1 合并链表  
//已知两个链表head1 和head2 各自从小到大有序，请把它们归并成一个链表依然从小到大有序  
class Node{  
int data  
Node next  
}  
Node Merge(Node head1 , Node head2) {  
//TODO  
}

private static Node marge(Node head1,Node head2) {

//判断空

if(head1 == null || head2 == null) {

return head1 == null? head2 : head1;

}

//判断合并后头节点

Node node = head1.getData() <= head2.getData() ? head1:head2;

//使用head1和head2的副本

Node current1 = node == head1 ? head1:head2;

Node current2 = node == head1 ? head2:head1;

//使用临时空间

Node temp1 = null;

Node temp2 ;

//判断哪个先比较完

while(current1!=null&&current2!=null){

if(current1.getData() <= current2.getData()){

//当副本current1的值小于等于副本current2中的值，将小值放入临时空间temp1

temp1 = current1;

//继续比较current1中下一个值

current1 = current1.getNext();

}else {

//使用临时空间temp2保存第一个元素小的副本current2的第二个节点及后续元素

temp2 = current2.getNext();

//将第一个值小的副本current2拷贝到临时空间temp1

temp1.setNext(current2);

//将较大值放入current2下一个节点

current2.setNext(current1);

//当副本current1的值不小于副本current2中的值，将小值放入临时空间temp1

temp1 = current2;

current2 = temp2;

}

}

temp1.setNext(current1 == null ? current2 : current1);

return node;

}

2 多线程  
2.1、吃桔子  
共计9个桔子，有3个小朋友，小朋友A每次拿2个桔子，小朋友B每次拿3个桔子，小朋友C每次拿1个桔子，小朋友10s吃1个桔子，吃完后继续去拿。  
小朋友每次拿桔子之前和拿了桔子之后，都会对桔子数量进行报数。如果剩余的桔子不够小朋友每次拿的数量，小朋友停止拿桔子，喊一声“不拿了”并退出游戏。  
请用java多线程程序表述上面的过程。

public class EatOrange {

static int number = 9;

public static void main(String[] args) {

final Object boy = new Object();

while (number > 0) {

//小孩A

new Thread(new Runnable() {

@Override

public void run() {

synchronized (boy) {

if (number >= 2) {

try {

number -= 2;

System.out.println("小孩A拿2个橘子还剩了" + number + "个橘子");

System.out.println("小孩A开始吃橘子");

boy.notifyAll();

boy.wait(2000);

System.out.println("小孩A吃完2个橘子");

} catch (InterruptedException e) {

e.printStackTrace();

}

}else {

Thread.interrupted();

System.out.println("A不拿了");

return;

}

}

}

}).start();

//小孩B

new Thread(new Runnable() {

@Override

public void run() {

synchronized (boy) {

if (number >= 3) {

try {

number -= 3;

System.out.println("小孩B拿3个橘子还剩了" + number + "个橘子");

System.out.println("小孩B开始吃橘子");

boy.notifyAll();

boy.wait(3000);

System.out.println("小孩B吃完3个橘子");

} catch (InterruptedException e) {

e.printStackTrace();

}

}else {

System.out.println("B不拿了");

return;

}

}

}

}).start();

//小孩C

new Thread(() -> { //lambda

synchronized (boy) {

if (number >= 1) {

try {

number -= 1;

System.out.println("小孩C拿1个橘子还剩了" + number + "个橘子");

System.out.println("小孩C开始吃橘子");

boy.notifyAll();

boy.wait(1000);

System.out.println("小孩C吃完1个橘子");

} catch (InterruptedException e) {

e.printStackTrace();

}

}else {

System.out.println("C不拿了");

return;

}

}

}).start();

}

}

}

2.2、跑步  
3个小朋友参加百米跑步比赛，A小朋友每秒跑5米，B小朋友每秒跑4米，C小朋友每秒跑3米。大家做好准备之后喊一声“我准备好了”，然后在听到发令枪响后一起起步，并在开始跑步时喊一声“我开始跑步了”，当一个小朋友跑到终点时，喊一声“我到终点了”，此时记录跑了多久。其他小朋友停止跑步，并报出自己跑了多少米，例如“x小朋友跑了yy米”。  
请用java程序表述上面的过程。

package Test.TWO;

public class RunRace {

static int distanceA = 100;

static int distanceB = 100;

static int distanceC = 100;

static int i = 0, j = 0, k = 0;

public static void main(String[] args) {

//A

new Thread(new Runnable() {

@Override

public void run() {

System.out.println("A准备好了");

while (distanceA > 0) {

distanceA -= 5;

i++;

if(distanceB<=0 || distanceC<=0){

break;

}

}

if(distanceB<=0){

System.out.println("A我跑了"+j\*5+"米");

}else if(distanceC<=0) {

System.out.println("A我跑了"+k\*5+"米");

}else{

System.out.println("A我跑完了");

}

}

}).start();

//B

new Thread(new Runnable() {

@Override

public void run() {

System.out.println("B准备好了");

while (distanceB > 0) {

distanceB -=4;

j++;

if(distanceA<=0 || distanceC<=0){

break;

}

}

if(distanceA<=0){

System.out.println("B我跑了"+i\*4+"米");

