Если вы хотите присоединиться к решению задач, вы можете написать здесь свое имя напротив темы, которую хотите решить, и выложить сюда же решение. Перед кодом стоит написать текст задачи для быстрого поиска по доку.

Например:

9.11 Дано слово s1. Получить слово s2, образованное нечетными буквами слова s1.

```
String evenLetters(String s){
// тут код
}
```

Тема в задачнике Златопольского	Я прорешаю эту тему!
Строки. Работа с символами строки	Слава - done!
Строки. Обработка строк с использованием оператора цикла с параметром	Тарас - done!
Строки. Обработка строк с использованием операторов цикла с условием	Ульяна - done!
Строки. Изменение исходных строковых величин	Ульяна - done! *9.95-9.97, 9.99-9.100, 9.102, 9.105, 9.108-9.110, 9.117, 9.119, 9.121, 9.123-9.124, 9.127-9.133, 9.135-9.136 - не решала, так как они очень похожи на прорешенные*
Строки. Обработка цифр в строке	Фарид – done!
Строки. Задачи повышенной сложности	

Вложенные циклы. Организация вывода с использованием вложенных циклов	Ира – done! *8.9, 8.10 - не решала, мало шансов их появления на экзамене*
Вложенные циклы. Обработка данных во время ввода с использованием вложенных циклов	Слава
Вложенные циклы. Вложенные циклы и целые числа	Ира – done! *8.39 - 8.45, 8.58, 8.59 - не решала, мало шансов их появления на экзамене (там комбинаторика и расчеты)*
Двумерные массивы. Заполнение и вывод массива нестандартными методами	Ира
Двумерные массивы. Расчетные задачи	
Двумерные массивы. Нахождение максимума и минимума	
Двумерные массивы. Проверка условия после выполнения расчетов	
Двумерные массивы. Обработка массива с использованием операторов цикла с условием	
Двумерные массивы. Работа с квадратными массивами	
Двумерные массивы. Изменение исходного массива	
Двумерные массивы. Работа с несколькими массивами	
Двумерные массивы. Двумерные символьные массивы	Гоша *12.278,*
Функции и процедуры. Рекурсия	Ира, Тарас, Слава
Решенные задачи от Жанны	Жанна

Решение задач:

Строки. Работа с символами строки:

```
// 9.13 Дано слово. Вывести на экран его третий символ.
// 9.15 Дано слово. Вывести на экран его k-й символ.
public static void printSymbolAt(String sourceString, int position) {
	System.out.println(sourceString.charAt(position));
}
```

```
// 9.14. Дано слово. Вывести на экран его последний символ.
public static void printLastSymbol(String sourceString) {
	System.out.println(sourceString.charAt(sourceString.length()-1));
}
```

```
// 9.16 Дано слово. Определить, одинаковы ли второй и четвертый символы в нем.
public static boolean isCharsAreTheSame(String sourceString, int first, int second) {
	return sourceString.charAt(first) == sourceString.charAt(second);
}
```

```
// 9.17. Верно ли, что оно начинается и оканчивается на одну и ту же букву?
public static boolean isStringBeginsAndEndsOnTheSameChar(String sourceString) {
	return sourceString.charAt(0) == sourceString.charAt(sourceString.length()-1);
}
```

```
// 9.18. Даны два слова.
// Верно ли, что первое слово начинается на ту же букву, на ко-торую заканчивается второе
слово?
public static boolean isBeginsOnSecondStringEnd(String firstString, String secondString) {
	return firstString.charAt(0) == secondString.charAt(secondString.length()-1);
}
```

```
// 9.19. Получить и вывести на экран буквосочетание, состоящее из его второго и четвертого
символа.
// 9.20. Получить и вывести на экран буквосочетание, состоящее из его третьего и
последнего символа.
// 9.21. Получить его часть, образованную второй, третьей и четвертой буквами.
public static void printSpecifiedChars(String sourceString, int... numbers ){
	char[] chars = new char[numbers.length];
	int index = 0;
	for (int number : numbers) {
		chars[index] = sourceString.charAt(number);
```

```
                index++;
        }
        System.out.println(Arrays.toString(chars));
}
```

```
//9.22. Дано слово, состоящее из четного числа букв.
// Вывести на экран его первуюполовину, не используя оператор цикла.
public static void printTheHalfOfTheString(String sourceString) {
        System.out.println(sourceString.substring(0, sourceString.length()/2));
}
```

```
//9.23. Дано слово. Получить его часть, образованную идущими подряд буквами,
// начиная с m-й и кончая n-й.
public static void printThePartOfString(String sourceString, int from, int to) {
        System.out.println(sourceString.substring(from,to));
}
```

```
// 9.24 - 9.30 Из слова <слово> путем "вырезок" и "склеек" его букв получить слова
<слова>.
public static String getStringWithCutAndMerge(String sourceString, int... positions) {
        char[] chars = new char[positions.length];
        int index = 0;
        for (int position : positions) {
                chars[index] = sourceString.charAt(position);
                index++;
        }
        return new String(chars);
}
```

```
// 9.31 - 9.36 Из слова <слово> путем замены его букв получить слово <слово>.
public static String replaceString(String sourceString, String from, String to) {
        return sourceString.replace(from, to);
}
```

```
// 9.37, 9.38 - Дано слово, поменять местами его части
public static String[] splitString(String sourceString, int... points){
        String[] parts = new String[points.length+1];
        int last = 0;
        for (int i = 0; i < points.length; i++) {
                parts[i] = sourceString.substring(last,points[i]);
                last = points[i];
        }
        parts[parts.length-1] = sourceString.substring(last);
        return parts;
}
String[] parts = Symbols.splitString("abcdefghijkl",4,8);
System.out.println(parts[0] + "|" + parts[1] + "|" + parts[2]); //abcd|efgh|ijkl
System.out.println(parts[2] + "|" + parts[1] + "|" + parts[0]); //ijkl|efgh|abcd
```

```
//9.39. Дано слово. Переставить первые три и последние три буквы,
//сохранив порядок их следования. С использованием оператора цикла.
public static String reverseCharsInString(String sourceString, int count){
	char[] chars = sourceString.toCharArray();
	for (int i = count-1; i >= 0; i--) {
		int endPosition = chars.length-count+i;
		chars[i] = sourceString.charAt(endPosition);
		chars[endPosition] = sourceString.charAt(i);
	}
	return String.valueOf(chars);
}
```

```
//9.40. Дано слово. Перенести первые k его букв в конец.
// Решение без использования оператора цикла
public static String moveFirstCharsToTheEndVarOne(String sourceString, int count) {
	char[] chars = sourceString.toCharArray();
	char[] moved = Arrays.copyOf(chars,count);
	System.arraycopy(chars,count,chars,0,chars.length-count);
	System.arraycopy(moved,0,chars,chars.length-count,count);
	return new String(chars);
}

public static String moveFirstCharsToTheEndVarTwo(String sourceString, int count) {
	return sourceString.substring(count).concat(sourceString.substring(0,count));
}
```

Строки. Обработка строк с использованием оператора цикла с параметром:

9.41. Дано название футбольного клуба. Напечатать его на экране "столбиком".

```
void tablePrint(String clubName) {
   for (Character letter : clubName.toCharArray()
           ) {
       System.out.println(letter);
   }
}
```

9.42. Составить программу, которая печатает заданное слово, начиная с последней буквы.

```
void reversePrint(String word){
   for (int i = word.length() - 1; i >= 0; i--) {
       System.out.print(word.charAt(i));
   }
}
```

9.43. Дано слово s1. Получить слово s2, образованное нечетными буквами слова s1.

```
String oddLetters(String originalWord) {
   StringBuilder resultWord = new StringBuilder();
   for (int i = 0; i < originalWord.length(); i += 2) {
       resultWord.append(originalWord.charAt(i));
   }
   return resultWord.toString();
}
```

9.44. Дано слово s. Получить слово t, получаемое путем прочтения слова s начиная с его конца.

```
String reverse(String originalWord){
   StringBuilder resultWord = new StringBuilder(originalWord);
   return resultWord.reverse().toString();
}
```

9.45. Получить строку, состоящую из пяти звездочек (символов "*").

```
String fiveStars(){
   StringBuilder result = new StringBuilder();
   for (int i = 0; i < 5; i++) {
       result.append("*");
   }
   return result.toString();
}
```

9.46. Получить строку, состоящую из восьми символов "_".

```
String eightLines() {
   StringBuilder result = new StringBuilder();
   for (int i = 0; i < 8; i++) {
       result.append("_");
   }
   return result.toString();
}
```

