5COSC023W - MOBILE APPLICATION DEVELOPMENT
Lecture 2: Introduction to Android Programming - Views vs Jetpack Compose

Dr Dimitris C. Dracopoulos

Android Programming

A Lottery Program

A lottery ticket consists of 6 unique numbers in the range between 1 and 59.
Write an Android application which calculates such a 6 lucky random unique numbers that the user can play in the next lottery. Every time a button is pressed a new set of unique numbers is generated.

The Layout

File `activity_main.xml` contains:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/bt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Generate"
        android:layout_gravity="center" />
    <TextView
        android:id="@+id/results"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text=""
        android:textSize="22sp"
        android:textAlignment="center"
        />
</LinearLayout>
```

The Activity code

```
package uk.ac.westminster.lottery

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Button
import android.widget.TextView
import java.util.*

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

        var results = findViewById<TextView>(R.id.results)
        var button = findViewById<Button>(R.id.bt)

        button.setOnClickListener {
            calculate(results)
        }
    }
```

The Activity code (cont'ed)

```
    fun calculate(results: TextView) {
        var res = mutableListOf<Int>()

        var gen: Random = Random()

        while (res.size < 6) {
            var new_number = 1 + gen.nextInt(59)
            if (new_number !in res)
                res.add(new_number)
        }

        results.setText(" ")
        for (i in res) {
            results.append("" + i + " ")
        }
    }
}
```

Jetpack Compose - The Story

- Since the second half of 2023, Jetpack Compose is the recommended way for the creation of the UI of Android applications.
- Completely different way of doing things, not just simply the UI.
- The whole code for an Android application needs to be changed, not just the UI.

The `Views` Way of Implementation

Still supported but *Compose* is the new recommended way.

- Every widget (e.g. `Button`, `TextView`) is a subclass of the `View` class.
- All widgets are objects which are manipulated by calling their methods and change their state (e.g. call the `set` method to change the text of a `TextView`)
- The widget objects are passed around the program as references in order to access and modify them in other parts of the application.
- Classic way of object oriented programming: object are passed to different parts of the code so they can be accessed and call methods on them.

Views vs Jetpack Compose

- **Views approach**: Widgets (e.g. buttons) are objects and we need to call their methods to change what they display.
- **Compose approach**: Everything is based on `Composable` functions which are responsible to emit one or more UI components.
- **Compose** describe *WHAT* to draw, while **View** describe *HOW* to draw UI elements.
- Declarative (Compose) UI definition vs Object Oriented Programming way.