Day 12 - Assignment

By Manoj Karnatapu - NBHealthCareTechnologies

Assignment 1

What is Exception Handling and why we need exception handling ?

Answer

- ✓ Exception Handling 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.

---------- OR ----------

- ✓ Exception Handling is done to ensure that our application will not crash.
- ✓ Exception Handling will not display any technical details, to make sure we handle errors gracefully and display friendly messages.

Note:

- ✓ In C#, exception is an event or object which is thrown at runtime. All exceptions are derived from System namespace.

Assignment 2

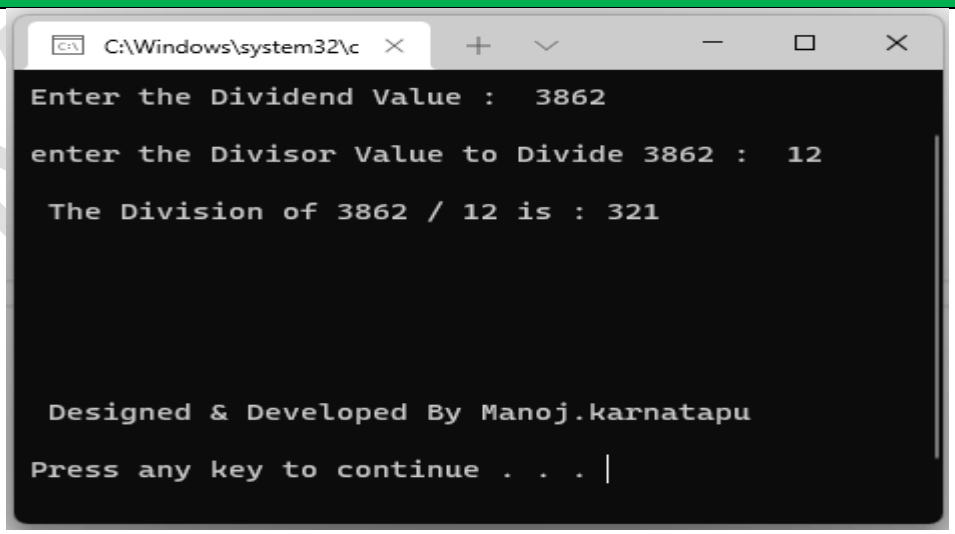
Write a C# Code for Division Program, add 3 Exceptions along with super exception/

Code

```
using System;
// Author : Manoj.Karnatapu
// Purpose : C# Code for Division program, & handling 3 exceptions along with super
exception catch at the End.
// for Reference, check Day12Project1 in the same Repository of GitHub.
namespace Day12Project1
{
    /// <summary>
    /// This is a Simple Division Program, With Catching Exceptions.
    /// </summary>
    internal class Program
    {
        /// <summary>
        /// Implementing a Division program, With an User Friendly Experience.
        /// </summary>
        /// <param name="args"></param>
        static void Main(string[] args)
        {
```

```
            // Try block with Logical Code Implementation Section
            try
            {
                int a, b , c;
                Console.Write("\nEnter the Dividend Value :  ");
                a = Convert.ToInt32(Console.ReadLine());
                Console.Write($"\nenter the Divisor Value to Divide {a} :  ");
                b= Convert.ToInt32(Console.ReadLine());
                c = a / b;
                Console.WriteLine($"\n The Division of {a} / {b} is : {c}");
                Console.ReadLine();
            }
            // Catch Block for OverFlowException
            catch (OverflowException)
            {
                Console.WriteLine("\nPlease Do Enter the Numbers Only in the Range of 0 to
50000");
            }
            // Catch Block for DivideByZeroException
            catch (DivideByZeroException)
            {
                Console.WriteLine("\nPlease Do Provide divisor Value, a Non-Zero Number to
Do Perfect Division.");
            }
            // Catch Block For FormatException
            catch (FormatException)
            {
                Console.WriteLine("\nPlease, Do Enter only Integers. Strings / Special
Characters are not Allowed to do Division as per Mathematics Standard.");
            }

            // Catch Block For Other Kind Of Exceptions.
            catch (Exception)
            {
                Console.WriteLine("\n\t Some Error Has Occured, Please Contact the Admin");
            }

            finally
            {
                Console.WriteLine("\n\n\n\n\n Designed & Developed By Manoj.karnatapu");
                Console.ReadLine();
            }
        }
    }
}
```

Output



Assignment 3

Research & Write at least 6 exceptions that occur in C# with Code Snippet.

