ListViews, Menus, context menus

CO5225 – Andrew Muncey

Contents

- ListViews
- ArrayAdapters
- Using the lateinit keyword
- Menus
- Context Menus

ListView

- Simple way of displaying items in a list
  - Not always the best approach – RecyclerView may be better
- Typically items will be stored in an ordered collection, such as
  - MutableList<String> (Java collection types can also be used)
- To get items into a ListView we need to use something that implements the ListAdapter interface
  - Such as ArrayAdapter

ArrayAdapter

- This allows a collection of items to be displayed in a given layout (which is repeated for each item in the list)
- Android provides some predefined layouts, or we can create our own
  - android.R.layout.simple_list_item_1 -> A text view
  - android.R.layout.simple_list_item_2 -> two text views (on two lines)
- If the item in the collection is not a string, the ArrayAdapter will call toString() on the object and show that
- Anything other than a layout with a single TextView requires us to implement the ListView interface ourselves
  - Can be done by subclassing ArrayAdapter

Creating an ArrayAdapter

- Adapter has a type which must match the collection type

```
private lateinit var adapter: ArrayAdapter<String>
```

- Constructor takes the context (activity), the layout and the list. e.g.:

```
adapter = ArrayAdapter( context: this,android.R.layout.simple_list_item_1, listItems)
```

Lateinit keyword

- Kotlin doesn’t permit variables to be declared without assigning a value
- Sometimes it’s not possible to init a variable in the constructor (or init block)
  - For example if we need to instantiate something in `onCreate()`
- Sometimes we don’t want to have to make something null if we know it will have a value when it’s first accessed
  - For example if the object is instantiated in `onCreate()`
- The `lateinit` keyword allows us to ‘promise’ that we will instantiate something before it’s used

Menus

- Basic Activity Template provides an app bar with a ‘main’ menu.
- Menu items can be set in the menu_main.xml resource file
- Actions for the menu items can be set by overriding onOptionsItemSelected(MenuItem) : Boolean
- We can create actions depending on the items id. E.g.

```
override fun onOptionsItemSelected(item: MenuItem): Boolean {
    return when (item.itemId) {
        R.id.menu_insert -> {
            //custom logic for this menu option here
            true
        }
        else -> super.onOptionsItemSelected(item)
    }
}
```

Context menus

- To add a context menu, create a menu xml file e.g.:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/menu_delete"
        android:title="@string/delete" />
</menu>
```

- Register a view (such as a list view) for context menus (in onCreate)

```
registerForContextMenu(listView)
```

Context menus continued

- Override onCreateContextMenu(), e.g.:

```
override fun onCreateContextMenu(menu: ContextMenu?, v: View?, menuInfo: ContextMenu.ContextMenuInfo?) {
    super.onCreateContextMenu(menu, v, menuInfo)
    menuInflater.inflate(R.menu.longpress, menu)
}
```

  - You may need to query the view to determine which context menu to inflate if you have multiple context menus

- Override onContextItemSelected(), e.g.:

```
override fun onContextItemSelected(item: MenuItem): Boolean {
    when (item.itemId) {
        R.id.menu_delete -> {
            //logic for this conext menu item
            return true
        }
    }
    return super.onContextItemSelected(item)
}
```