import java.util.\*;  
public class broken\_line {  
 static class loman implements Comparable<loman>{  
 int line;  
 double length;  
 ArrayList<Double> xCoord;  
 ArrayList<Double> yCoord;  
 public loman(ArrayList<Double>x, ArrayList<Double> y, int l) {  
 line=l;  
 xCoord=new ArrayList<Double>();  
 yCoord=new ArrayList<Double>();  
 xCoord.addAll(x);  
 yCoord.addAll(y);  
 length=countLength(xCoord, yCoord);  
 }  
 private double countLength(ArrayList<Double> x,

ArrayList<Double> y) {  
 *// TODO Auto-generated method stub* double sum;  
 double sqr;  
 double l=0;  
 for (int i=0;i<x.size()-1; i++) {  
 sum=0;  
 sum += ((y.get(i) - y.get(i + 1)) \* (y.get(i) - y.get(i + 1)) +

(x.get(i) - x.get(i + 1)) \* (x.get(i) - x.get(i + 1)));  
 sqr= Math.sqrt(sum);  
 l+=sqr;  
 }  
 return l;  
 }  
 @Override  
 public int compareTo(loman l) {return Double.compare(l.length, length);  
 }  
 public String toString() {  
 String str="";  
 str+="#"+line+" ";  
 for (int i=0; i<xCoord.size(); i++)  
 str+="("+xCoord.get(i)+":"+yCoord.get(i)+") ";  
 str+="length = "+length;  
 return str;  
 }  
 }  
 public static ArrayList<loman> lomanList=new ArrayList<loman>();  
 public static void main(String[] args) {  
 *// TODO Auto-generated method stub* ArrayList<String> text=new ArrayList<String>();  
 Scanner in=new Scanner(System.in);  
 System.out.println("Enter coordinates or press Ctrl+Z to exit");  
 while(in.hasNextLine())  
 text.add(in.nextLine());  
 in.close();  
 System.out.println("entering stopped");  
 for (int i=0; i<text.size(); i++)  
 processLine(text.get(i), i);  
 Collections.sort(lomanList);  
 System.out.println("Soted list of loman");  
 if (lomanList.isEmpty())  
 System.out.println("No loman");  
 for (loman l:lomanList)  
 System.out.println(l);  
 }  
 private static void processLine(String string, int i) {ArrayList<Double>x=new ArrayList<Double>();  
 ArrayList<Double>y=new ArrayList<Double>();  
 string= string.trim();  
 boolean flag=true;  
 int index1=0, index2=0, index3=0;  
 while (string.length()!=0)  
 {  
 index1=string.indexOf('(');  
 if (index1==-1) {  
 flag=false;  
 break;  
 }  
 index2=string.indexOf(')');  
 if (index2==-1) {  
 flag=false;  
 break;  
 }  
 String coord=string.substring(index1+1, index2);  
 index3=coord.indexOf(':');  
 if (index3==-1) {  
 flag=false;  
 break;  
 }  
 try {  
 String sX=coord.substring(0,index3);  
 double cX=Double.parseDouble(sX);  
 String sY=coord.substring(index3+1);  
 double cY=Double.parseDouble(sY);  
 x.add(cX);  
 y.add(cY);  
 }  
 catch(Exception e){  
 flag=false;  
 break;  
 }  
 string=string.substring(index2+1);  
 }  
 if (flag==true)  
 lomanList.add(new loman(x,y,i));  
 }  
}

import java.util.\*;  
public class numbs8\_10\_16 {  
 static class Word {protected String word;  
 protected int line; protected Word (String word, int line) {assert(word!=null); this.word = word;  
 this.line = line;}  
 public String toString() {return new String(word + "(" + line + ")");}  
 static boolean isSigned (String word) {return (word.charAt(0)=='-'||word.charAt(0)=='+');}  
 static boolean isOctal(String word) {final String octals = "01234567"; String w = isSigned(word)?word.substring(1):word; if(w.charAt(0)!='0') {return false;} for(int i = 1;i<w.length();i++) if(octals.indexOf(w.charAt(i))==-1)   
 return false; return true; }}  
 static class MyInt extends Word implements Comparable<MyInt>{ int val;  
 public MyInt(String word, int line, int v) {  
 super(word, line); val = v; }  
 public int compareTo(MyInt p) {  
 return Integer.compare(p.val, val); }  
 static MyInt parseDec(String w, int ln) {  
 try {int x = Integer.parseInt(w);  
 return new MyInt(w, ln, x); }  
 catch(Exception ignored) {}  
 return null; }  
 static MyInt parseHex(String w, int ln) {  
 String word = (w.charAt(0)=='0'&& Character.toLowerCase(w.charAt(1))=='x')?w.substring(2):w; try {int x = Integer.parseInt(word, 16);  
 return new MyInt(w,ln,x); }  
 catch(Exception ignored) {} return null; }  
 public void print() {  
 System.out.print(word+"("+line+")"+ " "); } }static class MyDbl extends Word implements Comparable<MyDbl>{double val;boolean exp;  
 public MyDbl(String word, int line, double v) {  
 super(word,line); val = v; exp = (word.indexOf('e')>0 || word.indexOf('E')>0); }  
 public boolean isExp() {return exp; }  
 public int compareTo(MyDbl p) {return Double.compare(val, p.val); }  
 static MyDbl parse(String w, int ln) {  
 try {double x = Double.parseDouble(w);  
 return new MyDbl(w, ln, x); }  
 catch(Exception ignored) {}  
 return null; }  
 public static boolean isFloat(String word) {  
 return word.contains("f") || word.contains("F");  
 }public void print() {System.out.print(word+"("+line+")"+ " ");}}  
 static class MyWord extends Word implements Comparable<MyWord>{  
 public MyWord(String word, int line) {  
 super(word, line); }  
 public int compareTo(MyWord p) {  
 return p.word.compareTo(this.word); }  
 public void print() { System.out.print(word+"("+line+")"+ " ");}}  
 static List <MyInt> di = new Vector<MyInt>();  
 static List <MyInt> hi = new Vector<MyInt>();  
 static List <MyDbl> df = new Vector<MyDbl>();  
 static List <MyDbl> de = new Vector<MyDbl>();  
 static List <MyWord> wd = new Vector<MyWord>();  
 public static void main(String[] args) {  
 Scanner in = new Scanner(System.in);  
 int ln = 0; System.out.println("Input strings");  
 while(in.hasNextLine()) { s = in.nextLine();  
 if(s.equals("")) break; procLine(s,ln);  
 ln++;}di.sort(null); df.sort(null); de.sort(null);  
 wd.sort(null); in.close();  
 private static void printRes() {System.out.println("целые числа в десятичном виде:"); for (MyInt anInt : di) anInt.print();  
 System.out.println("““целые числа в шестнадцатиричном виде:");  
 for (MyInt myInt : hi) myInt.print();  
 System.out.println("““действительные числа с плавающей точкой:");  
 for (MyDbl dbl : df) dbl.print();  
 System.out.println("““действительные числа в экспоненциальной форме:");  
 for (MyDbl myDbl : de) myDbl.print();  
 System.out.println("““текст:");  
 for (MyWord myWord : wd) myWord.print();  
 System.out.println();}  
 static void procLine(String s, int n) {  
 StringTokenizer st = new StringTokenizer(s,"\r”“ ");while(st.hasMoreTokens()) {String w = st.nextToken();if( Word.isOctal(w)) {  
 wd.add(new MyWord(w,n));  
 continue; }  
 if (MyInt.parseDec(w, n)!=null) {di.add(new MyInt(w,n,Integer.parseInt(w))); continue; }  
 if (MyInt.parseHex(w, n)!=null) {  
 hi.add(new MyInt(w,n,Integer.parseInt(w.substring(2), 16)));  
 continue;   
 if (MyDbl.parse(w, n)!=null) {  
 if (w.contains("f") || w.contains("F")) {  
 wd.add(new MyWord(w,n));  
 continue; }  
 MyDbl md=new MyDbl(w,n, Double.parseDouble(w));  
 if (md.isExp())  
 de.add(md);  
 else  
 df.add(md);  
 continue;  
 }  
 wd.add(new MyWord(w,n));  
 }  
 }  
}

import java.util.\*;  
public class numbs\_8 {  
 public static void main(String[] args) {Scanner in =new Scanner(System.in);  
 ArrayList<Couple> I\_8=new ArrayList <Couple>();  
 int count=1;  
 while(in.hasNext()){  
 String b=in.nextLine();  
 if(b.equals("exit"))  
 break;  
 StringTokenizer str = new StringTokenizer (b," ");  
 while(str.hasMoreTokens()){  
 String c=str.nextToken();  
 if(c.startsWith("0")){  
 try{  
 Integer.parseInt(c,8);  
 I\_8.add( new Couple(count,Integer.parseInt(c,8)) );  
 }  
 catch(Exception e) {  
 continue;  
 }  
 }  
 }  
 count++;  
 }  
Collections.sort(I\_8, new Comparator <Couple> () {  
 public int compare(Couple o1, Couple o2) {  
 if(o1.int\_8 < o2.int\_8)  
 return 1;  
 return -1;  
 }  
 });  
 for(Couple i:I\_8)   
 System.out.println(i);  
 }  
}  
class Couple{  
 int int\_8;  
 int number\_str;  
 public int getInt\_8(){return int\_8;}  
 Couple(int a,int b){  
 int\_8=b;  
 number\_str=a;  
 }  
 public String toString(){return int\_8+ " "+number\_str;}  
}

