Day 12(08-02-2022) Assignment
By
Sudha Kumari Sugasani

Q1.What is Exception Handling?
Why we need Exception Handling?

- Exception handling is a process to handle runtime exceptions.
- Exception handling is done to ensure that our application will not crash or will not display any technical details and to make sure we handle errors gracefully and display user friendly messages.

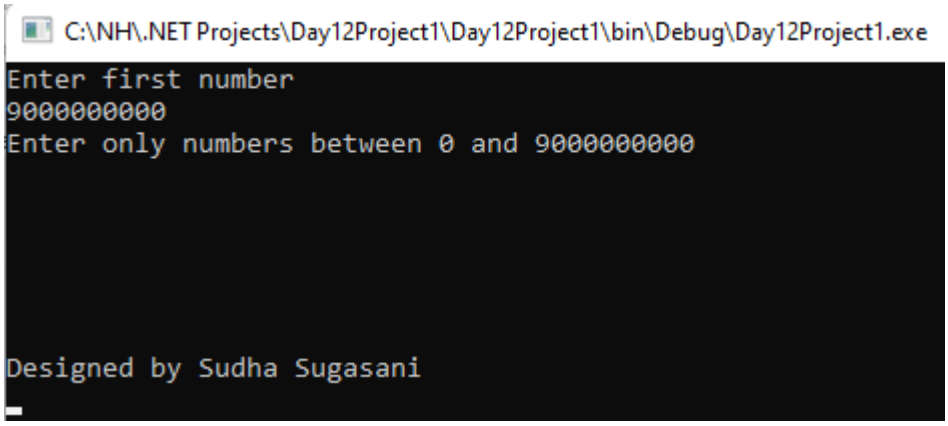
Q2.Write a simple division program to handle three exceptions discussed in the class, also add super exception at the last.

Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Day12Project1
{


/*************************************************************************
     * Author:Sudha Kumari Sugasani
     * Purpose:Program to handle three exceptions and adding super exception
     *         at last.
     *
*************************************************************************/
    internal class Program
    {
        static void Main(string[] args)
        {

            try
            {
                int fn, sn, division;
                Console.WriteLine("Enter first number");
                fn=Convert.ToInt32(Console.ReadLine());
                Console.WriteLine("Enter second number");
                sn=Convert.ToInt32(Console.ReadLine());
                division = fn / sn;
                Console.WriteLine($"The division of two numbers is
{division}");

            }
            catch(OverflowException)
            {
                Console.WriteLine("Enter only numbers between 0 and
9000000000");
                Console.ReadLine();
            }
            catch(DivideByZeroException)
            {
                Console.WriteLine("Cannot divide with zero");
                Console.ReadLine();
            }
            catch(FormatException)
```

```
            {
                Console.WriteLine("Enter only numbers");
                Console.ReadLine();
            }
            catch(Exception)
            {
                Console.WriteLine("Some error occured,Please contact admin");
                Console.ReadLine();
            }
            finally
            {
                Console.WriteLine("\n\n\n\n\nDesigned by Sudha Sugasani");
                Console.ReadLine();
            }
        }
    }
}
```

Output:



Q3.Research and write atleast six exceptions that occur in C# with sample code.

Ex1:Null reference Exception:

Reason:If we assign null value to the name variable and trying to do Length operation

Code:

1.

```
 static void Main(string[] args)
        {
            //NullReferenceException
            string name = "";
            if(name.Length>0)
            {
            Console.WriteLine("Hi");
            Console.ReadLine();
            }
        }
```

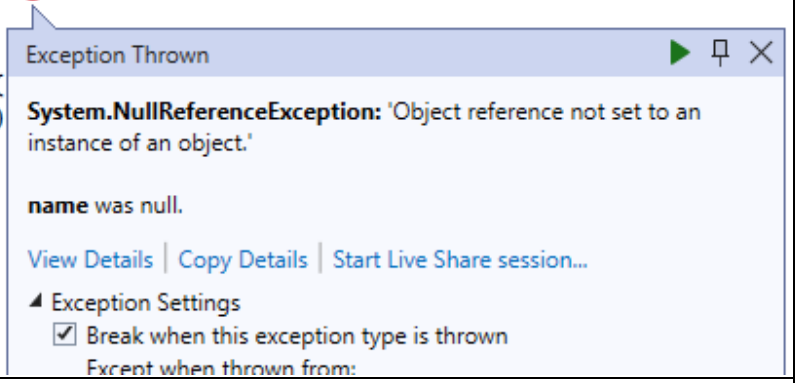
2.

