**Oops demo 1:**

namespace oopsdemo1\_pre

{

class Accenture

{

public void show()

{

Console.WriteLine("welcome to accenture");

display();

}

void display()

{

Console.WriteLine("display is private method");

}

}

internal class Program

{

static void Main(string[] args)

{

Accenture obj=new Accenture();

obj.show();

Console.ReadKey();

}

}

}

**Oops demo2 :**

//this keyword refer to instance

namespace oopsdemo2\_pre

{

class Employee

{

int empno;

string empname;

double salary;

public void setData(int empno, string empname, double salary)

{

this.empno = empno;

this.empname = empname;

this.salary = salary;

}

public void show()

{

Console.WriteLine("Employee Number:" + empno);

Console.WriteLine("Employee Name:" + empname);

Console.WriteLine("Employee Salary:" + salary);

}

}

internal class Program

{

static void Main(string[] args)

{

Employee e = new Employee();

e.setData(101, "ram", 56000);

e.show();

Console.ReadKey();

}

}

}

**Constructor demo:**

//this keyword refer to instance

namespace oopsdemo2\_pre

{

class Employee

{

int empno;

string empname;

double salary;

public Employee(int empno, string empname, double salary) //parameterized constructor

{

Console.WriteLine("\nparameterized constructor");

this.empno = empno;

this.empname = empname;

this.salary = salary;

}

public Employee()

{

Console.WriteLine("\nDefault constructor");

}

public void show()

{

Console.WriteLine("Employee Number:" + empno);

Console.WriteLine("Employee Name:" + empname);

Console.WriteLine("Employee Salary:" + salary);

}

}

internal class Program

{

static void Main(string[] args)

{

Employee e = new Employee(101, "ram", 56000);

e.show();

Employee e2 = new Employee();

Console.ReadKey();

}

}

}

**Method overloading demo**

namespace overloaddemo\_pre

{

class Accenture

{

public void addition(int a, int b)

{

int res=a+b;

Console.WriteLine("Addition:"+res);

}

public void addition(int a, int b ,int c)//method overloading

{

int res = a + b;

Console.WriteLine("Addition:" + res);

}

public void addition(double a, int b)

{

double res = a + b;

Console.WriteLine("Addition:" + res);

}

public void addition(int a, double b)

{

double res = a + b;

Console.WriteLine("Addition:" + res);

}

public void addition(double a, double b)

{

double res = a + b;

Console.WriteLine("Addition:" + res);

}

}

internal class Program

{

static void Main(string[] args)

{

Accenture obj= new Accenture();

obj.addition(10, 20.1);

Console.ReadKey();

}

}

}

**Inheritance demo**

namespace inheritdemo\_pre

{

class Animal

{

public void show()

{

Console.WriteLine("animals are breathing");

}

}

class Dog:Animal

{

public void display()

{

Console.WriteLine("dogs are barking");

}

}

internal class Program

{

static void Main(string[] args)

{

//Dog d = new Dog();

//d.show();

//d.display();

Animal a = new Dog();

a.show();

Console.ReadKey();

}

}

}

**Inheritance Demo 2:**

//before dervied class constructor ,base class constructor is called(bydefault base class default constructor)

namespace inheritdemo2\_pre

{

class Employee

{

protected int empno;

protected string empname;

protected double salary;

public Employee(int eno,string enm,double sal)

{

Console.WriteLine("\n Employee parameterized constructor called");

empno = eno;

empname = enm;

salary = sal;

}

public Employee()

{

Console.WriteLine("\nEmployee Default constructor called");

}

public void show()

{

Console.WriteLine("Employee Number:" + empno);

Console.WriteLine("Employee Name:" + empname);

Console.WriteLine("Employee Salary:" + salary);

}

}

class Manager:Employee

{

double bonus;

public Manager(int eno, string enm, double sal,double b):base(eno,enm,sal)

{

Console.WriteLine("\nManager parameterized constructor called");

empno = eno;

empname = enm;

salary = sal;

bonus = b;

}

public void display()

{

show();

Console.WriteLine("Manager Bonus:"+bonus);

}

}

internal class Program

{

static void Main(string[] args)

{

Manager m = new Manager(101,"pooja",67000,4000);

m.display();

Console.ReadKey();

}

}

}

Overriding demo:

//base class and derived class has same method name with same parameters is called as method overriding

namespace overridedemo\_pre

{

class Animal

{

virtual public void show()

{

Console.WriteLine("animals are breathing");

}

}

class Dog:Animal

{

override public void show()//method overriding

{

base.show();

Console.WriteLine("Dogs are barking");

}

}

internal class Program

{

static void Main(string[] args)

{

Dog d=new Dog();

d.show();

//Animal a = new Dog();

//a.show();

Console.ReadKey();

}

}

}

**Abstract class Demo:**

namespace abstractdemo\_pre

{

abstract class Shape

{

abstract public void area();

abstract public void show();

public void display()

{

Console.WriteLine("display is non-abstract method");

}

}

class Circle : Shape

{

public override void area()

{

double r, a;

const double pi = 3.14;

Console.WriteLine("Enter Radius:");

r=double.Parse(Console.ReadLine());

a = pi \* r \* r;

Console.WriteLine("Area of circle:" + a);