/////////////////////////////////////////////////////////////////////////////////

import java.util.\*;  
public class proritet {  
 public static void main(String[] args) {  
 HashMap<String, Integer> myMap = new HashMap<String, Integer>();  
 myMap.put("+", 0); myMap.put("\*", 1);  
 myMap.put("&", 2); myMap.put("^", 3);  
 myMap.put("~", 4); myMap.put("%", 5);  
 myMap.put("«", 6); myMap.put("/", 7);  
 myMap.put("!", 8); myMap.put("»>", 9);  
 ArrayList<String> myArray = new ArrayList<String>();  
 String str = "23423 + 324 \* 3244 & 42343 \* 32423 ^ 6 ~ 12345 % 56 « 987 / 23 ! 1230 »> 88";  
 Set<String> myArrayOfRazdeliteli = myMap.keySet();  
 for (String i : myArrayOfRazdeliteli) {  
 for (int j = 0; j < new StringTokenizer(str, i).countTokens() - 1; ++j) {myArray.add(i); }}  
 for (int i = 0; i < myArray.size(); ++i) {  
 for (int j = 0; j < myArray.size(); ++j) {  
 if (myMap.get(myArray.get(i)) > myMap.get(myArray.get(j))) {  
 String buf = myArray.get(i);  
 myArray.set(i, myArray.get(j));  
 myArray.set(j, buf); }}}  
 String str2 = "";StringTokenizer st = new StringTokenizer(str, "+\*&^~%«/!»>");  
 int j = 0; while (st.hasMoreTokens()) {  
 String w = st.nextToken();  
 str2 += w; if (myArray.size() > j) {  
 str2 += myArray.get(j).toString();  
 ++j; } }  
 System.out.println(str2); }}  
*//import java.util.Scanner;  
//public class prioritet {  
// public static void main(String[] args) {  
// Scanner in=new Scanner(System.in);  
// String result="";  
// String s="";  
// int p=0,m=0,u=0,d=0;  
// while(in.hasNext()){  
// if((s=in.nextLine()).equalsIgnoreCase("EXIT"))  
// break;  
// char[] str = s.toCharArray();  
// for(int i=0;i<str.length;i++){  
// if(1== ChekZnak(str[i])) u++;  
// if(2== ChekZnak(str[i])) d++;  
// if(3== ChekZnak(str[i])) p++;  
// if(4== ChekZnak(str[i])) m++;}  
// for(int i=0;i<str.length;i++){  
// if('-'== (str[i])||'+'== (str[i])||'\*'== (str[i])||'/'== (str[i])){ if(u!=0){ str[i]='\*'; u--;continue; }  
// if(d!=0){ str[i]='/';d --;continue; }  
// if(p!=0){ str[i]='+'; p--;continue; }  
// if(m!=0){ str[i]='-'; m--; continue; }}}  
// System.out.println(new String(str)); // }  
// System.out.println(result); }  
// private static int ChekZnak(char i) {  
// if(i=='\*') return 1; if(i=='/') return 2;  
// if(i=='+') return 3; (i=='-') return 4; return 0; }}*

import java.util.\*;  
public class row {  
 static class Sequence implements Comparable<Sequence> {  
 int line; double M=0;  
 double D=0; double S=0; C=0;  
 ArrayList<Double> arr;  
 ArrayList<String> arrStr;  
 public Sequence(ArrayList<Double> a,ArrayList<String> s , int l) {  
 arr = new ArrayList<Double>();  
 arrStr=new ArrayList<String>();  
 arr.addAll(a);  
 for (int i=0; i<a.size(); i++)  
 arrStr.add(s.get(i));  
 line=l;  
 if (arr.size()!=0) {  
 for (Double aDouble : arr) M += aDouble; }  
 if (arr.size()>1) {  
 for (Double aDouble : arr) D += ((aDouble - M) \* (aDouble - M));  
 D/=(arr.size()-1);  
 }  
 S=Math.sqrt(D);  
 if (M!=0)  
 C=(S/M)\*100;  
 }  
 @Override  
 public int compareTo(Sequence o) {return Double.compare(C, o.C);  
 }  
 public String toString() {  
 String str="#"+line+ " M="+ M+" D="+D+" C="+C+" Sequence: ";  
 for (String s : arrStr) str += s + " ";  
 return str;  
 }  
 }  
 static ArrayList<Double> arrList=new ArrayList<Double>();  
 static ArrayList<String> strList=new ArrayList<String>();  
 static ArrayList<Sequence> seqList=new ArrayList<Sequence>();  
 public static void main(String[] args) {  
 Scanner in = new Scanner(System.in);  
 int ln = 0;  
 System.out.println("Input strings");  
 while(in.hasNextLine()) {  
 String s = in.nextLine();  
 if(s.equals(""))  
 break;  
 arrList.clear();  
 strList.clear();  
 if (ProcLine(s))  
 seqList.add(new Sequence(arrList,strList, ln));  
 ln++;}  
 Collections.sort(seqList);  
 for (Sequence s:seqList)  
 System.out.println(s);  
 in.close();  
 }  
 static boolean ProcLine(String str) {  
 StringTokenizer st = new StringTokenizer(str,"\r”“ ");  
 while(st.hasMoreTokens()) {  
 String token = st.nextToken();  
 int num = 0;  
 double dnum=0;  
 try {  
 if(token.charAt(0) == '0' && Character.toLowerCase(token.charAt(1))=='x')  
 num = Integer.parseInt(token.substring(2), 16);  
 else throw new Exception();  
 }  
 catch (Exception e) {  
 try {  
 if(token.charAt(0) != '0') throw new Exception();  
 num = Integer.parseInt(token.substring(1), 8);  
 }  
 catch (Exception e1) {  
 try {  
 num = Integer.parseInt(token);  
 }  
 catch (Exception e2) {  
 try {  
 dnum = Double.parseDouble(token);  
 }  
 catch (Exception e3) {  
 return false;  
 }  
 arrList.add(dnum);  
 strList.add(token);  
 continue;  
 }  
 }  
 }  
 arrList.add((double)num);  
 strList.add(token);  
 }  
 return true;  
 }  
}

*//Вариант 1.  
//Из стандартного входного потока прочитать текст  
//и в стандартный выходной поток вывести все цитаты.  
//Цитата — это текст между двойными кавычками*import java.util.\*;  
public class task\_1 {  
 static ArrayList<String> quotesList = new ArrayList<>();  
 static void processingLine(String s){  
 int begin = s.indexOf('"');  
 int end = s.indexOf('"', begin + 1);  
 while(end != -1){  
 quotesList.add(s.substring(begin + 1, end));  
 s = s.substring(0, begin) + s.substring(end + 1, s.length());  
 begin = s.indexOf('"');  
 end = s.indexOf('"', begin + 1);  
 }  
 }  
 public static void main(String[] args){  
 Scanner scan = new Scanner(System.in);  
 System.out.println("Inter data to start: ");  
 String l;  
 while (scan.hasNextLine() && !(l = scan.nextLine()).equals(""))  
 processingLine(l);  
 scan.close();  
 System.out.println("Quotes: ");  
 for(String s : quotesList)  
 System.out.println(s);  
 }  
}

*//Вариант 2  
//Программа читает текст из стандартного входного потока, находит слова,  
//начинающиеся с заглавной буквы и выводит в стандартный выходной поток  
//с новой строки*import java.util.\*;  
public class task\_2 {  
 static ArrayList<String> wordsList = new ArrayList<String>();  
 static void processingLine(String s){  
 StringTokenizer sTokenized = new StringTokenizer(s);  
 while(sTokenized.hasMoreElements()) {  
 String w = sTokenized.nextToken();  
 if (Character.isUpperCase(w.charAt(0)))  
 wordsList.add(w);  
 }  
 }  
 public static void main(String[] args){  
 Scanner scan = new Scanner(System.in);  
 System.out.println("Inter data: ");  
 String l;  
 while(scan.hasNextLine() && !(l = scan.nextLine()).equals(""))  
 processingLine(l);  
 scan.close();  
 System.out.println("Words with 0-s upper case character: ");  
 for(String str:wordsList)  
 System.out.println(str);  
 }  
}  
*/\*Вариант 3.  
 Создать программу, котоая читает текст из стандартного входного потока,  
 разбивает его на предложения и выводит их в стандартный поток по одному  
 предложению в строке, переводя все слова Восклицательных предложений  
 в верхний регистр.\*/*import java.util.\*;  
public class task\_3 {  
 static ArrayList<String> sentencesList = new ArrayList<String>();  
 static void processingLine(String s){  
 for(int i = 0; i < s.length(); i++){  
 if (s.charAt(i) == '!'){  
 String sentence = s.substring(0, i + 1);  
 sentencesList.add(sentence.toUpperCase());  
 s = s.substring(i + 1);  
 i = 0;  
 }  
 else  
 if (s.charAt(i) == '.' || s.charAt(i) == '?') {  
 String sentence = s.substring(0, i + 1);  
 sentencesList.add(sentence);  
 s = s.substring(i + 1);  
 i = 0;  
 }  
 }  
 }  
 public static void main(String[] args){  
 Scanner scan = new Scanner(System.in);  
 System.out.println("Inter data: ");  
 String l;  
 while(scan.hasNextLine() && !(l = scan.nextLine()).equals(""))  
 processingLine(l);  
 System.out.println("Divided sentences: ");  
 for(String str:sentencesList)  
 System.out.println(str);  
 }  
}

