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 {

public static void main(String[] args){

String text="";

Scanner in = new Scanner(System.in);

while(in.hasNextLine()){

String buffer = in.nextLine();

if(buffer.equals("exit"))

break;

text +=buffer;

}

ArrayList<String> Lq = new ArrayList<String>();

boolean q = false;

String bufferQ="";

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

if(text.charAt(i)=='"' && q==false)

q = true;

else if (text.charAt(i)=='"' && q == true){

bufferQ+='"';

Lq.add(bufferQ);

bufferQ = "";

q = false;

}

if(q)bufferQ+=text.charAt(i);

}

for(String s : Lq)

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\_1 {

public static void main(String[] args){

Scanner in = new Scanner(System.in);

ArrayList<String> Words = new ArrayList<String>();

while(in.hasNextLine()){

String buffer = in.nextLine();

if(buffer.equals("exit"))

break;

StringTokenizer str = new StringTokenizer(buffer," .,!?\n\r", false);

while(str.hasMoreTokens()) {

String word = str.nextToken();

if (Character.isUpperCase(word.charAt(0)))

Words.add(word);

}

}

for(String s : Words)

System.out.println(s);

}

}

Вариант 3.

Программа читает текст из стандартного входного потока, разбивает его на предложения и выводит их в стандартный выходной поток по одному предложению в строке, переводя все слова восклицательных предложений в верхний регистр.

import java.util.ArrayList; + import java.util.Scanner;

public class task\_3 {

static ArrayList<String> tl = new ArrayList<String>(); //list of lines

static void procLine(String line) {

int idx = 0;

while(idx<line.length()) {

if(line.charAt(idx) == '.' || line.charAt(idx) == '?'){

tl.add(line.substring(0, idx+1));

line = line.substring(idx+1); idx = 0;

} else

if(line.charAt(idx) == '!') {

tl.add(line.substring(0, idx+1).toUpperCase());

line = line.substring(idx+1);

idx = 0;

}

idx++;

}

}

public static void main(String[] args){

Scanner sc = new Scanner(System.in);

System.out.println("Insert your text line by line:");

String text = "";

while(sc.hasNextLine()){

String buffer = sc.nextLine();

if(buffer.equals("exit"))

break;

text = text + buffer;

}

procLine(text);

sc.close();

for(String s : tl) { System.out.println(s); } } }Вариант 4.

Стандартный входной поток. Найти все вопросительные предложения и вывести каждый с новой строки.

import java.util.ArrayList; + import java.util.Scanner;

public class task\_4 {

static ArrayList<String> tl = new ArrayList<String>(); //list of lines

static String tmpLine = new String();

static void procLine(String line) {

int idx = 0;

while(idx<line.length()) {

if(line.charAt(idx) == '.' || line.charAt(idx) == '!') {

tmpLine = "";

line = line.substring(idx+1);

idx = 0;

} else

if(line.charAt(idx) == '?') {

tmpLine += line.substring(0, idx+1);

tl.add(tmpLine);

tmpLine = "";

line = line.substring(idx+1);

idx = 0;

}

idx++;

}

if(line.indexOf('.') == -1 && line.indexOf('!') == -1 && line.indexOf('?') == -1) {

tmpLine += line;

}

}

public static void main(String[] args){

Scanner sc = new Scanner(System.in);

System.out.println("Insert your text line by line:");

tmpLine = "";

while(sc.hasNextLine()){

String buffer = sc.nextLine();

if(buffer.equals("exit"))

break;

procLine(buffer);

}

if(!tmpLine.isEmpty())

tl.add(tmpLine);

sc.close();

for(String s : tl) { System.out.println(s); } } }

Вариант 5.

