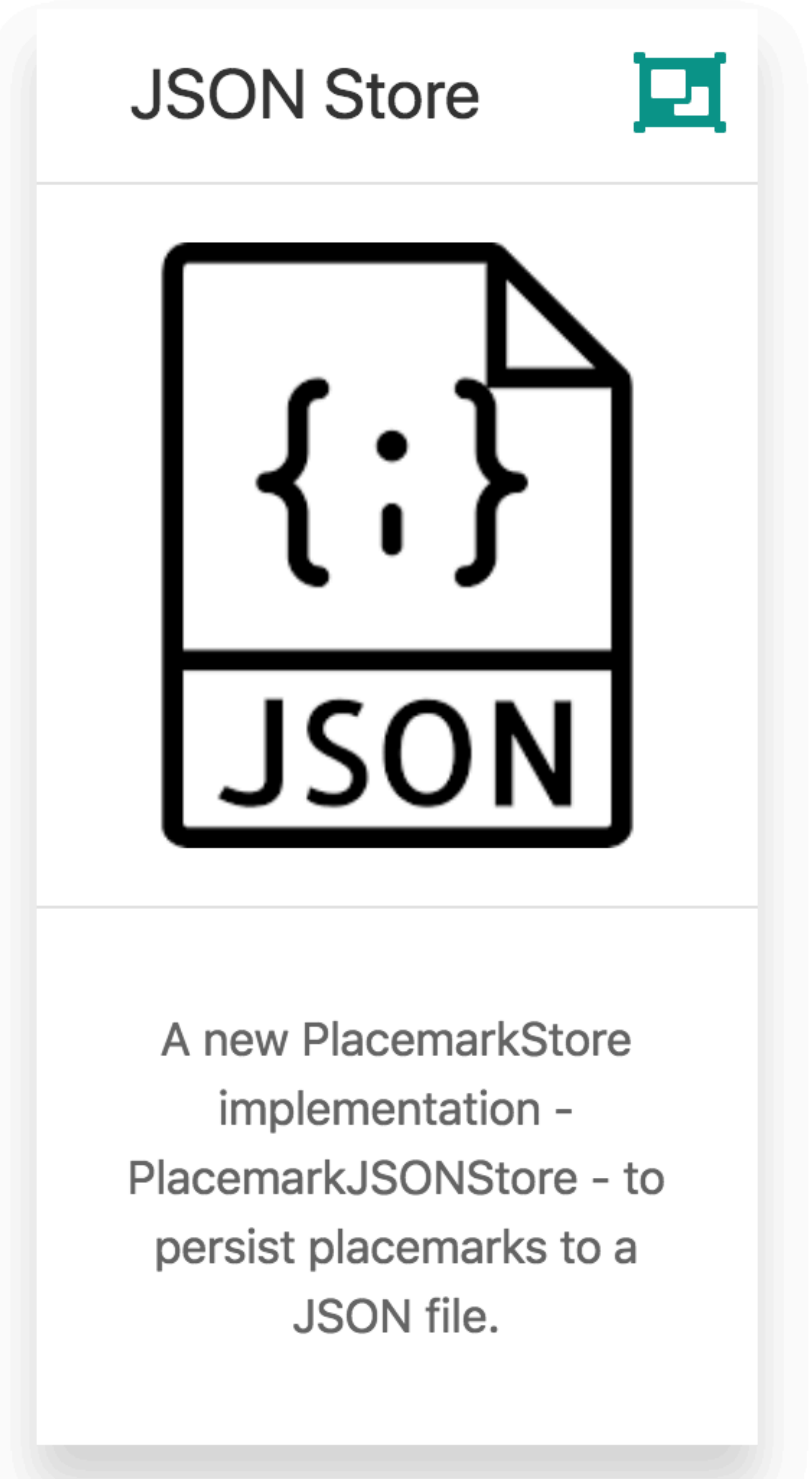
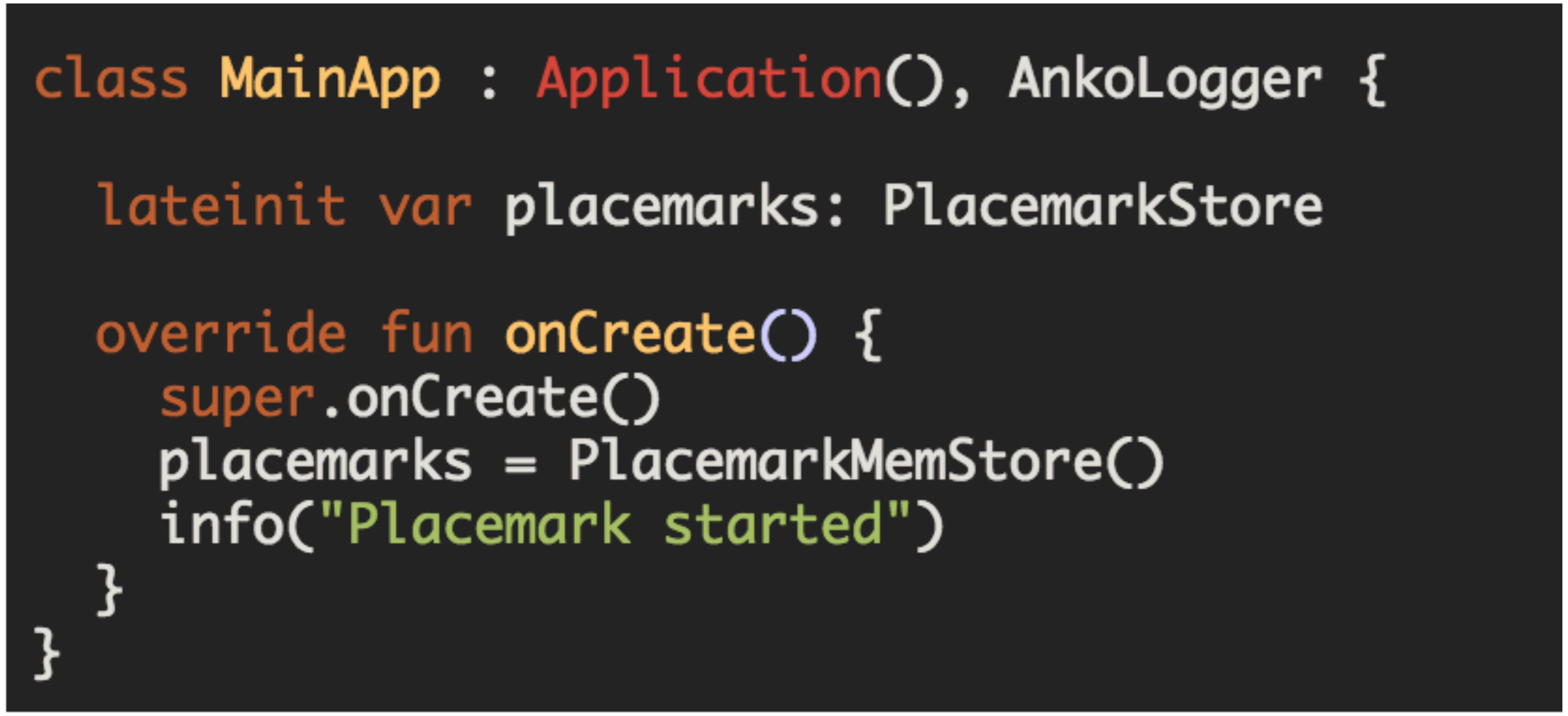
PlacemarkJSONStore



JSON Store

A new PlacemarkStore implementation - PlacemarkJSONStore - to persist placemarks to a JSON file.

PlacemarkStore Initialisation



```kotlin
class MainApp : Application(), AnkoLogger {

  lateinit var placemarks: PlacemarkStore

  override fun onCreate() {
    super.onCreate()
    placemarks = PlacemarkMemStore()
    info("Placemark started")
  }
}
```

