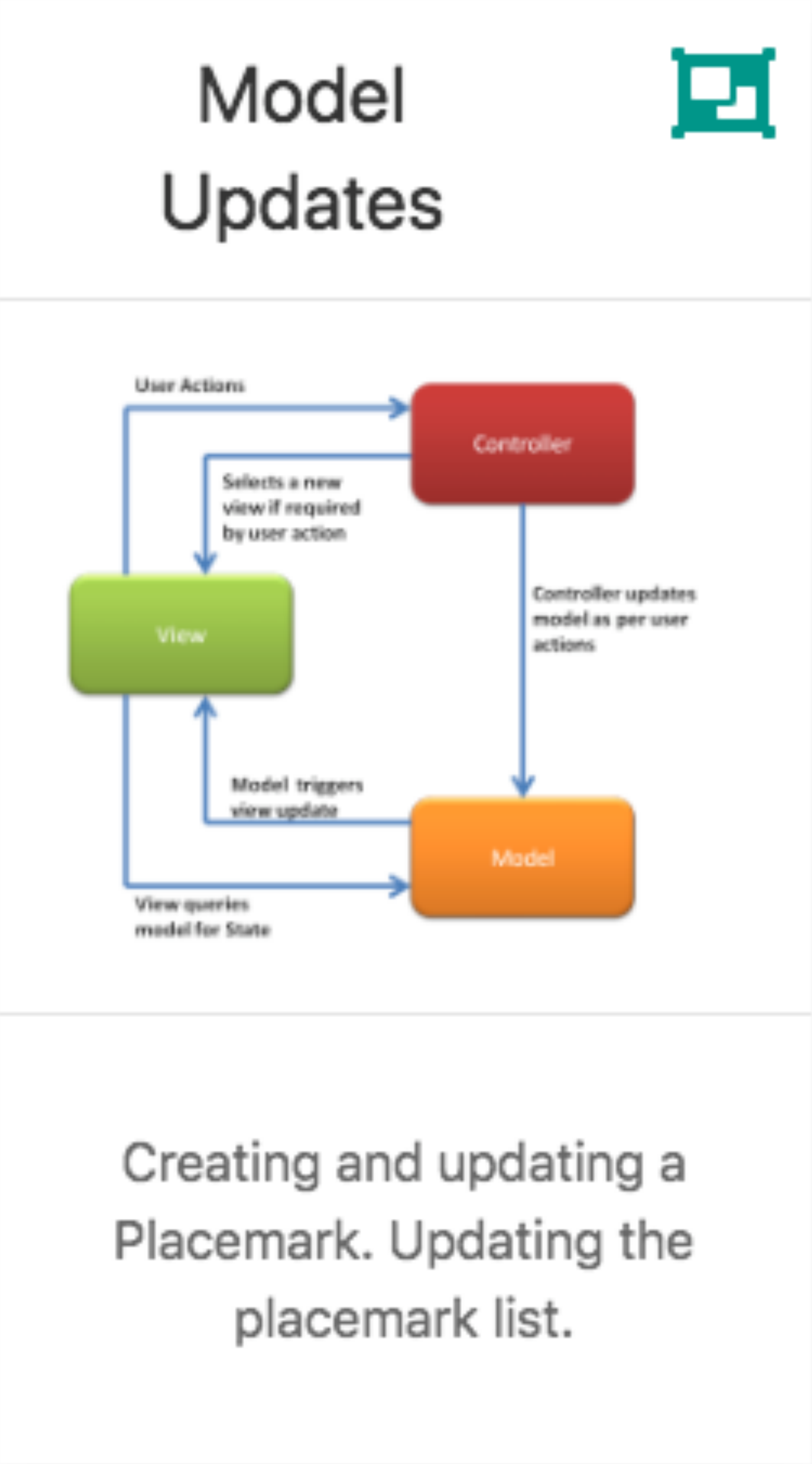
Model Updates



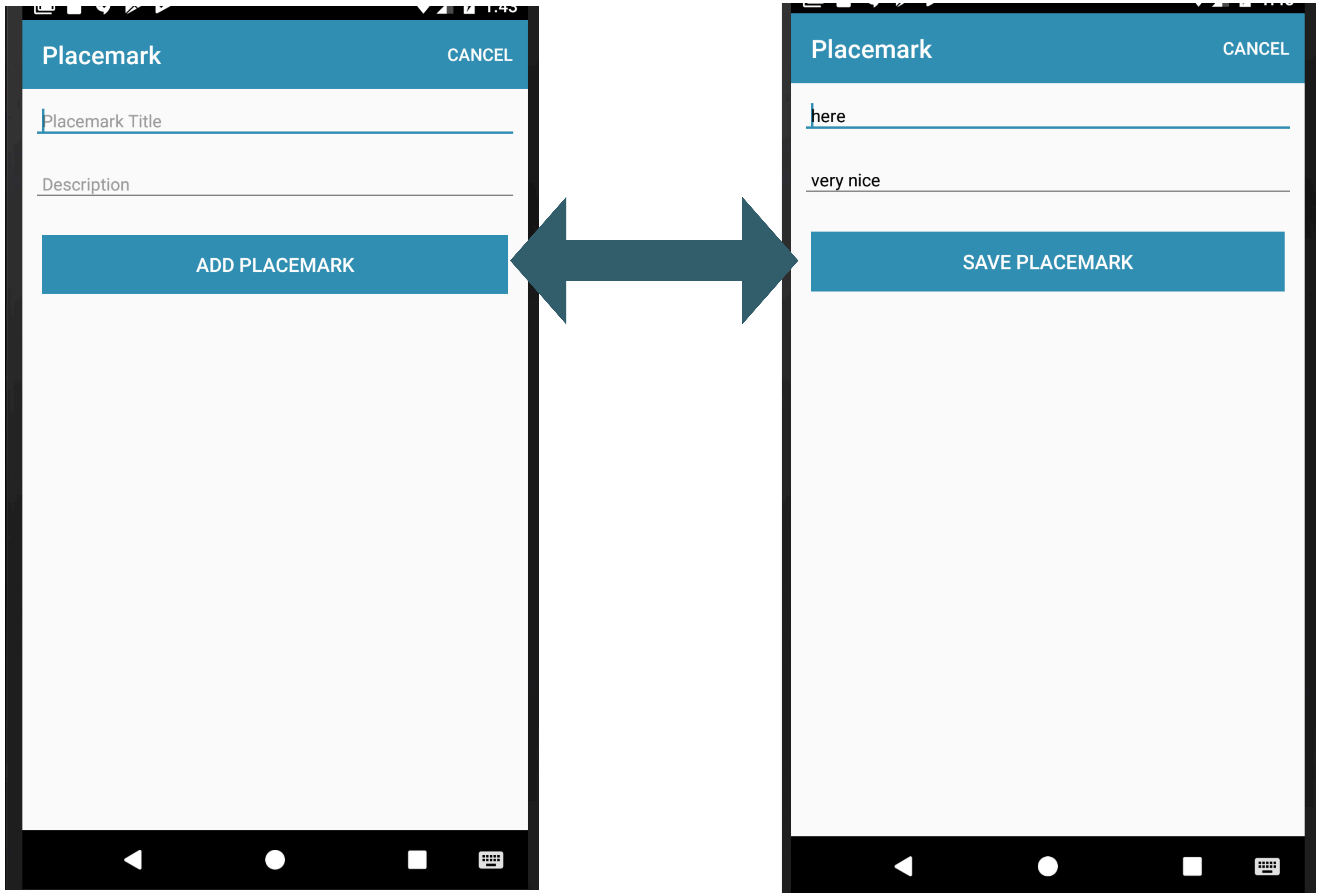
Model Updates

User Actions
Controller
Selects a new view if required by user action
View
Controller updates model as per user actions
Model triggers view update
Model
View queries model for State

Creating and updating a Placemark. Updating the placemark list.