9.47. Составить программу, формирующую строку, состоящую из любого заданного количества любых одинаковых символов.

```
String fillString(char symbol, int number) {
   StringBuilder result = new StringBuilder();
   for (int i = 0; i < number; i++) {
       result.append(symbol);
   }
   return result.toString();
}
```

9.48. Дано слово. Добавить к нему в начале четыре символа "+" и в конце — пять символов "–".

```
String appendPlusesAndMinusesToString(String originalString){
   StringBuilder result = new StringBuilder(originalString);
   return result.insert(0,"++++").append("-----").toString();
}
```

9.49. Дано слово. Добавить к нему в начале и конце столько звездочек, сколько букв в этом слове.

```
String appendStars(String originalString){
   StringBuilder result = new StringBuilder(originalString);
   for (int i = 0; i < originalString.length(); i++) {
       result.append("*");
   }
   return result.toString();
}
```

9.50. Даны два слова (первое длиннее второго). Заменить во втором слове соответствующее количество символов на первое слово.

```
String trimSecondWord(String firstWord, String secondWord){
   return firstWord.substring(0,secondWord.length());
}
```

9.51. Дано предложение. Напечатать все его буквы и.

```
void iLettersInSentence(String sentence){
   System.out.println(sentence.replaceAll("[^и]",""));
}
```

9.52. Дано предложение. Составить программу, которая печатает "столбиком" все вхождения в предложение некоторого символа.

```
void symbolPrint(String sentence, char symbol) {
   for (Character letter : sentence.toCharArray()) {
       if (letter == symbol) {
           System.out.println(letter);
       }
   }
}
```

9.53. Дано предложение. Вывести "столбиком" его третий, шестой и т. д. символы.

```
void everyThirdSymbolPrint(String sentence) {
   for (int i = 2; i < sentence.length(); i += 3) {
       System.out.println(sentence.charAt(i));
   }
}
```

9.54. Дано предложение. Вывести все буквы м и н в нем.

```
void mOrNLettersInSentence(String sentence) {
   System.out.println(sentence.replaceAll("[^(м|н)]", ""));
}
```

9.55. Дано предложение. Составить программу, которая выводит все вхождения в предложение двух заданных символов.

```
void specifiedSymbolsPrint(String sentence, char firstSymbol, char
secondSymbol) {
   for (Character symbol : sentence.toCharArray()
           ) {
       if (symbol == firstSymbol || symbol == secondSymbol) {
           System.out.println(symbol);
       }
   }
}
```

9.56. Дано предложение. Вывести все имеющиеся в нем буквосочетания нн.

```
void nnLettersInSentence(String sentence) {
   for (int i = 0; i < sentence.length() - 1; i++) {
       if (sentence.charAt(i) == 'н' && sentence.charAt(i + 1) ==
'н'){
           System.out.println("нн");
       }
   }
}
```

9.57. Дано предложение. Вывести "столбиком" все его буквы и, стоящие на четных местах.

```
void printEvenLetters(String sentence) {
   for (int i = 1; i < sentence.length(); i += 2) {
       System.out.println(sentence.charAt(i));
   }
}
```

9.58. Дано предложение. Вывести "столбиком" его первый, второй, пятый, шестой, девятый, десятый и т. д. символы.

```
void printSymbols(String sentence) {
   for (int i = 0; i < sentence.length(); i += 4) {
       System.out.println(sentence.charAt(i));
       System.out.println(sentence.charAt(i+1));
   }
}
```

9.59. Дано предложение. Определить число букв о в нем.

```
int alphabeticCount(String sentence) {
   int alphabetCount = 0;
   for (Character symbol :
           sentence.toCharArray()) {
       if (Character.isAlphabetic(symbol)) {
           alphabetCount++;
       }
   }
   return alphabetCount;
}
```

9.60. Дано предложение. Определить число пробелов в нем.

```
int spacesCount(String sentence) {
   return sentence.replaceAll("[^ ]","").length();
}
```

9.61. Дано предложение. Определить число вхождений в него некоторого символа.

```
int symbolCount(String sentence, char SearchedSymbol) {
   int symbolsCount = 0;
   for (Character symbol : sentence.toCharArray()
           ) {
       if (symbol == SearchedSymbol) {
           symbolsCount++;
       }
   }
   return symbolsCount;
}
```

9.62. Дано предложение. Определить долю (в %) букв а в нем.

```
double lettersPercent(String sentence) {
   int alphabetCount = 0;
   for (Character symbol :
           sentence.toCharArray()) {
       if (Character.isAlphabetic(symbol)) {
           alphabetCount++;
       }
   }
   return (double) alphabetCount / sentence.length() * 100;
}
```

9.63. Дан текст. Сколько раз в нем встречается символ "+" и сколько раз символ "*"?

```
void plusAndStarCounter(String sentence) {
   int starCounter = 0;
   int plusCounter = 0;
   for (Character symbol :
           sentence.toCharArray()) {
       if (symbol == '+') {
           plusCounter++;
       } else if (symbol == '*') {
           starCounter++;
       }
   }
   System.out.printf("plus: %d\nstar:
%d",plusCounter,starCounter);
}
```

9.64. Дано предложение. Определить, сколько в нем одинаковых соседних букв.

```
int doubleOccurrenceCounter(String sentence) {
   int doubleOccurrenceCounter = 0;
   for (int i = 0; i < sentence.length() - 1; i++) {
       if (Character.isAlphabetic(sentence.charAt(i)) &&
sentence.charAt(i) == sentence.charAt(i + 1)) {
           doubleOccurrenceCounter++;
       }
   }
   return doubleOccurrenceCounter;
}
```

9.65. Дано предложение. Определить:
а) число вхождений в него буквосочетания ро;

```
int roOccurrenceCounter(String sentence) {
   int roCounter = 0;
   for (int i = 0; i < sentence.length() - 1; i++) {
       if (sentence.charAt(i) == 'p' && sentence.charAt(i + 1) ==
'o') {
           roCounter++;
       }
   }
   return roCounter;
}
```

б) число вхождений в него некоторого буквосочетания из двух букв;

```
int occurrenceCounter(String sentence, String combination) {
   int startIndex = 0;
   int occurrenceCounter = 0;
   while (sentence.indexOf(combination, startIndex) != -1) {
       occurrenceCounter++;
       startIndex = sentence.indexOf(combination, startIndex) +
combination.length();
   }
   return occurrenceCounter;
}
```

в) число вхождений в него некоторого буквосочетания.

```
int occurrenceCounter(String sentence, String combination) {
   int startIndex = 0;
   int occurrenceCounter = 0;
   while (sentence.indexOf(combination, startIndex) != -1) {
       occurrenceCounter++;
       startIndex = sentence.indexOf(combination, startIndex) +
combination.length();
   }
   return occurrenceCounter;
}
```

9.66. Дано предложение. В нем слова разделены одним пробелом (начальные и конечные пробелы и символ "-" в предложении отсутствуют). Определить количество слов в предложении.

9.67. Дано предложение. В нем слова разделены одним или несколькими пробелами (символ "-" в предложении отсутствует). Определить количество слов в предложении. Рассмотреть два случая:

1) начальные и конечные пробелы в предложении отсутствуют;

```
int wordsCount(String sentence){
   return sentence.split(" ").length;
}
```

2) начальные и конечные пробелы в предложении имеются.

```
int wordsCount(String sentence){
   return sentence.trim().split(" ").length;
}
```

9.68. Дан текст. Подсчитать общее число вхождений в него символов "+" и "–".

```
int plusAndMinusCounter(String sentence) {
   int plusAndMinusCounter = 0;
   for (Character symbol :
           sentence.toCharArray()) {
       if (symbol == '+' || symbol == '-') {
           plusAndMinusCounter++;
       }
   }
   return plusAndMinusCounter;
}
```

9.69. Дан текст. Определить, сколько в нем предложений.

