1. 泛型即参数化类型，也就是说数据类型变成了一个可变的参数，在不使用泛型的情况下，参数的数据类型都是写死了的，使用泛型之后，可以根据程序的需要进行改变。

定义泛型的规则：

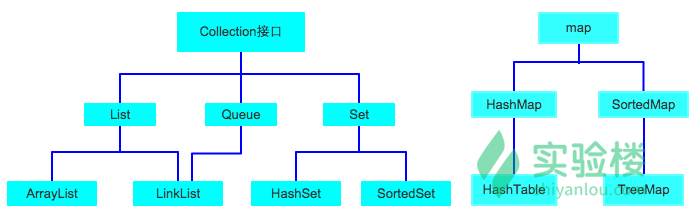
只能是类类型，不能是简单数据类型

泛型参数可以有多个

可以用使用 extends 语句或者 super 语句 如<T extends superClass>表示类型的上界，T 只能是 superClass 或其子类， <K super childClass>表示类型的下界，K 只能是 childClass 或其父类。

可以是通配符类型，比如常见的 Class<?>

1. 集合框架是为表示和操作集合而规定的一种统一的标准的体系结构。任何集合框架都包含三大内容：对外的接口、接口的实现和对集合运算的算法。





1. List 是一个接口，不能实例化，需要一个具体类来实现实例化。List 集合中的对象按照一定的顺序排放，里面的内容可以重复。



1. Map 接口也是一个非常重要的集合接口，用于存储键/值对。Map 中的元素都是成对出现的，键值对就像数组的索引与数组的内容的关系一样，将一个键映射到一个值的对象。一个映射不能包含重复的键；每个键最多只能映射到一个值。我们可以通过键去找到相应的值。

value 可以存储任意类型的对象，我们可以根据 key 键快速查找 value。Map 中的键/值对以 Entry 类型的对象实例形式存在。



1. HashMap 是基于哈希表的 Map 接口的一个重要实现类。HashMap 中的 Entry 对象是无序排列的，Key 值和 value 值都可以为 null，但是一个 HashMap 只能有一个 key 值为 null 的映射（key 值不可重复）。
2. Set 接口也是 Collection 接口的子接口，它有一个很重要也是很常用的实现类——HashSet，Set 是元素无序并且不包含重复元素的 collection（List 可以重复），被称为集。

HashSet 由哈希表（实际上是一个 HashMap 实例）支持。它不保证 set 的迭代顺序；特别是它不保证该顺序恒久不变。

1. java.util.Collections 是一个工具类，他包含了大量对集合进行操作的静态方法。



1. 插入排序

有一个已经有序的数据序列，要求在这个已经排好的数据序列中插入一个数，但要求插入后此数据序列仍然有序，这个时候就要用到一种新的排序方法——插入排序法,插入排序的基本操作就是将一个数据插入到已经排好序的有序数据中，从而得到一个新的、个数加一的有序数据，算法适用于少量数据的排序，时间复杂度为 O(n^2)。是稳定的排序方法。插入算法把要排序的数组分成两部分：第一部分包含了这个数组的所有元素，但将最后一个元素除外（让数组多一个空间才有插入的位置），而第二部分就只包含这一个元素（即待插入元素）。在第一部分排序完成后，再将这个最后元素插入到已排好序的第一部分中。

插入排序的基本思想是：每步将一个待排序的记录，按其关键码值的大小插入前面已经排序的文件中适当位置上，直到全部插入完为止。

1. 冒泡排序

冒泡排序（Bubble Sort），是一种计算机科学领域的较简单的排序算法。 它重复地走访过要排序的元素列，依次比较两个相邻的元素，如果他们的顺序（如从大到小、首字母从 A 到 Z）错误就把他们交换过来。走访元素的工作是重复地进行直到没有相邻元素需要交换，也就是说该元素已经排序完成。 这个算法的名字由来是因为越大的元素会经由交换慢慢“浮”到数列的顶端（升序或降序排列），就如同碳酸饮料中二氧化碳的气泡最终会上浮到顶端一样，故名“冒泡排序”。

