集合类型	描述
ArrayList	可以动态增长和缩减的一个索引序列
LinkedList	可以在任何位置高效插入和删除的一个有序序列
ArrayDeque	实现为循环数组的一个双端队列
HashSet	没有重复元素的一个无序集合
TreeSet	一个有序集
EnumSet	一个包含枚举类型值的集
LinkedHashSet	一个可以记住元素插入次序的集
PriorityQueue	允许高效删除最小元素的一个集合
HashMap	存储键/值关联的一个数据结构
TreeMap	键有序的一个映射
EnumMap	键属于枚举类型的一个映射
LinkedHashMap	可以记住键/值项添加次序的一个映射
WeakHashMap	值不会在别处使用时就可以被垃圾回收的一个映射
IdentityHashMap	用==而不是equals比较键的一个映射

链表List

- 可以重复
- 记录放入顺序
- ArrayList、 LinkedList 和 Vector 三个实现类

在Java中，所有链表实际上都是双向链接的——即每个链接还存放着其前驱的引用。

LinkedList类可以使用ListIterator类从前后两个方向遍历链表中的元素，以及删除和添加元素。

链表不支持快速随机访问。如果要查看链表中的第n个元素，就必须从头开始，越过n-1个元素。鉴于这个原因，当需要按整数索引访问元素时，通常不选用链表。

避免使用以整数索引表示链表中位置的所有方法。如果需要对集合进行随机访问，就使用数组或ArrayList，而不要使用链表

List 接口的一些方法

在链表末尾添加一个元素

```
boolean add(E e)
```

在指定位置添加一个元素

```
void add(int index, E element)
```

将一个集合添加到链表的末尾

```
boolean addAll(Collection<? extends E> c)
```

将一个集合添加到链表的指定位置

```
boolean addAll(int index, Collection<? extends E> c)
```

清除所有元素

```
void clear()
```

检索链表是否含有某个对象

```
boolean contains(Object o)
```

判断链表是否包含指定集合的全部元素

```
boolean containsAll(Collection<?> c)
```

void	add(int index, E element)	Inserts the specified element at the specified position in this list (optional operation).
boolean	add(E e)	Appends the specified element to the end of this list (optional operation).
boolean	addAll(int index, Collection<? extends E> c)	Inserts all of the elements in the specified collection into this list at the specified position (optional operation).
boolean	addAll(Collection<? extends E> c)	Appends all of the elements in the specified collection to the end of this list, in the order that they are returned by the specified collection's iterator (optional operation).
void	clear()	Removes all of the elements from this list (optional operation).
boolean	contains(Object o)	Returns true if this list contains the specified element.
boolean	containsAll(Collection<?> c)	Returns true if this list contains all of the elements of the specified collection.
static <E> List<E>	copyOf(Collection<? extends E> coll)	Returns an unmodifiable List containing the elements of the given Collection, in its iteration order.
boolean	equals(Object o)	Compares the specified object with this list for equality.
E	get(int index)	Returns the element at the specified position in this list.
int	hashCode()	Returns the hash code value for this list.
int	indexOf(Object o)	Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.
boolean	isEmpty()	Returns true if this list contains no elements.
Iterator<E>	iterator()	Returns an iterator over the elements in this list in proper sequence.
int	lastIndexOf(Object o)	Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.
ListIterator<E>	listIterator()	Returns a list iterator over the elements in this list (in proper sequence).
ListIterator<E>	listIterator(int index)	Returns a list iterator over the elements in this list (in proper sequence), starting at the specified position in the list.

ArrayList

就是一个动态数组

LinkedList

散列集

> 如果不在意元素的顺序，有几种能够快速查找元素的数据结构。其缺点是无法控制元素出现的次序。这些数据结构按照对自己最方便的方式组织元素

散列表 是一种用于快速查找对象的数据结构。散列表为每个对象计算一个整数，称为散列码（hash code）。散列码是由对象的实例字段得出的一个整数。更准确地说，有不同数据的对象将产生不同的散列码。

如果你定义你自己的类，你就要负责实现自己的hashCode方法。你的实现应该与equals方法兼容，即如果a.equals(b)为true，a与b必须有相同的散列码。

在Java中，散列表用链表数组实现。每个列表被称为桶（bucket）。当桶已经被填充时，这种现象称为散列冲突，这时需要将新对象与桶中的所有对象进行比较，查看这个对象是否已经存在，如果散列码合理地随机分布，桶的数目也足够大，需要比较的次数就会很少。

> 在Java 8中，桶满时会从链表变为平衡二叉树

映射Map

类型介绍

> Map 接口存储一组键值对象，提供key（键）到value（值）的映射
>
> **存储的是一对一对的键值对**

Java 自带了各种 Map 类。这些 Map 类可归为三种类型：

1. 通用Map，用于在应用程序中管理映射，通常在 java.util 程序包中实现

```
HashMap、Hashtable、Properties、LinkedHashMap、
IdentityHashMap、TreeMap、WeakHashMap、ConcurrentHashMap
```

2. 专用Map，通常我们不必亲自创建此类Map，而是通过某些其他类对其进行访问

```
java.util.jar.Attributes、
javax.print.attribute.standard.PrinterStateReasons、
java.security.Provider、java.awt.RenderingHints、
javax.swing.UIDefaults
```

3. 一个用于帮助我们实现自己的Map类的抽象类

```
AbstractMap
```

类型区别

HashMap

最常用的Map,它根据键的HashCode 值存储数据,根据键可以直接获取它的值，**具有很快的访问速度。**HashMap最多只允许一条记录的键为Null(多条会覆盖);允许多条记录的值为 Null。非同步的。

TreeMap

能够把它保存的记录根据键(key)排序,**默认是按升序排序**，也可以指定排序的比较器，当用Iterator 遍历TreeMap时，得到的记录是排过序的。TreeMap不允许key的值为null。非同步的。

Hashtable

与 HashMap类似,不同的是: key和value的值均不允许为null;它支持线程的同步，即任一时刻只有一个线程能写Hashtable,因此也导致了Hashtale在写入时会比较慢。

LinkedHashMap

保存了记录的插入顺序，在用Iterator遍历LinkedHashMap时，先得到的记录肯定是先插入的.在遍历的时候会比HashMap慢。key和value均允许为空，非同步的。

HashMap

```
    Map<String, String> map = new HashMap<String, String>();
    map.put("key1", "1");  // 插入键值对
    String str = map.get("key1");  // 获取键值对
```