

Mobile Application Development

Produced
by

David Drohan (ddrohan@wit.ie)

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



Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE



Placemark-Console

Version 4.0

```
Run: org.wit.placemark.console.main.MainKt (1)
"/Applications/Android Studio.app/Contents/jre/jdk/Contents/Home/bin/java" ...
65 [main] INFO org.wit.placemark.console.controllers.PlacemarkController - Launching Placemark Console App
Placemark Kotlin App Version 4.0
MAIN MENU
1. Add Placemark
2. Update Placemark
3. List All Placemarks
4. Search Placemarks
5. Delete Placemark
-1. Exit

Enter Option : 5
List All Placemarks
6352 [main] INFO org.wit.placemark.console.models.PlacemarkJSONStore - PlacemarkModel(id=5289129853945747330, title=New York New York, description=So Good They Named It Twice)

6352 [main] INFO org.wit.placemark.console.models.PlacemarkJSONStore - PlacemarkModel(id=9125089131636741554, title=Rings of Kerry, description=Some more places in the Kingdom)

6352 [main] INFO org.wit.placemark.console.models.PlacemarkJSONStore - PlacemarkModel(id=3677893589403729130, title=Blaa Land, description=just another description)
Enter id to Search/Update/Delete : 6352 [main] INFO org.wit.placemark.console.models.PlacemarkJSONStore - PlacemarkModel(id=7048570743572904841, title=sdfsdfs, description=sdfsdfs)
```

Run | TODO | Build | Terminal | Version Control

Gradle build finished in 220 ms (a minute ago) 20:1 LF : U



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



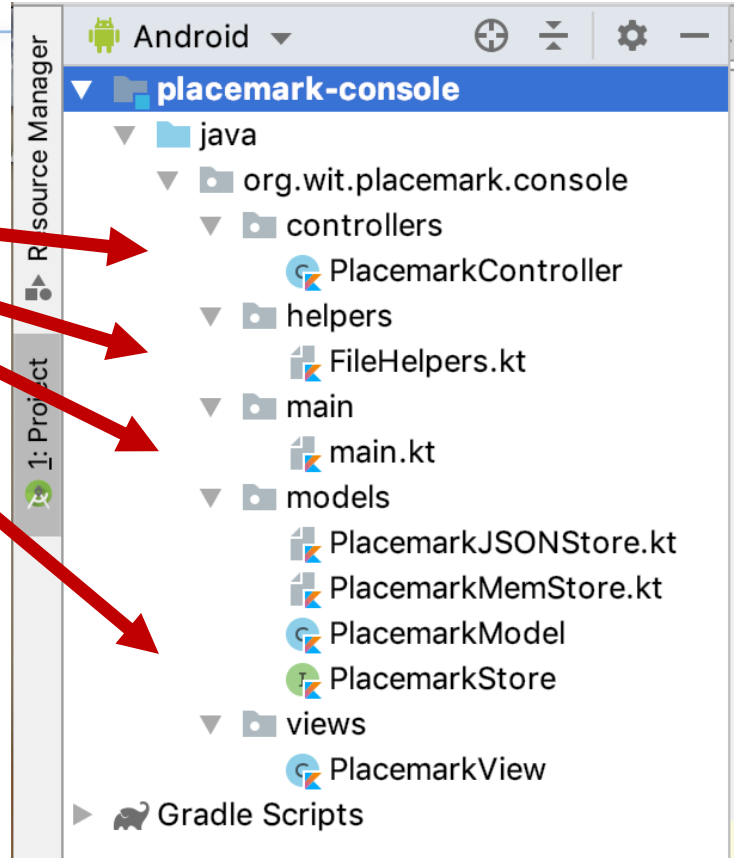
Features Covered (from Part 2)

