List

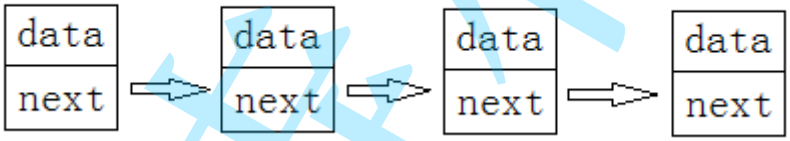
本节目标

> - 从理论上理解线性表和顺序表是什么，核心操作的流程
> - 会熟练使用 Java 中提供的 List 和 ArrayList
> - 可以独立实现一份自己的 ArrayList
> - 可以利用 List 和 ArrayList 完成简单的 OJ 笔试题

1. 认识线性表和顺序表

线性表 *(linear list)* 是n个具有相同特性的数据元素的有限序列。 线性表是一种在实际中广泛使用的数据结构，常见的线性表：顺序表、链表、栈、队列、字符串...

线性表在逻辑上是线性结构，也就说是连续的一条直线。但是在物理结构上并不一定是连续的，线性表在物理上存储时，通常以数组和链式结构的形式存储。



本节课我们重点研究线性表和顺序表；研究它的增删查改和遍历操作。

注意： 顺序表中一定要区分两个概念 **容量(capacity)** vs **元素个数(size)**；线性表的所有下标只和元素个数相关，和容量无关。

2. Java 中的 List 和 ArrayList

Collection：元素集合

List: 线性表

ArrayList: 顺序表

Collection 的官方文档

List 的官方文档

ArrayList 的官方文档

```
// 为了便于同学们理解，以下代码并不完全符合语法规则，并且去掉了泛型的语法

package java.util;

import java.util.Collection;
```

```java
import java.util.Comparator;
import java.util.Iterator;
import java.util.ListIterator;

/**
 * 线性结构
 * 特点：
 * 1. 元素和元素之间有前后关系
 * 2. 元素会有在第几个位置的概念，位置通过下标（index）表示，从 0 开始
 * 3. 插入可以根据位置的不同，分为：头插、尾插、按位置插入
 * 4. 删除可以根据位置的不同，分为：头删、尾删、按位置删除
 * 5. 遍历可以分为从前往后遍历和从后往前遍历
 * 6. Java 中，List 是一个接口，并且是 Collection 的子接口
 */
public interface List extends Collection {
    /**
     * 将 e 尾插到线性表中
     * @参数 e 待插入的元素
     * @返回值 一定是 true，表示插入成功。线性表是不会出现插入不成功的情况的
     */
    boolean add(元素类型 e);

    /**
     * 将 e 插入到线性表的 index 位置处；要求原来 index 及之后的元素全部向后移动
     * index 的可选范围是 0 <= index <= size()
     * @参数 index 插入位置（下标）
     * @参数 待插入的元素
     */
    void add(int index, 元素类型 e);

    /**
     * 删除 index 位置的元素，并返回该元素；要求 原来 index + 1 及之后元素全部向前移动
     * index 的可选范围是 0 <= index < size()
     * @参数 index 待删除位置（下标）
     * @返回值 被删除掉的元素
     */
    元素类型 remove(int index);

    /**
     * 删除从前往后遍历时，遇到的第一个相等的（equals）元素
     * @参数 待删除元素
     * @返回值 true：删除成功； false：没有找到相等的元素
     */
    boolean remove(元素类型 e);

    /**
     * 返回 index 位置的元素
     * index 的可选范围是 0 <= index < size()
     * @参数 index 获取元素的位置（下标）
     * @返回值 获取到的元素
     */
    元素类型 get(int index);

    /**
     * 用新的元素 e 替换 index 位置的元素，并返回 index 位置的原来的元素
     * index 的可选范围是 0 <= index < size()
     * @参数 index 待替换元素的位置（下标）
```

```
63          * @参数 e 要替换的新元素
64          * @返回值 index 位置的老元素
65          */
66         元素类型 set(int index, 元素类型 e);
67
68         /**
69          * 通过遍历的方式，判断与元素 e 相等（equals）的元素是否存在于线性表中
70          * @参数 e 待查找元素
71          * @返回 true：包含；false：不包含
72          */
73         boolean contains(元素类型 e);
74
75         /**
76          * 按照从前往后遍历的方式，找到第一个与元素 e 相等（equals）的元素的下标
77          * @param e 待查找元素
78          * @return >= 0 表示找到并且返回下标；-1 代表没有找到
79          */
80         int indexOf(元素类型 e);
81
82         /**
83          * 按照从后往前遍历的方式，找到第一个与元素 e 相等（equals）的元素的下标
84          * @param e 待查找元素
85          * @return >= 0 表示找到并且返回下标；-1 代表没有找到
86          */
87         int lastIndexOf(元素类型 e);
88
89         /**
90          * 清空线性表，也就是，调用 clear() 后，线性表的 size() == 0；isEmpty() ==
    true
91          */
92         void clear();
93
94         /**
95          * 返回线性表中已有元素的个数
96          * @return 返回元数个数
97          */
98         int size();
99
100        /**
101         * 返回线性表是不是一个空的容器
102         * @return true 为空容器
103         */
104        boolean isEmpty();
105
106        ///// 以下的使用频率略低
107        Iterator iterator();
108        void sort(Comparator 比较器);
109        List subList(int fromIndex, int toIndex);
110
111        ///// 以下的方法，了解即可
112        boolean addAll(Collection 集合);
113        boolean addAll(int index, Collection 集合);
114        boolean containsAll(Collection 集合);
115        boolean removeAll(Collection 集合);
116        boolean retainAll(Collection 集合);
117
118        Object[] toArray();
119    }
```

```
package java.util;

public interface ArrayList implements List {
    public ArrayList() { ... };

    public ArrayList(Collection 集合) { ... };

    public ArrayList(int 初识容量) { ... };
}
```

