**Лабораторная № 31**

Разработать **программы** (использовать **String, , StringBuilder и StringBuffer и StringTokenizer)**

***Выполняем одну задачу согласно списку. Используем свои классы***

1. Строка состоит из слов. За один просмотр символов строки найти все самые длинные слова, символы в которых идут в строгом возрастании кодов, и занести их в новую строку. Слова в исходной строке разделяются некоторым множеством разделителей. Слова в новой строке должны разделяться ровно одним пробелом.
2. За один просмотр символов строки найти все слова, содержащие наибольшее количество различных символов, и занести их в новую строку. Слова в исходной строке разделяются некоторым множеством разделителей. Слова в новой строке должны разделяться ровно одним пробелом.
3. Строка состоит из слов. За один просмотр символов строки найти все слова, начинающиеся с гласных букв, и занести их в новую строку, заменяя первую букву каждого слова на прописную. Слова в исходной строке разделяются некоторым множеством разделителей. Слова в новой строке должны разделяться ровно одним пробелом.
4. Словом в строке считается последовательность букв латинского алфавита, остальные символы являются разделителями между словами. За один просмотр символов строки найти все слова с максимальной долей гласных букв (т.е. прописных и строчных символов ’a’, ’e’, ’i’, ’o’, ’u’, ’y’) и занести их в новую строку. Слова в исходной строке разделяются некоторым множеством разделителей. Слова в новой строке должны разделяться ровно одним пробелом.
5. Задана строка из слов. Слова в исходной строке разделяются некоторым множеством разделителей. Среди слов, состоящих только из цифр, найти слово, содержащее максимальное число нулей. Если таких слов больше одного, найти предпоследнее из них. Слова в новой строке должны разделяться ровно одним пробелом.
6. Задана строка из слов. Слова в исходной строке разделяются некоторым множеством разделителей. Поместить в начало строки слова, содержащие только цифры, а затем – все остальные слова. Порядок слов внутри заданных групп не должен изменяться. Слова в новой строке должны разделяться ровно одним пробелом.
7. Словом в строке считается последовательность букв латинского алфавита, остальные символы рассматриваются как разделители между словами. Слова в каждой из двух исходных строк упорядочены по алфавиту. Необходимо сформировать новую строку из всех слов исходных строк. Слова в новой строке должны быть также упорядочены по алфавиту. Прописные и строчные буквы в словах не различать. Слова в исходной строке разделяются некоторым множеством разделителей. Слова в новой строке должны разделяться ровно одним пробелом.
8. Из заданной строки получить новую строку, выполняя замены по следующему правилу:

символы латинского алфавита следует оставить без изменений;

символ «пробел» следует заменить символом ‘+’.

Слова в исходной строке разделяются некоторым множеством разделителей. Слова в новой строке должны разделяться ровно одним пробелом.

1. Дана строка, слова в которой состоят из букв латинского алфавита и десятичных цифр. Остальные символы считаются разделителями между словами. Получить новую строку, выполняя в заданной строке замены по следующему правилу. Все слова, состоящие только из букв, записываются строчными буквами. Слова в исходной строке разделяются некоторым множеством разделителей. Слова в новой строке должны разделяться ровно одним пробелом.
2. Дана строка, слова в которой состоят из букв латинского алфавита и десятичных цифр. Остальные символы считаются разделителями между словами. Получить новую строку, выполняя в заданной строке замены по следующему правилу. Все слова, имеющие длину более 4 символов и состоящие только из букв, заменяются словами, записанными в обратном порядке. Слова в исходной строке разделяются некоторым множеством разделителей. Слова в новой строке должны разделяться ровно одним пробелом.
3. Задана строка, слова в которой состоят из букв латинского алфавита и десятичных цифр. Остальные символы – разделители между словами. Получить новую строку, выполняя в заданной строке замены по следующему правилу. Все слова, имеющие длину менее 5 символов и состоящие только из цифр, заменяются словами, получаемыми из исходных путём дописывания ведущих нулей до получения длины слова, равной 5. Слова в исходной строке разделяются некоторым множеством разделителей. Слова в новой строке должны разделяться ровно одним пробелом.
4. Дана строка, слова в которой состоят из букв латинского алфавита и десятичных цифр. Остальные символы – разделители между словами. Получить новую строку, выполняя в заданной строке замены по следующему правилу. Во всех словах, состоящих только из букв, прописные буквы заменяются на строчные. Слова в исходной строке разделяются некоторым множеством разделителей. Слова в новой строке должны разделяться ровно одним пробелом.
5. Задана строка из слов. Слова в исходной строке разделяются некоторым множеством разделителей. 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 между словами должно остаться таким же, как и в исходной строке. Слова в новой строке должны разделяться ровно одним пробелом.
6. Задана строка из слов. Слова в исходной строке разделяются некоторым множеством разделителей. Переставить в ней слова, состоящие только из букв латиницы, так, чтобы они были упорядочены по алфавиту. При этом другие слова должны оставаться на месте. Количество пробелов между словами должно остаться таким же, как и в исходной строке. Слова в новой строке должны разделяться ровно одним пробелом.
7. Словом в строке считается последовательность букв латинского алфавита, остальные символы – разделители между словами. Переставить в строке только слова – палиндромы, чтобы они были упорядочены по алфавиту. При этом другие слова и разделители должны оставаться на месте. Слова в исходной строке разделяются некоторым множеством разделителей. Слова в новой строке должны разделяться ровно одним пробелом.
8. Задана строка из слов. Слова в исходной строке разделяются некоторым множеством разделителей. Заменить в ней заданное слово на другое. Замену выполнить столько раз, сколько заданных слов в строке. Слова в новой строке должны разделяться ровно одним пробелом.

