(T7)TryCatch、CustomException、SqlException、FormatException、OverflowException
CourseGUID: 29f1196a-1950-41a4-b9c1-dd13a9e92d92
==========
(T7)TryCatch、CustomException、SqlException、FormatException、OverflowException
==========

==========

0. Summary

1.
System.Exception

1.1.
An exception is a Sub Class of System.Exception Class which
represents errors during a program is running.
1.1.1.
System.Exception.Message is description of exception.
1.1.2.
System.Exception.StackTrace is stack information of the exception.

1.2.
System.Exception SubClass
1.2.1.
System.FileNotFoundException :
  Occurs when read from a file that does not exist.
1.2.2.
System.SqlException :
  Occurs when read from a database table that does not exist
1.2.3.
System.FormatException
  Occurs when enter a Character into int type variable.
1.2.4.
System.OverflowException
  Occurs when enter a very big number ,that an integer cannot hold, into the int type variable.
1.2.5.
System.DivideByZeroException
  Occurs when there is an attempt to divide an integral or Decimal value by zero.

1.3.
Syntax:

```
//try
//{
//    //The code that can possibly cause an exception
//}
//catch (FileNotFoundException ex)
//{
//    //FileNotFoundException Exception Handing.
//    Console.WriteLine("{0} can not be found. ", ex.FileName);
//}
//catch (Exception ex)
//{
//    //Handles the rest of exception
//    Console.WriteLine(ex.Message);
```

```
//    Console.WriteLine(ex.StackTrace);
//}
//finally
//{
//    //The code that always executed no matter the exception is occured or not
//}
```

2.
Customize Exceptions

2.0.
Sometimes there is no existing system exception can fit your situation.
In this case, you might have to create your own Customize Exceptions.
E.g.
This online game does not allow user to use VPN

2.1.
// public class NoVpnConnectionException : Exception
The Customize Exceptions must extend the System.Exception class.
In addition the suffix should be "Exception" to fulfill the naming convention.

2.2.
// public NoVpnConnectionException(string message) : base(message)
The Customize Exceptions should always provide a public constructor which take a string message parameter,
and then pass it to base class constructor.

2.3.
// public NoVpnConnectionException(string message, Exception innerException) : base(message, innerException)
The Customize Exceptions should always provide the capability to handle inner exception.

2.4.
//[Serializable]
//public class NoVpnConnectionException : Exception
...
//public NoVpnConnectionException(SerializationInfo info, StreamingContext context): base(info, context)
The Customize Exceptions should always support Serializable,
thus, it can work across application domains.