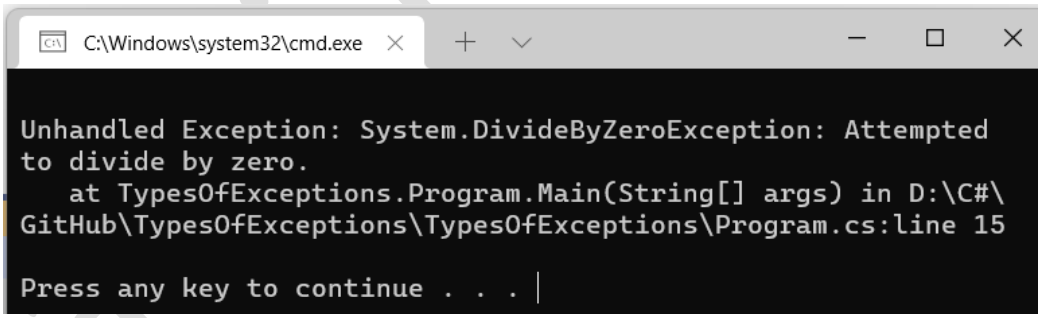
Answer

(1). Exception Type : DivideByZeroException

Code :

```
using System;

namespace TypesOfExceptions
{
    internal class Program
    {
        /// <summary>
        /// DivideByZeroException
        /// </summary>
        /// <param name="args"></param>
        static void Main(string[] args)
        {
            int a = 5;
            int b = 0;
            int c = a / b;
            Console.WriteLine(c);
        }
    }
}
```

Output :



(2). Exception Type : System.IO.FileNotFoundException

Code :

```
using System;
using System.IO;

namespace TypesOfExceptions
{
    internal class Program
    {
        /// <summary>
        /// IOException
        /// </summary>
        /// <param name="args"></param>
```

```
        static void Main(string[] args)
        {
            File.Open("D:\\ex.txt", FileMode.Open);
        }
    }
}
```

Output :

```
C:\Windows\system32\cmd.exe

Unhandled Exception: System.IO.FileNotFoundException: Could not find file 'D:\ex.txt'.
   at System.IO.__Error.WinIOError(Int32 errorCode, String maybeFullPath)
   at System.IO.FileStream.Init(String path, FileMode mode, FileAccess access, Int32 rights, Boolean useRights, FileShar
e share, Int32 bufferSize, FileOptions options, SECURITY_ATTRIBUTES secAttrs, String msgPath, Boolean bFromProxy, Boolea
n useLongPath, Boolean checkHost)
   at System.IO.FileStream..ctor(String path, FileMode mode, FileAccess access, FileShare share)
   at System.IO.File.Open(String path, FileMode mode)
   at TypesOfExceptions.Program.Main(String[] args) in D:\C#\GitHub\TypesOfExceptions\TypesOfExceptions\Program.cs:line
14
Press any key to continue . . .
```

(3). Exception Type : StackOverflowException

Code :

```
using System;

namespace TypesOfExceptions
{
    internal class Program
    {
        static void Recurse(int val)
        {
            Console.WriteLine(val);
            Recurse(++val);
        }
        /// <summary>
        /// StackOverFlowException
        /// </summary>
        /// <param name="args"></param>
        static void Main(string[] args)
        {
            Recurse(0);
        }
    }
}
```

Output :

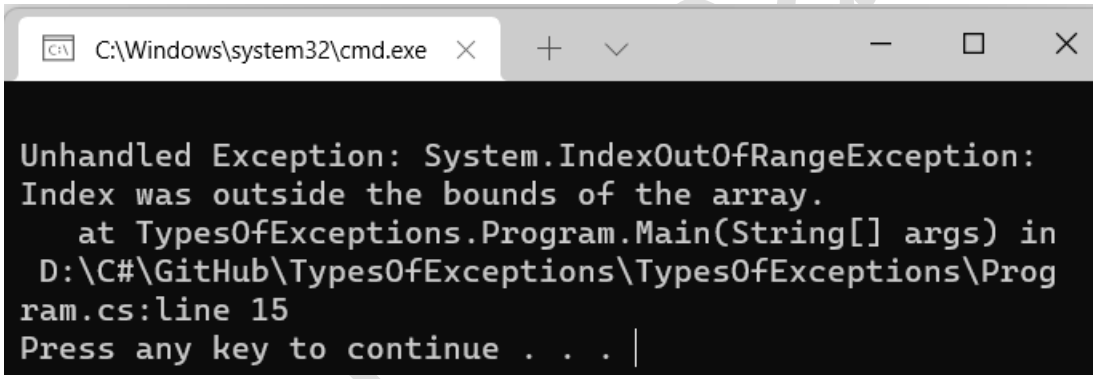
```
C:\Windows\system32\cmd.exe

14307
14308
14309
14310
14311
14312

Process is terminated due to StackOverflowException.
Press any key to continue . . .
```

