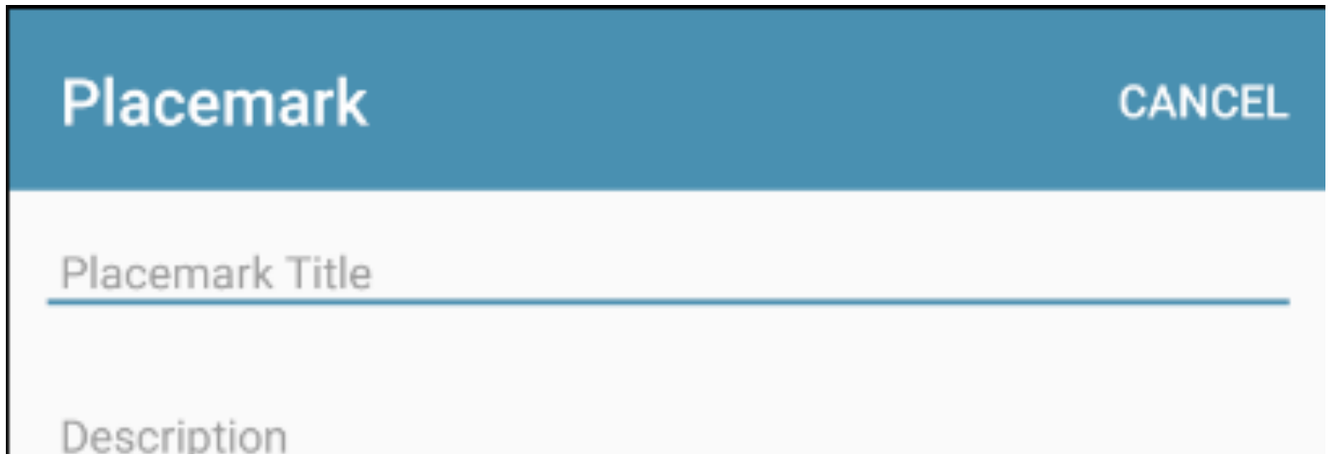
Exercise 1

Incorporate new 'Cancel' action into `PlacemarkActivity`. This should return to PlacemarkList without adding a new Placemark, Use the `RESULT_CANCELED` return code.



strings.xml

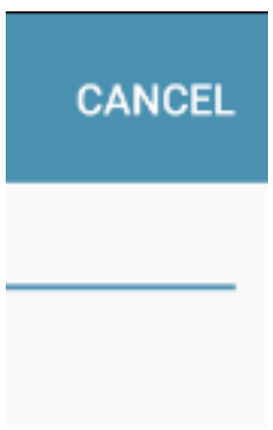
```
...
  <string name="menu_cancelPlacemark">Cancel</string>
...
```

menu_placemark.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto">

  <item
      android:id="@+id/item_cancel"
      android:title="@string/menu_cancelPlacemark"
      app:showAsAction="always"/>
</menu>
```

PlacemarkActivity

```
...
 override fun onCreate(savedInstanceState: Bundle?) {
    ...
    toolbarAdd.title = title
    setSupportActionBar(toolbarAdd)
    ...
    }
  }
...

  override fun onCreateOptionsMenu(menu: Menu?): Boolean {
    menuInflater.inflate(R.menu.menu_placemark, menu)
    return super.onCreateOptionsMenu(menu)
  }

