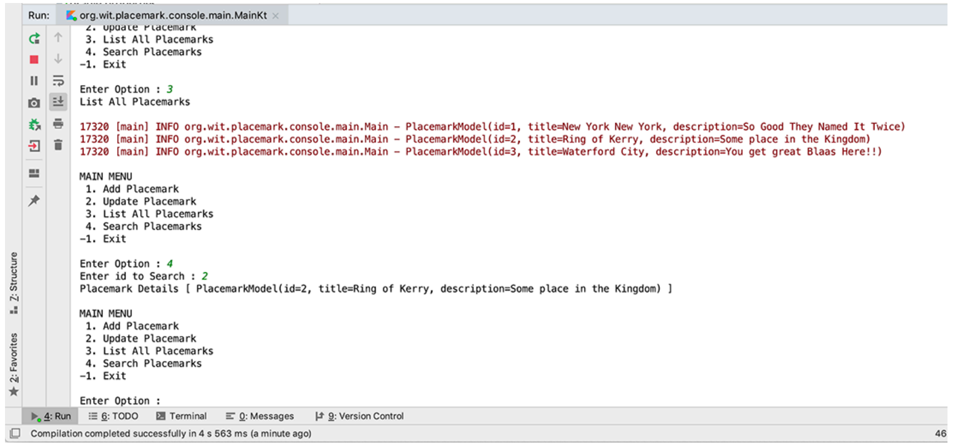
Mobile Application Development

Produced by David Drohan (ddrohan@wit.ie)

Department of Computing & Mathematics
Waterford Institute of Technology
http://www.wit.ie

Placemark-Console Version 2.0



```
 2. Update Placemark
 3. List All Placemarks
 4. Search Placemarks
-1. Exit

Enter Option : 3
List All Placemarks

17320 [main] INFO org.wit.placemark.console.main.Main - PlacemarkModel(id=1, title=New York New York, description=So Good They Named It Twice)
17320 [main] INFO org.wit.placemark.console.main.Main - PlacemarkModel(id=2, title=Ring of Kerry, description=Some place in the Kingdom)
17320 [main] INFO org.wit.placemark.console.main.Main - PlacemarkModel(id=3, title=Waterford City, description=You get great Blaas Here!!)

MAIN MENU
 1. Add Placemark
 2. Update Placemark
 3. List All Placemarks
 4. Search Placemarks
-1. Exit

Enter Option : 4
Enter id to Search : 2
Placemark Details [ PlacemarkModel(id=2, title=Ring of Kerry, description=Some place in the Kingdom) ]

MAIN MENU
 1. Add Placemark
 2. Update Placemark
 3. List All Placemarks
 4. Search Placemarks
-1. Exit

Enter Option :
```

Compilation completed successfully in 4 s 563 ms (a minute ago) 46

Features Covered (from Part 1)

