* **自定义泛型类**

class Person<T>{

//使用T类型定义变量

private T info;

//使用T类型定义一般方法

public T getInfo(){

return info;

}

public void setInfo(T info){

this.info = info;

}

//使用T类型定义构造器

public Person(){}

public Person(T info){

this.info = info;

}

//static的方法中不能声明泛型

//public static void show(T t){

//}

//不能在try-catch中使用泛型定义

//try{}

//catch(T t){}

}

* **自定义泛型类的使用**
* 当实例化泛型类的对象时，指明泛型的类型。指明以后，对应的类中所有使用泛型的位置都变为实例化中指定的泛型的类型。
* 如果我们自定义了泛型类，但是实例化时没有使用，那么默认类型是Object类的。
* **泛型方法**
* 方法，也可以被泛型化，不管此时定义在其中的类是不是泛型化的。在泛型方法中可以定义泛型参数，此时，参数的类型就是传入数据的类型。
* 泛型方法的格式：
* [访问权限] <泛型> 返回类型 方法名([泛型标识 参数名称]) 抛出的异常

/\*\*  
 \* 声明泛型方法  
 \*/  
public <E> E getE(E e) {  
 return e;  
}

// 当通过对象调泛型方法时，指明泛型方法的类型。  
Integer i = order.getE(34);  
Double d = order.getE(2.3);

/\*\*  
 \* 实现数组到集合的复制  
 \*/  
public <E> List<E> fromArrayToList(E[] e,List<E> list) {  
 for (E e1:e) {  
 list.add(e1);  
 }  
 return list;  
}

Integer[] in = new Integer[]{1,2,3};  
ArrayList<Integer> list2 = new ArrayList<>();  
List<Integer> list3 = order.fromArrayToList(in, list2);  
System.***out***.println(list3);