**Необходимая информация:**

1. **Ввод строки с клавиатуры.**

/\* подключение необходимой библиотеки \*/

import java.io.\*;

InputStreamReader isr = new InputStreamReader(System.in);

BufferedReader br = new BufferedReader(isr);

try

{

String line = br.readLine();

}

catch (IOException e)

{

System.out.println("Ошибка чтения с клавиатуры");

}

1. **Класс String**

Конструкторы:

**String()**, **String(String str)**, **String(byte[] asciichar)**, **String(char[] c)**, **String(StringBuffer sbuf)** и др.

Методы:

**concat(String s)** или “+” – слияние строк (создает новую строку);

"Hello".concat(" World") -> "Hello World"

**equals(Object ob)** и **equalsIgnoreCase(String s**) – сравнение строк с учетом и без учета регистра соответственно (сравнивает и возвращает Boolean);

**compareTo(String s)** и **compareToIgnoreCase (String s)** – лексикографическое сравнение строк с учетом и без учета регистра;

**contentEquals(StringBuffer ob)** – сравнение строки и содержимого объекта типа **StringBuffer**;

**substring(int n, int m)** – извлечение из строки подстроки длины **m-n**, начиная с позиции **n**;

"Hello World".substring(6) -> "World"

**substring(int n)** – извлечение из строки подстроки, начиная с позиции **n**;

"Hello World".substring(3,8) -> "lo Wo"

**length()** – определение длины строки;

**valueOf(**значение**)** – преобразование переменной базового типа к строке;

**toUpperCase()/toLowerCase()** – преобразование всех символов вызывающей строки в верхний/нижний регистр;

**replace(char с1, char с2)** – замена в строке всех вхождений первого символа вторым символом;

"Hello".replace('l' , 'w') -> "Hewwo"

**trim()** – удаление всех пробелов в начале и конце строки;

**charAt(int position)** – возвращение **символа** из указанной позиции (нумерация с нуля);

**getBytes(**параметры**)**, **getChars(**параметры**)** – извлечение **символов** строки в виде массива байт или символов;

**indexOf** – возвращает индекс символа, который необходимо найти, если символа нет, то -1;

**lastIndexOf** - возвращает индекс символа, который необходимо найти, если символа нет, то -1;

**Внимание!!!** Во всех случаях вызова методов, изменяющих строку, создается новый объект типа **String**.

**Внимание!!!** Сохранить изменения в объекте класса **String** можно только с применением оператора присваивания, т.е. установкой ссылки на новый объект.

1. **Класс StringBuffer**

Класс **StringBuffer** является “близнецом” класса **String**, но, в отличие от последнего, содержимое и размеры объектов класса **StringBuffer** можно изменять. Объекты классов **StringBuffer** и **String** можно преобразовывать друг в друга. Конструктор класса **StringBuffer** может принимать в качестве параметра объект **String** и целочисленный (неотрицательный) размер буфера. Объекты этого класса можно преобразовать в объект класса **String** методом **toString()** или с помощью конструктора класса **String**.

Методы класса:

**setLength(int n)** – установка размера содержимого буфера.

**capacity()** – определение размера буфера объекта;

**append(**параметры**)** – добавление символов, значений базовых типов, массивов и строк;

**insert(**параметры**)** – вставка символа, объекта или строки в указанную позицию;

**deleteCharAt(int index)** – удаление символа;

**delete(int start, int end)** – удаление подстроки;

**reverse()** – обращение содержимого объекта.

В классе присутствуют также методы, аналогичные методам класса **String**, такие как **replace()**, **substring()**, **charAt()**, **length()**, **getChars()**, **indexOf()** и др.

1. **Класс StringTokenizer**

Класс **StringTokenizer** из пакета **java.util**.

Первый конструктор **StringTokenizer (String str)** создает объект, готовый разбить строку *str* на слова, разделенные пробелами, символами табуляций  '*\t*', перевода строки '*\n*' и возврата каретки '*\r*' . Разделители не включаются в число слов.

Второй конструктор *StringTokenizer* (*String* s*tr*, *String* *delimeters*) задает разделители вторым параметром delimeters , например:

StringTokenizer("Казнить,нельзя:пробелов-нет", " \t\n\r,:-");

Здесь первый разделитель — пробел. Потом идут символ табуляции, символ перевода строки, символ возврата каретки, запятая, двоеточие, дефис. Порядок расположения разделителей в строке *delimeters* не имеет значения. Разделители не включаются в число слов.

В разборе строки на слова активно участвуют два метода:

метод *nextToken()* возвращает в виде строки следующее слово;

логический метод *hasMoreTokens()* возвращает *true* , если в строке еще есть слова, и *false* , если слов больше нет.

Третий метод *countTokens()* возвращает число оставшихся слов.

Четвертый метод *nextToken(String newDelimeters)* позволяет "на ходу" менять разделители. Следующее слово будет выделено по новым разделителям *newDelimeters*; новые разделители действуют далее вместо старых разделителей, определенных в конструкторе или предыдущем методе *nextToken()* .

String s = "Строка, которую мы хотим разобрать на слова";

StringTokenizer st = **new** StringTokenizer(s, " \t\n\r,.");

**while**(st.hasMoreTokens())

{

// Получаем слово и что-нибудь делаем с ним, например, просто выводим на экран

System.out.println(st.nextToken()) ;

}