* **C#.NET Fundamentals:**

- .NET is a multi-lingual programming language.

- Such as C#.Net, VB.NET, F#.NET, J#.NET and so on.

- Helps in creating:

1. Console Application

2. Windows Application

3. Web Application

4. Mobile Application

5. Client-Server Based Application and so on.

--------------------------------------------------------

* **.NET Framework vs .NET Core**

Common Language Runtime (.Net Framework)

Core - Common Language Runtime (.Net Core)

Application developed with .NET Framework only runs on Windows Platform.

Application developed with .NET Core runs on Windows/Linux/Mac or other OS.

.NET Framework Application deployed was paid with Enterprise Edition of VS.

.NET Core is open source.

.NET Framework has restricted support to Client-Side Frameworks.

.NET Core is full supported with Client-Side Frameworks such as Angular, React.

.NET Framework allows you to develop applications using editor i.e., VS.

.NET Core also allows you to develop applications using Command Line Interface (CLI).

--------------------------------------------------------

Namespace -> Class -> Attributes and Methods

Namespace: A collection of classes, interfaces, events and so on.

Class: A class is a collection of data members and member functions.

Data Member: A variable declared at class level.

Member Function: A method declared at class level.

--------------------------------------------------------

System - Namespace and Console - Class

Below are the static methods of Console class for Read/Write Operations:

Output Methods: Write and WriteLine

Input Methods: Read, ReadLine and ReadKey

--------------------------------------------------------

* **Type Casting:**

A mechanism to convert one data type to another data type.

Type Casting is possible if both data types are compatible to each other.

Otherwise, you will get an InvalidCastException.

**1. Implicit Casting**

Implicit Casting is being done by the compiler automatically and there is no data loss chances. This casting converts a smaller data type into the larger data types. This is a safe type casting or type conversion.

**2. Explicit Casting**

Explicit Casting is being done by a developer. It includes conversion of larger data type into smaller data type. This is a un-safe type conversion.

--------------------------------------------------------

* **Conditional Constructs**

- Basic If Statements

- If..Else Statements

- If..Else If..Else Statements

- Nested If..Else Statements

- switch Case Statements

* **Looping Constructs**

- For Loop

- While Loop

- Do..While Loop

- Foreach Loop / Enhanced Loop

* **Non-Conditional Constructs**

- Goto Statements

- Break Statements

- Continue Statements

--------------------------------------------

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CSharp\_Basics

{

class ConditionalContructExample

{

static void Main(string[] args)

{

int temperature = 19;

if (temperature < 20)

{

Console.WriteLine("Its very cold outside");

}

if (temperature < 30)

{

Console.WriteLine("Its moderate cold outside");

}

if (temperature < 45)

{

Console.WriteLine("Its very hot outside");

}

Console.ReadLine();

}

}

}

--------------------------------------------------------

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CSharp\_Basics

{

class ConditionalContructExample

{

static void Main(string[] args)

{

int sellingPrice = 200, costPrice = 120;

if (sellingPrice > costPrice)

{

Console.WriteLine("Its Profit");

}

if (costPrice > sellingPrice)

{

Console.WriteLine("Its Loss");

}

if (costPrice == sellingPrice)

{

Console.WriteLine("No Profit No Loss");

}

Console.ReadLine();

}

}

}

--------------------------------------------------------

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CSharp\_Basics

{

class ConditionalContructExample

{

static void Main(string[] args)

{

int sellingPrice = 200, costPrice = 120;

if (sellingPrice > costPrice)

{

Console.WriteLine("Its Profit");

}

if (costPrice > sellingPrice)

{

Console.WriteLine("Its Loss");

}

else

{

Console.WriteLine("No Profit No Loss");

}

Console.ReadLine();

}

}

}

--------------------------------------------------------

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CSharp\_Basics

{

class ConditionalContructExample

{

static void Main(string[] args)

{

int sellingPrice = 200, costPrice = 120;

if (sellingPrice > costPrice)

{

Console.WriteLine("Its Profit");

}

else if (costPrice > sellingPrice)

{

Console.WriteLine("Its Loss");

}

else

{

Console.WriteLine("No Profit No Loss");

}

Console.ReadLine();

}

}

}

--------------------------------------------------------

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CSharp\_Basics

{

class ConditionalContructExample

{

static void Main(string[] args)

