**数组**

**数组**不是Java所特有的，几乎所有编程语言都有自己的数组实现。在Java中，数组本身就是一个class。一个数组是一个类对象

一维数组的声明方式：

type var[]; 或type[] var;

声明数组时不能指定其长度（数组中元素的个数），

Java中使用关键字new创建数组对象，格式为：  
数组名 = new 数组元素的类型 [数组元素的个数]

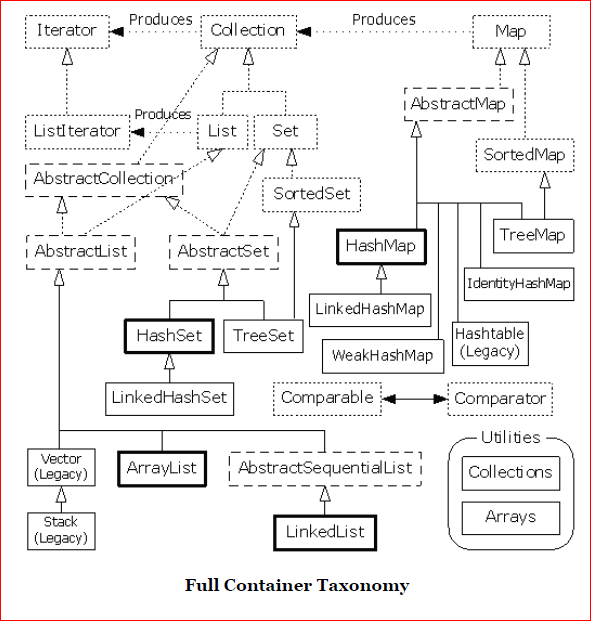
TestNew.java

初始化：

1.动态初始化：数组定义与为数组分配空间和赋值的操作分开进行；TestD.java  
2.静态初始化：在定义数组的同时就为数组元素分配空间并赋值；TestS.java  
3.默认初始化：数组是引用类型，它的元素相当于类的成员变量，因此数组分配空间后，每个元素也被按照成员变量的规则被隐式初始化。TestDefault.java

**3.3 java.util**

**3.3.1 集合**





**Collection、Collections**

Collection是个[java](https://www.baidu.com/s?wd=java&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9PWcvuWTkuWb3uyRvuHTL0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6K1TL0qnfK1TL0z5HD0IgF_5y9YIZ0lQzqlpA-bmyt8mh7GuZR8mvqVQL7dugPYpyq8Q1RvrjDLrHn1P6).util下的接口，它是各种集合结构的父接口 继承于它的接口主要有Set 和List.

Collections是个[java](https://www.baidu.com/s?wd=java&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9PWcvuWTkuWb3uyRvuHTL0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6K1TL0qnfK1TL0z5HD0IgF_5y9YIZ0lQzqlpA-bmyt8mh7GuZR8mvqVQL7dugPYpyq8Q1RvrjDLrHn1P6).util下的类，它包含有各种有关集合操作的静态方法。他提供一系列静态方法实现对各种集合的搜索、排序、[线程安全](https://www.baidu.com/s?wd=%E7%BA%BF%E7%A8%8B%E5%AE%89%E5%85%A8&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9PWcvuWTkuWb3uyRvuHTL0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6K1TL0qnfK1TL0z5HD0IgF_5y9YIZ0lQzqlpA-bmyt8mh7GuZR8mvqVQL7dugPYpyq8Q1RvrjDLrHn1P6)化等操作。

**Array、Arrays**

Array类主要提供了动态创建和访问 [Java](https://www.baidu.com/s?wd=Java&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9PWcvuWTkuWb3uyRvuHTL0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6K1TL0qnfK1TL0z5HD0IgF_5y9YIZ0lQzqlpA-bmyt8mh7GuZR8mvqVQL7dugPYpyq8Q1RvrjDLrHn1P6) 数组的方法。

Arrays包含用来操作数组（比如排序和搜索）的各种方法。此类还包含一个允许将数组作为列表来查看的静态工厂。

List：有序，可重复的集合。

Set：无序、不可重复的集合。

Map：具有映射关系的集合。

**List**

ArrayList(数组) LinkedList(双向链表)

**Set**

HashSet (无序不可重复 底层HashMap)

LinkedHashSet

**Map**

Hashtable (线程安全 效率低 父- Dictionary 键值对不可以为null)

HashMap (线程不安全 效率高 父- AbstractMap 键最多一个null 值可以多个null)

Properties (Hashtable子类 读写资源文件 键值只能是字符串)

**Queue、Deque**

Deque接口继承了Queue接口,而Queue接口继承了Collection接口,  
LinkedList实现了Deque接口

(顶级接口)Collection-->Queue-->Deque-->LinkedList(实现类)





**泛型**

泛型（Generic type 或者 generics）是对java 语言的类型系统的一种扩展，以支持创建可以按类型进行参数化的类。可以把类型参数看作是使用参数化类型时指定的类型的一个占位符，就像方法的形式参数是运行时传递的值的占位符一样。

可以在集合框架（Collection framework）中看到泛型的动机。例如，Map 类允许您向一个 Map 添加任意类的对象，即使最常见的情况是在给定映射（map）中保存某个特定类型（比如 String）的对象。

因为 Map.get() 被定义为返回 Object，所以一般必须将 Map.get() 的结果强制类型转换为期望的类型，如下面的代码所示：

Map m = new HashMap();  
m.put("key", "blarg");  
String s = (String) m.get("key");

要让程序通过编译，必须将 get() 的结果强制类型转换为 String，并且希望结果真的是一个 String。但是有可能某人已经在该映射中保存了不是 String 的东西，这样的话，上面的代码将会抛出 ClassCastException。

理想情况下，您可能会得出这样一个观点，即 m 是一个 Map，它将 String 键映射到 String 值。这可以让您消除代码中的强制类型转换，同时获得一个附加的类型检查层，该检查层可以防止有人将错误类型的键或值保存在集合中。这就是泛型所做的工作

Map<String,String> m = **new** HashMap<String,String>();

m.put("key", "hello");

String s = m.get("key");

有两种方式可以进行集合排序 ：

1. 集合中对象的所属类实现了 java.lang.Comparable 接口

2. 为集合指定比较器 java.lang.Comparator 的实现类

comparable 是通用的接口，用户可以实现它来完成自己特定的比较，而 comparator 可以看成一种算法的实现，在需要容器集合 collection 需要比较功能的时候，来指定这个比较器，这可以看出一种设计模式，将算法和数据分离。

前者应该比较固定，和一个具体类相绑定，而后者比较灵活，它可以被用于各个需要比较功能的类使用。可以说前者属于“静态绑定”，而后者可以“动态绑定”。

一个类实现了 Camparable 接口表明这个类的对象之间是可以相互比较的。如果用数学语言描述的话就是这个类的对象组成的集合中存在一个全序。这样，这个类对象组成的集合就可以使用 Sort 方法排序了。

而 Comparator 的作用有两个：

　　 1. 如果类的设计师没有考虑到 Compare 的问题而没有实现 Comparable 接口，可以通过　 Comparator 来实现比较算法进行排序

2. 为了使用不同的排序标准做准备，比如：升序、降序或其他什么序

**3.3.2 日期时间**

**Date**

**DateFormate SimpleDateFormate**

**Calendar**