### 1题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**topping1**;

  修改传递进来的map集合并返回：如果存在key “ice cream”，  
则将其value设置为“cherry”。 在所有情况下，增加 key “bread”，value 是“butter”。

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| topping1({"ice cream":"peanuts"}) | {"bread":"butter","ice cream":"cherry"} |
| topping1({}) | {"bread":"butter"} |
| topping1({"pancake":"syrup"}) | {"bread":"butter","pancake":"syrup"} |

**public** **class** Itheima{

**public** Map<String, String> topping1(Map<String, String> map){

**if** (map.containsKey("ice cream")) {

map.put("ice cream", "cherry");

}

map.put("bread", "butter");

**return** map;

}

}

### 2题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**pairs**;

  给定一个非空字符串数组，创建并返回一个Map <STRING，STRING>，在字符串数组中，  
将每个不同字符串的第一个字符作为一个key，最后一个字符作为value。

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| pairs(["code","bug"]) | {"b":"g","c":"e"} |
| pairs(["man","moon","main"]) | {"m":"n"} |
| pairs(["man","moon","good","night"]) | {"g":"d","n":"t","m":"n"} |

**public** **class** Itheima {

**public** Map<String, String> pairs(String[] arr ) {

Map<String, String> map = **new** HashMap<>();

**for** (String s : arr) {

map.put(s.substring(0, 1), s.substring(s.length()-1, s.length()));

}

**return** map;

}

}

### 3题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**topping2**;

  修改传递进来的map集合：如果存在 “ice cream”,就增加键值对 key为 “yogurt”，value和“ice cream”具有相同的值。   
如果存在key “spinach”，就将其value更改为“nuts”。其余不变

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| topping2({"ice cream":"cherry"}) | {"yogurt":"cherry","ice cream":"cherry"} |
| topping2({"spinach":"dirt","ice cream":"cherry"}) | {"yogurt":"cherry","spinach":"nuts","ice cream":"cherry"} |
| topping2({"yogurt":"salt"}) | {"yogurt":"salt"} |

**public** **class** Itheima {

**public** Map<String, String> topping2(Map<String, String> map) {

**if** (map.containsKey("ice cream")) {

map.put("yogurt", map.get("ice cream"));

}

**if** (map.containsKey("spinach")) {

map.put("spinach", "nuts");

}

**return** map;

}

}

### 4题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**loneSum**;

  给定3个int值，A B C，返回他们的总和。然而，如果其中一个值与另一个值相同，则它们不参与求和。

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| loneSum(1,2,3) | 6 |
| loneSum(3,2,3) | 2 |
| loneSum(3,3,3) | 0 |

**public** **class** Itheima {

**public** **int** loneSum(**int** A, **int** B, **int** C) {

**int** sum ;

**if** (A == B && B== C) {

**return** sum = 0;

}**else** **if** (A != B && A == C) {

**return** sum = B;

}**else** **if** (A == B && A != C) {

**return** sum = C;

}**else** **if** (B == C && A != B) {

**return** sum = A;

}**else** **if** (A != B && A != C && B != C) {

**return** sum = A + B + C;

}**else**{

**return** sum = 0;

}

}

}

### 5题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**has22**;

  给定一个整型数组，如果数组中出现两个相邻的索引中的元素都是2，则返回true

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| has22([1,2,2]) | true |
| has22([1,2,1,2]) | false |
| has22([2,1,2]) | false |

**public** **class** Itheima {

**public** **boolean** has22(**int**[] arr) {

**for** (**int** i = 0; i < arr.length - 1; i++) {

**if** (2 == arr[i] && 2 == arr[i + 1])

**return** **true**;

}

**return** **false**;

}

}

### 6.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**mapAB**;

  修改并返回给定的映射如下：在键值对映射中可以包含或可以不包含“a”和“b”键。   
如果ab两个键都存在，则将它们的2个字符串的值附加在一起并将结果存储在键“ab”下。

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| mapAB({"b":"There","a":"Hi"}) | {"b":"There","a":"Hi","ab":"HiThere"} |
| mapAB({"a":"Hi"}) | {"a":"Hi"} |
| mapAB({"b":"There"}) | {"b":"There"} |

**public** **class** Itheima {

**public** Map<String, String> mapAB(Map<String, String> map){

**if** (map.containsKey("a")&&map.containsKey("b")) {

map.put("ab",map.get("a")+map.get("b") );

}

**return** map;

}

}

### 7.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**closeFar**;

  给定三个int，a b c，如果b或c中的一个是与a相差最多为1，并且另一个与其他两个值相差2或者更大，则返回true。  
注意：Math.abs（num）计算数字的绝对值。

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| closeFar(1,2,10) | true |
| closeFar(1,2,3) | false |
| closeFar(4,1,3) | true |

**public** **class** Itheima {

**public** **boolean** closeFar(**int** a, **int** b, **int** c) {

**if** (Math.*abs*(Math.*abs*(b) - Math.*abs*(a)) <= 1) {

**if** ((Math.*abs*(Math.*abs*(c) - Math.*abs*(b)) >= 2) && (Math.*abs*(Math.*abs*(c) - Math.*abs*(a)) >= 2)){

**return** **true**;

}

}

**if** (Math.*abs*(Math.*abs*(c) - Math.*abs*(a)) <= 1) {

**if** ((Math.*abs*(Math.*abs*(b) - Math.*abs*(c)) >= 2) && (Math.*abs*(Math.*abs*(b) - Math.*abs*(a)) >= 2)){

**return** **true**;

}

}

**return** **false**;

}

}

### 8.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**sum28**;

  给定一个整型数组，如果数组中所有2的总和为8，则返回true。

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| sum28([2,3,2,2,4,2]) | true |
| sum28([2,3,2,2,4,2,2]) | false |
| sum28([1,2,3,4]) | false |

**public** **class** Itheima {

**public** **boolean** sum28(**int**[] arr) {

**int** count = 0;

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

**if** (2 == arr[i]) {

count++;

