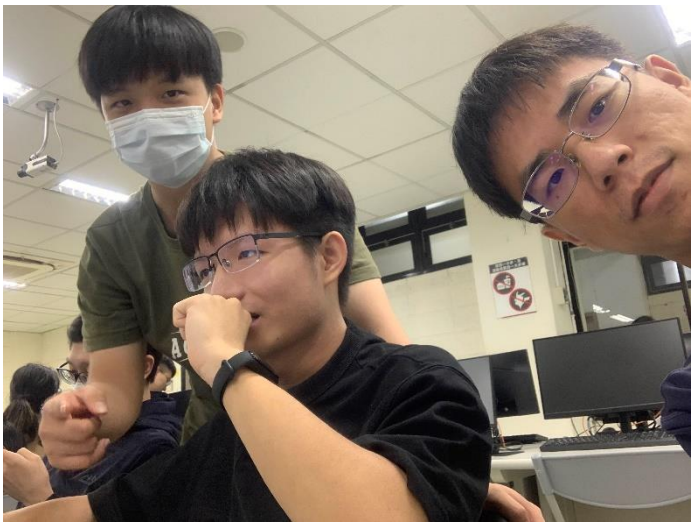
Kotlin 2

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: 2024/11/06 下午 2 點 30
地點:共同科館

實作流程

除去先前提過的幾種便捷語法，下列是我在不同 Lab 中學到的新語法。(左側為 JAVA 右側為 Kotlin)
|CHAPTER 04|

```java
if(intent != null && intent.getExtras() != null){
    Bundle b = intent.getExtras();
    String str1 = b.getString("drink");
    String str2 = b.getString("sugar");
    String str3 = b.getString("ice");
    tv_drink.setText(String.format("飲料: %s\n\n甜度: %s\n\n冰塊: %s",str1,str2,str3));
}
```

```kotlin
intent?.extras?.let { b ->
    val str1 = b.getString( key: "drink")
    val str2 = b.getString( key: "sugar")
    val str3 = b.getString( key: "ice")
    tvDrink.text = "飲料: $str1\n\n甜度: $str2\n\n冰塊: $str3"
}
```

在 Java 中，只能照像 C 語言那樣慢慢寫判斷式，但在 kotlin 中使用?來判斷，使其變得很簡潔。

```java
private TextView tv_drink;
```

```kotlin
private lateinit var tvDrink: TextView
private var buSelect: Button? = null
```

在變數宣告中，java 能以較簡短的方式宣告，而在 kotlin 有兩種不同的宣告方式，分為含有 lateinit 與沒有兩種，若選用後者則須宣告時初始化，在變數宣告中，我還是認為 java 更簡單。

```java
bu_send = findViewById(R.id.bu_send);
bu_send.setOnClickListener(view -> {
    set_drink = findViewById(R.id.ed_drink);
    String drink = set_drink.getText().toString();
    Intent i = new Intent();
    Bundle b = new Bundle();
    b.putString("drink", drink);
    b.putString("sugar", sugar);
    b.putString("ice", ice_opt);
    i.putExtras(b);
    setResult(MainActivity2.RESULT_OK, i);
    finish();
});
```

```kotlin
buSend = findViewById(R.id.bu_send)
buSend.setOnClickListener { it: View!
    setDrink = findViewById(R.id.ed_drink)
    val drink = setDrink.text.toString()
    val intent = Intent().apply { this: Intent
        putExtras(Bundle().apply { this: Bundle
            putString("drink", drink)
            putString("sugar", sugar)
            putString("ice", iceOpt)
        })
    }
    setResult(RESULT_OK, intent)
    finish()
```

