Effective Kotlin 讀書會 S1E2
補充資料

關注第一梯次 Effective Kotlin 讀書會活動:
https://tw.kotlin.tips/study-jams/6

共讀書本及討論群

- 導讀投影片:第一章 第二章
- 活動錄影:
- 共讀書本:https://leanpub.com/effectivekotlin
- 讀書會討論群
    - Line 群 https://line.me/ti/p/-gPLrcTieE (請先加聖佑朋友後，傳訊息給他，會再邀請入群)
    - Telegram 群 https://t.me/joinchat/DAUjOBrfu7_05sa9YYb5FQ

導讀筆記

- Item 7 返回值可能有例外時，優先使用null或 Failure
    - 需要額外訊息使用 failure，其他return null。
    - List 用 getOrNull 超出範圍不會拋錯會回傳 null
- Item 8:處理Nulls的屬性
    - 使用?安全呼叫、使用if條件判斷null、使用!!
    - 預防性和攻擊性的程式設計
        - 使用?.條件作為防禦，例如?.print()
        - Require、check、assert等方法作為攻擊性編程
    - 當我們對未能達成的事抱有強烈期望時，最好拋出例外已告知開發人員。
        - throw、非空斷言(!!)、requireNotNull、checkNotNull)
    - 使用!!的問題
        - Nullability訊息被忽略了，可能導致之後接手的開發人員重構上會有問題
    - 應該盡量避免使用!!，但它並不是規範，需要小心看待!!這件事
    - 處理集合(List等)時，返回空列表，而不是null
    - 一定會被初始化才用 lateinit
    - Unpack指的是，在使用前需要判斷是否為空
    - 補充 Unpack：Unpack 是 Monad,可以把 Optional 想成盒子，值在盒子裡面
    - JVM 上的原型資料不能使用 lateinit，要改用 Delegates.notNull
- Item 9 關閉資源用 use
    - 使用Resource時，使用try-finally調用close的方法，達到安全的關閉使用。但這樣的結構是複雜又不正確的(拋出 error 是無法被正確抓到)。

    - 使用use()來進行Resource的關閉(必須是Closeable的物件、AutoCloseable)是一件安全又簡單的選擇
    - 補充 Resource : 沒有正確關閉檔案資源，就有可能讓資源卡在那邊。不一定會被GC掉
- Item 10：寫單元測試
    - 對工程師而已，首要任務就是確保程式可以正常運作
    - 最好的方式就是寫測試來檢查程式，尤其是單元測試

第二章 可讀性

- Item 11 : design for readability
    - Less code syntax, more common structure
    - Debug tool and refactor tool are build for common structure
    - Reducing cognitive load
- Item 12 : Operator meaning should be consistent with its function name
    - Do not override it because you can,operator could use for increase readability, but also the opposite
- Item 13 : Use operator to increase readability
    - Common use case to override operator is DSL, unit of measure, money wrapper, other kind of number
    - 精度：小數位精度可以用 BigDecimal 來判別不同精度 (1.0 1.00 不同精度)，常用在 測量單位、金額類別、時間單位
- Item 14 : Specific variable type when it is not clear
    - Cost a little, but can help a lot
    - Avoid accidently infer type changed
    - Helpful when code review or pr
- Item 15 : consider referencing receiver explicitly
    - Choose proper receiver function to increase readability
    - Use label to mark the receiver
    - Respect the design of DSL
- Item 16 : Properties should represent state, not behavior
    - properties represent accessors, not field
    - Getter, setter allow variable represent result of behavior
    - The difference between variable and function is, we expect reading a variable so be quick, 適合用 function 情境：操作複雜 、有商業邏輯 、有預期的行為
- Item 17 : consider naming argument
    - independent of order
    - Use it when
        - with default argument
        - with same type as other parameter
        - receive lambda, when it is not the last argument
- Item 18 : Respect coding conventions
    - https://kotlinlang.org/docs/coding-conventions.html
    - Just do it !!

- 講者補充的
    - 可以用 function programming style 完成裝飾者模式，能使程式碼結構更好讀，避免像是 callback hell 的結構，但是在混用 proxy, decorator 兩種模式時，因為 proxy 可能會改變行為，要特別注意順序
- 其他
    - 可讀性特點：註解、有意義的命名、好猜
    - 有註解還是沒註解好？有註解可能比較好懂，但沒更新的註解會騙人
    - 不要只為有經驗的工程師寫 code，降低認知負荷（不熟悉語法者也看得懂）
    - 精度：小數位精度可以用 BigDecimal 來判別不同精度 (1.0 1.01 不同精度)，常用在 測量單位、金額類別、時間單位
    - 類型推斷很方便，但在PR或MR會造成不好閱讀
    - 16. Property is not just field：kotlin 每個變數都有 getter setter，使用 property前都可以做處理再回傳值
    - 適合用 function 情境：操作複雜 、有商業邏輯 、有預期的行為
    - 用 chain 的方式取代 一層層的block .let(::Print)，這種方要注意順序性。
        - 建議使用 Infix function 而非 override operator

參考資料

-