Kotlin homework2

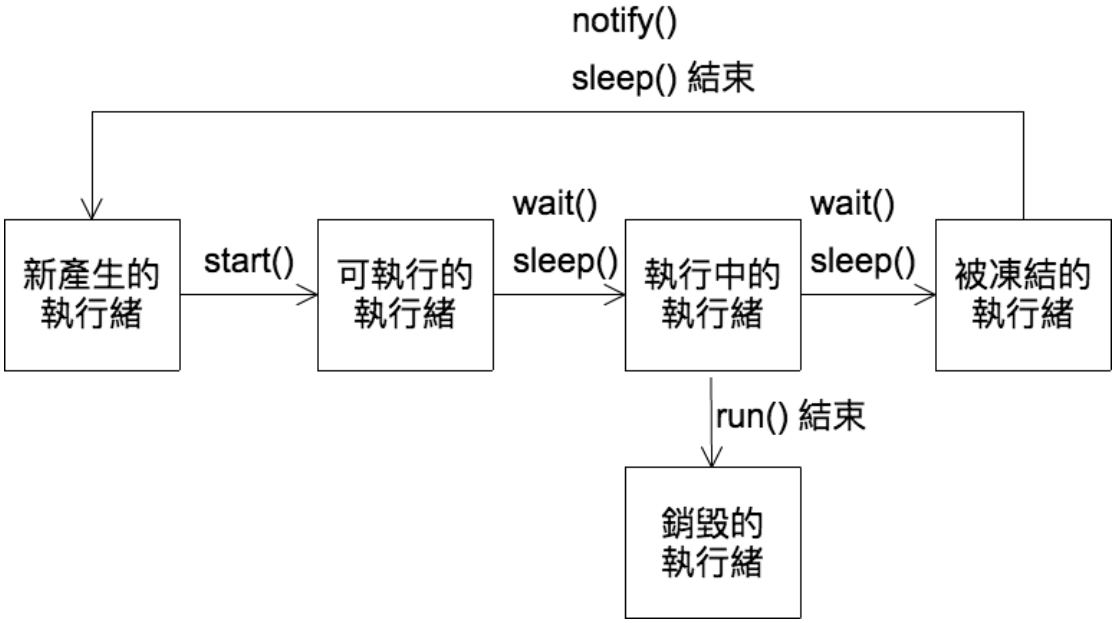
1.生命週期

Java 與 Kotlin 差異

kotlin 這個程式有頻繁使用一個 log 是用於做打印的

```
        setContentView(R.layout.activity_main)
        Log.e( tag: "MainActivity", msg: "onCreate")
        //建立 FragmentPagerAdapter 物件
        val adapter = ViewPagerAdapter(supportFragmentManager)
        //連接 Adapter，讓畫面(Fragment)與 ViewPager 建立關聯
        findViewById<ViewPager>(R.id.viewPager).adapter = adapter
    }
    override fun onRestart() {
        super.onRestart()
        Log.e( tag: "MainActivity", msg: "onRestart")
    }
    override fun onStart() {
        super.onStart()
        Log.e( tag: "MainActivity", msg: "onStart")
    }
    override fun onResume() {
        super.onResume()
        Log.e( tag: "MainActivity", msg: "onResume")
    }
    override fun onPause() {
        super.onPause()
        Log.e( tag: "MainActivity", msg: "onPause")
    }
    override fun onStop() {
        super.onStop()
        Log.e( tag: "MainActivity", msg: "onStop")
    }
    override fun onDestroy() {
        super.onDestroy()
        Log.e( tag: "MainActivity", msg: "onDestroy")
    }
}
```

而 java 的執行續則是這樣



2.訊息提示

java

```
btn.setOnClickListener(new View.OnClickListener() {
    @Override
```

Kt

```
val btn  =findViewById<Button>(R.id.button)
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")
```

Kotlin 不需要 new 跟 overrid

java

```
private void showToast(){
    Toast toast = new Toast(MainActivity.this);
```

kotlin

```
private fun showToast() {
    val toast = Toast( context: this)
    toast.setGravity(Gravity.TOP,  xOffset: 0,  yOffset: 50);
    toast.duration = Toast.LENGTH_SHORT;
```

Kotlin 用 val 連接

Java

```java
public void onClick(View v) {
    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("自訂義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

```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",DialogInterface.OnClickListener(){dialog,which ->
        Toast.makeText( context: this, text: "dialog關閉",Toast.LENGTH_SHORT).show()
    })

    dialog.setNegativeButton( text: "自訂義Toast",DialogInterface.OnClickListener(){dialog,which ->
        showToast()

    })
    dialog.setPositiveButton( text: "顯示list",DialogInterface.OnClickListener(){dialog,which ->
        showListDialog()
    })

    dialog.show()
}
```

Java 使用的程式碼較長 Kotlin 用 dialog, witch 可以直接省去顯示對話框

3.點餐系統

Java Button 按下後 跑 public 裡的 Intent