- ❑ 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

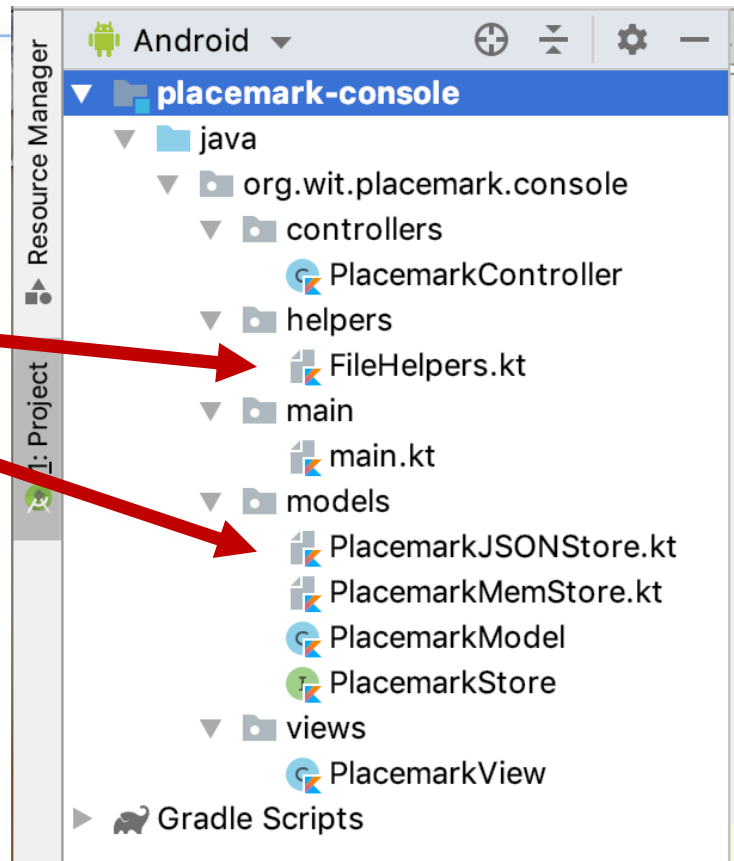
- ❑ Again, more complex
 - Multiple packages



Project Structure

□ Again, more complex

- Multiple packages
- Helper File & JSON Support



main.kt

❑ Codebase in main file substantially Reduced

- **fun main**

- Single line of code




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

Classes & Interfaces

Placemark-Console Version 4.0

Interface **PlacemarkStore**

```
interface PlacemarkStore {  
    fun findAll(): List<PlacemarkModel>  
    fun findOne(id: Long): PlacemarkModel?  
    fun create(placemark: PlacemarkModel)  
    fun update(placemark: PlacemarkModel)  
    fun delete(placemark: PlacemarkModel)  
}
```



- ❑ Additional **delete** function to allow for full CRUD support in implemented classes

Class **PlacemarkJSONStore**

```
class PlacemarkJSONStore : PlacemarkStore {  
    var placemarks = mutableListOf<PlacemarkModel>()  
  
    init {...}  
    override fun findAll(): MutableList<PlacemarkModel> {...}  
    override fun findOne(id: Long) : PlacemarkModel? {...}  
    override fun create(placemark: PlacemarkModel) {...}  
    override fun update(placemark: PlacemarkModel) {...}  
    override fun delete(placemark: PlacemarkModel) {...}  
    internal fun logAll() {...}  
    private fun serialize() {...}  
    private fun deserialize() {...}  
}
```


- ❑ Implements **PlacemarkStore** – allows for Placemark objects to be stored in JSON file using **serialize()** and **deserialize()**

Class PlacemarkJSONStore

```
class PlacemarkJSONStore : PlacemarkStore {  
  
    var placemarks = mutableListOf<PlacemarkModel>()  
  
    init {...}  
    override fun findAll(): MutableList<PlacemarkModel> {...}  
    override fun findOne(id: Long) : PlacemarkModel? {...}  
    override fun create(placemark: PlacemarkModel) {...}  
    override fun update(placemark: PlacemarkModel) {...}  
    override fun delete(placemark: PlacemarkModel) {...}  
    internal fun logAll() {...}  
    private fun serialize() {  
        val jsonString = GsonBuilder.toJson(placemarks, listType)  
        write(JSON_FILE, jsonString)  
    }  
    private fun deserialize() {  
        val jsonString = read(JSON_FILE)  
        placemarks = Gson().fromJson(jsonString, listType)  
    }  
}
```

