Basics of C#

1) Explore the basic concepts.

using System;

namespace CSharpPractice

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Enter your name ");

string name = Console.ReadLine();

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

Console.WriteLine(" Enter your Age ");

int age = Convert.ToInt32(Console.ReadLine());

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

// Data types and varibles

int num = 5;

double doubleNum = 5.99;

char letter = 'D';

bool boolValue = true;

Console.WriteLine(num);

Console.WriteLine(doubleNum);

Console.WriteLine(letter);

Console.WriteLine(boolValue);

// Working with strings

string greeting = "Hello , How are you";

Console.WriteLine(greeting.ToUpper());

Console.WriteLine(greeting.ToLower());

Console.WriteLine(greeting.Length);

string firstName = "Sreehari ";

string lastName = "J";

string fullName = string.Concat(firstName, lastName);

Console.WriteLine("Concatination ",fullName);

Console.WriteLine(" First letter of name " + firstName[0]);

Console.WriteLine("Get the index " + firstName.IndexOf('h'));

}

}

}

2) Conditional statements and loops

// If else statments

namespace Conditional\_Statments

{

internal class Program

{

static void Main(string[] args)

{

Console.WriteLine("Enter a number :");

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

if (number < 100)

{

Console.WriteLine($"{number} is less than 100");

}

else if (number > 100)

{

Console.WriteLine($"{number} is greater than 100");

}

else {

Console.WriteLine("Enter a valid number ");

}

}

}

}

Switch case

int day = 4;

switch (day)

{

case 1:

Console.WriteLine("Monday");

break;

case 2:

Console.WriteLine("Tuesday");

break;

case 3:

Console.WriteLine("Wednesday");

break;

case 4:

Console.WriteLine("Thursday");

break;

case 5:

Console.WriteLine("Friday");

break;

case 6:

Console.WriteLine("Saturday");

break;

case 7:

Console.WriteLine("Sunday");

break;

}

For and while loop

// while loop

int i = 0;

while (i < 5)

{

Console.WriteLine(i);

i++;

}

// for loop

for (int i = 0; i < 5; i++)

{

Console.WriteLine(i);

}

3) Array

string [] studentNames = { "vishnu" , "sree", "visal" };

// print details using for loop

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

{

Console.WriteLine(studentNames[i]);

}

// sort array

Array.Sort(studentNames);

// print details using for loop

Console.WriteLine("after sorted");

foreach (string i in studentNames)

{

Console.WriteLine(i);

}

Console.WriteLine("maximum" + studentNames.Max());

//using System.Linq

int[] myNumbers = { 5, 1, 8, 9 };

Console.WriteLine(myNumbers.Max());

Console.WriteLine(myNumbers.Min());

Console.WriteLine(myNumbers.Sum());

3) Class And objects

Create an object

namespace ClassAndObjects

{

class Program

{

string color = "red";

void SetData()

{

Console.WriteLine("success");

}

static void Main(string[] args)

{

Program myObj = new Program();

Console.WriteLine(myObj.color);

myObj.SetData();

}

}

}

Constructors

namespace ClassAndObjects

{

class Program

{

public Program(string color) {

Console.WriteLine(color);

}

void SetData()

{

Console.WriteLine("success");

}

static void Main(string[] args)

{

Program myObj = new Program("red");

myObj.SetData();

}

}

}

Inheritance

using System;

namespace OOPS

{

class ParentClass

{

public void ParentMethod(string name)

{

Console.WriteLine(name);

}

}

class ChildClass : ParentClass

{

public ChildClass()

{

Console.WriteLine("child class executed sucessfully");

}

}

class InheritanceProgram

{

static void Main(string[] args)

{

ChildClass obj = new ChildClass();

obj.ParentMethod("Parent class executed");

}

}

}

Properties in c#

namespace OOPS

{

class AddProperty

{

private string name;

public string Name

{

get { return name; }

set { name = value; }

}

}

class PropertiesProgram

{

static void Main(string[] args)

{

AddProperty obj = new AddProperty();

obj.Name = "sree";

Console.WriteLine(obj.Name);

}

}

}

Overriding

namespace OOPS

{

class Payment

{

public virtual void MakePayment()

{

Console.WriteLine(" Choose a payment gate way ");

}

}

class RazorpayPayment : Payment

{

public override void MakePayment()

{

Console.WriteLine(" Choose Razor payment ");

}

}

class PaytmPayment : Payment

{

public override void MakePayment()

{

Console.WriteLine(" Choose Paytm payment ");

}

}

internal class OverridingProgram

{

static void Main (string[] args)

{

Payment doPayment = new Payment();

Payment doRazorpay = new RazorpayPayment();

Payment doPaytmPayment = new PaytmPayment();

doPayment.MakePayment();

doRazorpay.MakePayment();

doPaytmPayment.MakePayment();

}

}

}

Abstract class

using System;

namespace OOPS

{

abstract class Animal

{

public abstract void animalSound();

}

class Dog : Animal

{

public override void animalSound()

{

Console.WriteLine("The dog makes a sound");

}

}

class AbstractclassProgram

{

static void Main(string[] args)

{

Dog dogObject = new Dog();

dogObject.animalSound();

}

}

}

Exception handling

namespace OOPS

{

class ExceptionProgram

{

static void Main(string[] args)

{

try

{

int[] myNumbers = { 1, 2, 3 };

Console.WriteLine(myNumbers[10]);

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

finally

{

Console.WriteLine("Always executed");

}

}

}

}