**Lab 8- Throw Exception in Kotlin**

Below is a lab exercise on throwing exceptions in Kotlin. This exercise focuses on how to manually throw exceptions using the throw keyword.

// Lab Exercise: Throw Exception in Kotlin

// Task 1: Create a function named 'validateAge' that takes an age as a parameter

// and throws an IllegalArgumentException if the age is negative.

fun validateAge(age: Int) {

if (age < 0) {

throw IllegalArgumentException("Age cannot be negative")

}

}

fun main() {

// Task 2: Call the 'validateAge' function with different ages and handle the exceptions.

try {

validateAge(25)

println("Age is valid.")

} catch (e: IllegalArgumentException) {

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

}

try {

validateAge(-5)

println("Age is valid.")

} catch (e: IllegalArgumentException) {

println("Exception caught: ${e.message}") // Expected: Age cannot be negative

}

// Task 3: Create a function named 'divide' that takes two integers as parameters

// and throws an ArithmeticException if the second number is zero.

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

if (b == 0) {

throw ArithmeticException("Cannot divide by zero")

}

return a / b

}

// Task 4: Call the 'divide' function with different numbers and handle the exceptions.

try {

val result1 = divide(10, 2)

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

} catch (e: ArithmeticException) {

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

}

try {

val result2 = divide(8, 0)

println("Result 2: $result2") // Expected: Cannot divide by zero

} catch (e: ArithmeticException) {

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

}

}

**Instructions:**

* Create the function 'validateAge' as instructed in the comments.
* In the main function, call the 'validateAge' function with different ages and handle the exceptions.
* Create the function 'divide' as instructed in the comments.
* Call the 'divide' function with different numbers and handle the exceptions.

This exercise is designed to reinforce your understanding of throwing exceptions in Kotlin using the throw keyword.