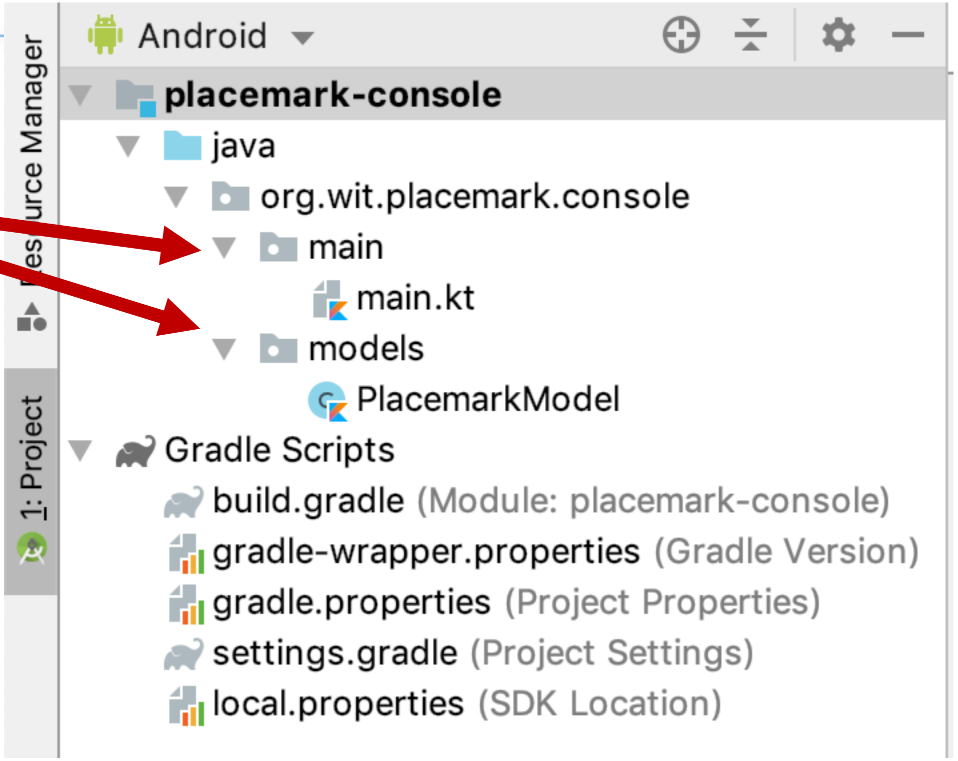
- ❑ Basic Types
- ❑ Local Variables (val & var)
- ❑ Functions
- ❑ Control Flow (if, when, for, while)
- ❑ Strings & String Templates
- ❑ Ranges (and the *in* operator)
- ❑ Type Checks & Casts
- ❑ Null Safety
- ❑ Comments

Features Covered (from Part 2)

- ❑Writing Classes (properties and fields)
- ❑Data Classes (just for data)
- ❑Collections: Arrays and Collections
- ❑Collections: *in* operator and lambdas
- ❑Arguments (default and named)

Project Structure

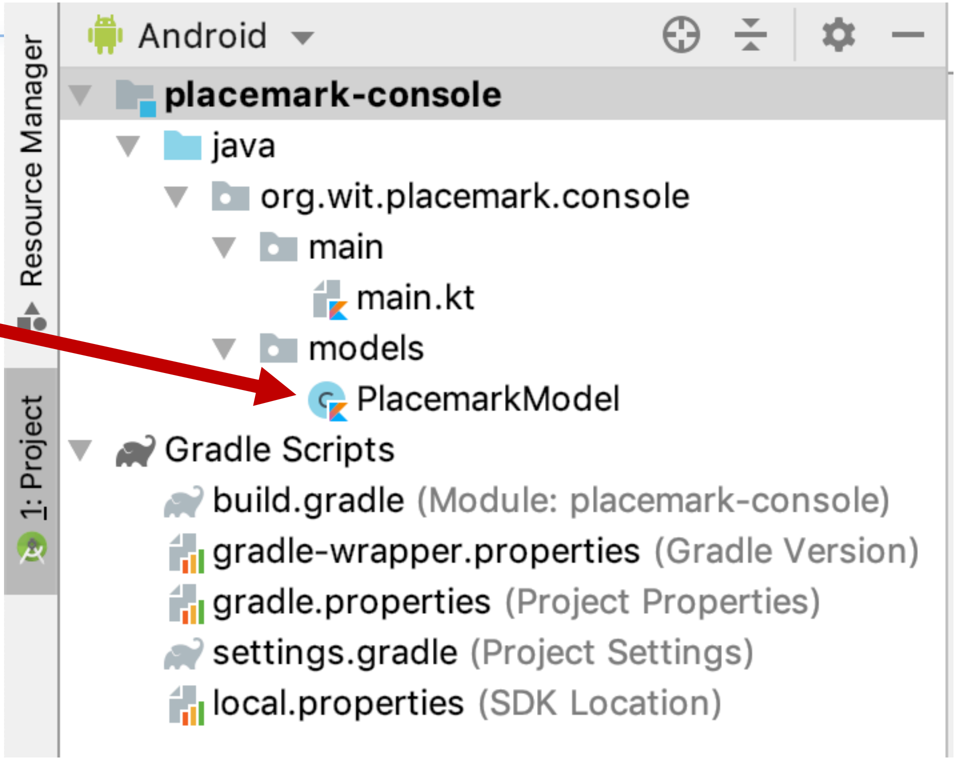
- ❑ Still Fairly basic
  - Now 2 packages



```
placemark-console
  java
    org.wit.placemark.console
      main
        main.kt
      models
        PlacemarkModel
Gradle Scripts
  build.gradle (Module: placemark-console)
  gradle-wrapper.properties (Gradle Version)
  gradle.properties (Project Properties)
  settings.gradle (Project Settings)
  local.properties (SDK Location)
```

