**Initializing an object:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Inheritance

{

using System;

public class BankAccount

{

private string accountNumber;

private decimal balance;

public string AccountNumber

{

get { return accountNumber; }

set { accountNumber = value; }

}

public decimal Balance

{

get { return balance; }

private set { balance = value; }

}

public void Deposit(decimal amount)

{

balance += amount;

}

public void Withdraw(decimal amount)

{

if (amount <= balance)

balance -= amount;

}

}

public class Program

{

public static void Main()

{

BankAccount myAccount = new BankAccount();

myAccount.AccountNumber = "3083";

myAccount.Deposit(1000);

myAccount.Withdraw(600);

Console.WriteLine("Account Number: " + myAccount.AccountNumber);

Console.WriteLine("Balance: $" + myAccount.Balance);

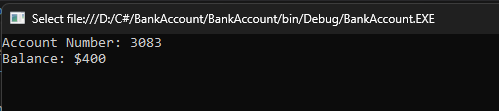
Console.ReadKey();

}

}

}

**Output:**

****

**Inheritance:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Inheritance

{

class Vehicle1

{

public string name;

public void engine()

{

Console.WriteLine("This is the engine of the vehicle");

}

}

// derived class of Vehicle

class Car : Vehicle1

{

public void display()

{

Console.WriteLine("This car is" + name);

}

}

class Vehicle

{

static void Main(string[] args)

{

// object of derived class

Car c = new Car();

c.name = " Toyota ";

// method of base class

c.engine();

// method from own class

c.display();

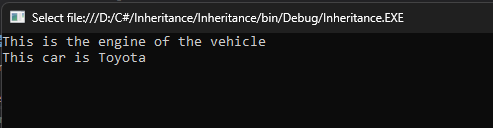
Console.ReadKey();

}

}

}

**Output:**

****

**Default Constructor:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Employee

{

public class Program

{

public Program()

{

Console.WriteLine("Default Constructor Invoked");

}

public static void Main(string[] args)

{

Program e1 = new Program();

Program e2 = new Program();

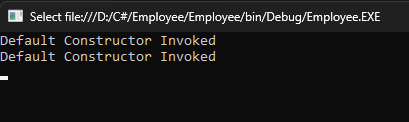
Console.ReadKey();

}

}

}

**Output:**

****

**Parameterized Constructor**

public class Employee

{

public int id;

public String name;

public float salary;

public Employee(int i, String n,float s)

{

id = i;

name = n;

salary = s;

}

 public void display()

        {

            Console.WriteLine(id + " " + name+" "+salary);

        }

   }

class TestEmployee{

public static void Main(string[] args)

{

Employee e1 = new Employee(101, "Sonoo", 890000f);

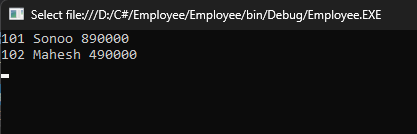
Employee e2 = new Employee(102, "Mahesh", 490000f);

e1.display();

e2.display();

}

**Output:**

****

**Abstract classes**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Employee

{

//abstract class

abstract class Shape

{

//abstract methods

public abstract double calculateArea();

public abstract void displayDetails(double area);

}

//Rectangle class inheriting Shape class

class Rectangle : Shape

{

//private data members

private double length;

private double breadth;

public Rectangle(double length, double breadth)

{

this.length = length;

this.breadth = breadth;

}

//overriding abstract methods of Shape class using 'override’ keyword

public override double calculateArea()

{

return (length \* breadth);

}

public override void displayDetails(double area)

{

Console.Write("Length of rectangle: " + length);

Console.Write("\nBreadth of rectangle: " + breadth);

Console.Write("\nArea of rectangle: " + area);

}

}

//Square class inheriting Shape class

class Square : Shape

{

//private data members

private double side;

public Square(double side)

{

this.side = side;

}

//overriding abstract methods of Shape class using 'override' keyword

public override double calculateArea()

{

return (side \* side);

}

public override void displayDetails(double area)

{

Console.Write("Length of a side of square: " + side);

Console.Write("\nArea of square: " + area);

}

}

public class AbstractionDemo

{

public static void Main(string[] args)

{

double area;

//creating reference of Shape class using Rectangle class

Shape shapeRec = new Rectangle(5, 6);

area = shapeRec.calculateArea();

shapeRec.displayDetails(area);

Console.WriteLine("\n");

//creating reference of Shape class using Square class

Shape shapeSquare = new Square(4);

area = shapeSquare.calculateArea();

shapeSquare.displayDetails(area);

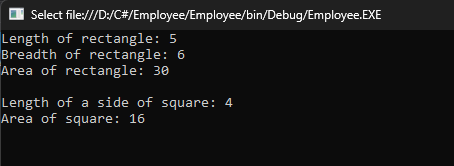
Console.ReadKey();

}

}

}

**Output:**

****

**Encapsulation**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Employee

{

public class Person

{

private string name;

private int age;

public string Name

{

get { return name; }

set { name = value; }

}

public int Age

{

get { return age; }

set { age = value; }

}

}

class Program

{

static void Main(string[] args)

{

Person person = new Person();

person.Name = "John";

person.Age = 30;

Console.WriteLine("Name: " + person.Name);

Console.WriteLine("Age: " + person.Age);

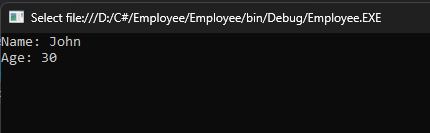
Console.ReadKey();

}

}

}

**Output:**

****

**Extension Method in C#:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Extension\_Program

{

class Geek

{

// Method 1

public void M1()

{

Console.WriteLine("Method Name: M1");

}

// Method 2

public void M2()

{

Console.WriteLine("Method Name: M2");

}

// Method 3

public void M3()

{

Console.WriteLine("Method Name: M3");

}

}

static class NewMethodClass

{

// Method 4

public static void M4(this Geek g)

{

Console.WriteLine("Method Name: M4");

}

// Method 5

public static void M5(this Geek g, string str)

{

Console.WriteLine(str);

}

}

public class Program {

// Main Method

public static void Main(string[] args)

{

Geek g = new Geek();

g.M1();

g.M2();

g.M3();

g.M4();

g.M5("Method Name: M5");

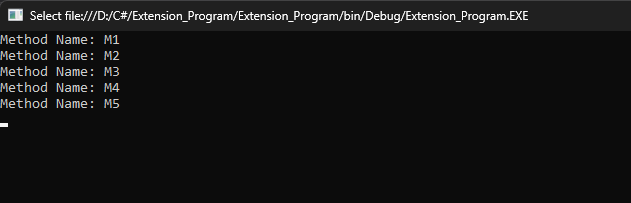
Console.ReadKey();

}

}

}

**Output:**

****