Лабораторна робота №11.

Регулярні вирази. Перевірка даних

**Мета роботи:**

* Ознайомлення з принципами використання регулярних виразів для перевірки рядка на відповідність шаблону.

**1. Вимоги до лабораторної роботи**

* 1. **Розробник:**

*студент Литвин Ігнатій Ігоревич; КІТ-26А; Варіант №7*

* 1. **Загальне завдання**

Продемонструвати ефективне (оптимальне) використання регулярних виразів для перевірки коректності (валідації) даних, що вводяться, перед записом в domain-об’єкти відповідно до призначення кожного поля для заповнення розробленого контейнера:

* при зчитуванні даних з текстового файла в автоматичному режимі;
* при введенні даних користувачем в діалоговому режимі.
  1. **Прикладна задача**

Бюро знайомств. Запис про клієнта: стать; реєстраційний номер; дата реєстрації; відомості про себе (довільний набір властивостей: ім’я, зріст, колір очей, дата народження, хобі тощо); вимоги до партнера (довільний набір властивостей).

**2. Опис програми**

Програма реалізована у вигляді інтерактивного консольного вікна з діалоговим режимом роботи з користувачем.

Основне призначення: демонстрація управління масивом domain-об’єктів. Реалізовано додавання та генерування нових об’єктів, видалення, показ інформації.

**2.1 Ієрархія та структура класів**

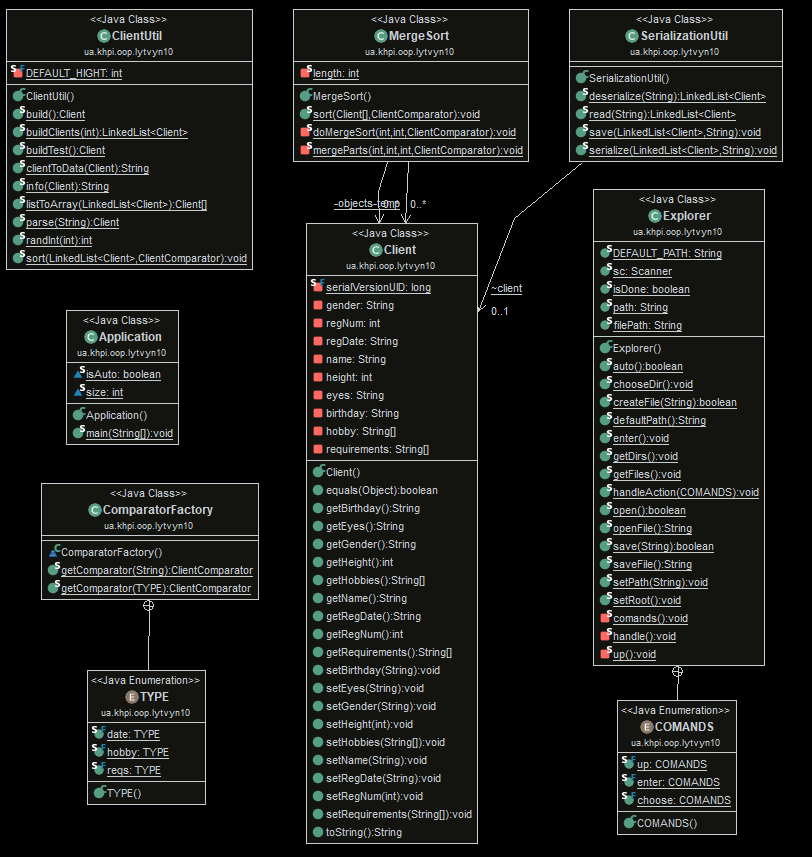
****

Рисунок 1 « Ієрархія та структура класів»

Рисунок 1 « Ієрархія та структура класів»

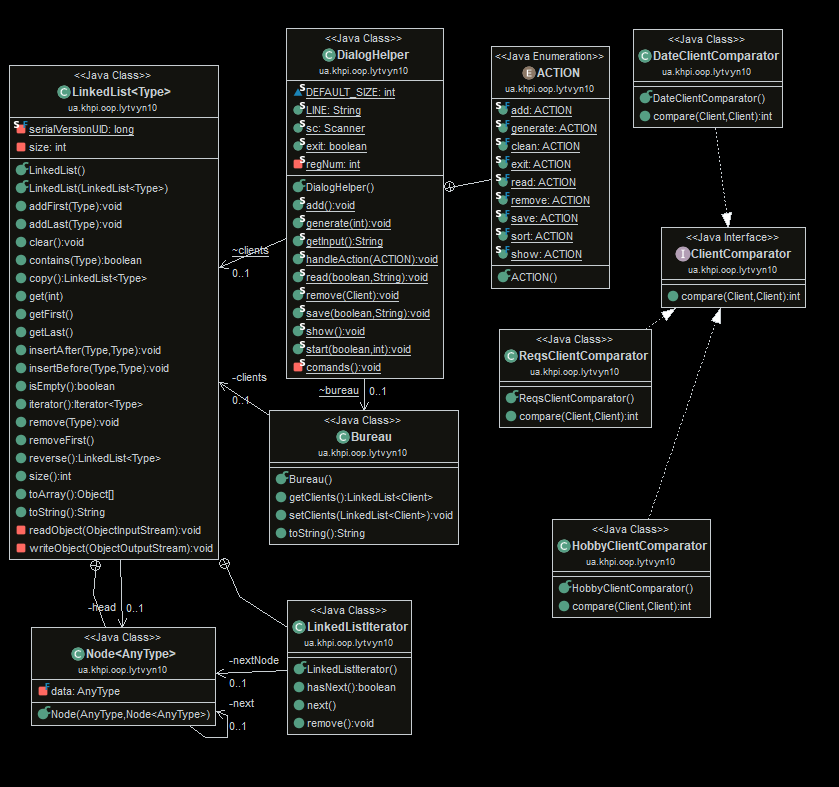


Рисунок 2 « Ієрархія та структура класів»

**2.2 Важливі фрагменти програми**

package ua.khpi.oop.lytvyn11;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.Random;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

/\*\*

\* Містить методи створення та виведення клієнтів.

