

讀書會

- 組員：龍品勳、張耀文、時顯文
- 討論時間：2024/11/4 10:00 AM
- 地點：圖書館

照片：



心得報告

● 心得：Java 和 Kotlin 語法的差異

1. Null 安全性：

- Kotlin 有內建的 Null 安全，防止 `NullPointerException`。變數預設為不可為 Null，需使用 `?` 表示可為 Null，例如 `val name: String? = null`。
- Java 沒有內建的 Null 安全性，必須透過檢查或第三方工具來處理 `NullPointerException`。

2. 擴展函數（Extension Functions）：

- Kotlin 允許開發者通過擴展函數來為現有的類添加新功能，而無需繼承或修改原始類。例如，`fun String.isValidEmail(): Boolean { /*...*/ }` 可以直接在 `String` 類型上使用。
- Java 沒有擴展函數的功能，通常需要通過繼承、工具類或接口來實現類似效果。

3. 簡化的語法：

- Kotlin 提供了更簡潔的語法，減少冗長的代碼。例如，Kotlin 中的資料類（Data Class）可以通過 `data class` 一行定義，並自動生成 `toString`、`equals`、`hashCode` 等方法。
- Java 需要手動編寫這些方法，且語法相對較繁瑣。

4. Lambda：

- Kotlin 完全支援函數式編程，提供 Lambda 表達式和集合操作函數（如 `map`、`filter`），讓代碼更簡潔、直觀。
- Java 從 Java 8 開始才引入 Lambda 表達式，但在集合操作上不如 Kotlin 靈活，且函數式編程特性較有限。

5. 協程（Coroutines）：

- Kotlin 提供協程以支持異步和非阻塞操作，使得寫異步代碼更簡單，例如 `launch` 和 `async` 可以輕鬆實現非同步處理。
- Java 沒有原生的協程支持，必須依賴 `CompletableFuture`、`ExecutorService` 或第三方庫來實現異步處理。

● ChatGPT：

1. 請簡單比較物件導向及函式導向的差別：

物件導向（Object-Oriented Programming, OOP）與函式導向（Functional

Programming, FP) 是兩種常見的編程範式，各有不同的特點和適用情境：

- 基礎概念

- 物件導向：OOP 將程式分解成「物件」，物件是由資料（屬性）和操作這些資料的行為（方法）組成。程式是透過物件的交互來運作的。
- 函式導向：FP 將程式視為一組函式的組合，函式是純粹的，即相同輸入總是產生相同輸出。程式的運行是函式的應用與組合。

- 核心思想

- 物件導向：重視「物件」和「封裝」。OOP 主要關注如何將資料和行為封裝在一個個的物件中，並利用繼承、多型等特性來增強系統的彈性和擴展性。
- 函式導向：重視「純函數」和「不可變性」。FP 偏重用函數處理資料，盡量避免副作用，並鼓勵不可變性，使程式更具可預測性和易於並行處理。

- 狀態管理

- 物件導向：OOP 透過物件的屬性來儲存狀態，各個物件的狀態可能會隨著操作而改變。
- 函式導向：FP 通常避免可變狀態，傾向使用不可變資料結構和純函數來避免狀態改變。

- 優缺點

- 物件導向：
 - 優點：適合大型系統，易於模組化，方便維護與擴展。
 - 缺點：對於小型且數學計算密集的應用可能顯得笨重。
- 函式導向：
 - 優點：易於測試和調試，適合數學運算、資料轉換和併發處理。
 - 缺點：對於習慣 OOP 的人來說，可能較難理解且缺少面向物件的組織性。

- 典型語言

- 物件導向：Java、C++、Python
- 函式導向：Haskell、Erlang、F#

總結來說，物件導向更注重封裝性和對象間的關係，而函式導向則著重函數純度和不可變性，兩者在不同應用場景中各有優勢。

2. 請簡單比較指標及參考的差別：

指標（Pointer）與參考（Reference）都是在程式語言中用來操作記憶體地址的

機制，但兩者有一些重要差別：

- 定義與用途

- 指標：是一個變數，儲存另一個變數的記憶體地址，可以用來直接操作記憶體。
- 參考：是一個變數的別名（或映射），本身並不儲存記憶體地址，而是另一個變數的「參考」或「代理」。

- 可修改性

- 指標：可以被重新指向不同的地址，即可以改變指向的對象。
- 參考：一旦初始化後，就無法再重新指向另一個對象，是一種不可改變的連結。

- 記憶體運算

- 指標：可以進行地址運算（如位移操作），以存取不同的記憶體位置。
- 參考：無法直接操作地址，也不支援指標運算。

- 空指標與 **NULL**

- 指標：可以指向 **NULL** 或空地址，表示它目前不指向任何有效物件。
- 參考：必須指向有效的物件，無法指向 **NULL**，初始化時必須綁定一個有效變數。

- 語法和使用場景

- 指標：使用 ***** 來定義，並通過 **&** 來取得變數地址。主要用於需要直接操作記憶體的場景，如系統編程和資料結構。
- 參考：使用 **&** 來定義和取得引用。主要用於簡化代碼，避免不必要的拷貝操作（例如 **C++** 中傳遞參考參數）。

- 安全性與易用性

- 指標：操作不當容易引發空指標或記憶體洩漏問題，因此需要更謹慎的記憶體管理。
- 參考：相對安全，沒有空指標和指標位移等問題，語法上也更簡潔。

總結

指標提供靈活的記憶體操作能力，但需要小心管理；參考則是一種更高層次的抽象，使用上更安全和簡單。

3. 請問 Android Studio 是由誰研發的：

Android Studio 是由 **Google** 研發的。它於 2013 年 Google I/O 大會首次推出，旨在成為 Android 應用程式開發的官方整合開發環境 (IDE)。Android Studio 是基於 JetBrains 的 IntelliJ IDEA 平台進行開發的，Google 在這個基礎上加入了許多專為 Android 開發設計的功能，如模擬器、即時預覽、UI 編輯器、APK 分析工具等，讓開發者更容易創建、測試和優化 Android 應用。

- GitHub 程式連結：<https://github.com/WoemTheCat/Kotlin-Homework2>

- GitHub 的截圖：