1. 归并排序

归并排序（MERGE-SORT）是建立在归并操作上的一种有效的排序算法,该算法是采用分治法（Divide and Conquer）的一个非常典型的应用。将已有序的子序列合并，得到完全有序的序列；即先使每个子序列有序，再使子序列段间有序。若将两个有序表合并成一个有序表，称为二路归并。

1. 快速排序

快速排序（英语：Quicksort），又称划分交换排序（partition-exchange sort），简称快排，一种排序算法，最早由东尼·霍尔提出。在平均状况下，排序 n 个项目要 O(nlogn) 次比较。在最坏状况下则需要 O(n^2) 次比较，但这种状况并不常见。事实上，快速排序 O(nlogn)通常明显比其他算法更快，因为它的内部循环（inner loop）可以在大部分的架构上很有效率地达成。

1. 线性搜索

线性搜索或顺序搜索是一种寻找某一特定值的搜索算法，指按一定的顺序检查数组中每一个元素，直到找到所要寻找的特定值为止。是最简单的一种搜索算法。

1. 二分查找

在计算机科学中，二分搜索（英语：binary search），也称折半搜索（英语：half-interval search）、对数搜索（英语：logarithmic search），是一种在有序数组中查找某一特定元素的搜索算法。搜索过程从数组的中间元素开始，如果中间元素正好是要查找的元素，则搜索过程结束；如果某一特定元素大于或者小于中间元素，则在数组大于或小于中间元素的那一半中查找，而且跟开始一样从中间元素开始比较。如果在某一步骤数组为空，则代表找不到。这种搜索算法每一次比较都使搜索范围缩小一半。

1. 异常指不期而至的各种状况，它在程序运行的过程中发生。作为开发者，我们都希望自己写的代码永远都不会出现bug，然而现实告诉我们并没有这样的情景。如果用户在程序的使用过程中因为一些原因造成他的数据丢失，这个用户就可能不会再使用该程序了。所以，对于程序的错误以及外部环境能够对用户造成的影响，我们应当及时报告并且以适当的方式来处理这个错误。
2. 异常通常有四类：

Error：系统内部错误，这类错误由系统进行处理，程序本身无需捕获处理

Exception：可以处理的异常

RuntimeException：可以捕获，也可以不捕获的异常

继承 Exception 的其他类：必须捕获，通常在 API 文档中会说明这些方法抛出哪些异常。

* 1. Exception 异常下又主要分为两大类异常，一个是派生于 RuntimeExcption 的异常，一个是除了 RuntimeExcption 体系之外的其他异常。
  2. throw 抛出异常

当程序运行时数据出现错误或者我们不希望发生的情况出现的话，可以通过抛出异常来处理。

* 1. throws 声明异常

throws 用于声明异常，表示该方法可能会抛出的异常。如果声明的异常中包括 checked 异常（受查异常），那么调用者必须处理该异常或者使用 throws 继续向上抛出。throws 位于方法体前，多个异常使用,分割。

* 1. 通常抛出异常后，还需要将异常捕获。使用try和catch语句块来捕获异常，有时候还会用到finally。

对于上述三个关键词所构成的语句块，try语句块是必不可少的，catch和finally语句块可以根据情况选择其一或者全选。你可以把可能发生错误或出现问题的语句放到try语句块中，将异常发生后要执行的语句放到catch语句块中，而finally语句块里面放置的语句，不管异常是否发生，它们都会被执行。

1. 自定义一个异常类非常简单，只需要让它继承 Exception 或其子类就行。在自定义异常类的时候，建议同时提供无参构造方法和带字符串参数的构造方法，后者可以为你在调试时提供更加详细的信息。
2. 当异常抛出后，我们可以通过异常堆栈追踪程序的运行轨迹，以便我们更好的DEBUG。