# Parcelable

# Parcelable

build.gradle

```
androidExtensions {
  experimental = true
}
```

Enable advanced Kotlin features to simplify android patterns

## Kotlin Android Extensions

Edit Page

**Author** Yan Zhulanow

This tutorial describes how to use Kotlin Android Extensions to improve support for Android development.

View Binding ← Already using this feature

Parcelable ← We want to start using this

# Parcel

A Parcel is a message container. A message being data and object references. Parcel, like Parcelable, Intents, and Bundles are part of the IPC family in android. IPC stands for inter-process communication — ***it is Androids' framework for moving data from one component of an app to another component of the same app***.

https://medium.com/@rayacevedo45/android-parcel-and-parcelable-865c398d5053

# Parcel

**added in API level 1**

Summary: Fields | Methods | Protected Methods | Inherited Methods | [Expand All]

```
public final class Parcel
extends Object
```

java.lang.Object

↳ android.os.Parcel

---

Container for a message (data and object references) that can be sent through an IBinder. A Parcel can contain both flattened data that will be unflattened on the other side of the IPC (using the various methods here for writing specific types, or the general `Parcelable` interface), and references to live `IBinder` objects that will result in the other side receiving a proxy IBinder connected with the original IBinder in the Parcel.

# Parcelable

**added in API level 1**

Summary: Nested Classes | Constants | Methods | [Expand All]