Declare placemarks as “PlacemarkStore” type

Then create a PlaceMemStore on initialisation

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

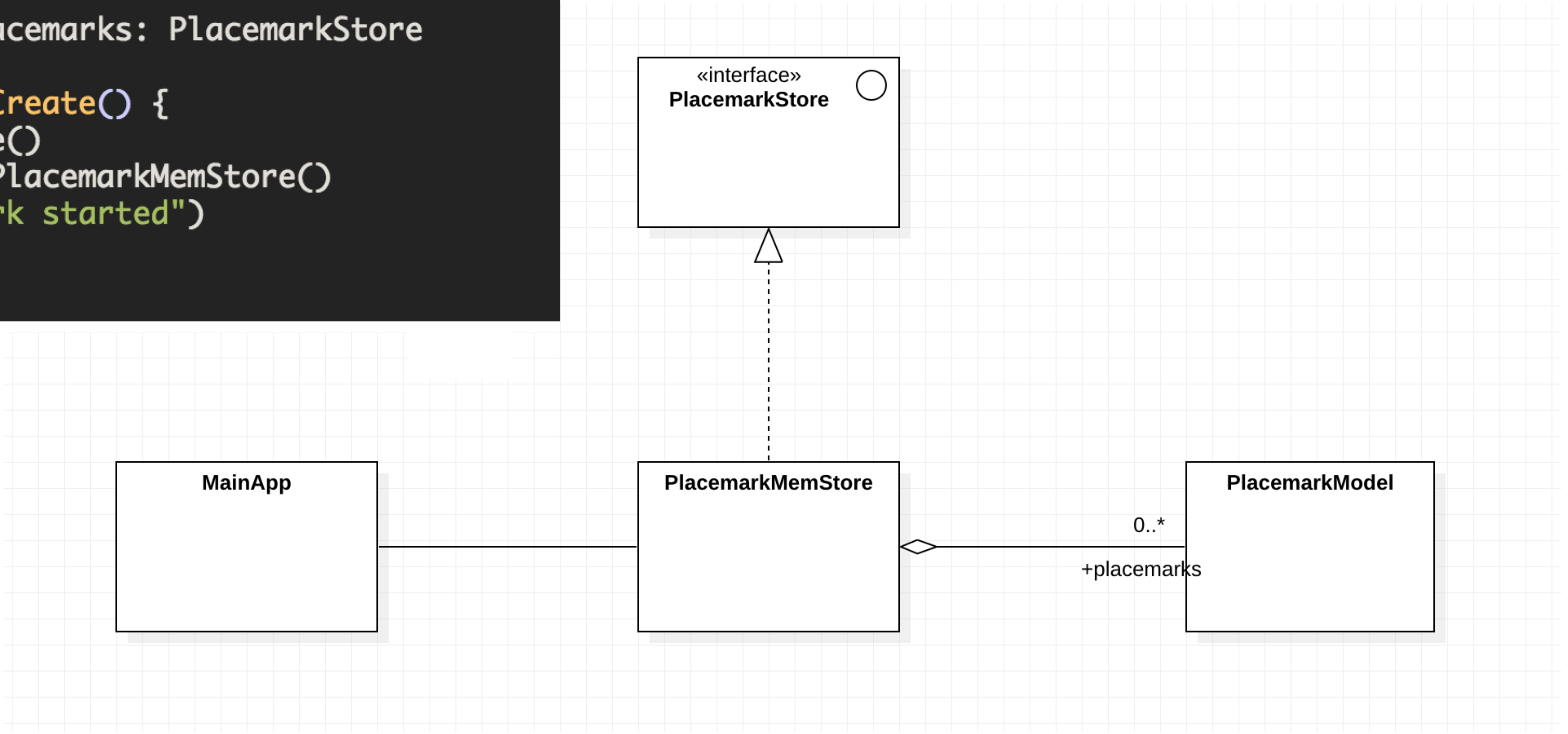
PlacemarkStore

```
class PlacemarkMemStore : PlacemarkStore, AnkoLogger {

  val placemarks = ArrayList<PlacemarkModel>()

  override fun findAll(): List<PlacemarkModel> {
    return placemarks
  }

  override fun create(placemark: PlacemarkModel) {
    placemark.id = getId()
    placemarks.add(placemark)
    logAll()
  }

  override fun update(placemark: PlacemarkModel) {
    var foundPlacemark: PlacemarkModel? = placemarks.find { p -> p.id == placemark.id }
    if (foundPlacemark != null) {
      foundPlacemark.title = placemark.title
      foundPlacemark.description = placemark.description
      foundPlacemark.image = placemark.image
      foundPlacemark.lat = placemark.lat
      foundPlacemark.lng = placemark.lng
      foundPlacemark.zoom = placemark.zoom
      logAll();
    }
  }

  override fun delete(placemark: PlacemarkModel) {
    placemarks.remove(placemark)
  }

  fun logAll() {
    placemarks.forEach { info("${it}") }
  }
}
```

PlacemarkMemStore

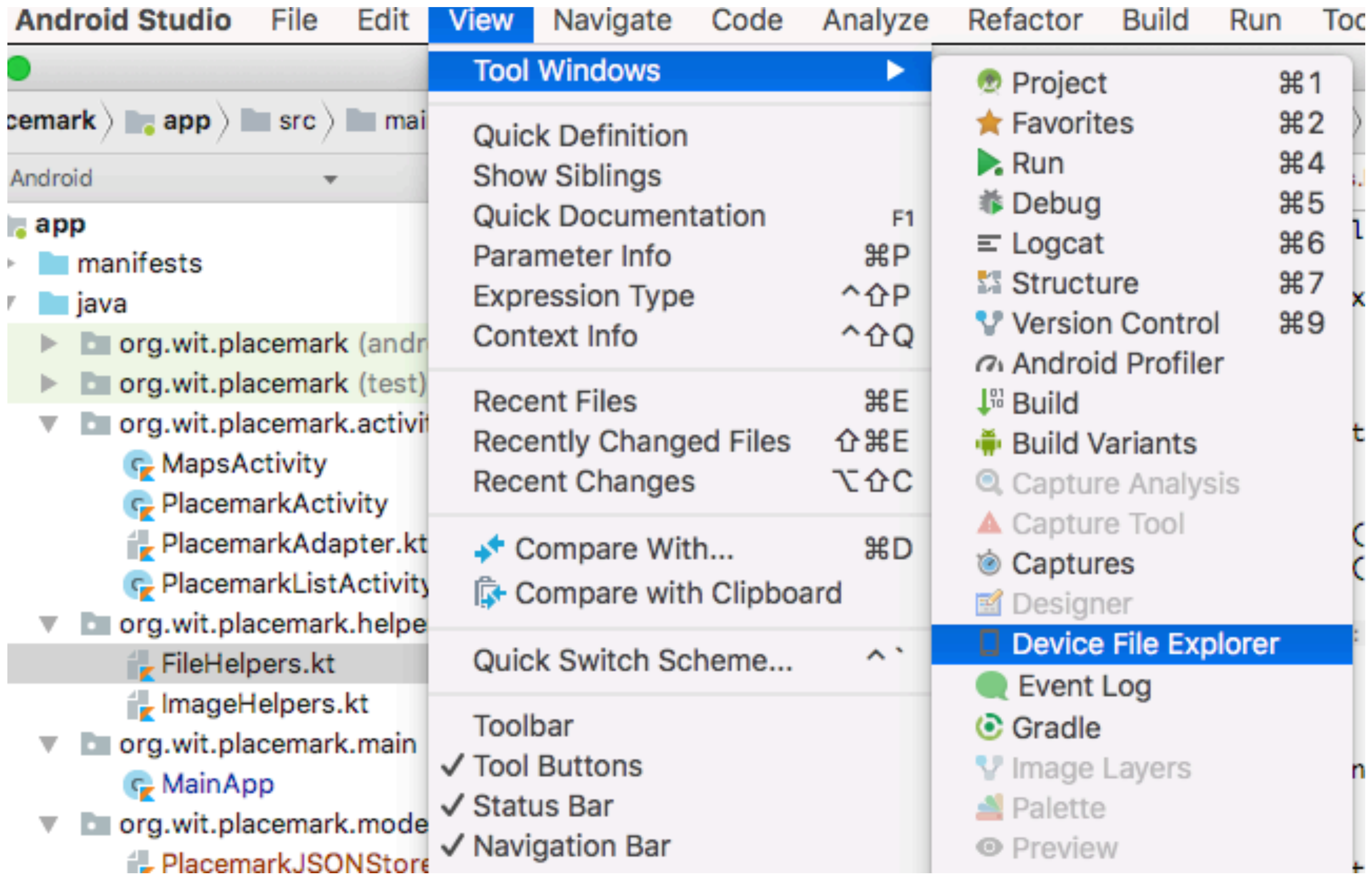
PlacemarkStore & PlacemarkMemStore

```kotlin
class MainApp : Application(), AnkoLogger {

  lateinit var placemarks: PlacemarkStore

  override fun onCreate() {
    super.onCreate()
    placemarks = PlacemarkMemStore()
    info("Placemark started")
  }
}
```

