**Інтерактивні консольні програми для платформи Java SE**

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

**Вимоги**

1. Використовуючи програму рішення завдання [лабораторної роботи №3](https://oop-khpi.gitlab.io/%D0%B7%D0%B0%D0%B2%D0%B4%D0%B0%D0%BD%D0%BD%D1%8F/task03/), відповідно до [прикладної задачі](https://oop-khpi.gitlab.io/%D0%B7%D0%B0%D0%B2%D0%B4%D0%B0%D0%BD%D0%BD%D1%8F/task03/" \l "_4) забезпечити обробку команд користувача у вигляді текстового [меню](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%BD%D1%8E_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)" \l ".D0.9C.D0.B5.D0.BD.D1.8E_.D0.B2_.D0.B8.D0.BD.D1.82.D0.B5.D1.80.D1.84.D0.B5.D0.B9.D1.81.D0.B5_.D0.BA.D0.BE.D0.BC.D0.B0.D0.BD.D0.B4.D0.BD.D0.BE.D0.B9_.D1.81.D1.82.D1.80.D0.BE.D0.BA.D0.B8):
   * введення даних;
   * перегляд даних;
   * виконання обчислень;
   * відображення результату;
   * завершення програми і т.д.
2. Забезпечити обробку параметрів [командного рядка](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BA%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D0%BD%D0%BE%D0%B3%D0%BE_%D1%80%D1%8F%D0%B4%D0%BA%D0%B0) 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.

**Розробник**: Рябов Олексій Володимирович КІТ119а №18.

**Опис програми**

**Засоби ООП**: клас, метод классу.

**Структура класів:** один публічний клас Main, один утилітарний клас Helper.

**Важливі фрагменти програми:**

**private** **static** **int** *mean* = 0;

**static** ArrayList<StringBuilder> *line* = **new** ArrayList<StringBuilder>() ;

**static** **private** **void** findmean(ArrayList<StringBuilder> line) {

**int** m = 0;

**for**(**int** i =0;i<line.size();i++)

m+=line.get(i).length();

*setMean*(m/line.size());

**if**(*debag*)

System.***out***.println("Mean: " + *mean*);

**return**;

}

**private** **static** **boolean** *debag* = **false**;

**public** **static** **void** debagMod(**boolean** debagMod) {

*debag* = debagMod;

System.***out***.printf("Debag mod ");

**if** (*debag*) {

System.***out***.printf("on\n");

}**else**

{

System.***out***.printf("off\n");

}

}

**public** **static** **void** start(StringBuilder ln) {

**if**(*debag*) {

System.***out***.println("Start line: " + ln);

};

**if**(ln==**null**||ln.length()==0) {

System.***out***.println("Line is empty");

**return**;

}

*clottingSpace*(ln);

*line* = *findArray*(ln);

*findmean*(*line*);

**if**(*getMean*() == 0) {

**throw** **new** NullPointerException("Line have nott element or size of line is zero");

}

}

**static** **private** ArrayList<StringBuilder> findArray(StringBuilder ln){

ArrayList<StringBuilder> line = **new** ArrayList<StringBuilder>();

ArrayList<Integer> IndexSpace = **new** ArrayList<Integer>();

**int** length = ln.length();

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

**if**(ln.charAt(i)==' ')

IndexSpace.add(i);

}

**if**(IndexSpace.size() == 0) {

line.add(**new** StringBuilder(ln));

}

**else**

{

line.add(**new** StringBuilder(ln.substring(0, IndexSpace.get(0))));

IndexSpace.add(length);

**for**(**int** i=1;i<IndexSpace.size();i++) {

**if**(ln.substring(IndexSpace.get(i-1)+1, IndexSpace.get(i))!="")

line.add(**new** StringBuilder(ln.substring(IndexSpace.get(i-1)+1, IndexSpace.get(i))));

}

}

**return** line;

}

**static** **private** StringBuilder clottingSpace(StringBuilder line) {

**for**(**int** i = 0; i< line.length()-1;i++)

{

**if**(line.charAt(i)==' '&&line.charAt(i+1)==' ') {

line.deleteCharAt(i+1);

i--;

}

}

**if**(line.charAt(line.length()-1)==' ')

line.deleteCharAt(line.length()-1);

**if**(line.charAt(0)==' ')

line.deleteCharAt(0);

**if**(*debag*)

System.***out***.println("Line after clotting space: " + line);

**return** line;

}

**static** **public** **void** printline() {

**if**(*line*.isEmpty()) {

System.***out***.println("Line is empty");

**return**;

}

**if**(*debag*)

System.***out***.println("Count line: " + *line*.size());

System.***out***.println("Lines are bigger than average: ");

**for**(StringBuilder n : *line*) {

**if**(n.length() >= *mean*) {

System.***out***.println(n+"\nKenght of line:" + n.length());

}

}

System.***out***.println("Lines smaller than average: ");

**for**(StringBuilder n : *line*) {

**if**(n.length() < *mean*) {

System.***out***.println(n +"\nLenght of line:" + n.length());

}

}

}

**public** **static** **int** getMean() {

**return** *mean*;

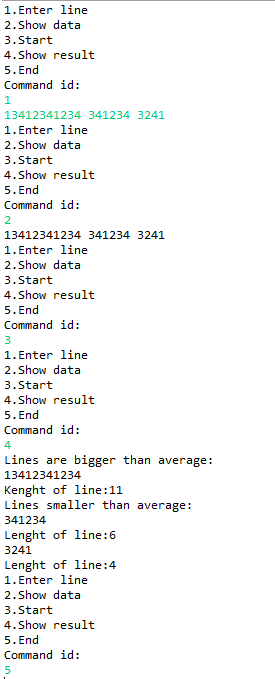
}

**public** **static** **void** setMean(**int** mean) {

Helper.*mean* = mean;

}

**Результати роботи**



**Висновки**

Оволодів навичками роботи з консольним компілюванням, створення консольного меню та режиму “debag”.