}

}

**return** count \* 2 == 8;

}

}

### 9.题目描述:

  书写一个工具类，类名为**ExamUtils**,

  书写一个工具方法，方法名为**transition**,

  要求此方法能将传入的**字符串参数**进行处理，将处理后得到的**整型数字返回**给此方法;

  处理方式为:

    例如：传入整型数字为: **一万二千三百四十五**

    要求处理后为: **12345**

    即 **将中文数字转换为阿拉伯数字显示方式**

### 提示:

  1. **通常工具类中工具的方法都为公有静态方法**

  2. [0123456789]分别对应[零一二三四五六七八九]

  3. 中文的标准位数: 十百千万

  4. 数字判断只需要做到5位数

**class** ExamUtils {

**public** **static** **int** conversion(String st) {

HashMap<Character, Integer> hm = **new** HashMap<>(); // 值为 10的次方 如果值为2，则代表10的二次方

hm.put('十', 1);

hm.put('百', 2);

hm.put('千', 3);

hm.put('万', 4);

String zh = "零一二三四五六七八九";

**int** sum = 0;

**if**(st.length() == 1){

**if**(st.equals("十"))

**return** 10;

**else**

**return** zh.indexOf(st);

}

**for** (**int** i = 1; i < st.length(); i++) {

**char** ch = st.charAt(i); // 1万零四十

**if** (hm.containsKey(ch)) {

**int** num = (**int**) Math.*pow*(10, hm.get(ch))

\* zh.indexOf(st.charAt(i - 1)); // 10 ^ 4 \* 1

sum += num;

} **else** **if** (i == st.length() - 1) {

sum += zh.indexOf(ch);

}

}

**return** sum;

}

### }

### 10.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**centeredAverage**;

  给定一个整型数组，去掉数组中最大和最小的值, 返回其他元素的平均值

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| centeredAverage([1,2,3,4,100]) | 3 |
| centeredAverage([1,1,5,5,10,8,7]) | 5 |
| centeredAverage([-10,-4,-2,-4,-2,0]) | -3 |

**public** **class** Itheima {

**public** **int** centeredAverage(**int**[] arr) {

**int** a = 0;

**int** max = arr[0];

**int** min = arr[0];

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

a += arr[i];

}

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

**if** (arr[i] > max) {

max = arr[i];

}

}

a = a - max;

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

**if** (min > arr[i]) {

min = arr[i];

}

}

a = a - min;

a = a / (arr.length - 2);

**return** a;

}

}

### 11.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**maxBlock**;

  给定一个字符串，返回该字符串中连续出现个数最多的字符的数量。

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| maxBlock("hoopla") | 2 |
| maxBlock("abbCCCddBBBxx") | 3 |
| maxBlock("") | 0 |

**public** **class** Itheima {

**public** **int** maxBlock(String s) {

**if**(s.equals(""))**return** 0;

**int** count = 1;

**int** max = 0;

**char**[] crr = s.toCharArray();

**for** (**int** j = 0; j < crr.length; j++) {

**if**(j==crr.length-1)**break**;

**if**(crr[j]==crr[j+1]){

count++;

}**else**{

count=1;

}

**if**(max<count){

max = count;

}

}

**return** max;

}

}

方式二：

**public** **class** Itheima{

**public** **int** maxBlock(String str){

**if** (str.length()==0) {

**return** 0;

}

ArrayList<String> list = **new** ArrayList<>();

String regex = "(.)\\1+";

Pattern p = Pattern.*compile*(regex);

Matcher m = p.matcher(str);

**while**(m.find()){

list.add(m.group());

}

**int** max = 1;

**for** (String string : list) {

**if** (string.length()>max) {

max = string.length();

}

}

**return** max;

}

}

### 12.题目描述:

     /root/bootstrap/result/高东军/Exam-04/extra/itheima.txt文件中存储了**多行数字**，

**读取每行的数字**，将结果打印到控制台。

     将文件中读取出的数字进行**去重操作**，将结果打印到控制台。

     将去重后的数字按照**由小到大的方式进行排列**，并将排列后的结果打印到控制台。

     将排序后的结果**按行存储,按行存储,按行存储(重要事情说三遍,需要换行)**到/root/bootstrap/result/高东军/Exam-04/extra/itcast.avi文件中。

### 提示:

     逻辑在Demo类的main函数中实现;

     例如/root/bootstrap/result/高东军/Exam-04/extra/itheima.txt中按行存储了 9,2,5,2,6,5,4,9

     打印的结果:

     第一行--> **初始: 9 2 5 2 6 5 4 9**

     第二行--> **去重: 9 2 5 6 4**

     第三行--> **排序: 2 4 5 6 9**

**import** java.io.BufferedReader;

**import** java.io.BufferedWriter;

**import** java.io.FileReader;

**import** java.io.FileWriter;

**import** java.io.IOException;

**import** java.util.Comparator;

**import** java.util.TreeSet;

**public** **class** Demo{

**public** **static** **void** main(String[] args) **throws** IOException {

List<Integer> list = **new** ArrayList<>();

List<Integer> newList = **new** ArrayList<>();

StringBuffer sb1 = **new** StringBuffer();

BufferedReader br = **new** BufferedReader(**new** FileReader(**new** File("/root/bootstrap/result/备用2/Exam-04/extra/itheima.txt")));

String s;

**while**((s=br.readLine()) != **null**){

list.add(Integer.*parseInt*(s));

}

br.close();

//读取数据然后输出

// Collections.sort(list);

sb1.append("初始:");

**for** (Integer i : list) {

sb1.append(i+" ");

}

System.*out*.println(sb1.toString());

//创建新集合去重

**for** (Integer i : list) {

**if**(!newList.contains(i)){

newList.add(i);

}

}

sb1.delete(0, sb1.length());

sb1.append("去重:");

**for** (Integer i : newList) {

sb1.append(i+" ");

}

System.*out*.println(sb1.toString());

//排序

Collections.*sort*(newList);

sb1.delete(0, sb1.length());

sb1.append("排序:");

