# **Тема:** [Утилітарні класи. Обробка масивів і рядків](https://oop-khpi.gitlab.io/" \l "tasks).

**Мета:**

* Розробка власних утилітарних класів.
* Набуття навичок вирішення прикладних задач з використанням масивів і рядків.

**1 ВИМОГИ  
1.1 Розробник**

Інформація про розробника:

- Когутенко Олександр Олексійович;

- КІТ-119Д;

- 11 варіант.

**1.2 Загальне завдання**

1. Розробити та продемонструвати консольну програму мовою Java в середовищі Eclipse для вирішення прикладної задачі за номером, що відповідає збільшеному на одиницю залишку від ділення на 15 зменшеного на одиницю номера студента в журналі групи.

2.При вирішенні прикладних задач використовувати [латинку](https://uk.wikipedia.org/wiki/%D0%9B%D0%B0%D1%82%D0%B8%D0%BD%D1%81%D1%8C%D0%BA%D0%B0_%D0%B0%D0%B1%D0%B5%D1%82%D0%BA%D0%B0).

3.Продемонструвати використання об'єктів класу [StringBuilder](http://docs.oracle.com/javase/8/docs/api/java/lang/StringBuilder.html) або [StringBuffer](http://docs.oracle.com/javase/8/docs/api/java/lang/StringBuffer.html).

4.Застосувати функціональну (процедурну) декомпозицію - розробити власні утилітарні класи та для обробки даних використовувати відповідні [статичні](https://docs.oracle.com/javase/tutorial/java/javaOO/classvars.html) методи.

5.Забороняється використовувати засоби обробки регулярних виразів: класи пакету [java.util.regex](https://docs.oracle.com/javase/8/docs/api/java/util/regex/package-summary.html) ([Pattern](https://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html), [Matcher](https://docs.oracle.com/javase/8/docs/api/java/util/regex/Matcher.html) та ін.), а також відповідні методи класу [String](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html) (matches, replace, replaceFirst, replaceAll, split).

1.3 Задача

Ввести текст. У тексті слова заданої довжини замінити зазначеним рядком. Вивести початковий текст та результат.

2 ОПИС ПРОГРАМИ

2.1 Засоби ООП

Використовуеться інкапсуляція у классі HelperClassWithString.

2.2 Ієрархія та структура класів

Маємо два класи: Main та HelperClassWithString.

У Main виконується інтерактив з користувачем. Тобто форматируваний вивід та корректне прийняття данних.

У HelperClassWithString виконуються основні дії за прикладною задачею. Тобто знаходження слів за вказаною кількістю літер та зміна їх на вказаний рядок. Знаходив такі слова за пропусками: знаходив різницю між індексами пропусків і якщо вона ровнялась вказаній довжині то я по <преший індекс> + 1 змінював слово (тобто до пропуска) навказаний рядок.

2.3 Важливі фрагменти програми

package ua.khpi.oop.kogutenko03;  
  
import java.util.Scanner;  
  
*/\*\*  
 \* The type Main.  
 \*/*  
public class Main {  
  
 */\*\*  
 \* The entry point of application.  
 \*  
 \** ***@param*** *args the input arguments  
 \*/* public static void main(String[] args)  
 {  
 try (Scanner scanner = new Scanner(System.*in*))   
 {  
 ...  
 String mainStr = scanner.nextLine();  
 HelperClassWithString mainHelperStr = new HelperClassWithString(mainStr);  
   
 ...  
 int length = scanner.nextInt();  
   
 ...  
 scanner.nextLine();  
 String newWord = scanner.nextLine();  
   
   
 String newStr = mainHelperStr.*replaceAllWordsOn*(length, newWord);//new String(mainHelperStr.replaceAllWordsOn(length, newWord));  
 ...

scanner.close();  
 }   
 catch (Exception e)  
 {  
 System.*out*.println(e);  
 }  
   
 }  
  
}

package ua.khpi.oop.kogutenko03;  
import java.lang.StringBuilder;  
  
*/\*\*  
 \* The type Helper class with string.  
 \*/*public class HelperClassWithString  
{  
 private static StringBuilder *str*;  
 private static int *count*;  
 private static int *length*;  
  
 */\*\*  
 \* Instantiates a new Helper class with string.  
 \*  
 \** ***@param*** *str the str  
 \*/* public HelperClassWithString(String str)  
 {  
 if(str.isEmpty())  
 {  
 System.*out*.println("Line is empty");  
 this.*str* = new StringBuilder("EmptyLine");  
 }  
 else  
 {  
 //System.out.println("Constract with param");  
 this.*str* = new StringBuilder(str);  
 *CountWordsInHelper*();  
 *length* = this.*str*.length();  
 }  
 }  
  
 */\*\*  
 \* Sets length.  
 \*  
 \** ***@param*** *len the length  
 \*/* public static void setLength(int len)  
 {  
 *length* = len;  
 }  
  