найти длинные числа, отсортировать и вывести с номерами строк, в которых они встречались

import java.util.Comparator; + java.util.List; + java.util.Scanner; + java.util.StringTokenizer; + java.util.Vector;

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 (new Long(value)).compareTo(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 e){}

return null;

}

}

static void procLine(String line,int pos){

StringTokenizer st = new StringTokenizer(line," \n\t");

while(st.hasMoreTokens()){

String word = new String(st.nextToken());

Object obj = WLongInt.parseLong(word,pos);

if(obj != null){

li.add((WLongInt) 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()){

String buffer = sc.nextLine();

if(buffer.equals("exit"))

break;

procLine(buffer,n++);

}

sc.close();

li.sort(null);

for(WLongInt i:li)

System.out.print(i); } }

Вариант 6.

Программа читает из входного потока текст, находит в нём слова, состоящие только из цифр, и выводит их через выходной поток по одному слову в строке.

import java.util.\*;

public class task\_6 {

public static boolean check(String userNameString) {

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

if (Character.isDigit(userNameString.charAt(i)) == false)

return false;

return true;

}

public static void main (String args[]) {

System.out.println("Введите данные. Для окончания ввода жми Ctrl + z");

Scanner in = new Scanner(System.in);

ArrayList<String> List = new ArrayList<String>();

while (in.hasNextLine()) {

String str = in.nextLine();

StringTokenizer strtok = new StringTokenizer(str);

while (strtok.hasMoreTokens()) {

String word = strtok.nextToken();

if (check(word) == true) List.add(word);

}

}

if (List.isEmpty() == true) System.out.println("Нет слов, состоящих из цифр!");

else {

for (String i: List)

System.out.println(i);

}

in.close();

System.exit(0);

}

}

Вариант 7.

В тексте найти слова, состоящие из цифр, вывести их шестнадцатиричные записи, каждую запись выводить с новой строки

import java.util.ArrayList;

import java.util.Scanner;

import java.util.StringTokenizer;

public class task\_7 {

static ArrayList<String> hexs = new ArrayList<String>();

static void processLine (String line) {

StringTokenizer sttok = new StringTokenizer(line);

while (sttok.hasMoreTokens()) {

String word = sttok.nextToken();

if ( word.matches("[0-9]+") ) {

try {

String hex = Integer.toHexString(Integer.parseInt(word));

hexs.add(word + ": " + hex);

}

catch (Exception e) {

hexs.add(word + " can't be converted.");

}

}

}

}

public static void main(String[] args) {

System.out.println("Enter text or Ctrl+Z:");

Scanner in = new Scanner(System.in);

while (in.hasNextLine()) {

String line = in.nextLine();

processLine(line);

}

in.close();

System.out.println("Hexs:" + ((hexs.size() == 0)?" ...":""));

for (String word: hexs)

System.out.println(word);

System.out.println("the end.");

}

}

Вариант 8.

Слова разделены пробелами. Все слова в строке которые состоят только из букв отсортировать и сохранять количество пробелов

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

import java.util.Scanner;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class task\_8 {

//Slova razdeleny probelami. Vse slova v stroke kotorye sostoyat

//tol'ko iz bukv otsortirovat' i sohranyat' kolichestvo probelov

public static void main(String[] args) {

Scanner in=new Scanner(System.in);

ArrayList <String> result= new ArrayList <String>();

while (in.hasNext()) {

String stroka = in.nextLine();

if(stroka.equals("exit"))

break;

ArrayList <String> str= new ArrayList <String>();

String[] arr = stroka.split(" ");

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

if (isWord(arr[i])) {

str.add(arr[i]);

arr[i] = null;

}

Collections.sort(str,new Comparator <String> () {

public int compare(String o1, String o2) { return o1.compareTo(o2); } });

int j = 0;

String s = "";

for (String i : arr) {

if (i == null)

s += str.get(j++);

else

s += i;

s += " ";

}

result.add(s);

}

for(String i : result)

System.out.println(i);

}

private static boolean isWord(String word) {

Pattern p = Pattern.compile("^[A-z,A-y]+");

Matcher m = p.matcher(word);

return m.matches();

}

}

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.ArrayList; + java.util.Scanner; + java.util.StringTokenizer;

class Word {

String str;

int begPos, endPos;

public Word (String str, int begPos) {

this.str = str;

this.begPos = begPos;

endPos = begPos + str.length() - 1;