**for** (Integer i : newList) {

sb1.append(i+" ");

}

System.*out*.println(sb1.toString());

BufferedWriter bw = **new** BufferedWriter(**new** FileWriter(**new** File("/root/bootstrap/result/备用2/Exam-04/extra/itheima.avi")));

**for** (Integer i : newList) {

bw.write(i+"");

bw.newLine();

}

bw.close();

}

### }

### 13.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**tenRun**;

  有一个整型的数组，如果数组的元素是10的倍数，后边改为该数，直到遇到下一个10的倍数的元素，后边改为该数即可，因此{2，10，3，4，20，5} 改为{2,10,10,10,20,20}

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| tenRun([2,10,3,4,20,5]) | [2,10,10,10,20,20] |
| tenRun([10,1,20,2]) | [10,10,20,20] |
| tenRun([10,1,9,20]) | [10,10,10,20] |

**public** **class** Itheima {

**public** **int**[] tenRun(**int**[] arr) {

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

**if**(i == arr.length - 1)**break**;

**if** (arr[i] % 10 == 0) {

**if** (arr[i + 1] % 10 != 0) {

arr[i + 1] = arr[i];

}

}

}

**return** arr;

}

}

### 14.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**more14**;

  给定一个整型数组，如果1的出现次数大于4出现的次数，则返回true

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| more14([1,4,1]) | true |
| more14([1,4,1,4]) | false |
| more14([1,1]) | true |

**public** **class** Itheima {

**public** **boolean** more14(**int**[] arr) {

**int** count1 = 0;

**int** count4 = 0;

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

**if** (arr[i] == 1) {

count1++;

}

**if** (arr[i] == 4) {

count4++;

}

}

**if** (count1 > count4) {

**return** **true**;

}

**return** **false**;

}

}

### 15.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**gHappy**;

  如果字符串中的’g’的相邻左边或者右边有另外一个’g’，则’g’在这个字符串中是happy的，  
如果字符串中所有的’g’都是happy的则返回true，否则返回false。

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| gHappy("xxggxx") | true |
| gHappy("xxgxx") | false |
| gHappy("xxggyygxx") | false |

**public** **class** Itheima {

**public** **boolean** gHappy(String st) {

Pattern p = Pattern.*compile*("g+");

Matcher m = p.matcher(st);

**while**(m.find()) {

**if**(m.group().length() == 1)

**return** **false**;

}

**return** **true**;

}

}

### 16.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**countYZ**;

  给定一个字符串，找出里面以“y”或者“z”结尾的单词的个数，也就是说，跟在“y”或者“z”后面的字符不是英文单词。  
“y”在“heavy”中和“z”在“fez”中计数，而“y”在“yellow”中不计数，（不区分大小写）

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| countYZ("fez day") | 2 |
| countYZ("day fez") | 2 |
| countYZ("day fyyyz") | 2 |

**public** **class** Itheima {

**public** **int** countYZ(String str) {

**int** num = 0;

**if**(str == **null**){

num = 0;

}**else** **if**(str == ""){

num = 0;

}**else** {

String[] arr = str.split("[^a-zA-Z]");

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

**if**(arr[i].endsWith("z") || arr[i].endsWith("y") ||arr[i].endsWith("Z") || arr[i].endsWith("Y")){

num++;

}

}

}

**return** num;

}

}

### 17.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**only14**;

  给定一个整型数组，如果有元素都是1和4或者其中之一，就返回true。

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| only14([1,4,1,4]) | true |
| only14([1,4,2,4]) | false |
| only14([1,1]) | true |

**public** **class** Itheima {

**public** **boolean** only14(**int**[] arr) {

**if**(arr.length == 0)

**return** **false**;

**for** (**int** x = 0; x < arr.length; x++) {

**if** (arr[x] != 1 && arr[x] != 4) {

**return** **false**;

}

}

**return** **true**;

}

}

### 18.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**twoTwo**;

  给定一个整型数组，如果数组中出现的每个2位于另一个2旁边，则返回true。

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| twoTwo([4,2,2,3]) | true |
| twoTwo([2,2,4]) | true |
| twoTwo([2,2,4,2]) | false |

**public** **class** Itheima {

**public** **boolean** twoTwo(**int**[] arr) {

**if** (arr==**null**) {

**return** **false**;

}

**if** (arr.length==0||arr.length==1) {

**return** **false**;

}

**if** (arr[0]==2) {

**if** (arr[1]!=2) {

**return** **false**;

}

}

**if** (arr[arr.length-1]==2) {

**if** (arr[arr.length-2]!=2) {

**return** **false**;

}

}

**for** (**int** i = 1; i < arr.length-1; i++) {

**if** (arr[i]==2) {

**if** (arr[i-1]!=2 && arr[i+1]!=2) {

**return** **false**;

}**else**{

**continue**;

}

}

}

**return** **true**;

}

### 19.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**fizzArray**;

  给一个任意的数字，返回创建一个长度是该数字的整型数组，该数组中的元素按自然顺序递增，  
例如数字是7，那么返回长度为7的整型数组，数组的元素为{0，1，2，3，4，5，6}

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| fizzArray(4) | [0,1,2,3] |
| fizzArray(1) | [0] |
| fizzArray(10) | [0,1,2,3,4,5,6,7,8,9] |

**public** **class** Itheima{

**public** **int**[] fizzArray(**int** a){

**int**[] arr = **new** **int**[a];

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

arr[i] = i;

}

**return** arr;

}

}

### 20.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**fizzArray2**;

  给一个任意的数字，返回创建一个长度是该数字的字符串数组，该数组中的元素按自然顺序递增，  
例如数字是4，那么返回长度为4的字符串数组，数组的元素为{“0”，“1”，“2”，“3”}

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| fizzArray2(4) | ["0","1","2","3"] |
| fizzArray2(10) | ["0","1","2","3","4","5","6","7","8","9"] |
| fizzArray2(2) | ["0","1"] |

**public** **class** Itheima {

**public** String[] fizzArray2(**int** num){

String[] str = **new** String[num];

