**Lab 6- Try-Catch in Kotlin**

Below is a lab exercise on try-catch in Kotlin. This exercise focuses on understanding how to use try-catch blocks for exception handling in Kotlin.

// Lab Exercise: Try-Catch in Kotlin

// Task 1: Create a function named 'divide' that takes two integers as parameters

// and returns the result of dividing the first number by the second number.

// Handle the division by zero exception using try-catch and return a default value in such cases.

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

return try {

a / b

} catch (e: ArithmeticException) {

println("Exception caught: ${e.message}")

-1 // Default value for division by zero

}

}

fun main() {

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

val result1 = divide(10, 2)

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

val result2 = divide(8, 0)

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

val result3 = divide(15, 3)

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

}

**Instructions:**

* Create the function 'divide' as instructed in the comments.
* In the main function, call the 'divide' function with different numbers and print the results.
* Observe how the try-catch block handles the division by zero exception.

This exercise is designed to reinforce your understanding of try-catch blocks for exception handling in Kotlin. Feel free to experiment with additional scenarios and modify the code as needed.