**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ЛАБОРАТОРНАЯ РАБОТА №3

по дисциплине: **«Визуальные средства разработки программных**

**приложений»**

на тему: **Работа со строками**

Выполнил: студент гр. ИТП-31

Дашкевич Д.А.

Принял: ассистент

Белявский Е. В.

Гомель 2019

**Цель**: изучить работу со строками в языке программирования *Java*.

**Ход работы**

**Вариант 2**

# **Задание:** в лабораторной работе необходимо выполнить четыре задания:

# 1. К слою доступа к данным, разработанном в лабораторной работе № 2 добавить: Конфигурацию (логин/пароль для БД, строку подключения и т.д.) и запросы SQL переместив в файлы properties. Для их чтения использовать класс ConfigurationManager построенный с использованием шаблона Singleton.

# 2. Добавить вывод сообщений в лог-файл с использованием логгера log4j. Во всех блоках try-catch заменить System.out.println на вывод в лог файл.

3. Написать программу для выполнения следующих действий с регулярными выражениями (по вариантам).

Написать регулярное выражение, определяющее является ли данная строка GUID с или без скобок. Где GUID это строчка, состоящая из 8, 4, 4, 4, 12 шестнадцатеричных цифр, разделенных тире. Пример правильных выражений: e02fd0e4-00fd-090A-ca30-0d00a0038ba0.–пример неправильных выражений: e02fd0e400fd090Aca300d00a0038ba0.

4. Создать программу обработки текста учебника по программированию с использованием классов: Символ, Слово, Предложение, Знак препинания и др. Во всех задачах с формированием текста заменять табуляции и последовательности пробелов одним пробелом. Для выделения нужных элементов из текста использовать регулярные выражения.

Рассортировать слова текста по возрастанию доли гласных букв (отношение количества гласных к общему количеству букв в слове).

**Результат работы программы:**

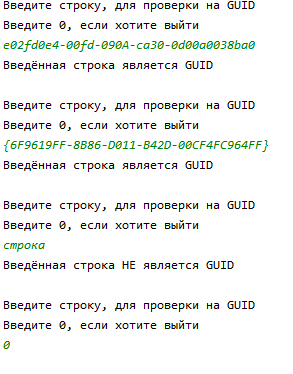
****

Рисунок 1 – Вывод в консоль работы программы, проверяющей строку на GUID

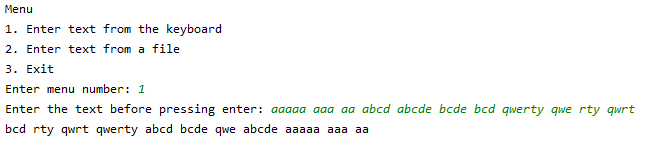


Рисунок 2 – Результат ввода слов и вывода их в отсортированном порядке по возрастанию по доле гласных букв

**Вывод:** в результате выполнения лабораторной работы была изучена работа со строками в языке программирования *Java,* были изучены регулярные выражения и их применение, написаны программы, использующие регулярные выражения для работы со строками.

**Приложение А**

**Исходный код программы**

**Main.java:**

**public class** Main {  
 **public static void** main(String[] args) {  
 Scanner in = **new** Scanner(System.***in***);  
 **boolean** exit = **false**;  
 **while** (!exit) {  
 System.***out***.println(**"Введите строку, для проверки на GUID"**);  
 System.***out***.println(**"Введите 0, если хотите выйти"**);  
 String s = in.nextLine();  
 **if** (s.equals(**"0"**)) {  
 exit = **true**;  
 } **else** {  
 **if** (Checker.*checkGuid*(s)) {  
 System.***out***.println(**"Введённая строка является GUID\n"**);  
 } **else** {  
 System.***out***.println(**"Введённая строка НЕ является GUID\n"**);  
 }  
 }  
 }  
 }  
}

**Checker.java:**

**public class** Checker {  
 **private static final** String ***REGEX*** = **"(?=\\{)\\{[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}}|[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}"**;  
  
 **public static boolean** checkGuid(String guid) {  
 **return** Pattern.*matches*(***REGEX***, guid);  
 }  
}

**Runner.java:**

**public class** Runner {  
 **public static void** main(String[] args) {  
 Menu menu = **new** Menu();  
 menu.start();  
 }  
}