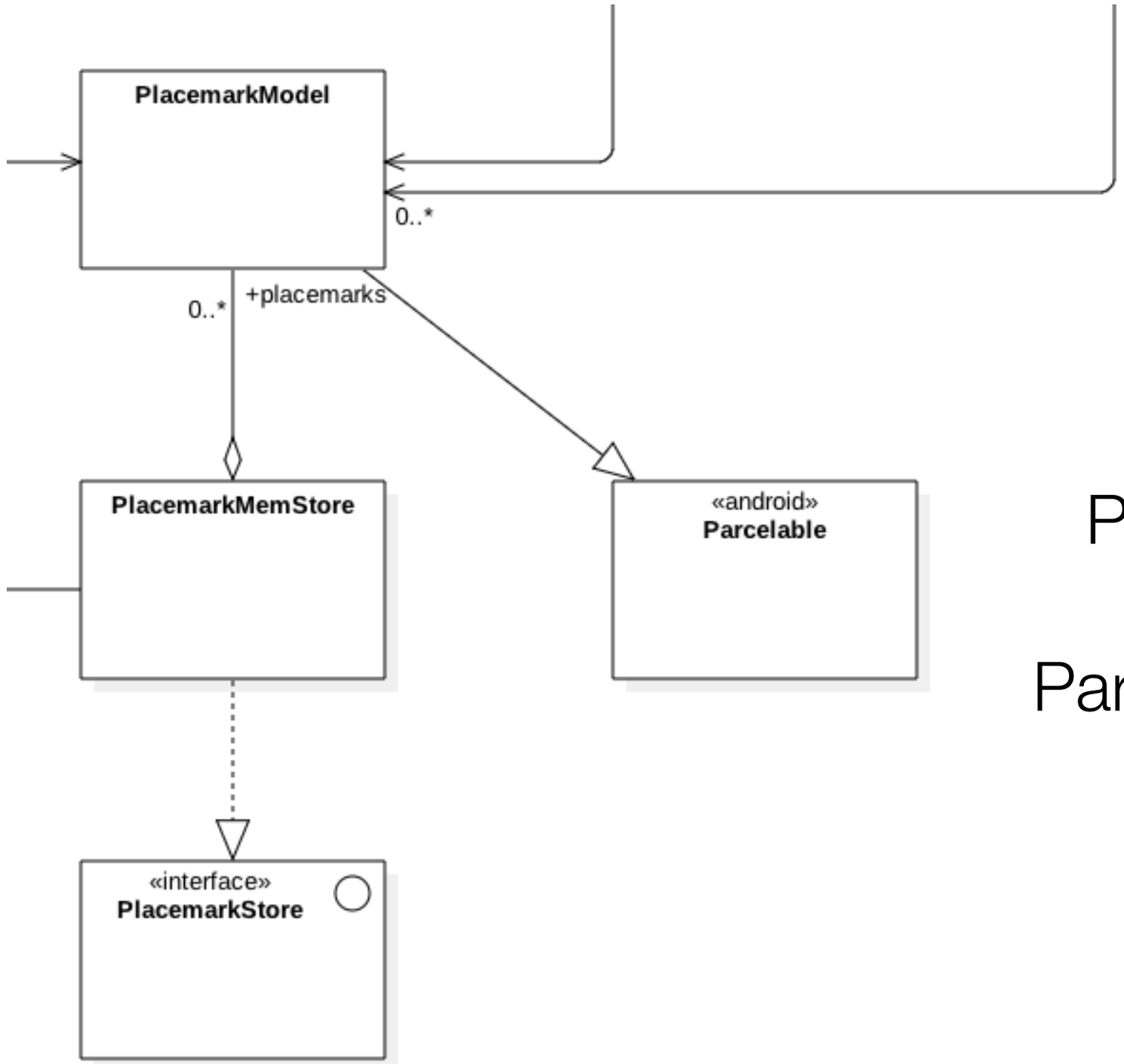
```
public interface Parcelable
```

android.os.Parcelable

Known Indirect Subclasses

AbsSavedState,AbsoluteSizeSpan,AccessibilityEvent,AccessibilityNodeInfo,AccessibilityServiceInfo,Acce
332 others.

Interface for classes whose instances can be written to and restored from a `Parcel`. Classes implementing the Parcelable interface must also have a non-null static field called `CREATOR` of a type that implements the `Parcelable.Creator` interface.




PlacemarkModel equipped with Parcelable capability

# Parcelable

```
@Parcelize
data class PlacemarkModel(var title: String = "",
                          var description: String = "") : Parcelable
```

"***Parcelabe***" equips our data class with Parcelize implementation

PlacemarkModel objects can now be passed between Activities

```
cemarkActivity
cemarkAdapter
cemarkListActivity
5
6  @Parcelize
7  data class PlacemarkModel(var title: String = "",
                                                    var description: String = "") : Parcelable
Class 'PlacemarkModel' is not abstract and does not implement abstract member