}

}

public class task\_9 {

static ArrayList<String> lines = new ArrayList<String>();

static String getReverseLine (String line) {

StringBuffer res = new StringBuffer(line);

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

ArrayList<String> rList = new ArrayList<String>();

StringTokenizer sttok = new StringTokenizer(line);

int pos = 0;

while (sttok.hasMoreElements()) {

String token = sttok.nextToken();

pos = line.indexOf(token, pos);

list.add(new Word(token, pos));

rList.add(token);

pos += token.length();

}

for (int i = list.size() - 1; i >= 0; --i)

res.replace(list.get(i).begPos, list.get(i).endPos + 1, rList.get(list.size() - 1 - i));

return res.toString();

}

public static void main(String[] args) {

System.out.println("Введите текст, для окончания используйте Ctrl+Z:");

Scanner in = new Scanner(System.in);

while (in.hasNextLine()) {

String line = in.nextLine();

lines.add(getReverseLine(line));

}

in.close();

System.out.println("Text:");

for (String item: lines)

System.out.println(item);

if ( lines.size() == 0 )

System.err.println("Текста нет...");

System.out.println("Программа завершена успешно.");

}

}

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.ArrayList; + java.util.Scanner;

class task\_9 {

static ArrayList<String> tl = new ArrayList<String>(); //list of lines

static void procLine(String text, String arg){

int idx = 0;

while(idx<text.length()){

if(text.charAt(idx) == '.' || text.charAt(idx) == '?' || text.charAt(idx) == '!') {

if(text.substring(0, idx+1).contains(arg))

tl.add(text.substring(0, idx+1));

text = text.substring(idx+1);

idx = 0;

}

idx++;

}

}

public static void main(String[] args){

if ( args.length != 1 )

throw new IllegalArgumentException("You don't enter one argument.");

Scanner sc = new Scanner(System.in);

System.out.println("Insert your text line by line:");

String text = "";

while(sc.hasNextLine()){

String buffer = sc.nextLine();

if(buffer.equals("exit"))

break;

text = text + buffer;

}

procLine(text, args[0]);

sc.close();

System.out.println("---Full text---: ");

for(String s : tl) { System.out.println(s); }

System.out.println("------end------\n"); } }

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 Palindrom implements Comparable<Palindrom> {

String str;

int oldBegPos, oldEndPos;

public Palindrom (String str, int oldBegPos) {

this.str = str;

this.oldBegPos = oldBegPos;

this.oldEndPos = oldBegPos + str.length() - 1;

}

public Palindrom (Palindrom p) {

str = p.str;

oldBegPos = p.oldBegPos;

oldEndPos = p.oldEndPos;

}

public int compareTo (Palindrom p) {

return str.compareTo(p.str);

}

public static boolean isPalindrom (String str) {

String rev = "";

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

rev = rev.concat("" + str.charAt(i));

return rev.equals(str);

}

}

public class task\_11 {

static ArrayList<String> text = new ArrayList<String>();

static String getDelim (String str) {

String res = "";

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

char ch = str.charAt(i);

if ( !Character.isLetter(ch) )

res = res.concat("" + ch);

}

return res;

}

static String processLine (String line) {

StringBuilder res = new StringBuilder(line);

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

ArrayList<Palindrom> sortlist = new ArrayList<Palindrom>();

StringTokenizer sttok = new StringTokenizer(line, getDelim(line));

int endPos = 0;

while (sttok.hasMoreElements()) {

String word = sttok.nextToken();

endPos = line.indexOf(word, endPos);

if ( Palindrom.isPalindrom(word) ) {

Palindrom p = new Palindrom(word, endPos);

list.add(p);

sortlist.add(p);

}

endPos += word.length();

}

Collections.sort(sortlist);

for (int i = sortlist.size() - 1; i >= 0; --i) {

res.replace(list.get(i).oldBegPos, list.get(i).oldEndPos + 1, sortlist.get(i).str);