```
static void Main(String[] args)
        {

            //NullReferenceException
            List<int> test = new List<int>() { 4, 5, 6, 7, 8, 9, 10, 11, 12,
13, 14, 15 };
            int a = test[-1];
```

```
            Console.ReadLine();
        }
```

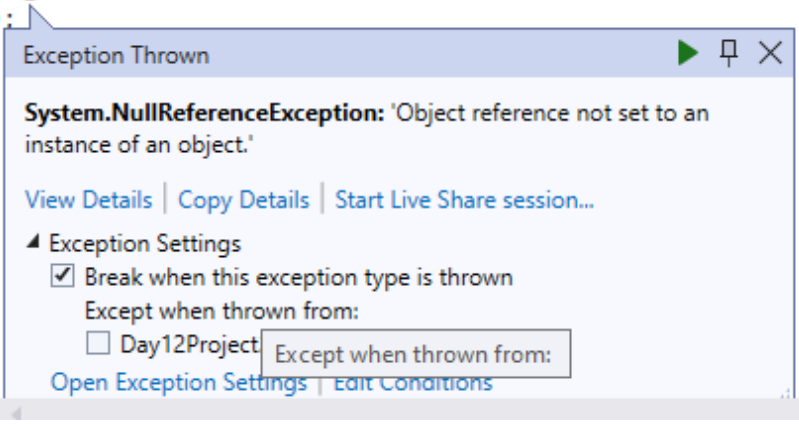
Exception1:

```
internal class Program
{
    0 references
    static void Main(string[] args)
    {
        //NullReferenceException
        string name = "";
        if(name.Length>0)
        {
        Console.WriteLine(
        Console.ReadLine()
        }
    }
}
```



Exception2:

```
//NullReferenceException
List<int> test = new List<int>() { 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 };
int a = test[-1];
Console.ReadLine();
```



Ex2:InvalidCastException:

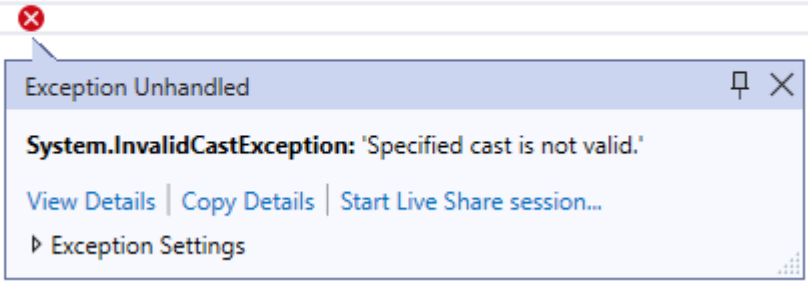
Reason:When we are doing type casting,if type casting is not supported,it will occur.

Code:

```
static void Main(string[] args)
        {
            //InvalidCastException
            Object obj=new Object();
            int a;
            a=(int)obj;
            Console.WriteLine(a);
            Console.ReadLine();            }
        }
```

Exception:

```
object obj = new object();
int a;
a=(int)obj;
```



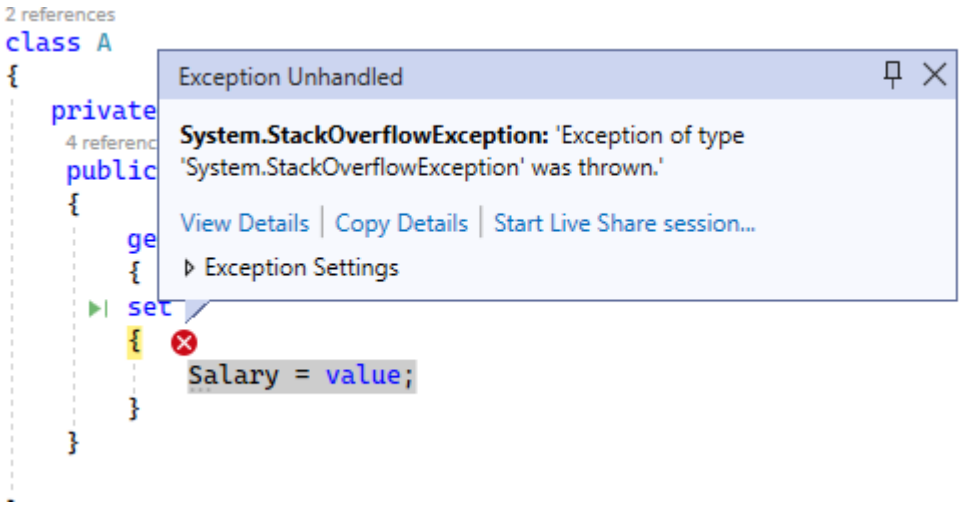
Example3:Stack OverflowException

Reason:The exception that is raised whenever we are calling too many methods/properties into one another.

Code:

```
//StackOverflowException
class A
{
   private int salary;
    public int Salary
    {
        get
        { return Salary; }
        set
        {
            Salary = value;
        }
    }

}
static void Main(String[] args)
 {

    A obj=new A();
    obj.Salary = 5;
    Console.WriteLine(obj.Salary);
    Console.ReadLine();
 }
```

Exception:



Example4:ArrayTypeMismatchException:

Purpose:When the system cannot convert the element to the type declared for the Array

Code:

```
static void Main(String[] args)
        {

            //ArrayTypeMismatchException
            string[] data1 = { "hi", "hello" };
            object[] data2 = data1;
            data2[0] = 55;
            Console.WriteLine(data1);
            Console.ReadLine();

        }
```

Exception:

```
//ArrayTypeMismatchException
string[] data1 = { "hi", "hello" };
object[] data2 = data1;
data2[0] = 55;
Console.WriteLine(data1);
Console.ReadLin
```

Exception Unhandled

System.ArrayTypeMismatchException: 'Attempted to access an element as a type incompatible with the array.'

This exception was originally thrown at this call stack:
Day12Project2.Program.Main(string[]) in Program.cs

View Details | Copy Details | Start Live Share session...

Exception Settings

Break when this exception type is thrown

Except when thrown from:

Day12Project2.exe

Open Exception Settings | Edit Conditions

Ex5.ArgumentNullException:

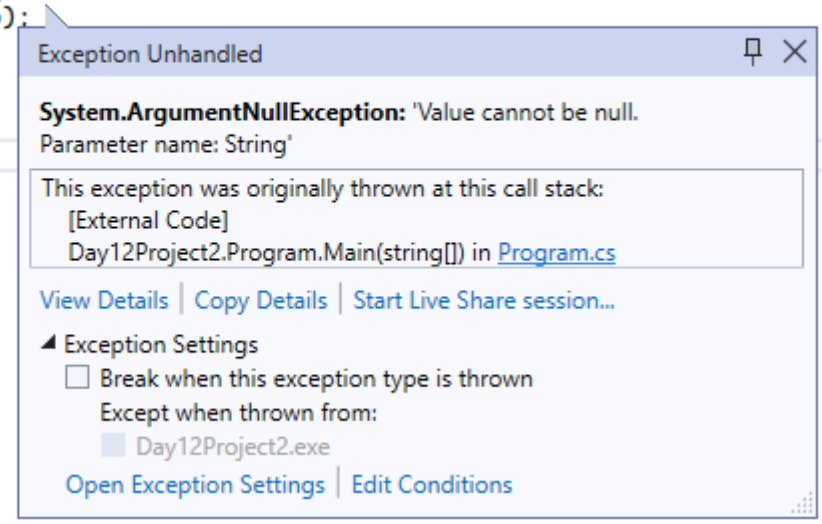
Purpose: The exception thrown when a null reference is passed to a method that does not accept it as a valid argument.

Code:

```
static void Main(String[] args)
        {

            //ArgumentNullException
            string a = null;
            int b=int.Parse(a);
            Console.WriteLine(b);
            Console.ReadLine();

        }
```

Exception:

```
//ArgumentNullException
string a = null;
int b=int.Parse(a);
Console.WriteLine(b);
Console.ReadLine();
```



Example 6:IndexOutOfRangeEXception

Purpose :If you want to assign the exceeded index of values it will raise exception.

Code:

```
static void Main(String[] args)
        {
            //IndexOutOfRangeException
            int []a = new int[] { 1, 2, 3, 4, 5, 6, 7, 8 };
            a[10] = 15;
            Console.WriteLine(a[10]);
            Console.ReadLine();

        }
```

Exception:

