C#程序设计

第6章 类和对象

杨琦

西安交通大学
计算机教学实验中心
http://ctec.xjtu.edu.cn

授课内容

- 6.1 理解面向对象思想
- 6.2 理解类和对象
- 6.3 掌握字段和方法
- 6.4 掌握只读字段和常量
- 6.5 掌握重载
- 6.6 掌握构造函数
- 6.7 理解析构函数
- 6.8 理解垃圾回收

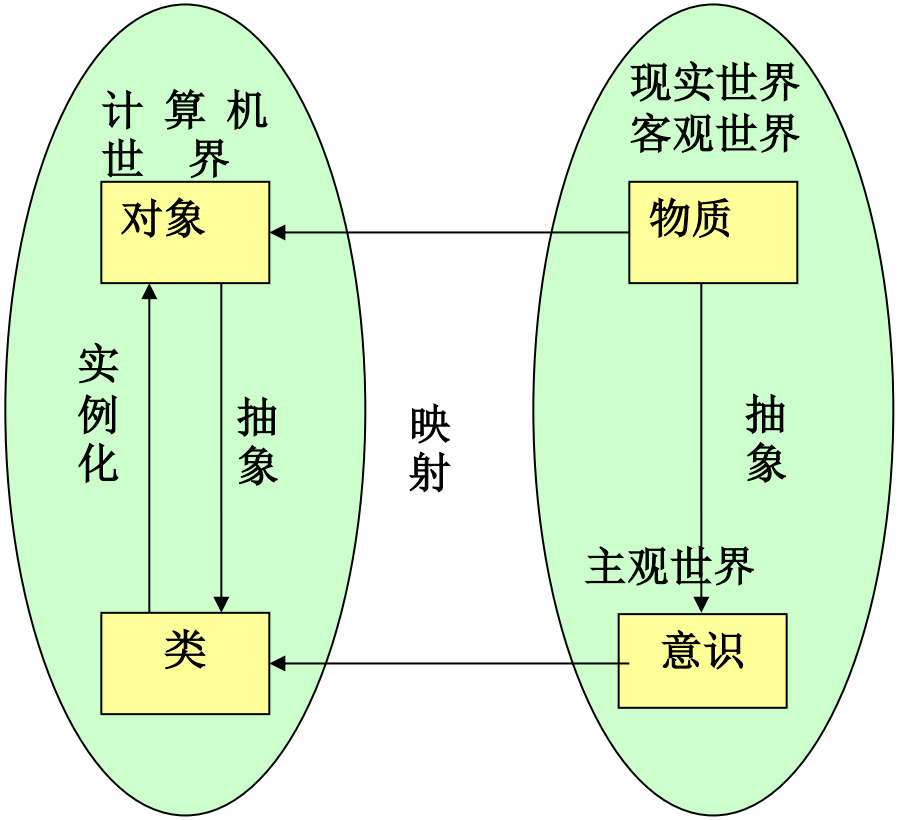
6.1、面向对象基本概念

- 面向机器的程序设计：为特定的硬件系统专门设计，运行速度和效率都很高，但可读性能和可移植性能很差

- 面向过程的程序设计(Structure Programming)
  - 以具体的解题过程为研究和实现的主体，采用函数来描述操作
  - 主要焦点放在指令的组合和优化上。

- 面向对象程序设计(Object Oriented Programming)
  - 以需解决的问题中所涉及的各种对象为主体
  - 以数据为中心，采用对象来描述内部属性和操作方法
  - 主要焦点放在组织程序的数据和功能上

从面向过程到面向对象



- 客观世界的组成：
  - 物质
  - 意识
- **物质**是系统中用来描述客观事物的一个实体，它是构成系统的一个基本单位。
- **意识**是物质之间的关联；物质的构成、规律。

从面向过程到面向对象

对象的概念是面向对象技术的核心所在。面向对象技术中的对象就是现实世界中某个具体的物理实体在计算机逻辑中的映射和体现。

一部移动电话：它由天线、发射部件、接收部件、显示屏、按键、专用集成电路芯片及外壳组成；

它有着其实在的功能，可以打电话，可以发短消息，可以存储、输入和编辑各种个人信息，甚至可以上网。

一辆自行车，它由车架、车轮、脚踏和传动机构、变速机构等组成；它具有代步功能，它可以变速骑行。

面向对象程序设计的特征

- **抽象：数据抽象+代码抽象，如电视机、学生**

- **封装：封装是把一个对象的外部特征和内部实现细节分离开来**
  - 指将一个数据和与这个数据有关的操作集合在一起，形成一个能动的实体（对象），用户不必知道对象行为的实现细节，只需根据对象提供的外部接口访问对象即可
  - 封装可以隐藏实现细节，使得代码模块化