然而在數值回傳的部分 java 與 kotlin 也有細微的差異，差在 kotlin 將一些程式合併起來，我自己覺得並沒有比較簡化。

|CHAPTER 06|

```java
btn.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View view) {

        final AlertDialog.Builder dialog = new AlertDialog.Builder(MainActivity.this);
        dialog.setTitle("請選擇功能");
        dialog.setMessage("請根據下方按鈕選擇要顯示的物件");

        dialog.setNeutralButton("取消", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                Toast.makeText(MainActivity.this, "dialog關閉", Toast.LENGTH_SHORT).show();
            }
        });

        dialog.setNegativeButton("自定義Toast", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) { showToast(); }
        });

        dialog.setPositiveButton("顯示list", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                showListDialog();
            }
        });
        dialog.show();
    }
});
```

```kotlin
btn.setOnClickListener { it: View!
    val dialog = AlertDialog.Builder( context: this@MainActivity)
    dialog.setTitle("請選擇功能")
    dialog.setMessage("請根據下方按鈕選擇要顯示的物件")

    dialog.setNeutralButton( text: "取消") { _, _ ->
        Toast.makeText( context: this@MainActivity, text: "dialog關閉", Toast.LENGTH_SHORT).show()
    }

    dialog.setNegativeButton( text: "自定義Toast") { _, _ ->
        showToast()
    }

    dialog.setPositiveButton( text: "顯示list") { _, _ ->
        showListDialog()
    }
    dialog.show()
}
```

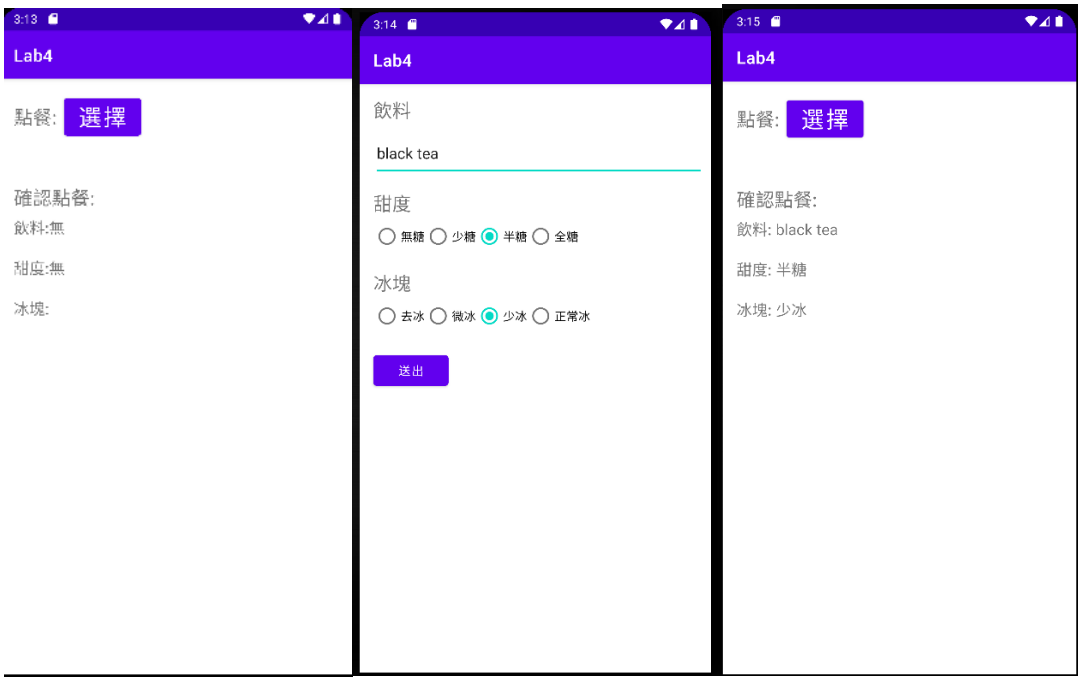
在這段程式中，kotlin 又發揮了它的長處，大幅化簡程式，去除許多難以了解的函數。

