# 创建类

语法：

[public] class 类名{

字段；

属性；

方法；

}

Tips：this代表当前这个类的对象。

类中的成员如果不加访问修饰符，默认的是private。

# 静态成员和非静态成员的区别

1. 在非静态类中，可以有实例成员，也可以有静态成员
2. 在调用实例成员的时候必须实例化类，静态方法可以直接类名.方法名
3. 静态函数只能访问静态成员，不能访问实例成员。
4. 实例函数中，既可以使用静态成员，也可以使用实例成员
5. 静态类中只允许出现静态成员。

Tips：静态成员：类中的所有方法都能访问(公用RBQ)

# 使用：

如果想要类当做一个工具类去使用，这个时候可以考虑将类设置成静态类

# This关键字

1. 代表当前对象。
2. 在类中显示的调用本类的构造函数 :this

# 析构函数

语法：

~类名(){

一些释放资源的操作

}

作用:释放资源

# 属性

作用：保护字段，对字段的赋值和取值进行限定

本质：get方法，set方法

属性并不存储值，而是对字段值的限定

Person p = new Person();

p.Name = “张三”; set

Console.WriteLine(p.Name); get

# 命名空间

类是属于命名空间

导入命名空间快捷键：Alt+shift+F10

引用同一个解决方案下不同项目的命名空间

1. 为项目添加另一个项目的引用
2. 手动添加项目的命名空间

# 值类型和引用类型

值类型：

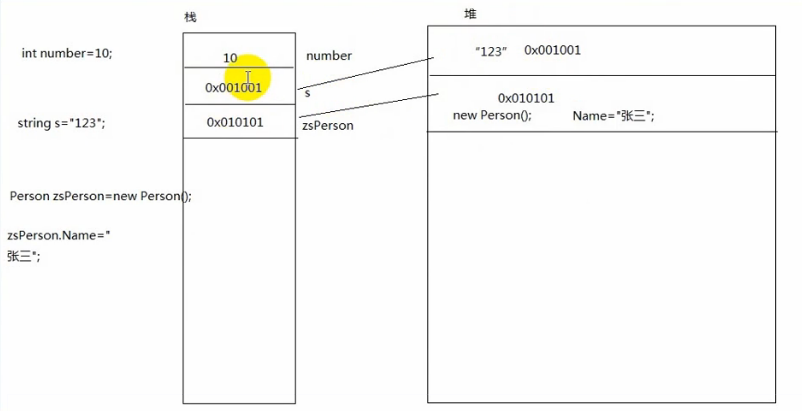
char、double、decimal、int、struct、enum

引用类型：

String、自定义的类、数组

区别：

1. 存储的地方不一样。
   1. 值类型：栈
   2. 引用类型：堆
2. 传递方式不一样。
   1. 值传递
   2. 引用传递



# String

特性：字符串的不可变性

更改了string的值之后，会更改引用的地址。

方法：

String：char类型的只读数组

String→char[] string.ToCharArray();

Char→string new string(char []);

转大小写：

String name = “c#”;

Name = name.ToUpper(); 转大写

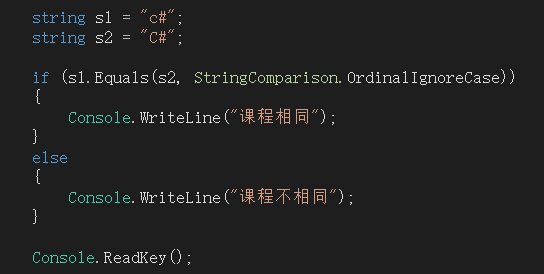
Name = name.ToLower(); 转小写

转换为字节数组:byte [ ] buff = Encoding.default.GetBytes(string s);

比较方法：

Equals：对比两个字符串的值

使用equals忽略大小写



属性：

Length：获得当前字符串中的字符个数。

分割：

Split()

String teststring = “test\_asd, …+”;

String [] str = string.split(new char[] {“\_”,” ”,”…”,”+”},stringSplitOptions.RemoveEmptyEntries);

替换：

Replace(old str,new str);

包含：

Contains(“xxx”);

截取：

Substring(start index,length); 包含截取位置

以..开始 以…结束

StartsWith(“”)

EndWith(“”)

Join

把字符串里面的每一个元素按照指定字符串链接。

# 继承

关键字 ：

特性：单根型，传递性

例如 student ：person

创建多个类的时候有拥有多个相同的字段时，可以把公共的字段提取出来创建一个父类。

子类继承了父类了什么？

1. 父类的非私有的方法和属性。
2. 子类没有继承父类的构造方法。

Tips：子类之所以能够用到父类的非私有的属性和方法，是因为子类在初始化对象的时候先创建了父类的对象(默认调用父类无参的构造函数)，之后才创建自己的对象。

Tips：子类初始化的时候会先在内部初始化一个父类，这样才能使用父类的方法，所以会走父类默认的构造方法，如果想走固定的构造方法，则需要使用base关键字。

调用父类的构造方法进行初始化

Public 类名(参数列表): base(与父类参数一一对应){ this.\_me = value}

:new（不建议使用）

隐藏父类的成员

# 里氏转换

1. 子类可以赋值给父类(创建父类的时候可以用子类赋值)。
2. 如果父类中装的是子类对象，那么可以将这个父类强转为子类对象。

IS:表示类型转换，如果转换成功则返回true，否则false。

AS：表示转换类型，如果能转返回对象，否则返回null。

数组相比集合的瓶颈

1. 长度不可变
2. 类型单一

方法:

Add() 添加单个元素

AddRange() 添加一个集合，并且能直接打印输出

Clean() 删除所有的元素

Remove（） 删除单个元素（写谁删除）

RemoveAt（） 根据下标删除

RemoveRange（star，end） 从起始位置删除指定范围的元素

Reverse（）反转

Insert（index,obj）指定位置插入元素

InsertRange（index，obj） 指定的位置插入一个集合

Contains（） 是否包含某个指定的元素

长度问题：

每次集合中实际包含的个数超过了可包含的次数，ArrayList就会在内存中多申请一倍的空间以保证集合长度够用。

# Hashtable

Tips:键是唯一的，不能重复；值可以重复

想替换值可以 对象[键] = 新值；

对象名.containsKey：是否包含某个Key

对象名.containsValue：是否包含某个Value

Clean（） 移除所有元素

Remove（） 根据键移除值

# Path

路径工具类

GetFileName(path) 获得文件名

GetFileName Extension（path） 获得文件名不要扩展名

# File

Creat()创建一个文件

Delete()删除一个文件

Copy()复制一个文件

# 编码格式

编码：以什么样的格式把字符串转换成二进制

# 装箱和拆箱

装箱：值类型转换为引用类型

拆箱：引用类型转换为值类型

Tips：如果有继承关系，那么可能发生装箱或者拆箱，如果没有，那么一定不会发生任何的装箱或拆箱。

应尽量避免装箱或者拆箱

# IO流

FileStream 和 File区别:

File是一次性读完

FileStream 是每次读固定的一点 (水缸舀水)

基本步骤

1. 创建文件读取流

FileStream fsRead = new FileStream(path,FileMode.openOrCreat,FileAssess.Read);

1. 创建水瓢

Byte [] b = new byte[1024\*1024\*5];

1. 读取文件,返回读取到有效位数

Int r = FsRead.Read(b,0,b.Length);

1. 创建输出流

FileStreamfsWrite= new FileStream(path,FileMode.openOrWrite,FileAssess.Write);

FsWrite.write(b,0,r);

1. 关闭资源

如果不想手动写 可以运用

Using () {} 把操作写在里面

# 多态

概念：让一个对象能够表现出多种的状态(类型)

多态运用的核心思想：

父类的对象指向子类的实现同时代表子类，这样就可以用一个实例代表多个对象，调用的时候调用父类对象即可实现多对象的调用。

引用其他对象的方式：

1. 创建构造方法，传入需要的对象
2. 在方法的参数列表中写入需要的对象
3. 把对象设置成字段，提供其属性，通过属性获得

多态的步骤

1. 找出对象的共有特征和行为，创建其父类。
2. 如果父类也需要实现的话可以使用虚方法，关键字为virtual，子类重写父类的方法关键字为override；如果不需要父类的具体实现或者不知道具体动作是什么，可以采用接口或者抽象方法。
3. 创建父类对象指向子类的实现。

实现方法：

1. 虚方法
   1. 把父类的方法标记为虚方法，使用关键字virtual，然后子类方法中使用关键字override；
2. 抽象类:当父类不知道怎么实现方法的时候，交给子类具体实现
   1. 抽象类概念：关键字是abstract，并且没有任何的实现
   2. 子类实现抽象类，必须实现抽象类里面的所有抽象成员（除非子类也是抽象类）
   3. 抽象类之中可创建非抽象的成员
   4. 抽象成员必须在抽象类当中
   5. 子类重写方法的返回值和参数必须和父类的相同
3. 接口:一种规范、行为、能力

[public] interface I….able{

成员();

}

Tips:1.接口中成员的访问修饰符默认为public

2.不能写具体方法的函数(不能拥有方法体)

3.不能有字段

（只能有方法）

1. 不能有构造方法
2. 接口和接口之间可以实现多继承
3. 子类必须实现接口中的所有方法
4. 接口只能继承于接口
5. 多继承的时候父类必须要先写在接口前面

# 实现场景

# 访问修饰符

1. public:公共的，公开的
2. private:私有的，只能在类的内部访问
3. protected:受保护的，只能在类的内部或者子类中访问。
4. internal:只能在当前程序集(当前项目)中访问
5. internal protected:

tips:能够修饰类的访问修饰符只有public 和 internal，

子类的访问权限不能高于父类的访问权限，因为会暴露父类成员。

# 简单工厂模式

根据用户的要求，提供一个父类,根据要求装入特定的子类

# 值传递和引用传递

1. 值传递时复制的是值本身
2. 引用对象传递的时复制的是对这个对象的引用（即堆中的内存地址）

Tips：如果参数列表加了ref，在进行值传递的时候，传递的是值的地址。

# 序列化与反序列化

序列化：对象转化为二进制

反序列化：二进制转化为对象

作用：传输数据

Tips：序列化的对象不能被继承

序列化步骤:

1. 在类的上面写上关键字[Serializable]
2. 通过流把对象写到特定的路径
3. 创建BinaryFormatter对象序列化

反序列化步骤：

1. 通过流读取文件
2. 创建BinaryFormatter对象反序列化
3. 强转反序列化得到的Object对象