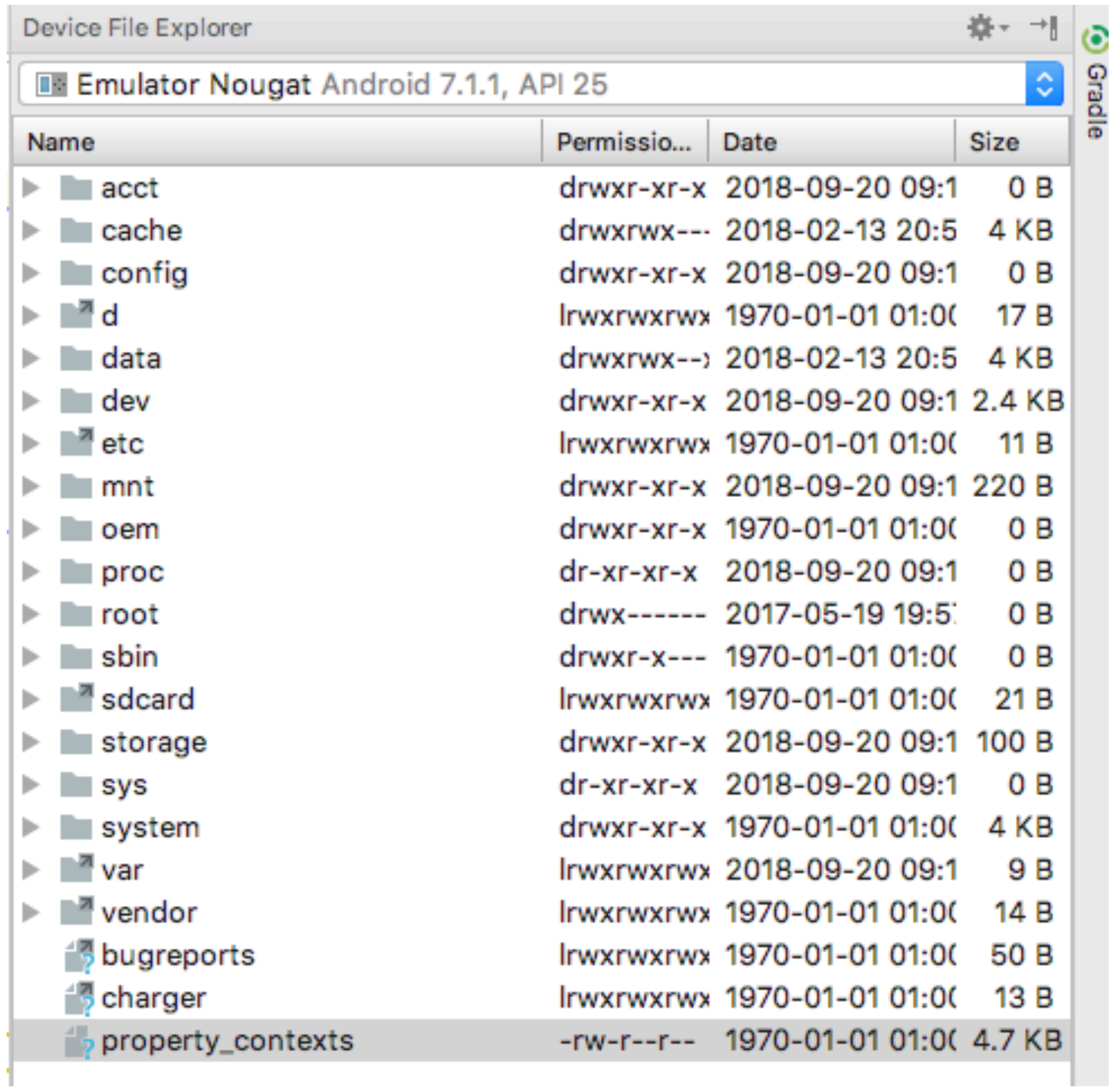


Android File System

Android Devices support a full filesystem



This file system can be browsed from Studio (emulator or actual device)



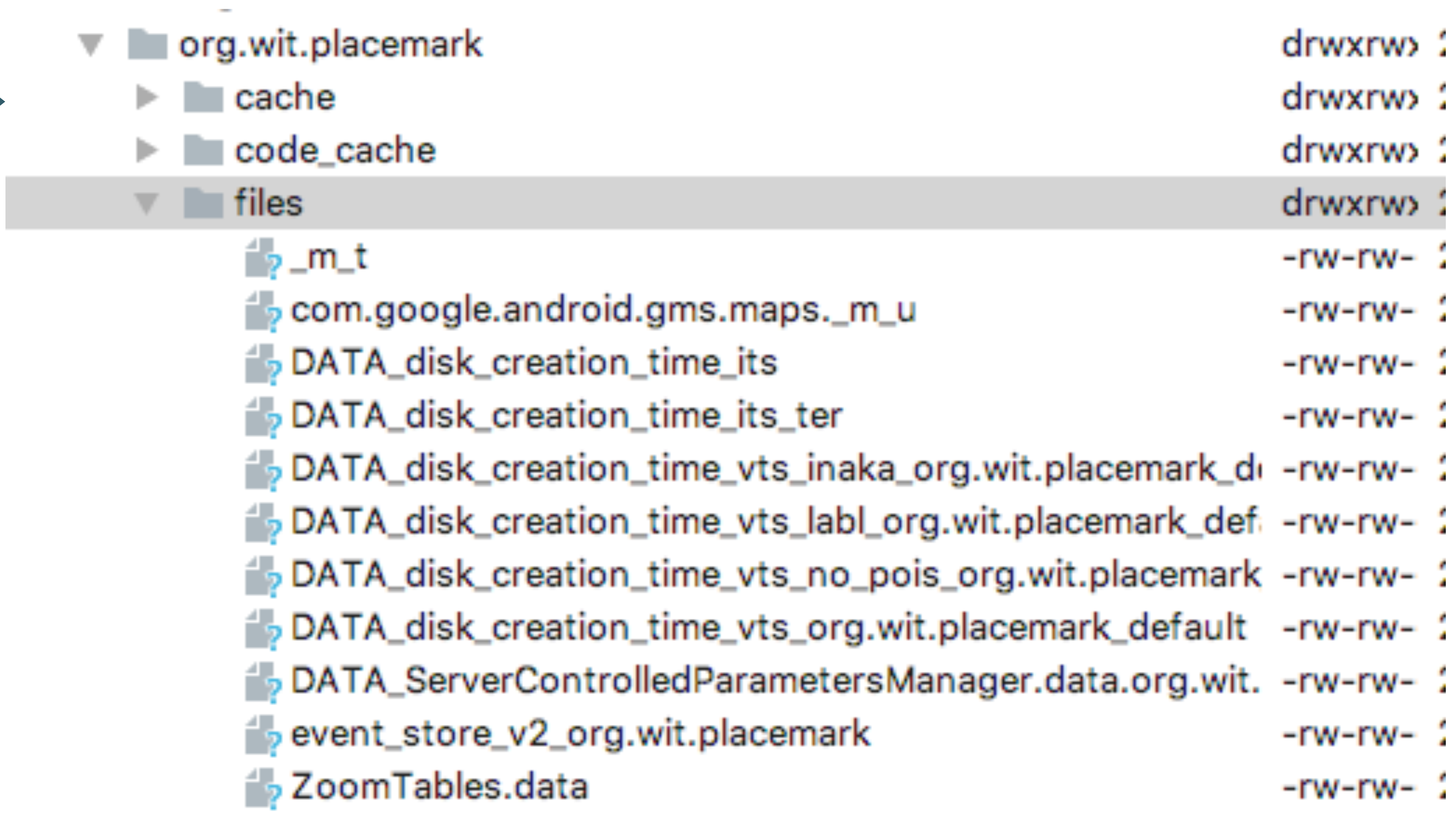
Emulator Nougat Android 7.1.1, API 25

Name	Permissio...	Date	Size
acct	drwxr-xr-x	2018-09-20 09:1	0 B
cache	drwxrwx---	2018-02-13 20:5	4 KB
config	drwxr-xr-x	2018-09-20 09:1	0 B
d	lrwxrwxrwx	1970-01-01 01:0(	17 B
data	drwxrwx--)	2018-02-13 20:5	4 KB
dev	drwxr-xr-x	2018-09-20 09:1	2.4 KB
etc	lrwxrwxrwx	1970-01-01 01:0(	11 B
mnt	drwxr-xr-x	2018-09-20 09:1	220 B
oem	drwxr-xr-x	1970-01-01 01:0(	0 B
proc	dr-xr-xr-x	2018-09-20 09:1	0 B
root	drwx------	2017-05-19 19:5	0 B
sbin	drwxr-x---	1970-01-01 01:0(	0 B
sdcard	lrwxrwxrwx	1970-01-01 01:0(	21 B
storage	drwxr-xr-x	2018-09-20 09:1	100 B
sys	dr-xr-xr-x	2018-09-20 09:1	0 B
system	drwxr-xr-x	1970-01-01 01:0(	4 KB
var	lrwxrwxrwx	2018-09-20 09:1	9 B
vendor	lrwxrwxrwx	1970-01-01 01:0(	14 B
bugreports	lrwxrwxrwx	1970-01-01 01:0(	50 B
charger	lrwxrwxrwx	1970-01-01 01:0(	13 B
property_contexts	-rw-r--r--	1970-01-01 01:0(	4.7 KB

Application Folder

Applications will be largely confined to a specific folder created when the app is installed

We can read/write files too this folder - and also browse their contents in Studio

