Лабораторна робота №4

**ІНТЕРАКТИВНІ КОНСОЛЬНІ ПРОГРАМИ ДЛЯ ПЛАТФОРМИ JAVA SE**

**Мета:** Реалізація діалогового режиму роботи з користувачем в консольних програмах мовою Java.

**ВИМОГИ**

**Розробник:**

- Чугунов Вадим Юрійович;

- КІТ-119а;

- Варіант №24.

**Загальне завдання:**

1) Використовуючи програму рішення завдання лабораторної роботи №3, відповідно до прикладної задачі забезпечити обробку команд користувача у вигляді текстового меню:

* введення даних;
* перегляд даних;
* виконання обчислень;
* відображення результату;
* завершення програми і т.д.

2) Забезпечити обробку параметрів командного рядка для визначення режиму роботи програми:

* параметр "-h" чи "-help": відображається інформація про автора програми, призначення (індивідуальне завдання), детальний опис режимів роботи (пунктів меню та параметрів командного рядка);
* параметр "-d" чи "-debug": в процесі роботи програми відображаються додаткові дані, що полегшують налагодження та перевірку працездатності програми: діагностичні повідомлення, проміжні значення змінних, значення тимчасових змінних та ін.

**ОПИС ПРОГРАМИ**

**Опис змінних:**

String text; // текст, який ми ініціалізуємо та проводимо в ньому пошук

Scanner in; // змінна для введення даних

ArgumentTaker argTaker; // обробник консольних команд (-h, -d)

final String[] arr; // масив підрядків який буде заповнений результатом Split методу

HashMap<String, Integer> wordToCount //наша хеш таблиця для виводу слова та його поторів в тексті

**Ієрархія та структура класів:**

**class** Main – точка входу в програму.

**class** ArgumentTaker – клас, який реалізує обробку консольних команд.

**class** HelpMethods – клас, який має метод Split та метод, який виконує завдання.

**class** Interface – клас, який реалізує діалогове меню.

**ТЕКСТ ПРОГРАМИ**

Текст класу **Main**:

package ua.oop.khpi.chugunov04;

public class Main {

/\*\*

\* An entry point - main method.

\* @param args - arguments of main method

\*/

public static void main(final String[] args) {

final int exit = 0;

final int setValues = 1;

final int getValues = 2;

final int exec = 3;

final int printResult = 4;

String text = null;

ArgumentTaker argTaker = new ArgumentTaker(args);

if (!argTaker.empty()) {

argTaker.execute();

}

if (ArgumentTaker.isDebug()) {

System.out.println("\n~~~~~~~~~~YOU ARE IN DEBUG MODE~~~~~~~~~~");

System.out.println("\n~The debug mode will help you learning code easier~");

System.out.format("\n==> Debugging...<==");

}

/\*\*

\* Interface menu to dialog with user

\*/

do {

Interface.mainMenu();

Interface.enterChoice();

switch (Interface.getChoice()) {

case exit:

if (ArgumentTaker.isDebug()) {

System.out.println("\nYour choice is (0). Exiting...");

System.out.format("\n==> Debugging...<==");

System.out.println("\n~~~~~~~~~~DEBUG MODE OFF~~~~~~~~~~");

}

break;

case setValues:

if (ArgumentTaker.isDebug()) {

System.out.format("\n==> Debugging...<==");

System.out.println("\nYour choice is (1). Setting values...");

}

text = Interface.AddValues();

break;

case getValues:

if (ArgumentTaker.isDebug()) {

System.out.format("\n==> Debugging...<==");

System.out.println("\nYour choice is (2). Getting values...");

}

if (text != null ) {

Interface.printValue(text);

} else {

System.out.format("%nFirst you need to add values.");

}

break;

case exec:

if (ArgumentTaker.isDebug()) {

System.out.format("\n==> Debugging...<==");

System.out.println("\nYour choice is (3). Executing task...");

}

if (text != null) {

final String[] arr1 = HelpMethods.SplitString(text);

System.out.println("\nLoading...");

System.out.println("\nTask competed...");

if (ArgumentTaker.isDebug()) {

System.out.format("\n==> Debugging...<==");

}

} else {

System.out.format("%nFirst you need to add values.");

}

break;

case printResult:

if (ArgumentTaker.isDebug()) {

System.out.format("\n==> Debugging...<==");

System.out.format("%nYour choice is (4). " + "Printing out result...%n");

}

if (text != null) {

final String [] arr2 = HelpMethods.SplitString(text);

HelpMethods.PrintResult(arr2);

if (ArgumentTaker.isDebug()) {

System.out.format("\n==> Debugging...<==");