Exercise 2: Add Button Label

When you select a placemark - and the PlacemarkActivity is launched, the button label is 'Add Placemark'. See if you can change this to 'Save Placemark'. This is only to be changed if PlacemarkActivity is launched with a placemark passed to it. Make sure the `Save Placemark` string is externalized (in strings.xml)



here

very nice

Exercise 3 Solution

Introduce a local flag in onCreate(), always initialised to false:

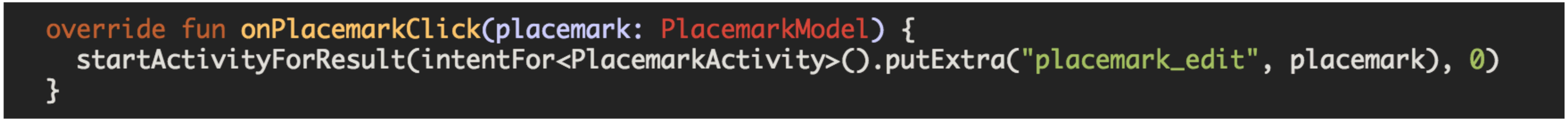
PlacemarkActivity

```
var edit = false
```

Set the flag to true if we have a placemark passed:

```
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)
}
```

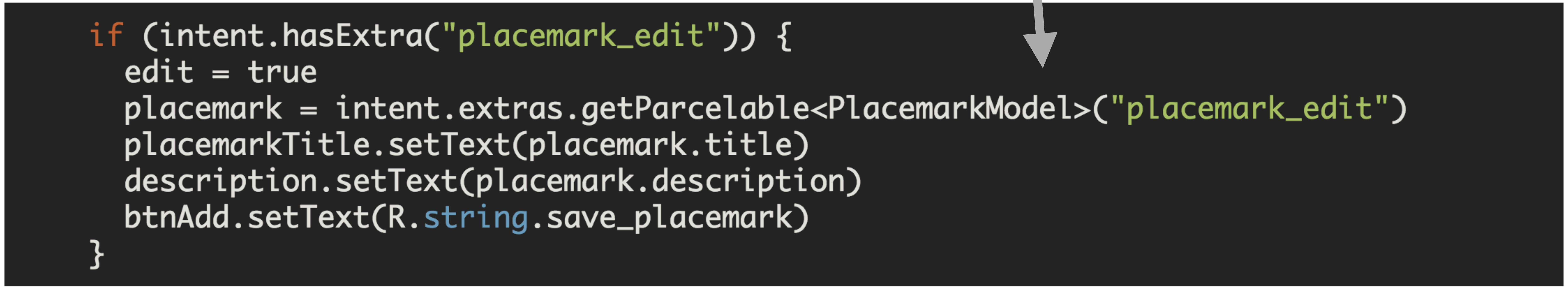
PlacemarkListActivity



```
override fun onPlacemarkClick(placemark: PlacemarkModel) {
  startActivityForResult(intentFor<PlacemarkActivity>().putExtra("placemark_edit", placemark), 0)
}
```