|CHAPTER 05|

在 Lab5 中，宣告 viewPager2 元件，並建立四份 activity，三份用於管理三個頁面，另一個用於後端管理這三個頁面的連結。

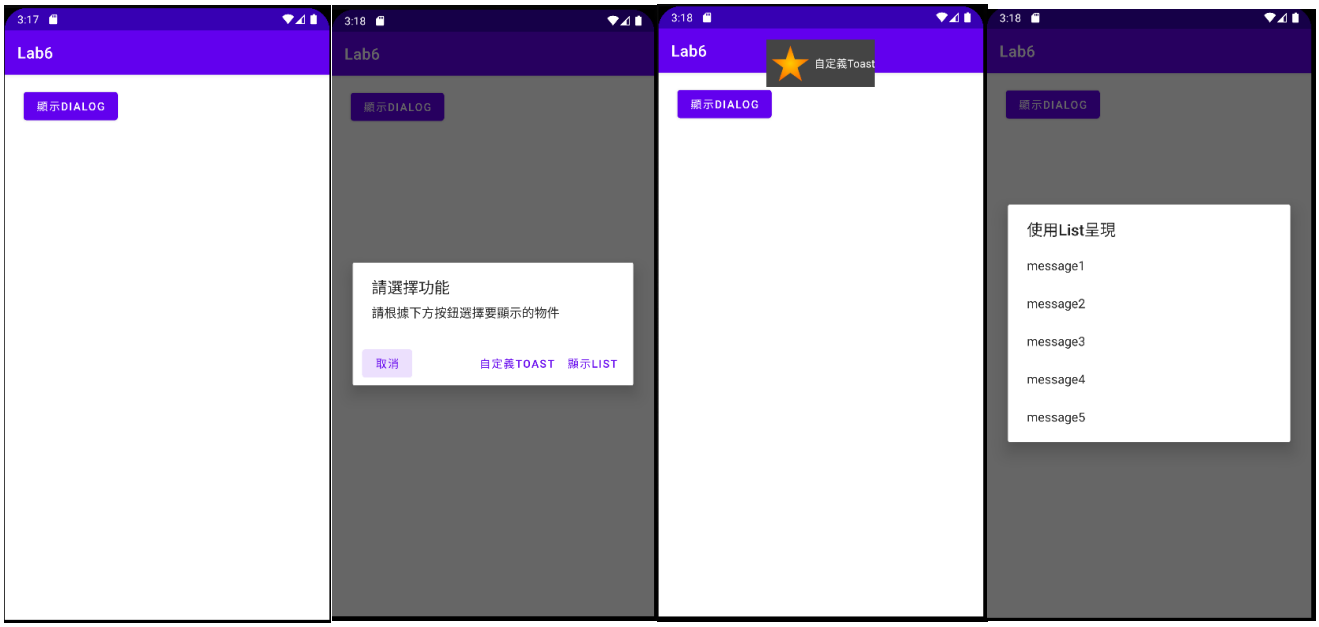
執行畫面

|CHAPTER 04|



|CHAPTER 05|

|CHAPTER 06|



詢問 GPT 問題

1.
```
intent?.extras?.let { b ->
    val str1 = b.getString( key: "drink")
    val str2 = b.getString( key: "sugar")
    val str3 = b.getString( key: "ice")
    tvDrink.text = "飲料: $str1\n\n甜度: $str2\n\n冰塊: $str3"
}
```
?在這邊是什麼意思

在 Kotlin 中，`?` 是空安全運算符，用於表示某個變數或表達式可以為 `null`。在你的程式碼中，`intent?.extras?.let { ... }` 使用了這個運算符來避免 `NullPointerException`。以下是每個部分的含義：

1. `intent?`：這裡的 `?` 表示如果 `intent` 為 `null`，整個表達式會直接返回 `null`，並不會繼續執行後面的部分。
2. `.extras?`：同樣的，`extras` 可能為 `null`，因此 `.extras?` 確保如果 `intent.extras` 是 `null`，就會停止執行而不再進入 `let` 區塊。
3. `.let { b -> ... }`：當 `intent` 和 `intent.extras` 都不為 `null` 時，`let` 函數會被調用，把 `extras` 賦值給變數 `b`，讓你在 `{ ... }` 區塊中安全地使用。

這樣寫可以避免 `null` 引起的閃退問題（`NullPointerException`）。

?在 kotlin 中?可允許變數有 null 的狀態而不會產生 NullPointerException 的報錯，不過在之後變數中都需要在結尾加上?。

2. |CHAPTER 05|的運作邏輯

這五個程式一起運作，組成了一個 Android App 專案的主要架構，透過 `ViewPager2` 顯示多個 `Fragment`。各程式的作用如下：

1. `MainActivity`

`MainActivity` 是應用程式的主活動，用於顯示和管理 `ViewPager2` 的 UI 以及初始化應用程式的主要設定。

- `onCreate` 方法中：
  - 設定主活動的 UI 佈局為 `activity_main`。
  - 使用 `ViewCompat.setOnApplyWindowInsetsListener` 設定頂部和底部的系統條（如狀態欄）的插入，以適應各種裝置。
  - 初始化並設定 `ViewPager2` 和 `ViewPagerAdapter`，使其可以載入多個 Fragment 頁面，並設定鄰近頁面緩存為 1 頁。
- 其他生命周期方法如 `onStart`、`onResume`、`onPause` 等，則只是打印對應的 Log 以監控活動的狀態變化。

