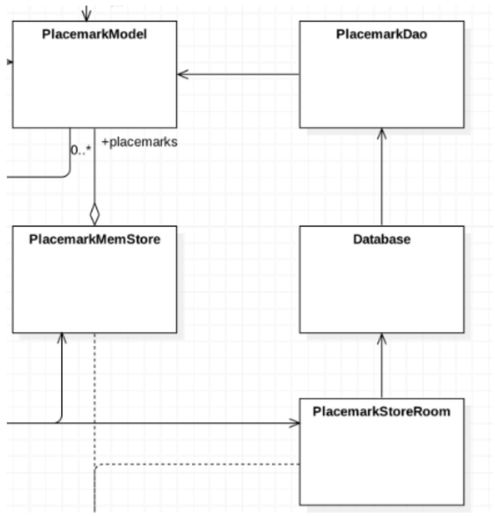
Rooms in Placemark

Rooms in Placemark



Implementation of PlacemarkStoreRoom, which stores placemarks in a SQLite database

build.gradle

A new plugin at the top of the file:

```
apply plugin: "kotlin-kapt"
```

Annotation Processor

New libraries:

Room Library

```
implementation 'androidx.room:room-runtime:2.1.0'
kapt 'androidx.room:room-compiler:2.1.0'
```

Annotation Processing with Kotlin

Annotation processors (see JSR 269) are supported in Kotlin with the *kapt* compiler plugin.

Being short, you can use libraries such as Dagger or Data Binding in your Kotlin projects.

Please read below about how to apply the *kapt* plugin to your Gradle/Maven build.

Using in Gradle

Apply the `kotlin-kapt` Gradle plugin:

```
apply plugin: 'kotlin-kapt'
```

Or you can apply it using the plugins DSL:

```
plugins {
    id "org.jetbrains.kotlin.kapt" version "1.3.10"
}
```

Then add the respective dependencies using the `kapt` configuration in your `dependencies` block:

```
dependencies {
    kapt 'groupId:artifactId:version'
}
```

PlacemarkModel

```
package org.wit.placemark.models

import android.os.Parcelable
import androidx.room.Entity
import androidx.room.PrimaryKey
import kotlinx.android.parcel.Parcelize

@Parcelize
@Entity
data class PlacemarkModel(@PrimaryKey(autoGenerate = true) var id: Long = 0,
                          var title: String = "",
                          var description: String = "",
                          var image: String = "",
                          var lat : Double = 0.0,
                          var lng: Double = 0.0,
                          var zoom: Float = 0f) : Parcelable
```

We have included 2 additional annotations:

- @Entity
- @PrimaryKey

These annotations will enable PlacemarkModel objects to be stored in a Room database.

Mark class as an @Entity - it can be stored in a database

```
@Parcelize
@Entity
data class PlacemarkModel(@PrimaryKey(autoGenerate = true) var id: Long = 0,
                          var title: String = "",
                          var description: String = "",
                          var image: String = "",
                          var lat : Double = 0.0,
                          var lng: Double = 0.0,
                          var zoom: Float = 0f) : Parcelable
```

Mark id @PrimaryKey + have it autoGenerated by db

```
@Dao
interface PlacemarkDao {

  @Insert(onConflict = OnConflictStrategy.REPLACE)
  fun create(placemark: PlacemarkModel)

  @Query("SELECT * FROM PlacemarkModel")
  fun findAll(): List<PlacemarkModel>

  @Update
  fun update(placemark: PlacemarkModel)
}
```

Defines an Interface to the PlacemarkTable

The implementation of this interface is generated by the rooms libraries

```
@Dao
interface PlacemarkDao {

  @Insert(onConflict = OnConflictStrategy.REPLACE)
  fun create(placemark: PlacemarkModel)

  @Query("SELECT * FROM PlacemarkModel")
  fun findAll(): List<PlacemarkModel>

  @Update
  fun update(placemark: PlacemarkModel)
}
```

Create a placemark (replace if id already exists)

```
@Dao
interface PlacemarkDao {

  @Insert(onConflict = OnConflictStrategy.REPLACE)
  fun create(placemark: PlacemarkModel)

  @Query("SELECT * FROM PlacemarkModel")
  fun findAll(): List<PlacemarkModel>

  @Update
  fun update(placemark: PlacemarkModel)
}
```