*/\*Вариант 4  
 Стандартный входной поток.  
 Найти все вопросительные предложения и вывести каждый с новой строки.\*/*import java.util.\*;  
public class task\_4 {  
 static ArrayList<String> questionsList = new ArrayList<String>();  
 static void processingLine(String s){  
 for (int i = 0; i < s.length(); i++){  
 if (s.charAt(i) == '?'){  
 String sentence = s.substring(0, i + 1);  
 questionsList.add(sentence);  
 s = s.substring(i + 1); i = 0; }  
 else if (s.charAt(i) == '.' || s.charAt(i) == '!'){  
 s = s.substring(i + 1); i = 0; }}}  
 public static void main(String[] args){  
 Scanner scan = new Scanner(System.in);  
 System.out.println("Inter data: ");  
 String l;  
 while(scan.hasNextLine() && !(l = scan.nextLine()).equals(""))  
 processingLine(l);  
 System.out.println("Questions: ");  
 for(String str:questionsList)  
 System.out.println(str); }}  
*//Вариант 5  
//найти длинные числа, отсортировать и вывести с номерами строк, в которых они встречались*import java.util.\*;  
public class task\_5 {  
 public static class Word {  
 private int pos;  
 private String word;  
 protected Word(String \_word,int \_pos){  
 pos = \_pos;  
 word = \_word; }  
 public String toString(){  
 return new String(word+"("+pos+")");}}  
 public static class WLongInt extends Word implements Comparable<WLongInt>,Comparator<Long>{  
 protected long value;  
 protected WLongInt(String \_word,int \_pos,long \_value){ super(\_word,\_pos); value = \_value; }  
 public int compareTo(WLongInt wli){  
 return Long.compare(value, wli.value); }  
 public int compare(Long arg0, Long arg1) {  
 return (int)(arg0 - arg1); }  
 public static WLongInt parseLong(String \_word,int \_pos){  
 try{ long \_value;  
 if(\_word.charAt(\_word.length()-1) == 'l' ||  
 \_word.charAt(\_word.length()-1) == 'L')  
 \_value = Long.parseLong(\_word.substring(0, \_word.length()-1)); else \_value = Long.parseLong(\_word);  
 return new WLongInt(\_word,\_pos,\_value);  
 }catch(Exception ignored){}  
 return null; }}  
 static void procLine(String line,int pos){  
 StringTokenizer st = new StringTokenizer(line," ““\t"); while(st.hasMoreTokens()){  
 String word = new String(st.nextToken());  
 WLongInt obj = WLongInt.parseLong(word,pos);  
 if(obj != null){  
 li.add(obj); }}}  
 static List<WLongInt> li = new Vector<WLongInt>();  
 public static void main(String[] args){  
 Scanner sc = new Scanner(System.in);  
 System.out.println("Insert your text line by line:");  
 int n=0;  
 while(sc.hasNextLine()){  
 procLine(sc.nextLine(),n++);  
 }  
 sc.close();  
 li.sort(null);  
 System.out.println("---All long nums---: ");  
 for(WLongInt i:li)  
 System.out.print(i);  
 System.out.println("---End---: ");  
 }  
}

*//Вариант 6*

*//Найти в тексте слова состоящие из цифр и //вывести*import java.util.\*;  
public class task\_6 {  
 static ArrayList<String> stringDigitsList = new ArrayList<String>();  
 public static boolean isDigit(String s){  
 for (int i = 0; i < s.length(); i++)  
 if (!Character.isDigit(s.charAt(i)))  
 return false;  
 return true;  
 }  
  
 public static void main(String[]args){  
 Scanner scan = new Scanner(System.in);  
 System.out.println("Enter data: ");  
 String l;  
 while(scan.hasNextLine() && !(l = scan.nextLine()).equals("")){  
 StringTokenizer st = new StringTokenizer(l);  
 while(st.hasMoreElements()){  
 String w = st.nextToken();  
 if (isDigit(w))  
 stringDigitsList.add(w);  
 }  
 }  
 System.out.println("Digits: ");  
 for(String str:stringDigitsList)  
 System.out.println(str);  
 }  
}

*/\* Вариант 7*

*Из входного потока прочитать слова.  
 \* Те которые только из цифр преобразовать в 16-ю запись и вывести 1 слово/строку  
 \*/*import java.util.\*;  
public class task\_7 {  
 static ArrayList<String> hexList = new ArrayList<String>();  
 public static boolean isDigit(String s){  
 for (int i = 0; i < s.length(); i++)  
 if (!Character.isDigit(s.charAt(i)))  
 return false;  
 return true;  
 }  
 public static void processingLine(String s){  
 StringTokenizer st = new StringTokenizer(s);  
 while(st.hasMoreElements()){  
 String w = st.nextToken();  
 String hex;  
 if(isDigit(w)) {  
 hex = Integer.toHexString(Integer.parseInt(w));  
 hexList.add(hex);}}}  
 public static void main(String[] args){  
 Scanner scan = new Scanner(System.in);  
 System.out.println("Inter data: ");  
 String l;  
 while(scan.hasNextLine() && !(l = scan.nextLine()).equals(""))  
 processingLine(l);  
 System.out.println("Hexes: ");  
 for(String h:hexList)  
 System.out.println(h); }}

*//Вариант 8.  
//Слова разделены пробелами.  
//Все слова в строке которые состоят только из букв отсортировать и сохранять количество пробелов*import java.util.\*;  
public class task\_8 {  
 static ArrayList<String> wordsList = new ArrayList<String>();  
 public static boolean isWord(String s) {  
 for (int i = 0; i < s.length(); i++)  
 if(!Character.isLetter(s.charAt(i)))  
 return false;  
 return true; }  
 public static void processingLine(String s){  
 StringTokenizer st = new StringTokenizer(s);  
 while(st.hasMoreElements()){  
 String w = st.nextToken();  
 if(isWord(w))  
 wordsList.add(w); }}  
 public static void main(String[]args){  
 Scanner scan = new Scanner(System.in);  
 System.out.println("Enter data: ");  
 String l;  
 while(scan.hasNextLine() && !(l = scan.nextLine()).equals(""))  
 processingLine(l);  
 System.out.println("Words: ");  
 for (String str:wordsList)  
 System.out.println(str); }}

*//Вариант 9.  
//Из стандартного входного потока считываются строки.  
//В каждой строке необходимо переставить слова в обратном порядке.  
//Вывести весь текст в стандартный выходной поток  
//в тексте в каждом предложении вывести слова в обратном порядке*import java.util.\*;  
class Word{  
 String value;  
 int begin;  
 int end;  
 public Word(String s, int b){  
 value = s;  
 begin = b;  
 end = b + s.length() - 1;  
 }  
}  
public class task\_9 {  
 static ArrayList<String> linesList = new ArrayList<String>();  
 public static String processLine(String s){  
 StringBuilder sBld = new StringBuilder(s);  
 ArrayList<Word> wordsList = new ArrayList<Word>();  
 ArrayList<String> stringList = new ArrayList<String>();  
 StringTokenizer sTknz = new StringTokenizer(s);  
 int position = 0;  
 while(sTknz.hasMoreElements()){  
 String w = sTknz.nextToken();  
 stringList.add(w);  
 position = s.indexOf(w);  
 wordsList.add(new Word(w, position));  
 }  
 for(int i = wordsList.size() - 1; i >= 0; i--)  
 sBld.replace(wordsList.get(i).begin, wordsList.get(i).end + 1, stringList.get(wordsList.size() - 1 - i));  
 return sBld.toString();  
 }  
  
 public static void main(String[]args){  
 Scanner scan = new Scanner(System.in);  
 System.out.println("Enter data: ");  
 String l;  
 while(scan.hasNextLine() && !(l = scan.nextLine()).equals(""))  
 linesList.add(processLine(l));  
 System.out.println("Reversed lines: ");  
 for(String str:linesList)  
 System.out.println(str); }}

*//Вариант 10.  
//Создать программу, которая читает текст из стандартного входного потока  
//и печатает в выходной поток предложения исходного текста, в которых встречается фраза,  
//указанная как обязательный параметр программы.*import java.util.\*;  
public class task\_10 {  
 static ArrayList<String> foundSentencesList = new ArrayList<String>();  
 static void findArgs(String txt, String args){  
 for (int i = 0; i < txt.length(); i++){  
 if (txt.charAt(i) == '.' || txt.charAt(i) == '!' ||txt.charAt(i) == '?'){  
 if((txt.substring(0, i + 1)).contains(args))  
 foundSentencesList.add(txt.substring(0, i + 1));  
 txt = txt.substring(i + 1);  
 i = 0; }}}  
 public static void main(String[]args){  
 Scanner scan = new Scanner(System.in);  
 System.out.println("Enter data: ");  
 String l;  
 String text = "";  
 while(scan.hasNextLine() && !(l = scan.nextLine()).equals("")){  
 text += l;  
 }  
 findArgs(text, args[0]);  
 System.out.println("Sentences with required args: ");  
 for (String str:foundSentencesList)  
 System.out.println(str);  
 }  
}

