Exception Handling

Errors :- Which interrupts the compilation of the application or gives us illogical results.

Types of Errors

Compilation Error :- syntax error / rectify it at design time

Logical Error :- Will not give expected output / we test our logic before putting it in development.

Logical checks o/p

>=35 and <60 Second class

>=60 and <75 First class

>75 Distinction

Else Fail

65

75 fail

Runtime Error :- Something Unexpected occurring at run time. Are called as Exceptions. They abnormally or abruptly terminates our application.

How to resolve this, by Implementing Exception Handling technique ie using try{} catch() block.

Guidelines

1. Every try block must contain atleast one catch block immediately after it.
2. catch block takes an object of the exception class as a parameter
3. finally block :- It is optional. This block when defined will execute regardless of an exception.

Predefined Exception class are present in the library

Like IndexOutOfBoundsException

DivideByZeroException…

Exception is the root class :- Logic for abnormal termination, it contains read-only property (Message) to display an error message which is declared as virtual ie the child class can override this property.

Derived exceptions are :-

ApplicationException 🡪 Non-fatal Errors

For example :- as per application requirement, let say we don’t want divisor to be odd number , these are not caused by CLR, developer will perform the actions. (Application manually needs to throw the exception – so as to catch it(resolve it)

SystemException (Fatal Errors) These type of actions should never be performed, system will not allow to perform 🡪 DividebyZero (it is automatically thrown by System) 🡪 we only need to catch it.

CLR takes responsibility of exceptions.

IndexOutOfBoundsException

FormatException

ArithmeticException

DivideByZeroException

OverFlowException

class Program

{

static void Main(string[] args)

{

DirectoryInfo dinfo;

try

{

dinfo = new DirectoryInfo(@"F:\PhilipTraining\c#demos");

// get all files in the directory , name and their size

FileInfo[] filesindir = dinfo.GetFiles();

foreach (FileInfo file in filesindir)

{

Console.Write("File Name :{0} \t Size : {1} \t Created on : {2} \n", file.Name, file.Length, file.CreationTime);

}

}

catch (DirectoryNotFoundException ex)

{

Console.WriteLine("Directory not found");

}

finally

{

Console.WriteLine("Happy Learning !! Welcome again");