9.70. Дано предложение. Определить, сколько в нем гласных букв.

```
int vowelCounter(String sentence) {
   String vowels = "ауоыэиеяюё";
   int vowelsCounter = 0;
   for (Character letter : sentence.toCharArray()
           ) {
       if (vowels.contains(letter.toString())) {
           vowelsCounter++;
       }
   }
   return vowelsCounter;
}
```

9.71. Дано предложение. Определить, каких букв в нем больше: м или н.

```
void mOrNCounter(String sentence) {
   int mCounter = 0;
   for (Character symbol :
           sentence.toCharArray()) {
       if (symbol == 'м') {
           mCounter++;
       } else if (symbol == 'н') {
           mCounter--;
       }
   }
   System.out.println(mCounter > 0 ? "м" : "н");
}
```

9.72. Дано предложение. В нем слова разделены одним пробелом (символ "-" в предложении отсутствует). Верно ли, что число слов в предложении больше трех?

```
boolean moreThanThreeWords(String sentence) {
   return sentence.trim().split(" ").length > 3;
}
```

9.73. Дано предложение, в котором имеются буквы с и Т. Определить, какая из них встречается позже (при просмотре слова слева направо). Если таких букв несколько, то должны учитываться последние из них. Оператор цикла с условием не использовать.

```
void sToccurrence(String sentence) {
   System.out.println(sentence.lastIndexOf('c') >
sentence.lastIndexOf('T') ? "c" : "T");
}
```

9.74. Дан текст. Верно ли, что в нем есть пять идущих подряд одинаковых символов?

```
boolean fiveSymbols(String sentence) {
   int count = 0;
   for (int i = 0; i < sentence.length() - 1; i++) {
       if (sentence.charAt(i) == sentence.charAt(i + 1)) {
           count++;
       } else {
           count = 0;
       }
       if (count == 4) {
           return true;
       }
   }
   return false;
}
```

Строки. Обработка строк с использованием операторов **цикла с условием**

(почти нигде в решении нет цикла с условием, но на экзамене задачи будут даваться без этой оговорки.)

9.75. Дано предложение. Напечатать все его символы, предшествующие первой запятой. Рассмотреть два случая:
1) известно, что в предложении запятые имеются;

```
private void printAllSymbolsBeforeComma1(String sentence) {
      System.out.println(sentence.subSequence
            (0, sentence.indexOf(",")));
}
```

2) в предложении запятых может не быть.

```
private void printAllSymbolsBeforeComma2(String sentence) {
      if (sentence.contains(",")) {
            System.out.println(sentence.subSequence
                  (0, sentence.indexOf(",")));
      } else {
            System.out.println("No comma");
      }
}
```

9.76. Дано предложение, в котором имеется несколько букв *е*. Найти:
а) порядковый номер первой из них;
б) порядковый номер последней из них.

```
private void getFirstAndLastEIndex(String sentence) {
      System.out.println("First 'e': " +
            (sentence.indexOf("e") + 1));
      System.out.println("Last 'e': " +
            (sentence.lastIndexOf("e") + 1));
}
```

9.77. Дано предложение. Определить, есть ли буква *а* в нем. В случае положительного ответа найти также порядковый номер первой из них.

```
private void findALetter(String sentence) {
      if (sentence.contains("a")) {
            System.out.println("'a': " +
                  (sentence.indexOf("a") + 1));
```

```
      } else {
            System.out.println("No 'a' letter");
      }
}
```

9.78. Дано слово. Проверить, является ли оно "перевертышем" (*перевертышем* называется слово, читаемое одинаково как с начала, так и с конца). *Глава 9 102*

```
private void isPalindrome(String word) {
      System.out.println((new StringBuilder(word).
            reverse().toString().toLowerCase().
            equals(word.toLowerCase())) ?
            word + " - palindrome" : word + " - not palindrome");
}
```

9.79. Дан текст. Определить количество букв *и* в первом предложении. Рассмотреть два случая:
1) известно, что буквы *и* в этом предложении есть;

```
private void getNumberOfILetter1(String text) {
      String[] sentences = text.split("\\.");
      System.out.println("Number of 'и' letters: " +
            (sentences[0].length() -
            sentences[0].replaceAll("и", "").length()));
}
```

2) букв *и* в тексте может не быть.

```
private void getNumberOfILetter2(String text) {
      if (text.contains("и")) {
            String[] sentences = text.split("\\.");
            System.out.println("Number of 'и' letters: " +
                  (sentences[0].length() -
                  sentences[0].replaceAll("и", "").length()));
            } else {
                  System.out.println("No 'и' letter");
      }
}
```

9.80. Дана последовательность символов, в начале которой имеется некоторое количество одинаковых символов. Определить это количество. Рассмотреть два случая:
1) известно, что не все символы последовательности одинаковые;
2) все символы последовательности могут быть одинаковыми.

```
private void getNumberOfSameSymbols(String sequence) {
```

```
      int numberOfSameSymbols = 1;
      for (int i = 1; i < sequence.length(); i++) {
            if (sequence.charAt(i) == sequence.charAt(0)) {
                  numberOfSameSymbols++;
            } else {
                  break;
            }
      }
      System.out.println("Number of same symbols: "
            + numberOfSameSymbols);
}
```

9.81. Даны два слова. Определить, сколько начальных букв первого слова совпадает с начальными буквами второго слова.
Рассмотреть два случая:
1) известно, что слова разные;
2) слова могут быть одинаковыми.

```
private void compareWordsBeginnings(String word1, String word2) {
      int numberOfSameSymbols = 0;
      for (int i = 0; i < word1.length()
            && i < word2.length(); i++) {
            if (word1.charAt(i) == word2.charAt(i)) {
                  numberOfSameSymbols++;
            } else {
                  break;
            }
      }
      System.out.println("Number of same symbols: "
            + numberOfSameSymbols);
}
```

9.82. Дано предложение, в котором нет символа "-". Определить количество букв *о* в первом слове. Учесть, что в начале предложения могут быть пробелы.

```
private void getNumberOfOLetter(String sentence) {
      String[] words = sentence.trim().split(" ");
      System.out.println("Number of 'o' letters: " +
            (words[0].length() -
            words[0].trim().replaceAll("o", "").length()));
}
```

9.83. Дано предложение. Определить количество букв *н*, предшествующих первой запятой предложения. Рассмотреть два случая:
1) известно, что запятые в предложении есть;

```
private void findAllNLettersBeforeComma1(String sentence) {
      System.out.println("Number of 'н' letter before comma: " +
            (sentence.substring(0, sentence.indexOf(",")).length() -
                  sentence.substring(0, sentence.indexOf(",")).
                  replaceAll("н", "").
                  length()));
}
```

2) запятых в предложении может не быть.

```
private void findAllNLettersBeforeComma2(String sentence) {
      if (sentence.contains(",")) {
            System.out.println("Number of 'н' letter before comma: "
                  + (sentence.substring(0,sentence.indexOf(",")).
                  length()
                  - sentence.substring(0,sentence.indexOf(",")).
                  replaceAll("н","").length()));
      } else {
            System.out.println("No comma");
      }
}
```

9.84. Дано предложение. Определить порядковые номера первой пары одинаковых соседних символов. Если таких символов нет, то должно быть напечатано соответствующее сообщение.

```
private void findPairOFSameSymbols(String sentence) {
      for (int i = 0; i < sentence.length() - 1; i++) {
            if (sentence.charAt(i) == sentence.charAt(i + 1)) {
                  System.out.println("First pair: " + (i + 1) +
                        " and " + (i + 2));
                  break;
            }
      }
}
```

9.85. Дано предложение. Определить, есть ли в нем буквосочетания *чу* или *щу*. В случае положительного ответа найти также порядковый номер первой буквы первого из них.

```
private void findChuShchu(String sentence) {
      sentence = sentence.toLowerCase();
      int i = 0;
      while ((i < sentence.length() - 1) &&
            (sentence.charAt(i) != 'ч' || sentence.charAt(i) != 'щ')
            && (sentence.charAt(i + 1) != 'у')) {
```

```
            i++;
      }
      if (i == 0) {
            System.out.println("No 'чу/щу'");
      } else {
            System.out.println("First letter: " + (i + 1));
      }
}
```

9.86. Дана последовательность слов. Проверить, правильно ли в ней записаны буквосочетания *жи* и *ши*.

```
private void checkZshiShi(String sentence) {
      sentence = sentence.toLowerCase();
      if (sentence.contains("жы") || sentence.contains("шы")) {
            System.out.println("ERROR!");
      } else {
            System.out.println("correct");
      }
}
```

9.87. Дана последовательность слов. Проверить, правильно ли в ней записаны буквосочетания *ча* и *ща*. Исправить ошибки.

```
private void checkChaShcha(String sentence) {
      if (sentence.contains("чя") || sentence.contains("щя")
            || sentence.contains("Чя") || sentence.contains("Щя")) {
            System.out.println("ERROR!");
            System.out.println("Correct sentence: " + sentence.
                  replaceAll("чя", "ча").
                  replaceAll("щя", "ща").
                  replaceAll("Чя", "Ча").
                  replaceAll("Щя", "Ща"));
      } else {
            System.out.println("correct");
      }
}
```