*/\* Вариант 11  
 со стандартного входного потока прочитать строки и преобразовать следующим образом:  
 слова состоят только из букв, все остальное считать разделителями;  
 найти все палиндромы, отсортировать в исходной строке;  
 остальные слова меняться не должны, разделители должны оставаться  
 на местах и количество пробелов оставаться прежним \*/*import java.util.\*;  
class Palindrome implements Comparable<Palindrome>{  
 protected String value;  
 protected int begin, end;  
 public static boolean isPalindrome(String s){  
 String reversed = "";  
 for(int i = s.length() - 1; i >= 0; i--)  
 reversed += s.charAt(i);  
 return s.equals(reversed);  
 }  
 public Palindrome(String v, int b){  
 value = v;  
 begin = b;  
 end = b + v.length() - 1;  
 }  
 public Palindrome(Palindrome p){  
 value = p.value;  
 begin = p.begin;  
 end = p.end; }  
 @Override  
 public int compareTo(Palindrome p) {  
 return value.compareTo(p.value); }}  
public class task\_11 {  
 static ArrayList<String> linesList = new ArrayList<String>();  
 static String getDelimiters(String s){  
 String delims = "";  
 for(int i = 0; i < s.length(); i++)  
 if (!Character.isLetter(s.charAt(i)))  
 delims += s.charAt(i);  
 return delims;  
 }  
 static void processingLine(String s){  
 StringBuilder sortedLine = new StringBuilder(s);  
 ArrayList<Palindrome> beforePalindromesList = new ArrayList<Palindrome>();  
 ArrayList<Palindrome> afterPalindromesList = new ArrayList<Palindrome>();  
 StringTokenizer tokenizedLine = new StringTokenizer(s, getDelimiters(s));  
 int pos = 0;  
 while(tokenizedLine.hasMoreElements()){  
 String word = tokenizedLine.nextToken();  
 pos = s.indexOf(word, pos);  
 if (Palindrome.isPalindrome(word)){  
 beforePalindromesList.add(new Palindrome(word, pos));  
 afterPalindromesList.add(new Palindrome(word, pos));  
 }  
 pos += word.length();  
 }  
 afterPalindromesList.sort(null);  
 for(int i = beforePalindromesList.size() - 1; i >= 0; i--)  
sortedLine.replace(beforePalindromesList.get(i).begin, beforePalindromesList.get(i).end + 1, afterPalindromesList.get(i).value); linesList.add(sortedLine.toString());  
 public static void main(String[] args){  
 Scanner scan = new Scanner(System.in);  
 System.out.println("Enter data: ");  
 String l;  
 while(scan.hasNextLine() && !(l = scan.nextLine()).equals(""))  
 processingLine(l);  
 scan.close();  
 System.out.println("Edited data: ");  
 for(String line:linesList)  
 System.out.println(line);  
 }  
}

*/\* Вариант 12  
 \* содержимое стандартного входного потока скопировать в выходной, преобразуя строки входного файла следующим образом:  
 \* каждая входная строка состоит из слов, разделенных одним или несколькими пробелами.  
 \* Переставить в ней слова, состоящие только из цифр так, чтобы они были упорядочены по возрастанию суммы их цифр.  
 \* При этом другие слова должны оставаться на месте, количество пробелов должно оставаться таким же, как и в исходной строке.  
 \*/*import java.util.\*;  
class DigitalWord implements Comparable<DigitalWord>{  
 String value;  
 int begin, end, sum;  
 public static boolean isDigitalWord(String s){  
 for (int i = 0; i < s.length(); i++)  
 if (!Character.isDigit(s.charAt(i)))  
 return false;  
 return true;  
 }  
 public DigitalWord(String s, int b){  
 value = s;  
 begin = b;  
 end = b + s.length() - 1;  
 sum = 0;  
 for(int i = 0; i < s.length(); i++)  
 sum += (s.charAt(i) - '0');  
 }  
 @Override  
 public int compareTo(DigitalWord dw) {  
 return Integer.compare(sum, dw.sum);  
 }  
}  
public class task\_12 {  
 static ArrayList<String> linesList = new ArrayList<String>();  
 static void processingLine (String line) {  
 StringBuilder sBldr = new StringBuilder(line);  
 ArrayList<DigitalWord> beforeDigitsList = new ArrayList<DigitalWord>();  
 ArrayList<DigitalWord> afterDigitsList = new ArrayList<DigitalWord>();  
 StringTokenizer sttok = new StringTokenizer(line, " \t");  
 int pos = 0;  
 while (sttok.hasMoreElements()) {  
 String word = sttok.nextToken();  
 pos = line.indexOf(word, pos);  
 if ( DigitalWord.isDigitalWord(word) ) {  
 DigitalWord dw = new DigitalWord(word, line.indexOf(word));  
 beforeDigitsList.add(dw);  
 afterDigitsList.add(dw);  
 }  
 pos += word.length();  
 }  
 Collections.sort(afterDigitsList);  
 for (int i = afterDigitsList.size() - 1; i >= 0; --i)  
 sBldr.replace(beforeDigitsList.get(i).begin,  
 beforeDigitsList.get(i).end + 1,  
 afterDigitsList.get(i).value);  
 linesList.add(sBldr.toString());  
 }  
 public static void main(String[] args){  
 Scanner scan = new Scanner(System.in);  
 System.out.println("Enter data: ");  
 String l;  
 while(scan.hasNextLine() && !(l = scan.nextLine()).equals(""))  
 processingLine(l);  
 scan.close();  
 System.out.println("Edited data: ");  
 for(String line:linesList)  
 System.out.println(line);  
 }  
}

*/\* Вариант 13  
 \* заменить в строке 8 пробелов стоящих подряд на символ табуляции \*/*import java.util.\*;  
public class task\_13 {  
 static ArrayList<String> linesList = new ArrayList<String>();  
 static void processingLine(String s){  
 StringBuilder sBldr = new StringBuilder(s);  
 String spaces = " ";  
 int position = sBldr.indexOf(spaces, 0);  
 while(position != -1){  
 sBldr.replace(position, position + 8, "\t");  
 position = sBldr.indexOf(spaces, position);  
 }  
 linesList.add(sBldr.toString());  
 }  
 public static void main(String[] args){  
 Scanner scan = new Scanner(System.in);  
 System.out.println("Enter data: ");  
 String l;  
 while(scan.hasNextLine() && !(l = scan.nextLine()).equals(""))  
 processingLine(l);  
 System.out.println("Edited data: ");  
 for(String str:linesList)  
 System.out.println(str);  
 }  
}

*//Вариант 14.  
//  
//Ввести через стандартный поток текст,  
//  
//найти слово максимальной длины и вывести слово и сколько раз встречается*import java.util.\*;  
public class task\_14 {  
 static ArrayList<Word> wordsList = new ArrayList<Word>();  
 static class Word implements Comparable<Word>{  
 protected String value;  
 protected int count;  
 public Word(String v, int c){  
 value = v;  
 count = c; }  
 @Override  
 public int compareTo(Word o) {  
 return Integer.compare(value.length(), o.value.length());}}  
 static void processingLine(String s){  
 StringTokenizer sTokenized = new StringTokenizer(s);  
 while(sTokenized.hasMoreElements()){  
 String word = sTokenized.nextToken();  
 if(wordsList.size() == 0) {  
 wordsList.add(new Word(word, 1));  
 continue; }  
 boolean flagUnique = true;  
 for (Word wl : wordsList)  
 flagUnique = !wl.value.equals(word);  
 if (flagUnique)  
 wordsList.add(new Word(word, 1));  
 else {  
 for (Word wl : wordsList)  
 if (wl.value.equals(word))  
 wl.count++;}}}  
 public static void main(String[] args){  
 Scanner scan = new Scanner(System.in);  
 System.out.println("Enter data: ");  
 String l;  
 while(scan.hasNextLine() && !(l = scan.nextLine()).equals(""))  
 processingLine(l);  
 Collections.sort(wordsList);  
 System.out.println("The longest word: " + wordsList.get(wordsList.size() - 1).value +  
 "““Frequency: " + wordsList.get(wordsList.size() - 1).count);}}

*//Вариант 15  
// найти длинные числа, отсортировать и вывести с номерами строк, в которых они встречались*import java.util.\*;  
  
public class task\_15 {  
 static ArrayList<LongDigital> ldList = new ArrayList<LongDigital>();  
  
 static class LongDigital implements Comparable<LongDigital>{  
 int digitalValue;  
 int line;  
  
 public LongDigital(int dv, int l){  
 digitalValue = dv;  
 line = l;  
 }  
  
 public static boolean isLong(String s){  
 if (s.charAt(s.length() - 1) == 'l' || s.charAt(s.length() - 1) == 'L'){  
 for(int i = 0; i < s.length() - 1; i++)  
 if (!Character.isDigit(s.charAt(i)))  
 return false;  
 return true;  
 };  
 return false;  
 }  
 @Override  
 public int compareTo(LongDigital o) {  
 return Integer.compare(digitalValue, o.digitalValue);  
 }  
 }  
 static void processLine(String s, int n){  
 StringTokenizer st = new StringTokenizer(s);  
 while (st.hasMoreElements()){  
 String word = st.nextToken();  
 if (LongDigital.isLong(word)) {  
 String cutWord = word.substring(0, word.length() - 1);  
 ldList.add(new LongDigital(Integer.parseInt(cutWord), n));  
 }  
 }  
 }  
 public static void main(String[] args){  
 Scanner scan = new Scanner(System.in);  
 System.out.println("Enter data: ");  
 String l;  
 int n = 0;  
 while(scan.hasNextLine() && !(l = scan.nextLine()).equals("")) {  
 processLine(l, n);  
 n++;  
 }  
 Collections.sort(ldList);  
 System.out.println("Long numbers: ");  
 for(LongDigital dw : ldList)  
 System.out.println("Number: " + dw.digitalValue +  
 ", Line: " + dw.line);  
 }  
}