}

public override void show() { }

}

internal class Program

{

static void Main(string[] args)

{

// Circle c=new Circle();

//c.area();

Shape s=new Circle();

s.area();

Console.ReadKey();

}

}

}

Interface demo:

namespace interfacedemo\_pre

{

interface Shape

{

void area(); //abstract method

void show();

public void display() { }

}

class Circle : Shape

{

public void area()

{

double a, r;

const double pi = 3.14;

Console.WriteLine("Enter Radius:");

r=double.Parse(Console.ReadLine());

a = pi \* r \* r;

Console.WriteLine("Area of circle:" + a);

}

public void show() { }

}

internal class Program

{

static void Main(string[] args)

{

Circle c=new Circle();

c.area();

Shape s=new Circle();

s.area();

Console.ReadKey();

}

}

}

Interface demo 2:

namespace interfacedemo2\_pre

{

interface first

{

void show();

}

interface second

{

void display();

}

class third : first, second

{

public void show()

{

Console.WriteLine("first show");

}

public void display()

{

Console.WriteLine("second display");

}

}

internal class Program

{

static void Main(string[] args)

{

//third t=new third();

//t.show();

//t.display();

first f=new third();

f.show();

second s=new third();

s.display();

Console.ReadKey();

}

}

}

**Interface demo3:**

namespace interfacedemo3\_pre

{

interface first

{

void show();

}

interface second

{

void show();

}

class third : first, second

{

void first.show()

{

Console.WriteLine("first show");

}

void second.show()

{

Console.WriteLine("second show");

}

}

internal class Program

{

static void Main(string[] args)

{

first f = new third();

f.show();

second s = new third();

s.show();

Console.ReadKey();

}

}

}

Interface demo 4

//interface inheritance

namespace interfacedemo4\_pre

{

interface first

{

void show();

}

interface second:first //inheritance

{

void display();

}

class third : second //implements

{

public void display()

{

Console.WriteLine("second display");

}

public void show()

{

Console.WriteLine("first show");

}

}

internal class Program

{

static void Main(string[] args)

{

third t=new third();

t.show();

t.display();

Console.ReadKey();

}

}

}

**Interface demo 5:**

namespace interfacedemo5\_pre

{

class first

{

}

interface second

{

}

interface third

{

}

class four:first,second,third

{

}

internal class Program

{

static void Main(string[] args)

{

Console.WriteLine("Hello, World!");

}

}

}

**Static method demo:**

using System.ComponentModel.DataAnnotations;

namespace staticdemo\_pre

{

class Accenture

{

static public void show()

{

Console.WriteLine("welcome to accenture");

}

public void display()

{

Console.WriteLine("display is non-static/instance method");

}

}

internal class Program

{

static void Main(string[] args)

{

Accenture.show();

Console.ReadKey();

}

}

}

Static constructor demo:

namespace staticdemo2\_pre

{

class Accenture

{

public Accenture()

{

Console.WriteLine("instance constructor called");

}

static Accenture()

{

Console.WriteLine("static constructor called");

}

static public void show()

{

Console.WriteLine("show is static method");

}

}

internal class Program

{

static void Main(string[] args)

{

Accenture obj1=new Accenture();

Accenture obj2 = new Accenture();

Accenture obj3 = new Accenture();

Accenture obj4 = new Accenture();

Accenture.show();

Console.ReadKey();

}

}

}

**Property demo 1**

namespace propertydemo\_pre

{

class Accenture

{

int num;

public int show

{

set

{

num = value;

}

get

{

return num;

}

}

}

internal class Program

{

static void Main(string[] args)

{

Accenture obj=new Accenture();

obj.show = 100; //set

Console.WriteLine(obj.show); //get

Console.ReadKey();

}

}

}

Static property demo

namespace propertydemo2\_pre

{

class Accenture

{

static int num;

static public int show

{

set

{

num = value;

}

get

{

return num;

}

}

}

internal class Program

{

static void Main(string[] args)

{

Accenture.show = 100; //set

Console.WriteLine(Accenture.show); //get

Console.ReadKey();

}

}

}

**Inherits the property**

namespace propertydemo3\_pre

{

class Accenture

{

int num;

public int show

{

set

{

num = value;

}

get

{

return num;

}

}

}

class Employee : Accenture

{ }

internal class Program

{

static void Main(string[] args)

{

Employee obj = new Employee();

obj.show = 100; //set

Console.WriteLine(obj.show); //get

Console.ReadKey();

}

}

}

**Abstract property demo**

//abstract property

namespace propertydemo4\_pre

{

abstract class Accenture

{

public abstract int show { set; get; }

}

class Employee : Accenture

{

int num;

public override int show

{

set

{

num = value;

}

get

{

return num;

}

}

}

internal class Program

{

static void Main(string[] args)

{

Employee obj = new Employee();

obj.show = 100; //set

Console.WriteLine(obj.show); //get

Console.ReadKey();

}

}

}

**Read-only Property**

//read-only property

namespace propertydemo5\_pre

{

class Accenture

{

int num;

public Accenture(int n)

{

num = n;

}

public int show

{

get

{

return num;

}

}

}

internal class Program

{

static void Main(string[] args)

{

Accenture obj = new Accenture(200);

Console.WriteLine(obj.show); //get

Console.ReadKey();

}

}

}