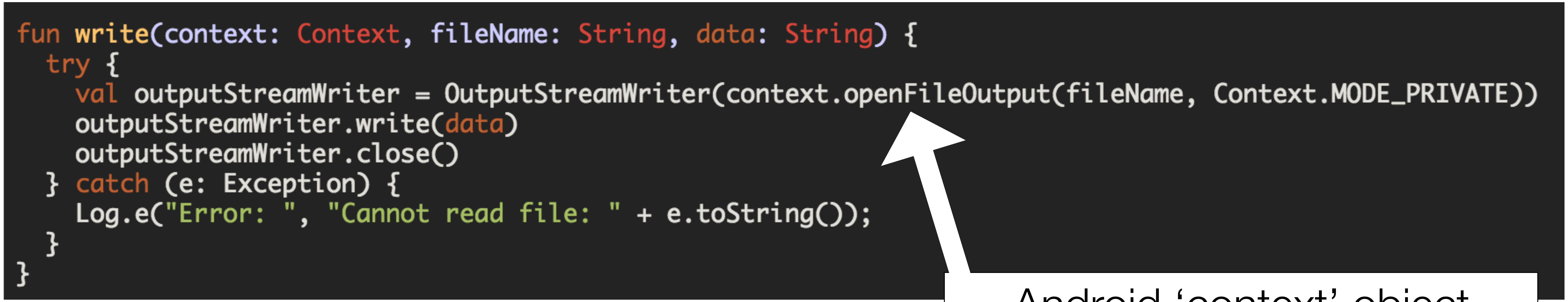


```
▼ org.wit.placemark                                                     drwxrwx
  ▶ cache                                                               drwxrwx
  ▶ code_cache                                                          drwxrwx
  ▼ files                                                               drwxrwx
      _m_t                                                              -rw-rw-
      com.google.android.gms.maps._m_u                                  -rw-rw-
      DATA_disk_creation_time_its                                       -rw-rw-
      DATA_disk_creation_time_its_ter                                   -rw-rw-
      DATA_disk_creation_time_vts_inaka_org.wit.placemark_d             -rw-rw-
      DATA_disk_creation_time_vts_labl_org.wit.placemark_def            -rw-rw-
      DATA_disk_creation_time_vts_no_pois_org.wit.placemark             -rw-rw-
      DATA_disk_creation_time_vts_org.wit.placemark_default             -rw-rw-
      DATA_ServerControlledParametersManager.data.org.wit.              -rw-rw-
      event_store_v2_org.wit.placemark                                  -rw-rw-
      ZoomTables.data                                                   -rw-rw-
```

`/data/data/org.wit.placemark/files`

File Helper functions: write()

Simple function to write a string to a file

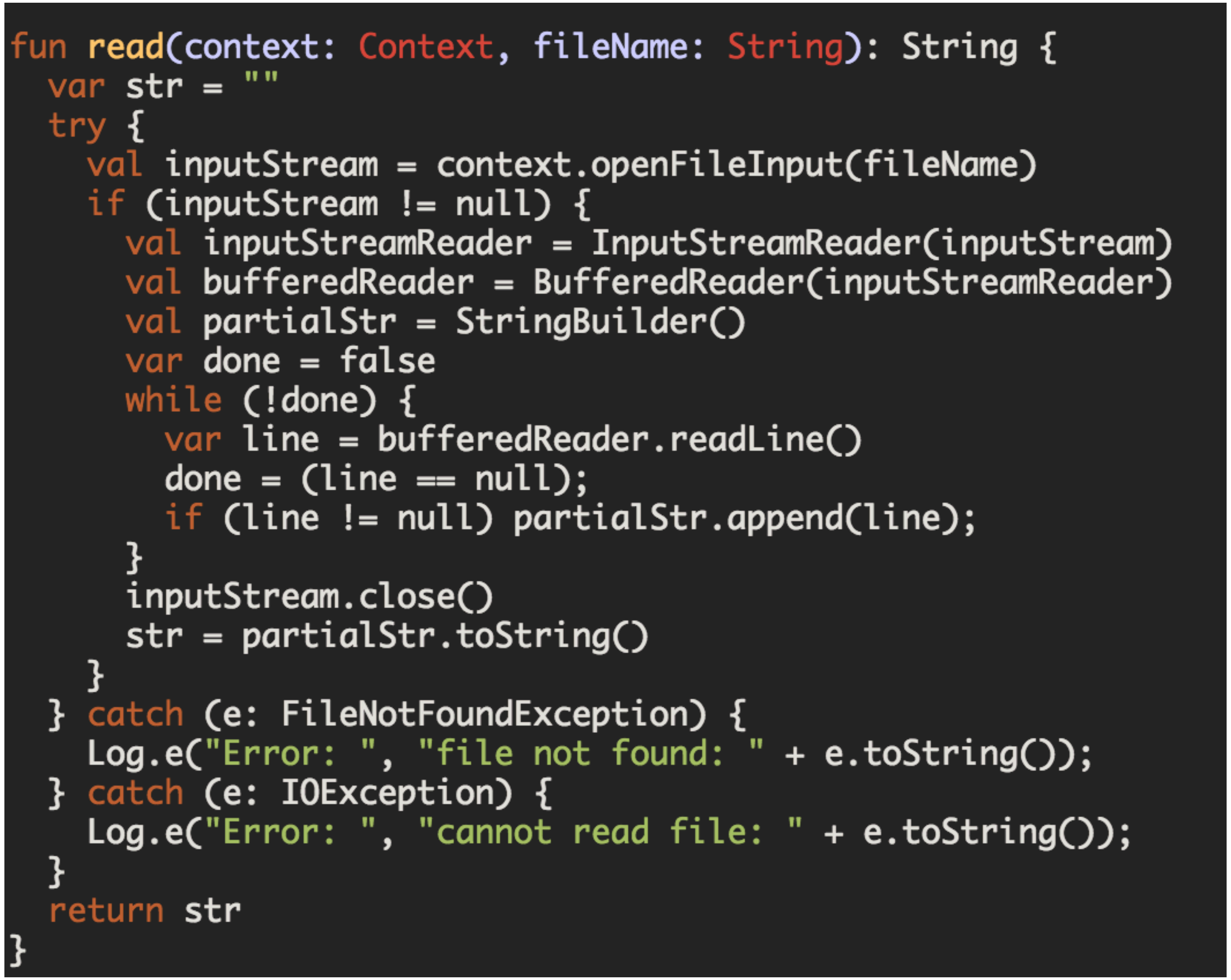


```kotlin
fun write(context: Context, fileName: String, data: String) {
  try {
    val outputStreamWriter = OutputStreamWriter(context.openFileOutput(fileName, Context.MODE_PRIVATE))
    outputStreamWriter.write(data)
    outputStreamWriter.close()
  } catch (e: Exception) {
    Log.e("Error: ", "Cannot read file: " + e.toString());
  }
}
```

Android ‘context’ object required to locate and open file in Application data folder

Uses standard Java Streams methods

File Helper functions: read()



```
fun read(context: Context, fileName: String): String {
  var str = ""
  try {
    val inputStream = context.openFileInput(fileName)
    if (inputStream != null) {
      val inputStreamReader = InputStreamReader(inputStream)
      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);
      }
      inputStream.close()
      str = partialStr.toString()
    }
  } catch (e: FileNotFoundException) {
    Log.e("Error: ", "file not found: " + e.toString());
  } catch (e: IOException) {
    Log.e("Error: ", "cannot read file: " + e.toString());
  }
  return str
}
```

Also uses standard Java Streams methods + context to open file

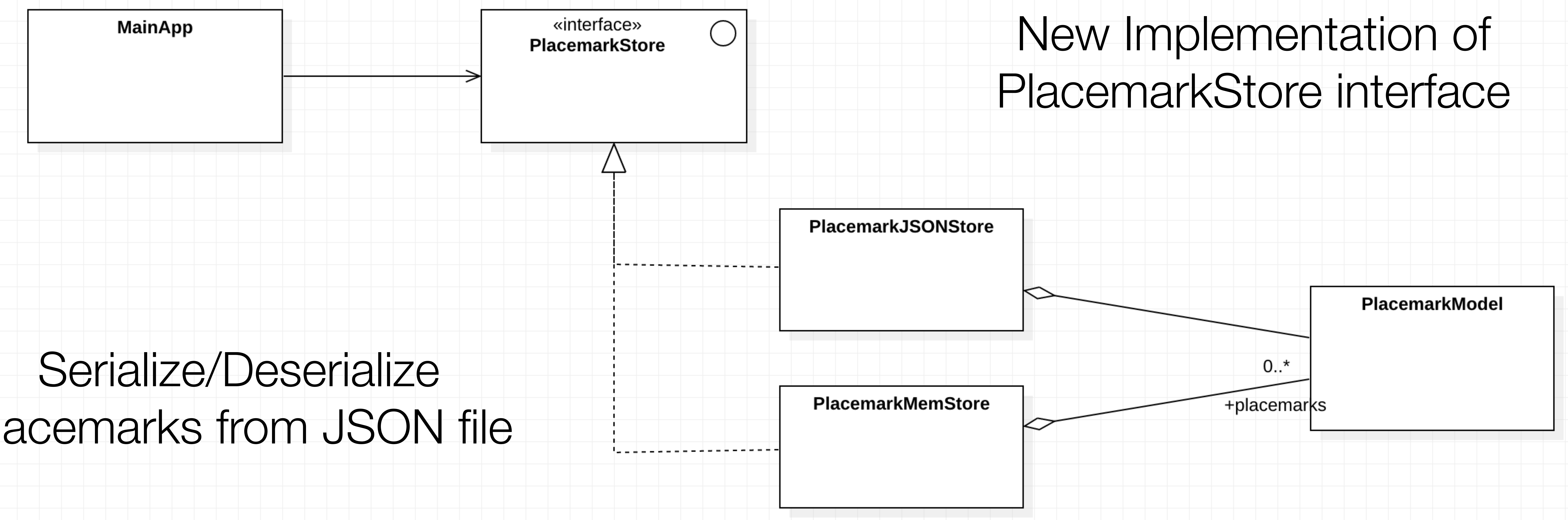
Read string line by line and return complete file contents.

