1．现在输入 n 个数字，以逗号，分开；然后可选择升或者降序排序

我的评论：

本题的 splitStringByComma(String)方法纯属多余，可以用 String 的 split 方法一句话代替，且可读性也更强，下面的一段话源自 JDK1.6API，StringTokenizer 类已不再提倡使用，保留仅为旧代码。

StringTokenizer 是出于兼容性的原因而被保留的遗留类(虽然在新代码中并不鼓励使用它)。建议所有寻求此功能的人使用 String 的 split 方法或 java.util.regex 包。

```
import java.util.*;

public class bycomma{

    public static String[] splitStringByComma(String source){

        if(source==null||source.trim().equals(""))

            return null;

       StringTokenizer commaToker = new StringTokenizer(source,",");

       String[] result = new String[commaToker.countTokens()];

        int i=0;

        while(commaToker.hasMoreTokens()){

            result[i] = commaToker.nextToken();

            i++;

         }

        return result;

    }

    public static void main(String args[]){

        String[] s = splitStringByComma("5,8,7,4,3,9,1");

          int[] ii = new int[s.length];

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

```
                ii[i] =Integer.parseInt(s[i]);
            }
        Arrays.sort(ii);
        //asc
        for(int i=0;i<ii.length;i++ ){
            System.out.println(ii[i]);
          }
       //desc
        for(int i=(s.length-1);i>=0;i--){
            System.out.println(ii[i]);
       }
    }
}
```

2.编写一个截取字符串的函数，输入为一个字符串和字节数，输出为按字节截取的字符串。 但是要保证汉字不被截半个，如"我 ABC"4，应该截为"我 AB"，输入"我 ABC 汉 DEF"，6，应该输出为"我 ABC"而不是"我 ABC+汉的半个"。

代码：

```
public static boolean isLetter(char c){
    int k=0X80;
    return c/k==0?true:false;

    }

    public static int lengths(String strSrc){
    if (strSrc==null){
```

```
        return 0;
    }
    int len=0;
    char[] strChar=strSrc.toCharArray();
    for (int i=0;i<strChar.length;i++){
        len++;
        if (!isLetter(strChar[i])) len++;
    }
    return len;
    }

    public static String subString(String origin,int len){
    if (origin==null || origin.equals("")|| len<1){
        return "";
    }
    if (len>lengths(origin)){
        return  origin;
    }
    byte[] strByte=new byte[len];
    System.arraycopy(origin.getBytes(),0,strByte,0,len);
    int count=0;
    for (int i=0;i<len;i++){
        int value=(int)strByte[i];
        if (value<0) count++;
```

```
        }
        if (count % 2 !=0){
              //len=(len==1)?++len:--len;
                --len;
        }
        return new String(strByte,0,len);
        }
    public static void main(String[] args) {
                    System.out.println(""+ subString("我 ABC 汉 DEF",6));
    }
```

3、排序都有哪几种方法？请列举。用 JAVA 实现一个快速排序。

排序的方法有：插入排序（直接插入排序、希尔排序），交换排序（冒泡排序、快速排序），选择排序（直接选择排序、堆排序），归并排序，分配排序（箱排序、基数排序）

快速排序的伪代码。

/ /使用快速排序方法对 a[0 :n- 1]排序从 a[0 :n- 1]中选择一个元素作为 m I d d l e，该元素为支点把余下的元素分割为两段 left 和 r I g h t，使得 l e f t 中的元素都小于等于支点，而 right 中的元素都大于等于支点递归地使用快速排序方法对 left 进行排序递归地使用快速排序方法对 right 进行排序所得结果为 l e f t + m I d d l e + r I g h t

//以下为 java 程序实现的快速排序算法：

```
    public static void sort(int[] data) {
              quickSort(data,0,data.length-1);
    }
    public static void quickSort(int[] data,int low,int high){
          int pivotIndex=(low+high)/2;
           swap(data,pivotIndex,high);
          int k=partition(data,low-1,high,data[high]);
```

```
            swap(data,k,high);
        if ((k-low)>1) partition(data,low,k-1);
        if ((high-k)>1) partition(data,k+1,high);
}
public static int partition(int[] data int low,int high, int pivot ){
         do {
                  while (data[++low]<pivot) ;
                  while (high!=0      && data[--high]>pivot);
                   swap(data,low,high);
         }
         while (low<high) ;
         swap(data,low,high);
         return low;
}
public static void swap(int[] data int low,int high){
        int tmp=data[low];
        data[low]=data[high];
        data[high]=tmp;
}
public static void main(String[] args){
     int[] data = new int[]{89,32,425,32,78,1,53,92};
     sort(data);
}
```

4.试用递归的方法写一下计算菲波那契数列的通项 f(n),已知 f1=1,f2=1,以后每项都是前

两项的和。

.....

public static long fibonacci(long m){

if (m==0 || m==1) return m;

else return fibonacci(m-1)+fibonacci(m-2);

}

5. 写一个 Singleton 出来。

Singleton 模式主要作用是保证在 Java 应用程序中，一个类 Class 只有一个实例存在。

我的评论：第一种形式是饿汉式单例类，第二种是懒汉式单例类；可以如此速记，饿汉式太饿了，所以迫不及待在内部 new 出一个实例，而懒汉式太懒了，所以知道应用时才检查有没有实例存在，如不存在才 new 一个实例出来。

一般 Singleton 模式通常有几种种形式:

第一种形式: 定义一个类，它的构造函数为 private 的，它有一个 static 的 private 的该类变量，在类初始化时实例话，通过一个 public 的 getInstance 方法获取对它的引用,继而调用其中的方法。

```
Public class Singleton {
        private Singleton(){}
        //在自己内部定义自己一个实例，是不是很奇怪？
    //注意这是 private  只供内部调用
    private static Singleton instance = new Singleton();
    //这里提供了一个供外部访问本 class 的静态方法，可以直接访问
    public static Singleton getInstance() {
         return instance;
    }
}
```

第二种形式:

```
public class Singleton {
    private static Singleton instance = null;
    public static synchronized Singleton getInstance() {
        //这个方法比上面有所改进，不用每次都进行生成对象，只是第一次
        //使用时生成实例，提高了效率！
            if (instance==null)
                instance＝new Singleton();
                 return instance;
        }
}
```

其他形式:

定义一个类，它的构造函数为 private 的，所有方法为 static 的。

一般认为第一种形式要更加安全些

6、创建一个静态方法，给它传入一个对象，请循环的打印出该对象所在类的类名和所实现的方法名（华为笔试最后一道编程）

```
import java.lang.reflect.*;
public class Test{
public static void test(Object obj){
        Class clazz=obj.getClass();
        //System.out.println("类名:"+clazz.getName());
        Method[] ms=clazz.getDeclaredMethods();
        long len=Array.getLength(ms);
        for(int i=0;i<len;i++){
```

```
            System.out.println("类名:"+clazz.getName()+"方法名:"+ms[i].getName());
        }
    }
    class A{
        public void b(){}
        public void c(){}
        public void d(){}
        public void e(){}
    }
    public static void main(String[] args){
        Test t=new Test();
        Test.A a=t.new A();
        test(a);
    }
    }
```

7、假设字符串类似这样的 aba 和 aab 就相等，现在随便给你二组字符串，请编程比较他们看是否相等

```
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
String s = null;
try {
s = br.readLine();
} catch (IOException e) {
e.printStackTrace();
}
```

```
StringTokenizer st = new StringTokenizer(s);
String s1 = st.nextToken();
String s2 = st.nextToken();
byte[] sa1 = s1.getBytes();
byte[] sb2 = s2.getBytes();
Arrays.sort(sa1);
Arrays.sort(sb2);
String ss1 = new String(sa1);
String ss2 = new String(sb2);
if(ss1.equals(ss2))
System.out.println("equal");
else
System.out.println("not equal");
```

8、给你一组字符串如：iu7i8hy4jnb2，让你编程输出里面的数字：7842

用正规表达式："iu7i8hy4jnb2".replaceAll("[^\\d]","");

9、给你一组字符串让你把它倒叙输出

```
public static String flashBack(String origin) {
    String result = "";
    for (int i = origin.length(); i > 0; i--) {
      String tmp = origin.substring(i - 1, i);
      result += tmp;
    }
    return result;
}
```

10、给你一组字符如{1，3，4，7，2，1，1，5，2}，让你输出里面出现次数最多且数值最大的一个，出现几次

```
public void fun4() {

    int[] a = { 4, 1, 2, 4, 5, 1, 1, 1, 5, 1, 3, 4, 5 };

    Arrays.sort(a);

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

        System.out.print(a[i] + " ");

    }

    System.out.println();

    int maxNumber = a[a.length - 1], maxCount = 1;

    int curNumber = a[a.length - 1], curCount = 1;

    for (int i = a.length - 1; i > 0; i--) {

        curNumber = a[i];

        if (a[i] == a[i - 1]) {

            curCount++;

        } else {

            System.out.println("i=" + i + ",curCount=" + curCount+ ",maxCount=" + maxCount + ",maxNumber=" + maxNumber);

            if (curCount > maxCount) {

                maxCount = curCount;

                maxNumber = curNumber;

            }

            curCount = 1;

        }

    }
```

```
        if (curCount > maxCount) {
            maxCount = curCount;
            //maxNumber = curNumber;
        }
    System.out.println("curCount=" + curCount + ",maxCount=" + maxCount + ",maxNumber="
+ maxNumber);
    }
```

11、求两个数的公约数，M，N

```
int divisor =1;
for (int i = 2; i <= b; i++) {
if(a%i==0 && b%i==0){
divisor = i;
}
}
System.out.println(a+"和"+b+"的最大公约数是:"+divisor);
}
```

12、实现数组复制

我的理解：这是深复制，数组 a，b 不再有关联

```
public void fun8(){
int[] a = {1,2,3,4,56,7,8};
int[] b = (int[])a.clone();
Conica.print(a);
Conica.print(b);
b[0]=100;
```

```
Conica.print(a);

Conica.print(b);

}
```

13、冒泡排序的实现

```
public void fun9(){

int[] a = {1,5,2,6,8,74,1,25,69,8};

Conica.print(a);

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

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

if(a[j]>a[j+1]){

int temp = a[j];

a[j] = a[j+1];

a[j+1] = temp;

}

}

}

Conica.print(a);

}
```

14、编程显示某一文件目录下的文件名

```
public void fun10(){

File file = new File("G:\\03月份");

if(file.exists()){

if(file.isDirectory()){

String[] files = file.list();
```

```
Conica.println(files);
}
}
}
```

15、从键盘输入4个十进制数字字符，将其转换为4位时间之数并显示出来

16、编程实现统计文本文件中某个单词的出现频率，并输出统计结果

用 HashMap 来解决

假设单词不存在跨行的,每个单词用,. ;分割

```
public static void countNum() throws IOException {
    BufferedReader br = null;
    try {
      br = new BufferedReader(new FileReader("c://file.txt"));
      Map map = new HashMap();
      for (String s = br.readLine(); s != null; s = br.readLine()) {
        StringTokenizer st = new StringTokenizer(s, ",.    ;");
        while (st.hasMoreTokens()) {
          String temp = st.nextToken();
          if (map.containsKey(temp)) {
            map.put(temp, new Integer((Integer)map.get(temp) + 1));
          } else {
            map.put(temp, new Integer(1));
          }
        }
      }
```

```
        for (Iterator it = map.entrySet().iterator(); it.hasNext();) {
          Map.Entry entry = (Map.Entry) it.next();
          System.out.println(entry.getKey() + "-->" + entry.getValue()
              + "times");
        }
      } finally {
        br.close();
      }
}
```

17、编程模仿 DOS 下的 dir 命令，列出某个目录下的内容

18、编程说明 String 和 StringBuffer 字符串的区别

19、编程计算 N!的程序，一个使用递归方法，一个不用递归方法

递归 ：

```
long fuction(int n){
     if (n==0) return 1;
else
         return n* fuction(n-1);
}
```

不递 ：

```
long s=1;
for(int i=2;i<=n;i++)
{
           s*=i;
}
```

20、编程实现 ASCII 码和 Unicode 码之间的转换

21.用1、2、2、3、4、5这六个数字，用 java 写一个 main 函数，打印出所有不同的排列，如：512234、412345等，要求： "4 "不能在第三位， "3 "与 "5 "不能相连.

此题具体算法及程序可参考：

http://topic.csdn.net/u/20070114/14/1170e023-e8f0-4331-8bd8-516c6f1e40da.html

22。一个字符串中可能包含 a~z 中的多个字符，如有重复，如 String data="aavzcadfdsfsdhshgWasdfasdf"，求出现次数最多的那个字母及次数，如有多个重复的则都求出。〔金山公司面试题〕

```
import java.util.ArrayList;

import java.util.Collections;

import java.util.Iterator;

import java.util.TreeSet;

public class FindRepeatChar {

public static void doString(String strInput) {

    char[] chars = strInput.toCharArray();

    ArrayList lists = new ArrayList();

    TreeSet set = new TreeSet();

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

     lists.add(String.valueOf(chars[i]));

     set.add(String.valueOf(chars[i]));

    }

    System.out.println(set);

    Collections.sort(lists);

    System.out.println(lists);

    StringBuffer sb = new StringBuffer();
```

```
for (int i = 0; i < lists.size(); i++) {
  sb.append(lists.get(i));
}
strInput = sb.toString();
System.out.println(strInput);
int max = 0;
String maxString = "";
ArrayList maxList = new ArrayList();
for (Iterator its = set.iterator(); its.hasNext();) {
  String os = (String) its.next();
  int begin = strInput.indexOf(os);
  int end = strInput.lastIndexOf(os);
  int value = end - begin + 1;
  if (value > max && value > 1) {
    max = value;
    maxString = os;
    maxList.add(os);
  } else if (value == max) {
    maxList.add(os);
  }
}
int index = 0;
for (int i = 0; i < maxList.size(); i++) {
  if (maxList.get(i).equals(maxString)) {
```

```
            index = i;
            break;
          }
        }
        System.out.println("出现最多的字符为：");
        for (int i = 0; i < maxList.size(); i++) {
          System.out.println(maxList.get(i) + "");
        }
        System.out.println();
        System.out.println("出现最多的次数为：" + max);
    }
    public static void main(String[] args) {
        String strInput = new String("aavzcadfdsfsdhshgWasdfasdf");
        doString(strInput);
    }
    }
```

23．金额转换，阿拉伯数字的金额转换成中国传统的形式如：（￥1011）—>（一千零一拾一元整）输出。

```
package test.money;
import java.text.NumberFormat;
import java.util.HashMap;
public class SimpleMoneyFormat {
public static final String EMPTY = "";
public static final String ZERO = "零";
```

```
public static final String ONE = "壹";

public static final String TWO = "贰";

public static final String THREE = "叁";

public static final String FOUR = "肆";

public static final String FIVE = "伍";

public static final String SIX = "陆";

public static final String SEVEN = "柒";

public static final String EIGHT = "捌";

public static final String NINE = "玖";

public static final String TEN = "拾";

public static final String HUNDRED = "佰";

public static final String THOUSAND = "仟";

public static final String TEN_THOUSAND = "万";

public static final String HUNDRED_MILLION = "亿";

public static final String YUAN = "元";

public static final String JIAO = "角";

public static final String FEN = "分";

public static final String DOT = ".";

private static SimpleMoneyFormat formatter = null;

private HashMap chineseNumberMap = new HashMap();

private HashMap chineseMoneyPattern = new HashMap();

private NumberFormat numberFormat = NumberFormat.getInstance();

private SimpleMoneyFormat() {

    numberFormat.setMaximumFractionDigits(4);
```

```
        numberFormat.setMinimumFractionDigits(2);
        numberFormat.setGroupingUsed(false);
        chineseNumberMap.put("0", ZERO);
        chineseNumberMap.put("1", ONE);
        chineseNumberMap.put("2", TWO);
        chineseNumberMap.put("3", THREE);
        chineseNumberMap.put("4", FOUR);
        chineseNumberMap.put("5", FIVE);
        chineseNumberMap.put("6", SIX);
        chineseNumberMap.put("7", SEVEN);
        chineseNumberMap.put("8", EIGHT);
        chineseNumberMap.put("9", NINE);
        chineseNumberMap.put(DOT, DOT);
        chineseMoneyPattern.put("1", TEN);
        chineseMoneyPattern.put("2", HUNDRED);
        chineseMoneyPattern.put("3", THOUSAND);
        chineseMoneyPattern.put("4", TEN_THOUSAND);
        chineseMoneyPattern.put("5", TEN);
        chineseMoneyPattern.put("6", HUNDRED);
        chineseMoneyPattern.put("7", THOUSAND);
        chineseMoneyPattern.put("8", HUNDRED_MILLION);
    }
    public synchronized static SimpleMoneyFormat getInstance() {
        if (formatter == null)
```

```
        formatter = new SimpleMoneyFormat();
        return formatter;
    }
    public String format(String moneyStr) {
        checkPrecision(moneyStr);
        String result;
        result = convertToChineseNumber(moneyStr);
        result = addUnitsToChineseMoneyString(result);
        return result;
    }
    public String format(double moneyDouble) {
        return format(numberFormat.format(moneyDouble));
    }
    public String format(int moneyInt) {
        return format(numberFormat.format(moneyInt));
    }

    public String format(long moneyLong) {
        return format(numberFormat.format(moneyLong));
    }
    public String format(Number moneyNum) {
        return format(numberFormat.format(moneyNum));
    }
    private String convertToChineseNumber(String moneyStr) {
```

```
String result;
StringBuffer cMoneyStringBuffer = new StringBuffer();
for (int i = 0; i < moneyStr.length(); i++) {//123363
  cMoneyStringBuffer.append(chineseNumberMap.get(moneyStr.substring(
      i, i + 1)));
}
// 拾佰仟万亿等都是汉字里面才有的单位，加上它们
int indexOfDot = cMoneyStringBuffer.indexOf(DOT);
int moneyPatternCursor = 1;
for (int i = indexOfDot - 1; i > 0; i--) {
  cMoneyStringBuffer.insert(i, chineseMoneyPattern.get(EMPTY
      + moneyPatternCursor));
  moneyPatternCursor = moneyPatternCursor == 8 ? 1
      : moneyPatternCursor + 1;
}
String fractionPart = cMoneyStringBuffer.substring(cMoneyStringBuffer
    .indexOf("."));
cMoneyStringBuffer.delete(cMoneyStringBuffer.indexOf("."),
    cMoneyStringBuffer.length());
while (cMoneyStringBuffer.indexOf("零拾") != -1) {//inclusive. exclusive.
  cMoneyStringBuffer.replace(cMoneyStringBuffer.indexOf("零拾"),
      cMoneyStringBuffer.indexOf("零拾") + 2, ZERO);
}
while (cMoneyStringBuffer.indexOf("零佰") != -1) {
```

```
  cMoneyStringBuffer.replace(cMoneyStringBuffer.indexOf("零佰"),
      cMoneyStringBuffer.indexOf("零佰") + 2, ZERO);
}
while (cMoneyStringBuffer.indexOf("零仟") != -1) {
  cMoneyStringBuffer.replace(cMoneyStringBuffer.indexOf("零仟"),
      cMoneyStringBuffer.indexOf("零仟") + 2, ZERO);
}
while (cMoneyStringBuffer.indexOf("零万") != -1) {
  cMoneyStringBuffer.replace(cMoneyStringBuffer.indexOf("零万"),
      cMoneyStringBuffer.indexOf("零万") + 2, TEN_THOUSAND);
}
while (cMoneyStringBuffer.indexOf("零亿") != -1) {
  cMoneyStringBuffer.replace(cMoneyStringBuffer.indexOf("零亿"),
      cMoneyStringBuffer.indexOf("零亿") + 2, HUNDRED_MILLION);
}
while (cMoneyStringBuffer.indexOf("零零") != -1) {
  cMoneyStringBuffer.replace(cMoneyStringBuffer.indexOf("零零"),
      cMoneyStringBuffer.indexOf("零零") + 2, ZERO);
}
if (cMoneyStringBuffer.lastIndexOf(ZERO) == cMoneyStringBuffer.length() - 1)
  cMoneyStringBuffer.delete(cMoneyStringBuffer.length() - 1,
      cMoneyStringBuffer.length());
cMoneyStringBuffer.append(fractionPart);
result = cMoneyStringBuffer.toString();
```

```
        return result;
    }
    private String addUnitsToChineseMoneyString(String moneyStr) {
        String result;
        StringBuffer cMoneyStringBuffer = new StringBuffer(moneyStr);
        int indexOfDot = cMoneyStringBuffer.indexOf(DOT);
        cMoneyStringBuffer.replace(indexOfDot, indexOfDot + 1, YUAN);
        cMoneyStringBuffer.insert(cMoneyStringBuffer.length() - 1, JIAO);
        cMoneyStringBuffer.insert(cMoneyStringBuffer.length(), FEN);
        if (cMoneyStringBuffer.indexOf("零角零分") != -1)// 没有零头，加整
          cMoneyStringBuffer.replace(cMoneyStringBuffer.indexOf("零角零分"),
              cMoneyStringBuffer.length(), "整");
        else if (cMoneyStringBuffer.indexOf("零分") != -1)// 没有零分，加整
          cMoneyStringBuffer.replace(cMoneyStringBuffer.indexOf("零分"),
              cMoneyStringBuffer.length(), "整");
        else {
          if (cMoneyStringBuffer.indexOf("零角") != -1)
            cMoneyStringBuffer.delete(cMoneyStringBuffer.indexOf("零角"),
                cMoneyStringBuffer.indexOf("零角") + 2);
          // tmpBuffer.append("整");
        }
        result = cMoneyStringBuffer.toString();
        return result;
    }
```

```
    private void checkPrecision(String moneyStr) {//5336.53663      10-5-1

        int fractionDigits = moneyStr.length() - moneyStr.indexOf(DOT) - 1;

        if (fractionDigits > 2)

          throw new RuntimeException("金额" + moneyStr + "的小数位多于两位。"); //  精度不能比分低

    }

    public static void main(String[] args) {

        System.out.println(getInstance().format(new Double(8951.11)));

    }

    }.
```