}else if(distanceC<=0) {

System.out.println("B我跑了"+k\*4+"米");

}else{

System.out.println("B我跑完了");

}

}

}).start();

//C

new Thread(() -> { //lambda

System.out.println("C准备好了");

while (distanceC > 0) {

distanceC -=3;

k++;

if(distanceA<=0 || distanceB<=0){

break;

}

}

if(distanceB<=0){

System.out.println("C我跑了"+j\*3+"米");

}else if(distanceA<=0) {

System.out.println("C我跑了"+i\*3+"米");

}else{

System.out.println("C我跑完了"+j);

}

}).start();

}

}

3分析字符串  
写字符串的分析函数，分析一个由字母和数字组成的字符串，例如aBc12D9  
3.1、输出字符串中字符和数字的个数。  
3.2、相连的数字不能分为2个，即12是作为一个数字统计，输出字符串中字符和数字的个数。  
3.3、字符大小写不区分，统计字符的个数及出现次数。  
3.4、统计出现次数最多的字符和数字。  
按照上面的要求，写4个函数

3.1 public void charAndNum(String s) {

int lowerCharNum = 0, superCharNum = 0, num = 0;

for (int i = 0; i < s.length(); i++) {

if (48 <= s.charAt(i) && s.charAt(i) <= 57) {

num++;

} else if (57 <= s.charAt(i) && s.charAt(i) <= 90) {

superCharNum++;

} else {

lowerCharNum++;

}

}

System.out.println("字符串中大写字符个数：" + superCharNum + " 小写字符个数：" + lowerCharNum + " 数字个数：" + num);

}

3.2 public void charAndNum2(String s) {

int lowerCharNum = 0, superCharNum = 0, num = 0;

for (int i = 0; i < s.length(); i++) {

if (48 <= s.charAt(i) && s.charAt(i) <= 57) {

num++;

int j = i + 1;

if (48 <= s.charAt(j) && s.charAt(j) <= 57) {

num--;

}

} else if (57 <= s.charAt(i) && s.charAt(i) <= 90) {

superCharNum++;

} else {

lowerCharNum++;

}

}

System.out.println("字符串中大写字符个数：" + superCharNum + " 小写字符个数：" + lowerCharNum + " 数字个数：" + num);

}

3.3 public void ignoreCase(String s) {

int charNum = 0, num = 0;

for (int i = 0; i < s.length(); i++) {

if (48 <= s.charAt(i) && s.charAt(i) <= 57) {

num++;

int j = i + 1;

if (48 <= s.charAt(j) && s.charAt(j) <= 57) {

num--;

}

} else {

charNum++;

}

}

System.out.println("字符字符个数：" + charNum + " 数字个数：" + num);

}

//不考虑连续数字

3.4 public void numOfCharAndNum(String s) {

Map<Character, Integer> map = new TreeMap<>();

for (int i = 0; i < s.length(); i++) {

Integer nums = map.get(s.charAt(i));

map.put(s.charAt(i), (nums == null) ? 1 : nums + 1);

}

//将字符数组中的第一个字符的个数作为哨兵进行比较

int number = map.get(s.charAt(0));

//list存放出现次数最多的字符

ArrayList<Character> listChar = new ArrayList<>();

ArrayList<Character> listInt = new ArrayList<>();

//找出出现最多次的字符

for (int i = 0; i < s.length(); i++) {

if (map.get(s.charAt(i)) > number) {

number = map.get(s.charAt(i));

if (48 <= s.charAt(i) && s.charAt(i) <= 57) {

listInt.clear();

listInt.add(s.charAt(i));

} else {

listChar.clear();

listChar.add(s.charAt(i));

}

} else if (map.get(s.charAt(i)) == number) {

if (48 <= s.charAt(i) && s.charAt(i) <= 57) {

listInt.add(s.charAt(i));

} else {

listChar.add(s.charAt(i));

}

}

}

System.out.print("最多的字符是：");

for (char c : listChar) {

System.out.println(c + " ");

}

System.out.print("最多的数字是：");

for (char c : listInt) {

System.out.println(c + " ");

}

}

备注：考虑连续数字暂时没有思路

4 删除字符  
要求：删除一个字符串中出现次数最多的字符，如果多个字符出现次数一样，则都删除。要求字符顺序不变。

public void deleteTheMost(String str){

char[] ch = str.toCharArray();

Map<Character,Integer> map = new TreeMap<>();

//遍历字符数组，统计每个字符出现的次数并存放在Map中（key：字符；value：出现次数）。

for (int i = 0; i < ch.length; i++) {

Integer num = map.get(ch[i]);

map.put(ch[i], (num==null)?1:num+1);

}

//将字符数组中的第一个字符的个数作为哨兵进行比较

int number = map.get(ch[0]);

//list存放出现次数最多的字符

ArrayList<Character> list = new ArrayList<>();

//找出出现最多次的字符

for (int i = 0; i < ch.length; i++) {

if(map.get(ch[i]) > number){

number = map.get(ch[i]);

list.clear();

list.add(ch[i]);

}else if(map.get(ch[i]) == number){

list.add(ch[i]);

}

}

//删除出现最多的字符

for (char c : list) {

System.out.println(c);

str = str.replace(String.valueOf(c), "");

}

System.out.println(str);

}