Could be optimised - but keep simple for debug purposes (for the moment)

File Helper functions: exists()

```kotlin
fun exists(context: Context, filename: String): Boolean {
  val file = context.getFileStreamPath(filename)
  return file.exists()
}
```

PlacemarkJSONStore Specification



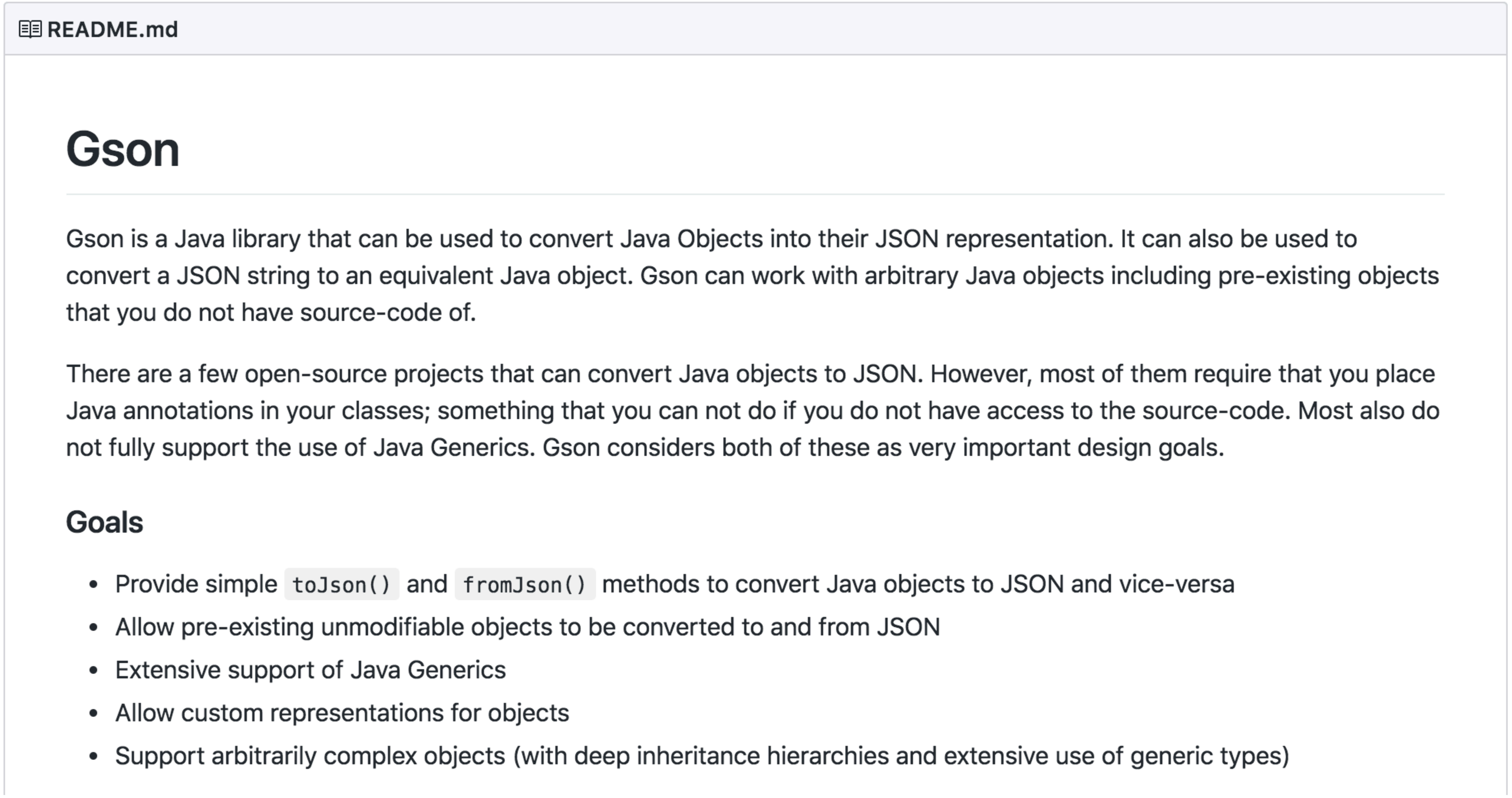
New Implementation of PlacemarkStore interface

Serialize/Deserialize Placemarks from JSON file

Use file helpers - read(), write() & exists()

GSON https://github.com/google/gson

Google Library to support JSON Encoding/ Decoding in Java



README.md

Gson

Gson is a Java library that can be used to convert Java Objects into their JSON representation. It can also be used to convert a JSON string to an equivalent Java object. Gson can work with arbitrary Java objects including pre-existing objects that you do not have source-code of.

There are a few open-source projects that can convert Java objects to JSON. However, most of them require that you place Java annotations in your classes; something that you can not do if you do not have access to the source-code. Most also do not fully support the use of Java Generics. Gson considers both of these as very important design goals.

Goals

- Provide simple `toJson()` and `fromJson()` methods to convert Java objects to JSON and vice-versa
- Allow pre-existing unmodifiable objects to be converted to and from JSON
- Extensive support of Java Generics
- Allow custom representations for objects
- Support arbitrarily complex objects (with deep inheritance hierarchies and extensive use of generic types)

build.gradle

```
implementation "com.google.code.gson:gson:2.8.5"
```

```
package org.wit.placemark.models

import android.content.Context
import com.google.gson.Gson
import com.google.gson.GsonBuilder
import com.google.gson.reflect.TypeToken
import org.jetbrains.anko.AnkoLogger
import org.wit.placemark.helpers.*
import java.util.*

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

fun generateRandomId(): Long {
  return Random().nextLong()
}

class PlacemarkJSONStore : PlacemarkStore, AnkoLogger {

  val context: Context
  var placemarks = mutableListOf<PlacemarkModel>()

  constructor (context: Context) {
    this.context = context
    if (exists(context, JSON_FILE)) {
      deserialize()
    }
  }

  override fun findAll(): MutableList<PlacemarkModel> {
    return placemarks
  }
```

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

  override fun update(placemark: PlacemarkModel) {
    ...
  }

  override fun delete(placemark: PlacemarkModel) {
   ...
  }

  private fun serialize() {
    val jsonString = gsonBuilder.toJson(placemarks, listType)
    write(context, JSON_FILE, jsonString)
  }

  private fun deserialize() {
    val jsonString = read(context, JSON_FILE)
    placemarks = Gson().fromJson(jsonString, listType)
  }
}
```

PlacmarkJSONStore - preamble

imports

```kotlin
import android.content.Context
import com.google.gson.Gson
import com.google.gson.GsonBuilder
import com.google.gson.reflect.TypeToken
import org.jetbrains.anko.AnkoLogger
import org.wit.placemark.helpers.*
import java.util.*
```

Filename for placemarks store

```kotlin
val JSON_FILE = "placemarks.json"
```

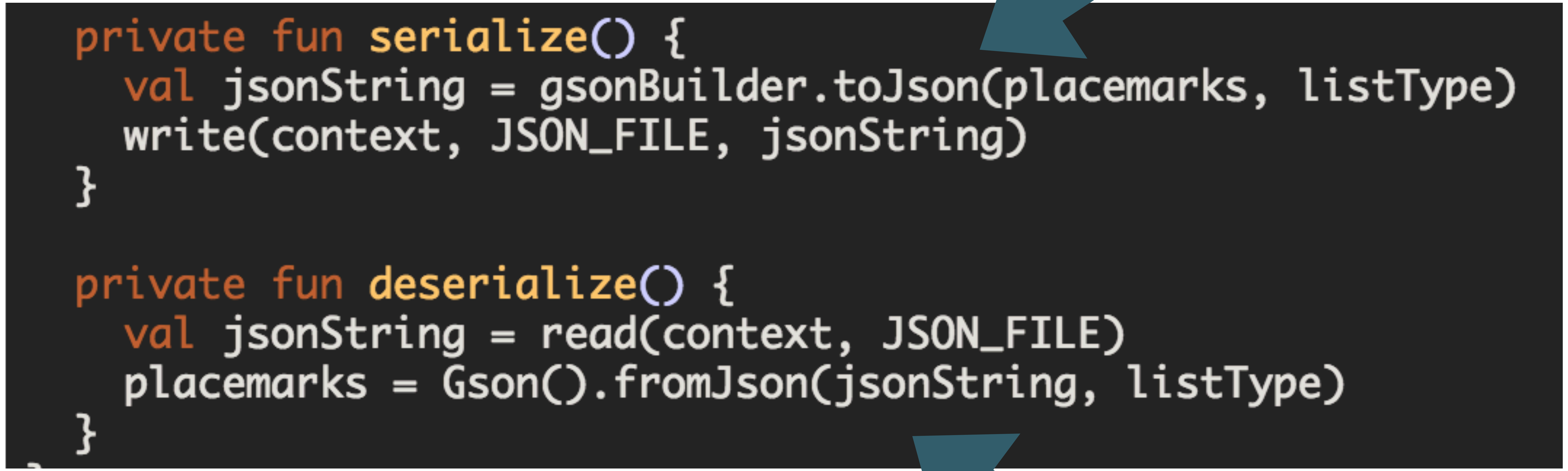
Helper variables for use with GSON parser