FileHelpers.kt

```
fun write( fileName: String, data: String) {  
    val file = File(fileName)  
    try {  
        val outputStreamWriter = OutputStreamWriter(FileOutputStream(file))  
        outputStreamWriter.write(data)  
        outputStreamWriter.close()  
    } catch (e: Exception) {  
        logger.error { "Cannot read file: " + e.toString() }  
    }  
}
```



- ❑ Main purpose to write out a stream of data to a file
- ❑ Data stored in JSON format

FileHelpers.kt

```
fun read(fileName: String): String {
    val file = File(fileName)
    var str = ""
    try {
        val inputStreamReader = InputStreamReader(FileInputStream(file))
        if (inputStreamReader != null) {
            val bufferedReader = BufferedReader(inputStreamReader)
            val partialStr = StringBuilder()
            var done = false
            while (!done) {
                var line = bufferedReader.readLine()
                done = (line == null);
                if (line != null) partialStr.append(line);
            }
            inputStreamReader.close()
            str = partialStr.toString()
        }
    } catch (e: FileNotFoundException) {...} catch (e: IOException) {...}
    return str
}
```

❑ Main purpose to read in a stream of data from a file (more later)

The Serialization Mechanism

Placemark-Console Version 4.0


The Serialization Mechanism

1. App is launched and a **PlacemarkController** object is created and started via **.start()**

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



2. The **PlacemarkController** creates a **PlacemarkJSONStore** object



```
val placemarks = PlacemarkJSONStore()
```


which in turn creates it's own internal list of **Placemark** objects for the app




```
var placemarks = mutableListOf<PlacemarkModel>()
```

The Serialization Mechanism

3. If a file already exists, **placemarks** is populated with the data from the file via **deserialize()**

```
init {  
    if (exists(JSON_FILE)) {  
        deserialize()   
    }  
}
```

4. Otherwise the data is written to file every time a new **Placemark** is created, updated or deleted with **serialize()**

```
override fun create(placemark: PlacemarkModel) {  
    placemark.id = generateRandomId()  
    placemarks.add(placemark)  
    serialize()   
}
```


The Serialization Mechanism

```

1 package org.wit.placemark.console.controllers
2
3 import ...
4
5 class PlacemarkController {
6
7     val placemarks = PlacemarkJSONStore()
8     val placemarkView = PlacemarkView()
9     val logger = KotlinLogging.logger {}
10
11     init {...}
12     fun start() {...}
13     fun menu(): Int { return placemarkView.menu() }
14     fun add(): ...
15     fun list(): {...}
16     fun update(): {...}
17     fun delete(): {...}
18     fun search(): {...}
19     fun search(id: Long) : PlacemarkModel? {...}
20     fun dummyData() {...}
21 }
    
```

```

1 package org.wit.placemark.console.models
2
3 import ...
4
5 private val logger = KotlinLogging.logger {}
6
7 val JSON_FILE = "placemarks.json"
8 val gsonBuilder = GsonBuilder().setPrettyPrinting().create()
9 val listType = object : TypeToken<java.util.ArrayList<PlacemarkModel>>() {}.type
10
11 fun generateRandomId(): Long {...}
12
13 class PlacemarkJSONStore : PlacemarkStore {
14
15     var placemarks = mutableListOf<PlacemarkModel>()
16
17     init {
18         if (exists(JSON_FILE)) {
19             deserialize()
20         }
21     }
22
23     override fun findAll(): MutableList<PlacemarkModel> {...}
24     override fun findOne(id: Long) : PlacemarkModel? {...}
25     override fun create(placemark: PlacemarkModel) {
26         placemark.id = generateRandomId()
27         placemarks.add(placemark)
28         serialize()
29     }
30
31     override fun update(placemark: PlacemarkModel) {...}
32     override fun delete(placemark: PlacemarkModel) {...}
33     internal fun logAll() {...}
34     private fun serialize() {...}
35     private fun deserialize() {...}
36 }
    
```



```

1 package org.wit.placemark.console.helpers
2
3 import ...
4
5 val logger = KotlinLogging.logger {}
6
7 fun write(fileName: String, data: String) {...}
8
9 fun read(fileName: String): String {...}
10
11 fun exists(fileName: String): Boolean {...}
    
```