public abstract fun writeToParcel(p0: Parcel!, p1: Int): Unit defined in android.os.Parcelable
```

Still Experimental, so Studio may report errors

```
cemark (test)
cemark.activities
markActivity
cemark.main
4  import kotlinx.android.parcel.Parcelize
5
6  @Parcelize
7  data class PlacemarkModel(var title: String = "",
                                                    var description: String = "") : Parcelable
This class implements Parcelable but does not provide a CREATOR field more... (⌘F1)
```

# PlacemarkListActivity


Previously, we start PlacemarkActivity without passing any values to it

```
override fun onPlacemarkClick(placemark: PlacemarkModel) {
  startActivityForResult(intentFor<PlacemarkActivity>(), 200)
}
```

Revised to pass PlacemarkModel object

```
override fun onPlacemarkClick(placemark: PlacemarkModel) {
  startActivityForResult(intentFor<PlacemarkActivity>().putExtra("placemark_edit",
                                                                 placemark), 201)
}
```

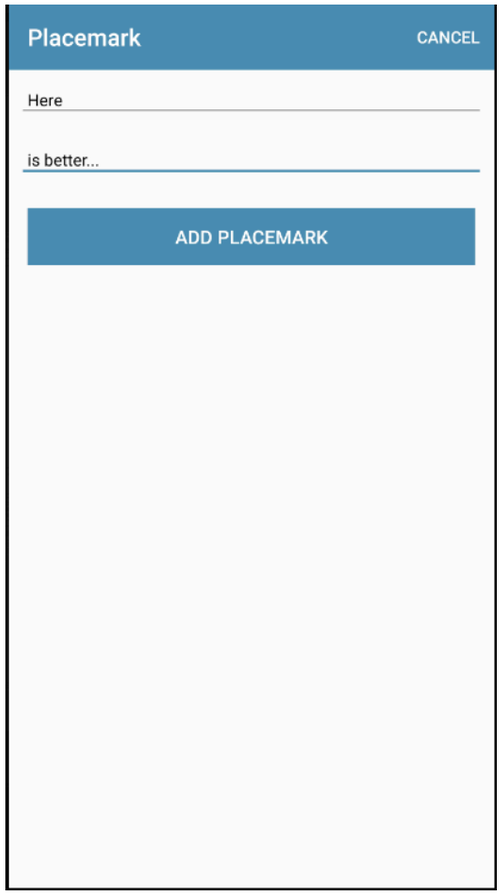
This is via ***putExtra*** method, which can send a Parcebable object to another activity

