**Session6: Statement & Expression\_\_Exception Handling**

* **Iteration Statement :--(** for, do(execute atleast once without checking condition), while, foreach):-- this is used to iterate over the elements. It allows to execute single statement repetedly. If the condition not satisfy if goes in to **infinite loop.**
* **Nested For Loop :**

**Foreach :** used with list ,array, collection etc.

* **Jump Statement: --** (break, continue, goto, return, yeild):-- jump statement used to break or continue the statement from where it’s used.

**Break:-** it break the program if the condition is satisfy. Used in selection and loop (for). Most used in switch. And control pass to next block.

**Continue:** it continue the statement from condition onward.widely used. It ignored the condition.

**Goto:**- directly execute labelled stmtn. Or labled block. Label is identifier. Single block can be referred with multiple block.

* **Exception Handling Statement: --** (Try, ctach, finally, multiplecatch, User Defined Try Cath):-- exception handling is used to manage the exception thrown by the program. It has different types. Error will be handeled by CLR. If the catch block not found then it will throw the unhandled exception throw that method and CLR.
  + **Try:-** the all logical code written in it but if the error is occurred then it will send the error to catch block and catch block will handle that error. In the program error is occurred then it will break the program.
  + **Catch :-** if the error is thrown by try block it will caught in catch. We can write multiple catch block but when the try block got exception it will terminate the program. And not moved or check the next catch block. Trow keyword is used to catch the error. Or return the error.
  + **Finally:-** finally block will always get executed. And it cleans the resource used by the try block. If the error occurred >>> execute || no error or exception >> Execute.

**Throw:-** throw keyword is used to provide or send custom exception in program. Throw new Exception is used to set the property or exception.

**Custom Exception OR User Defined Exception:-** user can define the exception. (nullRefrence Exception) >>> Datanot Valid OR Data null Exception. **Exception classs abc: exception.**

**Program.cs**

using AllSession;

using Session1.nestedNamespace; // Assembly Refrence OR Namespace OR PAckage

using System;

namespace Session1

{

class Program : S4\_AccessModifiereAndKeyword

{

static void Main(string[] args)

{

//Session 6 Exception Handling

S6\_StatmntAndExecptionHandlingProgram exception = new S6\_StatmntAndExecptionHandlingProgram();

exception.LoopThroughName();

exception.WhileLoopTable();

exception.DoWhileLoopTable();

exception.ContinueBreakGoTo();

exception.TryCatchBlockWithMultipleCatch();

exception.UserException();

}

}

}

**S6\_StatmntAndExecptionHandlingProgram.cs**

using System;

using System.Collections.Generic;

using System.Text;

using System.Text.RegularExpressions;

namespace AllSession

{

class S6\_StatmntAndExecptionHandlingProgram

{

string[] name = { "Amit", "Navyatha", "Eqbal", "Sayali", "Prakash", "Imran", "Abhilasha", "Aniket", "Arnav", "Ganesh" };

//ForLoop + Nested For

public void LoopThroughName()

{

Console.Write("Names Are: ");

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

{

Console.Write(name[i] + " ");

}

Console.WriteLine();

}

//While Loop

public void WhileLoopTable()

{

int number, i = 1;

Console.Write("\*\*\*\*\*\*\*\*You Are in While Loop\*\*\*\*\*\*\*\* \n Enter Number:\t ");

number = Convert.ToInt32(Console.ReadLine());

while (i <= 10)

{

Console.WriteLine($"{number} X {i} = " + number \* i);

i++;

}

}

//DoWhile Loop

public void DoWhileLoopTable()

{

int number, i = 1; // i== counter variable

Console.Write("\*\*\*\*\*\*\*\*You Are in DoWhile Loop\*\*\*\*\*\*\*\* \n Enter Number:\t ");

number = Convert.ToInt32(Console.ReadLine());

do

{

Console.WriteLine($"{number} X {i} = " + number \* i);

i++;

} while (i < 0);

}

//Continue Break Goto

public void ContinueBreakGoTo()

{

for (int i = 0; i <= 20; i++)

{

if (i == 8)

goto terminateProgram;

//continue;

//if (i == 17)

//break;

Console.WriteLine(i);

}

terminateProgram: // GotO

Console.WriteLine("This is End of The Program:: ");

}

//TRY CATCH BLOCK AND MULTIPLE CATCH

public void TryCatchBlockWithMultipleCatch()

{

Console.WriteLine("\*\*\*\*\*\*\*You are in TryCatch MultipleBlock!\*\*\*\*\*");

try

{

string name = null;

Console.WriteLine(name.Length);

int a = 100;

int b = 0;

int c = a / b;

}

catch (DivideByZeroException ex)

{

//Console.WriteLine(ex.Message);

throw;

}

catch (NullReferenceException ex)

{

Console.WriteLine(ex.Message);

}

catch (Exception ex) // Global Exception or Parent Exception of Other Cathc Block

{

Console.WriteLine(ex.Message);

}

}

//User Define Exception

public void UserException()

{

float num = 20;

float num1 = 0;

if (num1 == 0)

{

throw new S6\_\_CustomExceptionOrUserException(S6\_\_CustomExceptionOrUserException.UserException.INVALID\_DATA, "Data is Not Valid");

}

}

}

}

**Custome OR USerDefineException:**

**S6\_\_CustomExceptionOrUserException.cs**

using System;

using System.Collections.Generic;

using System.Text;

namespace AllSession

{

public class S6\_\_CustomExceptionOrUserException : Exception

{

public enum UserException

{

INVALID\_DATA, INVALID\_VALUE

}

private readonly UserException Type;

public S6\_\_CustomExceptionOrUserException(UserException Type, string Message) : base(Message)

{

this.Type = Type;

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*PROGRAM TASK\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Q] Write Program to print natural number from 1 to 100.

Q] Armstrong Number Palindrome Number Perfect Number Using Switch While and Do While.

Q] print perfect and palindrome number between 1 to 100 and.