1、泛型类、方法（Generic）

泛：不具体化

型：数据类型

泛型：就是没有具体化的数据类型

List<?>这个？表示数据类型（List也是具体化的）

泛型(定义)：定义没有具体化，不限制是什么数据类型，在使用的时候：必须具体化

泛型类：属性，字段，方法（参数，返回值），构造函数，

如果在一个类中，方法的参数或者返回值，写方法的时候不想确定，想在使用的时候确定；

在这个A类中，添加一个泛型类，在new A类的时候，使用其中方法的时候，如果参数或返回值，与你传递进来的参数或返回值的数据类型一致，就变成你new A类的

例： 类名<T1,T2>

需要用到Dynamic：动态类型

表示在运行时，解析需要操作的类型

//泛型是可以加约束的

//如果需要实现加减乘除，需要写很多的方法重载，针对不同的数据类型

//对此我们采用泛型方法

public T Add<T, T1, T2>(T1 num1,T2 num2) where T:struct ,T1,T2//约束返回值类型是，值类型（结构类型），T1,T2同样的约束为结构类型

{

// return num1+num2; 是错误的，当返回类型和参数类型不一样，无法转变

//需要通过Dynamic类型

dynamic a = num1;//将在运行a时，解析a

dynamic b = num2;

return a+b;

}

使用泛型可以有效的节省代码，并且维护非常方便

在方法名后添加<>，就是泛型方法

在类名后面添加<>,就是泛型类

public class Generic<T>

{

List<T> alist=new List<T>();

//索引器

//public T this[int index]{getlset}索引器的本质

/// <summary>

/// 这个类自己的索引器

/// </summary>

/// <param name="index"></param>

/// <returns></returns>

public T this[ int index]

{

get { return alist[index]; }

set { alist[index]=value; }

}

public List<T> list

{

get { return alist; }

}

}