placemark sent to PlacemarkActivity

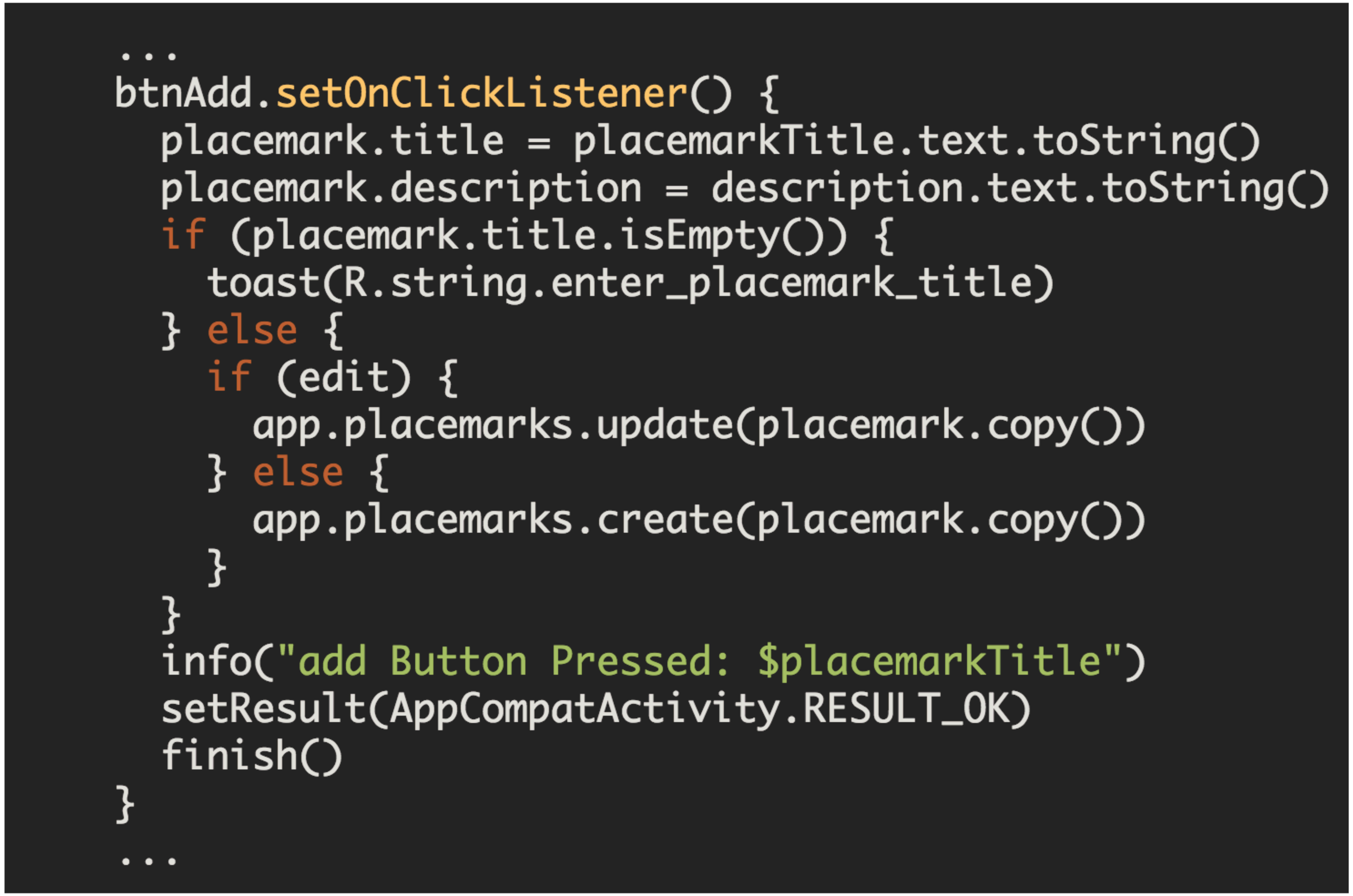
Set title to
‘Save” if
pacemaker
detected



```
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)
}
```

PlacemarkActivity

Then rework the add button event handler to follows:



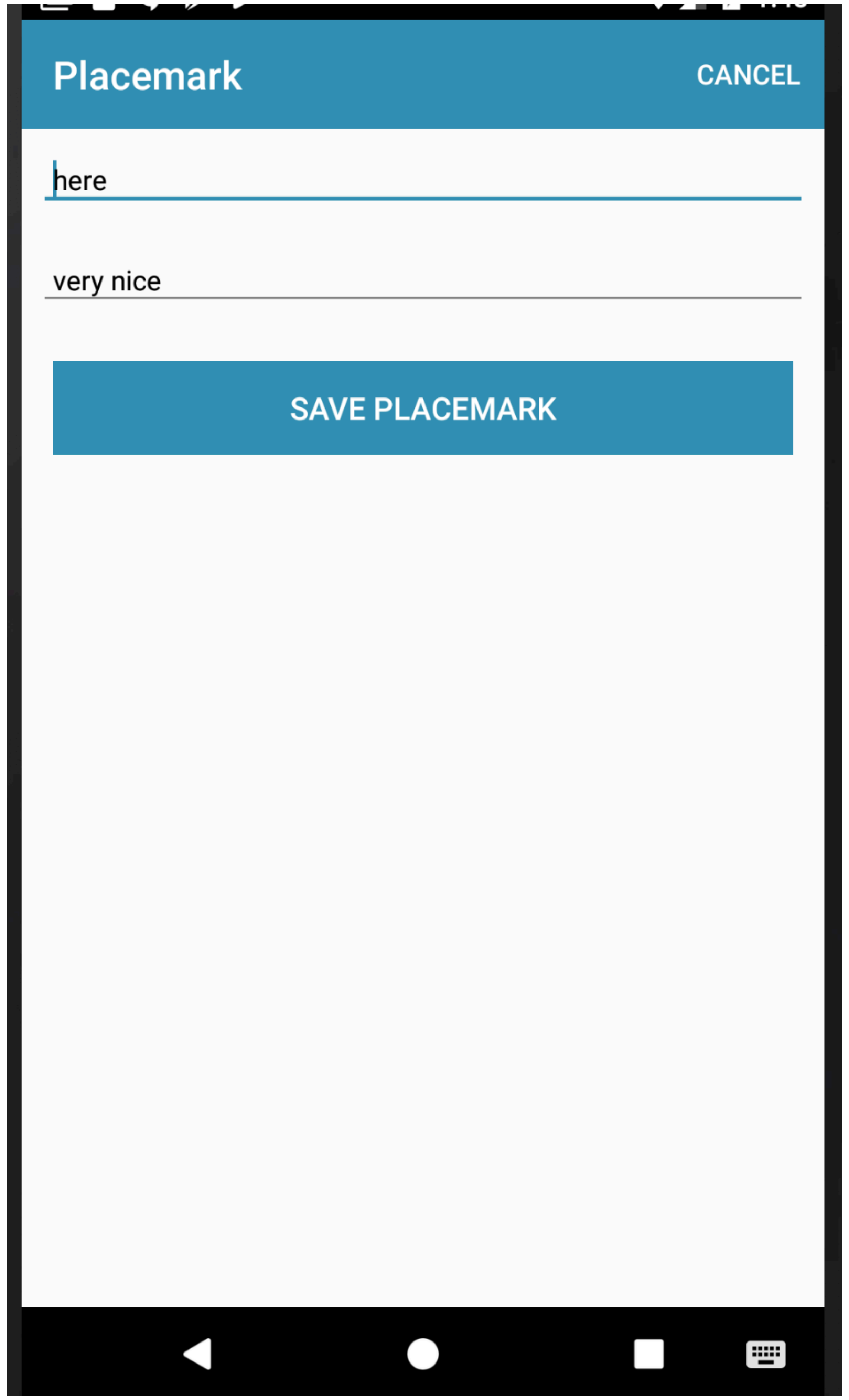
```
...
btnAdd.setOnClickListener() {
  placemark.title = placemarkTitle.text.toString()
  placemark.description = description.text.toString()
  if (placemark.title.isEmpty()) {
    toast(R.string.enter_placemark_title)
  } else {
    if (edit) {
      app.placemarks.update(placemark.copy())
    } else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

Check to see if we are **editing** an existing placemark

