1. 数据结构
   1. 常见的数据结构

数据存储的常用结构：栈、队列、数组、链表和红黑树。

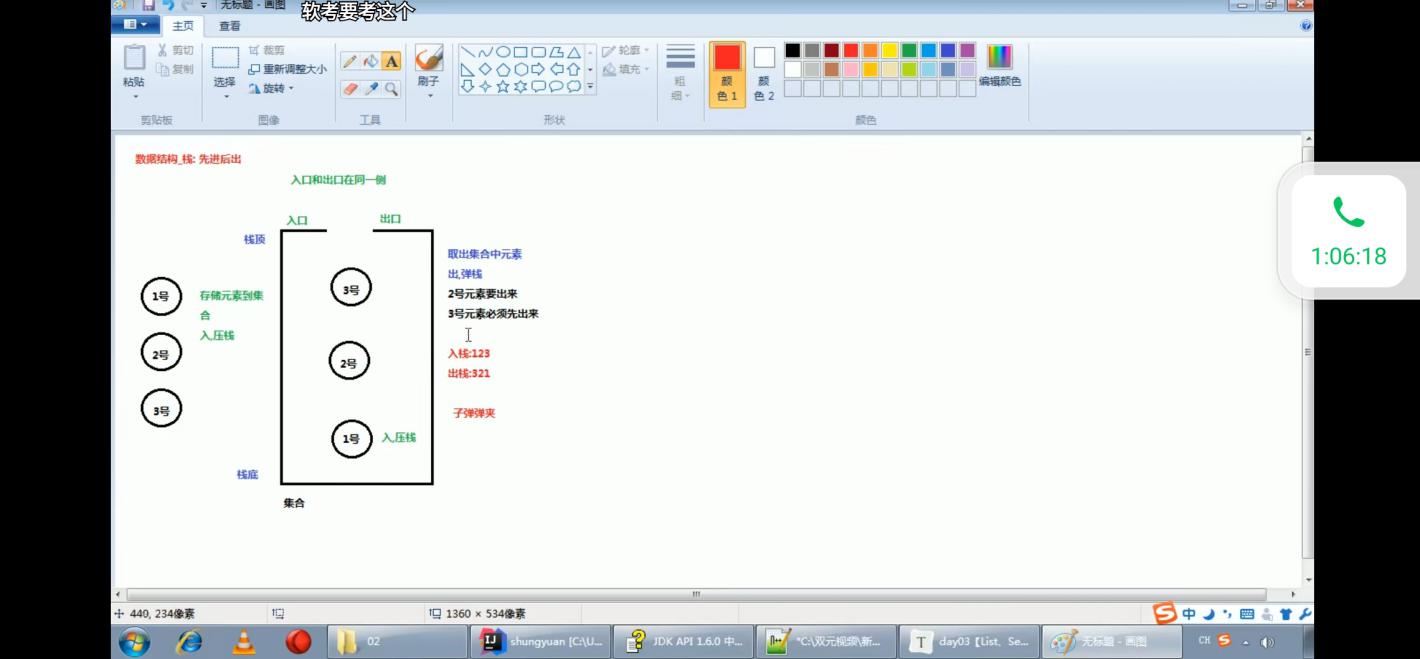
1.2 栈

栈：stack，又称堆栈，它是运算受限的线性表，其限制时仅允许在标的一端进行插入和删除操作，不允许在其他任何位置进行添加、查找、删除等操作。

简单的说：采用该结构的集合，对元素的存取有如下特点：

· 先进后出（即，存进去的元素，要在后它后面的元素依次取出后，才能取出该元素）。例如：子弹压进弹夹，先压进去的子弹在下面，后进去的子弹在上面，当开枪时，先弹出上面的子弹，然后才能弹出下面的子弹。