 */\*\*  
 \* Gets length.  
 \*  
 \** ***@return*** *the length  
 \*/* public static int getLength()  
 {  
 return *length*;  
 }  
  
 */\*\*  
 \* Sets str.  
 \*  
 \** ***@param*** *s the str  
 \*/* public static void setStr(String s)  
 {  
 *str* = new StringBuilder(s);  
 }  
  
 */\*\*  
 \* Gets str.  
 \*  
 \** ***@return*** *the str  
 \*/* public static StringBuilder getStr()  
 {  
 return *str*;  
 }  
  
 */\*\*  
 \* Gets count.  
 \*  
 \** ***@return*** *the count  
 \*/* public static int getCount()  
 {  
 return *count*;  
 }  
  
 */\*\*  
 \* Sets count.  
 \*  
 \** ***@param*** *c the count  
 \*/* public static void setCount(int c)  
 {  
 *count* = c;  
 }  
  
 */\*\*  
 \* Count words in helper int.  
 \*  
 \** ***@return*** *the int  
 \*/* public static int CountWordsInHelper()  
 {  
 int count = 0;  
 if(*str*.length() != 0){  
 count++;  
 //Проверяем каждый символ, не пробел ли это  
 for (int i = 0; i < *str*.length(); i++)   
 {  
 if(*str*.charAt(i) == ' ')  
 {  
 //Если пробел - увеличиваем количество слов на 1  
 count++;  
 }  
 }  
 }   
 *setCount*(count);  
 return *getCount*();  
 }  
  
 */\*\*  
 \* Count words in string int.  
 \*  
 \** ***@param*** *str the str  
 \** ***@return*** *the int  
 \*/* public static int CountWordsInString(String str)  
 {  
 int count = 0;  
 if(str.length() != 0){  
 count++;  
 //Проверяем каждый символ, не пробел ли это  
 for (int i = 0; i < str.length(); i++)   
 {  
 if(str.charAt(i) == ' ')  
 {  
 //Если пробел - увеличиваем количество слов на 1  
 count++;  
 }  
 }  
 }   
 return count;  
 }  
   
 private static int[] indexingSpaces(StringBuilder line)  
 {  
 int[] index = new int[*getCount*() + 1];  
 if(line.length() != 0)  
 {  
 //System.out.println("Indexing...");  
 for (int i = 0, indx = 1; i < line.length(); i++)   
 {  
 //System.out.printf("%-3c\_", line.charAt(i));  
 if( line.charAt(i) == ' '   
 || line.charAt(i) == '.'   
 || line.charAt(i) == '!'   
 || line.charAt(i) == '?'  
 || line.charAt(i) == ','  
 || line.charAt(i) == ','  
 || line.charAt(i) == '\0'  
 || line.charAt(i) == ',' && line.charAt(i + 1) == ' '  
 || line.charAt(i) == '!' && line.charAt(i + 1) == ' '  
 || line.charAt(i) == '?' && line.charAt(i + 1) == ' '  
 || line.charAt(i) == '\n'  
 || line.charAt(i) == '\0' && line.charAt(i) == '\n'  
 )  
 {  
 index[indx++] = i;  
 }  
 }  
 }  
 return index;  
 }  
  
 */\*\*  
 \* Replace all words on string.  
 \*  
 \** ***@param*** *len the len  
 \** ***@param*** *onLine the on line  
 \** ***@return*** *the string  
 \*/* public static String replaceAllWordsOn(int len, String onLine)  
 {  
 StringBuilder line = new StringBuilder(*str*);  
 int[] indexSpace = *indexingSpaces*(line);  
 indexSpace[0] = -1;  
 for (int i = 0; i < indexSpace.length - 1; i++)   
 {  
 if(i == 0 && ( indexSpace[i + 1] - indexSpace[i] - 1 == len ) )  
 {  
 //System.out.println("\n\nfirst indexing: " + res);  
 line.delete(indexSpace[i] + 1, indexSpace[i + 1]);  
 line.insert(indexSpace[i] + 1, onLine);   
 indexSpace = *indexingSpaces*(line);  
 }  
 else if( i > 0 && Math.*abs*(indexSpace[i + 1] - Math.*abs*(indexSpace[i]) - 1) == len) // ? | |[i + 1]| - [i] | == len ?  
 {  
 //line.replace(indexSpace[i] + 1, indexSpace[i + 1], onLine);  
 line.delete(indexSpace[i] + 1, indexSpace[i + 1]);  
 line.insert(indexSpace[i] + 1, onLine);  
 indexSpace = *indexingSpaces*(line);  
 }  
 }  
 String output =new String(line);  
 return output;  
 }  
}

**3 ВАРІАНТИ ВИКОРИСТАННЯ**

Программа предназначена за для зміни у всьому вказаному тексті заданої довжини слова на вказану рядок. Тобто для зміни туксту.

**ВИСНОВКИ**

Ознайомився з утилітарними класами та з обробкою масивів та рядків. Розробив власний утилітарний клас та набув навичок з вирішення прикладної задачі з використанням масивів та рядків.