If we are - call **update()**

Otherwise call **create()**

If cancel pressed, end activity without any update



here

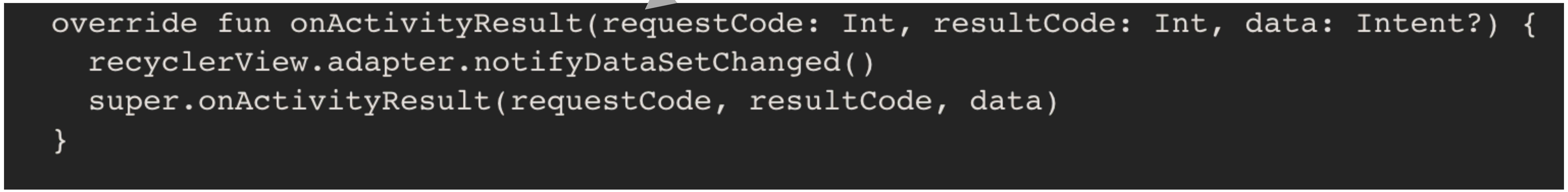
very nice

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

If cancel pressed, end
activity without any update

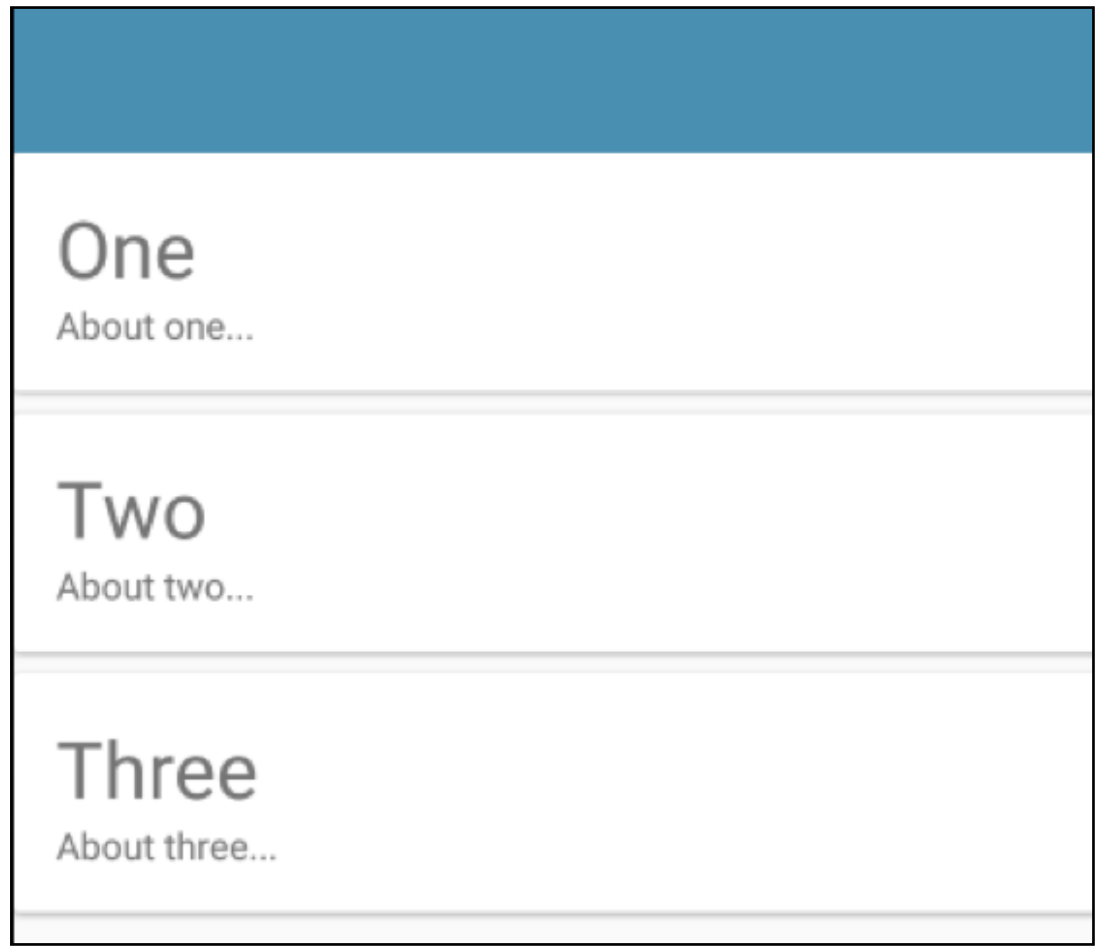
PlacemarkListActivity

When PlacemarkActivity finishes, and PlacemarkListActivity resumes, refresh the adapter



