Kotlin 1

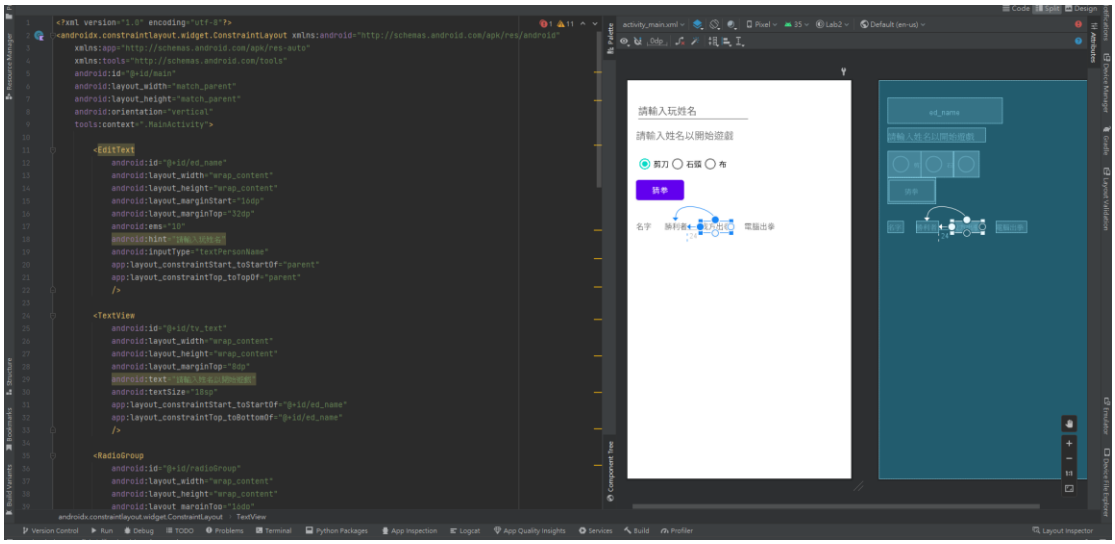
111360138 電子三甲 蔣安聖

讀書會

組員:
111360127 林煒哲
111360137 許鎧晏
111360138 蔣安聖
討論時間: 2024/10/23 18:00
地點:綜合科館

實作流程

將專案從 Java 轉成 Kotlin 並不需要更改 Layout 配置檔，沿用即可。



MainActivity

我們來比較 Java 與 Kotlin 的不同，左側為 Java 右側為 Kotlin。

```java
package com.example.javalab3;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;
```

```kotlin
package com.dolphinblast.lab2

import android.view.View
import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.RadioButton
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
```

Import 的部分基本上一樣，除了在 Java 中需要在行末加上分號

```java
public class MainActivity extends AppCompatActivity {
```

```kotlin
class MainActivity : AppCompatActivity() {
```

在主程式宣告中，明顯 Kotlin 比較簡單，不需要記什麼 public class xxx extends xxx，直接 class xxx : xxx 就結束了。

```java
private EditText ed_name;
private TextView tv_text, tv_name, tv_winner, tv_mmora, tv_cmora;
private RadioButton btn_scissor, btn_stone, btn_paper;
private Button btn_mora;
```

```kotlin
private lateinit var ed_name: EditText
private lateinit var tv_text: TextView
private lateinit var tv_name: TextView
private lateinit var tv_winner: TextView
private lateinit var tv_mmora: TextView
private lateinit var tv_cmora: TextView
private lateinit var btn_scissor: RadioButton
private lateinit var btn_stone: RadioButton
private lateinit var btn_paper: RadioButton
private lateinit var btn_mora: Button
```

宣告 Layout 中的物件時，我覺得 Kotlin 不能在宣告時將相同類型的變數寫在一行，有可能是我還不會寫，但我自己覺得 Java 的寫法對我來說比較直覺。

```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
```

```kotlin
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
```

