1. Java 集合框架：

如果并不知道程序运行时会需要多少对象，或者需要更复杂方式存储对象——可以使用 Java 集合框架

集合框架：java 的集合框架为程序提供了一种处理对象组的标准方式。集合框架包含的三个组件：

接口：是表示集合的抽象数据类型；

算法：是对实现接口的对象执行计算的方法；

实现：是接口的实际实现。

任何对象加入集合类后，自动转变为 Object 类型；取出时，需要进行强制类型转换，恢复为特定的类型.

2. 接口和实现类之间的关系： //更详细的参看 API 手册

3. 集合接口的特点：

集合就是将若干用途、性质相同或相近的“数据”组合而成一个整体。

Collection 接口存储一组不唯一，无序的对象

从体系上讲，集合类型可以归纳为三种：

集（Set）

Set 集合中不区分元素的顺序，不允许出现重复元素。存储一组唯一，无序的对象

列表（List）

List 集合区分元素的顺序，且允许包含重复元素。存储一组不唯一，有序（插入顺序）的对象

映射（Map）

映射中保存成对的“键-值”（Key-Value）信息，映射中不能包含重复的键，每个键最多只能映射一个值。存储一组键值对象，提供 key 到 value 的映射

Java 集合中只能保存引用类型的数据，实际上存放的是对象的引用而非对象本身。

```
List<String> list2 = list.subList(2, 6); // 修改 list2 会影响 list
```

4. 集合类构造方法

集合框架实现类应该提供两个“标准”构造方法，尽管无法强制执行此约定(因为接口不能包含构造方法)，但是 Java 平台库中所有通用的 Collection 实现都遵从它。

A. 无参数构造方法，用于创建空集合

B. 有参数构造方法，用于复制集合

5. Collection 接口： //详细方法参见 API 手册

java.util.Collection<E>

public interface Collection<E> extends Iterable<E>

Collection 接口位于集合框架的顶部。描述 Set 和 List 集合类型的根接口，其中定义了有关集合操作的普遍性方法。

描述	方法

添加元素	boolean add(Object o) / boolean addAll(Collection c)
删除元素	boolean remove(Object o) / boolean removeAll(Collection c)
	boolean retainAll(Collection c)
包含判断	boolean contains(Object o) / boolean containsAll(Collection c)
迭代输出	Iterator iterator()
等价判断	boolean equals(Object o)
返回大小	int size()
清除元素	void clear()
为空判断	boolean isEmpty()
返回数组	Object[] toArray()
获取哈希值	int hashCode()

6.List 接口：

public interface List<E> extends Collection<E>

用于**存储对象的有序列表，列表中可包含相同元素**。

特有方法列表：

描述	方法
添加元素	boolean add(int index, Object o) / boolean addAll(int index ,Collection c)
删除元素	Object remove(int index)

替换元素	Object set(int index, Object element)
截取列表	List subList(int fromIndex, int toIndex)
获取元素	Object get(int index)
获取索引	int indexOf(Object o) / int lastIndexOf(Object o)

7.List 的常见实现类：ArrayList、LinkedList

Java.util.ArrayList

public class **ArrayList<E>** extends AbstractList<E> implements List<E>, RandomAccess, Cloneable, Serializable

用来定义长度可变的对象引用数组，即动态数组。非同步类

ArrayList 实现了长度可变的数组，在内存中分配连续的空间。遍历元素和随机访问元素的效率比较高；

构造方法
public ArrayList()
构造一个初始容量为 10 的空列表
public ArrayList(int initialCapacity)
构造一个具有指定初始容量的空列表
public ArrayList(Collection c)
构造一个包含指定 collection 的元素的列表，元素是按照该 collection 迭代器返回的顺序排列

Java.util.LinkedList

public class **LinkedList<E>** extends AbstractSequentialList<E> implements List<E>, Deque<E>, Cloneable, Serializable

用来定义长度可变的对象引用链表结构，即动态链表。非同步类

LinkedList 采用链表存储方式。插入、删除元素时效率比较高

LinkedList 的特殊方法：

	说 明
void addFirst(Object o)	在列表的首部添加元素
void addLast(Object o)	在列表的末尾添加元素
Object getFirst()	返回列表中的第一个元素
Object getLast()	返回列表中的最后一个元素
Object removeFirst()	删除并返回列表中的第一个元素
Object removeLast()	删除并返回列表中的最后一个元素