```kotlin
val gsonBuilder = GsonBuilder().setPrettyPrinting().create()
val listType = object : TypeToken<java.util.ArrayList<PlacemarkModel>>() {}.type
```

Unique ID generator

```kotlin
fun generateRandomId(): Long {
  return Random().nextLong()
}
```

Methods to Read & Write Placemarks Array to/from Json file

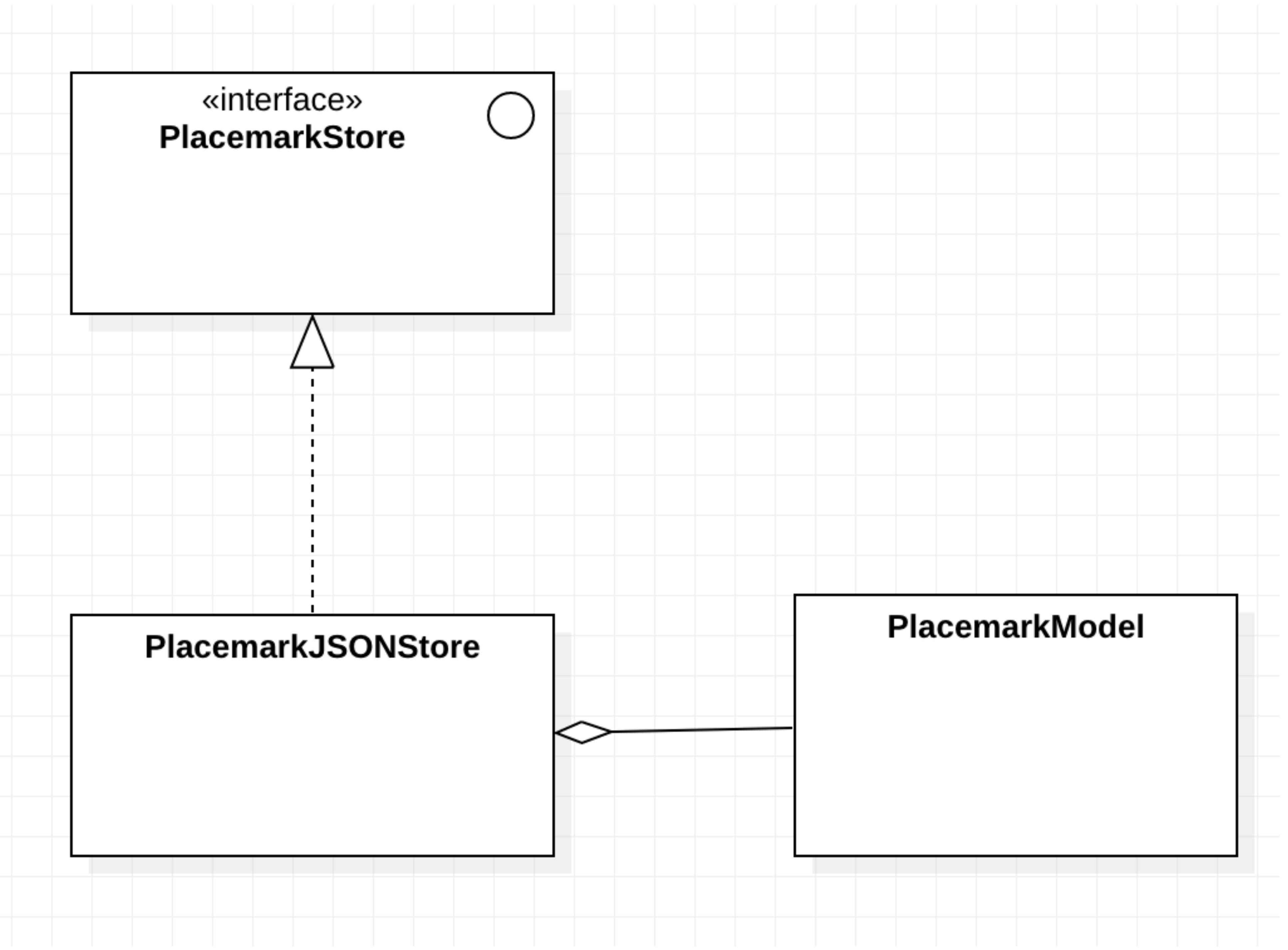


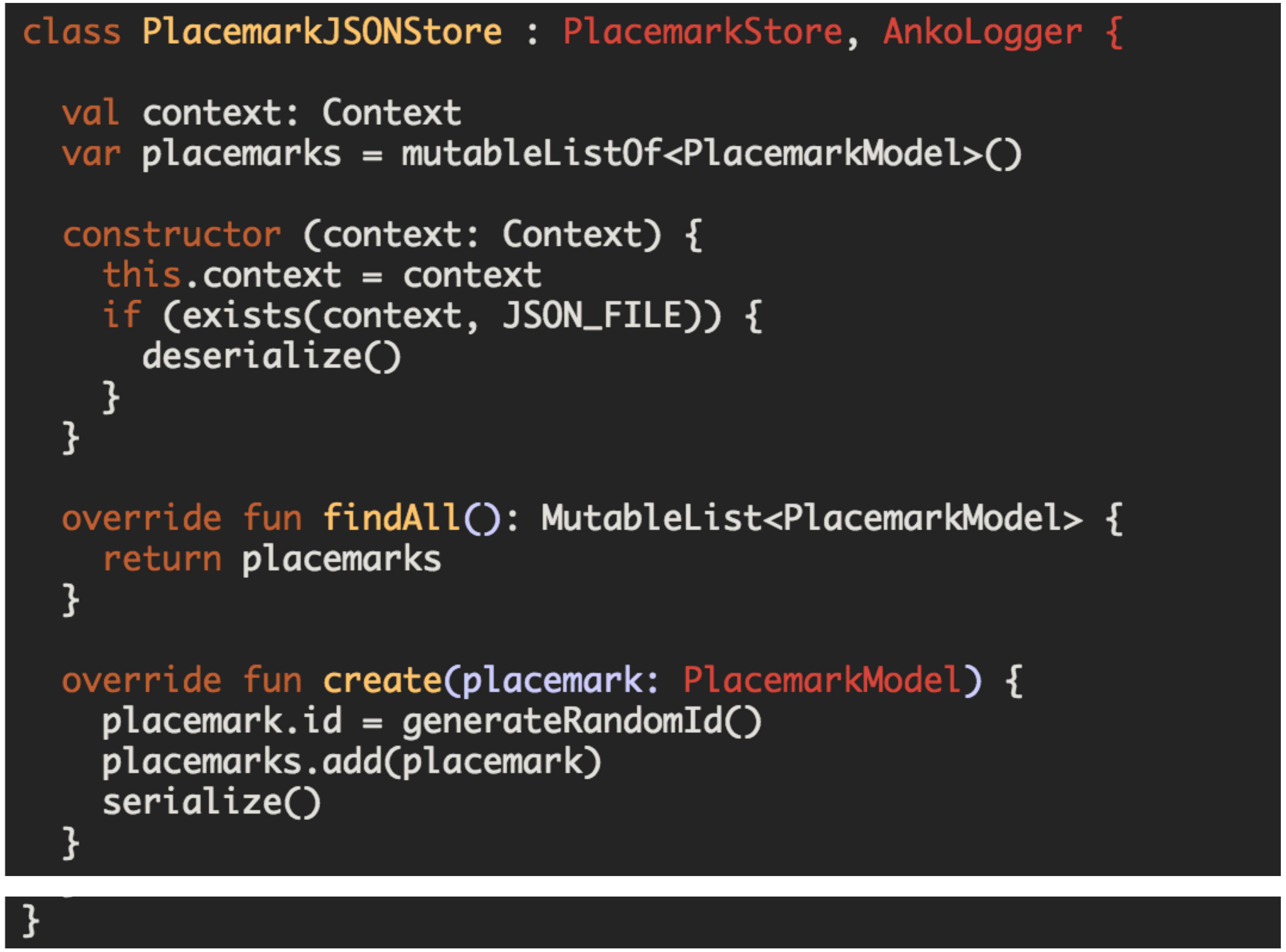
```
private fun serialize() {
  val jsonString = gsonBuilder.toJson(placemarks, listType)
  write(context, JSON_FILE, jsonString)
}

