Exception Handling Lab 1

Objectives

1. In this lab you will practice writing code that produces exceptions
2. This lab will introduce the practice of recovering from exceptions by handling them
3. After completing this lab students should be able to catch exceptions.

Overview

This lab will introduce the use of exceptions including throwing, catching, testing for, and defining exceptions.

Unit tests

For each method described below (excluding main), write tests to confirm that it performs the desired operation appropriately. If a method can produce an exception you should test for that behavior as well.

Instructions

Create a Calculator class with four static methods: `add` `subtract` `multiply` and `divide`. Each method should take two integers and return the result of the appropriate mathematical operation.

The `divide` method might throw an exception. Add that to its method header.

In a separate app class, write a main method that calls each of the `Calculator` methods at least twice. One of the calls to `divide` should cause division by zero. Your main method should include a try-catch block to handle this gracefully by printing an error message and continuing execution.

Part 2:

Modify your `divide` method to explicitly check for zero as the divisor, and explicitly throw an exception if this happens. It should still pass your tests and return the correct result for valid input values.

Part 3:

Define your own custom exception type, called `DivisionByZeroException` and modify your `divide`

method to throw this type. Adjust your main method and tests to be as specific as possible to catch or expect only this type of exception.

Part 4: (Optional Challenge)

Add a method called squareRoot. It should take one number, return its square root, and throw a `ComplexNumberException` if the result would be a complex number.