(4). Exception Type : IndexOutOfRangeException

Code :

```
using System;
namespace TypesOfExceptions
{
    internal class Program
    {
        /// <summary>
        /// IndexOutOfRangeException
        /// </summary>
        /// <param name="args"></param>
        static void Main(string[] args)
        {
            int[] arr = new int[10];
            arr[0] = 10;
            arr[10] = 20;
            arr[20] = 30;
        }
    }
}
```

Output :



(5). Exception Type : ArrayTypeMismatchException

Code :

```
using System;
namespace TypesOfExceptions
{
    internal class Program
    {
        /// <summary>
        /// ArrayTypeMismatchException
        /// </summary>
        /// <param name="args"></param>
        static void Main(string[] args)
        {
            string[] arr1 = { "Welcome", "to", "CSharp" };
            object[] arr2 = arr1;
            arr2[0] = 8;
        }
    }
}
```

Output :

```
C:\Windows\system32\cmd.exe

Unhandled Exception: System.ArrayTypeMismatchException: Attempted to access
 an element as a type incompatible with the array.
   at TypesOfExceptions.Program.Main(String[] args) in D:\C#\GitHub\TypesOf
Exceptions\TypesOfExceptions\Program.cs:line 14
Press any key to continue . . .
```

(6). Exception Type : OutOfMemoryException

Code :

```
using System;
namespace TypesOfExceptions
{
    internal class Program
    {
        /// <summary>
        /// OutOfMemoryException
        /// </summary>
        /// <param name="args"></param>
        static void Main(string[] args)
        {
            string val = new string('r', int.MaxValue);
        }
    }
}
```

Output :

```
C:\Windows\system32\

Unhandled Exception: OutOfMemoryException.
Press any key to continue . . .
```

Assignment 4

What is the Use of Finally Block, illustrate with an example ?

Answer

By using a finally block, you can run code even if an exception occurs in the try block and you can clean up any resources that are allocated in a try block.

Typically, the statements of a finally block run when control leaves a try statement.

Normal Code

```
using System;

// Purpose : if we use calculation in Try block, some of the execution may skip, because
of Error Occured.

namespace FinallyBlockUsage
{

    internal class Program
    {
        static void Main(string[] args)
        {
            int a, b, c, d, e, f;
            Console.WriteLine("\n\t\t ----------- Arithmatic Operations Calculations ----
-------");
            Console.Write("Enter Any number for Value-1 :  ");
            a = Convert.ToInt32(Console.ReadLine());
            Console.Write("Enter Any number for Value-2 :  ");
            b = Convert.ToInt32(Console.ReadLine());
            try
            {
                d = a / b;
                Console.WriteLine($"The Division of {a} and {b} is :  {d}");
                c = a + b;
                Console.WriteLine($"The Addition of {a} and {b} is :  {c}");
                e = a - b;
                Console.WriteLine($"The Subtraction of {a} and {b} is :  {e}");
                f = a * b;
                Console.WriteLine($"The Multiplication of {a} and {b} is :  {f}");
            }
            catch (DivideByZeroException)
            {
                Console.WriteLine("The Value2 Can't be Zero to Do Division operation");
            }
            finally
            {
                Console.ReadLine();
            }
        }
    }
}
```

Output

```
C:\Windows\system32\cmd.exe

                    ----------- Arithmatic Operations Calculations -----------
Enter Any number for Value-1 :  25
Enter Any number for Value-2 :  0
The Value2 Can't be Zero to Do Division operation

Press any key to continue . . .
```

