Exception Handling

Mistake in Taking Input

Suppose, that you have developed a Business Application, and your Client is using that application. Everything is working fine and he has entered a lot of data.
At one point, by mistake he has to enter an integer value but he entered a special character.

Mistake in Taking Input

Your application will crash while the Client is doing important work, especially the data is lost.

Invalid Calculation

Lets say, now the Client is performing some calculations, and he has divided some value with zero.

```
static void Main(string[] args)
{
    int x = 0;
    int div = 100 / x;
    Console.WriteLine(div);
    Console.ReadKey();
}
```

Invalid Calculation

Lets say, now the Client is performing some calculations, and he has divided some value with zero.

Again..

Exceptions

These are what we call Exceptions i.e., an event that occurs during the execution of a program and that disrupts the normal flow of instructions.

Exceptions

If exceptions are not handled, programs may crash and data may be lost. This can be very frustrating if it happens repeatedly, you lose lot of data every time.

Exceptions: How to detect?

The standard practice is to record all events in the application log file. The log file acts as a time machine helping you to go back in time to view all the phases the application went through and what led to the exception.

Exceptions: How to Handle?

In C#, exception handling is done with the following keywords:

1. try
2. catch

Exceptions: Working Example

In C#, exception handling is done as follows.

```
static void Main(string[] args)
{
    int x = 0;
    int div = 0;
    try
    {
        div = 100 / x;
        Console.WriteLine("This line is not executed");
    }
    catch (Exception e)
    {
        Console.WriteLine("Exception occured: " + e.Message);
    }
    Console.WriteLine("Result is + ", div);
    Console.ReadKey();
}
```

Exceptions: Working Example

In C#, exception handling is done as follows.

It means Try this code

```
static void Main(string[] args)
{
    int x = 0;
    int div = 0;
    try
    {
        div = 100 / x;
        Console.WriteLine("This line is not executed");
    }
    catch (Exception e)
    {
        Console.WriteLine("Exception occured: " + e.Message);
    }
    Console.WriteLine("Result is + ", div);
    Console.ReadKey();
}
```

Exceptions: Working Example

In C#, exception handling is done as follows.

```
static void Main(string[] args)
{
    int x = 0;
    int div = 0;
    try
    {
        div = 100 / x;
        Console.WriteLine("This line is not executed");
    }
    catch (Exception e)
    {
        Console.WriteLine("Exception occured: " + e.Message);
    }
    Console.WriteLine("Result is + ", div);
    Console.ReadKey();
}
```

If Exception occurs then catch it

Exceptions: Working Example

Output on the Console.

```
Exception occured: System.DivideByZeroException: Attempted to divide by zero.
   at Logging.Program.Main(String[] args) in G:\OOP 2022\Logging\Logging\Program.cs:line 18
Result is 0
```

Logging

This is just the basic on Exception Handling.

For further reading, Explore this link:

https://www.c-sharpcorner.com/article/exception-handling-in-C-Sharp/

Conclusion

- An exception is a problem that arises during the execution of a program.
- A C# exceptional handling is a response to an exceptional circumstance that arises while a program is running

Learning Objective

Keep the program running instead of Crashing through Exception Handling