java.util.CollectionJava认证考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/644/2021_2022_javautil_c104_644930.htm Collection 接口是一组允许重复的对象。 Set 接口继承 Collection，但不允许重复，使用自己内部的一个排列机制。 List 接口继承 Collection，允许重复，以元素安插的次序来放置元素，不会重新排列。 Map接口是一组成对的键－值对象，即所持有的是key-value pairs。Map中不能有重复的key。拥有自己的内部排列机制。 容器中的元素类型都为Object。从容器取得元素时，必须把它转换成原来的类型。 集合接口 1.Collection 接口 用于表示任何对象或元素组。想要尽可能以常规方式处理一组元素时，就使用这一接口。 (1) 单元素添加、删除操作： boolean add(Object o):将对象添加给集合 boolean remove(Object o): 如果集合中有与o相匹配的对象，则删除对象o (2) 查询操作： int size() ：返回当前集合中元素的数量 boolean isEmpty() ：判断集合中是否有任何元素 boolean contains(Object o) ：查找集合中是否含有对象o Iterator iterator() ：返回一个迭代器，用来访问集合中的各个元素 (3) 组操作 ：作用于元素组或整个集合 boolean containsAll(Collection c): 查找集合中是否含有集合c 中所有元素 boolean addAll(Collection c) : 将集合c 中所有元素添加给该集合 void clear(): 删除集合中所有元素 void removeAll(Collection c) : 从集合中删除集合c 中的所有元素 void retainAll(Collection c) : 从集合中删除集合c 中不包含的元素 (4) Collection转换为Object数组 ： Object[] toArray() ：返回一个内含集合所有元素的array Object[] toArray(Object[] a) ：

返回一个内含集合所有元素的array。运行期返回的array和参数a的型别相同，需要转换为正确型别。 此外，您还可以把集合转换成其它任何其它的对象数组。但是，您不能直接把集合转换成基本数据类型的数组，因为集合必须持有对象。“斜体接口方法是可选的。因为一个接口实现必须实现所有接口方法，调用程序就需要一种途径来知道一个可选的方法是不是不受支持。如果调用一种可选方法时，一个 UnsupportedOperationException 被抛出，则操作失败，因为方法不受支持。此异常类继承 RuntimeException 类，避免了将所有集合操作放入 try-catch 块。 ”Collection不提供get()方法。如果要遍历Collectin中的元素，就必须用Iterator。
1.1.AbstractCollection 抽象类 AbstractCollection 类提供具体“集合框架”类的基本功能。虽然您可以自行实现 Collection 接口的所有方法，但是，除了iterator()和size()方法在恰当的子类中实现以外，其它所有方法都由 AbstractCollection 类来提供实现。如果子类不覆盖某些方法，可选的如add()之类的方法将抛出异常。 1.2.Iterator 接口 Collection 接口的iterator()方法返回一个 Iterator。Iterator接口方法能以迭代方式逐个访问集合中各个元素，并安全的从Collection 中除去适当的元素。
(1) boolean hasNext(): 判断是否存在另一个可访问的元素
Object next(): 返回要访问的下一个元素。如果到达集合结尾，则抛出NoSuchElementException异常。 (2) void remove(): 删除上次访问返回的对象。本方法必须紧跟在一个元素的访问后执行。如果上次访问后集合已被修改，方法将抛出IllegalStateException。“ Iterator中删除操作对底层Collection也有影响。 ”迭代器是 故障快速修复（fail-fast）的。这意味

着，当另一个线程修改底层集合的时候，如果您正在用 Iterator 遍历集合，那么，Iterator就会抛出 ConcurrentModificationException （另一种 RuntimeException异常）异常并立刻失败 2.List接口 List 接口继承了 Collection 接口以定义一个允许重复项的有序集合。该接口不但能够对列表的一部分进行处理，还添加了面向位置的操作。 (1) 面向位置的操作包括插入某个元素或 Collection 的功能，还包括获取、除去或更改元素的功能。在 List 中搜索元素可以从列表的头部或尾部开始，如果找到元素，还将报告元素所在的位置 : void add(int index, Object element): 在指定位置index上添加元素element boolean addAll(int index, Collection c): 将集合c的所有元素添加到指定位置index Object get(int index): 返回List中指定位置的元素 int indexOf(Object o): 返回第一个出现元素o的位置，否则返回-1 int lastIndexOf(Object o) ：返回最后一个出现元素o的位置，否则返回-1 Object remove(int index) ：删除指定位置上的元素 Object set(int index, Object element) ：用元素element取代位置index上的元素，并且返回旧的元素 (2) List 接口不但以位置序列迭代的遍历整个列表，还能处理集合的子集： ListIterator listIterator() : 返回一个列表迭代器，用来访问列表中的元素 ListIterator listIterator(int index) : 返回一个列表迭代器，用来从指定位置index开始访问列表中的元素 List subList(int fromIndex, int toIndex) ：返回从指定位置fromIndex（包含）到toIndex（不包含）范围中各个元素的列表视图“对子列表的更改（如 add()、remove() 和 set() 调用）对底层 List 也有影响。 ”2.1.ListIterator接口 ListIterator 接口继承 Iterator 接口以支持添加或更改底层集合中的元素，还支持双

向访问。ListIterator没有当前位置，光标位于调用previous和next方法返回的值之间。一个长度为n的列表，有n 1个有效索引值： (1) void add(Object o): 将对象o添加到当前位置的前面 void set(Object o): 用对象o替代next或previous方法访问的上一个元素。如果上次调用后列表结构被修改了，那么将抛出IllegalStateException异常。 (2) boolean hasPrevious(): 判断向后迭代时是否有元素可访问 Object previous()：返回上一个对象 int nextIndex(): 返回下次调用next方法时将返回的元素的索引 int previousIndex(): 返回下次调用previous方法时将返回的元素的索引 