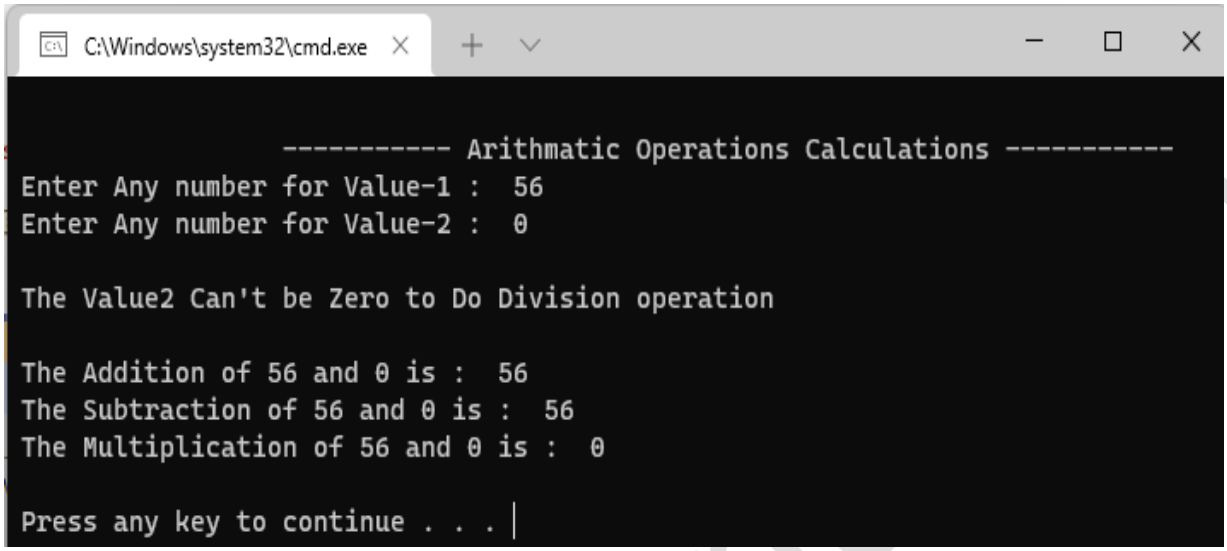
Here, we are implementing addition, subtraction, Multiplication in finally because, if we use in try block itself. When we come across the DivideByZero Exception. Though the Code is logically correct to do other arithmetic operations like add, sub, mul. Controller will not execute. To overcome this issue. We assign in finally block, irrespective of Division Operator in the try block.

Using Finally Block code:

```
using System;

// Purpose : Now Irrespective of Divide By Zero Error, we can see the Execution of Other
Operations.

namespace FinallyBlockUsage
{

    internal class Program
    {
        static void Main(string[] args)
        {
            int a, b, c, d, e, f;
            Console.WriteLine("\n\t\t ----------- Arithmatic Operations Calculations ----
-------");
            Console.Write("Enter Any number for Value-1 :  ");
            a = Convert.ToInt32(Console.ReadLine());
            Console.Write("Enter Any number for Value-2 :  ");
            b = Convert.ToInt32(Console.ReadLine());
            try
            {
                d = a / b;
                Console.WriteLine($"The Division of {a} and {b} is :  {d}");

            }
            catch (DivideByZeroException)
            {
                Console.WriteLine("\nThe Value2 Can't be Zero to Do Division
operation\n");
            }
            finally
            {
                c = a + b;
                Console.WriteLine($"The Addition of {a} and {b} is :  {c}");
                e = a - b;
                Console.WriteLine($"The Subtraction of {a} and {b} is :  {e}");
                f = a * b;
                Console.WriteLine($"The Multiplication of {a} and {b} is :  {f}");
                Console.ReadLine();
```

```
            }
        }
    }
}
```

Output



Assignment 5

What are the points about Exception Handling discussed in the class ?

Answer

- Exception Handling is done to Handle errors gracefully.
- Single **try** block can have multiple catch blocks.
- General Exception block should be implemented at the end of all exceptional catch blocks.
- Statements written in Finally Block are Executed irrespective of Exceptions.
- General Syntax for Exception Handling is :

```
try
{
    // Block of Code To Be Executed
}
catch (Exception)
{
    // Exception Message, for displaying purpose.
}
finally
{
    // These Block of statements will be executed, irrespective of Exceptions Occure.
}
```

Assignment 6

What are Compilation Errors& Runtime Errors, Write at least 3 differences.