我覺得 Java 與 Kotlin 建立畫面的宣告方式看起來很像，但有很多細節不同，我認為 Kotlin 比較簡單，因為我自己不太知道@Override 這個指令是做什麼用的，感覺很奇怪，沒在其他程式語言中看到@符號的程式碼，但 Kotlin 就沒這問題，跟 C 語言很像，我自己覺得比較好懂。

```java
ed_name = findViewById(R.id.ed_name);
tv_text = findViewById(R.id.tv_text);
tv_name = findViewById(R.id.tv_name);
tv_winner = findViewById(R.id.tv_winner);
tv_mmora = findViewById(R.id.tv_mmora);
tv_cmora = findViewById(R.id.tv_cmora);
btn_scissor = findViewById(R.id.btn_scissor);
btn_stone = findViewById(R.id.btn_stone);
btn_paper = findViewById(R.id.btn_paper);
btn_mora = findViewById(R.id.btn_mora);
```

```kotlin
ed_name = findViewById(R.id.ed_name)
tv_text = findViewById(R.id.tv_text)
tv_name = findViewById(R.id.tv_name)
tv_winner = findViewById(R.id.tv_winner)
tv_mmora = findViewById(R.id.tv_mmora)
tv_cmora = findViewById(R.id.tv_cmora)
btn_scissor = findViewById(R.id.btn_scissor)
btn_stone = findViewById(R.id.btn_stone)
btn_paper = findViewById(R.id.btn_paper)
btn_mora = findViewById(R.id.btn_mora)
```

連結元件的程式碼基本上就一樣，沒什麼特別的。\

```
btn_mora.setOnClickListener(view -> {
```

```
btn_mora.setOnClickListener { it: View!
```

開啟監聽的程式也很像，但 Kotlin 簡化了 Java 中指向結構的符號，更簡單明瞭了。

```
if(ed_name.length() < 1)
{
    tv_text.setText("請輸入玩家姓名");
}
else
{
    tv_name.setText(String.format("名字\n%s", ed_name.getText().toString()));

    if(btn_scissor.isChecked())
        tv_mmora.setText("我方出拳\n剪刀");
    else if(btn_stone.isChecked())
        tv_mmora.setText("我方出拳\n石頭");
    else
        tv_mmora.setText("我方出拳\n布");

    //產生隨機
    int computer_random = (int) (Math.random()*3);

    if(computer_random == 0)
        tv_cmora.setText("電腦出拳\n剪刀");
    else if(computer_random == 1)
        tv_cmora.setText("電腦出拳\n石頭");
    else
        tv_cmora.setText("電腦出拳\n布");
```

```
if (ed_name.length() < 1) {
    tv_text.text = "請輸入玩家姓名"
} else {
    tv_name.text = "名字\n${ed_name.text}"

    when {
        btn_scissor.isChecked -> tv_mmora.text = "我方出拳\n剪刀"
        btn_stone.isChecked -> tv_mmora.text = "我方出拳\n石頭"
        else -> tv_mmora.text = "我方出拳\n布"
    }

    // 產生隨機
    val computer_random = (Math.random() * 3).toInt()

    tv_cmora.text = when (computer_random) {
        0 -> "電腦出拳\n剪刀"
        1 -> "電腦出拳\n石頭"
        else -> "電腦出拳\n布"
    }
```

在 if 的語法中，兩者完全一樣，我想 if 的寫法在哪邊應該都差不多，但內部的程式就很不同了。

```
tv_text.setText("請輸入玩家姓名");
```

```
tv_text.text = "請輸入玩家姓名"
```

當初在 Java 中要更改物件的文字需要打一串，到了 Kotlin 中，直接將物件當作變數，更改其中的 text 類別即可，簡單許多。

```
tv_name.setText(String.format("名字\n%s", ed_name.getText().toString()));
```

```
tv_name.text = "名字\n${ed_name.text}"
```

我覺得簡化最多的是這行，我們只想實現在 TextView 中顯示在 Plain Text 輸入的文字，並加上名字這兩個文字，但卻需要使用諸多函數來實行，反之 Kotlin 就很像 Python，簡單粗暴，將這些物件當作變數來使用即可。

```
int computer_random = (int) (Math.random()*3);

if(computer_random == 0)
    tv_cmora.setText("電腦出拳\n剪刀");
else if(computer_random == 1)
    tv_cmora.setText("電腦出拳\n石頭");
else
    tv_cmora.setText("電腦出拳\n布");
```

```
var computer_random = (Math.random() * 3).toInt()

tv_cmora.text = when (computer_random) {
    0 -> "電腦出拳\n剪刀"
    1 -> "電腦出拳\n石頭"
    else -> "電腦出拳\n布"
}
```