OOA－Object Oriented Analysis	面向对象的分析
OOD－Object Oriented Design	面向对象的设计
OOI－Object Oriented Implementation	面向对象的实现

面向对象程序设计的特征

- **继承：继承是让某个类型的对象获得另一个类型的对象的特征。**
  - 通过继承创建的新类称为“子类”或“派生类”。
  - 被继承的类称为“基类”、“父类”或“超类”。
  - 继承的过程，就是从一般到特殊的过程。

- **多态：指允许不同类的对象对同一消息作出响应。**
  - 多态性具有灵活、抽象、行为共享、代码共享的优势
  - 很好的解决了应用程序函数同名问题

6.2、类与对象的声明和定义

- 面向对象程序设计具有许多优点：

- 1）开发时间短，效率高，可靠性高，所开发的程序更健壮。由于面向对象编程的可重用性，可以在应用程序中大量采用成熟的类库，从而缩短了开发时间。

- 2）应用程序更易于维护、更新和升级。继承和封装使得应用程序的修改带来的影响更加局部化。

6.2、类与对象的声明和定义

- 具有相同或相似性质的对象的抽象就是类。
- 对象的抽象是类，类的具体化就是对象，也可以说类的实例是对象
- 类（Class）类具有属性，它是对象的状态的抽象，用数据结构来描述类的属性。
- 类具有操作，它是对象的行为的抽象，用操作名和实现该操作的方法来描述。

6.2、类与对象的声明和定义

- 类具有表示其数据和行为的成员。这些成员包括：
  - 字段
  - 属性
  - 方法
  - 事件
  - 运算符重载
  - 索引器
  - 构造函数
  - 析构函数

6. 2、类与对象的声明和定义

- 类是用户自定义数据类型。

- C#定义类的一般形式为：
- [类修饰符] class 类名
- {
- 成员列表
- }

- 类定义与结构体类型定义一样，系统不会为它分配存储空间。

6. 2. 1. 类的封装

- 常见的类声明中只涉及4个访问修饰符，最常用的可访问性级别有：public和internal，类的默认可访问级别为internal 。

- 类的默认修饰符是internal；成员的修饰符是private

类成员的类型

数据成员（存储数据）	函数成员（执行代码）
字段 常量	方法 属性 索引 事件 运算符 构造函数 析构函数

6.2.1 类的成员定义

- 数据成员的一般定义形式

- 类在定义时必须给出各数据成员的声明，声明数据成员的一般形式为：

- class 类名
- {
- …… 访问修饰符 数据成员类型 数据成员名称列表; ……
  }

6.2.1 类的成员定义

- 2. 函数成员的一般定义形式
- 类的函数成员具备访问类数据成员的权限，一般被作为类与外界交互的接口。
- class 类名
- {
- ……
- 访问修饰符 返回类型 成员函数名(形式参数列表)
- {
- 函数体
- }
- ……
- }

6.2.1 类的成员定义

4个访问修饰符		4个类性质修饰符	
修饰符	含义	修饰符	含义
public	访问不受限制	new	适用于嵌套类。它指定类隐藏同名的继承成员
protected	访问范围限定于它所属的类 或从该类派生的类型	abstract	用于表示所修饰的类是不完整的，并且它只能用作基类，即表示该类为抽象类
internal	访问范围限定于此程序	sealed	适用于密封类。用于防止从所修饰的类派生出其他类
private	访问范围限定于它所属的类型	static	用于标记静态类。静态类不能实例化，不能用作类型，而且仅可以包含静态成员。

6.2.1 类的成员定义

- 3．成员的访问控制
- 类成员的访问源有两个：类成员和类用户。
- 类成员指类本身的函数成员，
- 类用户指类外部的使用者，如另一个类的成员函数等。

6.6.2 类对象的定义

- 类是引用类型的，当定义了对象之后，系统会为数据引用分配存储空间，但是用来保存对象的实际数据的空间并没有分配。

- 创建对象时需要使用new运算符。new运算符可以为任意指定类类型的实例分配并初始化内存。创建对象的语法形式如下：
- 对象名=new 类名();
- 对象的初始化
- C#中实现对象初始化的方式有两种：借助构造函数或使用对象初始化列表。