· 栈的入口、出口的都是栈的顶端位置。



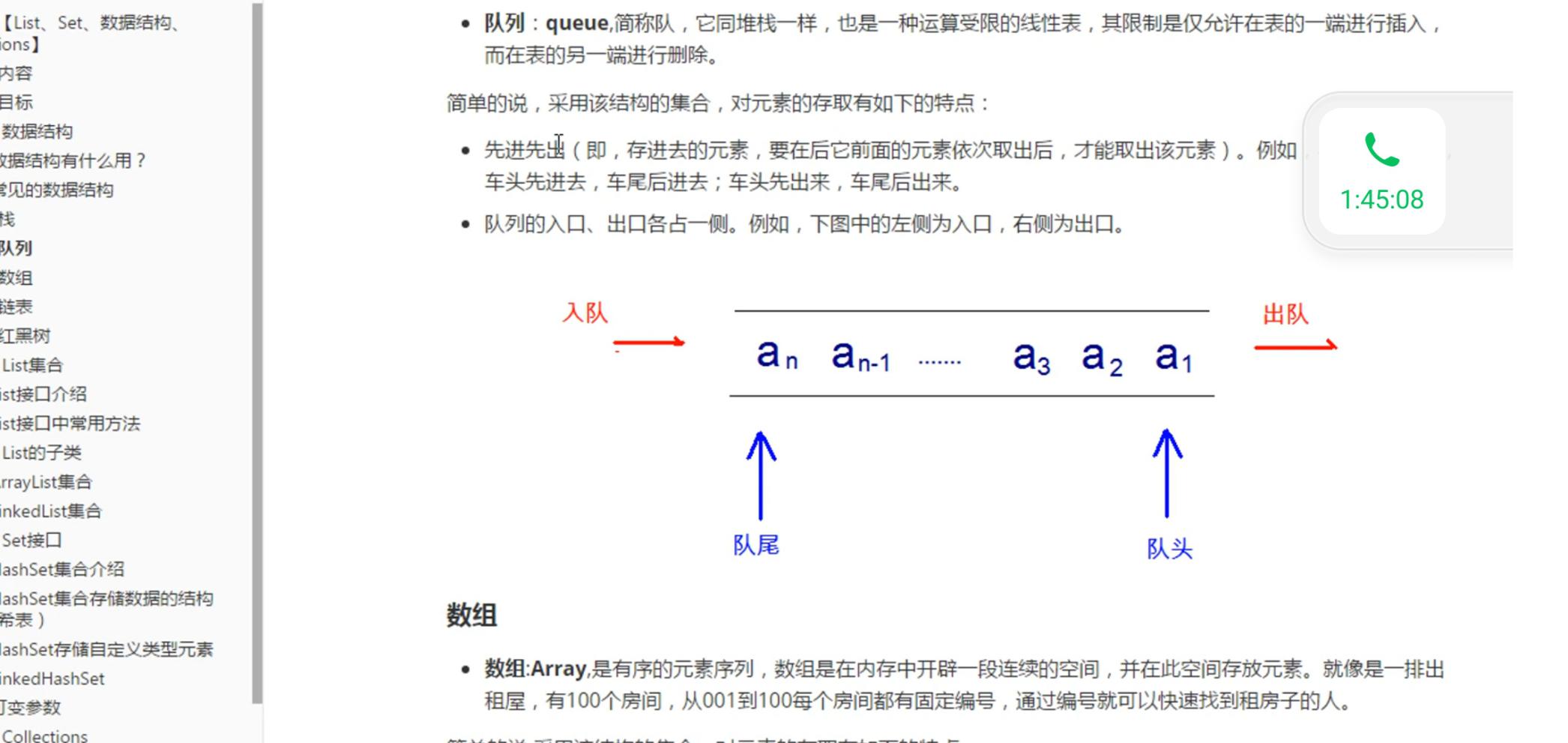
1.3 队列

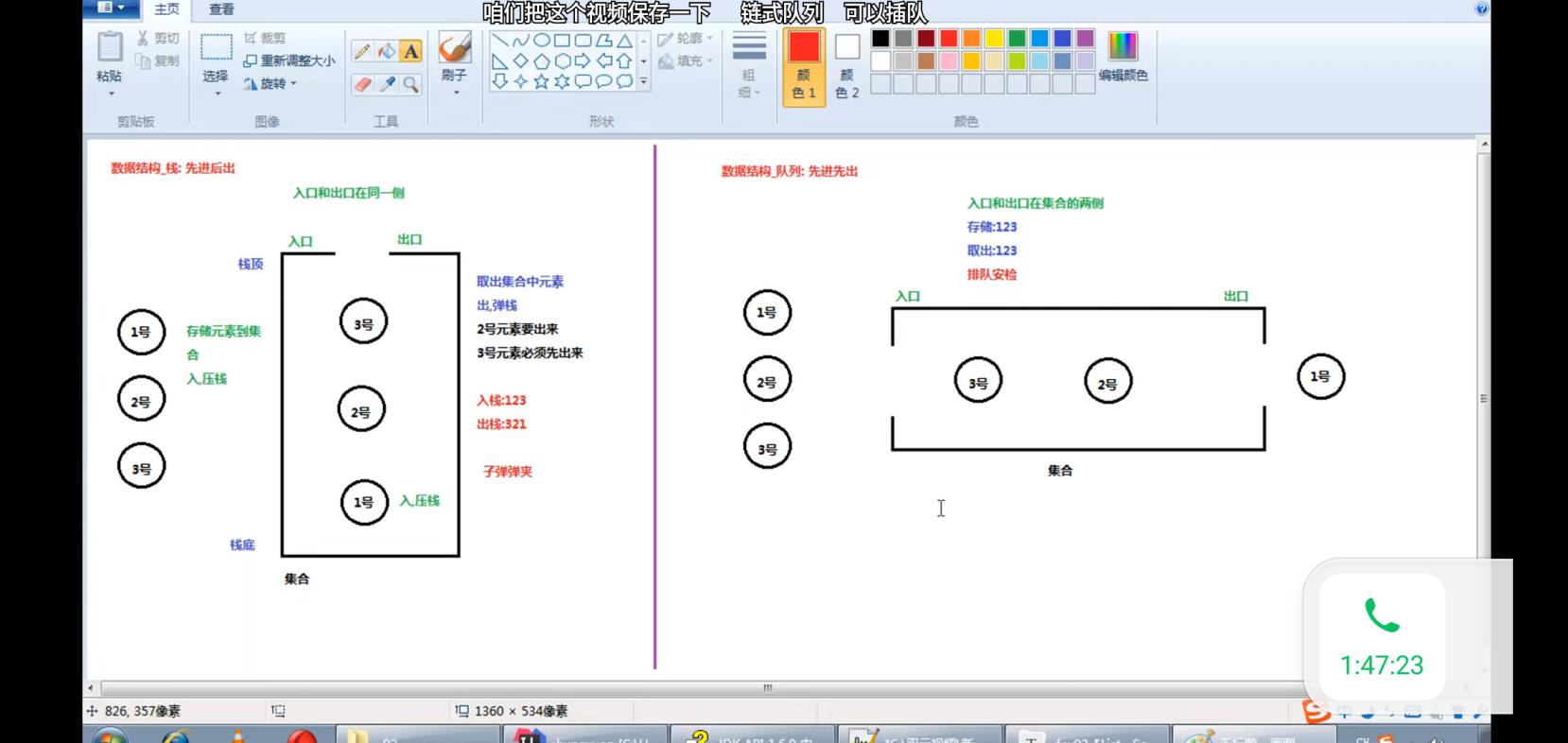
·队列：queue，简称队，它同堆栈一样，也是一种运算受限的线性表，其限制是仅允许在表的一端进行插入，而在表的另一端进行删除。

简单来说，采用该结构的集合，对元素的存取有如下的特点：

·先进先出（即，存进去的元素，要在后它的前面的元素依次取出后，才能取出该元素）。例如：小火车过山洞，车头先进去，车尾后进去；车头先出来，车位后出来）

·队列的入口：出口各占一侧。例如：下图中的左侧为入口，右侧为出口。





1.4数组

·数组Array，是有序的元素序列，数组是在内存中开辟的一段链接空间，并在此空间存放元素。就像是一排出租屋，有100个房间，从001到100每个房间都有固定的编号，通过编号的就可以快速找到租房子的人。