8. Set 接口： public interface Set<E> extends Collection<E>

9. Set 接口的常见的实现类：HashSet, TreeSet

Java.util.HashSet

public class HashSet<E> extends AbstractSet<E> implements Set<E>, Cloneable, Serializable

java.util.HashSet 类实现了 java.util.Set 接口，描述典型的 Set 集合结构。非同步类

注意事项

A. HashSet 中不允许出现重复元素，不保证集合中元素的顺序

B. HashSet 中允许包含值为 null 的元素，但最多只能有一个 null 元素

方法列表

构造方法
public HashSet()
构造一个新的空 set，默认初始容量是 16，加载因子是 0.75
public HashSet(int initialCapacity)
构造一个新的空 set，具有指定的初始容量和默认的加载因子 0.75
public HashSet(int initialCapacity, float loadFactor)
构造一个新的空 set，具有指定的初始容量和指定的加载因子
public HashSet(Collection c)
构造一个包含指定 collection 中的元素的新 set。使用默认的加载因子 0.75 和足以包含指定 collection 中所有元素的初始容量来创建 HashSet

Java.util.TreeSet

public class TreeSet<E> extends AbstractSet<E> implements NavigableSet<E>, Cloneable, Serializable

java.util.TreeSet 类也实现了 java.util.Set，它描述的是 Set 的一种变体——可以实现排序功能的集合。非同步类

在将对象元素添加到 TreeSet 集中时会自动按照某种比较规则将其插入到有序的对象序列中，以保证集合元素组成的对象序列时刻按照“升序”排列

方法列表

构造方法
public TreeSet()
造一个新的空 set，该 set 根据其元素的自然顺序进行排序。插入的元素都必须实现 Comparable 接口
public TreeSet(Comparator comparator)
构造一个新的空 TreeSet，根据指定比较器进行排序。插入的元素必须能够由该比较器进行相互比较
public TreeSet(Collection c)
构造一个包含指定 collection 元素的新 TreeSet，它按照其元素的自然顺序进行排序。插入该 set 的所有元素都必须实现 Comparable 接口。
public TreeSet(SortedSet s)
构造一个与指定有序 set 具有相同映射关系和相同排序的新 TreeSet

10. Map 接口： public interface Map<K,V>

专门处理键值映射数据的存储，可以根据键实现对值的操作。

Map 定义了存储键/值对映射的接口，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。

方法列表：

描述	方法
添加元素	Object put(Object key, Object value) / void putAll(Map m)
获取键值	Object get(Object key)
删除元素	Object remove(Object key)
包含判断	boolean containsKey(Object key) / boolean containsValue(Object value)
返回大小	int size()
清除元素	void clear()
为空判断	boolean isEmpty()
获取哈希值	int hashCode()
获取 key 的 set 视图	Set keySet()
获取 value 的 collection 视图	Collection values()
获取映射关系的 set 视图	Set<Map.Entry<K,V>> entrySet()

11. Map 接口常用的实现类：HashMap, TreeMap

java.util.HashMap

public class HashMap<K,V> extends AbstractMap<K,V> implements Map<K,V>, Cloneable, Serializable

该类基于哈希表实现了前述的映射集合结构。非同步类

注意事项

A. HashMap 结构不保证其中元素（映射信息）的先后顺序，并且允许使用 null“值”和 null“键”

B. 当集合中不存在当前检索的“键”所对应的映射值时，HashMap 的 get()方法会返回空值 null，而不会运行出错

C. 影响 HashMap 性能的参数：初始容量 Initial Capacity、加载因子 Load Factor

方法列表

构造方法	
public HashMap()	构造一个初始容量 16 和加载因子 0.75 的空 HashMap
public HashMap(int initialCapacity)	构造一个带指定初始容量和默认加载因子的空 HashMap
public HashMap(int initialCapacity, float loadFactor)	构造一个带指定初始容量和加载因子的空 HashMap
public HashMap(Map m)	构造一个映射关系与指定 Map 相同的新 HashMap

java.util.TreeMap

public class TreeMap<K,V> extends AbstractMap<K,V> implements NavigableMap<K,V>, Cloneable, Serializable