代码演示重要方法的使用。

2.1 扑克牌游戏

实现一副扑克牌的洗牌 + 发牌功能

TODO：利用查找能力，实现《抓小鬼》游戏

2.2 迭代能力(Iterable) 和 迭代器(Iterator)

每种容器(Collection)都是具备迭代能力(Iterable)的。所以，每种容器都自带一个方法，返回一个合适的迭代器(Iterator)以对容器进行无视实现差别的迭代。

```
package java.util;

/**
 * 迭代器，提供无视具体实现，遍历容器中每个元素的能力
 */
public interface Iterator {
    boolean hasNext();
    元素类型 next();
    void remove();
}


/**
 * 具备迭代的能力
 */
public interface Iterable {
    Iterator iterator();
}

/**
 * 每种容器都具备迭代的能力
 */
public interface Collection extends Iterable {
    ...
}
```

使用方式非常简单

```
ArrayList<String> list = new ArrayList<>();
list.add("你好");
list.add("中国");

```

```java
Iterator<String> it = list.iterator();

// 只要还有下一个元素，循环就继续
while (it.hasNext()) {
    // 获取下一个元素，并让迭代器走到之后的位置上
    String e = it.next();

    // 可以在合适的条件下，对当前迭代器正处于的元素进行删除
    it.remove();
}
```

3. 通过 OJ 题进行 List 的使用训练

存在连续三个奇数的数组

杨辉三角

4. 自己实现一个 ArrayList

```java
// 为了更接近真实的 java.util.ArrayList，我们使用 String 这种引用类型来做为元素类型
// 同时请回答每个方法的时间复杂度是什么。

class MyArrayList {
    private String[] array;
    private int size;

    boolean add(String e);

    void add(int index, String e);

    String remove(int index);

    boolean remove(String e);

    String get(int index);

    String set(int index, String e);

    boolean contains(String e);

    int indexOf(String e);

    int lastIndexOf(String e);

    void clear();

    int size();

    boolean isEmpty();
}
```