【Kotlin】seale class 與 enum 的比較

差異比較

特性	Sealed Class	Enum
使用情景	一組相關但不同的狀態或事件	一組固定的常量值
子類數量	可以有多個不同類型的子類	固定數量的常量
可擴展性	子類必須在相同文件中擴展	無法擴展，固定常量
狀態	可以有狀態（文字和屬性）	通常是不可變的
設計靈活性	更靈活，可以有複雜的層次結構	固定簡單
安全性	編譯期檢查未處理的分支	固定選項無需額外檢查

```
enum class MachineState {
    START, SLEEP, RUNNING
}


fun machineStateChecker(state: MachineState) = when (state) {
    MachineState.START -> "Machine is about to start"
    MachineState.SLEEP -> "Machine is sleeping"
    MachineState.RUNNING -> "Machine is running"
}

fun main(argv: Array<String>) {
    val machineState = MachineState.SLEEP
    val state =  machineStateChecker(machineState)
    println("current state = $state")
    //current state = Machine is sleeping
}
```

```
sealed class WorkingState {
    // data class
    data class Finished(val result: List<String>): WorkingState()
    data class ErrorHappened(val whatHappened: String): WorkingState()
    // object
    object Working: WorkingState()
    object EmptyResult: WorkingState()
}


fun machineStatePrinting(workingState: WorkingState) {
    when (workingState) {
        is WorkingState.Finished -> {
            if (workingState.result.isNotEmpty()) {
                for ((index, text) in workingState.result.withIndex()) {
                    println("the ${index + 1} result is $text")
                }
            }
        }
        is WorkingState.ErrorHappened -> {
            println("""
                Error Occurred: reason is
                ${workingState.whatHappened}
```

```
            """.trimIndent())
        }
        WorkingState.Working -> {
            println("Printing machine is working...")
        }
        WorkingState.EmptyResult -> {
            println("It's empty result.")
        }
    }
}

fun main(argv: Array<String>) {
    val machineState =
        WorkingState.Finished(
            mutableListOf(
                "print doc 1",
                "print doc 2",
                "print doc 3",
                "print doc 4"
            )
        )
    machineStatePrinting(machineState)
  // the job1 result is print doc 1
 // the job2 result is print doc 2
    // the job3 result is print doc 3
    // the job4 result is print doc 4

    val secondMachineState = WorkingState.ErrorHappened("no papper")
    machineStatePrinting(secondMachineState)
  // Error Occurred: reason is no papper

    val thirdMachineState = WorkingState.Working
    machineStatePrinting(thirdMachineState)
  // Printing machine is working...
}
```

來源：https://louis383.medium.com/kotlin-sealed-classes-%E7%9A%84%E5%9F%BA%E7%A4%8E%E4%BD%BF%E7%94%A8-de660dbb63d2

修訂版本 #7
★由 treeman 建立於 11 C 2024 10:00:04
由 treeman 更新於 31 C 2024 10:36:13