*//Вариант 16 #1  
// Вводится строка. Задача: посчитать выражение, состоящее из целых, действительных чисел.  
//операции + - \* /  
//если выражение некорректно, то вывести "?"  
//если всё ок, то вывести результат  
//присутствие лишних пробелов допустимо в исходных данных...  
//import java.util.\*;  
//public class task\_16 {  
// static void Checking(char[] mas, String str) {  
// for (int i = 0; i < str.length(); i++) {  
// if ((mas[i] == '\*' || mas[i] == '+' || mas[i] == '-' || mas[i] == '/') && (i == 0 || (i == str.length() - 1))) {  
// System.out.println("Некорректные данные");  
// System.exit(0); } else {  
// if ((mas[i] != 0 && mas[i] == '0' && mas[i - 1] == '/')) {  
// System.out.println("Деление на ноль");  
// System.exit(0);  
// } else {  
// if (i != 0 && (mas[i] == '\*' || mas[i] == '+' || mas[i] == '-' || mas[i] == '/') && (mas[i - 1] == '\*'  
// || mas[i - 1] == '+' || mas[i - 1] == '-' || mas[i - 1] == '/')) {  
// System.out.println("Некорректные данные");  
// System.exit(0);  
// }  
// if (i != 0 && (mas[i] == '(' || mas[i] == ')') && (mas[i - 1] == ')'|| mas[i - 1] == '(')) {  
// System.out.println("Некорректные данные");System.exit(0);}}}}}  
// static boolean Proov(String str){  
// if(str.compareTo("+") == 0 || str.compareTo("-") == 0 || str.compareTo("\*") == 0 || str.compareTo("/") == 0 || str.compareTo("(") == 0 || str.compareTo(")") == 0) return false; else true;  
// }  
// static void Push (Stack st, String str){ st.push(str);   
// static double PopNumbers(Stack numbers){  
// String s = new String();  
// s = (String) numbers.pop();  
// double sum = Double.parseDouble(s);  
// return sum;  
// }  
// static String PopElements(Stack elements){  
// String x = (String)elements.pop();  
// return x;  
// }  
// static String Swt(double x1, double x2, String str){  
// double res = 0;  
// switch (str){ case("\*"):= x1 \* x2; break;  
// case ("+"):res= x1 + x2; break;  
// case ("-"):res = x2 - x1; break;  
// case ("/"): res = x2 / x1;if (x1 == 0){  
// System.out.println("Деление на ноль");  
// System.exit(0); } break; }  
// String s = String.valueOf(res);  
// return s;  
// }  
// static void Func(Stack numbers, Stack elements){  
// while(!elements.empty()){  
// double x1 = PopNumbers(numbers);  
// double x2 = PopNumbers(numbers);  
// String str = PopElements(elements);  
// String s = Swt(x1, x2, str);  
// Push(numbers, s);  
// }  
// }  
// static void Skb(Stack numbers, Stack elements, String tok){  
// String str = PopElements(elements);  
// do {  
// double x1 = PopNumbers(numbers);  
// double x2 = PopNumbers(numbers);  
// String s = Swt(x1, x2, str);  
// Push(numbers, s);  
// str = PopElements(elements);  
// }  
// while(str.compareTo("(") != 0);  
// }  
// public static void main(String[] args) {  
// Scanner in = new Scanner(System.in);  
// System.out.println("Введите выражение: ");  
// String str = in.nextLine();  
// StringTokenizer tokenizer = new StringTokenizer(str, "\*/()+-", true);  
// Stack numbers = new Stack();  
// Stack elements = new Stack();  
// boolean exit = true;  
// char[] Line = str.toCharArray();  
// Checking(Line, str);  
// while(tokenizer.hasMoreTokens()){  
// String tok = tokenizer.nextToken();  
// boolean flag = Proov(tok);  
// if(flag){  
// Push(numbers, tok); }else{  
// if(tok.compareTo("\*") == 0 || tok.compareTo("/") == 0 || tok.compareTo("(") == 0){  
// if(tok.compareTo("(") == 0){  
// exit = false; }  
// Push(elements, tok); }  
// else{if(elements.empty()){  
// Push(elements, tok); }  
// else{ ((tok.compareTo("+") == 0 || tok.compareTo("-") == 0) && exit == true){  
// Func(numbers, elements);  
// Push(elements, tok);  
// }  
// else{  
// if(tok.compareTo(")") == 0) {  
// Skb(numbers, elements, tok);  
// exit = true; }else{  
// Push(elements, tok);}}}}}}  
// Func(numbers, elements);  
// System.out.println("Результат: " + PopNumbers(numbers)); }}  
  
//вариант 16 #2  
// \* содержимое стандартного входного потока скопировать в выходной, преобразуя строки входного файла следующим образом:  
// \* каждая входная строка состоит из слов, разделенных одним или несколькими пробелами.  
// \* Переставить в ней слова, состоящие только из букв так, чтобы они были упорядочены по алфавиту.  
// \* При этом другие слова должны оставаться на месте, количество пробелов должно оставаться таким же, как и в исходной строке.*import java.util.\*;  
public class task\_16{  
 static ArrayList<String> linesList = new ArrayList<String>();  
 static class Word implements Comparable<Word>{  
 protected String value;  
 protected int begin, end;  
 public Word(String s, int b){  
 value = s;  
 begin = b;  
 end = b + s.length() - 1; }  
 public static boolean isWord(String s){  
 for(int i = 0; i < s.length(); i++)  
 if(!Character.isLetter(s.charAt(i)))  
 return false;  
 return true; }  
 @Override  
 public int compareTo(Word o) {  
 return value.compareTo(o.value);}}  
 static void processLine(String s){  
 StringBuilder sb = new StringBuilder(s);  
 ArrayList<Word> wordsBefore = new ArrayList<Word>();  
 ArrayList<Word> wordsAfter = new ArrayList<Word>();  
 StringTokenizer st = new StringTokenizer(s);  
 int position = 0;  
 while(st.hasMoreElements()){  
 String word = st.nextToken();  
 position = s.indexOf(word, position);  
 if (Word.isWord(word)) {  
 wordsAfter.add(new Word(word, position));  
 wordsBefore.add(new Word(word, position));}}  
 Collections.sort(wordsAfter);  
 for(int i = wordsAfter.size() - 1; i >= 0; i--)  
 sb.replace(wordsBefore.get(i).begin, wordsBefore.get(i).end + 1, wordsAfter.get(i).value);  
 linesList.add(sb.toString());}  
 public static void main(String[] args){  
 Scanner scan = new Scanner(System.in);  
 System.out.println("Enter data: ");String l;  
 while(scan.hasNextLine() && !(l = scan.nextLine()).equals("")) processLine(l);  
 System.out.println("Edited data: ");  
 for(String line:linesList)  
 System.out.println(line);}}

*//Вариант 17.  
//считать с консоли размерность n матрицы.  
//Матрицу заполнить рандомными числами из диапазона от -n до n.  
//Транспонировать матрицу и вывести результат*import java.util.\*;  
public class task\_17 {  
 static void processMatrix(int n){  
 int[][] A = new int[n][n];  
 System.out.println("Generated matrix: ");  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < n; j++) {  
 A[i][j] = (int) (Math.random() \* (2 \* n + 1)) - n; System.out.printf("%5d", A[i][j]);}   
 System.out.println();}  
 System.out.println("Transported generated matrix: ");int[][] B = new int[n][n];  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < n; j++) {B[i][j] = A[j][i];  
 System.out.printf("%5d", B[i][j]);} }  
 System.out.println();}}  
 public static void main(String[]args){  
 Scanner scan = new Scanner(System.in);  
 System.out.println("Enter matrix size: ");  
 int n = scan.nextInt();  
 processMatrix(n);}}

*//Вариант 18  
// Из стандартного входного потока прочитать текст.  
//Разделители - пробелы(один или несколько) и знак табуляции.  
//В выходной поток:  
//1) преобразовать строки таким образом, чтобы их длина не была больше 80, при этом нельзя "херачить слова"  
//2) заменить \t на пробелы(без учёта позиционного табулирования)  
//3) заменить несколько пробелов на один  
//4) удалить пробелы:  
//\* в начале строки  
//\* в конце строки  
//\* перед знаками препинания*import java.util.\*;  
public class task\_18 {  
 static ArrayList<String> linesList = new ArrayList<String>();  
 static String editLine(String s){  
 StringTokenizer st = new StringTokenizer(s, " \t");  
 StringBuilder ns = new StringBuilder();  
 ArrayList<String> wordsList = new ArrayList<String>();  
 while(st.hasMoreElements()) {  
 String word = st.nextToken();  
 wordsList.add(word);  
 }  
 for(int i = 0; i < wordsList.size() - 1; i++){  
 if (wordsList.get(i + 1).equals("!") || wordsList.get(i + 1).equals("?") || wordsList.get(i + 1).equals(".") ||  
 wordsList.get(i + 1).equals(",") || wordsList.get(i + 1).equals(":") || wordsList.get(i + 1).equals(";"))  
 ns.append(wordsList.get(i));  
 else  
 ns.append(wordsList.get(i)).append(' ');  
 }  
 ns.append(wordsList.get(wordsList.size() - 1));  
 s = ns.toString();  
 return s;  
 }  
 static void editText(String s){  
 StringTokenizer st = new StringTokenizer(s);  
 int position = 0;  
 while(st.hasMoreElements()){  
 String word = st.nextToken();  
 position = s.indexOf(word, position);  
 if (word.length() + position >= 80) {  
 linesList.add(s.substring(0, position - 1));  
 s = s.substring(position);  
 position = 0;  
 }  
 }  
 if (s.length() > 0)  
 linesList.add(s);  
 }  
 public static void main(String[] args){  
 Scanner scan = new Scanner(System.in);  
 System.out.println("Enter text: ");  
 String l;  
 StringBuilder text = new StringBuilder();  
 while(scan.hasNextLine() && !(l = scan.nextLine()).equals(""))  
 text.append(editLine(l)).append(' ');  
 l = text.toString();  
 l = l.substring(0, l.length() - 1);  
 editText(l);  
 System.out.println("Edited text: ");  
 for(String line:linesList)  
 System.out.println(line);  
 }  
}