6.6.2 类对象的定义

- 2. 使用对象初始化列表实现对象的初始化
- 使用对象初始化列表进行对象初始化的语法形式如下：
- 或
- //包括构造函数参数列表的形式1
- new 类名(参数列表){对象初始化列表}

- new 类名{对象初始化列表}//形式2

- 3. 对象的运算
- 同一类的两个对象支持赋值、相等与不相等运算。

例6-1　简单的日期类

类名：Date
数据成员： 年（整型） 月（整型） 日（整型）
成员函数： 初始化（年，月，日） 输出_年月日（）

例6-1　定义一个Date类

- 输入和输出
- date1:
- 2013-3-28
- date2:
- 2013-3-28

例6-1　定义一个Date类

```
1.  using System;
2.  class Date{
3.      int day = 1, month = 1, year = 1900;
4.      public Date(int yy, int mm, int dd)    {
5.          init(yy, mm, dd);
6.      }
7.      public Date(Date d)    {
8.          year = d.year; month = d.month; day = d.day;
9.      }
10.     public void init(int yy, int mm, int dd)    {
11.         month = (mm >= 1 && mm <= 12) ? mm : 1;
12.         year = (yy >= 1900 && yy <= 2100) ? yy : 1900;
13.         day = (dd >= 1 && dd <= 31) ? dd : 1;
14.     }
```

例6-1　定义一个Date类

```
1.   public void print_ymd()
2.      { Console.WriteLine(year + "-" + month + "-" + day); }
3.   }
4.   class My  {
5.      static int Main()    {
6.         Date date1 = new Date(2013, 3, 28);
7.         Date date2 = new Date(date1);
8.         Console.WriteLine("date1:");
9.         date1.print_ymd();
10.        Console.WriteLine("date2:");
11.        date2.print_ymd();
12.        return 0;
13.     }
14.  }
```