2. `FirstFragment`

`FirstFragment` 是 `ViewPager2` 中的第一個頁面，繼承自 `Fragment` 類別，包含了 Fragment 的標準生命週期方法，並且在每個方法內使用 `Log.e` 來記錄生命週期事件。此 Fragment 的主要作用是定義和顯示對應的 UI 佈局 `fragment_first`。

- `onCreateView`：填充並返回 Fragment 的佈局檔案。
- `onAttach` 和 `onDetach`：標記 Fragment 與活動的連接和分離階段。
- `onViewCreated`：指示 Fragment 的 UI 元件已經建立。

3. `SecondFragment`

`SecondFragment` 與 `FirstFragment` 類似，也是繼承自 `Fragment` 類別，用於 `ViewPager2` 的第二頁。

- 使用同樣的生命周期方法來記錄事件。
- 定義並顯示 `fragment_second` 佈局。

4. `ThirdFragment`

`ThirdFragment` 是 `ViewPager2` 的第三個頁面，也繼承自 `Fragment` 類別。

- 與 `FirstFragment` 和 `SecondFragment` 類似，包含完整的生命週期方法來記錄日誌。
- 定義並顯示 `fragment_third` 佈局。

5. `ViewPagerAdapter`

`ViewPagerAdapter` 是 `ViewPager2` 的適配器（adapter），繼承自 `FragmentStateAdapter`，用於管理和返回對應的 Fragment 頁面。

- `getItemCount`：返回 Fragment 的數量（此處為 3）。
- `createFragment`：根據頁面位置 `position`，返回對應的 `FirstFragment`、`SecondFragment` 或 `ThirdFragment`。

3. private lateinit var tvDrink: TextView

在 Kotlin 中，`lateinit` 的使用對變數初始化的要求產生了不同影響，以下是具體差異：

1. `lateinit` 的作用

- `lateinit` 是用來標記延遲初始化的屬性，僅適用於 **var 變數**，而且類型必須是非空類型（如 `TextView`，不能是 `TextView?`）。
- 使用 `lateinit` 的變數允許在稍後進行初始化，而不是在定義時立即初始化，這在 Android 的 Activity 或 Fragment 中很常見，因為視圖的綁定通常在 `onCreate` 或 `onViewCreated` 之後才執行。

2. 不使用 `lateinit` 的情況

- 不加 `lateinit` 時，變數必須在聲明時立即賦值，否則編譯器會報錯。
- 如果不加 `lateinit` 又希望延遲初始化，通常可以將變數定義為 **可空類型**（`TextView?`）並在使用時進行判斷處理。

3. 加 `lateinit` 和不加的具體差異

- 加 `lateinit`：

```kotlin
private lateinit var tvDrink: TextView

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    tvDrink = findViewById(R.id.tvDrink) // 延遲初始化
}
```

使用 `lateinit` 可以在稍後進行初始化，但在使用 `tvDrink` 前必須先確保已經初始化。若在初始化前訪問，會拋出 `UninitializedPropertyAccessException`。

- 不加 `lateinit`，但定義為可空類型：

```kotlin
private var tvDrink: TextView? = null

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    tvDrink = findViewById(R.id.tvDrink)
}
```

當 `tvDrink` 是可空類型時，使用時需要進行空檢查（`tvDrink?.text = "some text"`），避免因未初始化而出現空指針異常。

4. 何時使用 `lateinit`

- `lateinit` 適合需要非空變數且必須延遲初始化的情況，例如 Android 的視圖綁定。
- 若變數允許為空，且可用空安全運算符來進行操作（如 `tvDrink?.text`），則可以不用 `lateinit`。

總結

加上 `lateinit` 可以避免空檢查且便於在 Activity 或 Fragment 中延遲初始化，但使用時需確保初始化順序。

心得

在這三個 Lab 中，我學習到要如何開啟新畫面、回傳數值、Taost 訊息、畫面捲動等，我認為這些功能都是撰寫一個 APP 會需要用到的功能，尤其是開啟新畫面，我還沒在市面上看過只有一個畫面的 APP，因此這個語法更要牢記。

GitHub 程式連結與截圖

https://github.com/DolphinBlast/JavaHW05