Get a List of all Placemarks

```
@Dao
interface PlacemarkDao {

  @Insert(onConflict = OnConflictStrategy.REPLACE)
  fun create(placemark: PlacemarkModel)

  @Query("SELECT * FROM PlacemarkModel")
  fun findAll(): List<PlacemarkModel>

  @Update
  fun update(placemark: PlacemarkModel)
}
```

Update an existing Placemark

PlacemarkDao

```
package org.wit.placemark.room

import androidx.room.*
import org.wit.placemark.models.PlacemarkModel

@Dao
interface PlacemarkDao {

  @Insert(onConflict = OnConflictStrategy.REPLACE)
  fun create(placemark: PlacemarkModel)

  @Query("SELECT * FROM PlacemarkModel")
  fun findAll(): List<PlacemarkModel>

  @Query("select * from PlacemarkModel where id = :id")
  fun findById(id: Long): PlacemarkModel

  @Update
  fun update(placemark: PlacemarkModel)

  @Delete
  fun deletePlacemark(placemark: PlacemarkModel)
}
```

Database

```
package org.wit.placemark.room

import androidx.room.Database
import androidx.room.RoomDatabase
import org.wit.placemark.models.PlacemarkModel

@Database(entities = arrayOf(PlacemarkModel::class), version = 1)
abstract class Database : RoomDatabase() {

  abstract fun placemarkDao(): PlacemarkDao
}
```

Interface to Entier Database

Database version number

Database

```
package org.wit.placemark.room

import androidx.room.Database
import androidx.room.RoomDatabase
import org.wit.placemark.models.PlacemarkModel

@Database(entities = arrayOf(PlacemarkModel::class), version = 1)
abstract class Database : RoomDatabase() {

  abstract fun placemarkDao(): PlacemarkDao
}
```

Provide access to all Dao objects (only one so far)