9.88. Дано предложение. Напечатать все символы, расположенные между первой и второй запятой. Если второй запятой нет, то должны быть напечатаны все символы, расположенные после единственной имеющейся запятой.

```
private void findAllSymbolsBetweenCommas(String sentence) {
      String[] parts = sentence.split(",");
      System.out.println(parts[1]);
}
```

9.89. Дано предложение, в котором имеются одна буква *с* и одна буква *Т*. Определить, какая из них встречается раньше (при просмотре слова слева направо).

```
private void findFirstCOrTLetter(String sentence) {
      String whoIsFirst =
            sentence.toLowerCase().indexOf('c') <
            sentence.toLowerCase().indexOf('т') ?
            "'c'" : "'т'";
      System.out.println(whoIsFirst + " is earlier");
}
```

Строки. Изменение исходных строковых величин

9.90. Дано предложение. Все буквы е в нем заменить буквой и.

```
private void replaceE(String text) {
    System.out.println(text.replaceAll("е", "и"));
}
```

9.91. Дано предложение. Все пробелы в нем заменить символом "_".

```
private void replaceSpace(String text) {
    System.out.println(text.replaceAll(" ", "_"));
}
```

9.92. Дано предложение. Все его символы, стоящие на четных местах, заменить буквой ы.

```
private void replaceEvenSymbols(String text) {
    char[] newString = new char[text.length()];
    for (int i = 0; i < text.length(); i++) {
        if ((i + 1) % 2 != 0) {
            newString[i] = text.charAt(i);
        } else {
            newString[i] = 'ы';
        }
    }
    System.out.println(newString);
}
```

9.93. Дано предложение. Все его символы, стоящие на третьем, шестом, девятом и т. д. местах, заменить буквой а.

```
private void replaceThirds(String text) {
    char[] newString = new char[text.length()];
    for (int i = 0; i < text.length(); i++) {
        if ((i + 1) % 3 != 0) {
            newString[i] = text.charAt(i);
        } else {
            newString[i] = 'a';
        }
    }
    System.out.println(newString);
```

```
}
```

9.94. Дано предложение. Заменить в нем все вхождения буквосочетания ах на ух.

```
private void replaceAh(String text) {
    System.out.println(text.replaceAll("ax", "yx"));
}
```

9.98. Дано предложение. Заменить в нем все вхождения подстроки s1 на подстроку s2..

```
private void replaceAh(String text, String s1, String s2) {
    System.out.println(text.replaceAll(s1, s2));
}
```

9.101. Дано слово. Поменять местами его третью и последнюю буквы.

```
private void swapLetters(StringBuilder sb) {
    char tmp = sb.charAt(2);
    sb.setCharAt(2, sb.charAt(sb.length() - 1));
    sb.setCharAt((sb.length() - 1), tmp);
    System.out.println(sb);
}
```

9.103. Дано слово из четного числа букв. Поменять местами первую букву со второй, третью — с четвертой и т. д.

```
private void swapAllLetters(StringBuilder sb) {
    for (int i = 0; i < (sb.length() - 1); i += 2) {
        char tmp = sb.charAt(i);
        sb.setCharAt(i, sb.charAt(i + 1));
        sb.setCharAt((i + 1), tmp);
    }
    System.out.println(sb);
}
```

9.104. Дано слово из четного числа букв. Поменять местами его половины следующим способом:
первую букву поменять с последней, вторую — с предпоследней и т. д.

```
private void swapWord(StringBuilder sb) {
    for (int i = 0; i < sb.length() / 2; i++) {
        char tmp = sb.charAt(i);
        sb.setCharAt(i, sb.charAt(sb.length() - 1 - i));
        sb.setCharAt((sb.length() - 1 - i), tmp);
    }
```

```
        System.out.println(sb);
    }
```

9.106. User input for 9.106

```
    private int[] getUserRange() {
        int userInt1 = 0;
        int userInt2 = 0;
        BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));
        try {
            do {
                System.out.println("Enter first number from 1 to
14: ");
                userInt1 = Integer.parseInt(reader.readLine());
                System.out.println("Enter second number from 2 to
15: ");
                userInt2 = Integer.parseInt(reader.readLine());
            } while (userInt1 > 14 || userInt1 < 1 ||
                    userInt2 > 15 || userInt2 < 2 ||
                    userInt1 > userInt2);
        } catch (IOException e) {
            e.printStackTrace();
        }
        return new int[]{userInt1, userInt2};
    }
```

9.106. Дано слово из 15-ти букв.
Переставить в обратном порядке буквы, расположенные между k-й и s-й буквами (т.е. с (k + 1)-й по (s – 1)-ю).
Значения k и s вводятся с клавиатуры, k < s.

```
    private void swapFromKToS(StringBuilder sb, int[] range) {
        int i, j;
        for (i = range[0], j = range[1] - 2; i < j; i++, j--) {
            char tmp = sb.charAt(i);
            sb.setCharAt(i, sb.charAt(j));
            sb.setCharAt(j, tmp);
        }
        System.out.println(sb);
    }
```

9.107. Дано слово. Поменять местами первую из букв а и последнюю из букв о.
Учесть возможность того, что таких букв в слове может не быть.

```
  private void replaceOAndA(StringBuilder sb) {
        if (sb.toString().contains("a") &&
sb.toString().contains("o")) {
```

```
            int firstA = sb.indexOf("a");
            int lastO = sb.lastIndexOf("o");
            sb.setCharAt(firstA, 'o');
            sb.setCharAt(lastO, 'a');
            System.out.println(sb);
        } else {
            System.out.println("No 'a' or/and 'o' letter");
        }
    }
```

9.111. Дано слово.
Если его длина нечетная, то удалить среднюю букву, в противном случае — две средних буквы.

```
    private void deleteMiddle(StringBuilder sb) {
        if (sb.length() % 2 != 0) {
            sb.deleteCharAt(sb.length() / 2).append("_");
        } else {
            sb.delete(sb.length() / 2 - 1, sb.length() / 2 +
1).append("__");
        }
        System.out.println(sb);
    }
```

9.112. Дано предложение. Удалить из него все символы с n1-го по n2-й (n1 n2).

```
    private void deleteFromN1ToN2(StringBuilder sb, int n1, int
n2) {
        int oldLength = sb.length();
        sb.delete(n1 - 1, n2);
        int newLength = sb.length();
        for (int i = 0; i < oldLength - newLength; i++) {
            sb.append("_");
        }
        System.out.println(sb);
    }
```

9.113. Дано предложение. Удалить из него все буквы c.

```
    private void deleteS(StringBuilder sb) {
        for (int i = 0; i < sb.length(); i++) {
            if (sb.charAt(i) == 'c') {
                sb.deleteCharAt(i).append("_");
            }
        }
        System.out.println(sb);
    }
```

9.114. Дано слово. Удалить из него все повторяющиеся буквы, оставив их первые вхождения, т. е. в слове должны остаться только различные буквы.

```
    private void deleteSameLetters(StringBuilder sb) {
        for (int i = 0; i < sb.length() - 1; i++) {
            for (int j = i + 1; j < sb.length(); j++) {
                if (sb.charAt(i) == sb.charAt(j)) {
                    sb.deleteCharAt(j).append("_");
                }
            }
        }
        System.out.println(sb);
    }
```

9.115. Дано предложение. Удалить из него все буквы о, стоящие на нечетных местах.

```
    private void deleteOddO(StringBuilder sb) {
        ArrayList<Integer> forDelete = new ArrayList<>();
        for (int i = 0; i < sb.length(); i += 2) {
            if (sb.charAt(i) == 'o') {
                forDelete.add(i);
            }
        }
        for (int i = forDelete.size() - 1; i >= 0; i--) {
            sb.deleteCharAt(forDelete.get(i)).append("_");
        }
        System.out.println(sb);
    }
```