**Menu.java:**

**class** Menu {  
  
 **private static final** Scanner ***in*** = **new** Scanner(System.***in***);  
  
 **void** start() {  
 **boolean** exit = **false**;  
 **while** (!exit) {  
 printMenu();  
 **int** k = -1;  
 **try** {  
 k = Integer.*parseInt*(***in***.nextLine());  
 } **catch** (NumberFormatException e) {  
 System.***out***.println(**"Incorrect input\n"**);  
 }  
 **switch** (k) {  
 **case** 1:  
 enterTextFromTheKeyboard();  
 **break**;  
 **case** 2:  
 enterTextFromFile();  
 **break**;  
 **case** 3:  
 exit = **true**;  
 **break**;  
 }  
 }  
 }  
  
 **private void** printMenu() {  
 System.***out***.println(**"Menu"**);  
 System.***out***.println(**"1. Enter text from the keyboard"**);  
 System.***out***.println(**"2. Enter text from a file"**);  
 System.***out***.println(**"3. Exit"**);  
 System.***out***.print(**"Enter menu number: "**);  
 }  
  
 **private void** enterTextFromTheKeyboard() {  
 System.***out***.print(**"Enter the text before pressing enter: "**);  
 String text = ***in***.nextLine();  
 sortAndPrintWords(text);  
 }  
  
 **private void** enterTextFromFile() {  
 System.***out***.print(**"Enter a file name: "**);  
 String filename = ***in***.nextLine();  
 **try** (Scanner file = **new** Scanner(**new** FileReader(filename))) {  
 sortAndPrintWords(file.useDelimiter(**"\\A"**).next());  
 } **catch** (IOException e) {  
 System.***out***.println(**"Error opening file"**);  
 }  
 }  
  
 **private void** sortAndPrintWords(String text) {  
 Parser parser = **new** Parser();  
 List<Word> words = parser.parse(text);  
 words.sort(Comparator.*comparingDouble*(Word::getShareOfVowelLetters));  
 **for** (Word word : words) {  
 System.***out***.format(**"%s "**, word.toString());  
 }  
 System.***out***.println();  
 }  
}

**Parser.java:**

**public class** Parser {  
  
 **public** List<Word> parse(String text) {  
 List<Word> words = **new** ArrayList<>();  
 Word word = **null**;  
 **boolean** previousIsLetter = **false**;  
 **for** (**int** i = 0; i < text.length(); i++) {  
 Symbol symbol = **new** Symbol(text.charAt(i));  
 **if** (Letter.*isLetter*(symbol)) {  
 **if** (previousIsLetter) {  
 word.addLetter(**new** Letter(symbol));  
 } **else** {  
 word = **new** Word(**new** Letter(symbol));  
 words.add(word);  
 }  
 previousIsLetter = **true**;  
 } **else** {  
 previousIsLetter = **false**;  
 }  
 }  
 **return** words;  
 }  
}

**Word.java:**

**public class** Word {  
 **private** List<Letter> **letters**;  
  
 Word(Letter letter) {  
 **letters** = **new** ArrayList<>();  
 **letters**.add(letter);  
 }  
  
 **void** addLetter(Letter letter) {  
 **letters**.add(letter);  
 }  
  
 **public double** getShareOfVowelLetters() {  
 **int** vowelsNumber = 0;  
 **for** (Letter letter : **letters**) {  
 **if** (letter.isVowel()) {  
 vowelsNumber++;  
 }  
 }  
 **return** (**double**) vowelsNumber / **letters**.size();  
 }  
  
 @Override  
 **public** String toString() {  
 StringBuilder builder = **new** StringBuilder();  
 **for** (Letter letter : **letters**) {  
 builder.append(letter.getSymbol().getCharacter());  
 }  
 **return** builder.toString();  
 }  
}

**Letter.java:**

**class** Letter {  
 **private** Symbol **symbol**;  
  
 **static boolean** isLetter(Symbol symbol) {  
 **return** Character.*isLetter*(symbol.getCharacter());  
 }  
  
 Letter(Symbol symbol) {  
 **if** (!Character.*isLetter*(symbol.getCharacter())) {  
 **throw new** IllegalArgumentException(**"The symbol is not a letter!"**);  
 }  
 **this**.**symbol** = symbol;  
 }  
  
 Symbol getSymbol() {  
 **return symbol**;  
 }  
  
 **boolean** isVowel() {  
 **return "aeiouAEIOUаоуеёыэяиюАОУЕЁЫЭЯИЮ"**.indexOf(**symbol**.getCharacter()) != -1;  
 }  
}

**Symbol.java:**

**class** Symbol {  
 **private final char character**;  
  
 Symbol(**char** character) {  
 **this**.**character** = character;  
 }  
  
 **char** getCharacter() {  
 **return character**;  
 }  
}