Project Structure

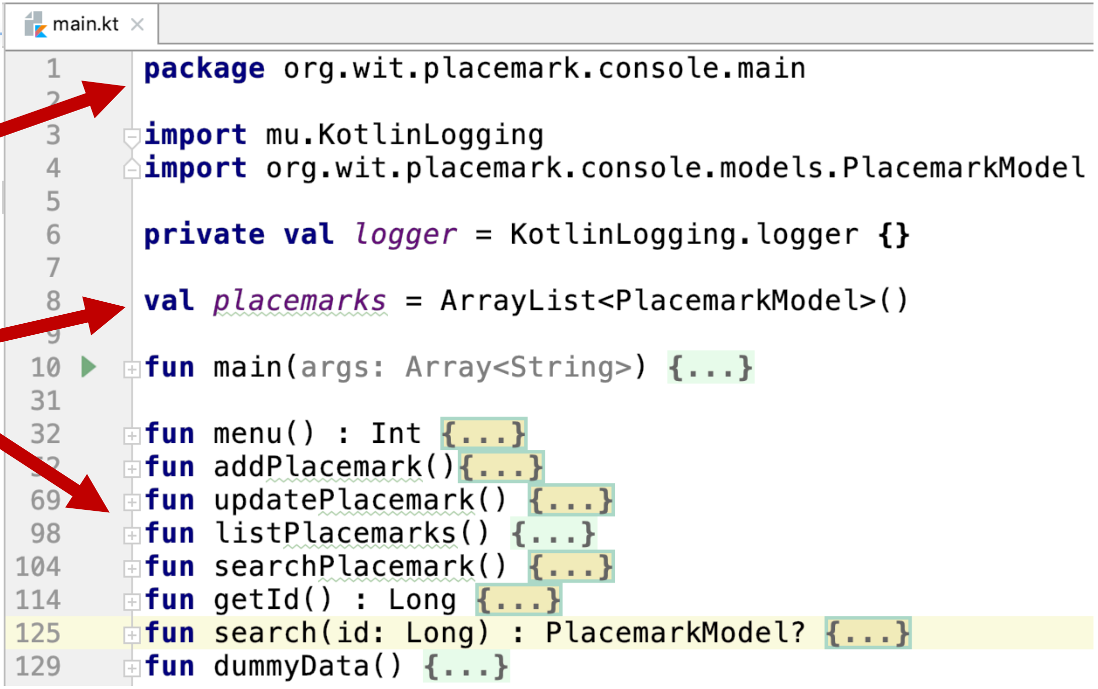
- ❑ Still Fairly basic
  - Now 2 packages
  - Model Introduced



```
placemark-console
  java
    org.wit.placemark.console
      main
        main.kt
      models
        PlacemarkModel
Gradle Scripts
  build.gradle (Module: placemark-console)
  gradle-wrapper.properties (Gradle Version)
  gradle.properties (Project Properties)
  settings.gradle (Project Settings)
  local.properties (SDK Location)
```

main.kt

- Some basic CRUD
  - Package **main**
  - 1 Kotlin source FILE
    - More Features