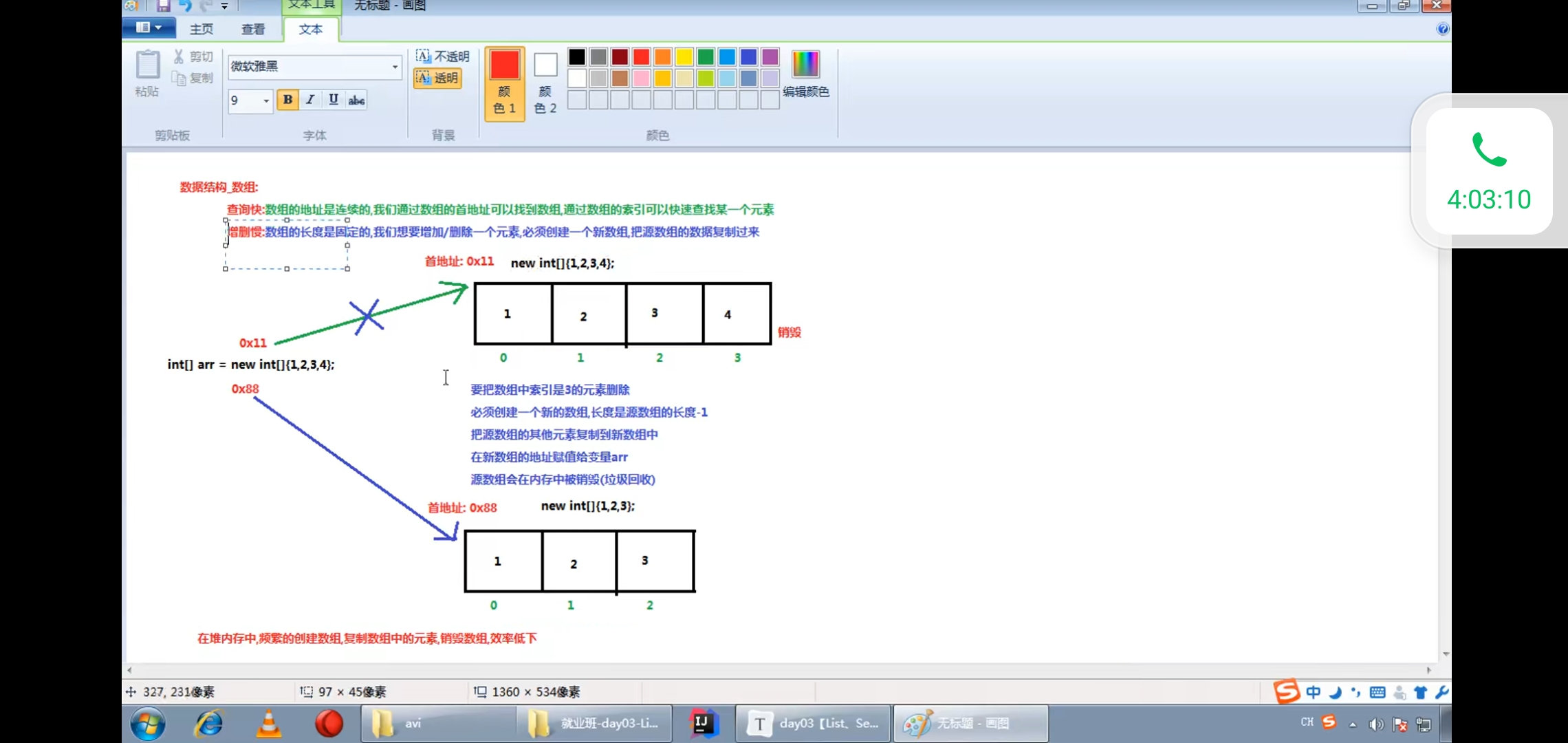
建档说明：采用该结构的集合，对元素的存取有如下的特点：

·查找元素快：通过索引，可以快速访问指定位置的元素

·增删元素慢

数组的长度是固定的，我们想要增加/删除一个元素，必须创建一个新数组，把源数组的数据复制过来

指定索引位置增加元素：需要创建一个新数组，将指定新元素存储在指定索引位置，再把原数组元素根据索引，复制到新数组对应索引的位置，如下图：



1.5 链表

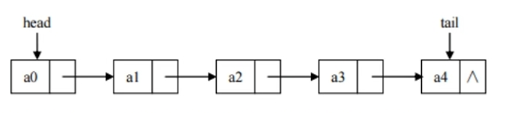
链表：linked list，是由一系列节点node（链表中的每一个元素称为结点）组成，结点可以在运行是动态生成。每个结点包括两个部分：一个是存储数据元素怒的数据域，另一个是存储下一个结点地址的指针域。我们常说的链表结构有单向链表与双向链表，那么这里给大家介绍的是单向链表。

