|  |
| --- |
| **DAY 12 ASSIGNMENT**  **By**  **ARUN KUMAR YADLAPALLI**  **@**  **NB Healthcare Technologies PVT LTD.** |

|  |
| --- |
| **Q1) What is exception handling ? Why we need exception handling?** |
| **A)** While executing C# code, different errors can occur. When an error occurs, C# will normally stop and generate an error message. The technical term for this is : C# will throw an exception(throw an error. ). We need to handle that exception in a different way.   * Exception handling is done to handle the errors gracefully without displaying any errors to the end customers. |

|  |
| --- |
| **Q2) Simple division program and handling three exceptions with a super exception.** |
| **Code:**  **namespace Day\_12\_Project\_1**  **{**  **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;**  **}**  **catch (OverflowException e)**  **{**  **Console.WriteLine("Enter digits in the range of 9-99999");**  **Console.ReadLine();**  **}**    **catch (DivideByZeroException ex)**  **{**  **Console.WriteLine("never divide a number with 0");**  **Console.ReadLine();**  **}**    **catch (FormatException fe)**  **{**  **Console.WriteLine("only numbers dude");**  **Console.ReadLine();**  **}**    **catch (Exception gh)**  **{**  **Console.WriteLine("Why late .. call customer care");**  **Console.ReadLine();**  **}**  **finally**  **{**  **Console.WriteLine("Well done boy");**  **Console.ReadLine();**  **}**  **}**  **}**  **}** |
| **Output :** |

|  |
| --- |
| **Q3) 6 Exceptions that occur in C#.** |
| **A)**   1. **Invalid operation exception:** Raised when a method call is invalid in an object’s current state. 2. **Not supported exception:** Raised when the specified key for accessing a member in a collection is not exists. 3. **Out of memory exception:** Raised when a program does not get enough memory to execute a code. 4. **Timeout exception:** The time travel allotted to an operation has expired. 5. **Index out of range exception:** Raised when an array index is outside the lower (or) upper bounds of an array (or) collection. 6. **Argument out of Range exception:** Raised when the value of an argument is outside the range of valid values. |

|  |
| --- |
| **Q4) What is the use of “finally” block? Give an example.** |
| **A)** The finally statement lets you execute code after try and catch blocks, regardless of the result.   * Example:   Try  {  Int [] numbers = {1,2,3};  Console.WriteLine (numbers[10]);  }  Catch(exception e)  {  Console.WriteLine(“Something went wrong”);  }  Finally  {  Console.WriteLine (“The operation is done.”);  } |

|  |
| --- |
| **Q5) 5 Points about exception handling.** |
| **A)**   1. Exception handling is done to handle the errors gracefully without displaying any errors to the end customers. 2. A single try block can have multiple catch blocks. 3. Always write the general (or) super exception at the last. 4. Statements written in the finally block will be executed whether the exception occurs (or) not. 5. General syntax floe for writing exception is   try  {  }  catch  {  }  finally  {  } |

|  |
| --- |
| **Q6) Differences between Compilation an Runtime errors.** |

|  |  |
| --- | --- |
| **Compilation error** | **Runtime error** |
| Compilation or compile time errors occur due to typing mistakes in the code. | Runtime errors are the errors that when the program is in running state. |
| It prevents the code from running. | It prevents the code from executing . |
| It has syntax errors**.** | It generally shows execution errors like exceptional errors. |