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

str[i] = i + "";

}

**return** str;

}

}

### 21.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**roundSum**;

  方法接受3个int类型的参数,每个参数的末位数进行四舍五入,例如15末尾四舍五入后就是20,12四舍五入后就是10  
把三个int参数四舍五入后的值求和返回. 为了避免代码重复，编写一个单独的帮助器“public int round10（int num）并调用它3次。完全编写帮助程序，并在与roundSum（）相同的缩进级别。

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| roundSum(16,17,18) | 60 |
| roundSum(12,13,14) | 30 |
| roundSum(6,4,4) | 10 |

**public** **class** Itheima{

**public** **int** roundSum(**int** a,**int** b,**int** c){

**int** sum =round10(a)+round10(b)+round10(c);

**return** sum;

}

**public** **int** round10(**int** num){

**if**(num %10>=5){

**return** num = num-num%10+10;

}

**return** num=num-num%10;

}

}

### 22.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**shiftleft**;

  给一个整型的数组，将数组的元素左移一位后返回一个新的数组，  
例如{6,2,5,3}返回{2,5,3,6}

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| shiftleft([6,2,5,3]) | [2,5,3,6] |
| shiftleft([1,2]) | [2,1] |
| shiftleft([1]) | [1] |

**public** **class** Itheima{

**public** **int**[] shiftleft(**int**[] arr){

**for** (**int** i = 0; i < arr.length-1; i++) {

**int** temp =arr[i];

arr[i] = arr[i+1];

arr[i+1]=temp;

}

**return** arr;

}

}

### 23.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**luckySum**;

  给定3个int值，A B C，返回他们的总和，然而，如果其中有一个值为13，则13和其右边的数都不参与运算。  
举例来说，如果B是13，那么B和C都不算。

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| luckySum(1,2,3) | 6 |
| luckySum(1,2,13) | 3 |
| luckySum(1,13,3) | 1 |

**public** **class** Itheima{

**public** **int** luckySum(**int** A, **int** B, **int** C) {

**if** (A == 13) {

**return** 0;

}

**if** (B == 13) {

**return** A;

}

**if** (C == 13) {

**return** A+B;

}

**return** A+B+C;

}

}

### 24.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**haveThree**;

  给定一个整型数组，如果元素3在数组中，间隔的恰好出现3次，就返回true。

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| haveThree([3,1,3,1,3]) | true |
| haveThree([3,1,3,3]) | false |
| haveThree([3,4,3,3,4]) | false |

**public** **class** Itheima {

//长度至少有[3,1,3,1,3]，5

**public** **boolean** haveThree(**int**[] arr) {

//整型数组为空，或者长度小于5，直接返回false

**if**(arr == **null** || arr.length < 5)

**return** **false**;

//计数

**int** count = 0;

//判断第一个有没有单独的3

**if**(arr[0] == 3 && arr[1] != 3)

count++;

//判断后面有没有单独的3

**for** (**int** i = 1; i < arr.length - 1; i++) {

**if**(arr[i - 1] != 3 && arr[i] == 3 && arr[i + 1] != 3)

count++;

}

//判断末尾是不是单独的3

**if**(arr[arr.length - 2] != 3 && arr[arr.length - 1] == 3)

count++;

**if**(count == 3)

**return** **true**;

**else**

**return** **false**;

}

}

### 25.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**equalIsNot**;

  给定一个字符串，如果字符串中出现的“is”和“not”个数相等，则返回true，否则返回false；

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| equalIsNot("This is not") | false |
| equalIsNot("This is notnot") | true |
| equalIsNot("noisxxnotyynotxisi") | true |

**public** **class** Itheima {

**public** **boolean** equalIsNot(String str ) {

String str1=str;

**int** count = 0;

**int** count1 = 0;

**int** index = 0;

**int** index1 = 0;

String s1 = "not";

String s = "is";

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

**if**(( index = str.indexOf(s)) != -1){

count ++;

str = str.substring(index+s.length());

}

}

**for** (**int** i = 0; i < str1.length(); i++) {

**if**(( index1 = str1.indexOf(s1)) != -1){

count1 ++;

str1 = str1.substring(index1+s1.length());

}

}

**if** (count == count1) {

**return** **true**;

}

**return** **false** ;

}

}

方式二：

**class** Itheima{

**public** **boolean** equalIsNot(String s){// 10

String s1 = s.replaceAll("not",""); // 4

String s2 = s.replaceAll("is", ""); // 8

**int** n=(s.length()-s1.length())/"not".length();

**int** i=(s.length()-s2.length())/"is".length();

**if**(i==n){

**return** **true**;

}

**return** **false**;

}

}

### 26.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**countEvens**;

  方法中传入一个int类型的数据,要求统计数组中偶数的个数,作为返回值return  
可以用%进行求余操作,来判断是否是偶数,例如5%2=1

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| countEvens([2,1,2,3,4]) | 3 |
| countEvens([2,2,0]) | 3 |
| countEvens([1,3,5]) | 0 |

**public** **class** Itheima{

**public** **int** countEvens(**int**[] arr) {

**int** count = 0;

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

**if** (arr[i]%2 == 0) {

count++;

}

}**return** count;

}

}

### 27.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**either24**;

  给定一个整型数组，如果数组中含有两个相邻的数字2，或者是数字4，就返回true，  
否则返回false，如果数组既含有两个相邻的数字2又含有两个相邻的数字4，就返回false。  
例如：([1, 2, 3, 2, 2, 4, 4]) -> false

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| either24([1,2,2]) | true |
| either24([4,4,1]) | true |
| either24([4,4,1,2,2]) | false |

**public** **class** Itheima{

**public** **boolean** either24(**int**[] arr){

**boolean** a = **false**;

**boolean** b = **false**;

**for** (**int** i = 0; i < arr.length-1; i++) {

**if** (arr[i] ==2 &&arr[i+1] ==2 ) {

a = **true**;

}

**if** (arr[i]==4&&arr[i+1]==4) {

b= **true**;