}

return res.toString();

}

public static void main(String[] args) {

System.out.println("Enter text or Ctrl+Z:");

Scanner in = new Scanner(System.in);

while (in.hasNextLine()) {

String line = in.nextLine();

if(line.equals("exit"))

break;

text.add(processLine(line));

}

in.close();

System.out.println("\nText:");

if ( text.size() == 0 )

System.err.println("....");

for (String line: text)

System.out.println(line);

System.out.println("The end."); } }

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.ArrayList; + java.util.Collections; + java.util.Scanner;

import java.util.StringTokenizer;

class DigitWord implements Comparable <DigitWord> {

String str;

int beginPos, endPos, sum;

public DigitWord (String str, int beginPos) {

this.str = str;

this.beginPos = beginPos;

endPos = beginPos + str.length() - 1;

sum = 0;

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

sum += str.charAt(i) - '0';

}

public int compareTo (DigitWord dw) {

return (sum < dw.sum) ? -1 : (sum == dw.sum) ? 0 : 1;

}

public static boolean isDigitWord (String word) {

String delim = "0123456789";

StringTokenizer sttok = new StringTokenizer(word, delim);

return !sttok.hasMoreElements();

}

}

public class task\_12 {

static String processLine (String line) {

StringBuilder res = new StringBuilder(line);

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

ArrayList<DigitWord> sortList = new ArrayList<DigitWord>();

StringTokenizer sttok = new StringTokenizer(line, " \t");

int pos = 0;

while (sttok.hasMoreElements()) {

String word = sttok.nextToken();

pos = line.indexOf(word, pos);

if ( DigitWord.isDigitWord(word) ) {

DigitWord dw = new DigitWord(word, pos);

list.add(dw);

sortList.add(dw);

}

pos += word.length();

}

Collections.sort(sortList);

for (int i = sortList.size() - 1; i >= 0; --i)

res.replace(list.get(i).beginPos, list.get(i).endPos + 1, sortList.get(i).str);

return res.toString();

}

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

public static void main(String[] args) {

System.out.println("Enter text or Ctrl+Z:");

Scanner in = new Scanner(System.in);

while (in.hasNextLine()) {

String line = in.nextLine();

if(line.equals("exit"))

break;

list.add(processLine(line));

}

in.close();

System.out.println("\nText:");

if ( list.size() == 0 )

System.err.println("...");

for (String line: list)

System.out.println(line);

} }

Вариант 13.

заменить в строке 8 пробелов стоящих подряд на символ табуляции

import java.util.ArrayList;

import java.util.Scanner;

public class task\_13 {

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

public static void main(String[] args) {

String text;

Scanner sc = new Scanner(System.in);

StringBuilder sb = new StringBuilder();

System.out.println("Input your text");

while(sc.hasNext()){

String buffer = sc.nextLine();

if(buffer.equals("exit"))

break;

sb.append(buffer).append("\n");

}

text = sb.toString();

sc.close();

System.out.println(text.replaceAll(" {8}","\t")); //"^ {8}[^\s]$"

}

}

Вариант 14.

Ввести через стандартный поток текст, найти слово максимальной длины и вывести слово и сколько раз встречается

import java.util.ArrayList; + import java.util.Scanner;

public class task\_13 {

public static int maxCount=0;

public static String maxString;

public static void main(String[] args){

System.out.println("Begining of the work:");

try (Scanner in = new Scanner(System.in)){

read(in);

System.out.println(maxString+" "+maxCount);

}catch (Exception e){

System.err.println(e);

}

}

private static void read(Scanner in) throws Exception {

class Word {

String s;

int count;

public Word(String s, int c){

this.s = new String(s);

this.count=c;

}

}

ArrayList<Word> words = new ArrayList<Word>();

while (in.hasNext()) {

String buffer = in.nextLine();

if(buffer.equals("exit"))

break;

String[] arr = buffer.split(" ");

boolean f = true;

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

for (int j=0;j<words.size();j++){//проверяет, были ли уже такие слова

if(arr[i].compareTo(words.get(j).s)==0){

words.get(j).count++;

f=false;

break;

}

}

if(f){//слово новое

words.add(new Word(arr[i],1));

}

f=true;

}

}//тут можно и частотный словарь распечатать

for (int i=0;i<words.size();i++){

if(words.get(i).count > maxCount && words.get(i).s.compareTo("")!=0) {

maxString=words.get(i).s;

maxCount=words.get(i).count;

}

}

}

}

Вариант 15.