数据域 指针域

|  |  |
| --- | --- |
| date | next |

简单的说：采用该结构的集合，对元素的存取有如下的特点：

·多个结点之间，通过地址进行链接。例如，多个人手拉手，每个人使用自己的右手拉住下一个人的左手，依次类推，这样多个人就链接到了一起。

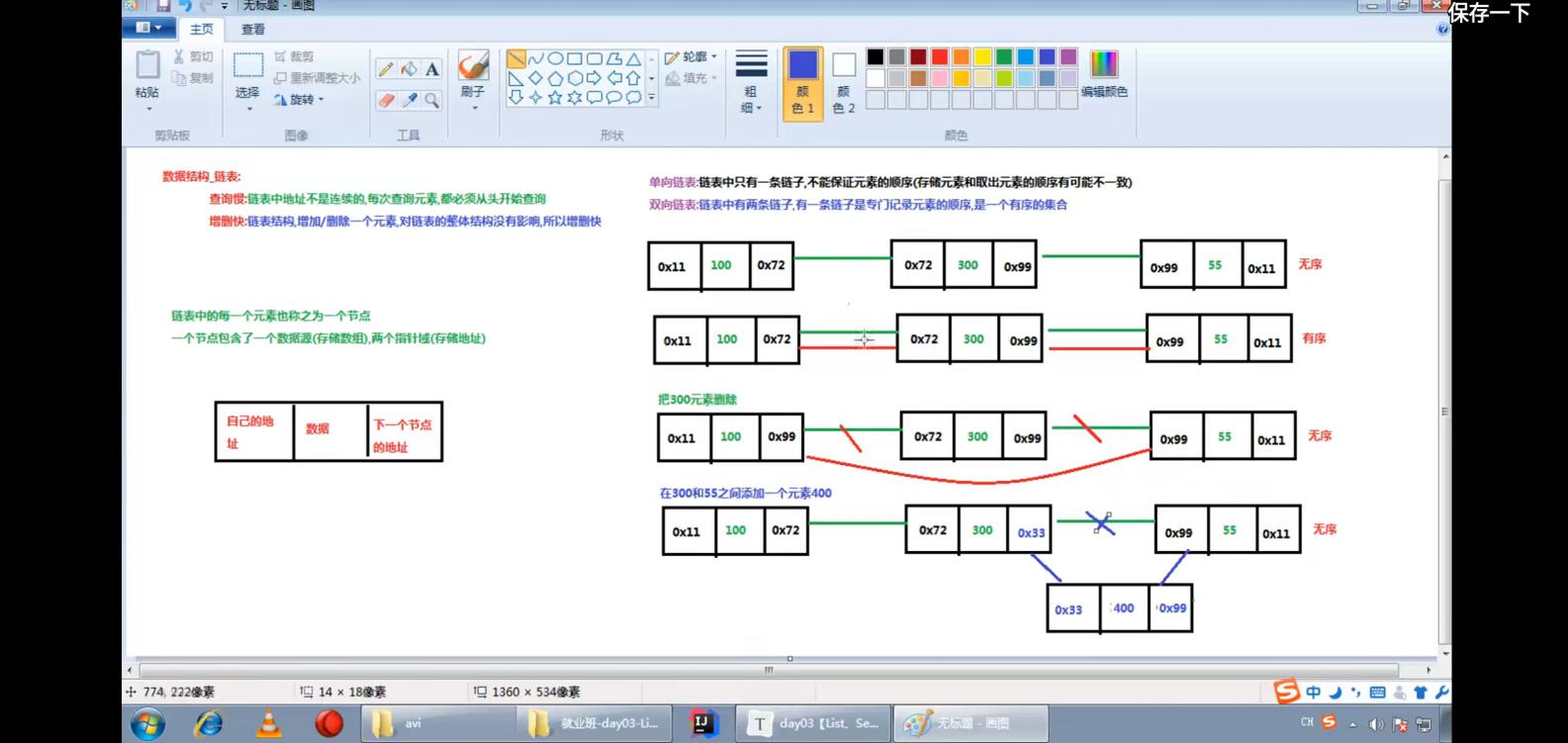


·查找元素慢：想要找某个元素，需要通过链接的节点，依次向后查找指定的元素。

·增删元素快：

增加元素：只需要修改链接下一个元素的地址值即可。

删除元素：直接在两个元素之间删除即可，剩下的两个元素相互链接即可。

n

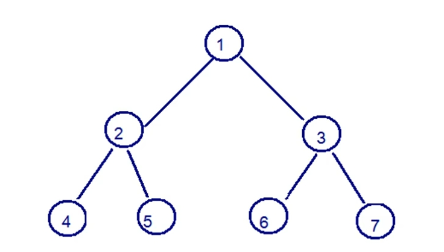
1.6 红黑树

·二叉树：binary tree，是每个结点不超过两个的有序树。

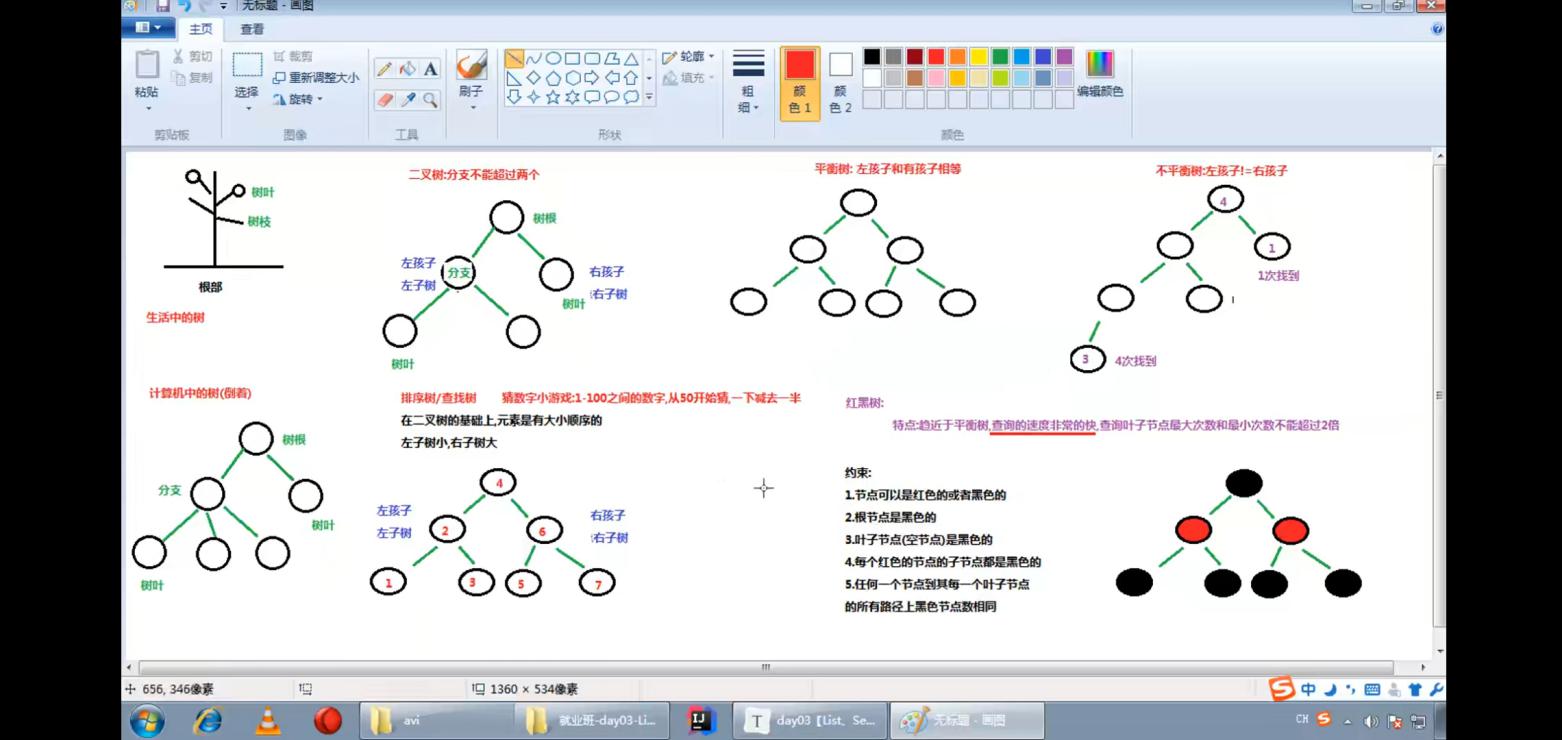
简单的理解，就是一种类似于我们生活中树的结构，只不过每个结点上都最多只能有两个子结点。

二叉树是每个节点最多两个子树的树结构。顶上的叫做根节点，两边被称作“左子树”和“右子树”。

如图：



红黑树的表示：



1. List集合

我们掌握了Collection接口的使用后，在看看Collection接口中的子类，他们具有哪些特征:

下边是Collection中常用的两个子类（java.util.List集合、java.util.Set集合）。

java.util.List接口继承自Collection接口，是单列集合的一个重要分支。

package demo02;  
  
import java.util.ArrayList;  
import java.util.Iterator;  
import java.util.List;  
  
/\*  
java.util.List接口 extends Collection接口  