}

} else {

System.out.format("%nERROR!Please, enter your values!.");

if (ArgumentTaker.isDebug()) {

System.out.format("\nA text hasn't our value (text = null)");

System.out.format("\n==> Debugging...<==");

}

}

break;

default:

System.out.println("\nERROR!ENTER A CORRECT VALUE!.");

}

} while (Interface.getChoice() != 0);

}

}

Текст класу **ArgumentTaker**:

package ua.oop.khpi.chugunov04;

public class ArgumentTaker {

/\*\* Checking an arguments of command line

\* for is debug mode on or no \*/

private String[] arguments;

private static boolean debug = false;

static boolean isDebug() {

return debug;

}

ArgumentTaker(final String[] args) {

this.arguments = args;

}

boolean empty() {

return arguments.length == 0;

}

void execute() {

for (String i : arguments) {

switch (i) {

case "-h":

System.out.println("\nAuthor: Chugunov Vadim, KIT-119a");

System.out.println("Task: Enter text. "

+ "Insert your text "

+ "after the program starts searching."

+ "\nOutput the initial text and result of your task.");

System.out.println("-h (-help) : Print info about "

+ "the task and console commands.");

System.out.println("-d (-debug) : "

+ "Displays additional data values of some variables.");

break;

case "-help":

System.out.println("\nAuthor: "

+ "Chugunov Vadim, KIT-119a");

System.out.println("Task: Enter text. "

+ "Insert your text "

+ "after the program starts searching. "

+ "Output the initial text and result of your task.");

System.out.println("-h (-help) : Print info about "

+ "the task and console commands.");

System.out.println("-d (-debug) : "

+ "Displays additional data values of some variables.");

break;

case "-d":

debug = true;

break;

case "-debug":

debug = true;

break;

default:

System.out.format("%n Incorrect command %s.%n", i);

System.out.println("-h, -help, -d, -debug "

+ "is only allowed.");

}

}

}

}

Текст класу **HelpMethods**:

package ua.oop.khpi.chugunov04;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

public class HelpMethods {

public static String[] SplitString(String text) {

if (ArgumentTaker.isDebug()) {

System.out.format("------------------------------------------------");

System.out.format("\n String text - our text before ~Split Method~))");

System.out.format("\n String text = "+text);

}

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

StringBuilder builder = new StringBuilder();

for(char symbol : text.toCharArray()) {

if((int)symbol == 32 | (int)symbol == 33 |(int)symbol == 58|(int)symbol == 44|(int)symbol == 46) {

words.add(builder.toString());

builder = new StringBuilder();

continue;

}

builder.append(symbol);

}

if(builder.length() != 0) {

words.add(builder.toString());

}

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

if(words.get(i).length() == 0) {

words.remove(i);

}

}

String[] output = new String[words.size()];

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

output[i] = words.get(i);

}

if (ArgumentTaker.isDebug()) {

System.out.format("\n String text - our text after ~Split Method~))");

System.out.format("\n String text = ");

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

System.out.format(output[i]);

}

System.out.format("\n------------------------------------------------");

System.out.println();

}

return output;

}

public static void PrintResult (String[]words){

if (ArgumentTaker.isDebug()) {

System.out.format("\n------------------------------------------------------------------------------");

System.out.format("\n String[]words - our text after (split method) that taken to complete the task!");

System.out.format("\n String[]words = ");

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

System.out.format(words[i]);

}

System.out.format("\n------------------------------------------------------------------------------");

System.out.format("\n A table with 2 colons: "

+"\n1) A word - word from our text;"

+"\n2) Count - a number of repetitions of a word."

);

}

System.out.println( "\n=============================");

System.out.println( "A Word" + "\t\t\t" + "Count");

System.out.println( "=============================");

HashMap<String, Integer> wordToCount = new HashMap<>();

for (String word : words) {

if (!wordToCount.containsKey(word)) {

wordToCount.put(word, 0);

}

wordToCount.put(word, wordToCount.get(word) + 1);

}

for (String word : wordToCount.keySet()) {

System.out.println(word + "\t\t\t" + wordToCount.get(word));

}

System.out.println( "=============================");

}

}

Текст класу **Interface**:

package ua.oop.khpi.chugunov04;

import java.util.Scanner;

public class Interface {

private Interface() {

}

/\*\* Gets values from user. \*/

private static Scanner scan = new Scanner(System.in);

private static int choice;

static int getChoice() {

return choice;

}

static void mainMenu() {

System.out.format("%n1. Enter values.%n");

System.out.format("2. Print values.%n");

System.out.format("3. Task completion.%n");

System.out.format("4. Print result.%n");

System.out.format("0. Exit.%n");