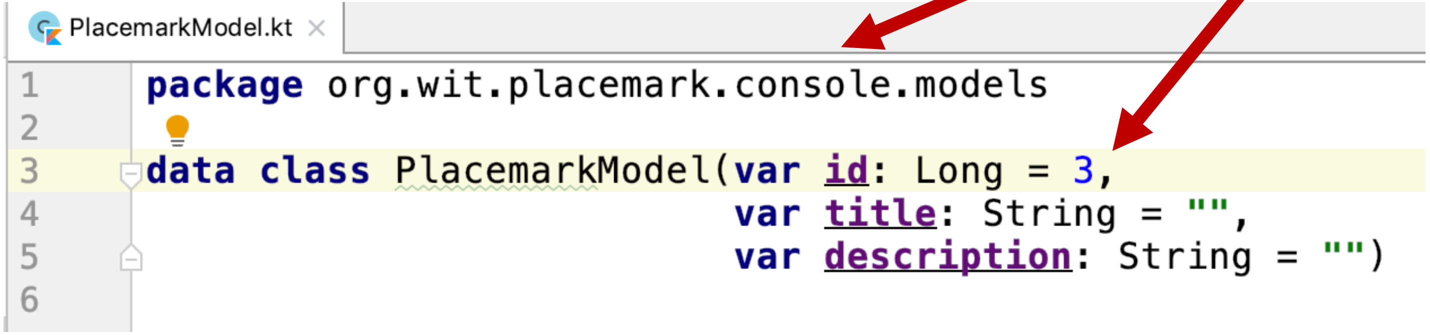


main.kt

```
package org.wit.placemark.console.main

import mu.KotlinLogging
import org.wit.placemark.console.models.PlacemarkModel

private val logger = KotlinLogging.logger {}

val placemarks = ArrayList<PlacemarkModel>()

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

fun menu() : Int {...}
fun addPlacemark(){...}
fun updatePlacemark() {...}
fun listPlacemarks() {...}
fun searchPlacemark() {...}
fun getId() : Long {...}
fun search(id: Long) : PlacemarkModel? {...}
fun dummyData() {...}
```

PlacemarkModel.kt

- Basic **`data`** class
  - Package **`models`**
  - Default Arguments



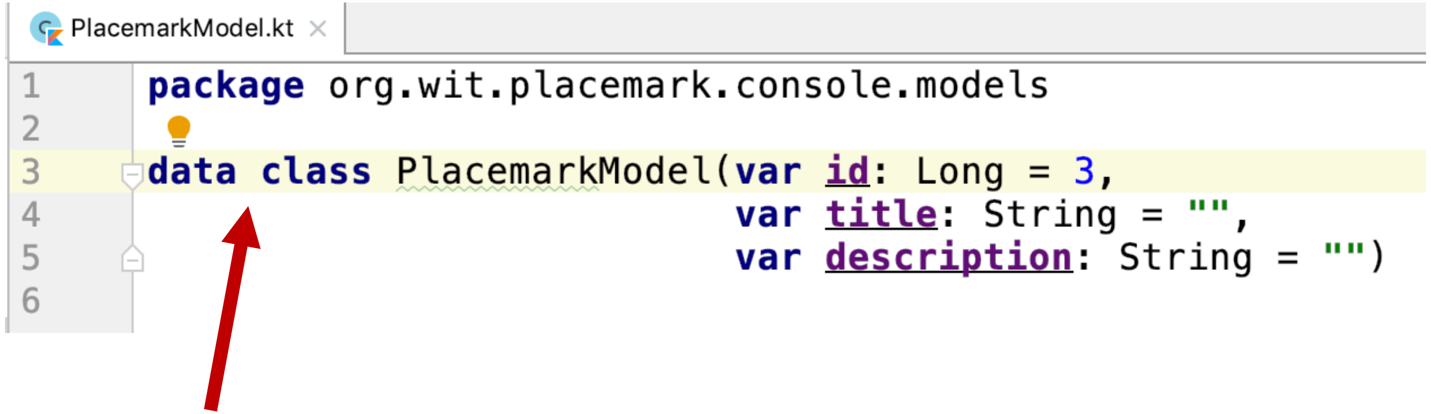
PlacemarkModel.kt

```kotlin
package org.wit.placemark.console.models

data class PlacemarkModel(var id: Long = 3,
                          var title: String = "",
                          var description: String = "")
```