9.116
Проверить, является ли "перевертышем" следующая символьная строка после удаления из нее всех пробелов:
а) АРГЕНТИНА МАНИТ НЕГРА;
б) ПОТ КАК ПОТОП;
в) А РОЗА УПАЛА НА ЛАПУ АЗОРА.

```
  private void isPalindrome() {
        String stringArgentina = "АРГЕНТИНА МАНИТ НЕГРА";
        String stringSweat = "ПОТ КАК ПОТОП";
        String stringRose = "А РОЗА УПАЛА НА ЛАПУ АЗОРА";

        String stringArgentinaWithoutSpaces =
stringArgentina.replaceAll(" ", "");
        String stringSweatWithoutSpaces = stringSweat.replaceAll("
", "");
        String stringRoseWithoutSpaces = stringRose.replaceAll("
", "");
```

```
        System.out.println((new
StringBuilder(stringArgentinaWithoutSpaces).
                reverse().toString().
                equals(stringArgentinaWithoutSpaces)) ?
                stringArgentina + " - palindrome" :
stringArgentina + " - not palindrome");
        System.out.println((new
StringBuilder(stringSweatWithoutSpaces).
                reverse().toString().
                equals(stringSweatWithoutSpaces)) ?
                stringSweat + " - palindrome" : stringSweat + " -
not palindrome");
        System.out.println((new
StringBuilder(stringRoseWithoutSpaces).
                reverse().toString().
                equals(stringRoseWithoutSpaces)) ?
                stringRose + " - palindrome" : stringRose + " -
not palindrome");
    }
```

9.118. Дано слово стекляный_. Исправить ошибку в нем.

```
    private void correctWord(StringBuilder sb) {
        int offset = sb.indexOf("н");
        sb.insert(offset, "н").deleteCharAt(sb.length() - 1);
        System.out.println(sb);
    }
```

9.120. Дано слово, оканчивающееся символом "_". Вставить букву т после k-й буквы.

```
    private void insertM(StringBuilder sb, int k) {
        sb.insert(k, "m").deleteCharAt(sb.length() - 1);
        System.out.println(sb);
    }
```

9.122. Дано слово, оканчивающееся символом "_".
Вставить заданную букву после первой буквы и.

```
    private void insertLetter(StringBuilder sb, char symbol) {
        sb.insert(sb.indexOf("и") + 1,
symbol).deleteCharAt(sb.length() - 1);
        System.out.println(sb);
    }
```

9.125. Дано ошибочно написанное слово рпроцессо.
Путем перемещения его букв получить слово процессор.

```
    private void correctProcessor() {
        StringBuilder proc = new StringBuilder("рпроцессо");
        proc.append(proc.charAt(0)).deleteCharAt(0);
        System.out.println(proc);
    }
```

9.126. Дано слово. Переставить его первую букву на место последней.
При этом вторую, третью, ..., последнюю буквы сдвинуть влево на одну позицию.

```
    private void moveFirstToEnd(StringBuilder sb) {
        sb.append(sb.charAt(0)).deleteCharAt(0);
        System.out.println(sb);
    }
```

9.134. Дано слово. Переставить его последнюю букву на место k-й.
При этом k-ю, (k + 1)-ю, ..., предпоследнюю буквы сдвинуть вправо на одну позицию.

```
    private void moveLastToK(StringBuilder sb, int k) {
        sb.insert(k - 1, sb.charAt(sb.length() -
1)).deleteCharAt(sb.length() - 1);
        System.out.println(sb);
    }
```

9.137. Дано слово из 12-ти букв. Переставить его буквы следующим способом:
первая, двенадцатая, вторая, одиннадцатая, ..., пятая, восьмая, шестая, седьмая.

```
    private void shuffleWord(StringBuilder sb) {
        for (int i = 0; i < sb.length() / 2; i++) {
            char tmp = sb.charAt(i);
            sb.deleteCharAt(i).
                    insert(i, sb.charAt(sb.length() - 1 - i)).
                    insert(sb.length() - 1 - i, tmp).
                    deleteCharAt(sb.length() - 1 - i);
        }
        System.out.println(sb);
    }
```

Строки. Обработка цифр в строке

9.138. Дан символ. Выяснить, является ли он цифрой.

```
private boolean isNumber(char inputChar){
        return Character.isDigit(inputChar);
    }
```

9.139. Дан текст. Напечатать все имеющиеся в нем цифры.

```
private void printAllDigits(String inputString) {
        for (int i = 0; i < inputString.length(); i++) {
            if (Character.isDigit(inputString.charAt(i))) {
                System.out.print(inputString.charAt(i));
            }
        }
    }
```

9.140. Дан текст. Определить количество цифр в нем.

```
 private int countDigits(String inputString){
        int result = 0;
        for (int i = 0; i < inputString.length(); i++) {
            if (Character.isDigit(inputString.charAt(i))) {
                result++;
            }
        }
        return result;
    }
```

9.141. Дан текст, в котором имеются цифры.
а) Найти их сумму.

```
private int sumOfDigits(String inputString){
        int sum = 0;
        for (int i = 0; i < inputString.length(); i++) {
            if (Character.isDigit(inputString.charAt(i))) {
                sum +=
Character.getNumericValue(inputString.charAt(i));
            }
        }
        return sum;
    }
```

б) Найти максимальную цифру.

```
 private int maxDigit(String inputString){
        int max = 0;
        for (int i = 0; i < inputString.length(); i++) {
            if (Character.isDigit(inputString.charAt(i)) &&
Character.getNumericValue(inputString.charAt(i)) > max) {
                max =
Character.getNumericValue(inputString.charAt(i));
            }
        }
        return max;
    }
```

9.142. Дан текст, в начале которого имеются пробелы и в котором имеются цифры. Найти порядковый номер максимальной цифры, начиная счет с первого символа, не являющегося пробелом. Если максимальных цифр несколько, то должен быть найден номер первой из них.

```
private int positionOfMaxDigit(String inputString) {
        inputString = inputString.trim();
        int positionOfMax = 0;
        int max = 0;
        for (int i = 0; i < inputString.length(); i++) {
            if (Character.isDigit(inputString.charAt(i)) &&
Character.getNumericValue(inputString.charAt(i)) > max) {
                max =
Character.getNumericValue(inputString.charAt(i));
                positionOfMax = i;
            }
        }
        return positionOfMax;
    }
```

9.143. Дан текст. Определить, является ли он правильной десятичной записью целого числа.

```
  private boolean isInteger(String inputString){
        for (int i = 0; i < inputString.length(); i++) {
            if(!Character.isDigit(inputString.charAt(i))){
                return false;
            }
        }
        return true;
    }
```

9.144. Дан текст, представляющий собой десятичную запись целого числа. Вычислить сумму цифр этого числа.

```
private int sumOfDigits(String inputString) {
        int sum = 0;
        for (int i = 0; i < inputString.length(); i++) {
            sum +=
Character.getNumericValue(inputString.charAt(i));
        }
        return sum;
    }
```

9.145. Дан текст, имеющий вид: "d1+ d2+ …+ dn " , где di — цифры (n > 1). Вычислить записанную в тексте сумму.

```
 private int sum(String inputString){
        int sum = 0;
        for (int i = 0; i < inputString.length(); i+=2) {
            sum +=
Character.getNumericValue(inputString.charAt(i));
        }
        return sum;
    }
```

9.146. Дан текст, имеющий вид: "d1 -d2+ d3- ...", где di — цифры (n > 1). Вычислить записанную в тексте алгебраическую сумму.

```
 private int sum(String inputString) {
        int sum = 0;
        for (int i = 0; i < inputString.length(); i += 2) {
            sum += Math.pow(-1, i / 2) *
Character.getNumericValue(inputString.charAt(i));
        }
        return sum;
    }
```

9.147. Дан текст, имеющий вид: "d1± d2± ... dn " , где di — цифры (n > 1). Вычислить записанную в тексте алгебраическую сумму.

```
 private int sum(String inputString) {
      int sum = Character.getNumericValue(inputString.charAt(0));
      char sign;
      for (int i = 1; i < inputString.length(); i += 2) {
         sign = inputString.charAt(i);
```

```
        if (sign == '+') {
            sum += Character.getNumericValue(inputString.charAt(i + 1));
        } else {
            sum -= Character.getNumericValue(inputString.charAt(i + 1));
        }
    }
    return sum;
}
```

9.148. Дан текст. Найти наибольшее количество идущих подряд цифр.

```
private int numberOfSameConsecutiveChars(String string) {
        if(string.length() == 0){
            return 0;
        }
        else {
            int currentLength = 1;
            int maxLength = 1;
            for (int i = 1; i < string.length(); i++) {
                if (string.charAt(i) == string.charAt(i - 1)) {
                    currentLength++;
                }
                if (currentLength > maxLength) {
                    maxLength = currentLength;
                }
                if (string.charAt(i) != string.charAt(i - 1)) {
                    currentLength = 1;
                }
            }
            return maxLength;
        }
    }
```