}

}

**if** (b== **false**&& a==**false**) {

**return** **false**;

}

**if** (b== **true**&& a==**true**) {

**return** **false**;

}

**else**{

**return** **true**;

}

}

}

### 28.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**evenlySpaced**;

  给定三个int，a b c，其中一个是小的，一个是中等，一个大。  
如果三个值均匀分布，就是小和中值之间的差异与中和大之间的差异相同，则返回true，

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| evenlySpaced(2,4,6) | true |
| evenlySpaced(4,6,2) | true |
| evenlySpaced(4,6,3) | false |

**public** **class** Itheima {

**public** **boolean** evenlySpaced(**int** a,**int** b,**int** c){

**if**(b-a == c-b){

**return** **true**;

}**else** **if**(b-c == a-b){

**return** **true**;

}**else** **if**(a-b == c-a){

**return** **true**;

}**else** **if**(a-c == b-a){

**return** **true**;

}**else** **if**(c-a == b-c){

**return** **true**;

}**else** **if**(c-b == a-c){

**return** **true**;

}

**return** **false**;

}

}

### 29.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**countTriple**;

  “三元组”是指一个连续出现的三次的字符，返回给定字符串中三元组的个数，  
“三元组”是可以重叠的：“AAAA”包含两个三元组

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| countTriple("abcXXXabc") | 1 |
| countTriple("xxxabyyyycd") | 3 |
| countTriple("a") | 0 |

**public** **class** Itheima {

**public** **int** countTriple(String str){

**int** count=0;

Pattern compile = Pattern.*compile*("(.)\\1{2,}");

Matcher ma = compile.matcher(str);

**while**(ma.find()){

String ge = ma.group();

**if**(ge.length()>3){

count =count + ge.length()-2;

}**else**{

count++;

}

}

**return** count;

}

}

### 30.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**noTeenSum**;

  给定3个int值，A B C，返回他们的总和。然而，如果有任何一个值是青值，则不参与运算，  
青值的范围是13～19之间且不包含15和16；  
可以定义一个公共的方法fixteen（int n），接受一个int类型的值，并判断是否为青值；这样就避免写三次判断的代码，

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| noTeenSum(1,2,3) | 6 |
| noTeenSum(2,13,1) | 3 |
| noTeenSum(2,1,14) | 3 |

**public** **class** Itheima {

**public** **int** noTeenSum(**int** A, **int** B, **int** C) {

**boolean** a = *fixteen*(A);

**boolean** b = *fixteen*(B);

**boolean** c = *fixteen*(C);

**return** (a == **true** ? 0 : A) + (b == **true** ? 0 : B) + (c == **true** ? 0 : C);

}

**private** **static** **boolean** fixteen(**int** a) {

**if** (a >= 13 && a <= 19 && a != 15 && a != 16) {

**return** **true**;

} **else**

**return** **false**;

}

}

### 31.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**has12**;

  给定一个整型数组，如果数组中有1元素, 并且在1元素后面的元素中还有2元素,   
则返回true, 否则返回false.

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| has12([1,3,2]) | true |
| has12([3,1,2]) | true |
| has12([3,1,4,5,2]) | true |

**public** **class** Itheima {

**public** **boolean** has12(**int** arr[]){

String s="";

**for** (**int** i : arr) {

s= s+i;

}

**int** a =s.indexOf("1");

**if**(a!=-1 ){

**if**(s.substring(a).contains("2"))

**return** **true**;

}

**return** **false**;

}

}

### 32.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**topping3**;

  修改接收到的map集合, 如果map集合中包含“potato”，就增加一个键值对 key的名称为 “fries”，value和“potato”具有相同的值  
map集合中包含“salad”，就增加键值对, key的名字为“spinach”，value和“salad”具有相同的值。

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| topping3({"potato":"ketchup"}) | {"fries":"ketchup","potato":"ketchup"} |
| topping3({"potato":"butter"}) | {"fries":"butter","potato":"butter"} |
| topping3({"salad":"oil","potato":"ketchup"}) | {"salad":"oil","fries":"ketchup","spinach":"oil","potato":"ketchup"} |

**public** **class** Itheima {

**public** Map<String,String> topping3(Map<String,String> map){

**if**(map.containsKey("potato")){

map.put("fries",map.get("potato"));

}

**if**(map.containsKey("salad")){

map.put("spinach",map.get("salad"));

}

**return** map;

}

}

### 33.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**wordLen**;

  给定一个字符串数组，返回一个Map <STRING，INTEGER>把数组中每个不同字符串作为key，值为该字符串的长度。

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| wordLen(["a","bb","a","bb"]) | {"a":1,"bb":2} |
| wordLen(["this","and","that","and"]) | {"that":4,"this":4,"and":3} |
| wordLen(["code","code","code","bug"]) | {"bug":3,"code":4} |

**public** **class** Itheima {

**public** **void** main(String[] args) {

String[] arr={"code","code","code","bug"};

Map<String, Integer> len = wordLen(arr);

System.*out*.println(len);

}

**public** Map<String,Integer> wordLen(String[] arr){

Map<String,Integer> map=**new** HashMap<>();

**for** (String s: arr) {

String key=s;

**int** value=s.length();

map.put(key, value);

}

**return** map;

}

}

### 34.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**sumNumbers**;

  给定一字符串，求出现在字符串中的数字之和。  
例如：sumNumbers("abc123xyz") → 123

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| sumNumbers("abc123xyz") | 123 |
| sumNumbers("aa11b33") | 44 |
| sumNumbers("7 11") | 18 |

**public** **class** Itheima {

**public** **int** sumNumbers(String s) {

ArrayList<String> list = **new** ArrayList<String>();

String[] arr = s.split("\\D");

**for**(String string : arr) {

list.add(string);

}

**int** sum = 0;

**for**(String line : list) {

**if**(!line.matches("")) {

**int** i = Integer.*parseInt*(line);

sum=sum+i;