If structure of database changes (new fields etc, this number can be increased

PlacemarkStoreRoom

```
class PlacemarkStoreRoom(val context: Context) : PlacemarkStore {

  var dao: PlacemarkDao

  init {
    val database = Room.databaseBuilder(context, Database::class.java, "room_sample.db")
        .fallbackToDestructiveMigration()
        .build()
    dao = database.placemarkDao()
  }

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

  override fun create(placemark: PlacemarkModel) {
    dao.create(placemark)
  }

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

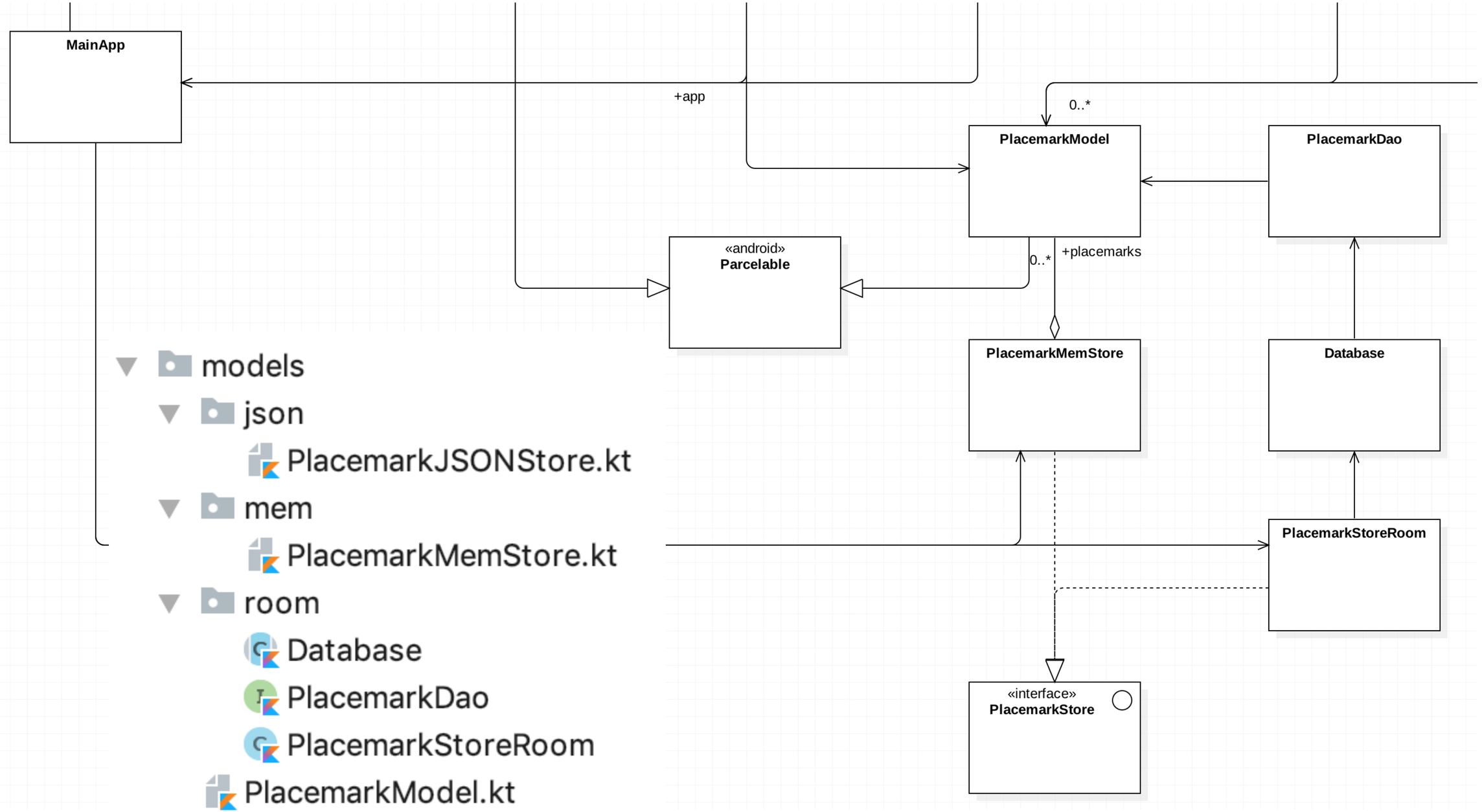
PlacemarkStoreRoom

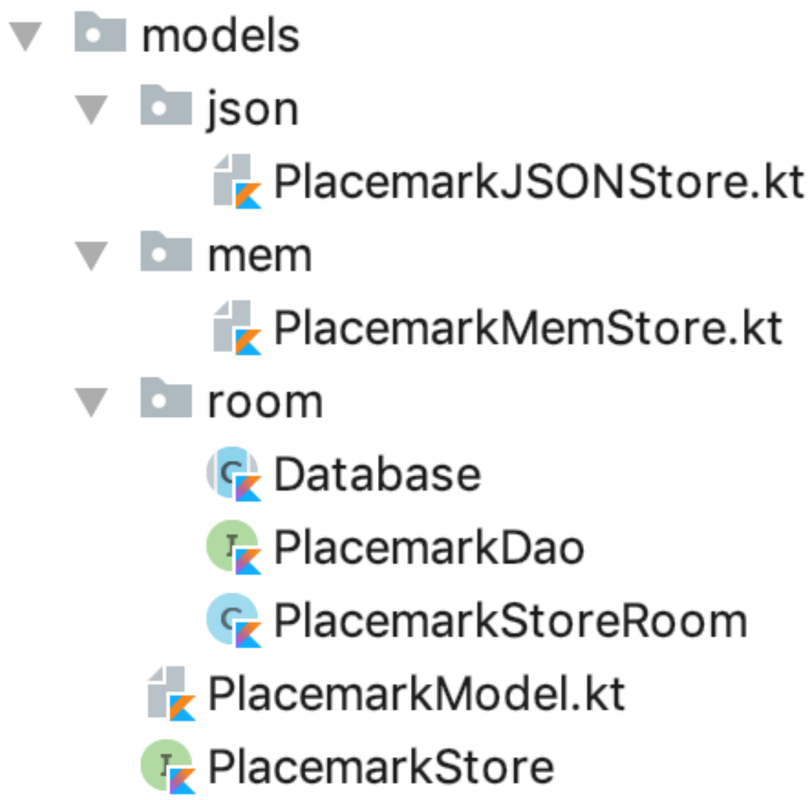
```
class PlacemarkStoreRoom(val context: Context) : PlacemarkStore {

  var dao: PlacemarkDao

  init {
    val database = Room.databaseBuilder(context, Database::class.java, "room_sample.db")
        .fallbackToDestructiveMigration()
        .build()
    dao = database.placemarkDao()
  }

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

  override fun create(placemark: PlacemarkModel) {
    dao.create(placemark)
  }

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

Create a Database object

Request a dao object from the database

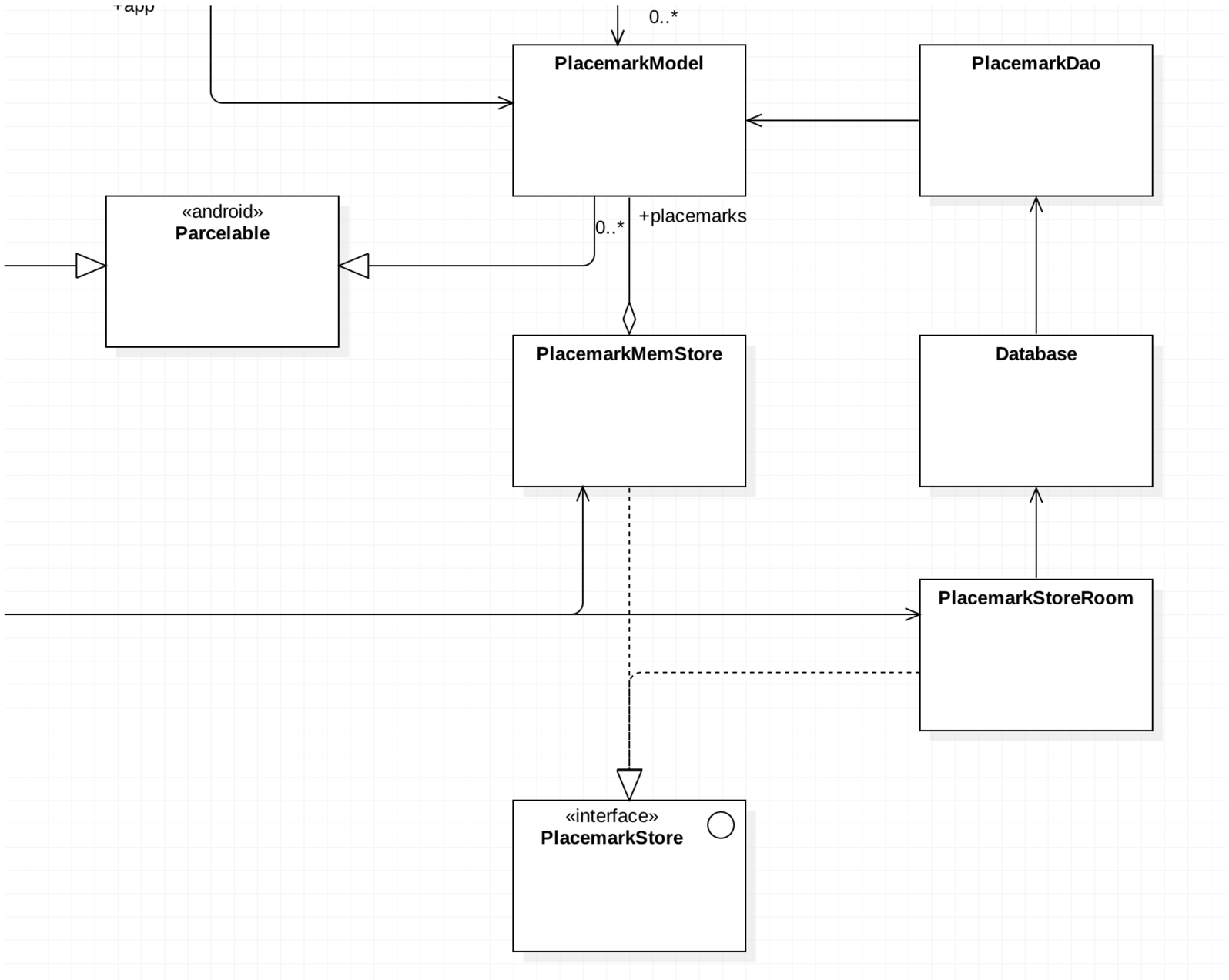
Use the dao to implement all PlacemarkStore features

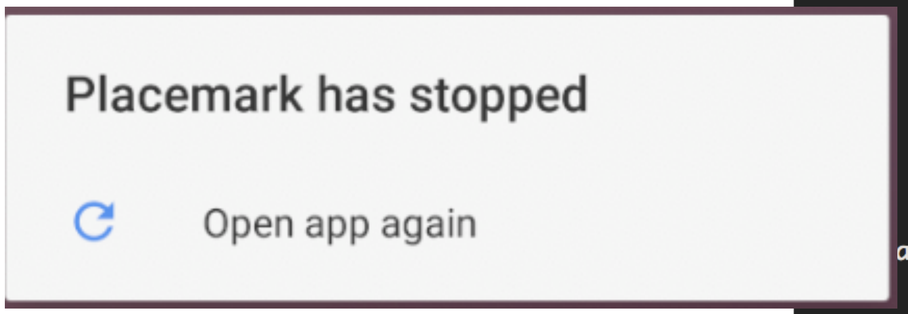

MainApp
+app
0..*
PlacemarkModel
PlacemarkDao
«android»
Parcelable
0..*
+placemarks
PlacemarkMemStore
Database
PlacemarkStoreRoom
«interface»
PlacemarkStore


models
json
PlacemarkJSONStore.kt
mem
PlacemarkMemStore.kt
room
Database
PlacemarkDao
PlacemarkStoreRoom
PlacemarkModel.kt
PlacemarkStore

Switch between in-memory and database placemarks

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

  lateinit var placemarks: PlacemarkStore

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


+app
0..*
PlacemarkModel
PlacemarkDao
«android»
Parcelable
0..*
+placemarks
PlacemarkMemStore
Database
PlacemarkStoreRoom
«interface»
PlacemarkStore




```
2019-11-14 10:54:45.126 13877-13877/org.wit.placemark E/AndroidRuntime: FATAL EXCEPTION: ma
    Process: org.wit.placemark, PID: 13877
    java.lang.RuntimeException: Unable to start activity ComponentInfo{org.wit.placemark/or
        at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:2665)
        at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:2726)
        at android.app.ActivityThread.-wrap12(ActivityThread.java)
        at android.app.ActivityThread$H.handleMessage(ActivityThread.java:1477)
        at android.os.Handler.dispatchMessage(Handler.java:102)
        at android.os.Looper.loop(Looper.java:154)
        at android.app.ActivityThread.main(ActivityThread.java:6119)
        at java.lang.reflect.Method.invoke(Native Method)
        at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:886)
        at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:776)
     aused by: java.lang.IllegalStateException: Cannot access database on the main thread
        at androidx.room.RoomDatabase.assertNotMainThread(RoomDatabase.java:261)
        at androidx.room.RoomDatabase.query(RoomDatabase.java:303)
        at androidx.room.util.DBUtil.query(DBUtil.java:54)
                                                   markDao_Impl.findAll(PlacemarkDao_Impl.java:143)
                                                   markStoreRoom.findAll(PlacemarkStoreRoom.kt:20)
                                                   emarklist.PlacemarkListPresenter.loadPlacemarks(Plac
                                                   emarklist.PlacemarkListView.onCreate(PlacemarkListVi
                                                   Create(Activity.java:6679)
                                                   callActivityOnCreate(Instrumentation.java:1118)
                                                   erformLaunchActivity(ActivityThread.java:2618)
                                                   andleLaunchActivity(ActivityThread.java:2726) 
                                                   wrap12(ActivityThread.java) 
                                                   .handleMessage(ActivityThread.java:1477) 
                                                   essage(Handler.java:102) 
        at android.os.Looper.loop(Looper.java:154) 
        at android.app.ActivityThread.main(ActivityThread.java:6119) 
        at java.lang.reflect.Method.invoke(Native Method) 
        at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:886)&
        at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:776) 
2019-11-14 10:54:45.518 1658-13224/? E/eglCodecCommon: glUtilsParamSize: unknow param 0x000
2019-11-14 10:54:45.518 1658-13224/? E/eglCodecCommon: glUtilsParamSize: unknow param 0x000
```

Caused by: java.lang.IllegalStateException: Cannot access database on the main thread since it may potentially lock the UI for a long period of time.