Data Classes

Placemark-Console Version 2.0

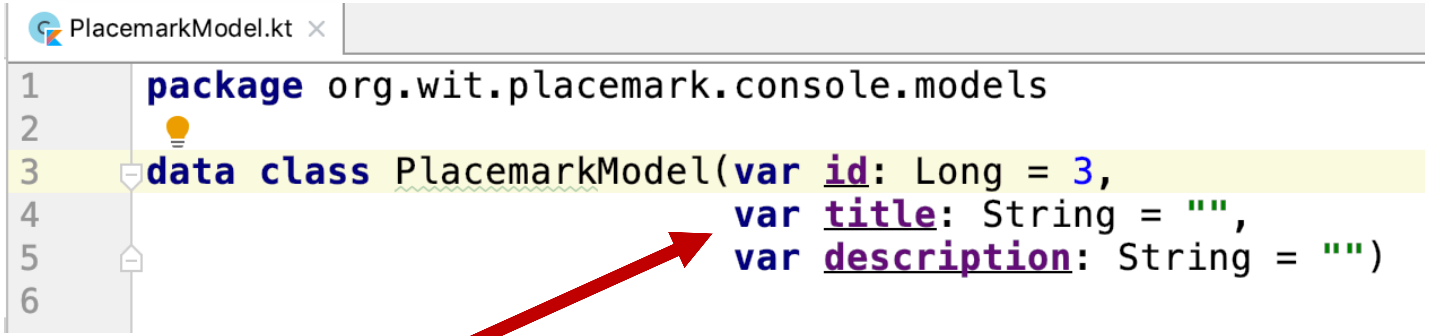
Data Classes & Arguments in Placemark



```kotlin
package org.wit.placemark.console.models

data class PlacemarkModel(var id: Long = 3,
                          var title: String = "",
                          var description: String = "")
```

- note **`data class`** *declaration*
  - ***optimized*** *for storing only data*
- We will use this class for modelling a **Placemark** object

Data Classes & Arguments in Placemark



PlacemarkModel.kt

```
package org.wit.placemark.console.models

data class PlacemarkModel(var id: Long = 3,
                          var title: String = "",
                          var description: String = "")
```

- 3 arguments, named, with **`default`** values
- Objects created like so

```
var aPlacemark = PlacemarkModel()
```

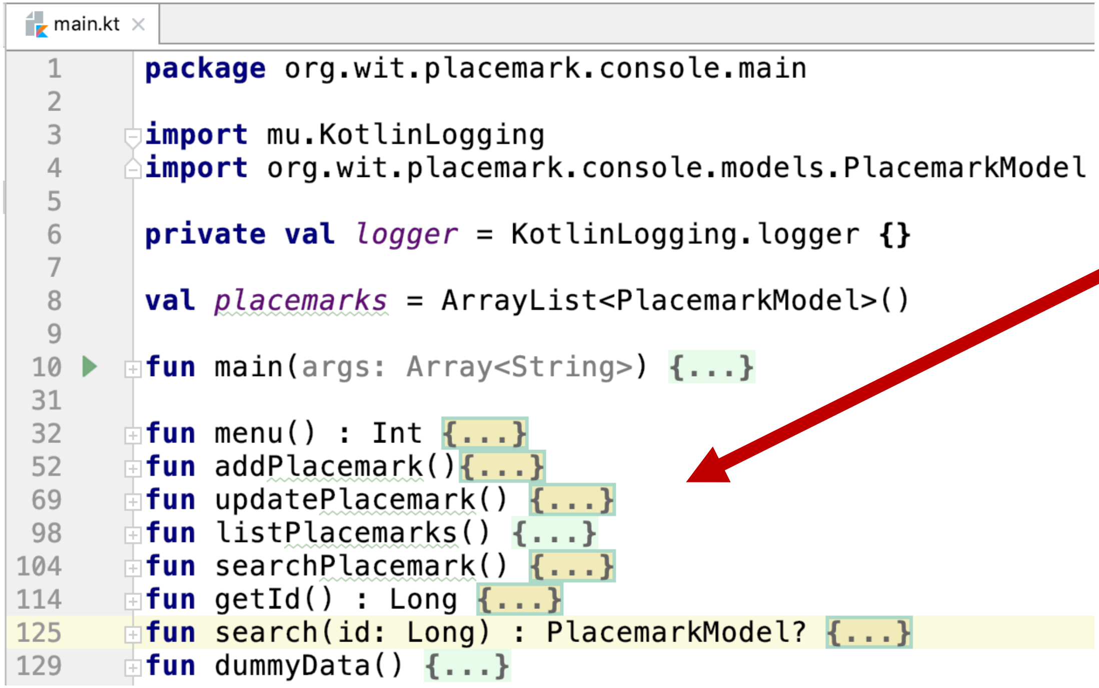
  - and

```
val placemarks = ArrayList<PlacemarkModel>()
```