我個人覺得 Kotlin 最有趣的部分在於，變數宣告以及 if else if else 的簡化，就以產生亂數為例子，正常來說我們需要宣告一個固定型態的變數，但在 Kotlin 中支援 var，我們不用特別指令資料類型，它會自動識別。
而 if 的簡化我覺得是這次 Kotlin 學習中令我最印象深刻的程式，如圖所示，

```
when {
    btn_scissor.isChecked -> tv_mmora.text = "我方出拳\n剪刀"
    btn_stone.isChecked -> tv_mmora.text = "我方出拳\n石頭"
    else -> tv_mmora.text = "我方出拳\n布"
}
```

它可以將 if else if 的結構變更成跟 switch 很像的結構，在 when 中，當條件式成立時，執行該行箭頭右側的程式，但如果判斷條件都與某變數有關，那麼還可以簡化，

```
when (computer_random) {
    0 -> tv_cmora.text = "電腦出拳\n剪刀"
    1 -> tv_cmora.text = "電腦出拳\n石頭"
    else -> tv_cmora.text = "電腦出拳\n布"
}
```

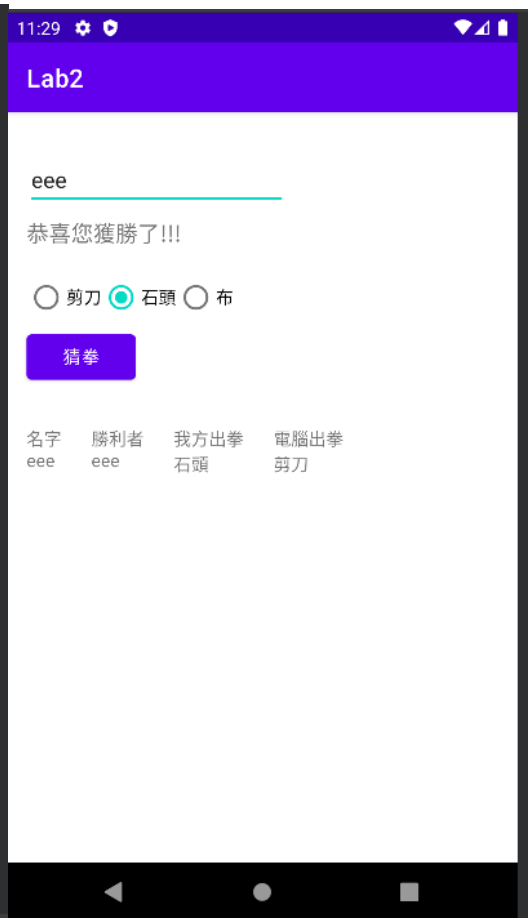
變成該結構，但如果執行的程式內容基本上一樣，還可以再簡化，

```
tv_cmora.text = when (computer_random) {
    0 -> "電腦出拳\n剪刀"
    1 -> "電腦出拳\n石頭"
    else -> "電腦出拳\n布"
}
```

變成這種簡潔的形式，我覺得在邏輯判斷的程式中，Kotlin 完全勝出，能以簡潔且具結構性的表示方式呈現邏輯判斷以及執行內容，這方式深深吸引了我。

執行畫面


11:29
Lab2
請輸入玩姓名
請輸入姓名以開始遊戲
剪刀 石頭 布
猜拳
名字 勝利者 我方出拳 電腦出拳


11:29
Lab2
eee
恭喜您獲勝了!!!
剪刀 石頭 布
猜拳
名字 勝利者 我方出拳 電腦出拳
eee eee 石頭 剪刀


11:30
Lab2
eee
可惜，電腦獲勝了!
剪刀 石頭 布
猜拳
名字 勝利者 我方出拳 電腦出拳
eee 電腦 剪刀 石頭


11:31
Lab2
eee
平局，請再試一次
剪刀 石頭 布
猜拳
名字 勝利者 我方出拳 電腦出拳
eee 平手 布 布

心得

在這次的 Kotlin 學習中，我覺得 Kotlin 跟 Python 很像，都沒有分號而是改用換行，我認為以換行代替分號是個很好的主意，這樣可以強迫開發者要好好縮排與換行，不會形成一行中有很多程式碼的問題。而 Java 與 Kotlin 比對的話，我自己比較喜歡 Kotlin，因為它簡化了許多符號以及語法，令我印象最深刻的就是 when 語法了，我沒想過有個程式語言可以把判斷式以這麼圖像化的方式呈現，在語法中所使用的箭頭以及條件式等結構，能讓我一眼就看出來這段程式的判斷邏輯，Kotlin 是個好東西。

GitHub 程式連結與截圖

https://github.com/DolphinBlast/JavaHW03