DAY 12 ASSIGNMENT PRESENTED

BY

POTUKANUMA JEEVITHA

08-02-2022

|  |
| --- |
| **1.WHAT IS EXCEPTION HANDLING AND WHY WE NEED EXCEPTION HANDLING.** |
| **ANS:-** |
| Exception handling is done to ensure that our application will not crash or will not display and technical details to make sure we handle errors gracefully and display friendly messages.  **Need:**  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 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 EXCEPTION AT THE LAST.** |
| **CODE:-** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace simple\_division\_program\_using\_4\_exception  {  internal class Program  {  static void Main(string[] args)  {  try  {  int a, b, c;  Console.WriteLine("Enter first number:");  a = Convert.ToInt32(Console.ReadLine());  Console.WriteLine("Enter second number:");  b = Convert.ToInt32(Console.ReadLine());  c = a / b;  Console.WriteLine("answer = {0}", c);  Console.ReadLine();  }  catch (OverflowException Ex)  {  Console.WriteLine("only numbers between 0 and 100000 are allowed");  Console.ReadLine();  }  catch(DivideByZeroException Ex)  {  Console.WriteLine("cannot divide with zero");  Console.ReadLine();  }  catch(FormatException Ex)  {  Console.WriteLine("only numbers are allowed. please double check");  Console.ReadLine();  }  catch(Exception )  {  Console.WriteLine("some error occured. please contact : admin@mycompany.com");  Console.ReadLine();  }  }  }  } |
| **OUTPUT:-** |
|  |

|  |
| --- |
| **3.RESEARCH AND WRITE ATLEAST 6 EXCEPTIONS THAT OCCUR IN C# WITH SAMPLE CODE.** |
| **Ans:-** |
| **Indexoutofrange exception;**  **Code:**  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace indexoutofrange\_exception  {  internal class Program  {  static void Main(string[] args)  {  int[] ar = { 7, 8, 14, 11, 30 };  for (int i=0; i<=ar.Length; i++)  Console.WriteLine(ar[i]);  }  }  } |
| **OUTOFMEMORY EXCEPTION:**  **CODE:**  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace outof\_memory\_exception  {  internal class Program  {  static void Main(string[] args)  {  string val = new string ('r', int.MaxValue);  }  }  } |
| **NULLREFERENCE EXCEPTION:**  **CODE:**  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace nullreference\_exception  {  internal class Program  {  static void Main(string[] args)  {  int[] values = null;  for (int ctr =0; ctr <=9; ctr++)  values[ctr] = ctr \* 2;  foreach (var value in values)  Console.WriteLine(value);  Console.ReadLine();  }  }  } |
| **ARRAYTYPEMISMATCH EXCEPTION:**  **CODE:**  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace arraytypemismatch\_exception  {  internal class Program  {  static void Main(string[] args)  {  string[] array1 = { "rose", "lotus", "apple" };  object[] array2 = array1;  array2[0] = 5;  Console.ReadLine();  }  }  } |
| **StackOverflow Exception:-**  **Code:**  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace stackoverflow\_exception  {  internal class Program  {  static void Recurse (int val)  {  Console.WriteLine (val);  Recurse(++val);  }  static void Main(string[] args)  {  Recurse(1);  Console.ReadLine();  }  }  } |

|  |
| --- |
| **4.WHAT IS THE USE OF “FINALLY” BLOCK ILLUSTRATE WITH AN EXAMPLE.** |
| **USE:-**   * used to define finally block, this block holds the default code. * You can clean up any resources that are allocated in a try block, and you can run code even if an exception occurs in the try block. Typically, the statements of a finally block run when control leaves a try statement. |
| **CODE:-**  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace division\_using\_finally\_block  {  internal class Program  {  static void Main(string[] args)  {  try  {  int a, b, c;  Console.WriteLine("Enter first number:");  a = Convert.ToInt32(Console.ReadLine());  Console.WriteLine("Enter second number:");  b = Convert.ToInt32(Console.ReadLine());  c = a / b;  Console.WriteLine("answer = {0}", c);  Console.ReadLine();  }  catch (OverflowException Ex)  {  Console.WriteLine("only numbers between 0 and 100000 are allowed");    }  catch (DivideByZeroException Ex)  {  Console.WriteLine("cannot divide with zero");    }  finally  {  Console.WriteLine("\n\n\n\n\n\n\n\n\n\n Designed by Jeevitha");  Console.ReadLine();  }  }  }  } |
| **Output:-** |

