|  |
| --- |
| Feb 8th Morning Assignment  By Surya Teja Chandolu |

|  |
| --- |
| 1. What is Exception Handling and why we need exception handling. |
| Exception Handling:  An exception is an occurrence that occurs during the execution of a programme that the programme code does not expect. The measures to be taken in the event of an exception are unknown to the programme.  As part of C#, you can use the try, catch, and finally keywords to try actions that may not succeed, to deal with failures when you determine it is necessary to do. |

|  |
| --- |
| 1. Write a simple division program and handle three exceptions discussed in the class., also add super exception at the last. |
| Code: |
| using System;  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Author: Surya Teja  \* Purpose: Divison of two numbers using try, catch and finally block  \* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  namespace Division  {  internal class Program  {  static void Main(string[] args)  {  int firstNumber, secondNumber, div;  try  {  Console.Write("Enter First Number:");  firstNumber = Convert.ToInt32(Console.ReadLine());  Console.Write("Enter Second Number:");  secondNumber = Convert.ToInt32(Console.ReadLine());  div = firstNumber / secondNumber;  Console.WriteLine($"Divison is :{div}");  }  catch (OverflowException)  {  Console.WriteLine($"Enter number range from {int.MinValue} to {int.MaxValue}");  }  catch (DivideByZeroException)  {  Console.WriteLine("Enter second number as other than zero");  }  catch (FormatException)  {  Console.WriteLine("Enter only numbers");  }  catch (Exception)  {  Console.WriteLine("Please contact Office");  }  finally  {  Console.ReadLine();  }  }  }  } |
| Output: |
|  |

|  |
| --- |
| 1. Research and write atleast six exceptions that occur in C# with sample code. |
| |  | | --- | | ArrayTypeMismatchException | | Code: | | using System;  namespace ConsoleApp1  {  internal class Program  {  static void Main(string[] args)  {  string[] name = { "Surya" };  object[] names = name;  names[0] = 3;  Console.ReadLine();  }  }  } | | Exception: | |  | |
| |  | | --- | | NullReferenceException | | Code: | | using System;  namespace ConsoleApp1  {  internal class Program  {  static void Main(string[] args)  {  string name = null;  Console.WriteLine(name);  if(name.Length == 0)  Console.WriteLine(name);  Console.ReadLine();  }  }  } | | Exception: | |  | |
| |  | | --- | | OutOfMemoryException | | Code: | | using System;  namespace ConsoleApp1  {  internal class Program  {  static void Main(string[] args)  {  string name = new string('S', int.MaxValue);  Console.ReadLine();  }  }  } | | Exception: | |  | |
| |  | | --- | | InvalidOperationException | | Code: | | using System;  using System.Collections.Generic;  using System.Linq;  namespace ConsoleApp1  {  internal class Program  {  static void Main(string[] args)  {  var num = new List<int> { 1, 5, 10 };  var gt = num.Where(x => x > 10).First();  Console.ReadLine();  }  }  } | | Exception | |  | |
| |  | | --- | | FileNotFoundException | | Code: | | using System;  using System.IO;  namespace ConsoleApp1  {  internal class Program  {  static void Main(string[] args)  {  StreamReader sr = new StreamReader(@"c:\Temp\sample.txt");  Console.WriteLine(sr.ReadToEnd());  Console.ReadLine();  }  }  } | | Exception: | |  | |
| |  | | --- | | IndexOutOFRangeException | | Code: | | using System;  namespace ConsoleApp1  {  internal class Program  {  static void Main(string[] args)  {  int [] data = new int[3];  data[0] = 10;  data[1] = 20;  data[2] = 30;  data[3] = 40;  data[4] = 50;  foreach (int i in data)  Console.WriteLine($"{i} ");  Console.ReadLine();  }  }  } | | Exception: | |  | |

|  |
| --- |
| 1. What is the use of "finally" block illustrate with an example. |
| Finally Block: A finally block contains all statements that must be executed regardless of whether or not an exception arises. Regardless of whether an exception happens in the try block or not, the statements in this block. |
| Example: |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Author: Surya Teja  \* Purpose: Divison of two numbers using try, catch and finally block  \* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  namespace Division  {  internal class Program  {  static void Main(string[] args)  {  int firstNumber, secondNumber, div;  try  {  Console.Write("Enter First Number:");  firstNumber = Convert.ToInt32(Console.ReadLine());  Console.Write("Enter Second Number:");  secondNumber = Convert.ToInt32(Console.ReadLine());  div = firstNumber / secondNumber;  Console.WriteLine($"Divison is :{div}");  }  catch (OverflowException)  {  Console.WriteLine($"Enter number range from {int.MinValue} to {int.MaxValue}");  }  catch (DivideByZeroException)  {  Console.WriteLine("Enter second number as other than zero");  }  catch (FormatException)  {  Console.WriteLine("Enter only numbers");  }  catch (Exception)  {  Console.WriteLine("Please contact Office");  }  finally  {  Console.WriteLine("\n\n\n\n\nDesigned by Surya");  Console.ReadLine();  }  }  }  } |
| Output: |
|  |

|  |
| --- |
| 1. Write the 5 points I explained about exception handling. |
| * Exception handling is done to handle errors gracefully and not to crash. * In Exception handling one try block can have multiple catch block. * We must initialize Exception only at last. * Finally block will execute all statements whether or not an exception arises. * Try 🡪 Catch 🡪 Finally. |

|  |
| --- |
| 1. What is compilation and Runtime error Write atleast 3 differences between them |
| |  |  | | --- | --- | | Compile Time Error | Run Time Error | | Errors that occur when you break the rules of writing syntax. | Errors occur while executing the program. | | Errors are detected by compiler which are easy to fix. | Errors are hard to find and fix the issue. | | It prevent running code with errors. | It run the code and display the wrong output. | | EX: missing {}, ;, “”, etc,. | EX: dividing by 0. | |

|  |
| --- |
| 1. Write any 6 compilation errors with small code snippet. Add compilation error screen shots |
| Input num is not initlize. |
| Class name is missing |
| In static keyword S should be in lower case. |
| System class is not added. |
| Cannot implement string to int. |
| Statement must end with ; |

|  |
| --- |
| 1. Write any 6 runtime errors with small code snippets and add run time error screen shots. |
| Number cannot divide by zero |
| Name cannot convert into integer. |
| Integer number is long |
| Array index is out of range |
| Length of Null value cannot be zero |
| Array type mismatch |