

Parcelable

Parcelable



Encapsulate model data for
transmission between
Activities

Kotlin Android Extensions

build.gradle

```
apply plugin: 'kotlin-android-extensions'

androidExtensions {
    experimental = true
}
```

Enable advanced Kotlin
features to simplify android
patterns

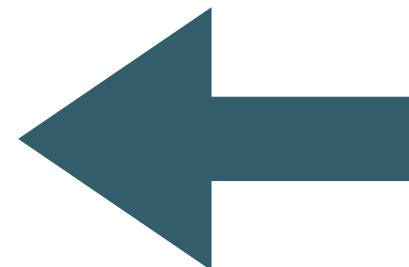
Kotlin Android Extensions

 Edit Page

<https://kotlinlang.org/docs/tutorials/android-plugin.html>

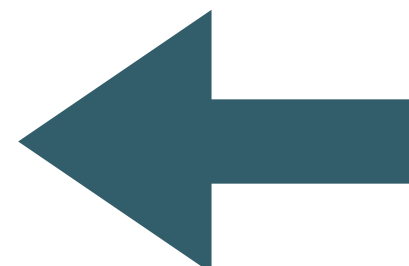
This tutorial describes how to use Kotlin Android Extensions to improve support for Android development.

View Binding



Already using this feature

Parcelable



We want to start using this

View Binding

Background

Every Android developer knows well the `findViewById()` function. It is, without a doubt, a source of potential bugs and nasty code which is hard to read and support. While there are several libraries available that provide solutions to this problem, those libraries require annotating fields for each exposed `View`.

The Kotlin Android Extensions plugin allows us to obtain the same experience we have with some of these libraries, without having to add any extra code.

In essence, this allows for the following code:

```
// Using R.layout.activity_main from the 'main' source set
import kotlinx.android.synthetic.main.activity_main.*

class MyActivity : Activity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        // Instead of findViewById<TextView>(R.id.textView)
        textView.setText("Hello, world!")
    }
}
```

`textView` is an extension property for `Activity`, and it has the same type as declared in `activity_main.xml` (so it is a `TextView`).

