|  |
| --- |
| Day-12 Morning Assignment  By  U.Joshna  [8-2-2022] |

|  |
| --- |
| 1.what is Exception handling and why we need exception handling?  Exception Handling in C# is **a process to handle runtime errors**. We perform exception handling so that normal flow of the application can be maintained even after runtime errors. In C#, exception is an event or object which is thrown at runtime. All exceptions the derived from System. |

|  |
| --- |
| 2.write a simple division program  And handle three exceptions discussed in the class, also add super  Super exception at the last? |
| Code: |
|  |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Day\_12\_project\_2  {    internal class Program  {    static void Main(string[] args)  {      try  {  int a, b, c;  Console.Write("Enter the Dividend Value : ");  a = Convert.ToInt32(Console.ReadLine());  Console.Write("Enter the Divisor Value to Divide {a} : ");  b = Convert.ToInt32(Console.ReadLine());    c = a / b;  Console.WriteLine("The Division of {a} / {b} is : {c}");  Console.ReadLine();  }    catch (OverflowException)  {  Console.WriteLine(" Please Do Enter the Numbers Only in the Range of 0 to 50000");    }    catch (DivideByZeroException)  {  Console.WriteLine(" Please Do Provide divisor Value, a Non-Zero Number to Do Perfect Division.");    }    catch (FormatException)  {  Console.WriteLine(" Please, Do Enter only Integers. Strings / Special Characters are not Allowed to do Division as per Mathematics Standard.");    }    catch (Exception)  {  Console.WriteLine(" Some Error Has Occured, Please Contact the Admin");  }  finally  {  Console.WriteLine(" Designed & Developed By Joshna");  Console.ReadLine();  }  }  }  } |
| Output: |
|  |
|  |
|  |

|  |
| --- |
| 3.Research and write atleast 6 exceptions that occur in C# with sample code.  1.StackOverflowException |
| Code: |
| using System;  namespace Day\_12\_Project\_3  {  internal class Program  {  static void Recurse(int val)  {  Console.WriteLine(val);  Recurse(++val);  }    static void Main(string[] args)  {  Recurse(1);    }  }  } |
| Output: |
|  |
| 2.IOException: |
| Code: |
| int a = 5;  int b = 10;  int c = a / b;  Console.WriteLine(c);   |  | | --- | | Output: | |  | | 3.ArrayTypeMismatchException: | | Code: | | string[] arr1 = { "Hello World" };  object[] arr2 = arr1;  arr2[0] = 5; | | Output: | |  | |  |      |  | | --- | | 4.OutOfMemoryException: | | Code: | | string val = new string('A', int.MaxValue); | | Output: | | 5.IndexOutOfRange: | | Code: | | int[] arr = new int[10];  arr[0] = 10;  arr[10] = 20;  arr[20] = 30; | | Output: | |
| 6.DivideByZero: |
| Code: |
| int a = 5;  int b = 0;  int c = a / b;  Console.WriteLine(c); |
| Output: |
|  |

|  |
| --- |
| 4.what is the use of “Finally” block illustrate with an example?  .By using a finally block, you can run code even if an exception occurs in the try block and you can clean up any resources that are allocated in a try block. Typically, the statements of a finally block run when control leaves a try statement. |
| Using Finally Block Code: |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Day\_12\_project\_4  {  internal class Program  {  static void Main(string[] args)  {  int a, b, c, d, e, f;  Console.WriteLine("Enter Arithmatic Operations Calculations");    Console.Write("Enter Any number for Value-1 : ");  a = Convert.ToInt32(Console.ReadLine());  Console.Write("Enter Any number for Value-2 : ");  b = Convert.ToInt32(Console.ReadLine());  try  {  d = a / b;  Console.WriteLine($"The Division of {a} and {b} is : {d}");  }  catch (DivideByZeroException)  {  Console.WriteLine($"The Value-2 Can't be Zero to Do Division operation");    }  finally  {  c = a + b;  Console.WriteLine($"The Addition of {a} and {b} is : {c}");  e = a - b;  Console.WriteLine($"The Subtraction of {a} and {b} is : {e}");  f = a \* b;  Console.WriteLine($"The Multiplication of {a} and {b} is : {f}");  Console.ReadLine();  }  }  }  } |
| Output: |
|  |
|  |
|  |
|  |

|  |
| --- |
| 5. What are the points about Exception Handling discussed in the class? |
| .Exception Handling is done to Handle errors gracefully  .Single try block can have multiple catch blocks  . General Exception block should be implemented at the end of all exceptional catch blocks  .Statements written in Finally Block are Executed irrespective of Exceptions  .General Syntax for Exception Handling is:  try  {  // Block of Code To Be Executed  }  catch (Exception)  {  // Exception Message, for displaying purpose.  }  finally  {  // These Block of statements will be executed, irrespective of Exceptions Occure.  } |
|  |

|  |  |
| --- | --- |
| 6. What is compilation and Runtime error  Write atleast 3 differences between them | |
| Compile Time Errors | Run Time Errors |
| 1.Compilers can easily detect compile-time errors during the development of code.  2.A compile-time error generally refers to the errors that correspond to the semantics or syntax.  3.We can easily fix a compile-time error during the development of code. | 1.A compiler cannot easily detect a runtime error. Thus, we need to identify it during the execution of code.  2.A runtime error refers to the error that we encounter during the code execution during runtime.  3.A compiler cannot identify a runtime error. But we can fix it after the execution of code and identification of the code in prior. |

|  |
| --- |
| 7. Write any 6 compilation errors with small code snippet.  Add compilation error screen shots. |
| 1.Not having proper knowledge of type casting. |
| Code: |
|  |
| 2. Not using the proper naming conventions |
| Code: |
|  |
| |  | | --- | | 3. Not importing Proper namespace. | | Code: | | 4. Missing Semicolon | | Code: | | 5. Not using proper Methods & properties | | Code: | |  | |  | |

|  |
| --- |
| 8. Write any 6 Runtime errors with sample code snippet & Runtime Errors Screenshots |
| 1. Divide by zero Runtime Error |
| Code: |
|  |
| 2. . Out of Memory Runtime error |
| Code: |
|  |
| 3. Null Reference Runtime Error |
| Code: |
|  |
| 4. Invalid Cast Runtime Error. |
| Code: |
|  |
|  |
| 5. Array Type Mismatch Runtime Error |
| Code: |
|  |
|  |
| 6. . Stack over flow Runtime Error |
| Code: |
| using System;  public class check  {  static void Recurse(int val)  {  Console.WriteLine(val);  Recurse(val);  }  public static void Main()  {  Recurse(0);  }  } |
| Output: |
|  |
|  |