5COSC023W - MOBILE APPLICATION DEVELOPMENT
Lecture 6: Android Shared Preferences

Dr Dimitris C. Dracopoulos

Saving Data in an Android Application

- Use `onSaveInstanceState()` for configuration changes or system destroying and re-creating the activity.
- Saving Key-Value Sets (small amounts)
- Saving in Files
- Saving in SQL databases (large amounts of structured data)

SharedPreferences (Saving Key-Value Sets)

To create a new shared preference file or access an existing one, call one of the following methods to get a `SharedPreferences` object:

- `getSharedPreferences()`: if you need multiple shared preferences files (the name of the preference file is the first argument) - can be called from any `Context` in the app

  ```
  sharedPref: SharedPreferences =
         getSharedPreferences("preference_filename",
                              Context.MODE_PRIVATE);
  ```

- `getPreferences()`: call from an activity to use only one shared preference file associated with the activity

  ```
  sharedPref = getPreferences(Context.MODE_PRIVATE);
  ```

  Usage of `MODE_WORLD_READABLE` or `MODE_WORLD_WRITEABLE` imply that any other app can access your data (if it knows the filename)

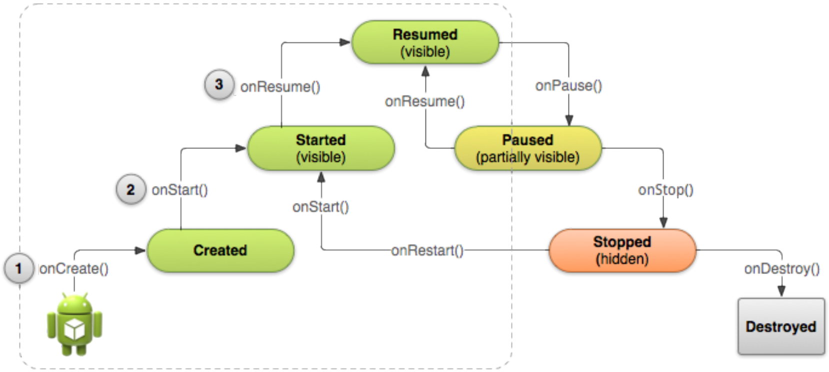
Saving Key-Value Sets (Writing to Shared Preferences)

1. Create a `SharedPreferences.Editor` by calling `edit()` on `SharedPreferences`.
2. Write the keys and values with `putInt()`, `putString()`, etc.
3. Call `apply()` or `commit()`.

```
sharedPref: SharedPreferences  = getActivity().getPreferences(
                                               Context.MODE_PRIVATE);
editor: SharedPreferences.Editor = sharedPref.edit(); // step 1
editor.putInt("key_name", newHighScore); // step 2
editor.apply();  // step 3
```

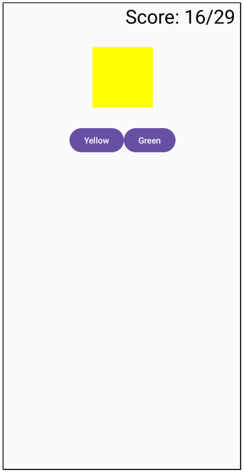
- `apply()` is an asynchronous call, i.e. it will not block the code waiting for the operation to complete. It returns no value.
- `commit()` is a synchronous (blocking) call and the current thread will be waiting for the operation to complete. It returns a boolean which is true if it completed successfully.

The Activity Lifecycle (cont'ed)



An Example Application for SharedPreferences

An application which the user can guess the displayed colour. The score is persisted even the application is killed and restarted (even if the device reboots).