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>

added in API level 1

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

Parcel

```
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]

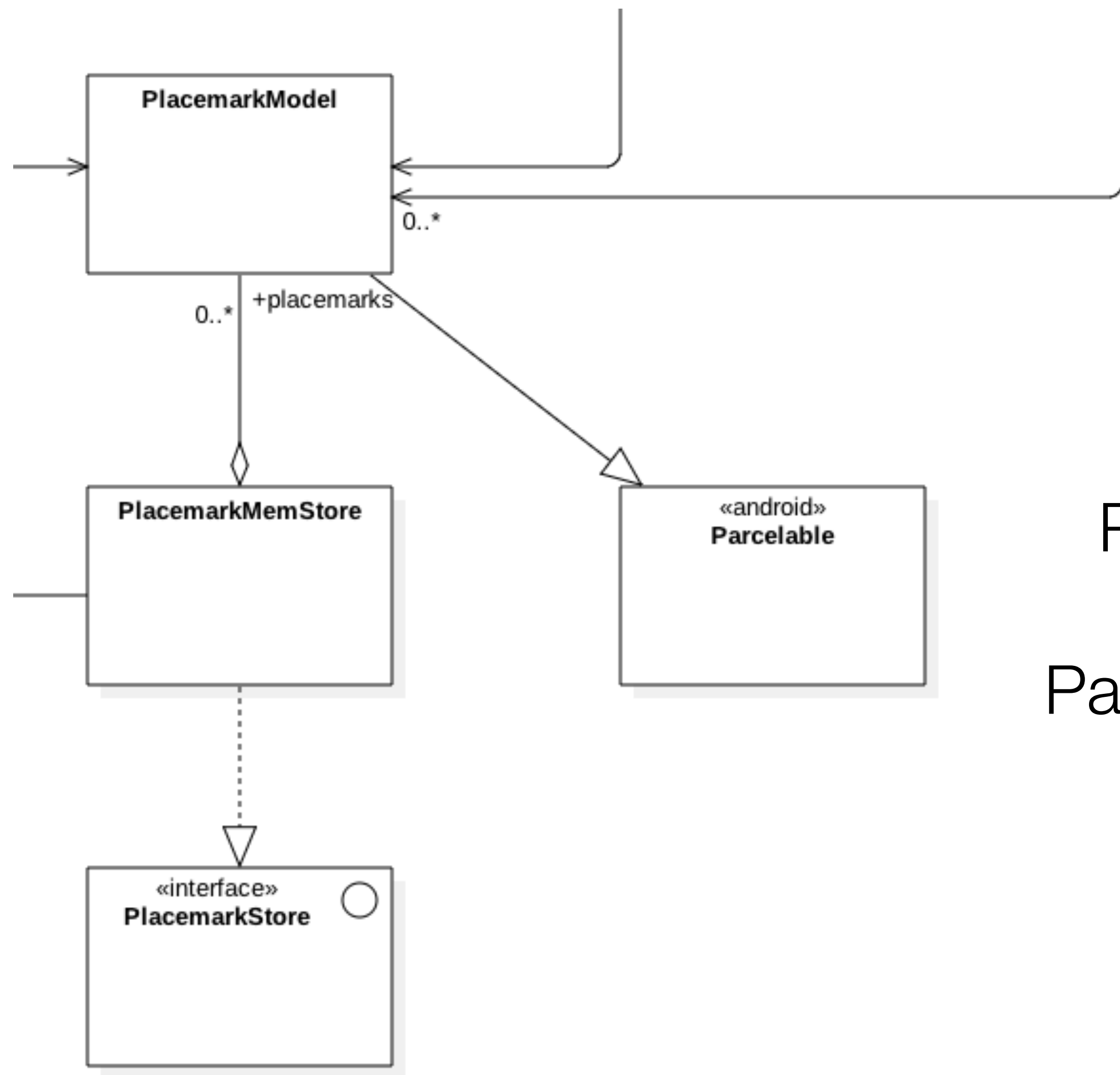
public interface Parcelable

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

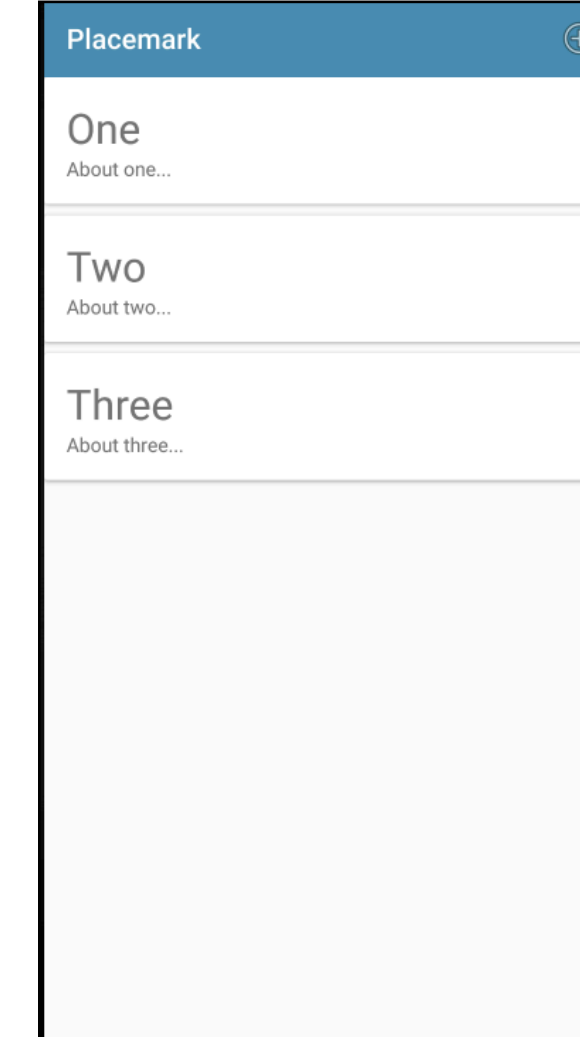
```
@Parcelize  
data class PlacemarkModel(var title: String = "",  
                           var description: String = "") : Parcelable
```

“***Parcelabe***” equips our data class with Parcelize
implementation

PlacemarkModel objects can now be passed
between Activities

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 Parcelable 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 placemark
values

(Look for 'placemark_edit' key injected by
PlacemarkListActivity)



Placemark CANCEL

Here

is better...

ADD PLACEMARK

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

```

PlacemarkActivity

Placemark
CANCEL

Here

is better...

ADD PLACEMARK

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

Placemark

CANCEL

Here

is better...

ADD PLACEMARK

Recover Placemark
object from Parcel
and update UI

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)
        toolbarAdd.title = title
        setSupportActionBar(toolbarAdd)
        info("Placemark Activity started..")
    }
```

```
    app = application as MainApp
    var edit = false
```

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

```
    btnAdd.setOnClickListener() {
        placemark.title = placemarkTitle.text.toString()
        placemark.description = description.text.toString()
        if (placemark.title.isEmpty()) {
            toast(R.string.enter_placemark_title)
        } else {
```

```
            if (edit) {
                app.placemarks.update(placemark.copy())
            } else {
                app.placemarks.create(placemark.copy())
            }
        }
```

```
        info("add Button Pressed: $placemarkTitle")
        setResult(AppCompatActivity.RESULT_OK)
        finish()
    }
}
```

The screenshot shows a mobile application interface for adding or editing a placemark. At the top, there is a blue header bar with the text "Placemark" on the left and a "CANCEL" button on the right. Below the header, there are two text input fields. The first field contains the text "Here" and the second field contains the text "is better...". Below these fields, there is a blue button with the text "ADD PLACEMARK".

Change the behaviour
based on weather in
Edit mode

PlacemarkActivity

If placemark
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 (placemark.title.isEmpty()) {
        toast(R.string.enter_placemark_title)
    } else {
        if (edit) {
            app.placemarks.update(placemark.copy())
        } else {
            app.placemarks.create(placemark.copy())
        }
    }
    info("add Button Pressed: $placemarkTitle")
    setResult(AppCompatActivity.RESULT_OK)
    finish()
}
```