9.149. Дан текст, в котором имеется несколько идущих подряд цифр. Получить число, образованное этими цифрами.

```
private int getNumber(String inputString) {
      inputString = inputString.replaceAll("\\D+","");
      return Integer.parseInt(inputString);
    }
```

9.150. Дан текст. Найти сумму всех имеющихся в нем чисел.

```
 private int getSumOfAllNumbers(String inputString) {
        String[] stringNumbers = inputString.replaceAll("\\D+", "
").trim().split(" ");
        int sum = 0;
        for (int i = 0; i < stringNumbers.length; i++) {
            sum += Integer.parseInt(stringNumbers[i]);
        }
        return sum;
    }
```

9.151. Дан текст. Найти максимальное из имеющихся в нем чисел.

```
private int getMaxOfAllNumbers(String inputString) {
        String[] stringNumbers = inputString.replaceAll("\\D+", "
").trim().split(" ");
        int max = Integer.parseInt(stringNumbers[0]);
        for (int i = 1; i < stringNumbers.length; i++) {
            int currentNumber =
Integer.parseInt(stringNumbers[i]);
            if (currentNumber > max) {
                max = currentNumber;
            }
        }
        return max;
    }
```

Строки. Задачи повышенной сложности

Вложенные циклы. Организация вывода с использованием вложенных циклов

```
//    8.1. Напечатать числа в виде следующей таблицы:
//    а) 5 5 5 5 5 5
//       5 5 5 5 5 5
//       5 5 5 5 5 5
//       5 5 5 5 5 5
//    б) 1 2 ... 10
//       1 2 ... 10
//       1 2 ... 10
//       1 2 ... 10
//    в) 41 42 ... 50
//       51 52 ... 60
//       ... ... ... ...
//       71 72 ... 80
private void print1() {
    System.out.println("Задача 8.1");
    System.out.println("а)");
    for (int i = 0; i < 4; ++i) {
        for (int j = 0; j < 6; ++j) {
            System.out.print(5 + " ");
        }
        System.out.println();
    }
    System.out.println("б)");
    for (int i = 0; i < 4; ++i) {
        for (int j = 1; j < 11; ++j) {
            System.out.print(j + " ");
        }
        System.out.println();
    }
    System.out.println("в)");
    for (int i = 4; i < 8; ++i) {
        for (int j = 1; j < 11; ++j) {
            System.out.print(i * 10 + j + " ");
        }
        System.out.println();
    }
    System.out.println();
}

//    8.2. Напечатать числа в виде следующей таблицы:
//    а) 5
//       5 5
//       5 5 5
//       5 5 5 5
//       5 5 5 5 5
```

```
//    б) 1 1 1 1 1
//       1 1 1 1
//       1 1 1
//       1 1
//       1
private void print2() {
    System.out.println("Задача 8.2");
    System.out.println("а)");
    for (int i = 1; i < 6; ++i) {
        for (int j = 0; j < i; ++j) {
            System.out.print(5 + " ");
        }
        System.out.println();
    }
    System.out.println("б)");
    for (int i = 5; i > 0; --i) {
        for (int j = i; j > 0; --j) {
            System.out.print(1 + " ");
        }
        System.out.println();
    }
    System.out.println();
}

//    8.3. Напечатать числа в виде следующей таблицы:
//    а) 1
//       2 2
//       3 3 3
//       4 4 4 4
//       5 5 5 5 5
//    б) 5 5 5 5 5
//       6 6 6 6
//       7 7 7
//       8 8
//       9
//    в) 10
//       20 20
//       30 30 30
//       40 40 40 40
//       50 50 50 50 50
//    г) 5 5 5 5 5
//       10 10 10 10
//       15 15 15
//       20 20
//       25
private void print3() {
    System.out.println("Задача 8.3");
    System.out.println("а)");
    for (int i = 1; i < 6; ++i) {
        for (int j = 0; j < i; ++j) {
            System.out.print(i + " ");
```

```
        }
        System.out.println();
    }
    System.out.println("б)");
    for (int i = 5; i < 10; ++i) {
        for (int j = 0, n = 10 - i; j < n; ++j) {
            System.out.print(i + " ");
        }
        System.out.println();
    }
    System.out.println("в)");
    for (int i = 1; i < 6; ++i) {
        for (int j = 0; j < i; ++j) {
            System.out.print(i * 10 + " ");
        }
        System.out.println();
    }
    System.out.println("г)");
    for (int i = 5; i < 30; i += 5) {
        for (int j = 0, n = 30 - i; j < n; j += 5) {
            System.out.print(i + " ");
        }
        System.out.println();
    }
    System.out.println();
}

//    8.4. Напечатать числа в виде следующей таблицы:
//    а) 0
//       1 0
//       2 1 0
//       3 2 3 0
//       4 3 4 1 0
//    б) 6 5 4 3 2
//       7 4 3 2
//       8 3 2
//       9 2
//       2
//    в) 30
//       29 30
//       28 29 30
//       27 28 29 30
//       26 27 28 29 30
//    г) 20 21 22 23 24
//       19 20 21 22
//       18 19 20
//       17 18
//       16
private void print4() {
    System.out.println("Задача 8.4");
    System.out.println("а)");
```

```
    for (int i = 0; i < 5; ++i) {
        for (int j = i; j > -1; --j) {
            if ((j == 0) || !(((i & 1) == 0 && (j & 1) == 0) || ((i &
1) != 0 && (j & 1) != 0))) {
                System.out.print(j + " ");
            } else {
                System.out.print(i + " ");
            }
        }
        System.out.println();
    }
    System.out.println("б)");
    for (int i = 0, j = 6; i < 10; i += 2, ++j) {
        for (int k = j - i < 3 ? 2 : j; k > 1; --k) {
            if (k == j - 1) {
                k -= i;
            }
            System.out.print(k + " ");
        }
        System.out.println();
    }
    System.out.println("в)");
    for (int i = 0; i < 5; ++i) {
        for (int j = 30 - i; j < 31; ++j) {
            System.out.print(j + " ");
        }
        System.out.println();
    }
    System.out.println("г)");
    for (int i = 0; i < 5; ++i) {
        for (int j = 20 - i; j < 25 - 2 * i; ++j) {
            System.out.print(j + " ");
        }
        System.out.println();
    }
    System.out.println();
}

//    8.5. Напечатать полную таблицу сложения в виде:
//       1 + 1 = 2 2 + 1 = 3 ... 9 + 1 = 9
//       1 + 2 = 3 2 + 2 = 4 ... 9 + 2 = 11
//       ... ... ... ...
//       1 + 9 = 10 2 + 9 = 11 ... 9 + 9 = 18
private void print5() {
    System.out.println("Задача 8.5");
    for (int i = 1; i < 10; ++i) {
        for (int j = 1; j < 10; ++j) {
            System.out.printf("%d + %d = %2d  ", j, i, j + i);
        }
        System.out.println();
    }
```

```
}

//    8.6. Напечатать полную таблицу сложения в виде:
//       1 + 1 = 2 1 + 2 = 3 ... 1 + 9 = 10
//       2 + 1 = 3 2 + 2 = 4 ... 2 + 9 = 11
//       ... ... ... ...
//       9 + 1 = 10 9 + 2 = 11 ... 9 + 9 = 18
private void print6() {
   System.out.println("Задача 8.6");
   for (int i = 1; i < 10; ++i) {
       for (int j = 1; j < 10; ++j) {
           System.out.printf("%d + %d = %2d  ", i, j, i + j);
       }
       System.out.println();
   }
}

//    8.7. Напечатать полную таблицу умножения в виде:
//       1 x 1 = 1 1 x 2 = 2 ... 1 x 9 = 9
//       2 x 1 = 2 2 x 2 = 4 ... 2 x 9 = 18
//       ... ... ... ...
//       9 x 1 = 9 9 x 2 = 18 ... 9 x 9 = 81
private void print7() {
   System.out.println("Задача 8.7");
   for (int i = 1; i < 10; ++i) {
       for (int j = 1; j < 10; ++j) {
           System.out.printf("%d * %d = %2d  ", i, j, i * j);
       }
       System.out.println();
   }
}
//    8.8. Напечатать полную таблицу умножения в виде:
//       1 x 1 = 1 2 x 1 = 2 ... 9 x 1 = 9
//       1 x 2 = 2 2 x 2 = 4 ... 9 x 2 = 18
//       ... ... ... ...
//       1 x 9 = 9
private void print8() {
   System.out.println("Задача 8.8");
   for (int i = 1; i < 10; ++i) {
       for (int j = 1; j < 10; ++j) {
           System.out.printf("%d * %d = %2d  ", j, i, j * i);
       }
       System.out.println();
   }
}

// Задачи 8.9 и 8.10 не решала. Мне кажется, что таких на экзамене
не будет.
```