3.
Programmers should never do Exception Handling Abuse
which means using exception to implement programming logic.

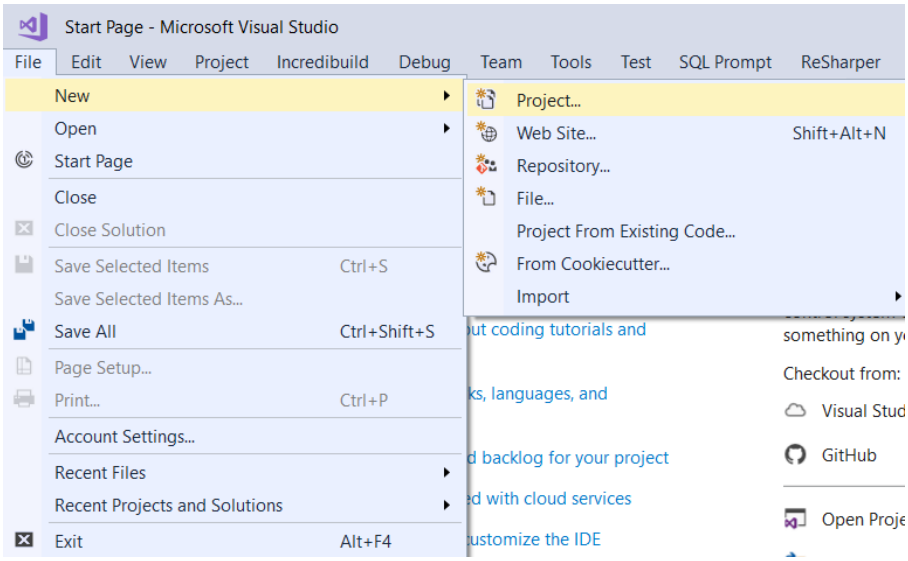
==========

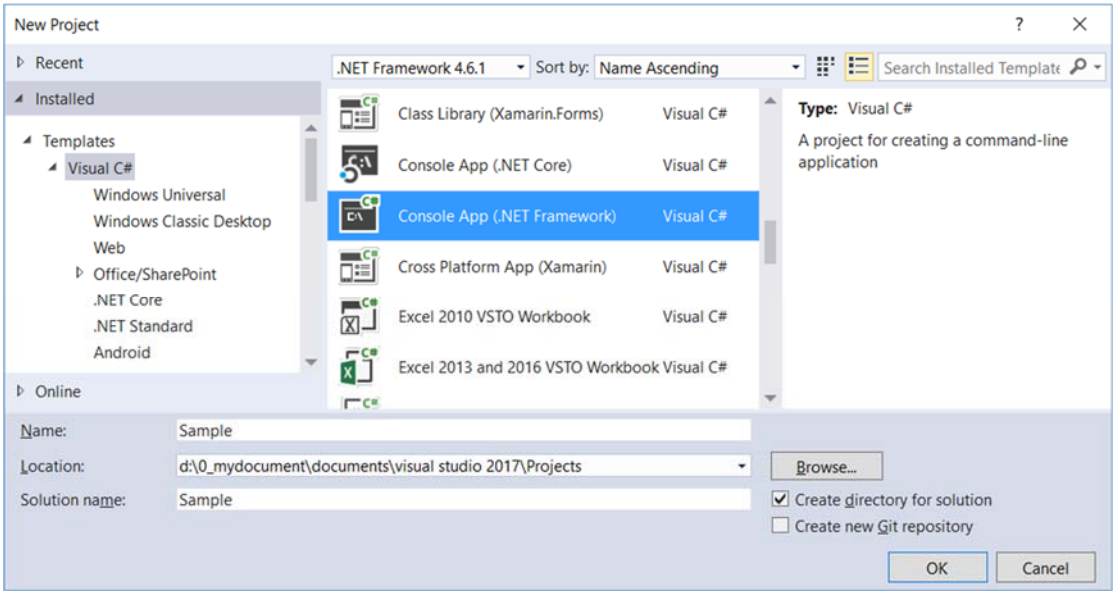
1. Create New Project

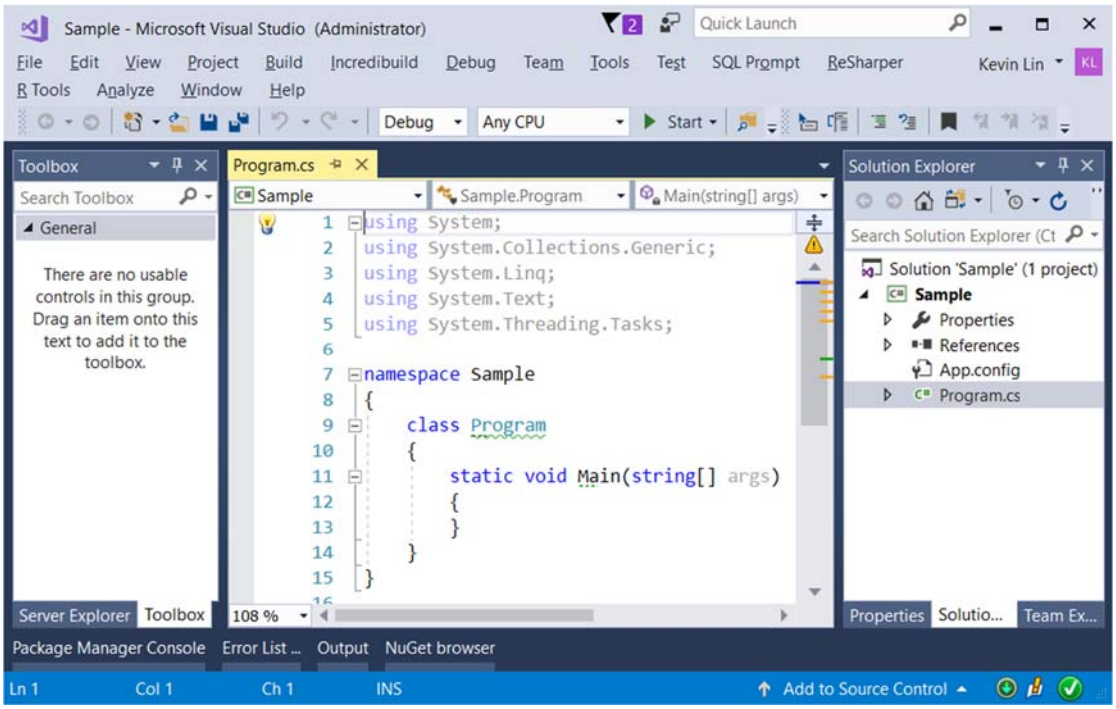
File --> New --> Project... -->

Visual C# --> **Console App (.Net Framework)** -->

Name: **Sample**


Start Page - Microsoft Visual Studio
File Edit View Project Incredibuild Debug Team Tools Test SQL Prompt ReSharper
New
Open
Start Page
Close
Close Solution
Save Selected Items Ctrl+S
Save Selected Items As...
Save All Ctrl+Shift+S
Page Setup...
Print... Ctrl+P
Account Settings...
Recent Files
Recent Projects and Solutions
Exit Alt+F4
Project...
Web Site... Shift+Alt+N
Repository...
File...
Project From Existing Code...
From Cookiecutter...
Import
Checkout from:
Visual Stud
GitHub
Open Proje


New Project
Recent
Installed
Templates
Visual C#
Windows Universal
Windows Classic Desktop
Web
Office/SharePoint
.NET Core
.NET Standard
Android
Online
.NET Framework 4.6.1
Sort by: Name Ascending
Search Installed Templat
Class Library (Xamarin.Forms) Visual C#
Console App (.NET Core) Visual C#
Console App (.NET Framework) Visual C#
Cross Platform App (Xamarin) Visual C#
Excel 2010 VSTO Workbook Visual C#
Excel 2013 and 2016 VSTO Workbook Visual C#
Type: Visual C#
A project for creating a command-line application
Name: Sample
Location: d:\0_mydocument\documents\visual studio 2017\Projects
Solution name: Sample
Browse...
Create directory for solution
Create new Git repository
OK
Cancel


Sample - Microsoft Visual Studio (Administrator)
Quick Launch
File Edit View Project Build Incredibuild Debug Team Tools Test SQL Prompt ReSharper
R Tools Analyze Window Help
Kevin Lin
Debug
Any CPU
Start
Toolbox
Search Toolbox
General
There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.
Program.cs
Sample
Sample.Program
Main(string[] args)
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace Sample
{
class Program
{
static void Main(string[] args)
{
}
}
}
Solution Explorer
Search Solution Explorer (Ct
Solution 'Sample' (1 project)
Sample
Properties
References
App.config
Program.cs
Server Explorer
Toolbox
108 %
Properties
Solutio...
Team Ex...
Package Manager Console
Error List ...
Output
NuGet browser
Ln 1
Col 1
Ch 1
INS
Add to Source Control