  override fun onOptionsItemSelected(item: MenuItem?): Boolean {
    when (item?.itemId) {
      R.id.item_cancel -> {
        setResult(RESULT_CANCELED)
        finish()
      }
    }
    return super.onOptionsItemSelected(item)
  }
...
```

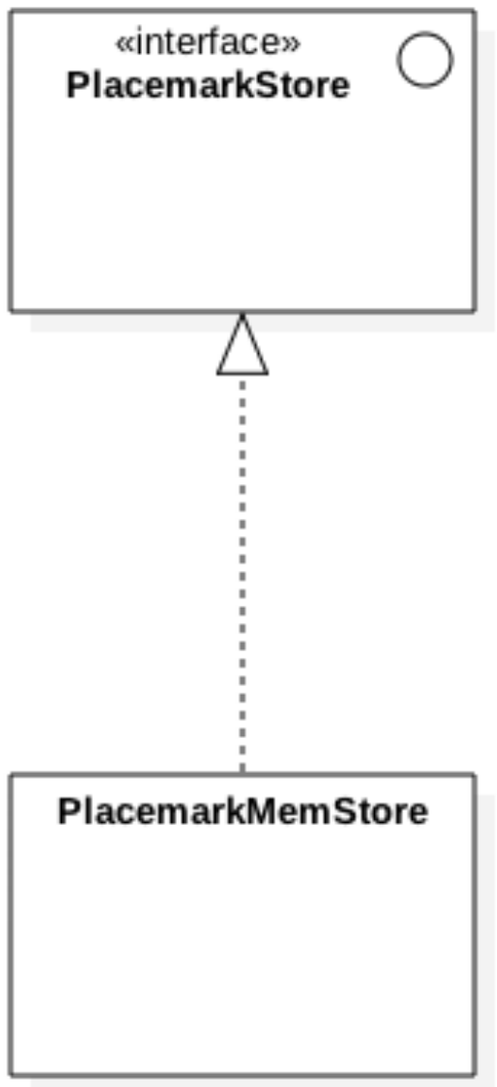


Exercise 2

Abstract & Encapsulate
Placemark storage and
management

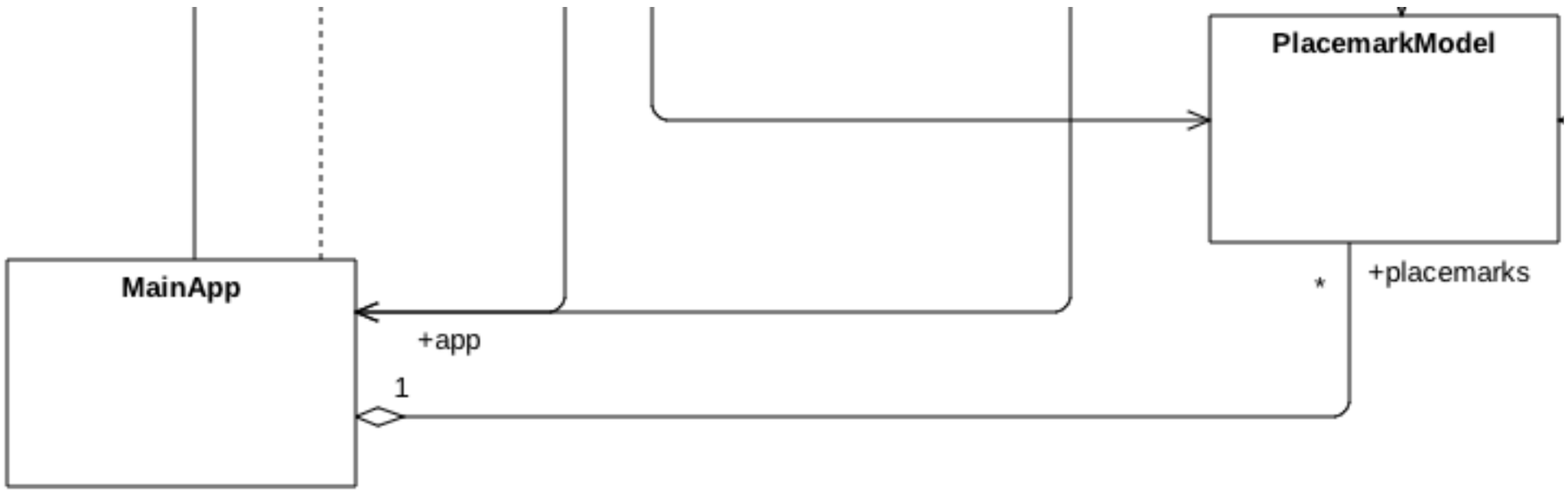
PlacemarkStore