найти длинные числа, отсортировать и вывести с номерами строк, в которых они встречались

import java.util.Comparator; + java.util.List; + java.util.Scanner;

import java.util.StringTokenizer; + java.util.Vector;

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 (new Long(value)).compareTo(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 e){}

return null;

}

}

static void procLine(String line,int pos){

StringTokenizer st = new StringTokenizer(line," \n\t");

while(st.hasMoreTokens()){

String word = new String(st.nextToken());

Object obj = WLongInt.parseLong(word,pos);

if(obj != null){

li.add((WLongInt) 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()){

String buffer = sc.nextLine();

if(buffer.equals("exit"))

break;

procLine(buffer,n++);

}

sc.close();

li.sort(null);

for(WLongInt i:li) System.out.print(i); } }

Вариант 16.

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 MainClass {

public static final HashMap<String, Integer> OPERATIONS = new HashMap<>();

public static boolean isNumber(String num) {

try {

Double.parseDouble(num);

} catch (NumberFormatException exc) {

return false;

}

return true;

}

public static String toRPN(String expr) {

Stack<String> stack = new Stack<>();

StringTokenizer tokenizer = new StringTokenizer(expr);

String result = "";

String token;

while (tokenizer.hasMoreTokens()) {

token = tokenizer.nextToken();

if (isNumber(token)) {

result += token + " ";

}

if (token.equals("(")) {

stack.push(token);

}

if (token.equals(")")) {

try {

while (!stack.empty() && !stack.peek().equals("(")) {

result += stack.pop() + " ";

}

stack.pop();

} catch (Exception e) {

System.err.println("?");

System.exit(1);

}

}

if (OPERATIONS.containsKey(token)) {

try {

if (!(token.equals("(") || token.equals(")"))) {

while (!stack.empty() && (OPERATIONS.get(stack.peek()) > OPERATIONS.get(token) || OPERATIONS.get(stack.peek()) == OPERATIONS.get(token))) {

result += stack.pop() + " ";

}

stack.push(token);

}

} catch (Exception e) {

System.err.println("?");

System.exit(1);

}

}

}

while (!stack.empty()) {

try {

result += stack.pop() + " ";

} catch (Exception e) {

System.err.println("?");

System.exit(1);

}

}

return result;

}

public static void evalRPN(String cleanExpr) {

Stack<Double> stack = new Stack<>();

StringTokenizer tokenizer = new StringTokenizer(cleanExpr);

String token;

while (tokenizer.hasMoreTokens()) {

token = tokenizer.nextToken();

Double tokenNum = null;

try {

tokenNum = Double.parseDouble(token);

} catch (NumberFormatException e) {

}

if (tokenNum != null) {

stack.push(Double.parseDouble(token + ""));

} else if (token.equals("\*")) {

double secondOperand = stack.pop();

double firstOperand = stack.pop();

stack.push(firstOperand \* secondOperand);

} else if (token.equals("/")) {

double secondOperand = stack.pop();

double firstOperand = stack.pop();

stack.push(firstOperand / secondOperand);

} else if (token.equals("-")) {

double secondOperand = stack.pop();

double firstOperand = stack.pop();

stack.push(firstOperand - secondOperand);

} else if (token.equals("+")) {

double secondOperand = stack.pop();

double firstOperand = stack.pop();

stack.push(firstOperand + secondOperand);

} else {//just in case

System.out.println("?");

return;

