Тема: Інтерактивні консольні програми для платформи Java SE.
Мета: Реалізація діалогового режиму роботи з користувачем в консольних програмах мовою Java.

1 ВИМОГИ

1.1 Розробник

Інформація про розробника:
- Когутенко Олександр Олексійович;
- КІТ-119Д;
- 11 варіант.

1.2 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.

2 ОПИС ПРОГРАМИ

2.1 Засоби ООП

Використовується інкапсуляція у классі HelperClassWithConsole.

2.2 Ієрархія та структура класів

Маємо два класи: Main та HelperClassWithConsole.

У Main виконується інтерактив з користувачем. Тобто форматируваний вивід та корректне прийняття данних з використанням прийняття параметрів що потрібні за прикладною задачею, однак добавивши пару своїх.

У HelperClassWithConsole виконуються основні дії за прикладною задачею. Тобто вивід допоміжної інформації, параметр який вказує значення

змінних та прикладна задача з лабораторної роботи №3.

2.3 Важливі фрагменти програми

```
package ua.khpi.oop.kogutenko04;

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)
  {
    HelperClass helper = new HelperClass();
    boolean help = informHelp(args);
    boolean debug = informDebug(args);
    boolean nextAction = true;
    int input;
    Scanner sc = new Scanner(System.in);
    while(nextAction) {
      System.out.println("Input choice\n" +
          "\t1 - next sentence\n" +
          "\t2 - exit\n" +
          "Enter choice > ");
      input = sc.nextInt();
      switch(input)
      {
        case 1:
          if (help == false && debug == false) {
            helper.changedText();
          } else if (help == true && debug == false) {
            helper.printHelpInfo();
            helper.changedText();
          } else if (help == false && debug == true) {
            helper.changedText();
            helper.debuggerInHelper();
          } else if (help == true && debug == true) {
            helper.printHelpInfo();
            helper.changedText();
            helper.debuggerInHelper();
          }
          break;
        case 2:
          nextAction = false;
          break;

      }
```

```
    }
  }

  private static boolean informHelp(String args[])
  {
    int index;
    boolean help = false;
    for (index = 0; index < args.length; index++) {
      String opt = args[index];
      switch (opt) {
        case "-h":
          help = true;
          System.out.println("Helper turn on");
          break;
        case "-help":
          help = true;
          System.out.println("Helper turn on");
          break;
        default:
          if (!opt.isEmpty() && opt.charAt(0) == '-') {
            System.out.println("is empty param");
          }
      }
    }
    return help;
  }

  private static boolean informDebug(String args[])
  {
    int index;
    boolean debug = false;
    loop:
    for (index = 0; index < args.length; index++) {
      String opt = args[index];
      switch (opt) {
        case "-d":
          debug = true;
          System.out.println("Debuger turn on");
          break;
        case "-debug":
          debug = true;
          System.out.println("Debuger turn on");
          break;
        default:
          if (!opt.isEmpty() && opt.charAt(0) == '-') {
            System.out.println("is empty param");
          }
          break loop;
      }
    }
    return debug;
```

```
    }
}
```

```
package ua.khpi.oop.kogutenko04;


import java.lang.StringBuilder;
import java.text.SimpleDateFormat;
//import java.util.Collections;
import java.util.Date;
import java.util.Scanner;

/**
 * The type Helper class with console.
 */
public class HelperClass {

    private StringBuilder str;
    private Integer count;
    private Integer length;

    /**
     * Instantiates a new Helper class with console.
     */
    public HelperClass() {
        str = new StringBuilder("EmptyLine");
        count = 0;
        length = 0;
    }

    /**
     * Instantiates a new Helper class with console.
     *
     * @param str the str
     */
    public HelperClass(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);
            count = CountWordsInHelper();
            length = this.str.length();
        }
    }

    /**
     * Gets info of helper object.
     *
     * @return the info of helper object
```

```
 */
public String getInfoOfHelperObject()
{
    return str.getClass().getSimpleName() + " 'str' has '" + getStr() + "'\n"
            + count.getClass().getSimpleName() + " 'count' has '" + getCount() + "'\n"
            + length.getClass().getSimpleName() + " 'length' has '" + getLength() + "'";
}

public void setInfoHelperObject(String str)
{
    setStr(str);
    setCount(CountWordsInHelper());
    setLength(str.length());
}
/**
 * Sets length.
 *
 * @param length the length
 */
public void setLength(int length) {
    this.length = length;
}

/**
 * Gets length.
 *
 * @return the length
 */
public int getLength() {
    return length;
}

/**
 * Sets str.
 *
 * @param str the str
 */
public void setStr(String str) {
    this.str = new StringBuilder(str);
}

/**
 * Gets str.
 *
 * @return the str
 */
public StringBuilder getStr()
{
    return str;
}

/**
```

```java
     * Gets count.
     *
     * @return the count
     */
    public int getCount() {
        return count;
    }

    /**
     * Sets count.
     *
     * @param count the count
     */
    public void setCount(int count) {
        this.count = count;
    }

    /**
     * Count words in helper int.
     *
     * @return the int
     */
    public 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 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 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') {
                index[indx++] = i;
            }
        }
    }
    return index;
}

/**
 * Replace all words on string.
 *
 * @param len    the len
 * @param onLine the on line
 * @return the string
 */
public 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;
```

```
    }

    /**
     * Print help info.
     */
    public void printHelpInfo() {
        System.out.println("Hello, my name is Alex Kogutenko\n" + "I am from Ukrain and studing
at NTU \"KHPI\"\n"
                + "This is a test console-project with a debug programs.\n"
                + "Such commands are present so far:\n"
                + "\t-h | -help  \t-\t command for summary information about other commands
(important to remmember!)\n"
                + "\t-d | -debug \t-\t file debugger command\n" );
        System.out.println("Tap any key...");
    }

    /**
     * Debugger in helper.
     */
    public void debuggerInHelper()
    {
        Date date = new Date();
        SimpleDateFormat formatDate = new SimpleDateFormat("'\nDate:' dd.MM.yyyy '\nTime:
'HH:mm:ss '\n'" );
        System.out.println(formatDate.format(date) + "values of the variables:\n"
                + getInfoOfHelperObject());

    }

    public void changedText()
    {
        System.out.println();
        System.out.println("Enter the text. In the text,\nreplace the words of the specified length
with the specified line");
        Scanner scan = new Scanner(System.in);
        System.out.print("Main line: ");
        String mainStr = scan.nextLine();
        setInfoHelperObject(mainStr);
        System.out.print("Enter number of letters in word which you want to changed: ");
        int length = scan.nextInt();
        System.out.print("Enter word to replace: ");
        scan.nextLine();
        String newWord = scan.nextLine();
        String newStr = replaceAllWordsOn(length, newWord);//new
String(mainHelperStr.replaceAllWordsOn(length, newWord));
        System.out.println("----------------------------------------");
        System.out.println("No changed line: " + mainStr);
        System.out.println("Result:          " + newStr);
        //scan.close();
    }

}
```