```
//IndexOutOfRangeException
int []a = new int[] { 1, 2, 3, 4, 5, 6, 7, 8 };
a[10] = 15;
Console.WriteLine(a[10]);
Console.Read
```

Q4.What is the use of Finally block illustrate with an example.

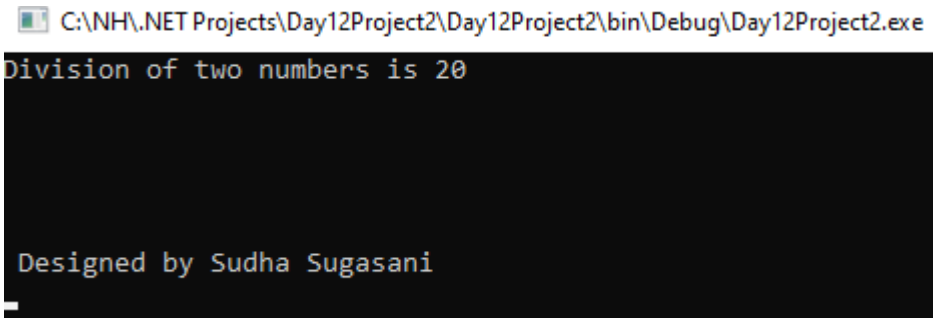
Statements in the finally block will be excecuted whether we get the exception or not.

Example code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace Day12Project2
{
    internal class Program
    {
        /*******************************************************
         * Author:Sudha Sugasani
         * Purpose:Program to show the use of finally block
         * *****************************************************/
        static void Main(string[] args)
        {
            try
            {
                int a = 100;
                int b = 5;
                int c = a / b;
                Console.WriteLine($"Division of two numbers is {c}");
            }
            catch(Exception)
            {

            }
            finally
            {
                Console.WriteLine("\n\n\n\n\n Designed by Sudha Sugasani");
                Console.ReadLine();
            }
        }
    }
}
```

Output:



Q5.Write the five points explained about Exception Handling

1.Exception Handling is done to ensure that our application will not crash or will not display any technical details and to make sure we handle errors gracefully and display user friendly mesaage.
2.A single try block can have multiple catch blocks.
3.Always write general exception at last.
4.Staments inside the finally block will execute whether we get the exception or not
5.General syntax for writing Exception Handling is

```
try
{
  //Error related code
}
catch
 {
    // Type of Exception
```

```
    }
    finally
    {
       //Message that we want to print irrespective of Exception
    }
```

Q6.What is Compilation and Runtime error .
Write atleast three differences between them.

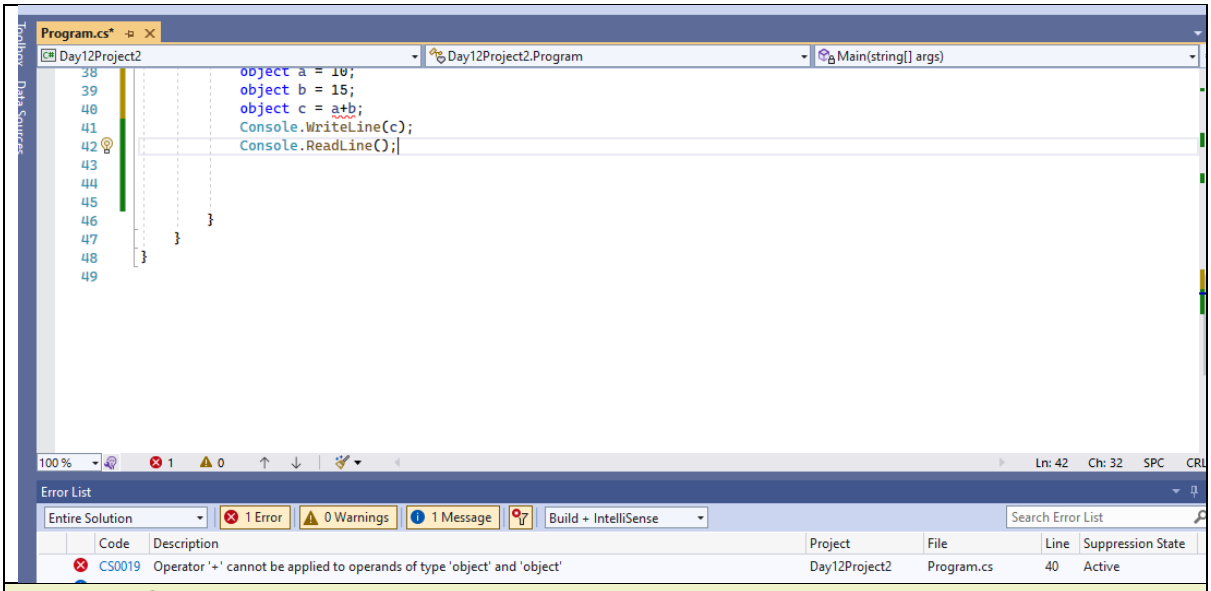
Compilation Error	Runtime Error
1.Errors that occur when we violate the rules of syntax are called Compilation Errors	1.Errors which occur during program execution After successful compilation are called Runtime Errors.
2.This compile error indicates something that must be fixed before the code can be compiled	2.This Runtime errors are hard to find because compiler will not indicate these runtime errors.
3.It includes syntax errors like missing semicolon(;),etc..	3.It includes errors like dividing by zero,etc..