*//Вариант 19.  
//у меня было сделать 2 точки,сделать прямую и определить по какую сторону лежат точки*import javax.swing.\*;  
import java.awt.\*;  
import java.util.\*;  
class Point\_ extends Canvas {  
 private static final long serialVersionUID = 1L;  
 Color clr;  
 int x, y;  
 Dimension dim = new Dimension(20, 20);  
 public Point\_ (int appWidth, int appHeight) {  
 super();  
 setMaximumSize(dim);  
 setBackground(null);  
 setBounds(0, 0, dim.width, dim.height);  
 clr = Color.black;  
 x = new Random().nextInt(appWidth - dim.width) + dim.width / 2;  
 y = new Random().nextInt(appHeight - dim.height) + dim.height / 2;  
 }  
 public void paint (Graphics g) {  
 g.setColor(clr);  
 g.fillOval(0, 0, dim.width, dim.height);  
 }  
 public void setColor (Color clr) {  
 this.clr = clr;  
 }  
 public Dimension getMinimumSize () { return dim; }  
 public Dimension getPreferredSize () { return dim; }  
}  
class Line\_ {  
 Color clr;  
 int x1, y1, x2, y2;  
 public Line\_ (Color c, int x1, int y1, int x2, int y2) {  
 clr = c;  
 this.x1 = x1;  
 this.y1 = y1;  
 this.x2 = x2;  
 this.y2 = y2;  
 }  
 public int compare (Point\_ p) {  
 double tg = (double)(y2 - y1) / (x2 - x1);  
 int xt = (int)(x1 - (double)(y1 - p.y) / tg);  
 if ( p.x < xt )  
 return -1;  
 else if ( p.x > xt )  
 return 1;  
  
 int yt = (int)(y1 - tg \* (x1 - p.x));  
 if ( p.y < yt )  
 return 1;  
 else if ( p.y > yt )  
 return -1;  
 return 0;  
 }  
}  
  
public class task\_19 extends JFrame {  
  
 private static final long serialVersionUID = 1L;  
 static final int CX = 600, CY = 500;  
 Canvas point\_first;  
 Canvas point\_second;  
 Line\_ Line\_;  
 String s;  
  
 public Color getHtmlColor (String rgb, Color def) {  
 if ( rgb == null || rgb.charAt(0) != '#' )  
 return def;  
 try {  
 return new Color(Integer.parseInt(rgb.substring(1), 16));  
 } catch (NumberFormatException e) {  
 return def;  
 }  
 }  
  
 public task\_19 () {  
 setSize(CX, CY);  
 setLayout(null);  
 setBackground(Color.white);  
 Color clr = Color.BLACK;  
 int x1 = 0, y1 = 0, x2 = 0, y2 = 0;  
 try {  
 clr = getHtmlColor(null, Color.BLACK);  
 x1 = new Random().nextInt(550);  
 y1 = new Random().nextInt(550);  
 x2 = new Random().nextInt(550);  
 y2 = new Random().nextInt(550);  
 } catch (Exception ignored) {}  
 Line\_ = new Line\_(clr, x1, y1, x2, y2);  
 point\_first = new Point\_(this.getWidth(), this.getHeight());  
 point\_second = new Point\_(this.getWidth(), this.getHeight());  
  
 int ans\_first = Line\_.compare((Point\_)point\_first);  
 int ans\_second = Line\_.compare((Point\_)point\_second);  
  
 Color cl\_first = Color.red;  
 Color cl\_second = Color.red;  
  
 s = "";  
  
 switch (ans\_first) {  
 case -1: cl\_first = Color.red; break;  
 case 1: cl\_first = Color.green; break;  
 default: cl\_first = Color.yellow;  
 }  
 switch (ans\_second) {  
 case -1: cl\_second = Color.red; break;  
 case 1: cl\_second = Color.green; break;  
 default: cl\_second = Color.yellow;  
 }  
 if(ans\_second == ans\_first){  
 if(ans\_second == -1)  
 s="точки лежат слева от прямой";  
 if(ans\_second == 1)  
 s="точки лежат справа от прямой";  
 if(ans\_second == 0)  
 s="точки лежат на прямой";  
 }else{  
 if(ans\_second + ans\_first == 0)  
 s="точки лежат по разные стороны от прямой";  
 else{  
 s="одна точка лежит на прямой, а вторая";  
 if(ans\_second==-1 || ans\_first==-1)  
 s+=" справа";  
 else  
 s+=" слева";  
 s+=" от прямой";  
 }  
 }  
  
 ((Point\_)point\_first).setColor(cl\_first);  
 point\_first.setLocation(((Point\_)point\_first).x, ((Point\_)point\_first).y);  
 add(point\_first);  
  
 ((Point\_)point\_second).setColor(cl\_second);  
 point\_second.setLocation(((Point\_)point\_second).x, ((Point\_)point\_second).y);  
 add(point\_second);  
  
 repaint();  
 }  
 public void paint (Graphics g) {  
 g.clearRect(0, 0, getWidth(), getHeight());  
 g.drawString(s, 0, 280);  
 g.drawLine(Line\_.x1, Line\_.y1, Line\_.x2, Line\_.y2);  
 }  
  
 public static void main(String[] args) {  
 task\_19 t = new task\_19();  
 t.setVisible(true);  
 }  
}

import javax.swing.\*;  
import java.awt.\*;  
import java.util.Random;  
  
*/\* Вариант 20  
 \* В следующих заданиях выполнить соответствующий рисунок в окне апплета:  
 \* 1. Создать классы Point и Line. Объявить массив из n объектов класса Point.  
 \* Для объекта класса Line определить какие из объектов Point лежат на одной стороне  
 \* от прямой линии и какие - на другой. Реализовать ввод данных для объекта Line и  
 \* случайное задание данных для объектов Point.  
 \*/*class Point extends Canvas {  
 private static final long serialVersionUID = 1L;  
 Color clr;  
 int x, y;  
 Dimension dim = new Dimension(20, 20);  
 public Point (int appWidth, int appHeight) {  
 super();  
 setMaximumSize(dim);  
 setBackground(null);  
 setBounds(0, 0, dim.width, dim.height);  
 clr = Color.black;  
 x = new Random().nextInt(appWidth - dim.width) + dim.width / 2;  
 y = new Random().nextInt(appHeight - dim.height) + dim.height / 2;  
 }  
 public void paint (Graphics g) {  
 g.setColor(clr);  
 g.fillOval(0, 0, dim.width, dim.height);  
 }  
 public void setColor (Color clr) {  
 this.clr = clr;  
 }  
 public Dimension getMinimumSize () { return dim; }  
 public Dimension getPreferredSize () { return dim; }  
}  
  
class Line {  
 Color clr;  
 int x1, y1, x2, y2;  
 public Line (Color c, int x1, int y1, int x2, int y2) {  
 clr = c;  
 this.x1 = x1;  
 this.y1 = y1;  
 this.x2 = x2;  
 this.y2 = y2;  
 }  
  
 public int compare (Point p) {  
 double tg = ((double)y2 - y1) / (x2 - x1);  
 double xt = (x1 - (y1 - p.y) / tg);  
 if (p.x < xt)  
 return -1;  
 else if (p.x > xt)  
 return 1;  
  
 double yt = (y1 - tg \* (x1 - p.x));  
 if (p.y < yt)  
 return 1;  
 else if (p.y > yt)  
 return -1;  
 return 0;  
 }  
}  
  
public class task\_20 extends JFrame {  
  
 private static final long serialVersionUID = 1L;  
 static final int CX = 600, CY = 500;  
 Canvas[] points;  
 Line line;  
  
 public Color getHtmlColor (String rgb, Color def) {  
 if ( rgb == null || rgb.charAt(0) != '#' )  
 return def;  
 try {  
 return new Color(Integer.parseInt(rgb.substring(1), 16));  
 } catch (NumberFormatException e) {  
 return def;  
 }  
 }  
  
 public task\_20 () {  
 setSize(CX, CY);  
 setLayout(null);  
 setBackground(Color.white);  
 Color clr = Color.BLACK;  
 int x1 = 0, y1 = 0, x2 = 0, y2 = 0, n = 0;  
 try {  
 clr = getHtmlColor(null, Color.BLACK);  
 x1 = new Random().nextInt(550);  
 y1 = new Random().nextInt(550);;  
 x2 = new Random().nextInt(550);;  
 y2 = new Random().nextInt(550);;  
 n = 10;  
 } catch (Exception ignored) {}  
 line = new Line(clr, x1, y1, x2, y2);  
 points = new Point[n];  
 for (Canvas p: points) {  
 p = new Point(this.getWidth(), this.getHeight());  
 int ans = (line).compare((Point)p);  
 Color cl;  
 switch (ans) {  
 case -1: cl = Color.red; break;  
 case 1: cl = Color.green; break;  
 default: cl = Color.yellow;  
 }  
 ((Point)p).setColor(cl);  
 p.setLocation(((Point)p).x, ((Point)p).y);  
 add(p);  
 }  
 repaint();  
 }  
 public void paint (Graphics g) {  
 g.setColor(line.clr);  
 g.drawLine(line.x1, line.y1, line.x2, line.y2);  
 }  
  
 public static void main(String[] args) {  
 task\_20 t = new task\_20();  
 t.setVisible(true);  
 }  
  
