|  |
| --- |
| Day 12 Assignment ByM.Pallavi |

1. What is Exception Handling and why we need exception handling.

|  |
| --- |
| Exception handling is responding to [exceptions](https://www.computerhope.com/jargon/e/exceptio.htm) when a computer program runs.  The process of catching the exception for converting the given exception message to an end-user understandable message and for stopping the abnormal termination of the program whenever runtime errors are occurring is called Exception Handling.  Examples include a [user](https://www.computerhope.com/jargon/u/user.htm) providing abnormal [input](https://www.computerhope.com/jargon/i/input.htm), a [file system](https://www.computerhope.com/jargon/f/filesyst.htm) error being encountered when trying to [read](https://www.computerhope.com/jargon/r/read.htm) or [write](https://www.computerhope.com/jargon/w/write.htm) a [file](https://www.computerhope.com/jargon/f/file.htm), or a program attempting to [divide by zero](https://www.computerhope.com/issues/ch000396.htm).  we need Exception Handling in C# because for two reasons.   1. To stop the abnormal termination of the program 2. To provide users understandable messages when an exception is raised. So that users can make a decision without the developer’s help. |

|  |
| --- |
| 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;  class Program  {  static void Main(string[] args)  {  try {  int a, b, c;  Console.WriteLine("enter a numnber");  a = Convert.ToInt32(Console.ReadLine());  Console.WriteLine("enter b value");  b = Convert.ToInt32(Console.ReadLine());  c = a / b;  Console.WriteLine("answer is= { 0}", c);  Console.ReadLine();  }  catch (OverflowException)  {  Console.WriteLine("only numbers between 0 and 5000000");  }  catch (DivideByZeroException)  {  Console.WriteLine("cannot divide with zero");  }  catch (FormatException)  {  Console.WriteLine("ONLY NUMBERS Are allowed");  }  catch(Exception)  {  Console.WriteLine("some error occured please contact @Admin.Mycompany.com");  }  }  } |
| Output: |

.

|  |
| --- |
| 3.What is the use of "finally" block illustrate with an example. |
| Uses:   * Finally, the block of code is executed regardless of what happens within the try block like an exception is thrown or not thrown, if there is a return statement, nothing matters. * The primary use of finally block of code is to release all the allocated expensive resources in the try block. * Finally, block ensures that an operation will be performed regardless of any exceptions thrown |
| Example:  using System;  namespace ConsoleApp1  {  namespace Hello\_World  {  class Program  {  int result;  Program()  {  result = 0;  }  public void division(int num1, int num2)  {  try  {  result = num1 / num2;  }  catch (DivideByZeroException e)  {  Console.WriteLine("Exception caught: {0}", e);  }  finally  {  Console.WriteLine("Result: {0}", result);  }  }  static void Main(String[] args)  {  Program d = new Program();  d.division(16,4 );  Console.ReadKey();  }  }  }  } |
| Output: |

4. Write the 5 points I explained about exception handling.

* 1.Exception handling done to handle errors gracefully without displaying any errors to end customers.
* 2.A single try block can have multliple catch blocks.
* 3.Always have general exaception is at last.
* 4.statemnts inside finally block executed irrespective of exceptions.

The general syntax for exception handling is

Try()

{

}

Catch()

{

}

Catch()

{

}

…

Finally{}

5.Research and write atleast 6 exceptions that occur in C# with sample code.

|  |
| --- |
| 1.[System. ArrayTypeMismatchException](https://docs.microsoft.com/en-us/dotnet/api/system.arraytypemismatchexception" \t "_blank)**:** |
| Reason:  when an attempt is made to store an element of the wrong type within an array. |
| Example:  Example code:  class Program  {  static void Main()  {  // Declares and assigns a string array.  // ... Then implicitly casts to base class object.  // ... Then assigns invalid element.  string[] array1 = { "cat", "dog", "fish" };  object[] array2 = array1;  array2[0] = 5;  }  } |
| Exception: |

|  |
| --- |
| 1. [NullReferenceException](https://docs.microsoft.com/en-us/dotnet/api/system.nullreferenceexception?view=netframework-4.8): : Raised when program access members of null object.   Reason :  values are declared to be an string array, but the number of elements that  it contains is never specified. |
| **Code:**  using System;  namespace ConsoleApp1  {  internal class Program  {  public static void Main()  {  string name = null;  var val = name.ToString();  Console.WriteLine(val);  }  }  } |
| **Output:** |

|  |
| --- |
| 3.[ArgumentException](https://docs.microsoft.com/en-us/dotnet/api/system.argumentexception?view=netframework-4.8" \t "_blank): Raised when a non-null argument that is passed to a method is invalid.  Reason: when a method is invoked and at least one of the passed arguments does not meet the parameter specification of the called method |
| **Program:**  sing System.Collections.Generic;  class Program  {  static void Main()  {  var dictionary = new Dictionary<string, int>();  int value = dictionary[null];  }  } |
| **Output:** |

