|  |
| --- |
| Day 12 (8 feb) assignment  by P Ramakrishna |

|  |
| --- |
| **1 .What is Exception Handling and why we need exception handling.** |
| Exceptions provide a way to transfer control from one part of a program to another. C# exception handling is built upon four keywords: **try**, **catch**, **finally**, and **throw**.   * **try** − A try block identifies a block of code for which particular exceptions is activated. It is followed by one or more catch blocks. * **catch** − A program catches an exception with an exception handler at the place in a program where you want to handle the problem. The catch keyword indicates the catching of an exception. * **finally** − The finally block is used to execute a given set of statements, whether an exception is thrown or not thrown. For example, if you open a file, it must be closed whether an exception is raised or not. * **throw** − A program throws an exception when a problem shows up. This is done using a throw keyword.   **we need exception handling:**  Exception handling ensures that the flow of the program doesn't break when an exception occurs. For example, if a program has bunch of statements and an exception occurs mid way after executing certain statements then the statements after the exception will not execute and the program will terminate abruptly. |

|  |
| --- |
| 2 .Write the 5 points I explained about exception handling. |
| * Exception handling are done to handle the errors gracefully and display a friendly message to a end customer. * A single try bock can have multiple catch block. * Always try to remember to write general execution at last. * Statement writren in the final block is excuted with irrespective of excitation. * The general syntax flow id is try, catch and finally block . |

|  |  |
| --- | --- |
| 3.What is the use of "finally" block illustrate with an example | |
| Code: using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace day\_12\_project\_1  {  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(c);  Console.ReadLine();  }  catch (OverflowException)  {  Console.WriteLine("only numbers between 0 and 50000");  }  catch (DivideByZeroException)  {  Console.WriteLine("Cannot divide by zero");  }  catch (FormatException)  {  Console.WriteLine("only numbers are allowed.please double check");  }  catch (Exception)  {  Console.WriteLine(" please contact ABC@company");  }  finally  {  Console.WriteLine("Designed by Ramakrishna");  }  Console.ReadLine();  }  }  } | |
| Output: |

|  |
| --- |
| 4 .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 day\_12\_project\_1  {  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(c);  Console.ReadLine();  }  catch (OverflowException)  {  Console.WriteLine("only numbers between 0 and 50000");  }  catch (DivideByZeroException)  {  Console.WriteLine("Cannot divide by zero");  }  catch(FormatException)  {  Console.WriteLine("only numbers are allowed.please double check");  }  catch(Exception)  {  Console.WriteLine(" please contact ABC@company");  }  Console.ReadLine();  }  }  } |
| Output: |

|  |
| --- |
| **5 .Research and write atleast 6 exceptions that occur in C# with sample code.** |
| **System .nullReferenceException:**  **Reason:**  This exception is thrown when you try to call a method/property/indexer/etc. on a variable that contains a null reference—that is, it doesn’t point to any object. The code below will cause a null reference exception: |
| System.Net.WebException:Reason: This exception is network related. It’s thrown if an error occurs when accessing the network using a  pluggable protocol.[.](https://docs.microsoft.com/dotnet/framework/network-programming/introducing-pluggable-protocols) When handling this exception, remember to verify the Response property, which will contain the response returned by the remote host. System.Data.SqlClient.SqlException:Reason: This exception is related to the database, specifically, SQL Server. It’s thrown when SQL Server returns an error or warning. The class has a property called Errors, which is a collection containing one or more instances of the SqlError class. That, in turn, contains detailed information about the errors that occurred. System.OutOfMemoryException:Reason: What happens is that this exception doesn’t refer to the physical memory available.  Code:public class employee  {  Publicstatic voidmain()  {  String val new tring(“r”,int)  }  } System.InvalidCastException:Reason: This exception also has a self-explanatory name. It’s thrown when code fails to do a cast from one type to another because there isn’t a cast defined. The following code would throw an exception of this type:  Object o =”10”;  Int x = int (o); System.IO.IOException :Reason  it’s the exception thrown when things go wrong during IO operations. |

|  |
| --- |
| 6 .What is complication error and runtime error |
| Compile\_time Errors:   * These are the syntax errors which are detected by the compiler. * They prevent the code from running as it detects some syntax errors. * It includes syntax errors such as missing of semicolon(;), misspelling of keywords and identifiers etc. |
| Runtime Errors:   * These are the errors which are not detected by the compiler and produce wrong results. * They prevent the code from complete execution. * It includes errors such as dividing a number by zero, finding square root of a negative number etc. |

|  |
| --- |
| 7.Write a examples of Compile and run time Errors |
| **Compile-Time Errors**: Errors that occur when you violate the rules of writing syntax are known as Compile-Time errors. This compiler error indicates something that must be fixed before the code can be compiled. All these errors are detected by the compiler and thus are known as compile-time errors.  Most frequent Compile-Time errors are:    * Missing Parenthesis (**}**) * Printing the value of variable without declaring it * Missing semicolon (terminator).  |  | | --- | | 1.Use of unassigned variable | |  | | Semi colon excepted: | |  |  |  | | --- | | Spelling mistake errors: | |  |  |  | | --- | | Length over flow: | |  |  |  | | --- | | Syntax errors: | |  |  |  | | --- | | Logical errors: | |  | |
| **Run-Time Errors**:  Errors which occur during program execution(run-time) after successful compilation are called run-time errors. One of the most common run-time error is division by zero also known as Division error. These types of error are hard to find as the compiler doesn’t point to the line at which the error occurs.   * logic Error. A logic error occurs when a developer enters the wrong statements into the application's source code. ... * Memory Leak. * Division by Zero Error. * Undefined Object Error. * Input/Output Device Error. * Encoding Error. |

|  |
| --- |
| Index out of range |
|  |