}

}

**return** sum;

}

}

### 35.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**sameEnds**;

  给定一个整型数组array和一个整型数字n，如果数组中的头n个数字和结尾的n个数字相同  
就返回true，否则返回false，如果n等于0的时候，不管任意数组都返回true。  
例如：([1, 2, 5, 2, 1], 0) -> true

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| sameEnds([5,6,45,99,13,5,6],1) | false |
| sameEnds([1,2,5,2,1],1) | true |
| sameEnds([1,2,5,2,1],2) | false |

**public** **class** Itheima {

**public** **boolean** sameEnds(**int**[] arr, **int** i) {

**if**(i == 0)

**return** **true**;

**if**(i >= arr.length)

i = arr.length;

ArrayList<Integer> al = **new** ArrayList<>();

**for** (**int** j = arr.length - i; j < arr.length; j++) {

al.add(arr[j]);

}

**for** (**int** j = 0; j < i ; j++) {

**if**(arr[j] != al.get(**new** Integer(j)))

**return** **false**;

}

**return** **true**;

}

}

### 36.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**mapBully**;

  修改并返回给定映射如下：如果键“a”具有值，则将键“b”设置为“a”的值，  
并将键“a”设置为值“”，取值并用空字符串替换它。如果不存在键“a”将原路返回。

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| mapBully({"b":"dirt","a":"candy"}) | {"b":"candy","a":""} |
| mapBully({"a":"candy"}) | {"b":"candy","a":""} |
| mapBully({"b":"carrot","c":"meh","a":"candy"}) | {"b":"candy","c":"meh","a":""} |

**public** **class** Itheima{

**public** Map<String, String> mapBully(Map<String, String> map){

**if** (map.containsKey("a")&&map.get("a")!=**null**) {

map.put("b", map.get("a"));

map.put("a", "");

}

**return** map;

}

}

### 37.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**lucky13**;

  给定一个整型数组，如果数组不包含1或3，则返回true。

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| lucky13([0,2,4]) | true |
| lucky13([1,2,3]) | false |
| lucky13([1,2,4]) | false |

**public** **class** Itheima {

**public** **boolean** lucky13(**int**[] arr) {

**boolean** b = **true**;

**for** (**int** i : arr) {

**if**(i != 1){

**if**(i != 3){

**continue**;

}**else**{

b = **false**;

**break**;

}

}**else**{

b = **false**;

**break**;

}

}

**return** b;

}

}

### 38.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**makeChocolate**;

  我们要做一包规定质量的巧克力，有每个一公斤和每个五公斤的巧克力棒，  
方法中有三个参数，第一个是小巧克力棒的个数，第二个是大巧克力棒的个数，第三个是规定的质量。  
假设我们总在使用小巧克力棒之前使用大巧克力棒，返回要使用多少个小巧克力棒才能完成规定的质量，如果不能完成则返回-1；

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| makeChocolate(4,1,9) | 4 |
| makeChocolate(4,1,10) | -1 |
| makeChocolate(4,1,7) | 2 |

**public** **class** Itheima {

**public** **int** makeChocolate(**int** xiao, **int** da, **int** zl) {

**int** sum = xiao + da \* 5;

**if** (zl <= sum) {

**if** (zl <= da \* 5)

zl = zl % 5;

**else**

zl -= da \* 5;

**for** (**int** i = 0; i <= xiao; i++) {

**if** (zl <= 0)

**return** i;

zl -= 1;

}

}

**return** -1;

}

}

方式二：

**f** (zl < da \* 5)

zl %= 5; // z1 = 1

**else**

zl -= da \* 5; //

**if** (zl > xiao)

**return** -1;

**else**

**return** zl;

### 39.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**bifDiff**;

  给定一个数组长度为1或更大的整型数组，返回数组中最大值和最小值之间的差值。

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| bifDiff([10,3,5,6]) | 7 |
| bifDiff([7,2,10,9]) | 8 |
| bifDiff([2,10,7,2]) | 8 |

**public** **class** Itheima{

**public** **int** bifDiff(**int**[] arr){

Arrays.*sort*(arr);

**int** max = arr[arr.length-1];

**int** min = arr[0];

**return** max-min;

}

}

### 40.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**no14**;

  给定一个整型数组，如果它同时包含1和4, 返回false,否则返回true;

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| no14([1,2,3]) | true |
| no14([1,2,3,4]) | false |
| no14([2,3,4]) | true |

**public** **class** Itheima {

**public** **boolean** no14(**int**[] arr){

ArrayList<Integer> list = **new** ArrayList<>();

**for**(**int** i : arr){

list.add(i);

}

**if**(list.contains(1) && list.contains(4)){

**return** **false**;

}

**return** **true**;

}

}

### 41.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**sum13**;

  有一个整型数组，返回的结果是数组所有元素的总和，如果数组中出现整型数13的时候，  
那么元素13和13后面的一个元素也不计入总和里面。

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| sum13([1,2,2,1]) | 6 |
| sum13([1,1]) | 2 |
| sum13([1,2,2,1,13]) | 6 |

**public** **class** Itheima{

**public** **int** sum13(**int**[] arr){

**int** sum=0;

**for** (**int** i = 0; i <=arr.length-1; i++) {

**if**(arr[i]!=13){

sum=sum+arr[i];

}**else**{

i=i+1;

}

}

**return** sum;

}

}

### 42.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**mapShare**;

  修改并返回给定map集合的键值对(key-value对)如下：如果key “a”具有值，则将key “b”设置为具有key “a”相同的值。  
在任何情况下如果有key “c”，就删除掉这个键值对 ，其余部分不变。

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| mapShare({"b":"bbb","c":"ccc","a":"aaa"}) | {"b":"aaa","a":"aaa"} |
| mapShare({"b":"xyz","c":"ccc"}) | {"b":"xyz"} |
| mapShare({"d":"hi","c":"meh","a":"aaa"}) | {"d":"hi","b":"aaa","a":"aaa"} |

**public** **class** Itheima{

**public** Map<String,String> mapShare(Map<String,String> map){

**if**(map.containsKey("a")){

map.put("b", map.get("a"));

}**else**{

map.put("b", map.get("b"));

}

**if**(map.containsKey("c")){

map.remove("c");

}

**return** map;

}

}

### 43.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**blackjack**;

  给定两大大于0的int类型值，返回两个数中不大于21且最接近21的数，如果两个数都大于21，则返回0；

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| blackjack(19,21) | 21 |
| blackjack(21,19) | 21 |
| blackjack(19,22) | 19 |

**public** **class** Itheima {

**public** **int** blackjack(**int** x,**int** y) {

**int** num = 0;

**if**(x>21&&y>21)

num=0;

**if** (x<=21&&y<=21)

num = (21-x)>(21-y)?y:x;

**if**(x<=21&&y>21)

num=x;

**if**(x>21&&y<=21)

num=y;

**return** num;

}

}

### 44.题目描述:

  书写一个工具类，类名为**SampleUtils**,

  书写一个工具方法，方法名为**conversion**,

  要求此方法能将传入的**整型数字参数**进行处理，将处理后得到的**字符串返回**给此方法;

  处理方式为:

    例如：传入整型数字为: **12345**

    要求处理后为: **一万二千三百四十五**

    即 **将阿拉伯数字转换为中文显示方式**

### 提示:

  1. **通常工具类中工具的方法都为公有静态方法**

  2. [0123456789]分别对应[零一二三四五六七八九]

  3. 中文的标准位数: 十百千万

  4. 数字判断只需要做到5位数

public class SampleUtils{

public static String conversion(int a){

TreeMap<Integer, String> tm = new TreeMap<>();

tm.put(0, "零");

tm.put(1, "一");

tm.put(2, "二");

tm.put(3, "三");

tm.put(4, "四");

tm.put(5, "五");

tm.put(6, "六");

tm.put(7, "七");

tm.put(8, "八");

tm.put(9, "九");

StringBuilder sb = new StringBuilder();

sb.append(a/10000==0?"":tm.get(a/10000%10)+"万");

sb.append(a/1000==0?"":tm.get(a/1000%10)+"千");

sb.append(a/100==0?"":tm.get(a/100%10)+"百");

sb.append(a/10==0?"":tm.get(a/10%10)+"十");

sb.append(a/1==0?"":tm.get(a/1%10)+"");

return sb.toString();

}

}

### 45.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**mirrorEnds**;

  给定一个字符串，在给定字符串的开始和结束时寻找一个镜像（向后）的字符串。  
换句话说，零或多个字符在一个给定字符串的逆序，在字符串的结尾（可能有重叠）。例如，字符串“abxyzba”镜子“AB”的结束。

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| mirrorEnds("abXYZba") | "ab" |
| mirrorEnds("abca") | "a" |
| mirrorEnds("aba") | "aba" |

**public** **class** Itheima{

**public** String mirrorEnds(String s) {

**int** index = 0;

**int** count = 0;

**for** (**int** i = 0, j = s.length() - 1; i <= j; i++, j--) {

**if**(s.charAt(i) == s.charAt(j))

count++;

**else** **if**(s.charAt(i) != s.charAt(j)) {

index = i;

**break**;

}

}

String string = "";

//如果遍历完了每对都相等，则返回原始字符串

**if**(count == (s.length() + 1 )/ 2)

string = s;

//如果没有遍历完就发现不想等的一对，则截取字符串

**else** **if**(index != 0)

string = s.substring(0, index);

**else**

string = "";

**return** string;

}

}

### 46.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**word0**;

  给定一个字符串数组，返回一个Map <STRING，INTEGER>，  
把数组中每个不同字符作为key，value都是0.例如，字符串数组["h","e","l","l","o"]键是"h" "e" "l" "o"，值是0。

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| word0(["a","b","a","b"]) | {"b":0,"a":0} |
| word0(["a","b","a","c","b"]) | {"b":0,"c":0,"a":0} |
| word0(["c","b","a"]) | {"b":0,"c":0,"a":0} |

**public** **class** Itheima{

**public** Map<String,Integer > word0(String[]arr){

HashMap<String, Integer> hm = **new** HashMap<>();

**for** (String s : arr) {

hm.put(s,hm.containsKey(s)?0:0);

}**return** hm;

}

}

### 47.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**withoutString**;

  给定两个字符串，base和remove，返回删除了remove字符串的base字符串（不区分大小写），  
并且返回的base字符串不含有remove的重叠事例。

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| withoutString("Hello there","llo") | "He there" |
| withoutString("Hello there","e") | "Hllo thr" |
| withoutString("Hello there","x") | "Hello there" |

**public** **class** Itheima{

**public** String withoutString(String s1,String s2){

**for** (**int** i = 0; i < s1.length()-s2.length(); i++) {

String sub = s1.substring(i, s2.length()+i);

**if** (sub.equalsIgnoreCase(s2)) {

s1 = s1.replaceAll(sub, "");

}

}

**return** s1;

}

}

### 48.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**has77**;

  给定一个整型数组，如果数组包含两个彼此相邻的7，或者有两个7由一个元素分隔，如{7，1，7}，则返回true。

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| has77([1,7,7]) | true |
| has77([1,7,1,7]) | true |
| has77([1,7,1,1,7]) | false |

**public** **class** Itheima {

**public** **static** **boolean** has77(**int**[] arr) {

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

**if** (i + 1 <= arr.length - 1 && arr[i] == 7 && arr[i + 1] == 7) {

**return** **true**;

} **else** **if** (i + 2 <= arr.length - 1 && arr[i] == 7

&& arr[i] == arr[i + 2] && arr[i + 1] != 7) {

**return** **true**;

}

}

**return** **false**;

}

}

### 49.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**sumDigits**;

  给定一个字符串，返回字符串中包涵的单个数字的之和，如果不含有数字则返回0；  
如果给定的字符串只包涵一个数字，需要Integer.parseInt(string)将其转化为int类型