3 ВАРІАНТИ ВИКОРИСТАННЯ

Програма предназначена за для розширеного використання лабороторної роботи №3 за допомогою використання параметрів перед запуском.

Рисунок 4.1 - входні параметри.

```
Input choice
    1 - next sentence
    2 - exit
Enter choice >
1
Hello, my name is Alex Kogutenko
I am from Ukrain and studing at NTU "KHPI"
This is a test console-project with a debug programs.
Such commands are present so far:
    -h | -help      -    command for summary information about other commands (important to remmember!)
    -d | -debug     -    file debugger command

Tap any key...

Enter the text. In the text,
replace the words of the specified length with the specified line
Main line: qwe qwe sd
Enter number of letters in word which you want to changed: 3
Enter word to replace: ++
-------------------------------------------
No changed line: qwe qwe sd
Result:          ++ ++ sd

Date: 21.12.2020
Time: 18:18:07
values of the variables:
StringBuilder 'str' has 'qwe qwe sd'
Integer 'count' has '3'
Integer 'length' has '10'
Input choice
    1 - next sentence
    2 - exit
Enter choice >
2

Process finished with exit code 0
```

Рисунок 4.2 - результат виконання.

ВИСНОВКИ

Розробив інтерактивну консольну програму для платформи Java SE.

Ознайомився з реалізацією діалогового режиму роботи з користувачем в консольних програмах мовою Java.