Exception

Exercise #01

Flutter Developer Bootcamp

# **Purpose**

The purpose of this exercise is likely to demonstrate how exception handling works in Dart, including how to define custom exceptions and handle different types of exceptions gracefully. It also illustrates the importance of error handling to maintain program robustness and stability.

# **Problem**

* Declare a variable, with name x inside a single inverted comma(string).
* Declare an Integer with name y and give any value.
* Divide x with y and get the result with the following values.

|  |  |
| --- | --- |
| x | **y** |
| ‘123’ | 5 |
| ‘al’ | 6 |
| 5 | 0 |
| 6 | 3 |

* Get all results and if the results are odd, is less than 5, create a custom exception and print the result is less than 5

The instructor will change values, the only objective is that the code must not break, so for that the program must be in a way to handle all kinds of exceptions.

# **How to Solve**

1. Checkout the workshop from Git Repo:

git clone -b <user-branch> <repo-URL>

2. Open the root folder inside VS Code

3. Open the root folder in terminal

4. Run the command dart run filename.dart

5. The code imports the dart:io library for input/output.

6. It defines a class Exceptionhandling that implements the Exception interface.

7. Inside Exceptionhandling, a message variable data is initialised.

8. The main function begins, taking a list of string args.

9. Inside the main function, there's a try block:

* It attempts to parse a string 'gi' into an integer, which would throw a FormatException.
* It performs an integer division, potentially throwing a IntegerDivisionByZeroException.
* If the result of the division is less than 5, it throws a custom exception.

10.Following the try block, there are several catch blocks:

* One for FormatException prints "format is incorrect".
* One for Exceptionhandling prints a predefined message.
* One for IntegerDivisionByZeroException prints "integer division zero number".
* The final catch block catches any other exceptions and prints their messages.

11. If no exceptions occur, the program continues execution after the catch blocks.

complain.

# **You Will Achieve**

This Dart code demonstrates exception handling in a simple scenario. Here's what it achieves:

1. Custom Exception Class: The Exception Handling class is created, which implements the Exception interface. It serves as a custom exception class.
2. Exception Handling: The main function utilises try, catch, and on blocks to handle different types of exceptions:

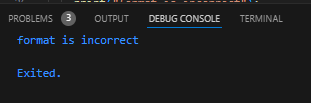
* try: This block wraps the code that might throw exceptions.
* catch: This block catches exceptions of specific types.
* on: This block catches exceptions of specific types that are subclasses of Exception.

1. Parsing: It attempts to parse a string x to an integer x1. If the parsing fails due to a FormatException, it's caught and handled.
2. Division: It performs an integer division operation (x1 ~/ y). If a division by zero occurs (IntegerDivisionByZeroException), it's caught and handled.
3. Custom Exception Handling: If the division result is less than 5, it throws a custom Exception Handling exception.
4. Output: Depending on the scenario, different messages are printed to the console to indicate the type of exception caught or the result of the operation.
5. Methods and Functions:

* The Exception Handling class contains a method data1(). However, this method is defined but not used in the provided code.

# **Screenshots**

## **Expected Output**



# **How to submit your Exercise**

Push your project back to the same git branch using command:

<command name>

# **Happy Coding!**