Вложенные циклы. Обработка данных во время ввода с использованием вложенных циклов

Вложенные циклы. Вложенные циклы и целые числа

```
//    8.25. Найти количество делителей каждого из целых чисел от 120 до 140.
private void task25() {
    for (int i = 24; i < 141; ++i) {
        System.out.printf("Number - %3d, count of divisors = %2d%n", i, countDivisors(i));
    }
}

private int countDivisors(int number) {
    if (number == 1) {
        return 1;
    }
    Map<Integer, Integer> primes = new HashMap<>();
    primeFactorization(number, primes);
    int divisorCount = 1;
    for (Integer value : primes.values()) {
        divisorCount *= (value + 1);
    }
    return divisorCount;
}

// рекурсивная функция поиска простых множителей
private void primeFactorization(int number, Map<Integer, Integer> result) {
    primeFactorization(number, result, 2);
}

// рекурсивная функция поиска простых множителей
private void primeFactorization(int number, Map<Integer, Integer> result, int factor) {
    if (factor > Math.sqrt(number)) {
        result.merge(number, 1, (exCount, newCount) -> exCount + newCount);
        return;
    }
    if (number % factor == 0) {
        result.merge(factor, 1, (exCount, newCount) -> exCount + newCount);
        primeFactorization(number / factor, result, factor);
```

```
    } else {
        primeFactorization(number, result, ++factor);
    }
}

//    8.26. Составить программу для графического изображения
делимости чисел от 1 до
//    n (значение n вводится с клавиатуры). В каждой строке надо
напечатать оче-
//    редное число и столько символов "+", сколько делителей у этого
числа. На-
//    пример, если n 4, то на экране должно быть напечатано:
//             1+
//             2++
//             3++
//             4+++
private void task26() {
    try (BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in))) {
        String line = reader.readLine();
        if (line.matches("\\d+")) {
            int n = Integer.parseInt(line);
            if (n > 1) {
                for (int i = 1; i <= n; ++i) {
                    int divisorCount = countDivisors(i);
                    System.out.printf("Number - %d", i);
                    for (int j = 0; j < divisorCount; ++j) {
                        System.out.print("+");
                    }
                    System.out.println();
                }
            }
        }
    } catch (IOException ex) {
        ex.printStackTrace();
    }
}

//    8.27. Найти все целые числа из промежутка от 1 до 300, у
которых ровно пять де-
//    лителей.
private void task27() {
    for (int i = 1; i < 300; ++i) {
        int divisorCount = countDivisors(i);
        if (divisorCount == 5) {
```

```
            System.out.printf("Number - %3d, count of divisors =
%2d%n", i, divisorCount);
        }
    }
}

//    8.28. Найти все целые числа из промежутка от 200 до 500, у
которых ровно шесть
//    делителей.
private void task28() {
    for (int i = 200; i < 500; ++i) {
        int divisorCount = countDivisors(i);
        if (divisorCount == 6) {
            System.out.printf("Number - %3d, count of divisors =
%2d%n", i, divisorCount);
        }
    }
}

//    8.29. Найти все целые числа из промежутка от a до b, у которых
количество дели-
//    телей равно k.
private void task29(int a, int b, int k) {
    if (b > a) {
        for (; a < b; ++a) {
            int divisorCount = countDivisors(a);
            if (divisorCount == k) {
                System.out.printf("Number - %3d, count of divisors =
%2d%n", a, divisorCount);
            }
        }
    }
}

//    8.30. Найти натуральное число из интервала от a до b, у
которого количество дели-
//    телей максимально. Если таких чисел несколько, то должно быть
найдено:
//    а) максимальное из них;
//    б) минимальное из них.
private void task30(int a, int b) {
    int max = -1;
    int min = Integer.MAX_VALUE;
    int maxDivisorCount = 0;
    if (b > a) {
        for (; a < b; ++a) {
```

```
            int divisorCount = countDivisors(a);
            if (maxDivisorCount <= divisorCount) {
                if (maxDivisorCount == divisorCount) {
                    if (max < a) {
                        max = a;
                    }
                    if (min > a) {
                        min = a;
                    }
                } else {
                    maxDivisorCount = divisorCount;
                    max = a;
                    min = a;
                }
            }
        }
        System.out.printf("Max number - %3d, max count of divisors =
%2d%n", max, maxDivisorCount);
        System.out.printf("Min number - %3d, max count of divisors =
%2d%n", min, maxDivisorCount);
    }
}


//    8.31. Найти все трехзначные простые числа (простым называется
натуральное чис-
//    ло, большее 1, не имеющее других делителей, кроме единицы и
самого себя).
private void task31() {
    boolean[] primes = getSieve(1000);
    for (int i = 100; i < 1000; ++i) {
        if (primes[i]) {
            System.out.println(i);
        }
    }
}

// решето Эратосфена
private boolean[] getSieve(int n) {
    boolean[] primes = new boolean[n + 2];
    Arrays.fill(primes, true);
    primes[0] = false;
    primes[1] = false;
    for (int i = 2; i < primes.length; ++i) {
        if (primes[i]) {
            for (int j = 2; i * j < primes.length; ++j) {
```

```
                primes[i * j] = false;
            }
        }
    }
    return primes;
}

//    8.32. Найти 100 первых простых чисел.
private void task32() {
    int count = 0;
    boolean[] primes = getSieve(1000);
    for (int i = 1; i < 1000; ++i) {
        if (count < 100 && primes[i]) {
            System.out.println(i);
            ++count;
        }
    }
}

//    8.33. Найти сумму делителей каждого из целых чисел от 50 до
70.
private void task33() {
    for (int i = 50; i < 70; ++i) {
        System.out.printf("Number %2d, sum of divisors = %3d%n", i,
sumDivisors(i) + i);
    }
}

// функция подсчета суммы делителей числа, не включающая само число
// такая функция удобна для определения совершенных чисел
private int sumDivisors(int n) {
    int sum = 0;
    for (int i = 1; i < n; ++i) {
        if (n % i == 0) {
            sum += i;
        }
    }
    return sum;
}

//    8.34. Найти все целые числа из промежутка от 100 до 300, у
которых сумма делите-
//    лей равна 50.
private void task34() {
    for (int i = 100; i < 300; ++i) {
        int divisorSum = sumDivisors(i) + i;
```

```
        if (divisorSum == 50) {
            System.out.printf("Number %2d, sum of divisors = %3d%n",
i, divisorSum);
        }
    }
}

//    8.35. Найти все целые числа из промежутка от 300 до 600, у
которых сумма делите-
//    лей кратна 10.
private void task35() {
    for (int i = 300; i < 600; ++i) {
        int divisorSum = sumDivisors(i) + i;
        if (divisorSum % 50 == 0) {
            System.out.printf("Number %2d, sum of divisors = %3d%n",
i, divisorSum);
        }
    }
}

//    8.36. Натуральное число называется совершенным, если оно равно
сумме своих де-
//    лителей, включая 1 и, естественно, исключая это самое число.
Например, со-
//    вершенным является число 6 ( 6 1 2 3 ). Найти все совершенные
числа,
//    меньшие 100 000.
private void task36() {
    for (int i = 1; i < 100000; ++i) {
        int divisorSum = sumDivisors(i);
        if (divisorSum == i) {
            System.out.printf("Number %2d is a perfect number%n",
i);
        }
    }
}

//    8.37. Найти натуральное число из интервала от a до b с
максимальной суммой де-
//    лителей.
private void task37(int a, int b) {
    int maxSum = 0;
    int max = 0;
    for (int i = a; i < b; ++i) {
        int divisorSum = sumDivisors(i);
        if (maxSum < divisorSum) {
```

```
            max = i;
            maxSum = divisorSum;
        }
    }
    System.out.printf("Number %2d is a number with max sum of
divisors - %d [%d, %d)%n", max, maxSum, a, b);
}

//    8.38. Два натуральных числа называются дружественными, если
каждое из них
//    равно сумме всех делителей другого (само другое число в
качестве делителя
//    не рассматривается). Найти все пары натуральных дружественных
чисел,
//    меньших 50 000.
private void task38() {
    for (int i = 1; i < 50000; ++i) {
        int divisorSum = sumDivisors(i);
        if (divisorSum > i && sumDivisors(divisorSum) == i) {
            System.out.printf("Number %d and number %d are
amicable%n", i, divisorSum);
        }
    }
}

//    8.54.*Дано натуральное число n. Получить все простые делители
этого числа.
private void task54(int number) {
    Map<Integer, Integer> primes = new HashMap<>();
    primeFactorization(number, primes);
    System.out.printf("Prime divisors of %d are ", number);
    for (Map.Entry<Integer, Integer> pair : primes.entrySet()) {
        for (int i = 0; i < pair.getValue(); ++i) {
            System.out.printf("%d ", pair.getKey());
        }
    }
    System.out.println();
}

//    8.55.*Дано натуральное число n. Получить все натуральные
числа, меньшие n
//    и взаимно простые с ним (два натуральных числа называются
взаимно про-
//    тыми, если их наибольший общий делитель равен 1).
private void task55(int number) {
    for (int i = 1; i < number; ++i) {
```

```
        if (getGCD(number, i) == 1) {
            System.out.printf("Coprime numbers: %d and %d%n",
number, i);
        }
    }
}

private int getGCD(int a, int b) {
    return b == 0 ? Math.abs(a) : getGCD(b, a % b);
}

//    8.56.*Даны целые числа n и m. Получить все натуральные числа,
меньшие n и вза-
//    имно простые с m.
private void task56(int n, int m) {
    for (int i = 1; i < n; ++i) {
        if (getGCD(m, i) == 1) {
            System.out.printf("Number %d < %d and coprime with
%d%n", i, n, m);
        }
    }
}

//    8.57.*Даны целые числа p и q. Получить все делители числа q,
взаимно простые с p.
private void task57(int p, int q) {
    for (int i = 1; i < q; ++i) {
        if (q % i == 0) {
            if (getGCD(p, i) == 1) {
                System.out.printf("Number %d is a divisor of %d and
coprime with %d%n", i, q, p);
            }
        }
    }
}

//    Задачи 8.39 - 8.45, 8.58, 8.59 не решала. Мало шансов, что они
будут на экзамене
```