List接口的特点：  
 1.有序的集合，存储元素和取出元素的顺序是一致的（存储123 取出123)  
 2.有索引，包含了一些带有索引的方法  
 3.允许存储重复的元素  
  
List接口中带有索引的方法（特有）：  
 - public void add(int index, E element) ：在列表的指定位置插入指定元素（可选操作）。  
 - public E get(int index) :返回列表中指定位置的元素。  
 - public E remove(int index) :移除列表中指定位置的元素（可选操作）。  
 - public E set(int index, E element) :用指定元素替换列表中指定位置的元素（可选操作）。  
注意：  
 操作索引的时候，一定要防止索引越界异常。  
 IndexOutOfBoundsException:索引越界异常，集合会报。  
 ArrayIndexOutOfBoundsException:数组索引越界异常。  
 StringIndexOutOfBoundsException:字符串索引越界异常。  
 \*/  
public class DemoList {  
 public static void main(String[] args) {  
 // 创建一个List集合对象，多态。  
 List<String> list = new ArrayList<>();  
 // 使用add方法往集合中添加元素。  
 // List中可以存储相同的元素。  
 list.add("a");  
 list.add("b");  
 list.add("c");  
 list.add("d");  
 list.add("a");  
 // 打印集合  
 System.out.println(list); // [a, b, c, d, a] 不是地址值，说明重写了toString方法。  
  