|  |
| --- |
| **4.System.InvalidCastException:** The InvalidCastException occurs when an explicit cast is applied,and the type is not in the same path of the type hierarchy. The cast does not succeed. |
| Code:  using System.IO;  using System.Text;  class Program  { static void Main()  {  // Creates a new object instance of type StringBuilder.  // ... Then uses implicit cast to object through assignment.  // ... Then tries to use explicit cast to StreamReader, but fails.  StringBuilder reference1 = new StringBuilder();  object reference2 = reference1;  StreamReader reference3 = (StreamReader)reference2;  }  } |
| **Output:** |

|  |
| --- |
| **5. System.IO.IOException:**  Reason: The errors that are generated by input, the output is handled by this exception. . |
| using System;  using System.IO;  class Program  {  static void Main()  {  try  {  File.Open("C:\\nope.txt", FileMode.Open);  }  catch (IOException)  {  Console.WriteLine("IO");  }  }  } |
| **Output:** |

|  |
| --- |
| **6. System. OutOfMemoryException:** it occurs when not enough memory is available. |
| using System;  class Program  {  static void Main()  {  // Attempt to create a string of 2.1 billion chars.  // ... This results in an out-of-memory error.  // ... It would require 4.2 billion bytes (4 gigabytes).  string value = new string('a', int.MaxValue);  }  } |
|  |

6. Write any 6 runtime errors with small code snippets and add run time error screen shots.

|  |
| --- |
| ERROR 1:  Program:  using System;  class Program  { static void Main()  {  {  int a = 10;  int c = a / 0;// Here number divisible zero error occurs  }  }  } |
| Output: |
| ERROR 2:  while accessing an element of a collection with an index that is outside of its range. It occurs when an invalid index is used to access a member of a collection. |
| Program:  namespace ConsoleApp3  {  internal class Program  {  static void Main(string[] args)  {  int[] arr = new int[] { 10, 29, 38, 44, 43 };  Console.WriteLine(arr[0]);  Console.WriteLine(arr[1]);  Console.WriteLine(arr[2]);  Console.WriteLine(arr[3]);  Console.WriteLine(arr[4]);  Console.WriteLine(arr[5]);  }  }  } |
| Output: |

|  |
| --- |
| ERROR 3: |
| using System;  class TestException : Exception  {  public override string Message  {  get  {  return "This exception means something bad happened";  }  }  }  class Program  {  static void Main()  {  try  {  throw new TestException();  }  catch (TestException ex)  {  Console.WriteLine(ex);  }  }  } |
| Output: |

|  |
| --- |
| ERROR 4: |
| using System;  using System.Collections.Generic;  class Program  {  static void Main()  {  var list = new List<int>() { 10, 20, 30 };  // Try to remove an element in a foreach list.  foreach (int value in list)  {  Console.WriteLine("ELEMENT: {0}", value);  list.Remove(value);  } }} |
| Output: |
| ERROR 5: |
| Program:  using System;  class Program  { static void Main()  {  {  int a = 10;  int c = a / 0;// Here number divisible zero error occurs  } }  } |
|  |

|  |
| --- |
| ERROR 6: |
| Code:  using System.IO;  using System.Text;  class Program  { static void Main()  {  // Creates a new object instance of type StringBuilder.  // ... Then uses implicit cast to object through assignment.  // ... Then tries to use explicit cast to StreamReader, but fails.  StringBuilder reference1 = new StringBuilder();  object reference2 = reference1;  StreamReader reference3 = (StreamReader)reference2;  }  } |
| Output: |

8. What is compilation and Runtime Error . Write Atleast 3 differences between them.

Runtime errors:

These errors occur during the run-time program execution after a successful compilation. Division error is one of the most common errors (runtime). It is very difficult for a compiler to find out a runtime error because it cannot point out the exact line at which this particular error occurs.

Compile time Errors:

These errors occur when we violate the rules present in a syntax. The compile-time error indicates something that we need to fix before compiling the code. A compiler can easily detect these errors.

Differences:

|  |  |
| --- | --- |
| **Complie Time Errors** | **Runtime errors** |
| 1.The code understands and checks for the object, which invokes a method. | 1.The code cannot understand which object is invoking the method, and code is compiled without knowing that information. |
| 2.These errors are referenced to an error in syntax or semantics. | 2.These errors are a reference to the execution of the code in a runtime environment. |
| 3.These errors can be fixed during the code development itself. | 3.These errors are rather difficult to interpret during development and can be fixed only when it starts showing up in the runtime environment. |