```
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
  recyclerView.adapter.notifyDataSetChanged()
  super.onActivityResult(requestCode, resultCode, data)
}
```

This will redraw the lists, updating contents



One
About one...

Two
About two...

Three
About three...

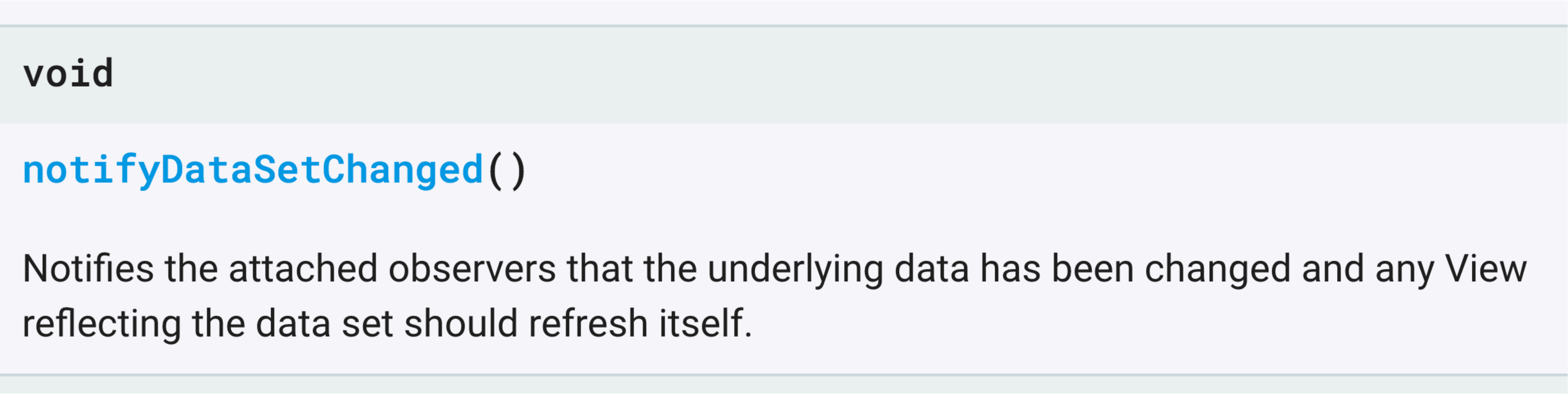
```
class PlacemarkListActivity : AppCompatActivity(), PlacemarkListener {

