Error: Any outcome which is not expected

Error :

1. Syntax Error : When we do not follow grammer / syntax of a language

Console.Write(“Hello”);

{}

Easy to find out / Easy to correct

They are detected at compile-time

1. Logical Error : Detected at run time . We always get result but that result is not correct

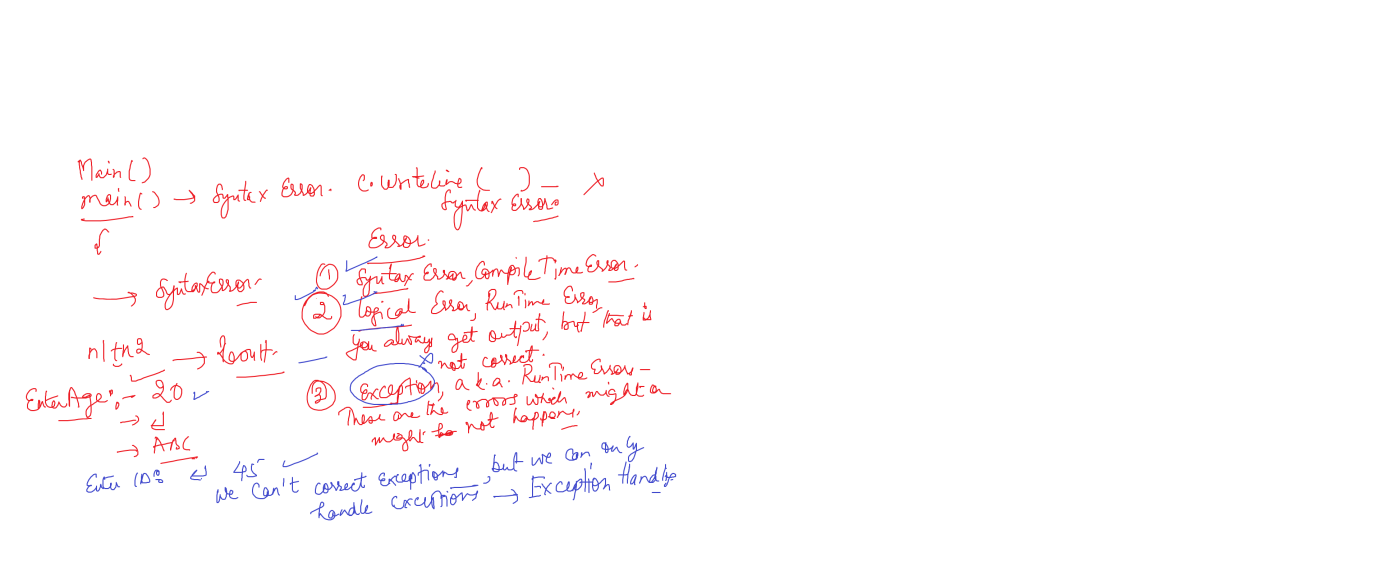
2 + 3 = 5

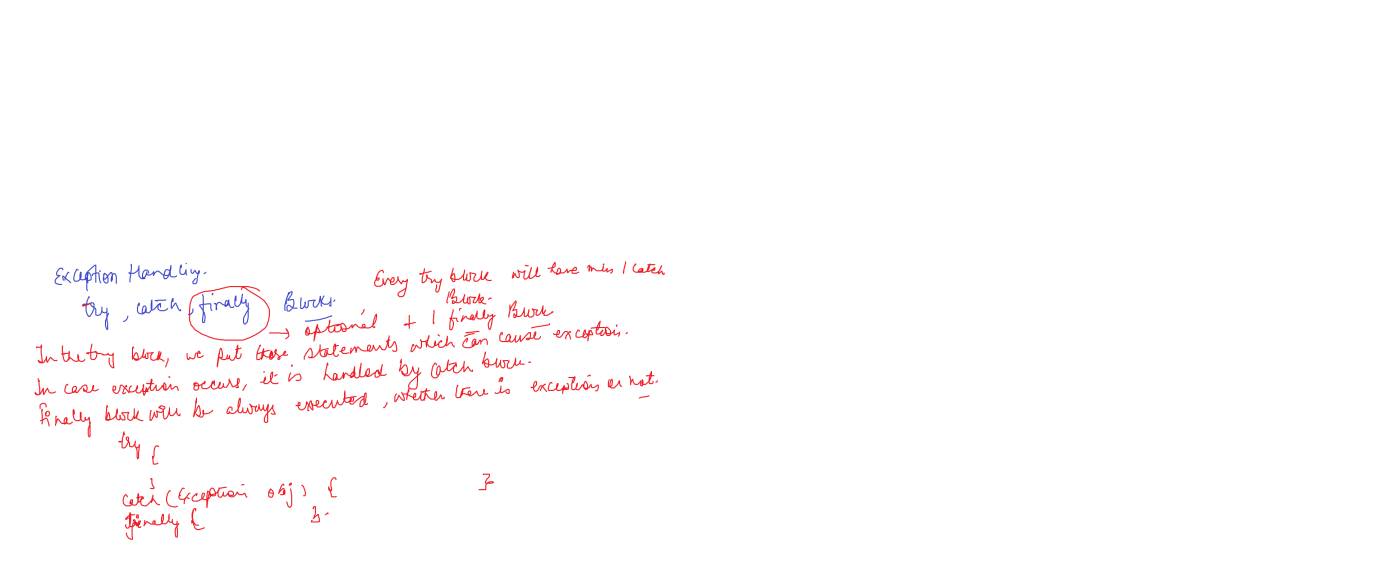
2-3 =-1

Difficult to find out

1. Run Time Error / Exception > Error which may or may not occur

It depends upon what we enter or what we select at run time





We can correct Syntax as well as logical errors

**We cannot correct exceptions**

**We have to handle them**

**Exception Handling**

**By using try & catch block**

**Try block : We put statements that can cause exception**

**Catch block : If some exception occurs, so this block handles it**

**Without EHandling**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp4

{

class Program

{

static void Main(string[] args)

{

int num1, num2;

Console.WriteLine("ENter No 1");