 // public void add(int index, E element) ：在列表的指定位置插入指定元素（可选操作）。  
 // 在 c 和 d 之间添加一个itheima  
 list.add(3,"itheima"); // 在索引值为3的地方添加itheima  
 System.out.println(list); // [a, b, c, itheima, d, a]  
  
 // public E get(int index) :返回列表中指定位置的元素,返回是被移除的元素。  
 // 移除元素  
 String remove = list.remove(2);  
 System.out.println("被移除的元素："+remove); //被移除的元素：c  
 System.out.println(list); // [a, b, itheima, d, a]  
  
 // public E remove(int index) :移除列表中指定位置的元素（可选操作）。  
 // 把最后一个元素a，替换为A  
 String set = list.set(4, "A");  
 System.out.println("被替换的元素："+set); //被替换的元素：a  
 System.out.println(list); // [a, b, itheima, d, A]  
  
 // List集合遍历有三种方式  
 // 使用普通的for循环。  
 for (int i = 0; i < list.size(); i++) {  
 // public E get(int index) :返回列表中指定位置的元素。  
 String s = list.get(i);  
 System.out.println(s);  
 }  
  
 // 使用迭代器  
 Iterator<String> it = list.iterator();  
 while(it.hasNext()){  
 String s = it.next();  
 System.out.println(s);  
 }  
  
 // 使用增强的for  
 for (String s : list) {  
 System.out.println(s);  
 }  
  
 String r = list.get(5); // 溢出  
 System.out.println(r); //java.lang.IndexOutOfBoundsException  
 }  
}

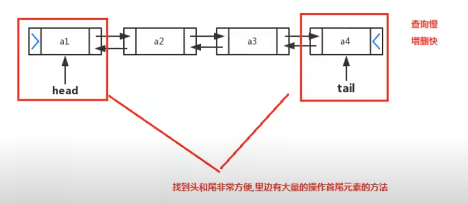
1. List的子类

3.1 ArrayList集合

java.util.ArrayList集合数据存储的结构是数组结构。元素的增删慢，查找快，由于日常的开发中使用最多的功能为查询数据、遍历数据，所以ArrayList最常用的集合。

3.2 LinkedList集合

java.util.LinkedList集合数据存储的结构是链表结构。方便元素的添加、删除的集合。



LinkedList集合的方法：

package demo02;  
  
import java.util.LinkedList;  
  
/\*  
java.util.LinkedList集合 implements List接口  
LinkedList集合的特点：  
 1.底层是一个链表结构：查询慢，增删快。  
 2.里边包含了大量操作首尾元素的方法  
注意：  
 使用LinkedList集合特有的方法，不能使用多态。  
  
 - public void addFirst(E e) ：将指定元素插入此列表的开头。  
 - public void addLast(E e)：将指定元素添加到此列表的结尾。  
 - public void push(E e)：将元素推入此列表所表示的堆栈。此方法等效于addFirst(E e);  
  
 - public E getFirst()：返回此列表的第一个元素。  
 - public E getLast()：返回此列表的最后一个元素。  
  
 - public E removeFirst()：移除并返回此列表的第一个元素。  
 - public E removeLast()：移除并返回此列表的最后一个元素。  
 - public E pop() ：从此列表所表示的堆栈处弹出一个元素。  
  
 -public boolean isEmpty():如果列表中不包含元素，则返回true。  
  
  
 \*/  
public class DemoLinkedList {  
 public static void main(String[] args) {  
 // 创建LinkedList集合对象  
 LinkedList<String> lisked = new LinkedList<>();  
 // 使用add方法向集合里边添加元素  
 lisked.add("a");  
 lisked.add("b");  
 lisked.add("c");  
 System.out.println(lisked); // [a, b, c]  
 demo01();  
 demo02();  
 demo03();  
 }  
  
 /\*  
 - public void addFirst(E e) ：将指定元素插入此列表的开头。  
 - public void addLast(E e)：将指定元素添加到此列表的结尾。  
 - public void push(E e)：将元素推入此列表所表示的堆栈。  
 \*/  
 public static void demo01(){  
  
 // 创建LinkedList集合对象  
 LinkedList<String> lisked = new LinkedList<>();  
 // 使用add方法向集合里边添加元素  
 lisked.add("a");  
 lisked.add("b");  
 lisked.add("c");  
 System.out.println(lisked); // [a, b, c]  
  
 // - public void addFirst(E e) ：将指定元素插入此列表的开头。  
 lisked.addFirst("wwww");  
 // lisked.push("www");  
 System.out.println(lisked); // [wwww, a, b, c]  
  
 // public void addLast(E e)：将指定元素添加到此列表的结尾。此方法等效于add()；  
 lisked.addLast("com");  
 System.out.println(lisked); // [wwww, a, b, c, com]  
 }  
  