Functions & Control Flow

Placemark-Console Version 2.0

Functions in Placemark



main.kt

```
package org.wit.placemark.console.main

import mu.KotlinLogging
import org.wit.placemark.console.models.PlacemarkModel

private val logger = KotlinLogging.logger {}

val placemarks = ArrayList<PlacemarkModel>()

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

fun menu() : Int {...}
fun addPlacemark(){...}
fun updatePlacemark() {...}
fun listPlacemarks() {...}
fun searchPlacemark() {...}
fun getId() : Long {...}
fun search(id: Long) : PlacemarkModel? {...}
fun dummyData() {...}
```

- ❑ more functions in our app to implement CRUD features
  - we'll look at a few here

Function `addPlacemark( )` (CREATE)

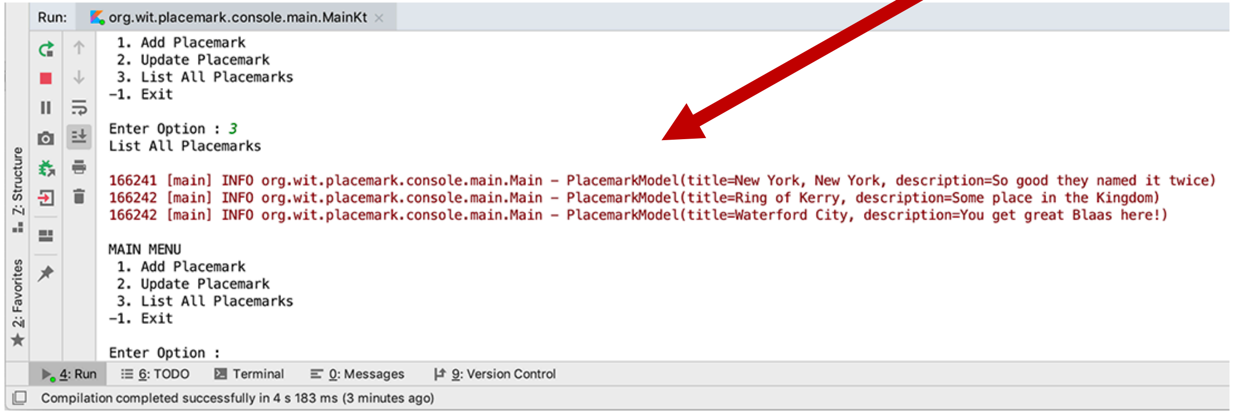
```
fun addPlacemark(){
    var aPlacemark = PlacemarkModel()
    println("Add Placemark")
    println()
    print("Enter a Title : ")
    aPlacemark.title = readLine()!!
    print("Enter a Description : ")
    aPlacemark.description = readLine()!!

    if (aPlacemark.title.isNotEmpty() &&
        aPlacemark.description.isNotEmpty()) {
            aPlacemark.id++
            placemarks.add(aPlacemark.copy())
            logger.info("Placemark Added : [ $aPlacemark ]")
    }
    else
        logger.info("Placemark Not Added")
}
```

- ❑ getting Placemark info
- ❑ adding to our **`placemarks`** collection (more later)

Function `listPlacemarks( )` (READ)

```kotlin
fun listPlacemarks() {
    println("List All Placemarks")
    println()
    placemarks.forEach { logger.info("${it}") }
    println()
}
```

- ❑ displaying Placemarks
- ❑ result



```
 1. Add Placemark
 2. Update Placemark
 3. List All Placemarks
-1. Exit

Enter Option : 3
List All Placemarks

166241 [main] INFO org.wit.placemark.console.main.Main - PlacemarkModel(title=New York, New York, description=So good they named it twice)
166242 [main] INFO org.wit.placemark.console.main.Main - PlacemarkModel(title=Ring of Kerry, description=Some place in the Kingdom)
166242 [main] INFO org.wit.placemark.console.main.Main - PlacemarkModel(title=Waterford City, description=You get great Blaas here!)

MAIN MENU
 1. Add Placemark
 2. Update Placemark
 3. List All Placemarks
-1. Exit

Enter Option :
```

