矩形类的定义

```
 class Rect
{
    private int length;
    private int width;

    public  Rect()
    {
        length = 0;
        width = 0;
    }

    public Rect(int length,int width)
    {
        this.length = length;
        this.width = width;
    }

    public int GetLength()
    {
        return length;
    }

    public int GetWidth()
    {
        return width;
    }

    public void SetLength(int length)
    {
        this.length = length;
    }
    public void SetWidth(int width)
    {
        this.width = width;
    }

    public int GetArea()
    {
```

```
            return width * length;
        }
    }
```

矩形类的调用

```
    class Program
    {
        //Main方法 程序入口 代码需要在Main方法中实现
        static void Main(string[] args)
        {
            Rect rect = new Rect(100,50);
            Console.WriteLine(rect.GetLength());
            Console.WriteLine(rect.GetWidth());
            rect.SetLength(200);
            Console.WriteLine(rect.GetArea());
        }
    }
```

常量

静态常量 ：const

动态常量 ：readonly

const 静态常量在程序编译期间确定并完成初始化 编译期常量

readonly 动态常量在程序运行时获得完成初始化（初始化推迟到构造方法中） 运行期常量

```
    class MyClass
    {
        public int a;
        public const int b=100;//const修饰的常量必须在声明时就赋值
        public readonly int c;//readonly 只读 修饰的常量在声明时可以赋值
或不赋值
```

```
6
7           public MyClass()
8           {
9               //b = 200; 不可赋值 因为b是const修饰的常量
10              c = 200;//可以赋值  readonly的常量可以在声明时或者构造方法中赋值
11          }
12
13          public void Fn()
14          {
15              //b = 200; 不可赋值
16              //c = 200; 不可赋值
17          }
18      }
```

const和readonly都是表示不可修改的常量

区别：

1 const在声明时必须设置常量值

readonly在声明时可以选择不设置 有可以在类的构造方法中设置常量值

2 对于类的每一个对象来说 const的常量值 是一样的 readonly常量值可能不一样

3 const的性能更高 对内存无开销 但是限制比较多

readonly则更灵活方便 但是会有内存开销

访问器和属性

访问类的私有变量 可以使用公有的方法来返回私有变量

更为简单的方式是访问器

set　　get

```
1       class MyClass
2       {
3           private int age=10;
4           public int Age//声明类型为Int的Age属性
5           {
6               get
```

```
            {
                if (age < 0)
                    return 0;
                return age;
            }
            set
            {
                age = value;
            }
        }

        private string name = "张收纳";
        public string Name
        {
            get { return name; }
        }
    }


    class Program
    {
        //Main方法 程序入口 代码需要在Main方法中实现
        static void Main(string[] args)
        {
            MyClass mc = new MyClass();
            mc.Age = 100;//调用Age属性的set访问器
            Console.WriteLine(mc.Age);//调用Age属性的get访问器
            Console.WriteLine(mc.Name);//调用Name属性的get访问器
            //mc.Name = "李四"; 编译出错 Name属性没有添加set访问器 所以是只读的
        }
    }
```

static 静态

C#不像C++和C等其他编程语言 C#没有全局变量的概念 如果需要用到全局 就需要使用static

```
    class MyClass
```

```
    {
        //x被定义成静态 意味着不管创建多少个对象  x都只有一份
        public static int x;

        public int GetX()
        {
            return x;
        }
    }


    class Program
    {
        //Main方法 程序入口 代码需要在Main方法中实现
        static void Main(string[] args)
        {

            //即使没有对象存在 x因为是静态变量 存储在静态全局区  依然存在
            MyClass.x = 100;

            MyClass mc1 = new MyClass();
            MyClass mc2 = new MyClass();

            //mc1.x = 200; 编译错误 使用对象名.无法获取静态变量
            MyClass.x = 200;//只能通过类名来访问静态变量

            //对象想获取静态变量 必须通过公有方法返回得到
            Console.WriteLine(mc1.GetX());
            MyClass.x = 20000;
            Console.WriteLine(mc2.GetX());
        }
    }
```

静态成员方法

静态成员方法 只能访问静态成员变量

静态成员方法的调用 也是通过类名调用 与对象无关

```

  class MyClass
```

```
    {
        //x被定义成静态 意味着不管创建多少个对象  x都只有一份
        public static int x;
        public int y;


        public static void Fn()
        {
           // Console.WriteLine(y); 不能访问 因为y不是静态成员
            Console.WriteLine(x);//可以访问 静态方法访问静态变量
        }
    }


    class Program
    {
        static int w;
        //Main方法 程序入口 代码需要在Main方法中实现
        static void Main(string[] args)
        {

            MyClass.x = 200;
            MyClass.Fn();//静态方法 使用 类名.访问调用
       }

        public static void fn()
        {
            w = 100;
        }
    }
```

因为Main方法是静态方法

所以即使没有Program类的对象 我们也可以执行 Main方法

如果在Program里定义普通成员变量 不能在Main方法中使用

实现

1 一个计算器类

包含 两个数值 (int)

包含 一个符号(char或 string)

在不实例化对象的情况下 实现 +-*/%的运算

2 实现一个静态的对象

保证 对象第一次被实例化后

此类不能再次被实例化