有两种方法可存储对象

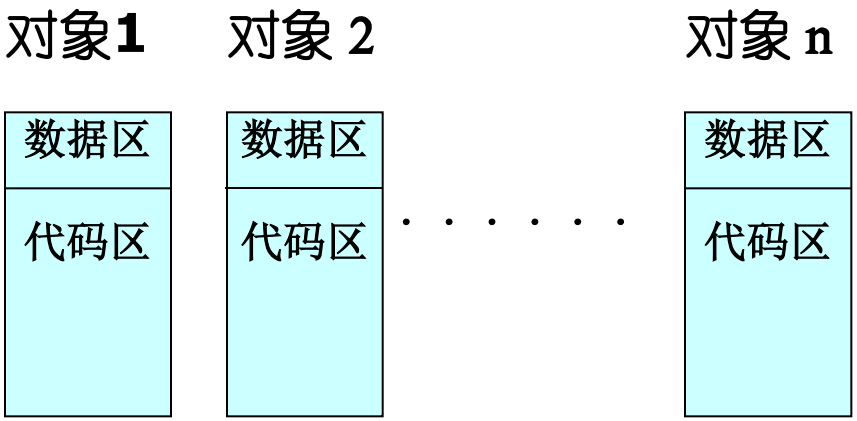


图1 各对象完全独立地安排内存的方案

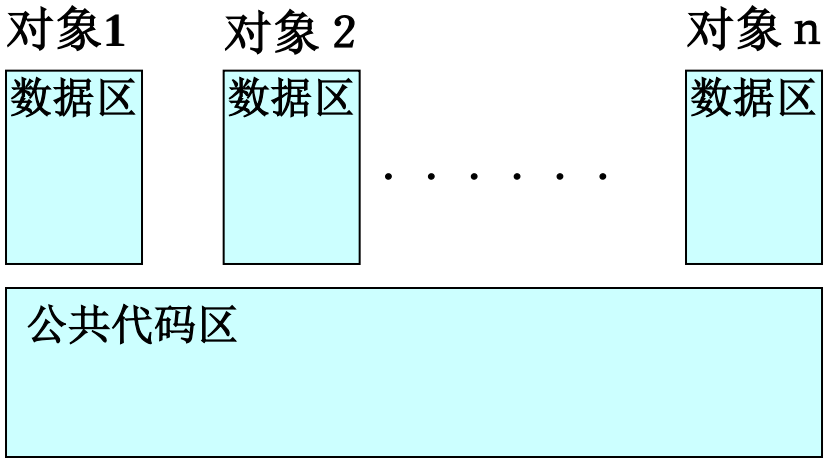


图2 各对象的代码区共用的方案

构造函数

- **构造函数（constructor）**
  - 用于对对象进行初始化的一个或一组函数。
- **构造函数是特殊的公有成员函数，其特征如下：**
  - **1.**函数名与类名相同。
  - **2.**构造函数无函数返回类型说明。
  - **3.**在新的对象被建立时，该对象所属的类的构造函数自动被调用。
  - **4.**构造函数可以重载。
  - **5.**默认的构造函数没有任何参数。但是如果您需要一个带有参数的构造函数可以有参数，这种构造函数叫做参数化构造函数。

析构函数

- 1. 析构函数的名称是在类的名称前加上一个波浪形（~）作为前缀，如~Person（）。
- 2. 析构函数没有函数值，也不带任何参数。
- 3. 一个类有一个也只有一个析构函数。
- 4. 析构函数用于在结束程序（比如关闭文件、释放内存等）之前释放资源。

例6-2　定义一个Person类

类名：Person
数据成员： 姓名（字符型） 性别（字符） 年龄（整型）
成员函数： Person （姓名，性别，年龄） 输出（）

例6-2　人事资料输入输出

输入和输出

```
Enter a person's name, age and sex:
zhangsan
19
f
person1:
zhangsan        19      f
person2:
zhangsan        19      m
Now destroying the instance of Person
Now destroying the instance of Person
```

```
using System;
class Person{
    string Name = "XXX";
    int Age = 0;
    char Sex = 'm';
    public Person(string name, int age, char sex) {
        Name = name;
        Age = age;
        Sex = (sex == 'm' ? 'm' : 'f');
    }
    ~Person()    {
        Console.WriteLine("Now destroying the instance of Person");
    }
    public void ShowMe()    {
        Console.WriteLine("{0} \t {1} \t  {2} ", Name, Age, Sex);
    }
}
```

```
class My{
    static int Main()    {
        string name;
        int age;
        char sex;
        Console.WriteLine("Enter a person's name, age and sex:");
        name = Console.ReadLine();
        age = Convert.ToInt32(Console.ReadLine());
        sex = Convert.ToChar(Console.ReadLine());
        Person person1 = new Person(name, age, sex);
        Console.WriteLine("person1: \t");
        person1.ShowMe();
        Person person2 = new Person("zhangsan", 19, 'm');
        Console.WriteLine("person2: \t");
        person2.ShowMe();
        return 0;
    }
}
```

例6-3 统计学生课程平均分

学号	姓名	高数	英语	体育	平均分
881300	Lucy	95	90	88	91
881301	Sarah	90	92	80	87
881302	David	72	75	80	75
881303	Tom	86	80	85	83
881304	John	50	45	100	65

例6-3 统计学生课程平均分

```
1.  using System;
2.  class Student  {
3.      public string ID = "0000";  //学号
4.      public string Name = "xxx";        //姓名
5.      public int ScoreMaths=0;  //数学成绩
6.      public int ScoreEnglish=0;         //英语成绩
7.      public int ScorePE=0;       //体育成绩
8.      public int GetGPA() {     //课程总分
9.          return ScoreMaths + ScoreEnglish + ScorePE;
10.     }
11.     public void ShowMe()    {
12.         Console.Write(ID + "\t" + Name + "\t");
13.         Console.Write(ScoreMaths + "\t" + ScoreEnglish + "\
    ScorePE + "\t");
```

例6-3 统计学生课程平均分

```
1.  class My {
2.      static int Main()  {
3.          Student[] xjtuStudent = new Student[2];
4.          int i;
5.          Console.WriteLine("请按以下顺序输入学生信息：学号
6.          for (i = 0; i < 2; i++)    {
7.              xjtuStudent[i] = new Student();
8.              xjtuStudent[i].ID = Console.ReadLine();
9.              xjtuStudent[i].Name = Console.ReadLine();
10.             xjtuStudent[i].ScoreMaths = Convert.ToInt32(Cons
11.             xjtuStudent[i].ScoreEnglish = Convert.ToInt32(Co
12.             xjtuStudent[i].ScorePE = Convert.ToInt32(Console
13.         }
```