Двумерные массивы. Заполнение и вывод массива нестандартными методами

Двумерные массивы. Расчетные задачи

Двумерные массивы. Нахождение максимума и минимума

Двумерные массивы. Проверка условия после выполнения расчетов

Двумерные массивы. Обработка массива с использованием операторов цикла с условием

Двумерные массивы. Работа с квадратными массивами

Двумерные массивы. Изменение исходного массива

Двумерные массивы. Работа с несколькими массивами

Двумерные массивы. Двумерные символьные массивы

```
/**
* 12.278.
Найти количество строк массива, в которых имеется ровно три буквы о.
*/
public class TripleOWordsCount {
  public static void main(String[] args) {
     int tripleOWord = 0;
     String[] array = {"doloto", "zoloto", "lot", "voda", "ooo", "oooo"};

     for (int i = 0; i < array.length ; i++) {
        int isO = 0;
        for (int j = 0; j <array[i].length() ; j++) {
           if(array[i].charAt(j) == 'o'){
              isO++;
           }
        }
        if (isO == 3){
           tripleOWord++;
        }
     }

     System.out.println("Число слов содержащих 3 \'o\' составляет " + tripleOWord);
  }
}
```

```java
/**
* 12.278. Найти количество строк массива, в которых имеется ровно три буквы
о.
*/
public class TripleOWordsCount {
  public static void main(String[] args) {
    int tripleOWord = 0;
    String[] array = {"doloto", "zoloto", "lot", "voda", "ooo", "oooo"};

    for (int i = 0; i < array.length ; i++) {
      int isO = 0;
      for (int j = 0; j <array[i].length() ; j++) {
        if(array[i].charAt(j) == 'o'){
          isO++;
        }
      }
      if (isO == 3){
        tripleOWord++;
      }
    }

    System.out.println("Число слов содержащих 3 \'o\' составляет " + tripleOWord);
  }
}
```

Функции и процедуры. Рекурсия

10.43. Написать рекурсивную функцию:
а) вычисления суммы цифр натурального числа;
б) вычисления количества цифр натурального числа.

```java
//Вычисление суммы цифр
public final class DigitsSum {
    public static void main(String[] args) {
        System.out.println(getDigitsSum(671));
    }

    private static int getDigitsSum(final int number) {
        if (number < 0) {
            throw new RuntimeException("The number must be natural");
        }

        if (number < 10) {
            return number;
        }

        final int quotient = number / 10;
        final int modulo = number % 10 ;
```

```
        return getDigitsSum(quotient) + modulo;
    }

}

// Вычисление количества цифр натурального числа
public final class FindDigitsNumber {
    public static void main(String[] args) {
        System.out.println("The number of digits is equal: " + getNumberOfDigits(123456789));
    }

    private static int getNumberOfDigits(final int number) {
        if (number < 0) {
            throw new RuntimeException("The number must be natural");
        }

        if (number < 10) {
            return 1;
        }

        final int quotient = number / 10;
        final int modulo = number % 10;

        return  getNumberOfDigits(modulo) + getNumberOfDigits(quotient);
    }
}
```

10.50. Написать рекурсивную функцию для вычисления значения так называемой функции Аккермана для неотрицательных чисел n и m. Функция Аккермана определяется следующим образом:

$$A\ n, m\ = \begin{cases} m+1, & \text{если } n=0, \\ A\ n-1, 1\ , & \text{если } n\neq 0,\ m=0, \\ A\ n-1, A\ n, m-1\ , & \text{если } n>0,\ m>0. \end{cases}$$

Функцию Аккермана называют дважды рекурсивной, т. к. сама функция и один из ее аргументов определены через самих себя.

Найти значение функции Аккермана для $n=1,\ m=3$.

Написать рекурсивную функцию для вычисления значения так называемой функции Аккермана для неотрицательных чисел n и m. Функция Аккермана определяется следующим образом:

```
// Функция Аккермана

public final class AkkermanFunction {
   public static void main(String[] args) {
      System.out.println(a(3, 1));
   }

   /* Функция Аккермана */
   private static int a(int m, int n) {
      if (m < 0 || n < 0)
         throw new RuntimeException("m & n must be a positive numbers");

      if (m == 0)
         return n + 1;

      if (m > 0 && n == 0)
         return a(m - 1, 1);

      return a(m - 1, a(m, n - 1 ));
   }
}
```

10.47. Написать рекурсивную функцию для вычисления k-го члена последовательности Фибоначчи. Последовательность Фибоначчи $f_1, f_2, \ldots$ образуется по закону: $f_1 = 1;\ f_2 = 1;\ f_i = f_{i-1} + f_{i-2}\ (i = 3, 4, \ldots)$.

Написать рекурсивную функцию для вычисления k-го члена последовательности Фибоначчи. Последовательность Фибоначчи

```
public final class FibonacciTest {
   public static void main(String[] args) {
      System.out.println("Fibonacci value equals" + fibonacciValue(12));
   }

   private static int fibonacciValue(final int number) {
      if (number < 0) {
         throw new RuntimeException("The value must be positive");
      }

      if (number == 0) {
         return 1;
      }
```

```
        if (number == 1) {
            return 1;
        }

        return fibonacciValue(number - 1) + fibonacciValue(number - 2);
    }
}
```

10.41. Написать рекурсивную функцию для вычисления факториала натурального числа n.

```
public final class Factorial {
    public static void main(String[] args) {
        System.out.println("The factorial value equals: " + getFactorial(11));
    }

    private static int getFactorial(final int number) {
        if (number < 0) {
            throw new RuntimeException("The number value must be positive");
        }

        if (number == 0) {
            return 1;
        }

        if (number == 1) {
            return 1;
        }

        return getFactorial(number - 1) * number;

    }
}
```

10.42. В некоторых языках программирования (например, в Паскале) не предусмотрена операция возведения в степень. Написать рекурсивную функцию для расчета степени n вещественного числа a (n — натуральное число).

```
public final class Power {
    static final int number = 2;
    static final double base = 6;

    public static void main(String[] args) {
        System.out.println("The value of base: " + base + " and power: " + number + " is equal:
" + getPower(number,base));
    }
```

```
    private static double getPower(final int number, final double base) {
        if (number < 0) {
            throw new RuntimeException("The number value must be positive");
        }

        if (number == 0) {
            return 1;
        }

        return (getPower(number - 1, base) * base);
    }
}
```

Решенные задачи от Жанны