An Example Application for SharedPreferences (cont'd)

```
package com.example.sharedpreferencescomposablelectureexample

import android.content.SharedPreferences
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.size
import androidx.compose.material3.Button
import androidx.compose.material3.ButtonDefaults
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.setValue
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.graphics.RectangleShape
import androidx.compose.ui.text.style.TextAlign
```

An Example Application for SharedPreferences (cont'd)

```
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import kotlin.random.Random

var colours = listOf(Color.Black, Color.Red, Color.Green, Color.Blue,
                     Color.Yellow, Color.White)
var colours_str = listOf("Black",  "Red",  "Green",  "Blue",
                         "Yellow", "White")

var correct = 0  // number of correct answers
var total = 0   // number of colours presented to the user
```

An Example Application for SharedPreferences (cont'd)

```kotlin
class MainActivity : ComponentActivity() {
    lateinit var prefs: SharedPreferences

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        // create the shared preferences object
        prefs = getSharedPreferences(
                "uk.ac.westminster.sharedpreferencescomposablelectureexample",
                 MODE_PRIVATE)
        // restore the data
        total = prefs.getInt("total", 0)
        correct = prefs.getInt("correct", 0)
        setContent {
            GUI()
        }
    }
```

An Example Application for SharedPreferences (cont'd)

```
override fun onPause() {
        super.onPause()

        // give me the editor associated with the sharedpreferences object
        // created in the onCreate() method
        var editor = prefs.edit()

        // start saving the data - in this case I just save the score
        editor.putInt("total", total)
        editor.putInt("correct", correct)

        // persist the data
        editor.apply()
    }
}
```

An Example Application for SharedPreferences (cont'd)

```
@Composable
fun GUI() {
    var colour_chosen by remember{ mutableStateOf(Color.Yellow) }

    //val index = Random.nextInt(colours.size)
    //colour_chosen = colours[index]
    val index = colours.indexOf(colour_chosen)
    val colour_chosen_str = colours_str[index]

    // second colour to be displayed as one of the 2 buttons
    var second_colour_str = colours_str[Random.nextInt(colours.size)]
    while (second_colour_str == colour_chosen_str)
        second_colour_str = colours_str[Random.nextInt(colours.size)]

    // determine whether the correct colour will be displayed as the first
    // or second button - correct_button = 0 for the first button,
    // 1 for the second
    val correct_button = Random.nextInt(2)

    var first_button_label = colour_chosen_str
    var second_button_label = second_colour_str
    if (correct_button == 1) {
        first_button_label = second_colour_str
        second_button_label = colour_chosen_str
    }
```

An Example Application for SharedPreferences (cont'd)

```
Column (
        modifier = Modifier.fillMaxSize(),
        horizontalAlignment = Alignment.CenterHorizontally,
    ) {
        Text("Score: $correct/$total",
            fontSize = 32.sp,
            modifier = Modifier
                .padding(bottom = 30.dp, end = 10.dp)
                .fillMaxWidth(),
            textAlign = TextAlign.End)
        Button(
            modifier = Modifier.size(height = 100.dp, width = 100.dp),
            onClick = {},
            shape = RectangleShape,
            colors = ButtonDefaults.buttonColors(
                                     containerColor = colour_chosen)) {
        }
```

An Example Application for SharedPreferences (cont'd)

```
            Row (
                modifier = Modifier.padding(top = 30.dp)
            ) {
                Button(onClick = {
                    ++total
                    if (correct_button == 0)
                        ++correct

                    colour_chosen = nextGame(colour_chosen)
                }) {
                    Text(first_button_label)
                }

                Button(onClick = {
                    ++total
                    if (correct_button == 1)
                        ++correct

                    colour_chosen = nextGame(colour_chosen)
                }) {
                    Text(second_button_label)
                }
            }
        }
    }
```

An Example Application for SharedPreferences (cont'd)

```
// choose a new colour to display and make sure it is different
//than the previous one
fun nextGame(previous_colour_chosen: Color): Color {
    var index = Random.nextInt(colours.size)
    var colour_chosen = colours[index]

    // choose a brand new colour if the same colour was produced
    while (previous_colour_chosen == colour_chosen) {
        index = Random.nextInt(colours.size)
        colour_chosen = colours[index]
    }

    return colour_chosen
}
```