例6-3 统计学生课程平均分

```
1.    Console.WriteLine("----------------------------------------
2.        Console.WriteLine("学号   姓名   高数   英语   体育   平
3.        Console.WriteLine("----------------------------------------
4.        for (i = 0; i < 2; i++)
5.        {
6.            xjtuStudent[i].ShowMe();
7.        }
8.        return 0;
9.    }
10. }
```

例6-4　点类

- 在二维平面空间上，使用x-y坐标可以确定一个点。定义一个点类，然后编写主程序测试这个点类。

- 程序运行结果：
- Point p: [30, 50]

点类

类名：Point
数据成员： x（整型） y（整型）
成员函数： 构造函数Point（ref Point p） 构造函数Point（） 输出点的坐标Print（）

例6-4　点类

```
1.  using System;
2.  class Point
3.  {
4.      int x,y;
5.      Point(int x, int y) {// 构造函数
6.          this.x = x;
7.          this.y=y;
8.      }
9.      Point(ref Point p){ // 构造函数
10.         this.x = p.x;
11.         this.y = p.y;
12.     }
```

例6-4　点类

```
1.    void print()
2.        {
3.            Console.WriteLine("[{0}, {1}] ", x, y);
4.        }
5.        static int Main()
6.        {
7.            Point p1 = new Point(12,18);
8.            Point p2 = new Point (ref p1);
9.            p2.print();
10.           return 0;
11.       }
12.   }
```

C#类的实例成员

- ① 实例成员
- 实例成员有时称为非静态成员，它与类的对象相关。
- 当字段、方法、属性、事件、索引器、构造函数或析构函数的声明中不包含 static 修饰符时，它声明为实例成员。

- 实例成员具有以下特点：
- 使用 “类对象名.成员名”的形式进行引用；
- 类的每个对象分别包含一组该类的所有实例字段；
- 实例函数成员作用于类的给定对象，可借助this访问器访问。

C#类的静态成员

- 使用 static 关键字把类成员定义为静态的。当我们声明一个类成员为静态时，意味着无论有多少个类的对象被创建，只会有一个该静态成员的副本。

- 关键字 static 意味着类中只有一个该成员的实例。

- 静态变量用于定义常量，因为它们的值可以通过直接调用类而不需要创建类的实例来获取。静态变量可在成员函数或类的定义外部进行初始化。您也可以在类的定义内部初始化静态变量。

静态成员具有下列特征：

- 使用 “类名.成员名”的形式进行引用；
- 静态字段的存储由类的所有对象共享，只有一个副本；
- 静态函数成员不能作用于具体的对象，不能使用 this访问器访问。

【例6-5】C#类的静态成员

- 输入输出
- 1　　2　　3　　4　　5　　6　　7　　8　　9　　10

【例6-5】C#类的静态成员

```
1.  using System;
2.  class My{
3.      static int count = 0;
4.      static int func()   {
5.          return ++count;
6.      }
7.      static int Main()   {
8.          // 分别调用次func( ) 函数
9.          for (int i = 0; i < 10; i++)
10.             Console.Write( func() +"\t" );
11.         Console.WriteLine( );
12.         return 0;
13.     }
14. }
```

【例6-6】定义计数器类Counter

- 定义计数器类Counter，要求具有以下成员：计数器值；
- 可进行增值计数inc；
- 可进行减值计数dec；
- 可设置计数器的值set;
- 可提供计数值showme。

【例6-6】定义计数器类Counter

```
1.  using System;
2.  class Counter
3.  {
4.      int x=0;
5.      public Counter(int a)
6.      { x = a; }
7.      public void inc()
8.      { x++; }
9.      public void dec()
10.     { x--; }
11.     public void ShowMe()
12.     { Console.WriteLine("counter:" + x); }
```

【例6-6】定义计数器类Counter

```
1.   static int Main()   {
2.         Counter x = new Counter(10);
3.         x.ShowMe();
4.         x.inc();
5.          x.ShowMe();
6.         x.dec();
7.          x.ShowMe();
8.         return 0;
9.      }
10. }
```

字段的初始化

- 对类的字段进行初始化时可以有两种方式：显式字段初始化和隐式字段初始化。

- ① 显式字段初始化
- 对字段进行显式初始化指在字段声明时直接在字段名后添加等号和字段初始值。

- ② 隐式字段初始化
- 当声明字段时没有初始化语句，则字段的值会被编译器设为默认值。默认值由字段的数据类型决定，预定义的简单类型字段被设置为0或false，引用类型字段被设为null。

