Chapter 4. Exception Handling

A reference of MSDN Library for Visual Studio 2017

IT Faculty, TDM University

Contents

Introduction

- An exception is any error condition or unexpected behavior encountered by an executing program.
- Exceptions can be raised because of a fault in your code or in code you call (such as a shared library), unavailable operating system resources, unexpected conditions the common language runtime encounters (such as code that cannot be verified), and so on.

Introduction

- In the .NET Framework, an exception is an object that inherits from the **Exception** Class class. An exception is thrown from an area of code where a problem has occurred. The exception is passed up the stack until the application handles it or the program terminates.

Exception Class

- The Exception class is the base class from which exceptions inherit. Most exception objects are instances of some derived class of **Exception**, but you can throw any object that derives from the Object class as an exception.

Exception Class

- Common Properties
  - **`Message`**: Provides details about the cause of an exception.
  - **`StackTrace`**: Contains a stack trace that can be used to determine where an error occurred. The stack trace includes the source file name and program line number if debugging information is available.

Try statement

- A common usage of **catch** and **finally** together is to obtain and use resources in a **try** block, deal with exceptional circumstances in a **catch** block, and release the resources in the **finally** block

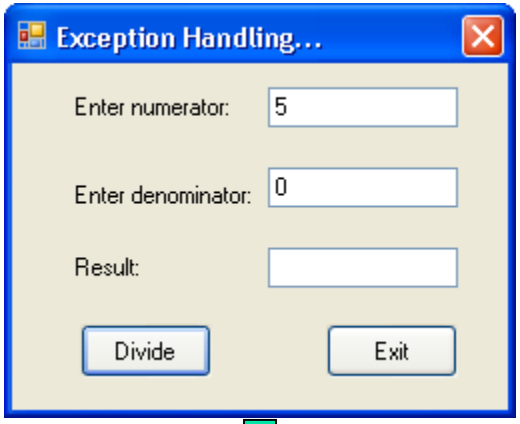
try statement

- try ...catch ... finally Statement

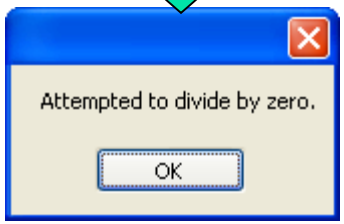
```
try
 {
   1. Include codes in which exceptions might occur.
 }
 catch (Exception )
 {
   2. Represent types of exceptions the catch can handle.
 }
 finally
 {
   3.(Optional) codes present here will always execute.
 }
```

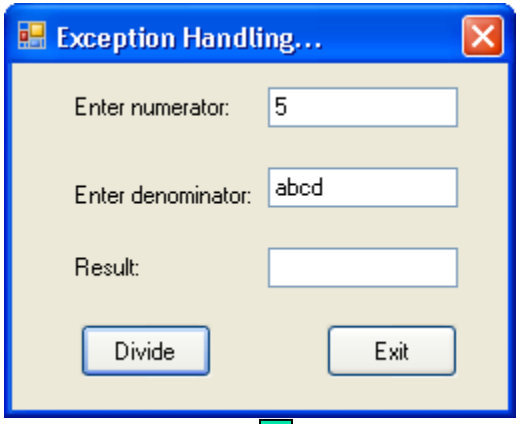
Example

- Example: Divide by zero

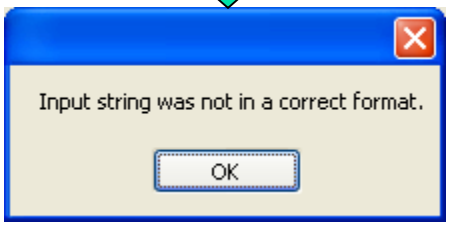


Error Message





Error Message



Example

- Code pattern

```
private void btnDivide_Click(object sender, EventArgs e)
    {
      textBox3.Clear();
      try
       {
         int a = Convert.ToInt32(textBox1.Text);
         int b = Convert.ToInt32(textBox2.Text);
         int c = a / b;
         textBox3.Text = c.ToString();
       }
      catch(Exception ex)
       {
          MessageBox.Show(ex.Message);
       }
    }
```