Compilation completed successfully in 4 s 183 ms (3 minutes ago)

Function **`updatePlacemark( )`** (UPDATE)

```kotlin
fun updatePlacemark() {
    println("Update Placemark")
    println()
    listPlacemarks()
    var searchId = getId()
    val aPlacemark = search(searchId)
    var tempTitle : String?
    var tempDescription : String?

    if(aPlacemark != null) {
        print("Enter a new Title for [ " + aPlacemark.title + " ] : ")
        tempTitle = readLine()!!
        print("Enter a new Description for [ " + aPlacemark.description + " ] : ")
        tempDescription = readLine()!!

        if (!tempTitle.isNullOrEmpty() && !tempDescription.isNullOrEmpty()) {
            aPlacemark.title = tempTitle
            aPlacemark.description = tempDescription
            println(
                "You updated [ " + aPlacemark.title + " ] for title " +
                        "and [ " + aPlacemark.description + " ] for description")
            logger.info("Placemark Updated : [ $aPlacemark ]")
        }
        else
            logger.info("Placemark Not Updated")
    }
    else
        println("Placemark Not Updated...")
}
```

- ❑ finding a Placemark
- ❑ updating
  - note we don't reference the collection – all done via reference **`aPlacemark`**

Helper Functions

```
fun getId() : Long {
    var strId : String? // String to hold user input
    var searchId : Long // Long to hold converted id
    print("Enter id to Search/Update : ")
    strId = readLine()!!
    searchId = if (strId.toIntOrNull() != null && !strId.isEmpty())
        strId.toLong()
    else
        -9
    return searchId
}

fun search(id: Long) : PlacemarkModel? {
    var foundPlacemark: PlacemarkModel? = placemarks.find { p -> p.id == id }
        return foundPlacemark
}
```

- ❑ getting a valid 'id'
- ❑ Searching for a specific Placemark
  - note we use the collections method `find`

Arrays & Collections

Placemark-Console Version 2.0

Arrays & Collections in Placemark

- ❑ **`val`** variable **`placemarks`** is declared as a **`ArrayList`** of **`PlacemarkModel`** objects

```
val placemarks = ArrayList<PlacemarkModel>()
```

- ❑ we can add to this collection like so

```
placemarks.add(PlacemarkModel(1, "New York New York", "So Good They Named It Twice"))
```

- ❑ or

```
placemarks.add(aPlacemark.copy())
```

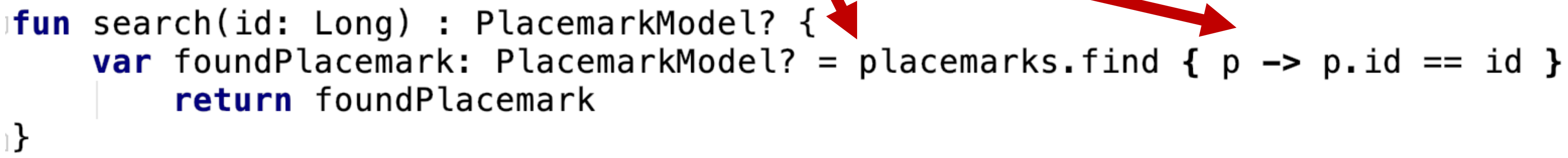
Note the use of **`.copy()`** (ensures a copy is stored, not actual object to avoid unexpected changes to collection data)

Lambdas

Placemark-Console Version 2.0

Lambdas in Placemark

- Here the Collections **`find`** function is a ‘Higher Order’ function which means we can assign it as a parameter and pass anonymous functions to it, and use lambda expressions



```
fun search(id: Long) : PlacemarkModel? {
    var foundPlacemark: PlacemarkModel? = placemarks.find { p -> p.id == id }
        return foundPlacemark
}
```

References

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

Thanks.