==========

2. Program

```
using System;
using System.IO;
using System.Runtime.Serialization;
using OnLineGame;
namespace Sample
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("TryCatchSample1(); ================================");
            TryCatchSample1();
            Console.WriteLine("NestedTryCatch(); ================================");
            NestedTryCatch();
            Console.WriteLine("NoVpnConnectionSample(); ================================");
            NoVpnConnectionSample();
            Console.WriteLine("ExceptionHandlingAbuseSample(); ================================");
            ExceptionHandlingAbuseSample();
            Console.WriteLine("FixExceptionHandlingAbuseSample(); ================================");
            FixExceptionHandlingAbuseSample();
            Console.ReadLine();
        }
        // 1. ------------------------------------------------------------
        // Try Catch
        public static void TryCatchSample1()
        {
            StreamReader reader = null;
            StreamWriter writer = null;
            try
            {
                // Reader
                reader = new StreamReader(@"D:\0_DeleteMe\Data.txt");
                //if the fielPath doesn't exist, throw System.FileNotFoundException
                string readerInputStr = reader.ReadToEnd();
                Console.WriteLine(readerInputStr);
                // Writer.
                string filePath = @"D:\0_DeleteMe\Data2.txt";
                writer = new StreamWriter(filePath);
                writer.Write(readerInputStr);
                writer.Close();
                Console.WriteLine("StreamWriter Updated.");
                //StreamWriter will create the file if it doesn't exist.
                //Thus, it will never throw System.FileNotFoundException.
            }
            catch (FileNotFoundException ex)
            {
```

```
                //FileNotFoundException Exception Handing.
                Console.WriteLine("{0} can not be found. Please create it.\n", ex.FileName);
                Console.WriteLine("Exception Message : {0} \n", ex.Message);
                Console.WriteLine("Stack Trace : {0} \n", ex.StackTrace);
                //1.2.1.
                //System.FileNotFoundException :
                //    Occurs when read from a file that does not exist.
            }
            catch (Exception ex)
            {
                //Handles the rest of exception
                Console.WriteLine(ex.Message);
                Console.WriteLine(ex.StackTrace);
            }
            finally
            {
                //The code that always executed no matter the exception is occured or not
                //Closes the stream and releases the resources.
                //if (reader != null)
                //{
                //    reader.Close();
                //}
                reader?.Close();
            }
        }

        // 2. ------------------------------------------------------------
        // Try Catch
        public static void NestedTryCatch()
        {
            try
            {
                try
                {
                    //1.2.3.
                    //System.FormatException
                    //    Occurs when enter a Character into int type variable.
                    //1.2.4.
                    //System.OverflowException
                    //    Occurs when enter a very big number ,that an integer cannot hold, into the int
type variable.
                    //1.2.5.
                    //System.DivideByZeroException
                    //    Occurs when there is an attempt to divide an integral or Decimal value by zero.
                    Console.WriteLine("Enter Dividend");
                    int dividend = Convert.ToInt32(Console.ReadLine());
                    // E.g.1.
                    // Enter "A" will cause System.FormatException
                    // E.g.2.
                    // Enter "12345678901" will cause System.OverflowException
                    Console.WriteLine("Enter Divisor");
                    int divisor = Convert.ToInt32(Console.ReadLine());
                    // E.g.3.
                    // Enter "0" in Divisor will cause System.DivideByZeroException
                    int quotient = dividend / divisor;
```

```
                    Console.WriteLine("Dividend/Divisor = {0}", quotient);
                }
                catch (Exception ex)
                {
                    // \n is new line for Console
                    Console.WriteLine("Exception : {0} \nMessage : {1} \nStackTrace : {2} \n",
ex.GetType().Name, ex.Message, ex.StackTrace);
                    string filePath = @"D:\0_DeleteMe\Log.txt";
                    if (File.Exists(filePath))
                    {
                        StreamWriter writer = new StreamWriter(filePath);
                        // \r\n is new line for Console for file, r means return,n means new line.
                        writer.Write("Exception : {0} \r\nMessage : {1} \r\nStackTrace : {2} \r\n",
ex.GetType().Name, ex.Message, ex.StackTrace);
                        writer.Close();
                        Console.WriteLine("Error Log Updated.");
                    }
                    else
                    {
                        //Throw the FileNotFoundException to outter Try Catch clause.
                        throw new FileNotFoundException(filePath + " does not Exist, please create it. \n",
ex);

                        //1.2.1.
                        //System.FileNotFoundException :
                        //    Occurs when read from a file that does not exist.
                    }
                }
            }
            catch (Exception ex)