```java
btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        startActivityForResult(new Intent(MainActivity.this,
                MainActivity2.class),1);
```

Kotlin 後面多了 MainActivity2:: class,java

```kotlin
btn.setOnClickListener(){ it: View!
    startActivityForResult(Intent( packageContext: this,
        MainActivity2::class.java), requestCode: 1)
```

Java 用 bundle b 去寫入 data

```java
Bundle b = data.getExtras();
```

Kotlin 直接 val 寫入

```kotlin
val b = data.extras!!
```

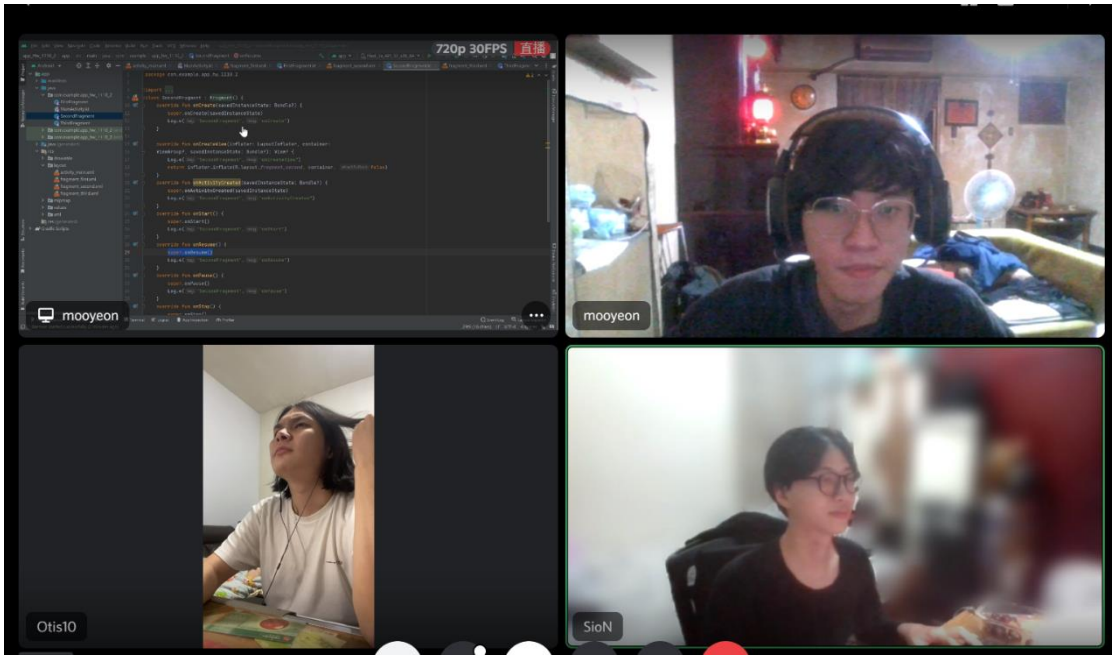
Java main2 裡面的 radiogroup 比較攏長 用 case 去判斷

```java
public void onCheckedChanged(RadioGroup radioGroup, int i ) {
    switch(i){
        case R.id.radioButton:
            sugar = "無糖";
            break;
        case R.id.radioButton2:
            sugar = "少糖";
            break;
        case R.id.radioButton3:
            sugar = "半糖";
            break;
        case R.id.radioButton4:
            sugar = "全糖";
            break;
    }
```

Kotlin 則用 i-> when 減少很多判斷的地方

```kotlin
rg1 = findViewById(R.id.radioGroup)
rg1.setOnCheckedChangeListener() { radioGroup, i ->
    when (i) {
        R.id.radioButton  -> sugar = "無糖"
        R.id.radioButton2 -> sugar = "少糖"
        R.id.radioButton3 -> sugar = "半糖"
        R.id.radioButton4 -> sugar = "全糖"
    }
```

讀書會:(
組員:郭鍵霆、張志權、許廷維
討論時間 11 月 7 號 晚上 8 點
地點:Discord



心得:
上網看了很多資料，像是 kotlin 生命週期，有 onCreate 再由 onStart 接力，onStart 是讓頁面看見，並讓 UI 與使用者互動，完成後，在跳至 OnResume 持續與使用者互動。
onPause 觸發就是偵測到信號，可以選擇接收，或是拒收，選接收就會進入 onStop 的階段，onStop 可設計將手機的資源釋放出來，讓給使用者進行其它操作使用。
問了同學很多，比較大的問題還是上網看生命週期，一開始不太知道在講什麼，看了一些資料才知道他是一個過程。
https://github.com/opert0816/kotlin-homework2.git