Day 12 Assignment

By

VARUN SAI KUMAR CHEGONI

NB Healthcare and Technology

Date: 08 Feb 2022

**Topics**

**C# Exception Handling**

**Compilation Error & Runtime Error**

**Content**

|  |  |  |
| --- | --- | --- |
| S.No | Content | Page No. |
| 1 | What is Exception Handling and why we need exception handling? |  |
| 2 | Write a simple division program and handle three exceptions discussed in the class., also add super exception at the last. |  |
| 3 | Research and write at least 6 exceptions that occur in C# with sample code. |  |
| 4 | What is the use of "finally" block illustrate with an example? |  |
| 5 | Write the 5 points I explained about exception handling. |  |
| 6 | What is compilation and Runtime error Write at least 3 differences between them |  |
| 7 | Write any 6 compilation errors with small code snippet. Add compilation error screen shots. |  |
| 8 | Write any 6 runtime errors with small code snippets and add run time error screen shots. |  |

|  |
| --- |
| 1. What is Exception Handling and why we need exception handling? |
| Answer: |
| * Exception Handling is done to ensure that our application will not crash. * Exception Handling will not display any technical details and to make sure we handle errors gracefully and display friendly message. |

|  |
| --- |
| 2. Write a simple division program and handle three exceptions discussed in the class., also add super exception at the last. |
| Code : |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace DivisionExceptionHandling  {  internal class Program  {  static void Main(string[] args)  {  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Author : Varun Sai Kumar Chegoni.  \* Purpose : simple division program and handle three exceptions discussed in the class., also add super exception at the last.  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  try  {  int a, b, ans; // Variable Declaration  Console.WriteLine("Enter First Number: ");  a=Convert.ToInt32(Console.ReadLine()); // Read Data  Console.WriteLine("Enter First Number: ");  b=Convert.ToInt32(Console.ReadLine()); // Read Data  ans = a/b; // Logic  Console.WriteLine("Answer = "+ans); // Output  Console.ReadLine();  }  catch (OverflowException)  {  Console.WriteLine("Please Enter Number within 0 to 999999999"); // Exception Output  Console.ReadLine();  }  catch (FormatException)  {  Console.WriteLine("Please Enter Numbers only"); // Exception Output  Console.ReadLine();  }  catch (DivideByZeroException)  {  Console.WriteLine("Number Cannot be divided by Zero"); // Exception Output  Console.ReadLine();  }  catch (Exception)  {  Console.WriteLine("Some Error Occured. Please Contact Us"); // Super Exception Output  Console.ReadLine();  }  }  }  } |
| Output : |
|  |

|  |
| --- |
| 3. Research and write at least 6 exceptions that occur in C# with sample code. |
| Answer : |
| **1. System.OutOfMemoryException:**  Reason: The error that are generated due to insufficient free memory is handled by this exception.  Example code: using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Check  {  internal class Program  {  static void Main(string[] args)  {  // a string variable is created and tried to store 2.1 billion characters and this causes an out of memory exception  string val = new string('r', int.MaxValue);  }  }  }    **2. System.NullReferenceException:**  Reason: The errors that are generated from referencing a null object is handled by this exception.  Example Code:  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Check  {  internal class Program  {  static void Main(string[] args)  {  //a string variable is defined, and it is referencing to null  string value = null;  //the length of the value referencing to null is checked if it is equal to zero causing an exception  if (value.Length == 0)  {  Console.WriteLine(value);  }  }  }  }    **3. System.InvalidCastException:**  Reason: The errors that are generated during typecasting is handled by this exception.  Example Code:  using System;  using System.IO;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Check  {  internal class Program  {  static void Main(string[] args)  {  // an instance of the string builder class is created which is then assigned to a new object through implicit casting and then casting is tried explicitly to convert the instance of stringbuilder class to streamreader class  StringBuilder ref1 = new StringBuilder();  object ref2 = ref1;  StreamReader ref3 = (StreamReader)ref2;  }  }  }    **4. System.ArrayTypeMismatchException:**  Reason: The errors that are generated when there is a mismatch of type with the array type is handled by this exception.  Example Code:  using System;  using System.IO;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Check  {  internal class Program  {  static void Main(string[] args)  {  // a string is defined and assigned the values which is then assigned to object class array and then an integer is tried to put in the same array  string[] arr1 = { "Welcome", "to", "CSharp" };  object[] arr2 = arr1;  arr2[0] = 8;  }  }  }    **5. System.StackOverflowException:**  Reason: The errors that are generated from stack overflowing is handled by this exception.  Example Code:  using System;  using System.IO;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Check  {  internal class Program  {  // a method called recurse is defined which takes a value as parameter and increases its value by one  static void Recurse(int val)  {  // since we have written a recursive loop and 0 is passed as a parameter, it ends in an infinite loop causing exception  Console.WriteLine(val);  Recurse(++val);  }  //main method is called  public static void Main()  {  //The recurse method is called to start the infinite recursion  Recurse(0);  }  }  } |

|  |
| --- |
| 4. What is the use of "finally" block illustrate with an example? |
| Answer : |
| Finally, is used to execute code irrespective of exception or results  Example Code:  using System;  using System.IO;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Check  {  internal class Program  {  public static void Main()  {  try  {  int[] numbers = { 1, 2, 3 };  Console.WriteLine(numbers[10]);  }  catch(Exception e)  {  Console.WriteLine("Error Occured");  }  finally  {  Console.WriteLine("The operation is done.");  Console.ReadLine();  }  }  }  } |
| Output : |
|  |

|  |
| --- |
| 5. Write the 5 points I explained about exception handling. |
| Answer: |
| * Exception Handling is done to handle exception gracefully without displaying any errors to the end-user. * A single try block can have multiple catch blocks. * Super Exception should be at the last to prevent the compilation error. * Statement written in Finally block will be executed all the time. * General Syntax for writing exception is  try catch  finally |
| 6. What is compilation and Runtime error Write at least 3 differences between them |
| Answer: |
| Compilation Error: Compilation or compile time errors occur due to typing mistakes in the code.  Runtime Error: Runtime errors are the errors that is occurred when the program is in running state.   |  |  | | --- | --- | | Compilation Error | Runtime error | | 1. Time period for translation of a source code to intermediate code is compile time. | 1. Time period between start and end of running intermediate code at runtime environment. | | 1. This is to check syntax and semantics of the code. | 1. This is to run the code. | | 1. Error gets detected by compiler without execution of the program. | 1. Only can detect after execution of the program. | | 1. Fixing an error at this stage is possible. | 1. Fixing an erroe requires going back to code. | |