num1 = Convert.ToInt16(Console.ReadLine());

Console.WriteLine("ENter No 2");

num2 = Convert.ToInt16(Console.ReadLine());

Console.WriteLine("Add is " + (num1 + num2));

}

}

}

**With EHandling**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp4

{

class Program

{

static void Main(string[] args)

{

int num1, num2;

try

{

Console.WriteLine("ENter No 1");

num1 = Convert.ToInt16(Console.ReadLine());

Console.WriteLine("ENter No 2");

num2 = Convert.ToInt16(Console.ReadLine());

Console.WriteLine("Add is " + (num1 + num2));

}

catch(Exception ex)

{

Console.WriteLine(ex.Message);

}

}

}

}

Exception is the inbuilt & base class for all exceptions

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp4

{

class Program

{

static void Main(string[] args)

{

int num1, num2;

try

{

Console.WriteLine("ENter No 1");

num1 = Convert.ToInt16(Console.ReadLine());

Console.WriteLine("ENter No 2");

num2 = Convert.ToInt16(Console.ReadLine());

Console.WriteLine(num1/num2);

}

catch(Exception ex)

{

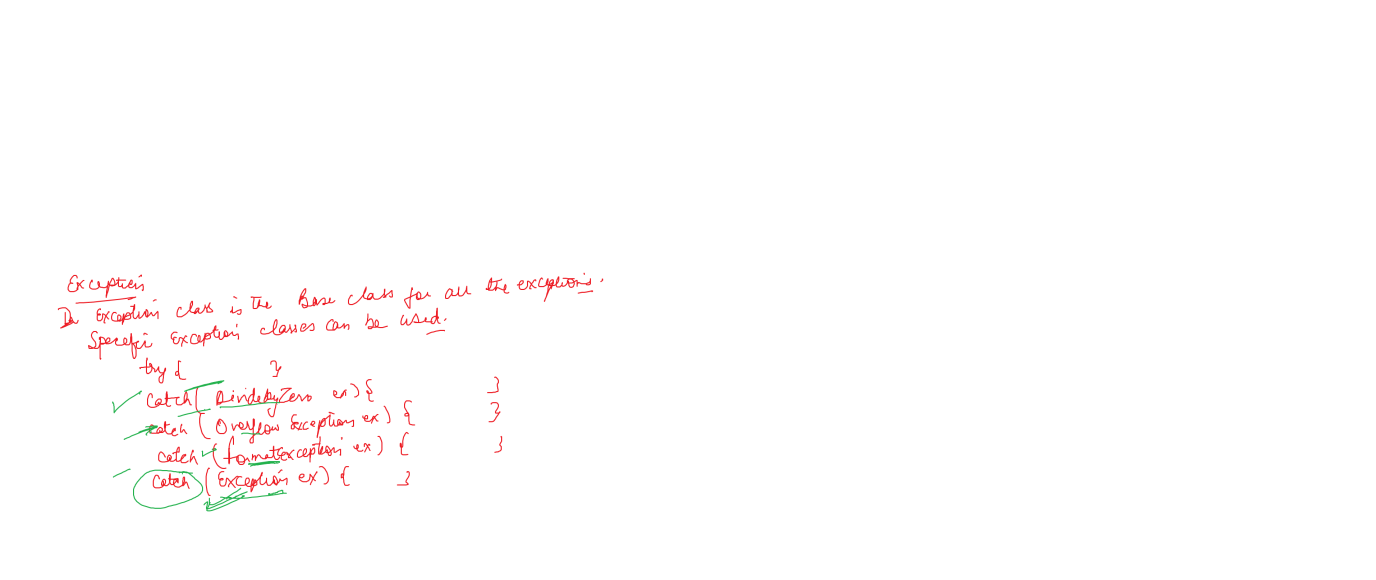
Console.WriteLine(ex.Message);