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| sumDigits("aa1bc2d3") | 6 |
| sumDigits("aa11b33") | 8 |
| sumDigits("Chocolate") | 0 |

**public** **class** Itheima{

**public** **int** sumDigits(String s){

**int** sum = 0;

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

**if** (s.charAt(i) <= '9' && s.charAt(i) >= 0) {

String str = **new** Character(s.charAt(i)).toString();

sum = sum + Integer.*parseInt*(str);

}

}

**return** sum;

}

}

### 50.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**firstChar**;

  给定一个非空字符串数组，为每个不同的第一个字符返回一个Map <STRING，STRING>，  
键为字符串的首个字符，值为含有该键首个字符的所有字符串进行追加，例如：["salt","tea","soda","toast"]字符串数组  
返回结果是{“t”:"teatoast","s":"saltsoda"}，键是不同字符串的首个字符，值是按它们在数组中的顺序追加在一起。

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| firstChar(["salt","tea","soda","toast"]) | {"t":"teatoast","s":"saltsoda"} |
| firstChar(["aa","bb","cc","aAA","cCC","d"]) | {"d":"d","b":"bb","c":"cccCC","a":"aaaAA"} |
| firstChar([]) | {} |

**public** **class** Itheima {

**public** Map<String,String> firstChar(String[] arr){

Map<String,String> map=**new** HashMap<>();

**if**(arr==**null**){

**return** map;

}

**for** (String s : arr) {

**char** c=s.charAt(0);

String key=c+"";

String value=s;

**if**(!map.containsKey(key)){

map.put(key, value);

}**else**{

map.put(key, map.get(key)+value);

}

}

**return** map;

}

}

### 51.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**wordCount**;

  给定一个字符串数组，返回一个Map <STRING，INTEGER>，每个不同的字符串作为key  
value为字符串在数组中出现的次数。

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| wordCount(["a","b","a","c","b"]) | {"b":2,"c":1,"a":2} |
| wordCount(["c","b","a"]) | {"b":1,"c":1,"a":1} |
| wordCount(["c","c","c","c"]) | {"c":4} |

**public** **class** Itheima{

**public** Map<String, Integer> wordCount(String[] arr){

Map<String, Integer> map = **new** HashMap<>();

**for** (String s : arr) {

map.put(s, map.containsKey(s)?map.get(s)+1:1);

}

**return** map;

}

}

### 52.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**sum67**;

  有一个整型数组，返回数组中的数字的总和，如果数组含有数6和7  
那么忽略不计入从6开始并延伸到有7的数字段（每6个将跟随至少一个7）。  
返回0表示没有数字。

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| sum67([1,2,2]) | 5 |
| sum67([1,2,2,6,99,99,7]) | 5 |
| sum67([1,1,6,7,2]) | 4 |

**public** **class** Itheima{

**public** **int** sum67(**int**[] arr){

**int** num = 0;

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

**if**(arr[i] != 6){

num += arr[i];

}**else**{

i++;

**while**(**true**){

**if**(arr[i] == 7){

**break**;

}

i++;

}

}

}

**return** num;

}

}

### 53.题目描述:

  书写一个工具类，类名为**StringUtils**;

### 要求:

  1. 书写**reverse**方法，接收String类型的参数，要求将接收的字符串进行反转操作，将反转后的字符串返回。

    例如: 传入字符串 **abcdef**，返回的字符串为 **fedcba**

  2. 书写**filter**方法，接收String类型的参数，要求将接收的字符串中的数字过滤掉，将过滤后的字符串返回。

例如: 传入字符串 abc123def，返回的字符串为 abcdef

**public** **class** StringUtils{

**private** StringUtils(){};

**public** **static** String reverse(String s){

StringBuilder sb = **new** StringBuilder(s);

**return** sb.reverse().toString();

}

**public** **static** String filter(String s){

String regex = "\\d";

**return** s.replaceAll(regex, "");

}

}

### 54.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**tripleUp**;

  给定一个整型的数组，如果某索引有三个相邻的递增的自然数，就返回true  
例如：{...4,5,6...}或{...23,24,25}等等。

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| tripleUp([1,4,5,6,2]) | true |
| tripleUp([1,2,3]) | true |
| tripleUp([1,2,4]) | false |

**public** **class** Itheima{

**public** **boolean** tripleUp(**int** [] arr){

**for** (**int** i = 0; i < arr.length-2; i++) {

**if** (arr[i]==arr[i+1]-1&&arr[i]==arr[i+2]-2) {

**return** **true**;

}

}

**return** **false**;

}

}

### 55.题目描述:

  书写一个类，类名为**Itheima**;

  类中有一个方法，方法名**makeBricks**;

  我们想做一排砖（长度自己定），我们有一些小砖（每块1英寸）和大砖（每块5英寸），  
如果用我们选择的砖块的数量能够拼接成功，则返回true；否则返回false，  
例如：makeBricks(3, 1, 8) → true

### 提示:

|  |  |
| --- | --- |
| **方法调用** | **期望值** |
| makeBricks(3,1,8) | true |
| makeBricks(3,1,9) | false |
| makeBricks(3,2,10) | true |

**public** **class** Itheima{

**public** **boolean** makeBricks(**int** small, **int** big, **int** len) {

//思路:先拿大的去填,大的不够,再拿小的去填

// a是代表自定义长度需要多少块5的长度之后还需要缺多少长度

**int** a = len % 5;

//b是代表自定长度需要多少块5的长度整砖

**int** b = len / 5;

//如果自定义需要的最大5长度的大于我们有的5的长度

**if** (b > big) {

//大的已经不够,就拿小的填,因为小的长度为1,大的加上小的够定义长度就肯定可以满足

**if** (len <= big \* 5 + small)

**return** **true**;

//大的不够小的也不够

**else**

**return** **false**;

} **else** {

//大的够了,自定义长度可能不是5的倍数,%5后的剩下的就得拿小的去填

//如果小的不够,就false

**if** (a > small)

**return** **false**;

//小的够,就true

**else**

**return** **true**;

}

}

}