Q7.Write any six compilation errors with small code snippet and add compilation error screenshots.

Example1:

```
using System;
using System.Activities.Expressions;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Day12Project2
{


    internal class Program
    {

        static void Main(String[] args)
        {

            object a = 10;
            object b = 15;
            object c = a+b;
            Console.WriteLine(c);
            Console.ReadLine();



        }
    }
}
```

Exception:

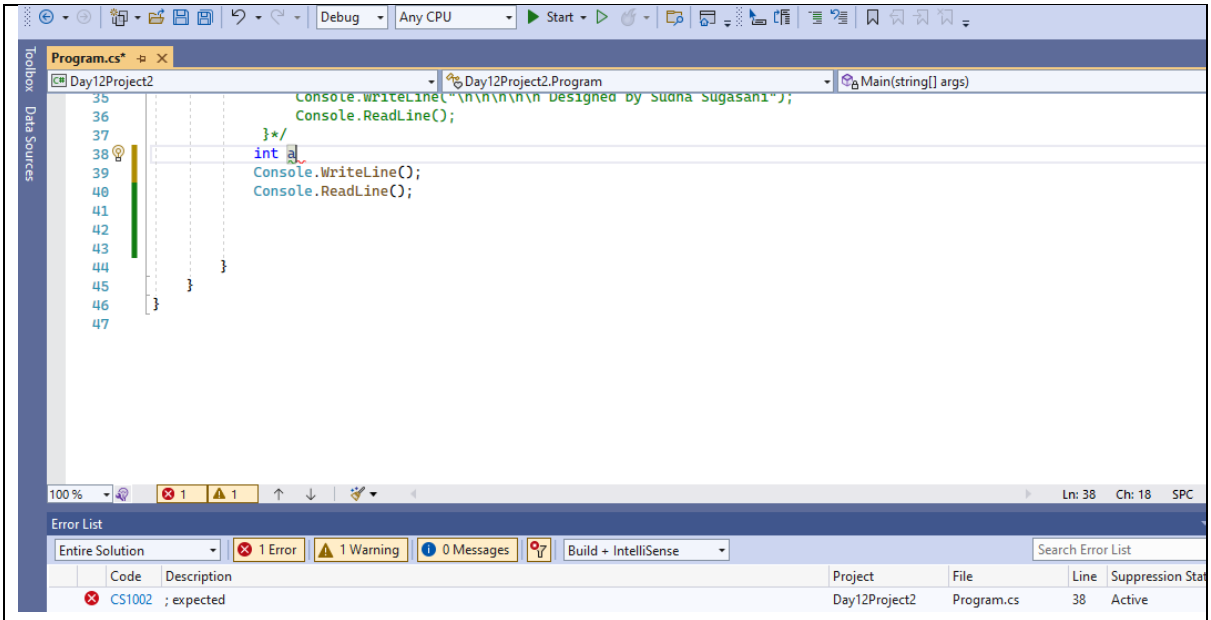


Example 2:

Code:

```
using System;
using System.Activities.Expressions;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Day12Project2
{


    internal class Program
    {


        static void Main(String[] args)
        {

            int a
            Console.WriteLine();
            Console.ReadLine();



        }
    }
}
```

Exception:

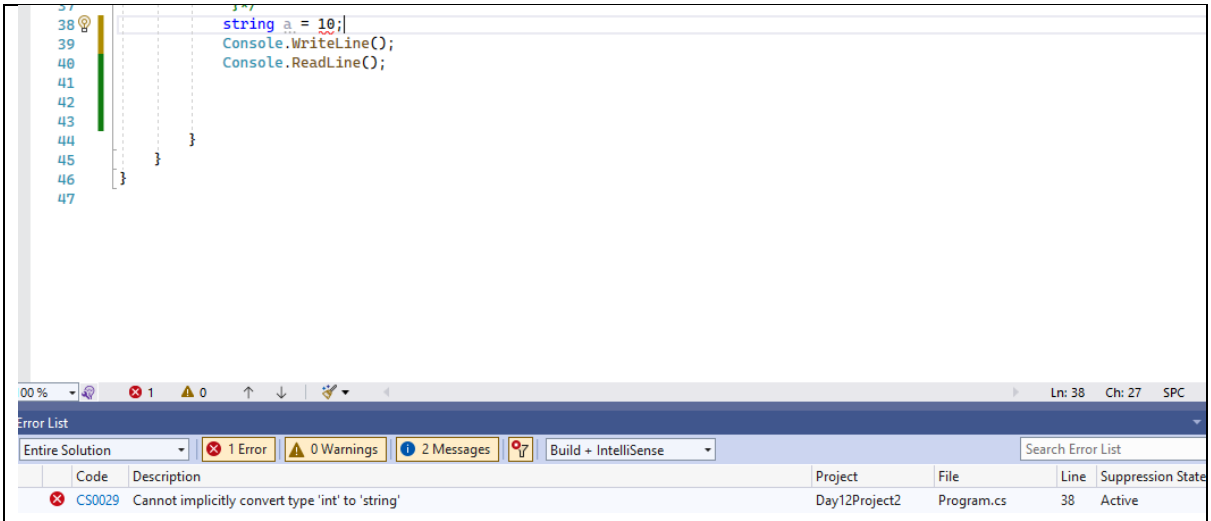


Example3:

Code:

```
using System;
using System.Activities.Expressions;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Day12Project2
{


    internal class Program
    {


        static void Main(String[] args)
        {
            string a = 10;
            Console.WriteLine();
            Console.ReadLine();

        }
    }
}
```

Exception:

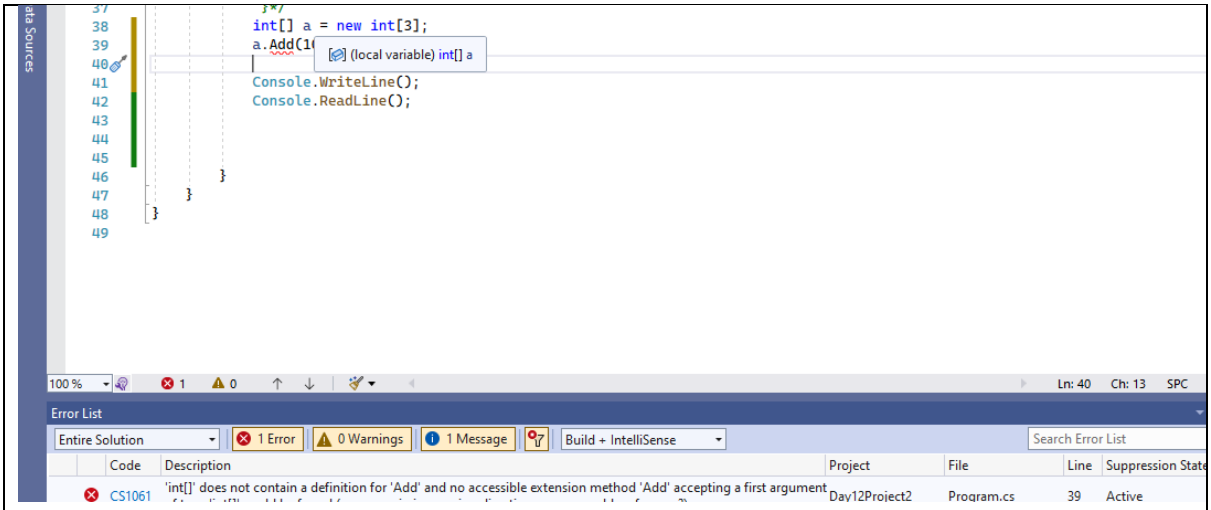


Example3:

Code:

```
using System;
using System.Activities.Expressions;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Day12Project2
{


    internal class Program
    {


        static void Main(String[] args)
        {

            int[] a = new int[3];
            a.Add(10);

            Console.WriteLine();
            Console.ReadLine();



        }
    }
}
```

Exception:

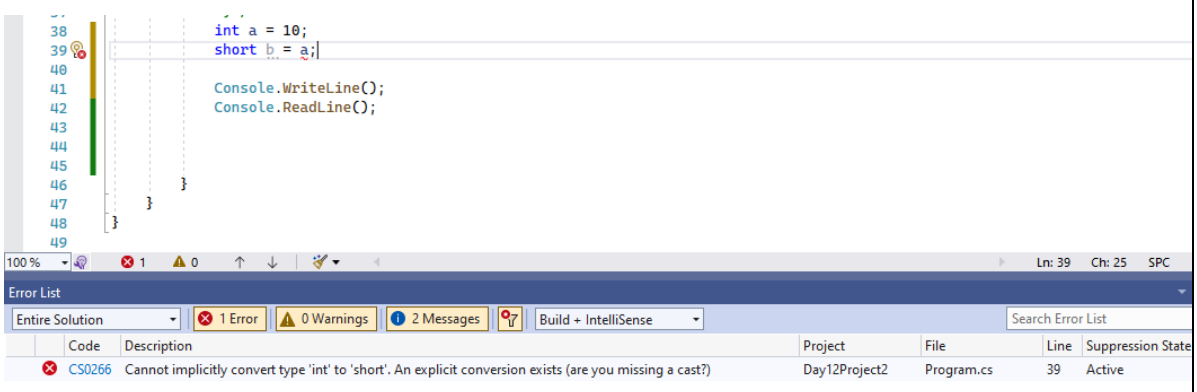


Example4:

Code:

```
using System;
using System.Activities.Expressions;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Day12Project2
{


    internal class Program
    {
            int a = 10;
            short b = a;

            Console.WriteLine();
            Console.ReadLine();



        }
    }
}
```

Exception:

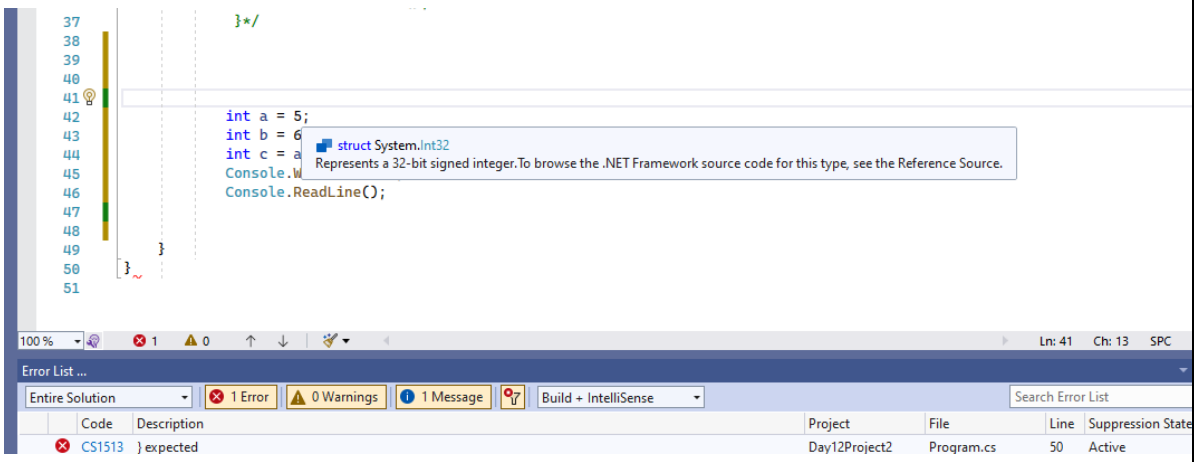


Example5:

Code:

```
using System;
using System.Activities.Expressions;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Day12Project2
{


    internal class Program
    {

            int a = 5;
            int b = 6;
            int c = a + b;
            Console.WriteLine(c);
            Console.ReadLine();


    }
}
```

Exception:



Example 6:

Code:

```
using System;
using System.Activities.Expressions;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Day12Project2
{


    internal class Program
    {
```

```
        static void Main(String[] args)
        {

            Console.WriteLine(c);
            Console.ReadLine();
        }

    }
}
```

Exception: