P131

1. 如何定义静态方法？静态方法有何特点？

答：静态方法：使用static修改的方法就是静态方法，否则就是非静态方法。

二、差异比较：

**第一：**C#静态成员：

（1）静态成员属于类所有，非静态成员属于类的实例所有。

（2）每创建一个类的实例，都会在内存中为非静态成员新分配一块存储。

总结：静态成员属于类所有，为各个类的实例所公用，无论类创建了多少实例，类的静态成员在内存中只占同一块区域。

**第二：**C#静态方法:

(1) C#静态方法属于类所有，类实例化前即可使用。

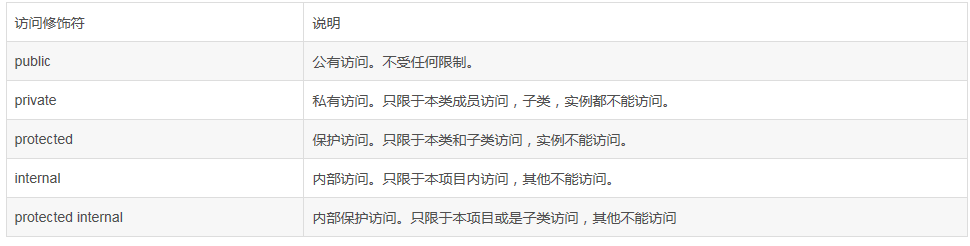
（2）非静态方法可以访问类中的任何成员，静态方法只能访问类中的静态成员。

（3）因为静态方法在类实例化前就可以使用，而类中的非静态变量必须在实例化之后才能分配内存。这样的话C#静态方法调用时无法判断非静态变量使用的内存地址，所以静态方法所以无法使用非静态变量，但静态变量地址对类来说是固定的，所以静态方法可以使用静态变量。

**第三：**C#静态方法是一种特殊的成员方法，它不属于类的某一个具体实例，而是属于类本身。所以对于静态方法不需要首先创建一个类的实例，而是采用    类名.静态方法   的格式。

1. 什么是访问控制符？有哪些访问控制符？那些可以修饰类？那些用来修饰域和方法？

答：C#共有五种访问修饰符：public、private、protected、internal、protected internal。



1. 编写一个C#程序定义一个表示学生的类student，包括域“学号”“班级”“姓名”“年龄”；方法获得其中的值。

class Student {

private int Sno;//学号

private int Gno;//班号

private string Sname;//姓名

private char Ssex;//性别

private int Sage;//年龄

public int Sno1 { get => Sno; set => Sno = value; }

public int Gno1 { get => Gno; set => Gno = value; }

public string Sname1 { get => Sname; set => Sname = value; }

public char Ssex1{ get => Ssex;

set {

if(value!='男'||value!='女')

{ Ssex = '男'; } Ssex = value; }}

public Student(int n,int g, string name, char sex,int age) { Sno1 = n; Gno1 = g; Sname1 = name; Ssex1 = sex;

Sage1= age; } public int Sage1 { get => Sage; set => Sage = value; }

public void OutStudent()

{Console.WriteLine("输出学生学号：{0}，年级号：{1}，姓名：{2}，性别：{3}，年龄：{4}",Sno1,Sage,Sname1,Ssex1,Sage1); } }

class Program{ static void Main(string[] args)

{ Console.WriteLine("请输入学生学号：");{

static void Main(string[] args) {

Console.WriteLine("请输入学生学号：");

int n1 = int.Parse(Console.ReadLine());

Console.WriteLine("请输入学生年级号：");

int n2 = int.Parse(Console.ReadLine());

Console.WriteLine("请输入学生姓名：");

string n3 = Console.ReadLine();

Console.WriteLine("请输入学生性别：");

char n4 = char.Parse( Console.ReadLine());

Console.WriteLine("请输入学生年龄：");

int n5 = int.Parse(Console.ReadLine());

Student s = new Student(n1,n2,n3,n4,n5);

s.OutStudent();

Console.WriteLine("修改xues年龄：");

s.Sage1 = int.Parse(Console.ReadLine());

s.OutStudent();} }

4．银行

static void Main(string[] args)

{

Account a = new Account("147852",10000);

Console.WriteLine("请输入密码");

a.CheckAccount(Console.ReadLine());

TrustAccount b = new TrustAccount();

while(a.check) {

Console.WriteLine("请输入服务数字:");

Console.WriteLine("数字 1 显示余额");

Console.WriteLine("数字 2 修改密码");

Console.WriteLine("数字 3 存取金额");

Console.WriteLine("数字 4 信誉额度");

Console.WriteLine("数字 5 退出ATM");

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

switch (severnumber) { case 1: a.ShowMoney(); break;

case 2: Console.WriteLine("请输入新密码"); a.SetAccount(Console.ReadLine());break;

case 3: Console.WriteLine("请输入存取金额，如-100或100");

a.SetMoney(Convert.ToInt32(Console.ReadLine()));break;

case 4: Console.WriteLine(b); break;

case 5: a.check = false; break;

default: Console.WriteLine("请输入正确的服务数字"); break; }}}

class Account

{

public Account(string account,decimal money)//构造函数

{

this.Account1 = account;

this.Money1 = money;

}

public Account() { }

private string account;//密码

public string Account1

{

get { return account; }

set { account = value; }

}

private decimal Money;//卡内金额

public decimal Money1

{

get { return Money; }

set { Money = value; }

}

public bool check;

public void SetMoney(decimal money)//存取

{

if (money >= 0)

{

this.Money1 += money;

Console.WriteLine("现余额"+this.Money1);

}

else

{

if (this.Money1 >= -money)

{

this.Money1 += money;

Console.WriteLine("现余额"+this.Money1);

}

else {

Console.WriteLine("金额不足");

}

return;

}

}

public void CheckAccount(string account)//核对密码

{

if (this.Account1 == account)

{

Console.WriteLine("密码正确");

check = true;

}

else

{

Console.WriteLine("密码错误，请重新输入");

check = false;

}

}

public void SetAccount(string account)//修改密码

{

this.Account1 = account;

Console.WriteLine("密码修改已完成");

}

public void ShowMoney()//显示余额

{

Console.WriteLine("您的余额：{0}",this.Money1);

}

}

class TrustAccount : Account

{

public TrustAccount() { }

private int trust = 100;//信誉额度

public int Trust

{

get { return trust; }

set { trust = value; }

}

public override string ToString()//重写ToString()方法

public void SetAccount(string account)//修改密码

{

this.Account1 = account;

Console.WriteLine("密码修改已完成");

}

public void ShowMoney()//显示余额

{

Console.WriteLine("您的余额：{0}",this.Money1);

}

}

class TrustAccount : Account

{

public TrustAccount() { }

private int trust = 100;//信誉额度

public int Trust

{

string s;

s = "你的信誉度:" + trust;

return s;

}

}

}

}