# PlacemarkActivity

```
override fun onCreate(savedInstanceState: Bundle?) {

  ...
  if (intent.hasExtra("placemark_edit")) {
    placemark = intent.extras.getParcelable<PlacemarkModel>("placemark_edit")
    placemarkTitle.setText(placemark.title)
    description.setText(placemark.description)
  }
  ...
}
```

In PlacemarkActivity, recover the placemark (if present), and update UI with pacemark values

(Look for ‘placemark_edit’ key injected by PlacemarkListActivity)



# IDs

```
@Parcelize
data class PlacemarkModel(var id: Long = 0,
                          var title: String = "",
                          var description: String = "") : Parcelable
```

PlacemarkModel objects need a unique ID if we are to manage them effectively

This ID can be used for update / delete methods in PlacemarkStore methods

Generate a unique ID

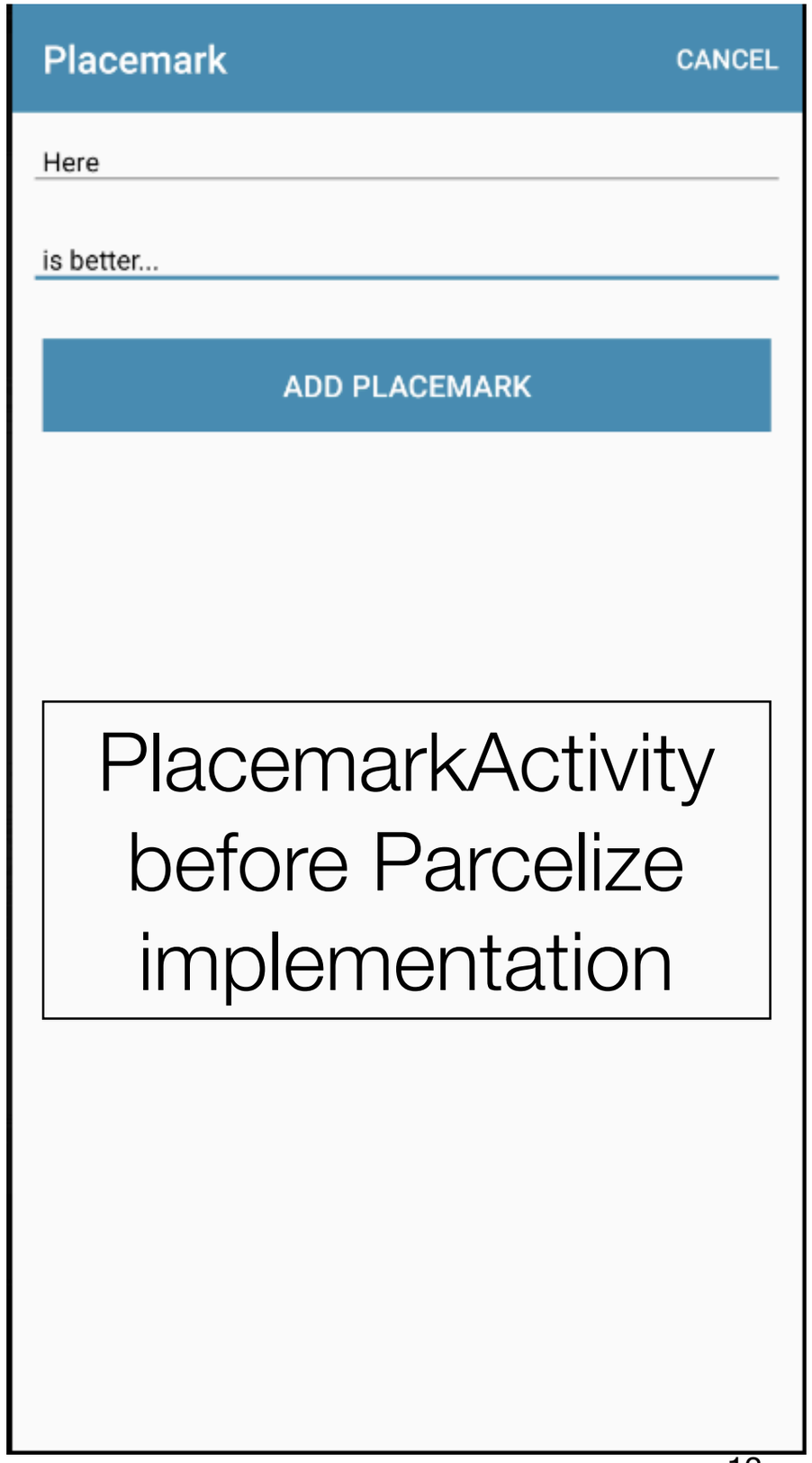
Insert ID into place mark before insertion

In Update method, find matching placemark and update its fields

```
var lastId = 0L

internal fun getId(): Long {
  return lastId++
}

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
    }
  }

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

```
class PlacemarkActivity : AppCompatActivity(), AnkoLogger {

  var placemark = PlacemarkModel()
  lateinit var app: MainApp

  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_placemark)
    app = application as MainApp

    toolbarAdd.title = title
    setSupportActionBar(toolbarAdd)

    btnAdd.setOnClickListener() {
      placemark.title = placemarkTitle.text.toString()
      placemark.description = description.text.toString()
      if (placemark.title.isNotEmpty()) {
        app.placemarks.create(placemark.copy())
        setResult(AppCompatActivity.RESULT_OK)
        finish()
      }
      else {
        toast("Please Enter a title")
      }
    }
  }

  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)
  }
}
```

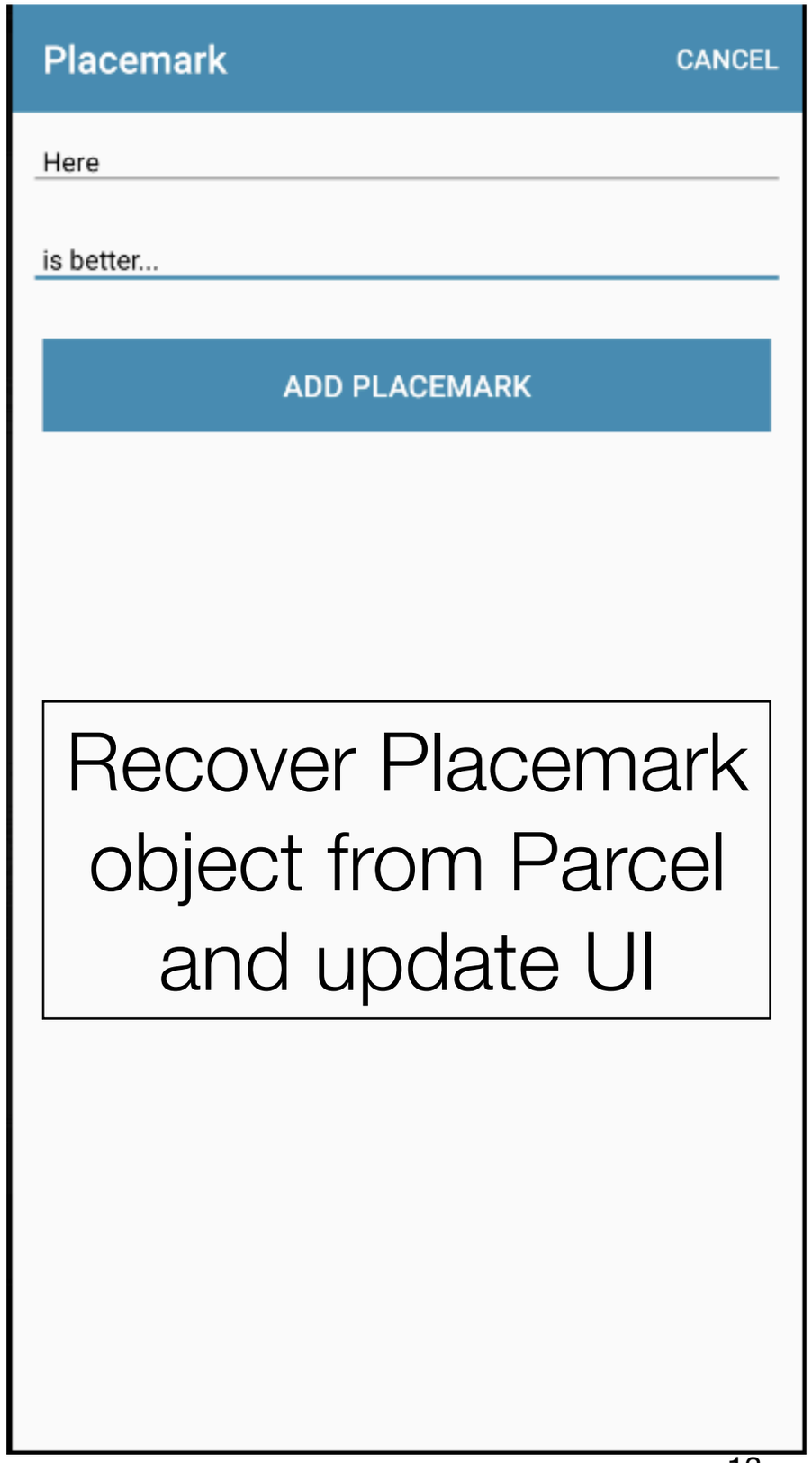
# PlacemarkActivity




PlacemarkActivity before Parcelize implementation

# PlacemarkActivity

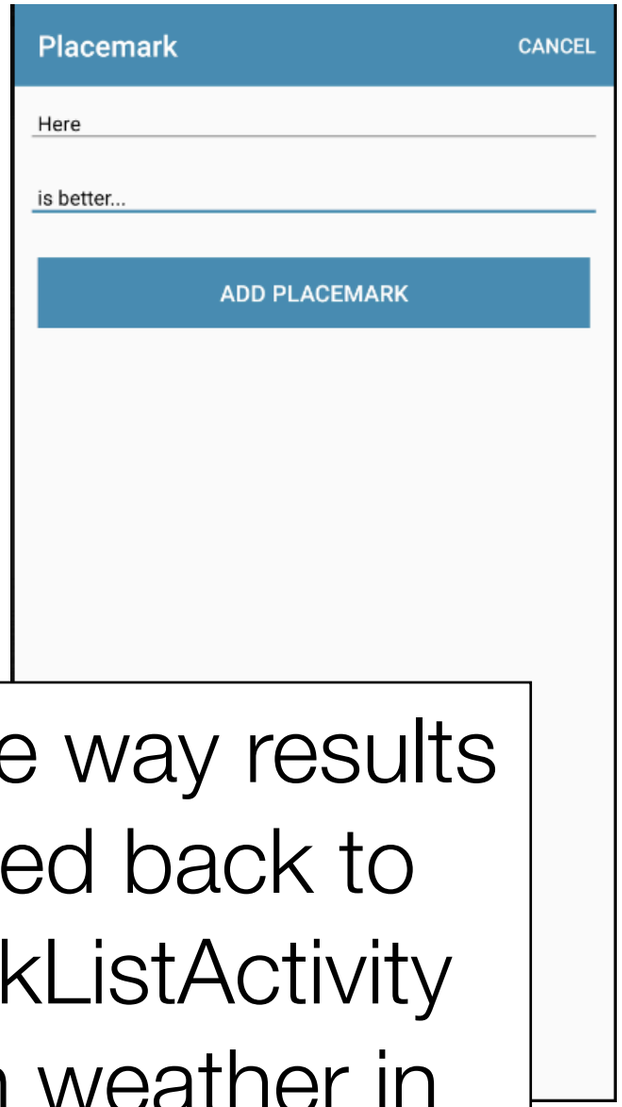
```
class PlacemarkActivity : AppCompatActivity(), AnkoLogger {

