|  |
| --- |
| DAY 12 Assignment  By  Nanam Vaishnavi  08-Feb-2022 |

3

|  |
| --- |
| **1) What is Exception Handling and why we need exception handling?** |
| **Ans:**  **Exceptional Handling :**  Exceptional Handling is a mechanism to handle runtime errors.  **Why we need exceptional handling :**   * We use exceptional handling to ensure that our application will not crash. * To make sure that we handle errors gracefully and display friendly messages. * We perform exceptional handling so that normal flow of application can be maintained even after runtime errors. |

|  |
| --- |
| **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 \_3\_Exception\_Handlings  {  internal class Program  {  static void Main(string[] args)  {  try  {  int a, b, c;  Console.WriteLine("Enter a: ");  a = Convert.ToInt32(Console.ReadLine());  Console.WriteLine("Enter b: ");  b = Convert.ToInt32(Console.ReadLine());    c = a / b;  Console.WriteLine(c);  Console.ReadLine();  }  catch (OverflowException)  {  Console.WriteLine("Enter numbers between 0 to 50000 ");  Console.ReadLine();  }  catch (FormatException)  {  Console.WriteLine("Enter Numbers. Please check again");  Console.ReadLine();  }  catch (DivideByZeroException)  {  Console.WriteLine("can't divide by zero");  Console.ReadLine();  }  }  } } |
| **OUTPUT** |
| **Overflow Exception** |
| **Format Exception DivideByZero Exception** |

|  |
| --- |
| **3. Research and write atleast 6 exceptions that occur in C# with sample code.** |
| 1. **ArgumentOutofRangeException**  * **Reason** : Raised when the value of argument is outside the range of valid values. * **Sample Code:**   var list = new List<string>();  Console.WriteLine("Number of items: {0}", list.Count);  try  {  Console.WriteLine("The first item: '{0}'", list[0]);  }  catch (ArgumentOutOfRangeException e)  {  Console.WriteLine(e.Message);  }  Console.ReadLine(); |
| 1. **IndexOutOfRangeException**  * **Reason:** Raised when an array index is outside the lower or upper bounds of an array or collection. * **Sample Code:**   public static void Main()  {  int[] tab = new int[5] { 2, 7, 3, 9, 5 };  int index = 5;  AccessElement(tab, index);  }  public static void AccessElement(int[] tab, int index)  {  Console.WriteLine("AccessElement {0}: {1}", index, tab[index]);  Console.ReadLine();  }  }  } |
| 1. **System.InvalidCastException**  * **Reason :** Handles errors generated during typecasting. * **SampleCode:**   try  {  StringBuild a = new StringBuild();  object b = new object();  StreamReader c = (StreamReader)b;  }  catch (InvalidCastException e)  {  Console.WriteLine("check is not defined");  } |
| 1. **OutOfMemoryException**  * **Reason :** Handles errors generated from insufficient free memory. * **Sample Code:**   public class Employee  {  public static void Main()  {  int v = new int (MinValue);  } |
| 1. **System.NullReferenceException**  * **Reason:** Handles errors generated from referencing a null object. * **Sample Code:**   try  {  string a = null;  if (a.Length == 0)  {  Console.WriteLine(a);  }  }  catch (NullReferenceException e)  {  Console.WriteLine("Access Null objects");  } |
| 1. **System.ArrayTypeMismatchException**  * **Reason :** Handles error generated when type is mismatched with the array type. * **Sample Code:**   try  {  string[] arr= { "Hello",”World”};  object[] arr2 = arr;  arr2[2] = 10;  }  catch (ArrayTypeMismatchException e)  {  Console.WriteLine("Not defined");  } |

|  |
| --- |
| **4. What is the use of "finally" block illustrate with an example.** |
| 1) A finally block contains crucial statements that must be executed whether the exception is occurred or not.  **Syntax :**  try  {  }  catch (Exception ex)  {  }  finally  {  }  **Example :**  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace \_3\_Exception\_Handlings  {    internal class Program  {  static void Main(string[] args)  {  try  {  int a, b, c;  Console.WriteLine("Enter a: ");  a = Convert.ToInt32(Console.ReadLine());  Console.WriteLine("Enter b: ");  b = Convert.ToInt32(Console.ReadLine());    c = a / b;  Console.WriteLine(c);  Console.ReadLine();  }    catch (FormatException)  {  Console.WriteLine("Enter Numbers. Please check again");  Console.ReadLine();  }  catch (DivideByZeroException)  {  Console.WriteLine("can't divide by zero");  Console.ReadLine();  }  finally  {  Console.WriteLine("Error Occured");  Console.ReadLine();  }  }  }  } |
| **OUTPUT** |
|  |

|  |
| --- |
| **5. Write the 5 points I explained about exception handling.** |
| 1. It is done to handle errors gracefully so that application will not crash. 2. A single try block can have multiple catch blocks. 3. Remember to write general exception only at the last. 4. Statements which are there in the finally block will be execute with irrespective of whether exception occurs or not. 5. **General Syntax :**   try  {  // Entire code  }  catch (Exception ex)  {  // Code to be executed when an exception is thrown.  }  finally  {  // all statements will execute.  } |

|  |  |
| --- | --- |
| **6) Differences between Compilation error and Runtime error.** | |
| **Compilation Error** | **Runtime Error** |
| 1) This is to check the syntax and semantic errors. | 1) This is to run the code. |
| 2) Error get detected by compiler without execution of the program. | 2) Only detect after execution of the program. |
| 3) Fixing an error at this stage is possible. | 3) Fixing an error requires going back to code. |

|  |
| --- |
| **7) Write any 6 compilation errors with small code snippet. Add compilation error screen shots.** |
| **1) Missing Semicolon:** |
| **2) Print the value without assigning** |
| **3) Spelling mistakes in the code** |
| **4) Namespace not included** |
| **5) Syntax Error** |
| **6) Logical error** |

|  |
| --- |
| **8) Write any 6 runtime errors with small code snippets and add run time error screen shots.** |
| **1) System.IndexOutOfRangeException** |
| **CODE** |
| public static void Main()  {  int[] tab = new int[5] { 2, 7, 3, 9, 5 };  int index = 5;  AccessElement(tab, index);  }  public static void AccessElement(int[] tab, int index)  { |
| **OUTPUT** |
|  |
| **2) System.NullReferenceException** |
| **CODE:** |
| {  string p = null;  if (p.Length == 0)  {  Console.WriteLine(p);  }  }  catch (NullReferenceException)  {  Console.WriteLine("Access Null objects");  } |
| **OUTPUT** |
|  |
|  |
| **3) OutOfMemoryException** |
| StringBuilder sb = new StringBuilder(15, 15);  sb.Append("Substring #1 ");  try  {  sb.Insert(0, "Substring #2 ", 1);  }  catch (OutOfMemoryException e)  {  Console.WriteLine("Out of Memory: {0}", e.Message);  Console.ReadLine();  } |
| **OUTPUT** |

**s**