【例6-7】设计一个Circle类

- 设计一个Circle（圆）类，其属性有圆心坐标x和y，半径r。其成员函数为Set(int,int,double)和double Area()，实现并测试这个类。

【例6-7】设计一个Circle类

类名：Circle
数据成员： 半径r（double） x坐标（double） y坐标（double）
成员函数： 构造函数Circle(double a, double b, double c) 设置圆心坐标和半径Set（double a, double b, double c） 输出圆心坐标和半径Print（）

【例6-7】设计一个Circle类

```
1.  class Circle
2.  {
3.      double x, y,r;
4.      public Circle(double a, double b,double c)
5.      {
6.          SetCircle (a, b, c);
7.      }
8.      public void SetCircle(double a, double b,double c)
9.      {
10.         x = a;      y = b;      r = c;
11.     }
12.
```

【例6-7】设计一个Circle类

```
1.  public void Print()
2.  {
3.          Console.WriteLine("[" + x + ", " + y + "], "+"半径="+r)
4.   }
5.  static int Main()   {
6.          Circle p = new Circle(30, 50,10);
7.          Console.Write("Circle p: ");
8.          p.Print();
9.          return 0;
10.     }
11. }
```

【例6-8】定义盒子Box类

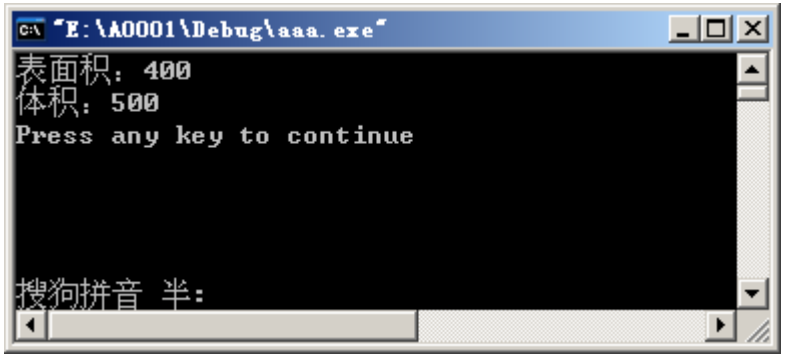
- 定义盒子Box类。要求具有以下成员：
- 可设置盒子形状；
- 可提供盒子体积；
- 可提供盒子表面积。

【例6-8】定义盒子Box类

类名：Box
数据成员： **长x**（double） 宽y（ double ） 高z（ double ）
成员函数： 构造函数Box(double a, double b, double c); 计算表面积Area(); 计算体积Volume(); 输出函数Print();

【例6-8】定义盒子Box类

```
1.  static int Main()    {
2.          Box b=new Box(10, 10, 5);
3.          b.Print();
4.          return 0;
5.      }
6.  class Box{
7.      private double x, y, z;
8.      public Box(double a, double b, double c)    {
9.          x = a; y = b; z = c;
10.     }
```