}

}

try {

System.out.println(stack.pop());

} catch (Exception e) {

System.err.println("?");

System.exit(1);

}

}

public static void main(String[] args) {

OPERATIONS.put("(", 0);

OPERATIONS.put(")", 0);

OPERATIONS.put("-", 1);

OPERATIONS.put("+", 1);

OPERATIONS.put("\*", 2);

OPERATIONS.put("/", 2);

Scanner scanner = new Scanner(System.in);

String cleanExpr = scanner.nextLine();

evalRPN(toRPN(cleanExpr));

}

}

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.Random;

import java.util.Scanner;

public class task\_17 {

static int[][] getMatrix (int n) {

int[][] matrix = new int[n][n];

for (int i = 0; i < n; ++i)

for (int j = 0; j < n; ++j)

matrix[i][j] = new Random().nextInt(2\*n+1) - n;

return matrix;

}

public static void main(String[] args) {

System.out.println("Enter matrix dimension:");

Scanner in = new Scanner(System.in);

int n = 0;

while (true) {

try {

n = in.nextInt();

if ( n <= 1 )

throw new IllegalArgumentException("n <= 1");

break;

}

catch (Exception e) {

System.out.println(e.getMessage() + "\nTry again...");

in.nextLine();

}

}

in.close();

int[][] matrix = getMatrix(n);

System.out.println("Matrix:");

for (int i = 0; i < n; ++i) {

for (int j = 0; j < n; ++j)

System.out.printf("%4d ", matrix[i][j]);

System.out.println();

}

for (int i = 0; i < n; ++i)

for (int j = i + 1; j < n; ++j) {

int tmp = matrix[i][j];

matrix[i][j] = matrix[j][i];

matrix[j][i] = tmp;

}

System.out.println("Transpon matrix:");

for (int i = 0; i < n; ++i) {

for (int j = 0; j < n; ++j)

System.out.printf("%4d ", matrix[i][j]);

System.out.println();

}

System.out.println("the end.");

}

}

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.io.\*;

public class Test {

public static final String fileNameIn = "in.txt";

public static final String fileNameOut = "out.txt";

public static void main(String[] args) {

try(BufferedReader reader = new BufferedReader(new FileReader(fileNameIn));

BufferedWriter writer = new BufferedWriter(new FileWriter(fileNameOut)))

{

String add = "";

while(reader.ready())

{

String line = reader.readLine();

String[] mas = transformLine(add + line);

line = mas[0];

writer.write(line + "\n");

add = mas[1];

while(add.length() > 80) {

String[] tmp = transformLine(add);

writer.write(tmp[0] + "\n");

add = tmp[1];

}

}

if(!add.equals(""))

{

writer.write(add + "\n");

}

}

catch(IOException e)

{

e.printStackTrace();

}

}

public static String[] transformLine(String line)

{

line = line.replaceAll("\\s+", " ").trim();

line = deleteAllSpaces(line);

boolean flag = true;

String[] mas = new String[2];

if(line.length() > 80)

{

mas = makeTransference(line);

}

else {

mas[0] = line;

mas[1] = "";

}

return mas;

}

public static String deleteSpace(String line, char c)

{

char space = ' ';

int index = line.length();

StringBuilder result = new StringBuilder(line);

while(true)

{

index = line.lastIndexOf(c, --index);

if(index == -1)

break;

if(index-1 != -1 && line.charAt(index-1) == space)

result.delete(index-1, index);

}

return result.toString();

}

public static String deleteAllSpaces(String line)

{

char[] separators = {',','.','!','?', ';',':',';'};

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

line = deleteSpace(line, separators[i]);

return line;

}

public static String[] makeTransference(String line)

{

int index = 80;

while(line.charAt(index) != ' ')

{

index--;

}

index++;

String word = line.substring(index, line.length());

StringBuilder old = new StringBuilder(line);

old.delete(index, line.length());

String[] mas = new String[2];

mas[0] = old.toString();

mas[1] = word;

return mas;

}

}