Exercise 1: The Basics (Division Safety)

Goal: Prevent a program crash when dividing by zero.

- **Task:** Create a program that takes two integers as input from the user.
- **Requirement:** Wrap the division logic in a try-catch block. Catch the specific exception for "Division by Zero" (e.g., ArithmeticException in Java/C# or ZeroDivisionError in Python).
- **Output:** If the user enters 0 as the divisor, print "Error: You cannot divide by zero." Otherwise, print the result.

Exercise 2: The "Finally" Guarantee

Goal: Understand that certain code *must* run, regardless of success or failure.

- **Task:** Simulate opening a database connection.
    1. Print "Opening connection..."
    2. Inside a try block, throw a random exception (e.g., "Data corruption detected").
    3. Catch the exception and print "Handling error: " + the message.
    4. Use a finally block to print "Closing connection...".
- **The Test:** Verify that "Closing connection" prints even though the program encountered an error.

Exercise 3: Multiple Catch Blocks

Goal: Learn how to handle different types of failures differently.

- **Task:** Create an array of 5 integers. Ask the user for an **index** and a **number** to divide that element by.
- **Requirement:** Use multiple catch blocks to handle:
    - ArrayIndexOutOfBoundsException: If the user asks for index 10.
    - NumberFormatException: If the user types "ABC" instead of a number.
    - ArithmeticException: If the user divides by zero.

Exercise 4: Custom Exceptions

Goal: Create your own business logic errors.

- **Task:** Create a "Bank Account" class with a withdraw method.
- **Requirement:** 1. Define a custom exception class named InsufficientFundsException. 2. If a user tries to withdraw more money than they have, throw your custom exception. 3. In your main program, catch that specific exception and suggest they deposit more money.

Exercise 5: The "Rethrow" Pattern

Goal: Learn how to catch an exception, log it, and then pass it up to a higher-level controller.

- **Task:** Create two functions: processData() and main().
    1. **processData()**: Try to parse a string that isn't a number. In the catch block, print "Log: Error occurred in processData." Then, use the throw keyword to re-throw that same exception object.
    2. **main()**: Call processData() inside its own try-catch block. Catch the re-thrown error and print "Final Catch: The application has safely stopped."
- **The Lesson:** This teaches you how to perform "local" cleanup (like logging) while still letting the "global" program know something went wrong.