private fun deserialize() {
  val jsonString = read(context, JSON_FILE)
  placemarks = Gson().fromJson(jsonString, listType)
}
```

Gson parser converts Placemarks to JSON string

Gson parser converts a JSON string to Placemarks list

Load placemarks when Store created





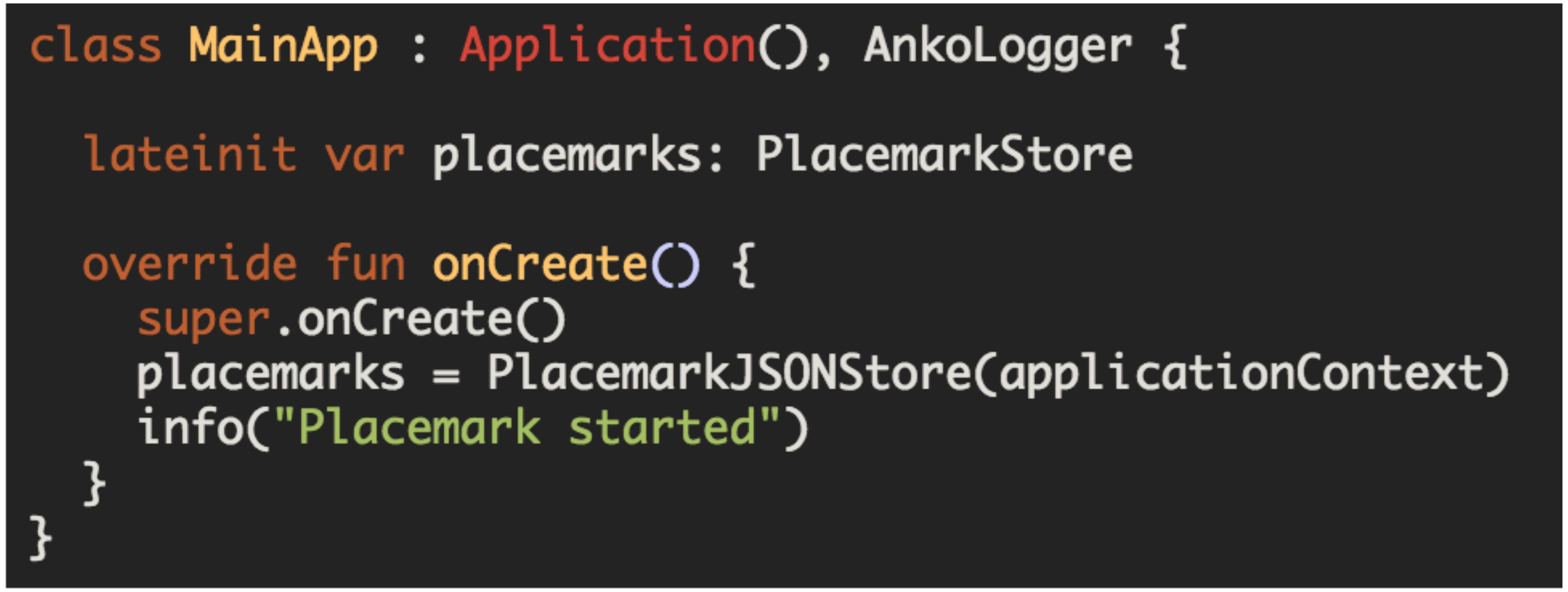
```
class PlacemarkJSONStore : PlacemarkStore, AnkoLogger {

  val context: Context
  var placemarks = mutableListOf<PlacemarkModel>()

  constructor (context: Context) {
    this.context = context
    if (exists(context, JSON_FILE)) {
      deserialize()
    }
  }

  override fun findAll(): MutableList<PlacemarkModel> {
    return placemarks
  }

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

}
```

Save place marks whenever each Placemark created

```
package org.wit.placemark.models

import android.content.Context
import com.google.gson.Gson
import com.google.gson.GsonBuilder
import com.google.gson.reflect.TypeToken
import org.jetbrains.anko.AnkoLogger
import org.wit.placemark.helpers.*
import java.util.*

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

fun generateRandomId(): Long {
  return Random().nextLong()
}

class PlacemarkJSONStore : PlacemarkStore, AnkoLogger {

  val context: Context
  var placemarks = mutableListOf<PlacemarkModel>()

  constructor (context: Context) {
    this.context = context
    if (exists(context, JSON_FILE)) {
      deserialize()
    }
  }

  override fun findAll(): MutableList<PlacemarkModel> {
    return placemarks
  }
```

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

  override fun update(placemark: PlacemarkModel) {
    ...
  }

  override fun delete(placemark: PlacemarkModel) {
   ...
  }

  private fun serialize() {
    val jsonString = gsonBuilder.toJson(placemarks, listType)
    write(context, JSON_FILE, jsonString)
  }

  private fun deserialize() {
    val jsonString = read(context, JSON_FILE)
    placemarks = Gson().fromJson(jsonString, listType)
  }
}
```