}

}

}

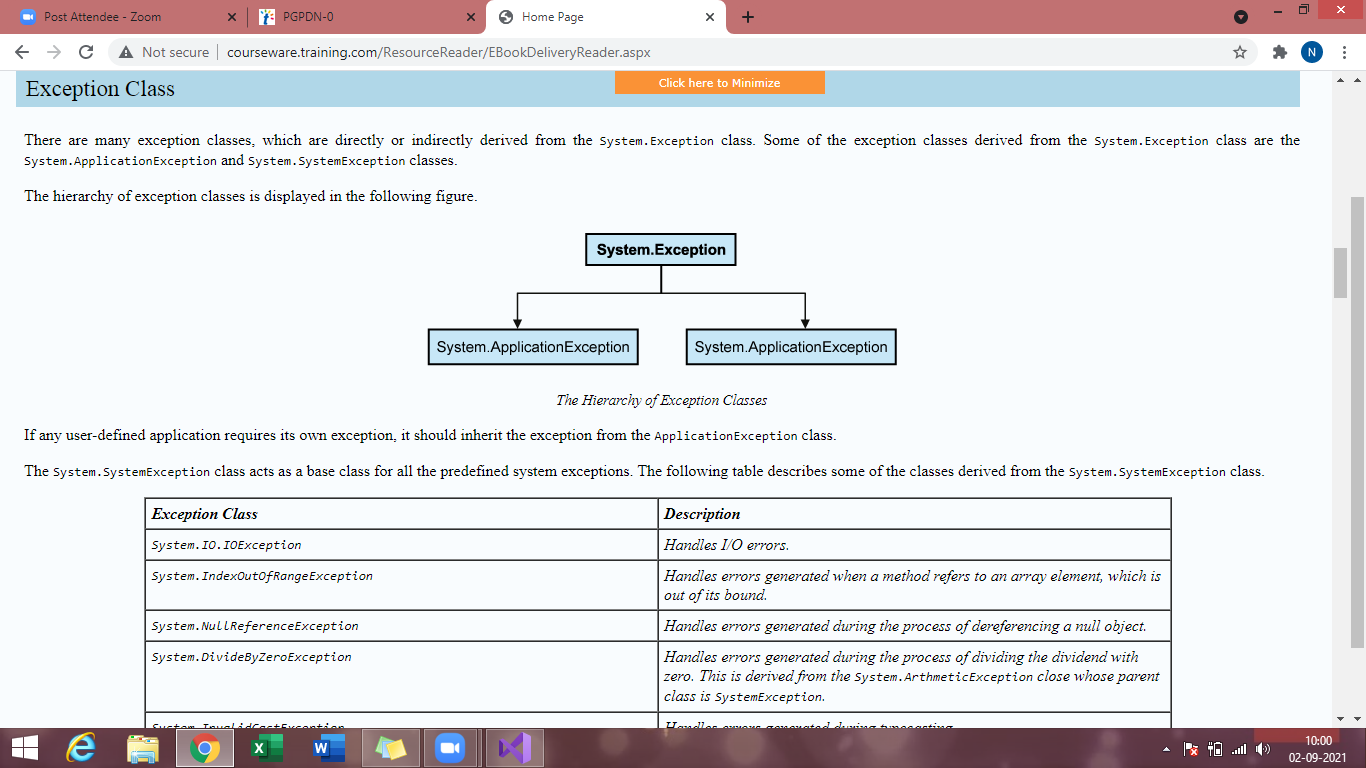
}

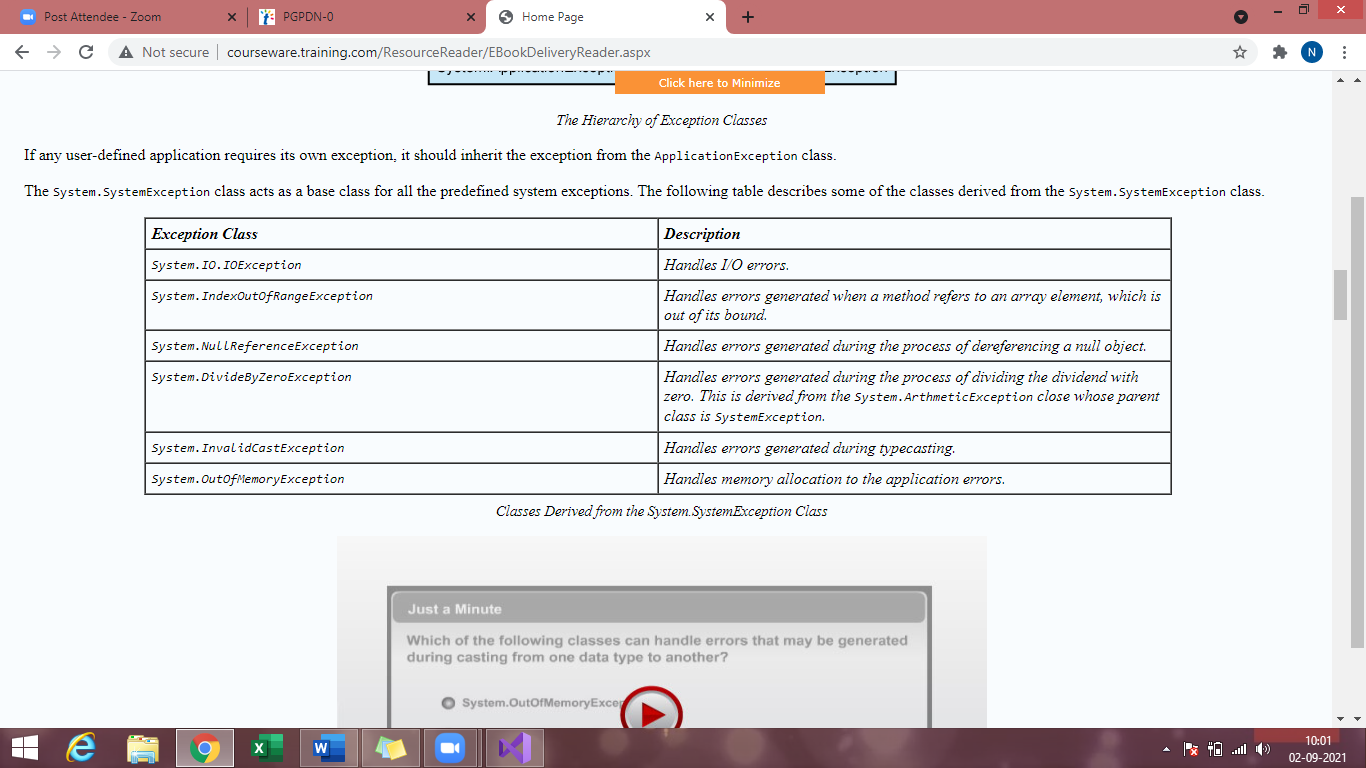
finally{

if (resource!=null){

//close that resource

}





Problem Statement :-

Calculating the sum of values in the integer array. We have to handle the exception while working with the arrays.

public void CalculateSum()

{

int i;

int sum = 0;

int[] number = new int[5] { 1, 2, 3, 4, 5 };

try

{

for (i = 0; i < 5; i++)

{

sum += number[i];

}

}

catch (IndexOutOfRangeException ex)

{

Console.WriteLine("Please check the iterations of the list");

}

finally

{

Console.WriteLine("Total : {0}", sum);

}

}

}

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Handling System Exception ");

ArrayOutOfIndex obj = new ArrayOutOfIndex();

obj.CalculateSum();

}

}

}

public void Divide()

{

int x, y, z=0;

Console.WriteLine("Enter two numbers");

try

{

x = Convert.ToInt32(Console.ReadLine());

y = Convert.ToInt32(Console.ReadLine());

z = x / y;

}

catch(DivideByZeroException dze)

{

Console.WriteLine("Cannot divide by zero");

}

catch(FormatException fe)

{

Console.WriteLine("please check the values and try again");

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

finally

{

Console.WriteLine(z);

}

Exception classes :- derived from base class Exception.

SystemException

ApplicationException :- derived from the Exception class

Userdefined Exceptions