            {
                //outer Try Catch Clause
                //ex.Message return the current exception message from outer Try Catch Clause.
                Console.WriteLine("Current Exception Message from outer Try Catch Clause : {0} \n",
ex.Message);
                //if the writer filePath doesn't exist, here will catch System.FileNotFoundException
                //inner Try Catch Clause
                //ex.InnerException.Message return the exception message from inner Try Catch Clause.
                //if ex.InnerException is exist then print it.
                if (ex.InnerException != null)
                {
                    Console.WriteLine("Exception Message From Inner Try Catch Clause : {0} \n",
ex.InnerException.Message);
                }
                // E.g.1.
                // Enter "A" will cause System.FormatException
                // E.g.2.
                // Enter "12345678901" will cause System.OverflowException
                // E.g.3.
                // Enter "0" in Divisor will cause System.DivideByZeroException
            }
        }

        // 3. ------------------------------------------------------------
        // Custom Exceptions
        public static void NoVpnConnectionSample()
        {
            try
```

```
            {
                throw new NoVpnConnectionException("User is using vpn");
            }
            catch (NoVpnConnectionException ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
        // 4. ------------------------------------------------------------
        // Custom Exception Handling Abuse Sample.
        // Programmers should never do Exception Handling Abuse
        // which means using exception to implement programming logic.
        private static void ExceptionHandlingAbuseSample()
        {
            try
            {
                //1.2.3.
                //System.FormatException
                //    Occurs when enter a Character into int type variable.
                //1.2.4.
                //System.OverflowException
                //    Occurs when enter a very big number ,that an integer cannot hold, into the int type
variable.
                //1.2.5.
                //System.DivideByZeroException
                //    Occurs when there is an attempt to divide an integral or Decimal value by zero.
                Console.WriteLine("Enter Dividend");
                int dividend = Convert.ToInt32(Console.ReadLine());
                // E.g.1.
                // Enter "A" will cause System.FormatException
                // E.g.2.
                // Enter "12345678901" will cause System.OverflowException
                Console.WriteLine("Enter Divisor");
                int divisor = Convert.ToInt32(Console.ReadLine());
                // E.g.3.
                // Enter "0" in Divisor will cause System.DivideByZeroException
                int quotient = dividend / divisor;
                Console.WriteLine("Dividend/Divisor = {0}", quotient);
            }
            catch (FormatException)
            {
                Console.WriteLine("Please enter an integer.");
            }
            catch (OverflowException)
            {
                Console.WriteLine("The integer should be between {0} and {1}.",
                    Int32.MinValue, Int32.MaxValue);
            }
            catch (DivideByZeroException)
            {
                Console.WriteLine("Divide By Zero is not allowed.");
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
```

```
        // 5. ------------------------------------------------------------
        // Custom Exception Handling Abuse Sample.
        // Programmers should never do Exception Handling Abuse
        // which means using exception to implement programming logic.
        private static void FixExceptionHandlingAbuseSample()
        {
            try
            {
                //1.2.3.
                //System.FormatException
                //    Occurs when enter a Character into int type variable.
                //1.2.4.
                //System.OverflowException
                //    Occurs when enter a very big number ,that an integer cannot hold, into the int type
variable.
                //1.2.5.
                //System.DivideByZeroException
                //    Occurs when there is an attempt to divide an integral or Decimal value by zero.
                Console.WriteLine("Enter Dividend");
                // int dividend = Convert.ToInt32(Console.ReadLine());
                // E.g.1.
                // Enter "A" will cause System.FormatException
                // E.g.2.
                // Enter "12345678901" will cause System.OverflowException
                int dividend;
                //int.TryParse() return false if the convertion fail.
                if (int.TryParse(Console.ReadLine(), out dividend))
                {
                    Console.WriteLine("Enter Divisor");
                    //int divisor = Convert.ToInt32(Console.ReadLine());
                    // E.g.3.
                    // Enter "0" in Divisor will cause System.DivideByZeroException
                    int divisor;
                    if (int.TryParse(Console.ReadLine(), out divisor))
                    {
                        if (divisor != 0)
                        {
                            int quotient = dividend / divisor;
                            Console.WriteLine("Dividend/Divisor = {0}", quotient);
                        }
                        else
                        {
                            Console.WriteLine("Divide by zero is not allowed.");
                        }
                    }
                    else
                    {