  var placemark = PlacemarkModel()
  lateinit var app: MainApp

  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_placemark)
    app = application as MainApp

    toolbarAdd.title = title
    setSupportActionBar(toolbarAdd)

    if (intent.hasExtra("placemark_edit")) {
      placemark = intent.extras.getParcelable<PlacemarkModel>("placemark_edit")
      placemarkTitle.setText(placemark.title)
      description.setText(placemark.description)
    }

    btnAdd.setOnClickListener() {
      placemark.title = placemarkTitle.text.toString()
      placemark.description = description.text.toString()
      if (placemark.title.isNotEmpty()) {
        app.placemarks.create(placemark.copy())
        setResult(AppCompatActivity.RESULT_OK)
        finish()
      }
      else {
        toast("Please Enter a title")
      }
    }
  }

  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)
  }
}
```




Recover Placemark object from Parcel and update UI

# PlacemarkActivity

```
class PlacemarkActivity : AppCompatActivity(), AnkoLogger {

  var placemark = PlacemarkModel()
  lateinit var app: MainApp
  var edit = false

  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_placemark)
    app = application as MainApp

    toolbarAdd.title = title
    setSupportActionBar(toolbarAdd)

    if (intent.hasExtra("placemark_edit")) {
      edit = true
      btnAdd.setText(R.string.save_placemark)
      placemark = intent.extras.getParcelable<PlacemarkModel>("placemark_edit")
      placemarkTitle.setText(placemark.title)
      description.setText(placemark.description)
    }

    btnAdd.setOnClickListener() {
      placemark.title = placemarkTitle.text.toString()
      placemark.description = description.text.toString()

      if (edit) {
        app.placemarks.update(placemark.copy())
        setResult(201)
        finish()
      }
      else {
        if (placemark.title.isNotEmpty()) {
          app.placemarks.create(placemark.copy())
          setResult(200)
          finish()
        }
        else {
          toast(R.string.enter_placemark_title)
        }
      }
    }
  }