 /\*  
 - public E getFirst()：返回此列表的第一个元素。  
 - public E getLast()：返回此列表的最后一个元素。  
 \*/  
 public static void demo02(){  
  
 // 创建LinkedList集合对象  
 LinkedList<String> lisked = new LinkedList<>();  
 // 使用add方法向集合里边添加元素  
 lisked.add("a");  
 lisked.add("b");  
 lisked.add("c");  
 System.out.println(lisked); // [a, b, c]  
  
 // 独立分支  
 //lisked.clear(); // 清空集合中的元素，在获取集合中的元素时会抛出NoSuchElementException  
  
 // -public boolean isEmpty():如果列表中不包含元素，则返回true。  
 if(!lisked.isEmpty()) {  
 String first = lisked.getFirst();  
 System.out.println(first); // a  
 String last = lisked.getLast();  
 System.out.println(last); // c  
 }  
 }  
  
 /\*  
 - public E removeFirst()：移除并返回此列表的第一个元素。  
 - public E removeLast()：移除并返回此列表的最后一个元素。  
 - public E pop() ：从此列表所表示的堆栈处弹出一个元素,此方法相当于 removeFirst()；  
  
 \*/  
 public static void demo03(){  
 // 创建LinkedList集合对象  
 LinkedList<String> lisked = new LinkedList<>();  
 // 使用add方法向集合里边添加元素  
 lisked.add("a");  
 lisked.add("b");  
 lisked.add("c");  
 System.out.println(lisked); // [a, b, c]  
  
 // String first = lisked.removeFirst();  
 String first = lisked.pop();  
 System.out.println("被移除的第一个元素："+first); //被移除的第一个元素：a  
 String last = lisked.removeLast();  
 System.out.println("被移除的最后一个元素："+last); //被移除的最后一个元素：c  
 System.out.println(lisked); //[b]  
 }  
}

1. Set接口

java.util.Set接口和java.util.List接口一样，同样继承自Collection接口，它与Collection接口中的方法基本一致，并没有对Collection接口进行功能上的扩充，只是比Collection接口更加严格。与List接口不同的是，Set接口中的元素无序，并且都会以某种规则保证存入的元素不出现重复。

Setj接口：一个不包含重复元素的Collection

Set集合有多个子类，我们介绍java.util.HashSet、java.util.LinkedHashSet两个集合。

tips: Set集合取出元素的方式可以采用：迭代器、增强for

4 .1 HashSet 集合介绍

java.util.HashSet 是 Set接口的一个实现类，它所存储的元素是不可重复的，并且元素都是无序的（即存取顺序可以不一致）java.util.HashSet 底层的实现其实就是一个 java.util.HashMap支持，由于我们暂时还没学习，先做了解。

HashSet 是根据对象的哈希值来确定元素在集合中的存储位置，因此具有良好的存储和查找性能。保证元素唯一性的方式依赖：hashCode与equals方法。

package demo02;  
  
import java.util.HashSet;  
import java.util.Iterator;  
import java.util.Set;  
  
/\*  
java.util.Set接口 extends Collection接口  
Set接口的特点：  
 1.不允许存储重复的元素。  
 2.，没有索引，没有带索引的方法，也不能使用普通的for循环遍历。  
java.util.HashSet集合 implements Set接口  
HashSet特点：  
 1.不允许存储重复的元素  
 2.没有索引，没有带索引的方法，也不能使用for循环进行遍历  
 3.是一个无序的集合，存储元素和取出元素的顺序可能不一样。  
 4.底层是一个哈希表结构（查询的速度非常快）  
  
 \*/  
public class DemoSet {  
 public static void main(String[] args) {  
 Set<Integer> set = new HashSet<>();  
 // 使用add方法往集合中添加元素  
 set.add(1);  
 set.add(2);  
 set.add(3);  
 set.add(1);  
 System.out.println(set); // [1, 2, 3] 不可存储重复元素  
  
 // 使用迭代器遍历set集合  
 Iterator<Integer> iterator = set.iterator();  
 while (iterator.hasNext()){  
 Integer n = iterator.next();  
 System.out.println(n); // 1,3,2  
 }  
 // 使用增强的for集合  
 for (Integer i : set) {  
 System.out.println(i); // 1,2,3  
 }   
 }  
}

1. Collections

5.1 常用功能

- java.utils.Collections 是集合工具类，用于对集合进行操作。部分方法如下：  
-public static <T> boolean addAll(Collection<T> c,T...elements)：往集合中添加一些元素。

-public static void shuffle(list<?> list) 打乱顺序，打乱集合的顺序

-public static <T> void sort(List<T> list):将集合中的元素按照默认规则排序。