  lateinit var app: MainApp

  ...

  override fun onOptionsItemSelected(item: MenuItem?): Boolean {
    when (item?.itemId) {
      R.id.item_add -> startActivityForResult<PlacemarkActivity>(0)
    }
    return super.onOptionsItemSelected(item)
  }

  override fun onPlacemarkClick(placemark: PlacemarkModel) {
    startActivityForResult(intentFor<PlacemarkActivity>().putExtra("placemark_edit", placemark), 0)
  }

  override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    recyclerView.adapter?.notifyDataSetChanged()
    super.onActivityResult(requestCode, resultCode, data)
  }
}
```

Start with no placemark (new)

Start with selected placemark

PlacemarkActivity has finished, refresh placemarkList

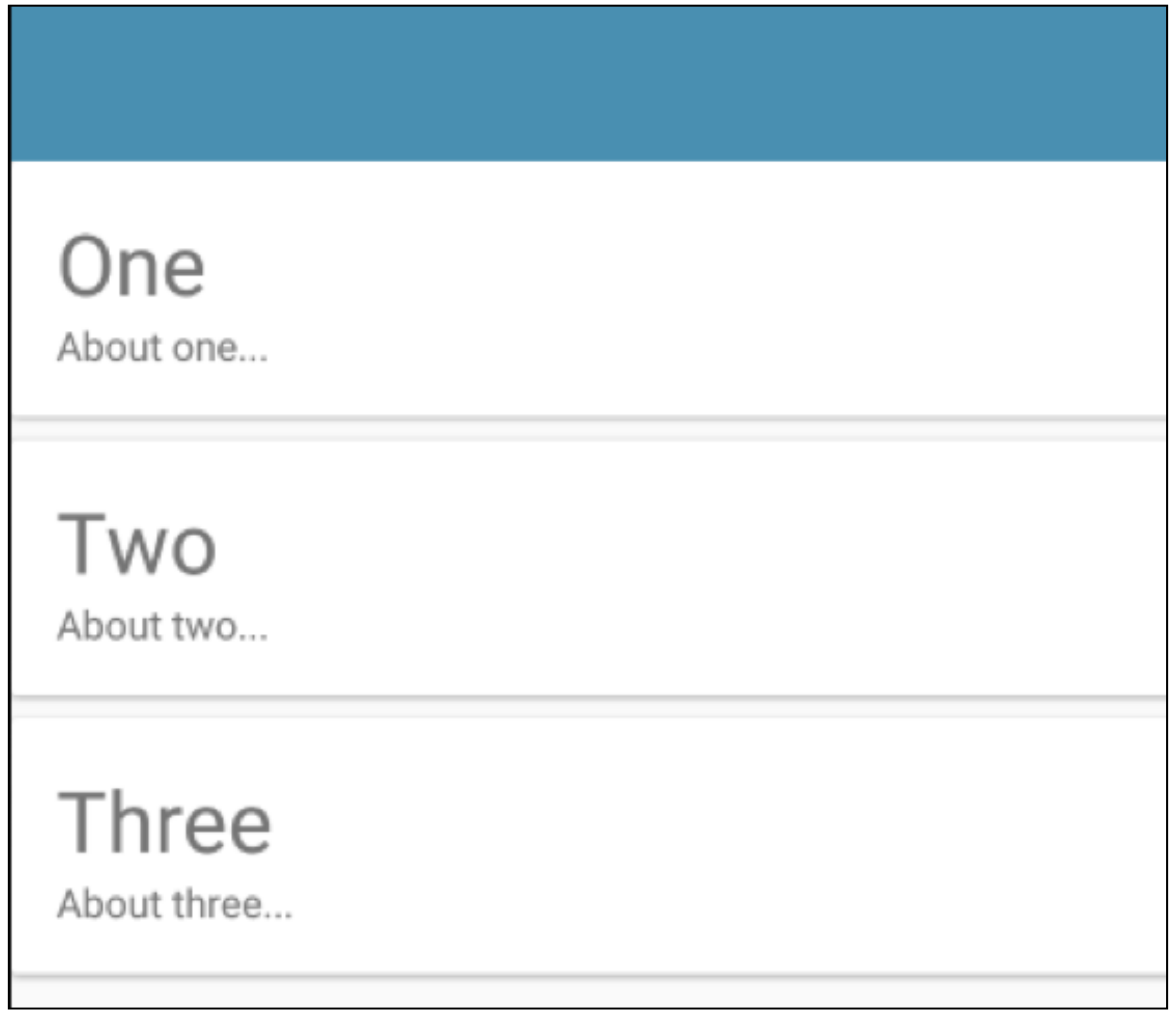
```
class PlacemarkListActivity : AppCompatActivity(), PlacemarkListener {

  override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    recyclerView.adapter?.notifyDataSetChanged()
    super.onActivityResult(requestCode, resultCode, data)
  }
}
```



`void`

`notifyDataSetChanged()`

Notifies the attached observers that the underlying data has been changed and any View reflecting the data set should refresh itself.


One
About one...
Two
About two...
Three
About three...