...
}
```




Change the way results are passed back to PlacemarkListActivity based on weather in Edit mode

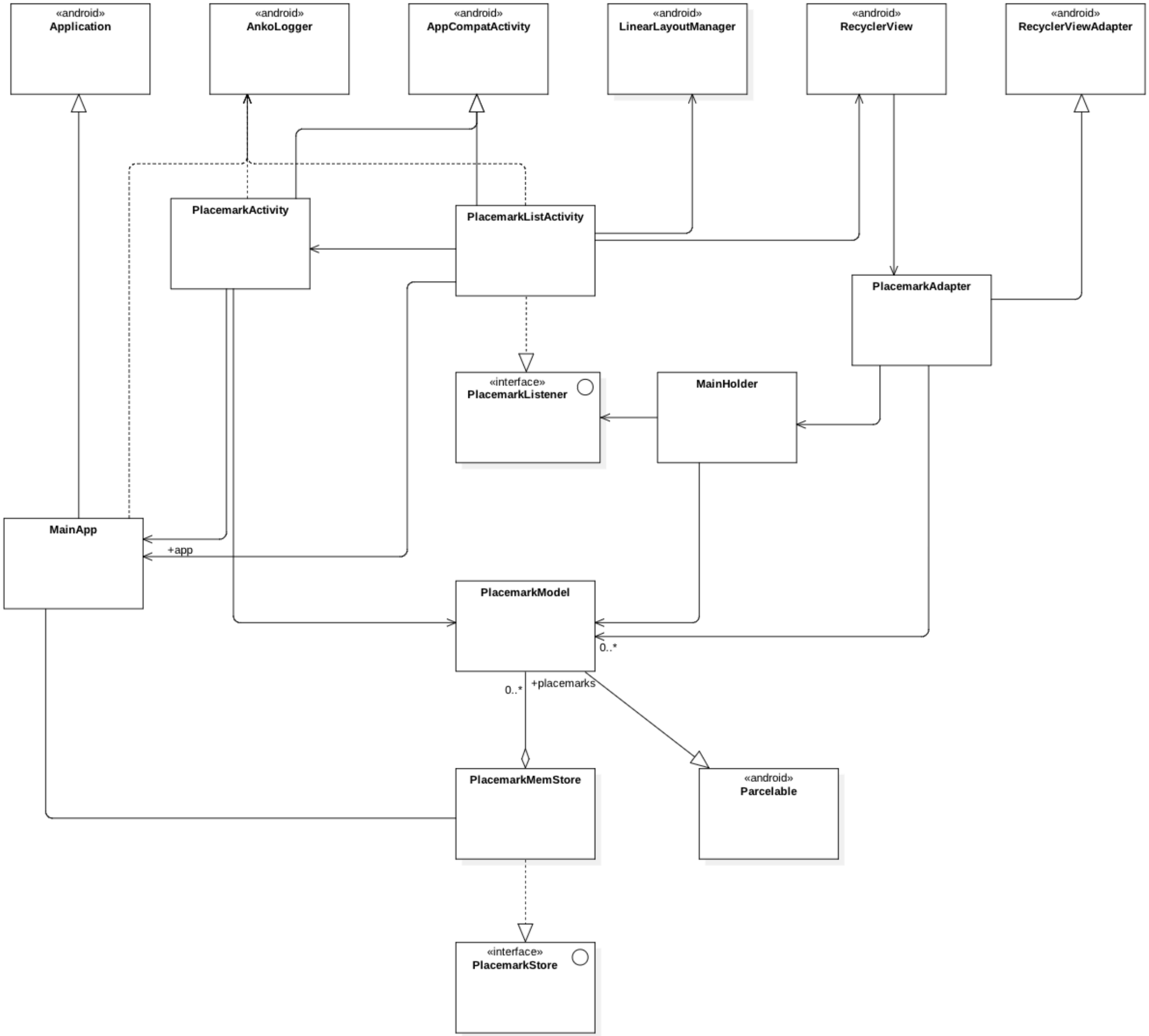
# PlacemarkActivity

If pacemark passed to activity, set edit mode to true

```
var edit = false
...
override fun onCreate(savedInstanceState: Bundle?) {

  if (intent.hasExtra("placemark_edit")) {
    edit = true
    btnAdd.setText(R.string.save_placemark)
    placemark = intent.extras.getParcelable<PlacemarkModel>("placemark_edit")
    placemarkTitle.setText(placemark.title)
    description.setText(placemark.description)
  }
```

If edit mode when button pressed, update existing placemark Otherwise, create new placemark

```
  btnAdd.setOnClickListener() {
    placemark.title = placemarkTitle.text.toString()
    placemark.description = description.text.toString()

    if (edit) {
      app.placemarks.update(placemark.copy())
      setResult(201)
      finish()
    }
    else {
      if (placemark.title.isNotEmpty()) {
        app.placemarks.create(placemark.copy())
        setResult(200)
        finish()
      }
      else {
        toast(R.string.enter_placemark_title)
      }
    }
  }
```


«android»
Application
«android»
AnkoLogger
«android»
AppCompatActivity
«android»
LinearLayoutManager
«android»
RecyclerView
«android»
RecyclerViewAdapter
PlacemarkActivity
PlacemarkListActivity
PlacemarkAdapter
«interface»
PlacemarkListener
MainHolder
MainApp
+app
PlacemarkModel
0..*
0..*
+placemarks
PlacemarkMemStore
«android»
Parcelable
«interface»
PlacemarkStore