On Leave : Abhishek(Responds late always) , Aniket

Anonymous Methods : Methods with no name

Add(int x, int y)

{

}

To make anonymous methods , we use Delegates

**Lambda Expression (They are short cut to write anonymous methods)**

**()=> {};**

* Lambda Operator

(input) => {output};

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassDemo5

{

delegate int Del(int x, int y);

//class number

//{

// public int add (int x, int y)

// {

// return x + y;

// }

//}

class AnonymousMethodDemo

{

static void Main()

{

// number number = new number();

//Del del = new Del(number.add);

Del del = delegate (int x, int y)

{

return x + y;

};

Console.WriteLine(del(10,20));

// Lambda Expression

// They are short way to write Anonymous Methods

Del del1 = (x, y) =>

{

return x + y;

};

Console.WriteLine(del1(10,20));

}

}

}

**Namespace : It is a logical collection of related classes**

**When we create project, whatever name we give to our project, we get a namespace with that name**

**Why do we need namespace?**

1. **So that we can create 2 classes with same name, it is not allowed, but we can do it in different namespaces**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace B1

{ class Student

{

}

}

namespace A1

{

class Student

{

}

}

1. **It allows to group logically related classes**

namespace PayRoll

{

class Employee

{

}

class Leave

{

}

class Account

{

}

}

**Exception Handling**

**What is an Error : Something wrong happens**

1. **Syntax Error / Compile Time Error > These are the errors occur when we do not follow the Syntax / grammar of the language**

**Easy to find out. We can easily correct these errors**

1. **Logical Error / Run Time Errors > These are errors which occurs when our logic is not correct.**

**Difficult to find out. In these type of Errors, we always get output, but that output is not correct**

1. **Exception / Run Time Errors > They are the errors which might or might not happen. It depends upon what user enters at run time**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassDemo

{

class ExceptionDemo

{

static void Main()

{

int x;

Console.WriteLine("ENter Number");

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

Console.WriteLine("Value of x is "+ x);

}

}

}

**What errors we can correct > Syntax Error and Logical Error**

**Can we correct Exceptions??? NO**

**We can handle them**

**How ?**

**By using try and catch block**

**Try block will contains statements which can cause error**

**Catch block handles the exception**

**Finally block will be called always (OPTIONAL)**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassDemo

{

class ExceptionDemo

{

static void Main()

{

int x=0;

try

{

Console.WriteLine("ENter Number");

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

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

finally

{

Console.WriteLine("Value of x is " + x);

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassDemo

{

class ExceptionDemo

{

static void Main()

{

int x=0, y=0 , res=0;

try

{

Console.WriteLine("ENter Value of x");

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

Console.WriteLine("Enter Value of y");

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

res = x / y;

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

finally

{

Console.WriteLine("Value of res is " + res);

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassDemo

{

class ExceptionDemo

{

static void Main()

{

int x=0, y=0 , res=0;

int[] num = new int[5];

try

{

Console.WriteLine("ENter Value of x");

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

Console.WriteLine("Enter Value of y");

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

res = x / y;

num[10] = 90;

}

catch(DivideByZeroException e)

{

Console.WriteLine(e.Message);

}

catch(FormatException e)

{

Console.WriteLine(e.Message);

}

catch(OverflowException e)

{

Console.WriteLine(e.Message);

}

catch(IndexOutOfRangeException ex)

{

Console.WriteLine(ex.Message);

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

finally

{

Console.WriteLine("Value of res is " + res);

}

}

}

}

Exceptions cud be Inbuilt Exceptions & User Defined / Custom Exceptions

For ex, no’ s shud be in range 1 - 100

Length of name shud be min 10

**To create custom exceptions**

1. Create a Class , inherit it from Exception Class
2. Add a constructor

var x = Console.ReadLine();

int x=

**User Defined Exception**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassDemo

{

**class CustomException : Exception**

**{**

**public CustomException(string message)**

**: base (message)**

**{**

**}**

**}**

class CustomExceptionDemo

{

static void Main()

{

string name = string.Empty;

int age=0;

try {

Console.WriteLine("Enter Name");

name = Console.ReadLine();

**if (name.Length < 10)**

**throw new CustomException("Min 10 chracters needed");**

Console.WriteLine("Enter Age");

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

**if (age < 20 || age > 50)**

**throw new CustomException("Age shud br from 20 to 50");**

}

catch(FormatException e)

{

Console.WriteLine(e.Message);

}

**catch (CustomException e)**

**{**

**Console.WriteLine(e.Message);**

**}**

catch (Exception e)

{

Console.WriteLine(e.Message);

}

finally

{

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

Console.WriteLine("Age is " + age);

}

} }

}

**LINQ**

**Language Integrated Query Language**

**Sql in RDBMS : Sql is a Query Language in RDBMS**

**LINQ > Language that can be used with any type of Data Source**

**Where Data Source could be Array , Collection , Database , XML File, etc….**

**LINQ can be used with any data source which either implements IQueryable or IEnumerable Interface**

**Collections & Arrays implement IEnumerable interface**

**Data Base implement IQueryable interface**

**Sql >**

**Select \* from tablename**

**LINQ**

**from rangevariable in Collectionname**

**Select rangevariable;**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassDemo

{

class LinqDemo

{

static void Main()

{

int[] num = new []{ 1,2,3,4,5,6,7,89,910};

for(int i=0;i<num.Length;i++)

Console.WriteLine(num[i]);

// LINQ Query to diaplay elements from this Array

var list= from temp in num

select temp;

foreach(var x in list)

Console.WriteLine(x);

// Give me sum of elements of Array

int sum = 0;

for (int i = 0; i < num.Length; i++)

sum += num[i];

// LINQ Query to get sum of an Array

sum = (from temp in num

select temp).Sum();

Console.WriteLine("Sum is " + sum);

double avg = (from temp in num

select temp).Average();

Console.WriteLine("Average is " + avg);

int max = (from temp in num

select temp).Max();

Console.WriteLine("Max no is "+ max);

// Collection

List<int> listofNumbers = new List<int>()

{ 1,2,3,4,5,6,7,8,9,10};

list = (from temp in listofNumbers

select temp);

max = (from temp in listofNumbers

select temp).Max();

}

}

}

**Thread :**

**Single Tasking > One task at a time**

**Multi Tasking > More than one task at a time**