}

}

}

}

****

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp4

{

class Program

{

static void Main(string[] args)

{

int num1, num2;

try

{

Console.WriteLine("ENter No 1");

num1 = Convert.ToInt16(Console.ReadLine());

Console.WriteLine("ENter No 2");

num2 = Convert.ToInt16(Console.ReadLine());

Console.WriteLine(num1/num2);

}

catch(DivideByZeroException ex)

{

Console.WriteLine(ex.Message);

}

catch(FormatException ex)

{

Console.WriteLine(ex.Message);

}

catch(OverflowException ex)

{

Console.WriteLine(ex.Message);

}

catch(Exception ex)

{

Console.WriteLine(ex.Message);

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp4

{

class Program

{

static void Main(string[] args)

{

int num1, num2;

int[] num = new int[10];

try

{

Console.WriteLine("ENter No 1");

num1 = Convert.ToInt16(Console.ReadLine());

Console.WriteLine("ENter No 2");

num2 = Convert.ToInt16(Console.ReadLine());

Console.WriteLine(num1/num2);

num[12] = 10;

}

catch(DivideByZeroException ex)

{

Console.WriteLine(ex.Message);

}

catch(FormatException ex)

{

Console.WriteLine(ex.Message);

}

catch(OverflowException ex)

{

Console.WriteLine(ex.Message);

}

catch(IndexOutOfRangeException ex)

{

Console.WriteLine(ex.Message);

}

catch(Exception ex)

{

Console.WriteLine(ex.Message);

}

}

}

}

**Custom Exception / User Defined Exception**

**To use it , we create a class , inherit that class from Exception class . and in that class, add a constructor**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp4

{

class Program

{

static void Main(string[] args)

{

int rn=0;

string name =string.Empty;

string uname = string.Empty;

try

{

Console.WriteLine("Enter RollNo");

rn = Convert.ToByte(Console.ReadLine());

Console.WriteLine("Enter Name");

name = Console.ReadLine();

Console.WriteLine("Enter User Name");

uname = Console.ReadLine();

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

finally

{

Console.WriteLine("Roll No is " + rn);

Console.WriteLine("Name is " + name);

Console.WriteLine("User Name is " + uname);

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp4

{

class CustomException : Exception

{

public CustomException(string msg): base(msg)

{

}

}

class Program

{

static void Main(string[] args)

{

int rn=0;

string name =string.Empty;

string uname = string.Empty;

try

{

Console.WriteLine("Enter RollNo");

rn = Convert.ToInt16(Console.ReadLine());

if (rn < 0)

throw new CustomException("RollNo can not be less than zero");

Console.WriteLine("Enter Name");

name = Console.ReadLine();

Console.WriteLine("Enter User Name");

uname = Console.ReadLine();

if (uname.Length < 6)

throw new CustomException("User Name : min 6 characters needed");

Console.WriteLine("Roll No is " + rn);

Console.WriteLine("Name is " + name);

Console.WriteLine("User Name is " + uname);

}

catch (CustomException ex)

{

Console.WriteLine(ex.Message);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

finally

{

}

}

}

}