Answer

Compilation Error	Runtime Error
1). This Error generally refers to the errors corresponding to the semantics or syntax	1). A runtime error refers to the error that we encounter during the code execution during runtime.
2). Most Compilation Errors are of Developer Mistakes.	2). Most Runtime Errors are of Logical Errors.
3). These Type of Errors, can easily identify and can be resolved easily.	3). These Errors are Hard to Identify and resolve them.
4). Compilers can easily detect compile-time errors during the development of code.	4). A compiler cannot easily detect a runtime error. Thus, we need to identify it during the execution of code

Assignment 7

Write any 6 Compilation errors with sample code snippet & compilation Errors Screenshots.

Answer

1. Not having proper knowledge of type casting.

```
using System;
using System.Collections.Generic;

namespace CompilationErrorsProject
{
    0 references
    internal class Program
    {
        0 references
        static void Main(string[] args)
        {
            int a = 100;

            string b = a;
            Console.WriteLine(b);
        }
    }
}
```

```
Build started...
1>------ Build started: Project: CompilationErrorsProject, Configuration: Debug Any CPU ------
1>D:\C#\GitHub\CompilationErrorsProject\CompilationErrorsProject\Program.cs(12,24,12,25): error CS0029: Cannot implicitly convert type 'int' to 'string'
========== Build: 0 succeeded, 1 failed, 0 up-to-date, 0 skipped ==========
```

2. Not using the proper naming conventions

```
using System;
using System.Collections.Generic;

namespace CompilationErrorsProject
{
    0 references
    internal class Program
    {
        0 references
        static void Main(string[] args)
        {
            string b = "Manoj";
            Console.Writeline(b);
        }
    }
}
```

```
Build started...
1>------ Build started: Project: CompilationErrorsProject, Configuration: Debug Any CPU ------
1>D:\C#\GitHub\CompilationErrorsProject\CompilationErrorsProject\Program.cs(11,21,11,30): error CS0117: 'Console' does not contain a definition for 'Writeline'
========== Build: 0 succeeded, 1 failed, 0 up-to-date, 0 skipped ==========
```

3. Not importing Proper namespace.

```
//using System;
using System.Collections.Generic;

namespace CompilationErrorsProject
{
    0 references
    internal class Program
    {
        0 references
        static void Main(string[] args)
        {
            string b = "Manoj";
            Console.WriteLine(b);
        }
    }
}
```

```
Build started...
1>------ Build started: Project: CompilationErrorsProject, Configuration: Debug Any CPU ------
1>D:\C#\GitHub\CompilationErrorsProject\CompilationErrorsProject\Program.cs(11,13,11,20): error CS0103: The name 'Console' does not exist in the curre
========== Build: 0 succeeded, 1 failed, 0 up-to-date, 0 skipped ==========
```

4. Missing Semicolon.

```
using System;
using System.Collections.Generic;

namespace CompilationErrorsProject
{
    0 references
    namespace CompilationErrorsProject
    {
        0 references
        static void Main(string[] args)
        {
            string b = "Manoj"
            Console.WriteLine(b);
        }
    }
}
```

```
Build started...
1>------ Build started: Project: CompilationErrorsProject, Configuration: Debug Any CPU ------
1>D:\C#\GitHub\CompilationErrorsProject\CompilationErrorsProject\Program.cs(10,31,10,31): error CS1002: ; expected
========== Build: 0 succeeded, 1 failed, 0 up-to-date, 0 skipped ==========
```

5. Not using proper Methods & properties.

```
using System;
using System.Collections.Generic;

namespace CompilationErrorsProject
{
    0 references
    internal class Program
    {
        0 references
        static void Main(string[] args)
        {
            int a = 4;
            int square_root = Math.SquareRoot(a);
        }
    }
}
```

```
Build started...
1>------ Build started: Project: CompilationErrorsProject, Configuration: Debug Any CPU ------
1>D:\C#\GitHub\CompilationErrorsProject\CompilationErrorsProject\Program.cs(11,36,11,46): error CS0117: 'Math' does not contain a definition for 'SquareRoot'
========== Build: 0 succeeded, 1 failed, 0 up-to-date, 0 skipped ==========
```

Assignment 8

Write any 6 Runtime errors with sample code snippet & Runtime Errors Screenshots.

