**Lab 7- Multiple Try-Catch in Kotlin**

Below is a lab exercise on handling multiple exceptions using multiple try-catch blocks in Kotlin. This exercise focuses on understanding how to handle different types of exceptions in distinct catch blocks. Feel free to use an IDE or an online Kotlin compiler to run the code.

fun performDivision(a: Int, b: Int): Int {

return try {

// Try block for division

a / b

} catch (e: ArithmeticException) {

// Catch block for division by zero

println("Division by zero exception: ${e.message}")

-1 // Default value for division by zero

} catch (e: IllegalArgumentException) {

// Catch block for other exceptions

println("Other exception: ${e.message}")

-1 // Default value for other exceptions

}

}

fun main() {

// Task 2: Call the 'performDivision' function with different numbers and print the results.

val result1 = performDivision(10, 2)

println("Result 1: $result1") // Expected: 5

val result2 = performDivision(8, 0)

println("Result 2: $result2") // Expected: -1 (division by zero)

val result3 = performDivision(15, 3)

println("Result 3: $result3") // Expected: 5

// Task 3: Create a function named 'validateInput' that takes a string as a parameter

// and validates if it contains only numeric characters. Handle NumberFormatException separately.

fun validateInput(input: String): Int {

return try {

// Try block for parsing

input.toInt()

} catch (e: NumberFormatException) {

// Catch block for invalid input

println("Invalid input exception: ${e.message}")

-1 // Default value for invalid input

}

}

// Task 4: Call the 'validateInput' function with different inputs and print the results.

val result4 = validateInput("123")

println("Result 4: $result4") // Expected: 123

val result5 = validateInput("abc")

println("Result 5: $result5") // Expected: -1 (invalid input)

}

**Instructions:**

* Create the function 'performDivision' as instructed in the comments.
* In the main function, call the 'performDivision' function with different numbers and print the results.
* Create the function 'validateInput' as instructed in the comments.
* Call the 'validateInput' function with different inputs and print the results.

This exercise is designed to reinforce your understanding of multiple try-catch blocks for handling different types of exceptions in Kotlin.