}

import javax.swing.\*;  
import javax.swing.Timer;  
import java.awt.\*;  
  
*/\* Вариант 24  
 В окне апплета по горизонтали двигается окружность, отражаясь от границ апплета.  
Да не просто отражается, а происходит упругое столкновение. \*/*public class task\_20\_1 extends JFrame {  
  
 static final int CX = 600, CY = 500;  
 int x,y;  
 final int speed = 5;  
 Timer t;  
 Color c = Color.GREEN;  
 boolean toRight = true;  
 int dx = 0;  
  
 public task\_20\_1 () {  
 setDefaultCloseOperation(EXIT\_ON\_CLOSE);  
 setSize(CX, CY);  
 x=0;  
 y = CY/2;  
 t = new Timer(50, e -> doMove());  
 t.start();  
 }  
  
 @Override  
 public void paint(Graphics g) {  
 g.clearRect(0,0,getWidth(),getHeight());  
 if (dx != 0)  
 g.setColor(Color.RED);  
 else  
 g.setColor(Color.BLACK);  
 g.fillOval(x,y,100 - 4 \* dx,100 - dx \* 4);  
 dx = 0;  
 }  
 void doMove () {  
 if (x + 50 >= getWidth() - 50) {  
 toRight = false;  
 dx = 10;  
 }  
 if (x - 50 <= 0) {  
 toRight = true;  
 dx = 10;  
 }  
 if (toRight)  
 x += 10;  
 else  
 x -= 10;  
 repaint();  
 }  
  
 public static void main(String[] args) {  
 new task\_20\_1().setVisible(true);  
 }  
}

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.Graphics;  
import java.util.function.Function;  
  
*/\* Вариант 21  
 \* Изобразить в окне апплета гармонические колебания точки вдоль некоторого  
 \* горизонтального отрезка. Если длина отрезка равна q, то расстояние от точки до левого  
 \* конца в момент времени t можно считать равным q(1+cos(wt))/2,где w - некоторая константа.  
 \* Указанные величины вводятся через параметры апплета.  
 \*/*public class task\_21 extends JFrame {  
  
 final int CX = 600, CY = 500;  
 Timer timer;  
 final int q = 350, w = 7;  
 int x, t;  
 Function<Integer,Integer> func = (t) -> (int)(0.5 \* q \* (1 + Math.cos(1.0 \* w \* t)));  
  
 public task\_21 () {  
 setDefaultCloseOperation(EXIT\_ON\_CLOSE);  
 t=0;  
 x=func.apply(t);  
 setSize(CX, CY);  
 timer = new Timer(100, e -> doMove());  
 timer.start();  
 }  
  
 @Override  
 public void paint(Graphics g) {  
 g.clearRect(0,0,getWidth(),getHeight());  
 g.setColor(Color.GREEN);  
 g.drawLine(0,getHeight()/2,getWidth(),getHeight()/2);  
 g.setColor(Color.BLACK);  
 g.fillOval(x,t,10,10);  
 }  
  
 void doMove () {  
 if (t>getHeight())  
 t=0;  
 t += 10;  
 x = func.apply(t);  
 repaint();  
 }  
  
 public static void main(String[] args) {  
 new task\_21().setVisible(true);  
 }  
}

*//Вариант 22.  
// Задача с параболой - одна из лабораторных по апплетам Y = X^2-1  
// X[-1,2] dx=0.05*import javax.swing.\*;  
import java.awt.\*;  
  
class Points {  
 int centerX;  
 int centerY;  
 double x;  
 double y;  
 int conclX;  
 int conclY;  
  
 public Points(int centerX, int centerY, double x, double y){  
 this.centerX = centerX;  
 this.centerY = centerY;  
 this.x = x;  
 this.y = y;  
 conclX = this.centerX + (int)(this.x\*75);  
 conclY = this.centerY - (int)(this.y\*75);  
 }  
  
 public Points (Points p){  
 conclX = p.conclX;  
 conclY = p.conclY;  
 }  
  
 public int p\_x(){  
 return conclX;  
 }  
  
 public int p\_y(){  
 return conclY;  
 }  
}  
  
class Draw\_Graph extends Canvas{  
 final static int centerX = (task\_22.Size\_win\_X / 2),  
 centerY = (task\_22.Size\_win\_Y / 2);  
 Color graph;  
 Color deck;  
 Dimension dim = new Dimension(task\_22.Size\_win\_X, task\_22.Size\_win\_Y);  
  
 public Draw\_Graph(Color graph, Color deck){  
 this.graph = graph;  
 this.deck = deck;  
 setBounds(0, 0, dim.width, dim.height);  
 }  
  
 public void paint(Graphics g){  
 g.setColor(graph);  
 graph\_x\_x(g);  
 g.setColor(deck);  
 deck\_graph(g);  
 }  
  
 public double func(double x){  
 return (Math.pow(x, 2) - 1);  
 }  
  
 void deck\_graph(Graphics g){  
 g.drawLine(0, centerY, task\_22.Size\_win\_X, centerY);  
 g.drawLine(centerX, 0, centerX, task\_22.Size\_win\_Y);  
  
 g.drawLine(175, 255, 175, 245);  
 g.drawLine(400, 255, 400, 245);  
 g.drawLine(325, 255, 325, 245);  
  
 g.drawLine(255, 175, 245, 175);  
 g.drawLine(255, 325, 245, 325);  
 g.drawLine(255, 100, 245, 100);  
 g.drawLine(255, 25, 245, 25);  
  
 g.drawString("0", 253, 262);  
 g.drawString("-1", 176, 262);  
 g.drawString("1", 326, 262);  
 g.drawString("2", 401, 262);  
  
 g.drawString("3", 257, 25);  
 g.drawString("2", 257, 100);  
 g.drawString("1", 257, 175);  
 g.drawString("-1", 257, 334);  
  
 }  
  
 void graph\_x\_x(Graphics g){  
 double x = -1;  
 double dx = 0.05;  
 Points p1, p2;  
 p1 = new Points(centerX, centerY, x, func(x));  
 while(x<=2)  
 {  
 p2 = new Points(p1);  
 x += dx;  
 p1 = new Points(centerX, centerY, x, func(x));  
 g.drawLine(p2.conclX, p2.conclY, p1.conclX, p1.conclY);  
 }  
 }  
}  
  
public class task\_22 extends JFrame {  
 static final int Size\_win\_X = 500, Size\_win\_Y = 500;  
 Canvas obj;  
  
 public task\_22(){  
 setSize(Size\_win\_X, Size\_win\_Y);  
 Color deck = new Color(0, 0, 0);  
 Color graph = new Color(0, 250, 0);  
 obj = new Draw\_Graph(graph, deck);  
 add(obj);  
 }  
  
 public static void main(String[] args) {  
 task\_22 t = new task\_22();  
 t.setVisible(true);  
 t.repaint();  
 }  
}

import javax.swing.\*;  
import java.applet.Applet;  
import java.awt.\*;  
  
  
public class task\_23 extends JFrame {  
  
 public static void main(String[] args) {  
 new task\_23().setVisible(true);  
 }  
  
 private static final long serialVersionUID = 1L;  
  
 static final int CX = 400, CY = 400;  
 static String str = "Odin";  
 static int pose1 = 10;  
 static int pose2 = 10;  
 static int pose3 = 10;  
 static int pose4 = 10;  
 static int next =0;  
 static String our="";  
 static int point=0;  
 static char cur1=str.charAt(point);  
 static char cur2=str.charAt(++point);  
 static char cur3=str.charAt(++point);  
 static char cur4=str.charAt(++point);  
 Timer t;  
 boolean stFlag = false;  
 public task\_23()  
 {  
 setDefaultCloseOperation(EXIT\_ON\_CLOSE);  
 setSize( CX, CY );  
 setLayout( null );  
 Color col = new Color(9, 154, 125);  
 setBackground( col );  
 Color colx = new Color(226, 226, 226);  
 col = Color.WHITE;  
 t = new Timer(100,e -> doMove());  
 t.start();  
  
 }  
 public void paint(Graphics g)  
 {  
 g.clearRect(0,0,getWidth(),getHeight());  
 g.drawString(cur1+" ",pose1,pose1);  
 g.drawString(cur2+" ",400-pose2,pose2);  
 g.drawString(cur3+" ",400-pose3,400-pose3);  
 g.drawString(cur4+" ",pose4,400-pose4);  
 g.drawString(our,100,200);  
  