System.out.format("Enter your choose: ");

}

static void enterChoice() {

choice = scan.nextInt();

scan.nextLine();

}

/\*\*

\* The Adding values to the text method

\*

\* @return the text that we have initialized

\*/

public static String AddValues(){

System.out.println( "Enter the text:");

Scanner in = new Scanner(System.in);

String text = in.nextLine();

return text;

}

/\*\*

\* The printing our text method

\*

\* @param text - value which we have initialized

\*/

public static void printValue(String text){

System.out.println("Your text is :"+ text);

}

}

**РЕЗУЛЬТАТ РОБОТИ ПРОГРАМИ**

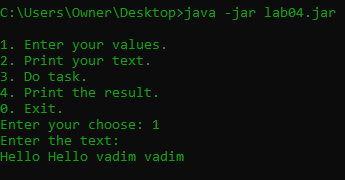
****

Рисунок 4.1 – Введення тексу для завдання

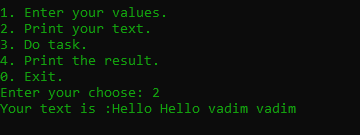
****

Рисунок 4.2 – Вивід початкового тексту

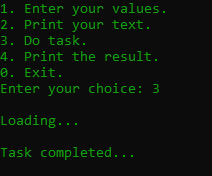
****

Рисунок 4.3 – Виконання завдання

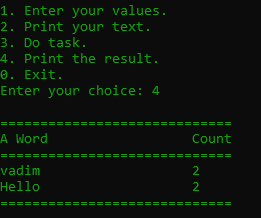
****

Рисунок 4.4 – Результат виконання завдання

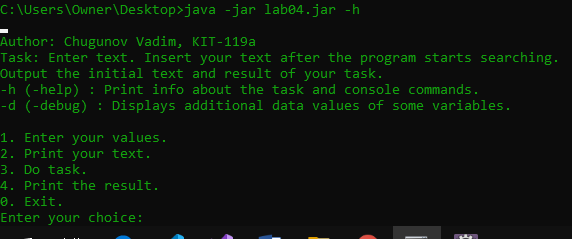
****

Рисунок 4.5 – Запуск програми за переданим параметром -h

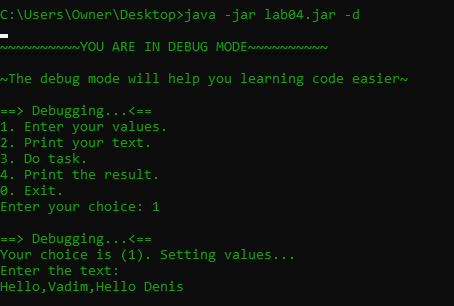
****

Рисунок 4.6 – Запуск програми за переданим параметром –d

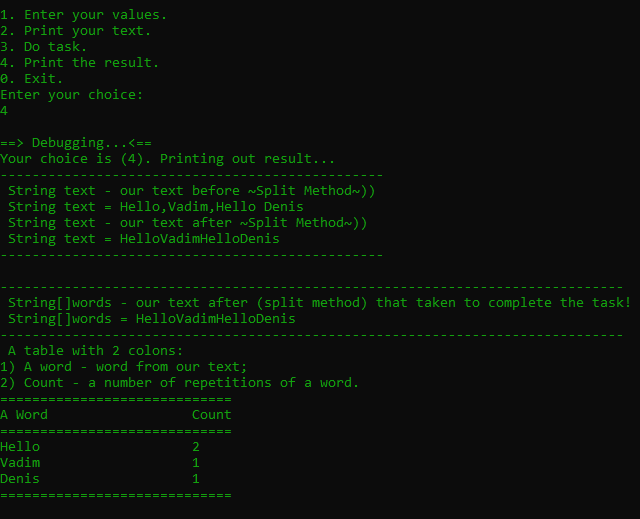


Рисунок 4.7 – Вивід початкових даних під час роботи програми у Debug режимі

**ВАРІАНТИ ВИКОРИСТАННЯ**

Програму можна використовувати для пошуку кількості поторів у тексті потрібного нам слова. Програму можна запускати у двох режимах: Debug (додається допоміжна інформація для користувача) та Release. Програма реалізує в собі роботу у режимі Help, який дає певну інформацію про автора програми, та забезпечує користувача потрібною інформаціє для комфортної роботи з програмою.

**ВИСНОВОК**

Під час виконання лабораторної роботи, отримали практичні навички реалізації діалогового режиму роботи з користувачем в консольній програмі. Реалізовали клас інтерфейсного меню з користувачем, клас, який виконує індивідуальне завдання, та класс, який дає змогу роботи програми у Debug та Help режимах.