The Serialization Mechanism – Using Gson

- ❑ Add necessary library to our dependencies

```
dependencies {  
    implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk8"  
    testCompile group: 'junit', name: 'junit', version: '4.12'  
  
    implementation 'org.slf4j:slf4j-simple:1.6.1'  
    implementation 'io.github.microutils:kotlin-logging:1.6.22'  
    implementation "com.google.code.gson:gson:2.8.5"  
}
```



The Serialization Mechanism – Using Gson

❑ Create a GsonBuilder

❑ Define Object Type for Builder

❑ Convert list of *placemarks* to JSON

❑ Convert JSON to list of *placemarks*

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

import ...

private val logger = KotlinLogging.logger {}

val JSON_FILE = "placemarks.json"
val gsonBuilder = GsonBuilder().setPrettyPrinting().create()
val listType = object : TypeToken<java.util.ArrayList<PlacemarkModel>>() {}.type

fun generateRandomId(): Long {...}

class PlacemarkJSONStore : PlacemarkStore {

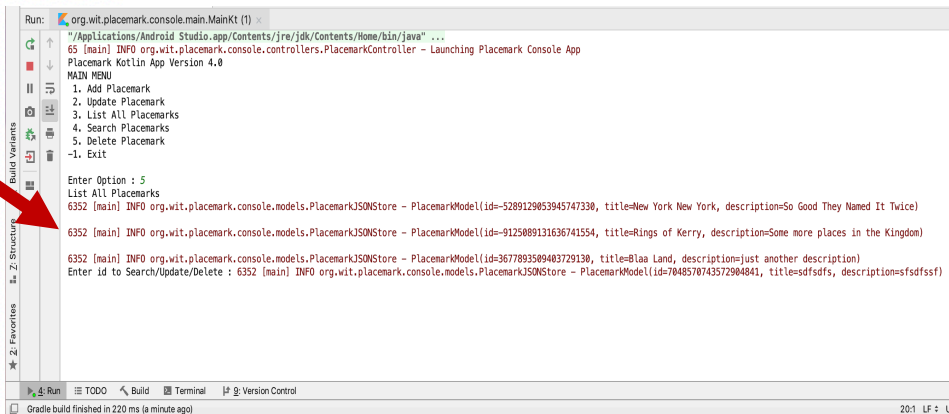
    var placemarks = mutableListOf<PlacemarkModel>()

    init {...}
    override fun findAll(): MutableList<PlacemarkModel> {...}
    override fun findOne(id: Long) : PlacemarkModel? {...}
    override fun create(placemark: PlacemarkModel) {...}
    override fun update(placemark: PlacemarkModel) {...}
    override fun delete(placemark: PlacemarkModel) {...}
    internal fun logAll() {...}
    private fun serialize() {
        val jsonString = gsonBuilder.toJson(placemarks, listType)
        write(JSON_FILE, jsonString)
    }
    private fun deserialize() {
        val jsonString = read(JSON_FILE)
        placemarks = Gson().fromJson(jsonString, listType)
    }
}
```

The Serialization Mechanism – Using Gson

```
[
  {
    "id": -5289129053945747330,
    "title": "New York New York",
    "description": "So Good They Named It Twice"
  },
  {
    "id": -9125089131636741554,
    "title": "Rings of Kerry",
    "description": "Some more places in the Kingdom"
  },
  {
    "id": 3677893509403729130,
    "title": "Blaa Land",
    "description": "just another description"
  },
  {
    "id": 7048570743572904841,
    "title": "sdfsdfs",
    "description": "sfsdfssf"
  }
]
```

 *placemarks.json*





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>