 }  
  
 public void doMove()  
 {  
 if(next==0)  
 {  
 pose1+=10;  
 repaint();  
 if(pose1==200)  
 {  
 pose1=10;  
 next+=1;  
 our+=cur1;  
 point++;  
 if(point>=str.length())  
 {  
 cur1=' ';  
 }  
 else  
 {  
 cur1=str.charAt(point);  
 }  
 if ( cur1 == ' ' && cur2 == ' ' &&cur3 == ' ' && cur4 == ' ')  
 {  
 stFlag = true;  
 }  
  
 }  
 }  
 else if(next==1)  
 {  
 pose2+=10;  
 repaint();  
 if(pose2==200)  
 {  
 pose2=10;  
 next+=1;  
 our+=cur2;  
 point++;  
 if(point>=str.length())  
 {  
 cur2=' ';  
 }  
 else  
 {  
 cur2=str.charAt(point);  
 }  
 if ( cur1 == ' ' && cur2 == ' ' &&cur3 == ' ' && cur4 == ' ')  
 {  
 stFlag = true;  
 }  
 }  
 }  
 else if(next==2)  
 {  
 pose3+=10;  
 repaint();  
 if(pose3==200)  
 {  
 pose3=10;  
 next+=1;  
 our+=cur3;  
 point++;  
 if(point>=str.length())  
 {  
 cur3=' ';  
 }  
 else  
 {  
 cur3=str.charAt(point);  
 }  
 if ( cur1 == ' ' && cur2 == ' ' &&cur3 == ' ' && cur4 == ' ')  
 {  
 stFlag = true;  
 }  
 }  
 }  
 else  
 {  
 pose4+=10;  
 repaint();  
 if(pose4==200)  
 {  
 pose4=10;  
 next=0;  
 our+=cur4;  
 point++;  
 if(point>=str.length())  
 {  
 cur4=' ';  
 }  
 else  
 {  
 cur4=str.charAt(point);  
 }  
 if ( cur1 == ' ' && cur2 == ' ' &&cur3 == ' ' && cur4 == ' ')  
 {  
 stFlag = true;  
 }  
 }  
 }  
  
 }  
  
}

import javax.swing.\*;  
import javax.swing.Timer;  
import java.awt.\*;  
  
*/\* Вариант 24  
 Создать апплет со строкой, которая движется горизонтально,  
 \* отражаясь от границ апплета и меняя при этом свой цвет на цвет.  
 \* Строка вводится через параметры апплета.  
 \*/*public class task\_24 extends JFrame {  
  
 static final int CX = 600, CY = 500;  
 int x,y;  
 final int speed = 5;  
 Timer t;  
 Color c = Color.GREEN;  
 String str;  
 Font font = new Font("Times New Roman", Font.BOLD, 30);  
 boolean toRight = true;  
  
 public task\_24 () {  
 setDefaultCloseOperation(EXIT\_ON\_CLOSE);  
 setSize(CX, CY);  
 x=0;  
 y = CY/2;  
 str = "god help me";  
 t = new Timer(50, e -> doMove());  
 t.start();  
 }  
  
 @Override  
 public void paint(Graphics g) {  
 g.clearRect(0,0,getWidth(),getHeight());  
 g.setFont(font);  
 g.setColor(c);  
 g.drawString(str, x,y);  
 }  
  
 public void doMove () {  
 if ( toRight ) {  
 x+=speed;  
 if ( x >= (getWidth() - getFontMetrics(font).stringWidth(str))) {  
 c = Color.BLUE;  
 toRight = false;  
 }  
 }  
 else {  
 x-=speed;  
 if ( x <= 0 ) {  
 c = Color.pink;  
 toRight = true;  
 }  
 }  
 y = getHeight() / 2;  
 repaint();  
 }  
  
 public static void main(String[] args) {  
 new task\_24().setVisible(true);  
 }  
}

*//Вариант 25.  
//  
//прочитать все данные с входного потока, разделить на слова,  
//выделить лексемы Java,представляющие собой дробные числа(float) со знаком.  
//Хранить эти значения с номерами входных строк. Вывести, предварительно отсортировав  
//в порядке убывания значений.  
//Слова разделены пробелами. Найти знаковые литералы float и  
//вывести в убывающей порядке с номером строки, в которой встречаются*import java.util.\*;  
import java.io.\*;  
  
public class task\_25 {  
  
 static ArrayList<floatNumber> fnList = new ArrayList<floatNumber>();  
  
 static class floatNumber implements Comparable <floatNumber>{  
 float value;  
 int line;  
  
 public floatNumber(String str, int n){  
 value = Float.parseFloat(str.substring(0, str.length() - 1));  
 line = n;  
 }  
  
 public static boolean isFloat(String s){  
 if (s.charAt(s.length() - 1) == 'f' || s.charAt(s.length() - 1) == 'F') {  
 try  
 {  
 float num = Float.parseFloat(s.substring(0, s.length() - 1));  
 return true;  
 }  
 catch(Exception e){ return false;}  
 }  
 return false;  
 }  
  
 @Override  
 public int compareTo(floatNumber o) {  
 return Float.compare(o.value, value);  
 }  
  
 public String toString() { return "Number: " + value + ", line: " + line;}  
 }  
  
 static void processLine(String s, int n){  
 StringTokenizer st = new StringTokenizer(s, " ");  
 while(st.hasMoreElements()){  
 String word = st.nextToken();  
 if(floatNumber.isFloat(word))  
 fnList.add(new floatNumber(word, n));  
 }  
 }  
  
 public static void main(String[]args){  
 Scanner scan = new Scanner(System.in);  
 System.out.println("Enter data: ");  
 String l;  
 int n = 0;  
 while(scan.hasNextLine() && !(l = scan.nextLine()).equals("")) {  
 processLine(l, n);  
 n++;  
 }  
 Collections.sort(fnList);  
 System.out.println("Float number after reversed sort: ");  
 for(floatNumber fnum:fnList)  
 System.out.println(fnum);  
 }  
}

import javax.swing.\*;  
import java.awt.\*;  
  
public class RotateTriangle extends JFrame{  
 private static final long serialVersionUID = 1L;  
  
 static final int CX = 600, CY = 600;  
 static int counts = 3;  
 static double xCentre = 300, yCentre = 300;  
 static int[] masx = new int [3];  
 static int[] masy = new int [3];  
 int angle=0;  
 Timer t;  
 int radius = 100;  
  
 int step = 5;  
 public RotateTriangle(){  
 setSize(CX, CY);  
 createTriangle(masx, masy, counts);  
 t = new Timer(50, e -> doMove());  
 t.start();  
 }  
  
 public void createTriangle(int [] xArr, int [] yArr, int counts){  
 double part = 360 \* 1.0 / (counts);  
 for (int i = 0 ; i < counts; i++){  
 xArr[i] = (int)xCentre + (int)(Math.cos(Math.toRadians(i \* part + angle)) \* radius);  
 yArr[i] = (int)yCentre + (int)(Math.sin(Math.toRadians(i \* part + angle)) \* radius);  
 }  
  
 }  
  
 public void doMove()  
 {  
 angle += step;  
 if (angle % 360 == 0)  
 angle = 0;  
  
 createTriangle(masx,masy,counts);  
 repaint();  
 }  
  
 public void paint(Graphics g) {  
 g.clearRect(0,0,getWidth(),getHeight());  
 g.drawPolygon(new Polygon(masx,masy,3));  
 }  
  
 public static void main(String[] args) {  
 RotateTriangle rt = new RotateTriangle();  
 rt.setVisible(true);  
 }  
}

import javax.swing.\*;  
import java.awt.\*;  
  
class Star extends JFrame{  
 static final int CX = 600, CY = 600;  
 int counts = 5;  
 Color bgColor = Color.blue;  
 Color startColor = Color.yellow;  
 int radius = 100;  
 int smallradius = radius / 2;  
 int xPos = 300;  
 int yPos = 300;  
 int [] xArr = new int[counts \* 2];  
 int [] yArr = new int[counts \* 2];  
 double startAngle = 0;  
 Timer t;  
  
 int step = 5;  
 public Star(){  
 setSize(CX, CY);  
 createStar(xArr, yArr, counts);  
 t = new Timer(50, e -> doMove());  
 t.start();  
 }  
  
 public void createStar(int [] xArr, int [] yArr, int counts){  
  
 double part = 360 \* 1.0 / (counts \* 2);  
  
 for (int i = 0 ; i < counts \* 2; i++){  
 if (i % 2 == 0){  
 xArr[i] = xPos + (int)(Math.cos(Math.toRadians(i \* part + startAngle)) \* radius);  
 yArr[i] = yPos + (int)(Math.sin(Math.toRadians(i \* part + startAngle)) \* radius);  
 }  
 else{  
 xArr[i] = xPos + (int)(Math.cos(Math.toRadians(i \* part + startAngle)) \* smallradius);  
 yArr[i] = yPos + (int)(Math.sin(Math.toRadians(i \* part + startAngle)) \* smallradius);  
 }  
 }  
  
 }  
  
 public void doMove() {  
 startAngle += step;  
 if (startAngle % 360 == 0)  
 startAngle = 0;  
  
 createStar(xArr,yArr,counts);  
 repaint();  
 }  
  
 public void paint(Graphics gr){  
 gr.setColor(bgColor);  
 gr.fillRect(0,0,this.getWidth(),this.getHeight());  
 gr.setColor(startColor);  
 gr.fillPolygon(xArr,yArr,counts \* 2);  
 }  
  
 public static void main(String[]args){  
 Star s = new Star();  
 s.setVisible(true);  
 }  
}