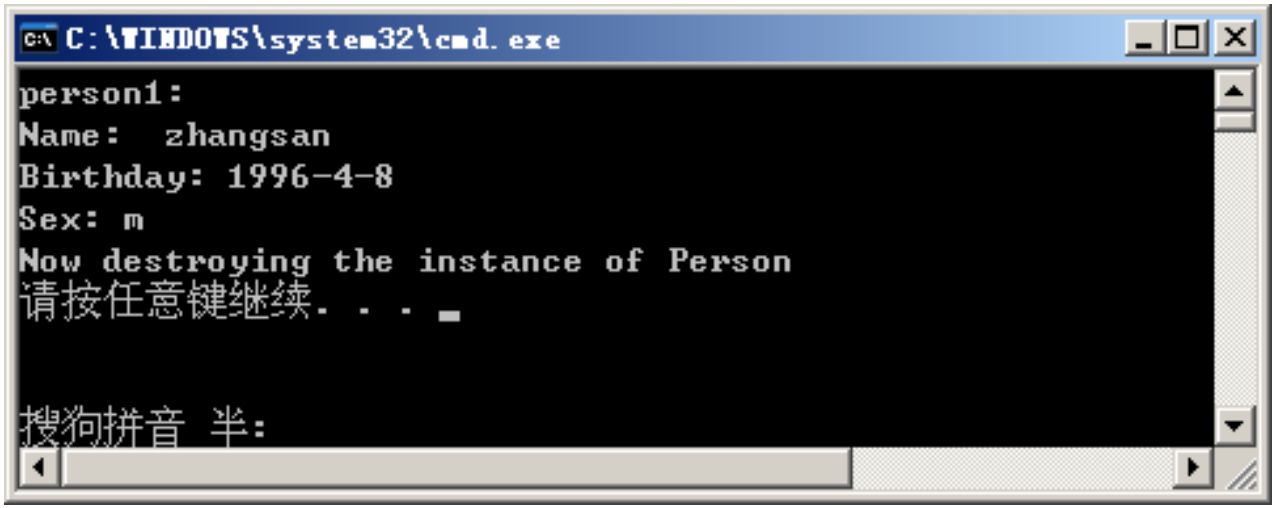


【例6-8】定义盒子Box类

```
1.      public double Area()   {
2.          return 2 * (x * y + y * z + x * z);
3.      }
4.      public double Volume()   {
5.          return x * y * z;
6.      }
7.      public void Print()   {
8.          Console.WriteLine("表面积：" + Area());
9.          Console.WriteLine("体积：" + Volume());
10.     }
11. };
```

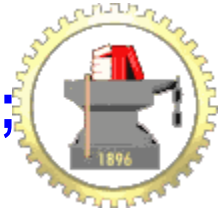
例6-9　定义一个DateTime和Person类

- Person类有姓名、出生日期、性别三个数据成员
- 请分别编写这两个类，并调用主程序测试。



```
using System;
class Person  {
    string Name = "XXX";
    DateTime Age =DateTime.Now;
    char Sex = 'm';
    public Person(string name, DateTime age, char sex)  {
        Name = name;        Age = age;        Sex = (sex == 'm' ? 'm' : 'f');
    }
    ~Person()  {
        Console.WriteLine("Now destroying the instance of Person");
    }
    public void ShowMe()  {
        Console.WriteLine("Name:  {0}", Name);
```

```
        Console.WriteLine("Birthday: " + Age.ToShortDateString());
        Console.WriteLine("Sex: {0}", Sex);
    }
}
class My  {
    static int Main()   {
        DateTime age = new DateTime(1996, 4, 8,0,0,0);
        Person person1 = new Person("zhangsan", age, 'm');
        Console.WriteLine("person1:");
        person1.ShowMe();
        return 0;
    }
}
```



6.2.2 方法

- 1．方法的定义

- 方法是具有名称的可执行代码块，可以从程序的很多不同地方执行，甚至从其它程序中执行。

- 方法的作用在于对类或者类对象的数据进行操作，实例方法既能够访问类的静态成员，也能够访问类的实例成员，而静态方法只能直接访问静态成员。

2．方法的调用

- 类的静态方法调用时需借助类名和成员运算符，语法形式如下：
- 类名.方法名(参数列表)//有参数的静态方法调用
- 类名.方法名()//无参数的静态方法调用

- 当调用者和被调用的静态方法同属一个类时，可以省略类名和成员运算符。
- 根据方法有无返回值，对类的静态方法调用时可以3种形式出现：方法调用表达式、方法调用语句和方法调用的实参。
- 类的静态方法调用支持嵌套调用和递归调用两种形式。

3．Main方法

- Main方法是C#程序的执行入口点，因此每个C#程序都必须有一个并且只能有一个叫做Main的方法才能被执行。
- static int Main(){……}
- static int Main(string[] args){……}

- 如果使用返回值，通常用于报告程序的成功与失败，0通常代表程序执行成功。

常量

- 1. 常量的定义
- 常量的声明语法如下：
- 访问修饰符 const 数据类型 常量名=常量值;

- 用于初始化常量成员的值在编译期必须是可计算的，而且通常是一个预定义简单类型或由它们组成的表达式。
- 成员常量只能在声明时赋值，任何在声明之后给常量成员赋值的动作都会造成编译错误。

- C#中没有全局常量，每个常量都必须声明在类型内。
- 使用readonly定义的只读字段只能在字段声明时初始化赋值或者类的构造函数中对其赋值。

结　束　语

- 学好程序设计语言的唯一途径是

上机练习

- 你的编程能力与你在计算机上投入的时间成

正比