Java.util.TreeMap 类实现了将 Map 映射中的元素按照“键”进行升序排列的功能。非同步类

注意事项

A. 其排序规则可以是默认的按照“键”的自然顺序排列，也可以使用指定的其他排序规则.

B. 向 TreeMap 映射中添加的元素“键”所属的类必须实现 Comparable 接口.

方法列表

构造方法
public TreeMap()
使用键的自然顺序构造一个新的、空的树映射。插入该映射的所有键都必须实现 Comparable 接口
public TreeMap(Comparator comparator)
构造一个新的、空的树映射，该映射根据给定比较器进行排序。插入该映射的键都必须由给定比较器进行相互比较
public TreeMap(Map m)
构造一个与给定映射具有相同映射关系的新的树映射，该映射根据其键的自然顺序 进行排序。插入此新映射的所有键都必须实现 Comparable 接口
public TreeMap(SortedMap m)
构造一个与指定有序映射具有相同映射关系和相同排序顺序的新的树映射

12. java.lang.Iterable/java.util.Iterator

java.lang.Iterable

实现这个接口允许使用“foreach” 循环、使用迭代器。

Collection 接口继承了 Iterable 接口.

方法列表

描述	方法
返回迭代器	Iterator iterator()

java.util.Iterator

对 collection 进行迭代的迭代器接口，运行迭代输出集合中的元素.

方法列表

描述	方法
判断	boolean hasNext() //判断是否存在另一个可访问的元素
返回	Object next() //返回要访问的下一个元素
移除	void remove()

13. java.lang.Comparable

实现这个接口的类，允许对该类的对象进行比较，通过这种比较后的排序称为自然排序，类的 compareTo 方法被称为它的自然比较方法。

用途

A. 实现此接口的对象列表(数组)可以通过 Collections.sort(Arrays.sort)进行自动排序

B. 实现此接口的对象可以用作有序映射中的键或有序集合中的元素，无需指定比较器

注意事项

用户在重写 compareTo()方法以定制比较逻辑时， 需要确保其与等价性判断方法 equals()保持一致.

方法列表

描述	方法
排序方法	int compareTo(Object o)
比较此对象与指定对象。如果该对象小于、等于或大于指定对象，分别返回负整数、零或正整数。	

14. java.util.Comparator

强行对某个对象列表(数组)进行整体排序的比较器

用途

1、可以将 Comparator 传递给 Collections.sort 或 Arrays.sort，从而允许在排序顺序上实现精确控制。

2、可以使用 Comparator 来控制某些数据结构(如有序 set 或有序映射)的顺序，或者为那些没有自然顺序的对象列表(数组)提供排序.

注意事项

用户在重写 compare(Object o1,Object o2)方法以定制比较逻辑时， 需要确保其与被比较对象的等价性判断方法 equals()保持一致

方法列表

描述	方法
大小比较	int compare(Object o1, Object o2)

比较用来排序的两个参数。根据参数 1 小于、等于或大于参数 2 分别返回负整数、零或正整数。	
等价判断	boolean equals(Object obj)
判断某个对象是否为一个 Comparator 且与当前的比较器使用相同的比较规则	

15. java.util.Collections

集合框架的工具类 Collections，非常类似数组的 Arrays 工具类。

java.util.Collections 类中定义了多种集合操作方法，实现了对集合元素的排序、取极值、批量拷贝、集合结构转换、循环移位以及匹配性检查等功能

此类完全由在 Collection 上进行操作或返回 Collection 的静态方法组成.

方法列表

public static void sort(List list)
public static void reverse(List list)
public static void shuffle(List list)
public static void copy(List dest, List src)
public static ArrayList list(Enumeration e)
public static int frequency(Collection c, Object o)
public static T max(Collection coll)
public static T min(Collection coll)

16. Vector 和 ArrayList 的异同

实现原理、功能相同，可以互用

主要区别

Vector 线程安全，ArrayList 重速度轻安全，线程非安全

长度需增长时，Vector 默认增长一倍，ArrayList 增长 50%

Hashtable 和 HashMap 的异同

实现原理、功能相同，可以互用

主要区别

Hashtable 继承 Dictionary 类，HashMap 实现 Map 接口

Hashtable 线程安全，HashMap 线程非安全

Hashtable 不允许 null 值，HashMap 允许 null 值