All UD exceptions are derived from ApplicationException class.

class MyException : ApplicationException{

public MyException(string message) : base(message)

}

1. We need to throw the custome exception out of a try block and catch it in a catch block.

class Calculate

{

int sum = 0;

int count = 0;

internal float average;

public void FindAverage(params int[] values)

{

count = values.Length;

if (count == 0)

{

throw (new CountIsZeroException("Count is zero for finding average, Try again!!!"));

}

else

{

foreach (int x in values)

{

sum += x;

}

average = sum / count;

}

}

}

public class CountIsZeroException : Exception

{

public CountIsZeroException(){}

public CountIsZeroException(string message) : base(message){}

public CountIsZeroException(string message, Exception ex) : base(message,ex){}

}

public class CountIz… : ApplicationException{

public CountIsZeroException(string message) : base(message){}

}

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Handling User Defined Exception ");

Calculate calc = new Calculate();

try

{

calc.FindAverage();

}

catch(CountIsZeroException ex)

{

Console.WriteLine(ex.Message);

}

finally

{

Console.WriteLine("The average is {0} ", calc.average);

}

}

}

}

You can throw the object of the exception by using throw statement.

try catch throw finally

Exception 🡪 base

Predefined exception

DivideBy…

IndexOut…

File… direc.. Out of Io