Batch #14 / Android Class Assignment - Week 2

1. **What is Inheritance? Try to explain in Mandarin.**

   Inheritance 繼承是 object-oriented language 的一種方便性的設計，它的目的在提高程式的重複使用，以歸納的方式將有相同特性的類別做分類。繼承的規則中，subclass 可以承接使用 superclass 中所有可存取的內容，保持 superclass 的狀態行為，呼叫相同的 variable, method 和 constructor，共同的資料方法只需要描述一次，同時 subclass 能增加更多成員做衍生。如果需要重新定義繼承後的內容，也可以用 override 做改寫，在相同的型別規範上展現個體的差異性。欲繼承的 superclass 必須是開放狀態，且常見規則上一 subclass 只能繼承一個 superclass，如果在類別或其中的成員前加上 final keywords，可以避免再被繼承或覆寫。

2. **How to split a string in Android? Explain your understanding, and even better to attach some sample code.**

   依據 Kotlin 官網 API 說明中，型別 String 有 4 種函式方法可以分割字串，分別為 chunked(), chunkedSequence(), split(), splitToSequence()，名稱相近的兩者差異僅為回傳型別不同，一為 List 集合，一為 Sequence 序列。

   I. chunked(), chunkedSequence()

      fun CharSequence.chunked(size: Int): List<String>

      將字串依固定字數做分割，包含保留空白字符

```
fun main (){
    val testStr = "Hello Android!";
    var result = testStr.chunked(3);
    println(result);
}
```

```
[Hel, lo , And, roi, d!]
```

   II. split(), splitToSequence()

      fun CharSequence.split(vararg delimiters: String/Char, ignoreCase: Boolean = false, limit: Int = 0): List<String>

      將字串依指定 String, Char, Pattern, Regex 做切割，包含保留空白字符

```
fun main (){
   val testStr = "Hello Android!" ;
   var result1 = testStr.split('o');
   var result2 = testStr.split("ll");
   println(result1);
   println(result2);

}
```

```
[Hell,  Andr, id!]
[He, o Android!]
```

3. **What kind of key-value pair collection do we use in Android? Attach a sample code that uses for-loop to print each key-value pair.**

在 Kotlin 常用的 Collection 集合中，Map 此類型介面，每筆資料都由一對 key/value 組成。在一個 Map 集合裡必須透過主鍵(Key)，才可操作相對應的元素值(value)，因此 key 不得重複，具有唯一性。Key/value 的型別宣告並無要求，但可透過 generic 泛型限制輸入型別。關於建立的 Map 集合資料，Library 提供各種宣告和增刪查改的對應方法，如果對相同的 key 操作，對應的 value 舊資料會被新資料取代。在 Kotlin 中，Map 介面被 AbstractMap 抽象類別和 MutableMap 介面繼承，最終由 HashMap 和 LinkedHashMap 兩類別 implement。

```
fun main (){
    val pi = 3.14;
    val testMap = mapOf(true to "真", "假" to false, "First" to "第一",
                        'A' to "甲上", 1 to "A", "pi" to pi, 1 to "Great!");
    for ((k, v) in testMap) {
        println("Key: $k -> Value: $v");
    }
}
```

```
Key: true -> Value: 真
Key: 假 -> Value: false
Key: First -> Value: 第一
Key: A -> Value: 甲上
Key: 1 -> Value: Great!
Key: pi -> Value: 3.14
```

4. **What's the difference between ArrayList and LinkedList? Try to explain in Mandarin.**

List 集合的特性是有特定的順序，且輸入的 value 可以重覆，因為依次排列可以使用 index 索引值或是 Iterator 查找 List 中的 value 值。ArrayList 和 LinkedList 兩者都 implement 了 List 介面，差別主要在資料的儲存方式及資料間的對應關係。ArrayList 的資料隨機存取基於動態陣列的延伸，對於每一空間都有明確索引值，因此查找修改效率較高；但對資料做新增刪除時，會移動所有集合節點以對應儲存空間，非常耗費時間，且經常性的預留空間，也佔據記憶體容量。LinkedList 的儲存方式著重於資料間的相互順序連結，利用索引值查找資料時，皆必須從頭(或尾)開始訪查，較不容易；不過，需要對資料進行增減時，因為相對資料關係的斷鏈及再連結，不需要對內存有任何位移消耗，節省許多時間。因此對於兩者的選擇，取決於操作的需求，以及程式提供的方法是否更有效率。在 kotlin 中只有提供 ArrayList 類別，如果要嘗試 LinkedList 必須藉由其他介面類別實作，或是呼叫 Java 的 LinkedList 類別。

5. **What are Value Type variables? List out the Value Type variables you know.**

   Value Type variables 是比較直觀的資料儲存方式，其變數對應的皆為 Value(實值)，內容儲存於 Stack(靜態內存)之中，在建立資料型別時，Stack 會分配一個空間來存放 Value。若將值複製到其他變數時，會將空間內的 Value 複製一份到新的變數空間，此後兩空間獨立運作互不干涉。和 Value Type 相對應的是 Reference Type，即大部分的類別型別儲存方式，其在 Stack 中儲存的僅為 Reference 參考值是記憶體的參考位址，此地址會指向動態 Heap 中資料真實儲存的空間。

   常見的 Value Type 有數值型別 Byte, Short, Int, Long, Float, Double, 布林型別 Boolean 和字元型別 Char，在 Java 和 Kotlin 當中，這些基本型別都有相對應的包裝類別(Wrapper Class)可做轉型彈性使用。

6. **What's the maximum value of Int? If you want to represent integers larger than the limit, what should you do instead of using int variables?**

   Int 的數值範圍為 -2147483648 ~ 2147483647，最大值為 2147483647，超過則會出現 overflow(溢位)的運算錯誤。預設數值會發生溢位時，可以使用 BigInteger 類別，透過 String 字串將龐大的數字做分段處理，而運算方式則必須搭配類別中的方法做使用。如果 kotlin 當中要轉型為此類別，Int 和 Long 類別分別也有 toBigInteger()方法可做使用。