Exception Handling Fundamentals & Techniques

Introduction & Importance

What are exceptions?

- Anomalous conditions that disrupt program flow.
- Can be caused by various factors (e.g., user input, resource errors, network issues).

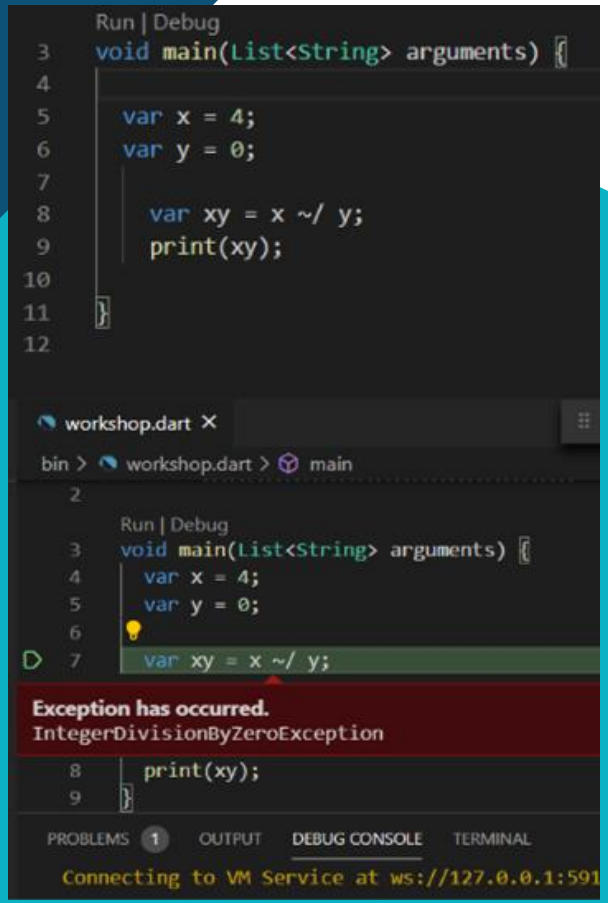
Importance of exception handling

- Ensures program stability and prevents crashes.
- Allows for graceful handling of errors.
- Improves application robustness and user experience.

Benefits of proper exception handling

- Increased code quality and maintainability.
- Easier debugging and troubleshooting.
- Improved error reporting and logging.
- Provides a more predictable and reliable program execution.

Exception



Here, We Programmed a execution to divide a variable x with y, but the user given is " x as 4 and y as 0 "

In ordinary arithmetic, the expression has no meaning, as there is no number which, when multiplied by 0, gives a (assuming a ≠ 0), and so division by zero is undefined.

So it will be a problem during execution.which automatically terminate the program.

So for those problems that rise during executions are called exception

And solving that is exception handling

Handling Exceptions

try, try block will carry the statement that has the chance to get an exception.

on, on block is set to know the specific exception and apply certain statements.

catch, catch block is set when the program doesn't know what exception occured (because no on block initiated), but the program must not break.

finally, finally block is the only block which will execute weather the exception is there or not.

```
try{
   //code that might throw an exception
}
on Exception1 {
  //code for handling exception
}
catch Expection2 {
  //code for handling exception
}
```

Examples:

In this code :
Case1 : When you know the exception is thrown , use ON
Case2: When you don't know the exception use catch

```
Run | Debug
1  void main(){
2    print("CASE 1:");
3    try{
4      int result =10~/0;
5      print("The result is $result");
6    }on IntegerDivisionByZeroException{
7      print("Cannot divided by zero");
8    }
9
10   print("");
11   print("CASE 2:");
12   try{
13     int result =10~/0;
14     print("The result is $result");
15   }catch(e){
16     print("The expection throw is $e");
17   }
18   }
```

```
PROBLEMS 41  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
CASE 1:
Cannot divided by zero

CASE 2:

Exited.
The expection throw is IntegerDivisionByZeroException
```

Finally case3:

```
19   print("");
20   print("CASE 3:");
21   try{
22     int result =10~/0;
23     print("The result is $result");
24   }catch(e){
25     print("The expection throw is $e");
26   }finally{
27     print("This is finally clause");
28   }
29   }
30
```

```
PROBLEMS 41  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
CASE 3:
The expection throw is IntegerDivisionByZeroException
This is finally clause
```

Sr.No	Exceptions & Description
1	**DeferredLoadException** Thrown when a deferred library fails to load.
2	**FormatException** Exception thrown when a string or some other data does not have an expected format and cannot be parsed or processed.
3	**IntegerDivisionByZeroException** Thrown when a number is divided by zero.
4	**IOException** Base class for all Inupt-Output related exceptions.
5	**IsolateSpawnException** Thrown when an isolate cannot be created.
6	**Timeout** Thrown when a scheduled timeout happens while waiting for an async result.

Custom Exception

Custom exceptions are exceptions we create.

For example, Think if a problem rises when the device is getting hot, but managing that problem is not an exception because that type of exception is not there, but it's a problem.So for that we will create an exception for the problems that are not listed in the types of exceptions.

But for creating an custom exception we must know about some keywords

- class
- constructor
- implements
- throw

class- class is where we group items related to a base item, in other words its an library.

constructor- constructor is responsible for allocating memory for new objects inside a class.

Implements- An implementation class defines the physical implementation of objects of a class.

throw- throw keyword is used to explicitly raise an exception. A raised exception should be handled to prevent the program from exiting abruptly.

(detailing of these keywords will be found in following modules)