                        Console.WriteLine("Valid integer should be between {0} and {1}.",
                        Int32.MinValue, Int32.MaxValue);
                    }
                }
                else
                {
                    Console.WriteLine("Valid integer should be between {0} and {1}.",
                        Int32.MinValue, Int32.MaxValue);
```

```
                }
            }
            catch (Exception ex)
            {
                // \n is new line for Console
                Console.WriteLine("Exception : {0} \nMessage : {1} \nStackTrace : {2} \n", ex.GetType().Name,
ex.Message, ex.StackTrace);
                // Writer.
                StreamWriter writer = new StreamWriter(@"D:\0_DeleteMe\Log.txt");
                // \r\n is new line for Console for file, r means return,n means new line.
                writer.Write("Exception : {0} \r\nMessage : {1} \r\nStackTrace : {2} \r\n",
ex.GetType().Name, ex.Message, ex.StackTrace);
                writer.Close();
                Console.WriteLine("Error Log Updated.");
                //StreamWriter will create the file if it doesn't exist.
                //Thus, it will never throw System.FileNotFoundException.
            }
        }
    }
}



// 3. ------------------------------------------------------------
// Custom Exceptions
//Sometimes there is no existing system exception can fit your situation.
//In this case, you might have to create your own Customize Exceptions.
namespace OnLineGame
{
    //The Customize Exceptions must extend the System.Exception class.
    //In addition the suffix should be "Exception" to fulfill the naming convention.
    [Serializable]
    public class NoVpnConnectionException : Exception
    {
        // The Customize Exceptions should always provide a public constructor
        //which take a string message parameter,
        //and then pass it to base class constructor.
        public NoVpnConnectionException(string message)
            : base(message)
        {
        }
        //The Customize Exceptions should always provide the capability to handle inner exception.
        public NoVpnConnectionException(string message, Exception innerException)
            : base(message, innerException)
        {
        }
        //The Customize Exceptions should always support Serializable, thus,
        //it can work across application domains.
        public NoVpnConnectionException(SerializationInfo info, StreamingContext context)
            : base(info, context)
        {
        }
    }
}



/*
1.
System.Exception
```

```
------------------------------------------------
1.1.
An exception is a Sub Class of System.Exception Class which
represents errors during a program is running.
1.1.1.
System.Exception.Message is description of exception.
1.1.2.
System.Exception.StackTrace is stack information of the exception.
------------------------------------------------
1.2.
System.Exception SubClass
1.2.1.
System.FileNotFoundException :
    Occurs when read from a file that does not exist.
1.2.2.
System.SqlException :
    Occurs when read from a database table that does not exist
1.2.3.
System.FormatException
    Occurs when enter a Character into int type variable.
1.2.4.
System.OverflowException
    Occurs when enter a very big number ,that an integer cannot hold, into the int type variable.
1.2.5.
System.DivideByZeroException
    Occurs when there is an attempt to divide an integral or Decimal value by zero.
------------------------------------------------
1.3.
Syntax:
//try
//{
//    //The code that can possibly cause an exception
//}
//catch (FileNotFoundException ex)
//{
//    //FileNotFoundException Exception Handing.
//    Console.WriteLine("{0} can not be found. ", ex.FileName);
//}
//catch (Exception ex)
//{
//    //Handles the rest of exception
//    Console.WriteLine(ex.Message);
//    Console.WriteLine(ex.StackTrace);
//}
//finally
//{
//    //The code that always executed no matter the exception is occured or not
//}
------------------------------------------------------------------------------------------------------
2.
Customize Exceptions
------------------------------------------------
2.0.
Sometimes there is no existing system exception can fit your situation.
In this case, you might have to create your own Customize Exceptions.
E.g.
This online game does not allow user to use VPN
------------------------------------------------
2.1.
// public class NoVpnConnectionException : Exception
The Customize Exceptions must extend the System.Exception class.
In addition the suffix should be "Exception" to fulfill the naming convention.
------------------------------------------------
2.2.
// public NoVpnConnectionException(string message) : base(message)
The Customize Exceptions should always provide a public constructor which take a string message
parameter,
```

```
and then pass it to base class constructor.
-------------------------------------------------
2.3.
// public NoVpnConnectionException(string message, Exception innerException) : base(message,
innerException)
The Customize Exceptions should always provide the capability to handle inner exception.
-------------------------------------------------
2.4.
//[Serializable]
//public class NoVpnConnectionException : Exception
...
//public NoVpnConnectionException(SerializationInfo info, StreamingContext context): base(info, context)
The Customize Exceptions should always support Serializable,
thus, it can work across application domains.
-----------------------------------------------------------------------------------------------------
3.
Programmers should never do Exception Handling Abuse
which means using exception to implement programming logic.
*/
```