{

/\* A User is only allowed to make a purchase when

\* he is logged In,

\* email is verified and

\* cardInfo is valid.\*/

bool isLoggedIn = true;

bool isEmailVerified = false;

bool cardInfo = true;

if (isLoggedIn)

{

Console.WriteLine("Logged In Success.");

if (isEmailVerified)

{

Console.WriteLine("Email is Verified.");

if (cardInfo)

{

Console.WriteLine("You can make a purchase");

}

else

{

Console.WriteLine("You are not allowed make a purchase.");

}

}

else

{

Console.WriteLine("You are not allowed make a purchase.");

}

}

else

{

Console.WriteLine("You are not allowed make a purchase.");

}

Console.ReadLine();

}

}

}

--------------------------------------------------------

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CSharp\_Basics

{

class ConditionalContructExample

{

static void Main(string[] args)

{

/\* A User is only allowed to make a purchase when

\* he is logged In,

\* email is verified and

\* cardInfo is valid.\*/

bool isLoggedIn = true;

bool isEmailVerified = true;

bool cardInfo = true;

if (isLoggedIn && isEmailVerified && cardInfo)

{

Console.WriteLine("You are allowed make a purchase.");

}

else

{

Console.WriteLine("You are not allowed make a purchase.");

}

Console.ReadLine();

}

}

}

--------------------------------------------------------

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CSharp\_Basics

{

class ConditionalContructExample

{

static void Main(string[] args)

{

/\* Allow a user to access a course if he is

\* logged in from Email

\* logged in from Google

\* logged in from FaceBook \*/

bool email = true;

bool facebook = false;

bool google = true;

if (email || facebook || google)

{

Console.WriteLine("Logged In Successfully.");

}

else

{

Console.WriteLine("Login Failed.");

}

Console.ReadLine();

}

}

}

--------------------------------------------------------

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CSharp\_Basics

{

class ConditionalContructExample

{

static void Main(string[] args)

{

/\* Show user a signout button if he is authenticated.

\* Otherwise show him option to Login/Signup. \*/

/\*First Way: \*/

bool authenticated = true;

if (authenticated)

Console.WriteLine("Show SignUp Button");

else

Console.WriteLine("Show Login Button");

/\*Second Way: \*/

string result = authenticated ? "SignUp Button" : "Login Button";

Console.WriteLine(result);

Console.ReadLine();

}

}

}

--------------------------------------------------------

// Never Executes

while (false)

{

Console.WriteLine("Never Executes");

}

// Executes Forever

while (true)

{

Console.WriteLine("Never Stops");

}

--------------------------------------------------------

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CSharp\_Basics

{

class LoopingConstructExample

{

static void Main(string[] args)

{

bool iNeedMoreCandles = true;

int candleCount = 0;

while (iNeedMoreCandles)

{

Console.Write("Do you need one more candle ? (Yes/No)");

string answer = Console.ReadLine();

if (answer == "Yes")

candleCount++;

else

iNeedMoreCandles = false;

}

Console.WriteLine("Candles Needed : {0}", candleCount);

Console.ReadLine();

}

}

}

------------------------------------------------------------

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CSharp\_Basics

{

class LoopingConstructExample

{

static void Main(string[] args)

{

/\*bool iNeedMoreCandles = true;

int candleCount = 0;

while (iNeedMoreCandles)

{

Console.Write("Do you need one more candle ? (Yes/No)");

string answer = Console.ReadLine();

if (answer == "Yes")

candleCount++;

else

iNeedMoreCandles = false;

}

Console.WriteLine("Candles Needed : {0}", candleCount);\*/

/\*int candleCount = 0;

for (string input = "Yes"; input == "Yes"; input = Console.ReadLine())

{

candleCount++;

Console.Write("Do you need one more candle ? (Yes/No)");

}

Console.WriteLine("Candles Needed : {0}", candleCount);\*/

/\*for (int i = 0; i < 5; i++)

{

Console.WriteLine("Print Me");

}\*/

/\*int j = 0;

while (j < 5)

{

Console.WriteLine("Print Me");

j++;

}\*/

/\*string answer;

int candleCount = 0;

do

{

candleCount++;

Console.Write("Do you need one more candle ? (Yes/No)");

answer = Console.ReadLine();

}

while (answer == "Yes" || answer == "Y");

Console.WriteLine("More Candles Needed : {0}", candleCount);\*/

Console.ReadLine();

}

}

}

-------------------------------------------