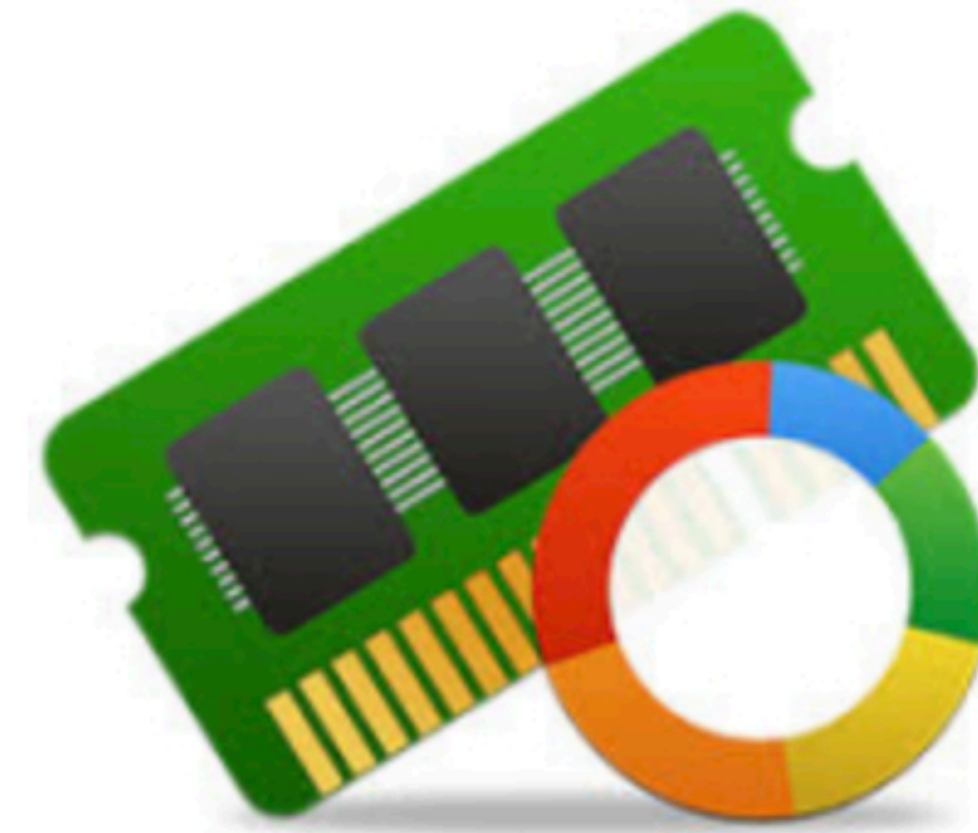
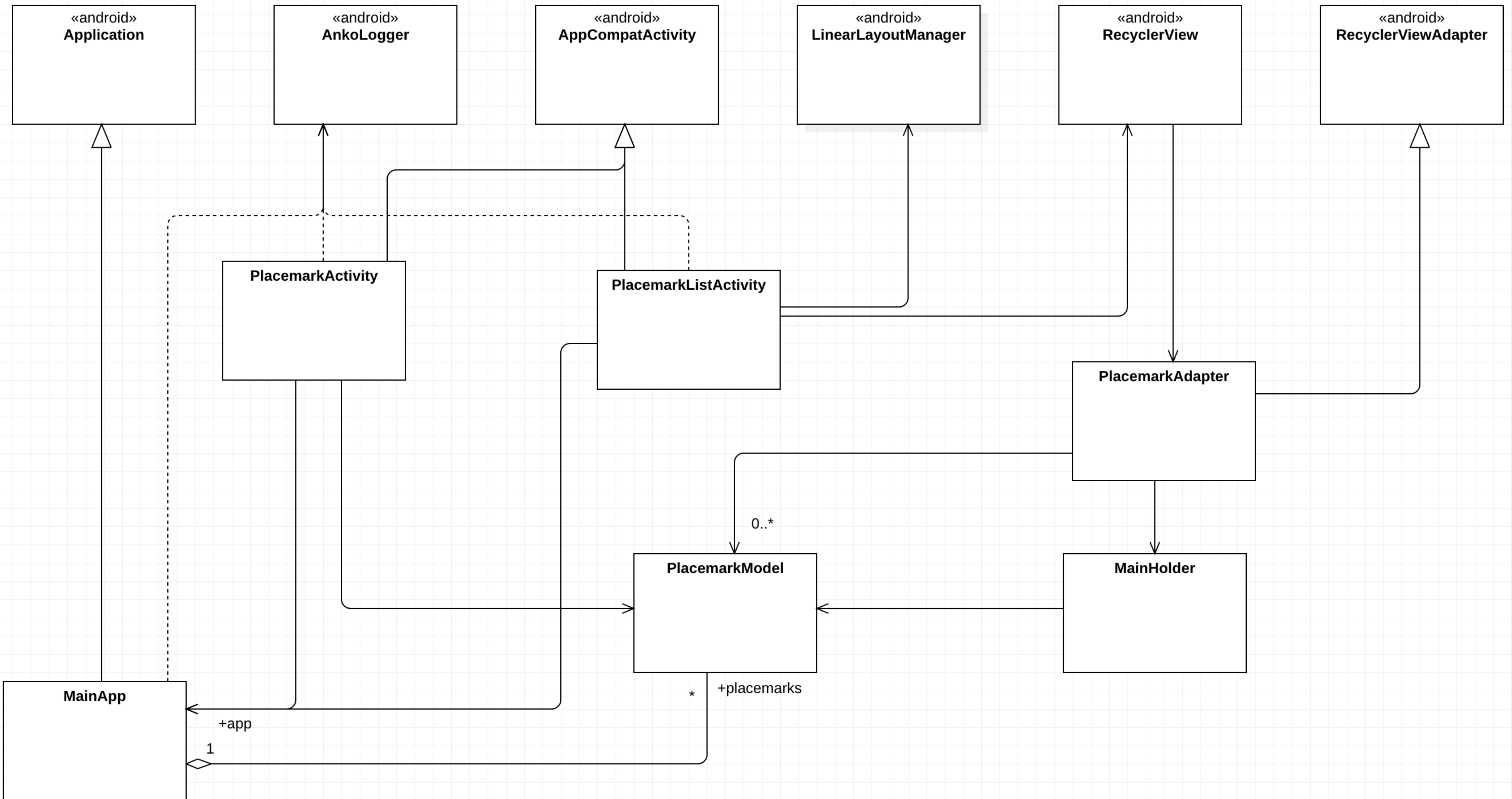


Memory Store

Memory Store



Abstract the Placemarks
data structure into
PlacemarkStore interface +
in-memory implementation.



PlacemarkStore

```
package org.wit.placemark.models

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

Interface defining an
abstract
PlacemarkStore *type*

PlacemarkMemStore

```
package org.wit.placemark.models

class PlacemarkMemStore : PlacemarkStore {
    val placemarks = ArrayList<PlacemarkModel>()

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

    override fun create(placemark: PlacemarkModel) {
        placemarks.add(placemark)
    }
}
```

Simple in-memory
implementation using
standard ArrayList
Data Structure

PlacemarkStore

```
package org.wit.placemark.models

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

PlacemarkMemStore

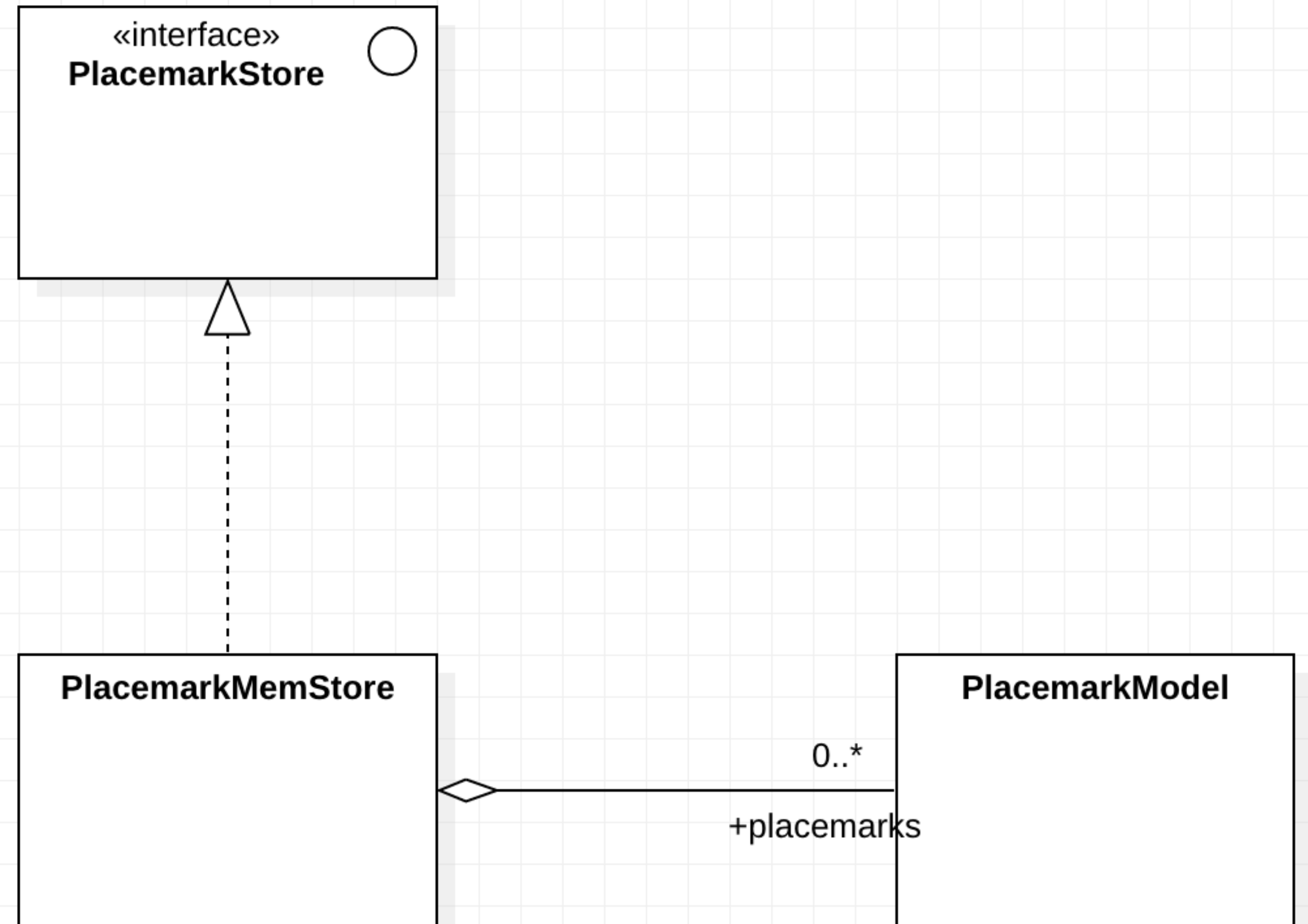
```
package org.wit.placemark.models

class PlacemarkMemStore : PlacemarkStore {

    val placemarks = ArrayList<PlacemarkModel>()

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

    override fun create(placemark: PlacemarkModel) {
        placemarks.add(placemark)
    }
}
```



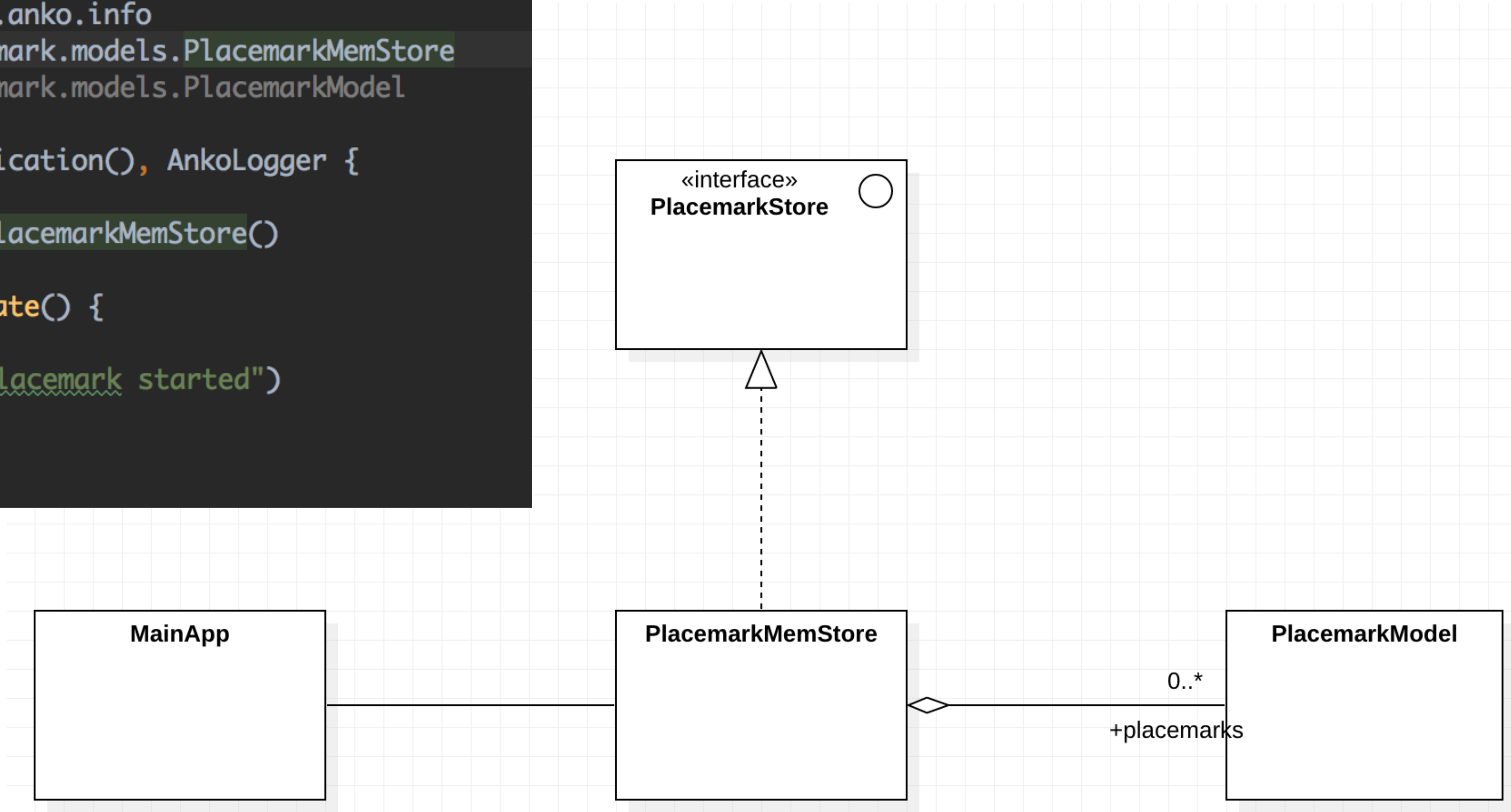
```
package org.wit.placemark.main

import android.app.Application
import org.jetbrains.anko.AnkoLogger
import org.jetbrains.anko.info
import org.wit.placemark.models.PlacemarkMemStore
import org.wit.placemark.models.PlacemarkModel

class MainApp : Application(), AnkoLogger {

    val placemarks = PlacemarkMemStore()

    override fun onCreate() {
        super.onCreate()
        info(message: "Placemark started")
    }
}
```



PlacemarkListActivity

```
// recyclerView.adapter = PlacemarkAdapter(app.placemarks)  
recyclerView.adapter = PlacemarkAdapter(app.placemarks.findAll())
```

PlacemarkActivity

```
// app.placemarks.add(placemark.copy())  
app.placemarks.create(placemark.copy())  
  
...  
// app.placemarks.forEach { info("add Button Pressed: ${it}") }  
app.placemarks.findAll().forEach{ info("add Button Pressed: ${it}") }  
...
```

Use PlacemarkStore
interface instead of
directly accessing
placemarks
ArrayList