|  |
| --- |
| **5. WRITE THE 5 POINTS I EXPLAINED ABOUT EXCEPTION HANDLING.** |
| **ANS:-** |
| * Exception handling is done to ensure that our application will not crash or will not display and technical details to make sure we handle errors gracefully and display friendly message. * A single try block can have multiple catch blocks. * Always have a general exception at the last. * Statements return inside finally block will be executed irrespective of exception occurs or not. * The syntax for exception handling:   try  {  //put the code here that may raise exceptions  }  catch  {  // handle exception here  }  finally  {  //final cleanup code  } |

|  |  |
| --- | --- |
| **6. DIFFERENCE BETWEEN THE COMPILATION ERROR AND RUN TIME ERROR.** | |
| **ANS:-** | |
| **COMPILATION ERROR** | **RUN TIME ERROR** |
| * A Compiler error happens when you try to compile the code. If you are unable to compile your code, that is a compiler error. | * A runtime error happens during the running of the program . if u compile and run your code, but then its fails during execution, that is runtime. |
| * Errors get detected by compiler without execution of the program. | * Only can detect after execution of the program. |
| * Fixing an error at the stage is possible. | * Fixing an error requires going back to code. |

|  |
| --- |
| **7. ANY 6 COMPILATION ERRORS WITH SMALL CODES SNIPPETS AND OUTPUT.** |
| **ANS:-** |
| SEMICOLON MISSING |
| USING LOWER CASE IN “WRITELINE” |
| NAMESPACE MISSING ERROR |
| VALUE MISSING ERROR |
| MISSING PARANTHESIS |

|  |
| --- |
| **8. ANY 6 RUN TIME ERRORS WITH SMALL CODES SNIPPETS AND OUTPUUT.** |
| **ANS:-** |
| **Indexoutofrange exception;**  **Code:**  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace indexoutofrange\_exception  {  internal class Program  {  static void Main(string[] args)  {  int[] ar = { 7, 8, 14, 11, 30 };  for (int i=0; i<=ar.Length; i++)  Console.WriteLine(ar[i]);  }  }  }  **Output:**    **OUTOFMEMORY EXCEPTION:**  **CODE:**  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace outof\_memory\_exception  {  internal class Program  {  static void Main(string[] args)  {  string val = new string ('r', int.MaxValue);  }  }  }  **OUTPUT:**    **NULLREFERENCE EXCEPTION:**  **CODE:**  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace nullreference\_exception  {  internal class Program  {  static void Main(string[] args)  {  int[] values = null;  for (int ctr =0; ctr <=9; ctr++)  values[ctr] = ctr \* 2;  foreach (var value in values)  Console.WriteLine(value);  Console.ReadLine();  }  }  }  **OUTPUT:**    **ARRAYTYPEMISMATCH EXCEPTION:**  **CODE:**  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace arraytypemismatch\_exception  {  internal class Program  {  static void Main(string[] args)  {  string[] array1 = { "rose", "lotus", "apple" };  object[] array2 = array1;  array2[0] = 5;  Console.ReadLine();  }  }  }  **OUTPUT:**    **StackOverflow Exception:-**  **Code:**  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace stackoverflow\_exception  {  internal class Program  {  static void Recurse (int val)  {  Console.WriteLine (val);  Recurse(++val);  }  static void Main(string[] args)  {  Recurse(1);  Console.ReadLine();  }  }  }  **Output:** |