```
package org.wit.placemark.models

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

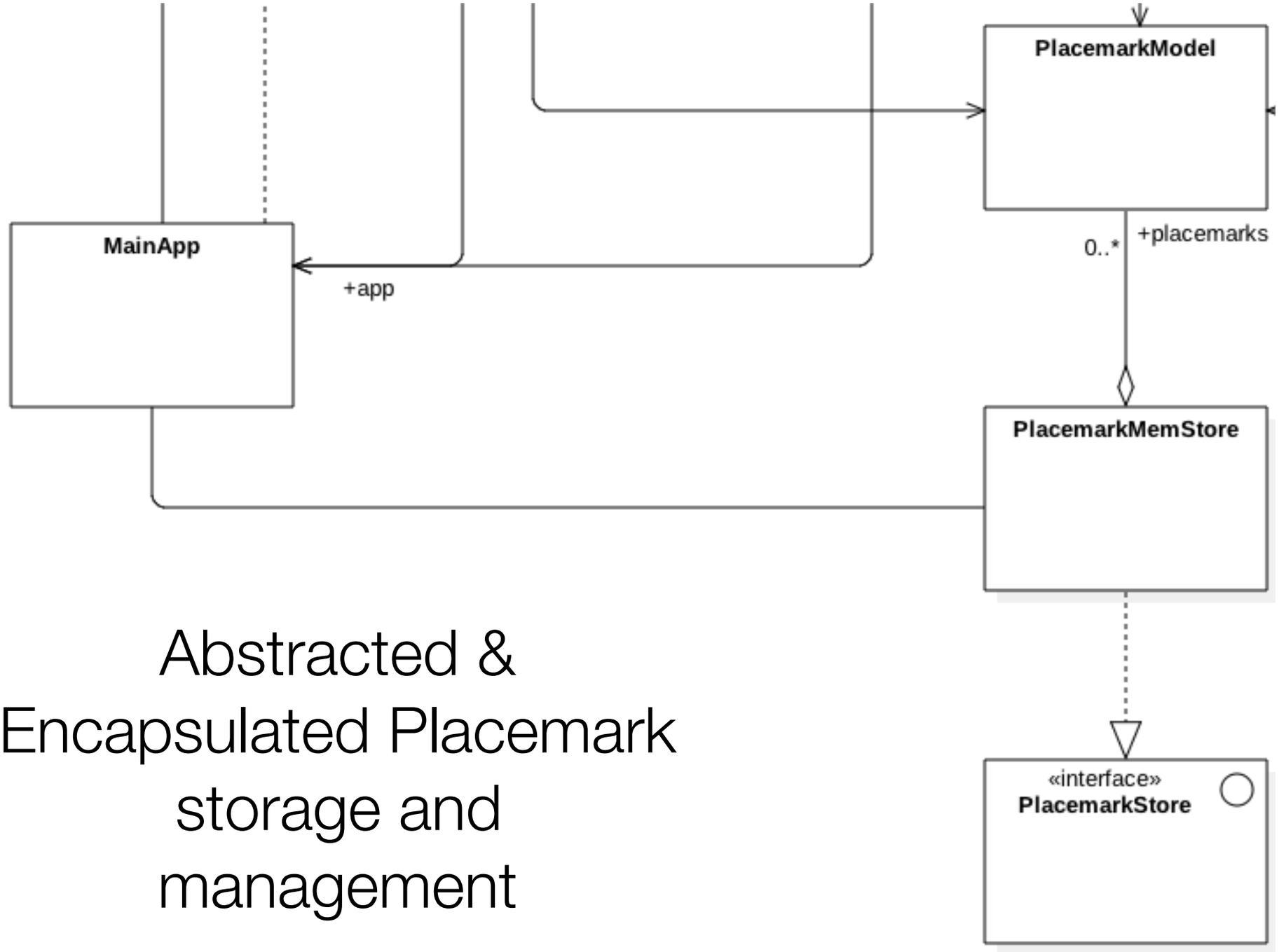


PlacemarkMemStore

```
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)
  }
}
```



Current Placemark
storage & management



Abstracted &
Encapsulated Placemark
storage and
management

MainApp

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

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}"
...
```

Exercise 4

Introduce a new method in PlacemarkMemStore which will log all placemarks. Make this an internal method (not in the interface). Call it whenever a new placemark is added.

PlacemarkMemStore

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

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

PlacemarkActivity

```
        // info("add Button Pressed: $placemarkTitle")
        // app.placemarks.findAll().forEach{ info("add Button Presse
```