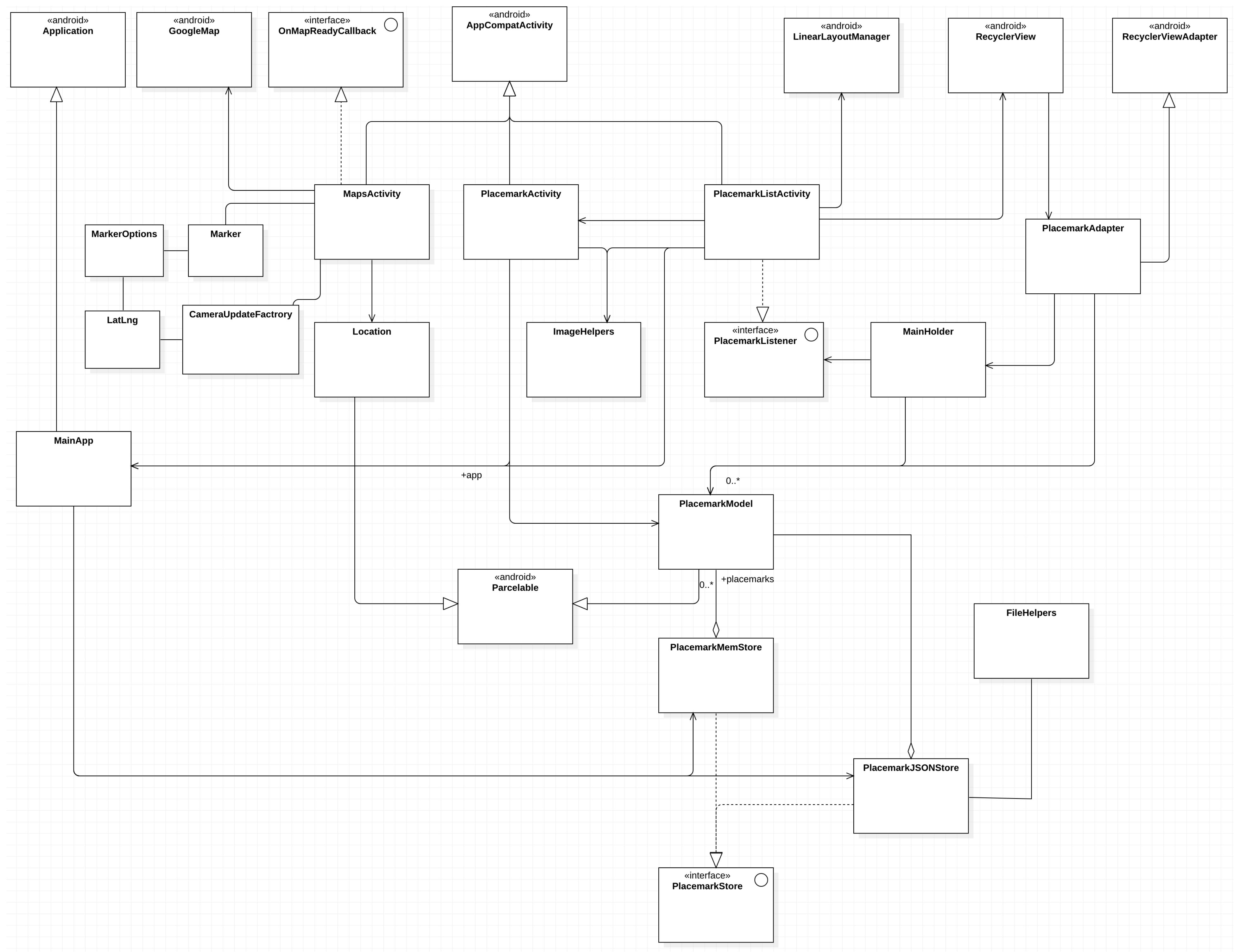
MainApp

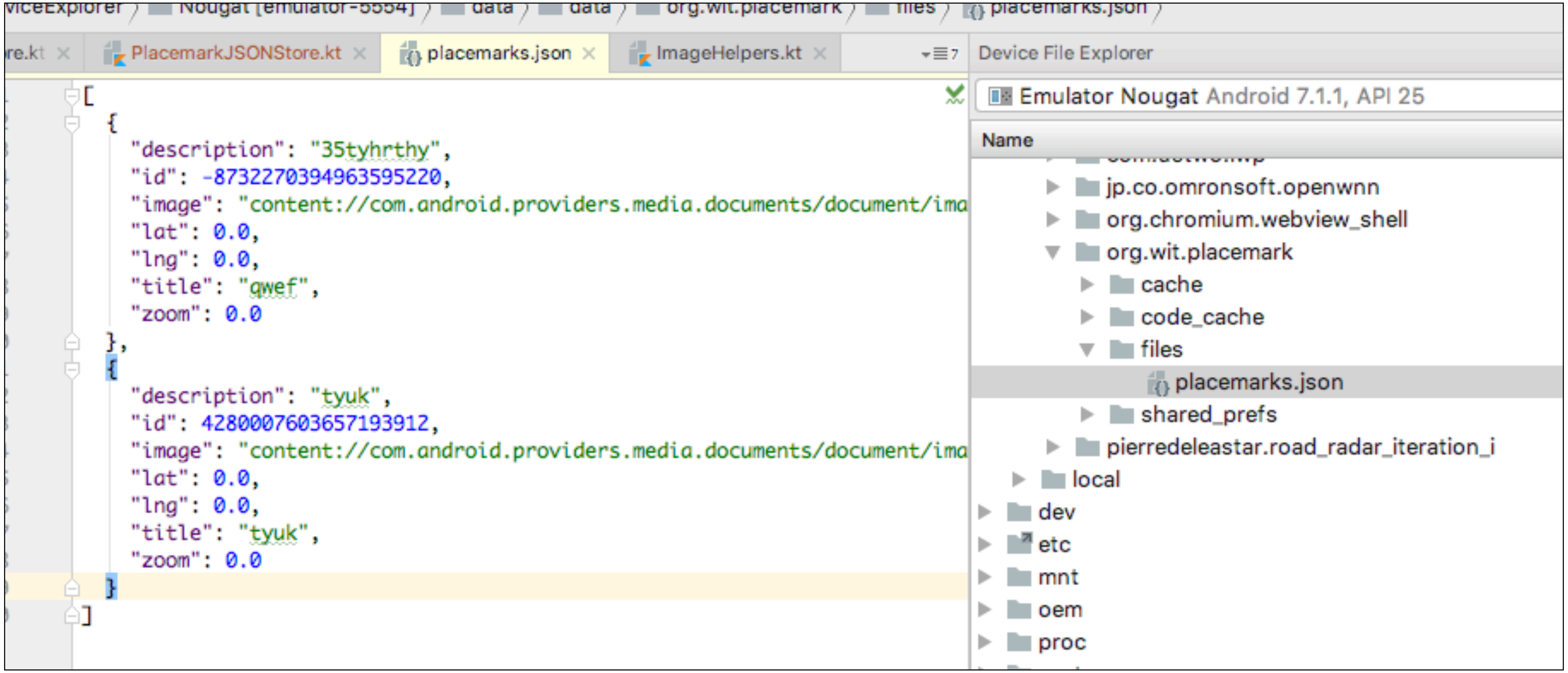


```
class MainApp : Application(), AnkoLogger {

  lateinit var placemarks: PlacemarkStore

  override fun onCreate() {
    super.onCreate()
    placemarks = PlacemarkJSONStore(applicationContext)
    info("Placemark started")
  }
}
```

Switch to using PlacemarkJSONStore

No other changes need to application




viceExplorer 〉 Nougat [emulator-5554] 〉 data 〉 data 〉 org.wit.placemark 〉 files 〉 placemarks.json 〉

re.kt × | PlacemarkJSONStore.kt × | placemarks.json × | ImageHelpers.kt × | Device File Explorer

```
[
  {
    "description": "35tyhrthy",
    "id": -8732270394963595220,
    "image": "content://com.android.providers.media.documents/document/ima
    "lat": 0.0,
    "lng": 0.0,
    "title": "qwef",
    "zoom": 0.0
  },
  {
    "description": "tyuk",
    "id": 4280007603657193912,
    "image": "content://com.android.providers.media.documents/document/ima
    "lat": 0.0,
    "lng": 0.0,
    "title": "tyuk",
    "zoom": 0.0
  }
]
```

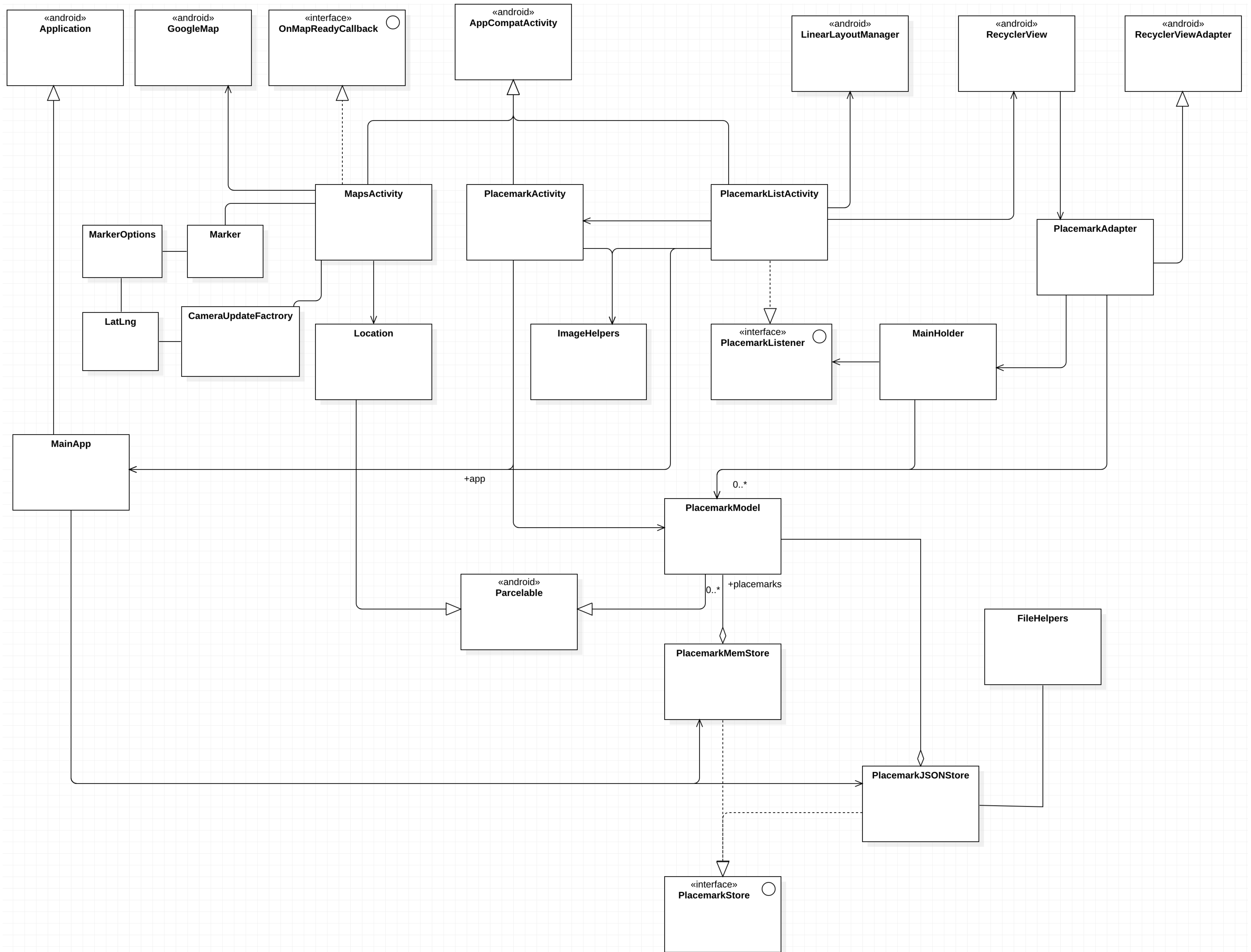
Emulator Nougat Android 7.1.1, API 25

Name

```
    ▶ jp.co.omronsoft.openwnn
    ▶ org.chromium.webview_shell
    ▼ org.wit.placemark
        ▶ cache
        ▶ code_cache
        ▼ files
            placemarks.json
        ▶ shared_prefs
    ▶ pierredeleastar.road_radar_iteration_i
  ▶ local
▶ dev
▶ etc
▶ mnt
▶ oem
▶ proc
```

Browse File in Studio

MainApp
+app
0..*
PlacemarkModel
«android»
Parcelable
0..*
+placemarks
PlacemarkMemStore
FileHelpers
PlacemarkJSONStore
«interface»
PlacemarkStore


«android»
Application
«android»
GoogleMap
«interface»
OnMapReadyCallback
«android»
AppCompatActivity
«android»
LinearLayoutManager
«android»
RecyclerView
«android»
RecyclerViewAdapter
MapsActivity
PlacemarkActivity
PlacemarkListActivity
MarkerOptions
Marker
PlacemarkAdapter
LatLng
CameraUpdateFactrory
Location
ImageHelpers
«interface»
PlacemarkListener
MainHolder
MainApp
+app
0..*
PlacemarkModel
«android»
Parcelable
0..*
+placemarks
FileHelpers
PlacemarkMemStore
PlacemarkJSONStore
«interface»
PlacemarkStore