Answer

1. Divide by zero Runtime Error.

```
using System;
using System.Collections.Generic;

namespace RuntimeErrorsProject
{
    0 references
    internal class Program
    {
        0 references
        static void Main(string[] args)
        {
            int a, b, c;
            Console.WriteLine("Enter any number : ");
            a = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Enter any numbers : ");
            b = Convert.ToInt32(Console.ReadLine());

            c = a / b;

            Console.WriteLine("The division of {a} / {b} is :  {c}");
        }
    }
}
```

```
C:\Windows\system32\cmd.exe
Enter any number :
15
Enter any numbers :
0

Unhandled Exception: System.DivideByZeroException: Attempted to divide by
zero.
   at RuntimeErrorsProject.Program.Main(String[] args) in D:\C#\GitHub\Com
pilationErrorsProject\CompilationErrorsProject\Program.cs:line 16
Press any key to continue . . .
```

2. Out of Memory Runtime error

```
using System;
using System.Collections.Generic;

namespace RuntimeErrorsProject
{
    0 references
    public class check
    {
        //main method is called
        0 references
        public static void Main()
        {
            // a string variable is created and tried to store 2.1 billion characters and this causes an out of memory exception
            string val = new string('r', int.MaxValue);
        }
    }
}
```

```
C:\Windows\system32\
Unhandled Exception: OutOfMemoryException.
Press any key to continue . . .
```

3. Null Reference Runtime Error.

```
using System;
using System.Collections.Generic;

namespace RuntimeErrorsProject
{
    0 references
    class check
    {
        //main method is called
        0 references
        static void Main()
        {
            //a string variable is defined, and it is referencing to null
            string value = null;
            //the length of the value referencing to null is checked
            //if it is equal to zero causing an exception
            if (value.Length == 0)
            {
                Console.WriteLine(value);
            }
        }
    }
}
```

```
C:\Windows\system32\cmd.exe

Unhandled Exception: System.NullReferenceException: Object referenc
e not set to an instance of an object.
   at RuntimeErrorsProject.check.Main() in D:\C#\GitHub\Compilation
ErrorsProject\CompilationErrorsProject\Program.cs:line 15
Press any key to continue . . .
```

4. Invalid Cast Runtime Error.

```
using System.IO;
using System.Text;
//a class called check is defined
0 references
class check
{
    //main method is called
    0 references
    static void Main()
    {
        // an instance of the string builder class is created
        // which is then assigned to a new object through implicit
        // casting and then casting is tried explicitly to convert
        // the instance of stringbuilder class to streamreader class

        StringBuilder ref1 = new StringBuilder();
        object ref2 = ref1;
        StreamReader ref3 = (StreamReader)ref2;
    }
}
```

```
C:\Windows\system32\cmd.exe

Unhandled Exception: System.InvalidCastException: Unable to cast object of
type 'System.Text.StringBuilder' to type 'System.IO.StreamReader'.
   at check.Main() in D:\C#\GitHub\CompilationErrorsProject\CompilationErro
rsProject\Program.cs:line 13
Press any key to continue . . .
```

5 . Array Type Mismatch Runtime Error

```
using System.IO;
using System.Text;
//a class called check is defined
0 references
class check
{
    //main method is called
    0 references
    static void Main()
    {
        // a string is defined and assigned the values
        // which is then assigned to object class array
        // and then an integer is tried to put in the
        // same array which causes an exception
        string[] arr1 = { "Welcome", "to", "CSharp" };
        object[] arr2 = arr1;
        arr2[0] = 8;
    }
}
```

C:\Windows\system32\cmd.exe

```
Unhandled Exception: System.ArrayTypeMismatchException: Attempted
to access an element as a type incompatible with the array.
   at check.Main() in D:\C#\GitHub\CompilationErrorsProject\Compil
ationErrorsProject\Program.cs:line 12
Press any key to continue . . .
```

6 . Stack over flow Runtime Error

```
using System;
//a class called check is defined
0 references
public class check
{
    // a method called recurse is defined which
    // takes a value as parameter and increases
    // its value by one
    2 references
    static void Recurse(int val)
    {
        // since we have written a recursive
        // loop and 0 is passed as a parameter,
        // it ends in an infinite loop causing exception
        Console.WriteLine(val);
        Recurse(++val);
    }
    //main method is called
    0 references
    public static void Main()
    {
        //The recurse method is called to start
        //the infinite recursion
        Recurse(0);
    }
}
```

```
C:\Windows\system32\cmd.exe

14334
14334
14334
14335

Process is terminated due to StackOverflowException.
Press any key to continue . . .
```