This version is terminated by Android

Cannot access database on the main thread

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

```kotlin
class PlacemarkMemStore : PlacemarkStore, AnkoLogger {

  val placemarks = ArrayList<PlacemarkModel>()

  suspend 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
    }
  }

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

Store - In-memory Implementation

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

```
class PlacemarkStoreRoom(val context: Context) : PlacemarkStore {

  var dao: PlacemarkDao

  init {
    val database = Room.databaseBuilder(context, Database::class.java, "room_sample.db")
        .fallbackToDestructiveMigration()
        .build()
    dao = database.placemarkDao()
  }

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

  override fun create(placemark: PlacemarkModel) {
    dao.create(placemark)
  }

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

Any of these database calls will trigger Termination of app

Store - Database Implementation

PlacemarkListPresenter

```
class PlacemarkListPresenter(view: BaseView) : BasePresenter(view) {

  fun doAddPlacemark() {
    view?.navigateTo(VIEW.PLACEMARK)
  }

  fun doEditPlacemark(placemark: PlacemarkModel) {
    view?.navigateTo(VIEW.PLACEMARK, 0, "placemark_edit", placemark)
  }

  fun doShowPlacemarksMap() {
    view?.navigateTo(VIEW.MAPS)
  }

  fun loadPlacemarks() {
    val placemarks = app.placemarks.findAll()
  }
}
```

Back up the call chain - this is a the call from a Presenter

How to use Anko's doAsync for background task

Aug 1, 2018

I wrote a series of post on how to handle background processing in an Android app. You don't really need to read them before you dive into this post, but it will give you context and more background about jank and how we use Threads, Handlers and AsyncTask to avoid it. The three previous post about jank are the following.

1. Android Jank. Running codes in the background using Java Threads
2. Android Threads, Handlers and Messages. Doing background work using Handlers and Messages, unlike Threads, Handler's and Messages are part of the Android Framework (not part of Java)
3. Android AsyncTask. Doing work in the background using the AsyncTask class. The AsyncTask, like Handlers and Messages, are also part of the Android Framework

https://workingdev.net/2018/08/android-using-ankos-doasync-to-do.html

```
doAsync {
  // do things in the background  // (1)
  activityUiThread {
    // make changes to the UI     // (2)
    textView.text = "Hello"
  }
}
```

In (1) we are in a backgrounf thread : safely make network requests, read/write large files, access database etc.

In (2), we can be confident that backround tasks have completed, and we can update UI accordingly

```
fun loadPlacemarks() {
  val placemarks = app.placemarks.findAll()
}
```

```
fun loadPlacemarks() {
  doAsync {
    val placemarks = app.placemarks.findAll()
    uiThread {
      view?.showPlacemarks(placemarks)
    }
  }
}
```

Background thread db access

Foreground (UI) Thread resumes - update View

PlacemarkPresenter

```
fun doAddOrSave(title: String, description: String) {
  placemark.title = title
  placemark.description = description
  doAsync {
    if (edit) {
      app.placemarks.update(placemark)
    } else {
      app.placemarks.create(placemark)
    }
    uiThread {
      view?.finish()
    }
  }
}
```

PlacemarkMapPresenter

```
fun doMarkerSelected(marker: Marker) {
  val tag = marker.tag as Long
  doAsync {
    val placemark = app.placemarks.findById(tag)
    uiThread {
      if (placemark != null) view?.showPlacemark(placemark)
    }
  }
}

fun loadPlacemarks() {
  doAsync {
    val placemarks = app.placemarks.findAll()
    uiThread {
      view?.showPlacemarks(placemarks)
    }
  }
}
```