\*

\* @author student Lytvyn I.I. KIT-26A

\*/

public class ClientUtil {

enum REGEX {

date, height, string, name, gender

}

private static final int MIN\_HEIGHT = 150;

private static final int MAX\_HEIGHT = 210;

static String[] filtrs = {

"((0[1-9]|[12]\\d)\\.(0[1-9]|1[012])|30\\.(0[13-9]|1[012])|31\\.(0[13578]|1[02]))\\.(19|20)\\d\\d",

"2[0-3]\\d|1[5-9]\\d",

"[а-яіїА-ЯІЇ][а-яіїА-ЯІЇ -']{1,50}",

"[А-ЯІЇ][а-яіїА-ЯІЇ -']{1,20}",

"(Чоловік)|(Жінка)" };

static String[] MALE\_NAMES = { "АНДРІЙ", "АРСЕН", "АРСЕНІЙ", "АРТЕМ",

"АРТУР", "БОГДАН", "ВАДИМ", "ВАСИЛЬ", "ВІКТОР", "ВІТАЛІЙ",

"ВЛАДИСЛАВ", "ВОЛОДИМИР", "ДАВИД", "ДАНИЇЛ", "ДАНИЛО", "ДЕМ'ЯН",

"ДЕНИС", "ДМИТРО", "ЗАХАР", "ІВАН", "ІГОР", "ІЛЛЯ", "ЛУК'ЯН",

"ЛЮБОМИР", "МАКСИМ", "МАРК", "МАРКІЯН", "МАРКО", "МАР'ЯН", "МАТВІЙ",

"МИКОЛА", "МИХАЙЛО", "НАЗАР", "НАЗАРІЙ", "ОЛЕГ", "ОЛЕКСАНДР",

"ОЛЕКСІЙ", "ОСТАП", "ПАВЛО", "РОМАН", "РОСТИСЛАВ", "РУСЛАН",

"СВЯТОСЛАВ", "СЕРГІЙ", "СТАНІСЛАВ", "СТЕПАН", "ТАРАС", "ТИМОФІЙ",

"ЮРІЙ", "ЯРОСЛАВ" };

static String[] FEMALE\_NAMES = { "АЛІНА", "АНАСТАСІЯ", "АНГЕЛІНА",

"АНДРІАНА", "АННА", "БОЖЕНА", "ВАЛЕРІЯ", "ВАСИЛИНА", "ВЕРОНІКА",

"ВІКТОРІЯ", "ВІРА", "ДАРИНА", "ДАРІЯ", "ДІАНА", "ЕВЕЛІНА", "ЕМІЛІЯ",

"ЄВА", "ЄЛИЗАВЕТА", "ЗЛАТА", "ІВАННА", "ІЛОНА", "ІРИНА", "КАРІНА",

"КАРОЛІНА", "КАТЕРИНА", "ЛІЛІЯ", "МАРГАРИТА", "МАРІЯ", "МАРТА",

"МАР'ЯНА", "МІЛАНА", "НАДІЯ", "НАТАЛІЯ", "ОКСАНА", "ОЛЕКСАНДРА",

"ОЛЕНА", "ОЛЕСЯ", "ОЛЬГА", "ПОЛІНА", "РОКСОЛАНА", "СОЛОМІЯ",

"СОФІЯ", "ТЕТЯНА", "УЛЯНА", "ХРИСТИНА", "ЮЛІАНА", "ЮЛІЯ", "ЮСТИНА",

"ЯНА", "ЯРИНА" };

static String[] BIRTHDAY = { "01.01.1999" };

static String[] EYES = { "Зелені" };

static String[] GENDER = { "Чоловік", "Жінка" };

static String[] HOBBIES = { "Спів", "Щось", "Щось", "Щось", "Щось", };

static String[] REQUIREMENTS = { "Щось", "Щось", "Щось", "Щось", "Щось", };

/\*\*

\* Створює нового клієнта бюро знайомств.

\*

\* @return client

\*/

public static Client build() {

final Client client = new Client();

final Date today = new Date();

final SimpleDateFormat ft = new SimpleDateFormat("dd.MM.yyyy");

try {

System.out.format("\nВведіть стать (Чоловік/Жінка).");

final String gender = DialogHelper.getInput();

if (check(REGEX.gender, gender)) {

client.setGender(gender);

} else {

return null;

}

System.out.format("\nВведіть ім'я.");

final String name = DialogHelper.getInput();

if (check(REGEX.name, name)) {

client.setName(name);

} else {

return null;

}

System.out.format("\nВведіть зріст.");

final String height = DialogHelper.getInput();

if (check(REGEX.height, height)) {

client.setHeight(Integer.parseInt(height));

} else {

return null;

}

System.out.format("\nВведіть колір очей.");

final String eyes = DialogHelper.getInput();

if (check(REGEX.string, eyes)) {

client.setEyes(eyes);

} else {

return null;

}

System.out.format("\nВведіть дату народження"

+ " у форматі dd.MM.yyyy.");

final String birthday = DialogHelper.getInput();

if (check(REGEX.date, birthday)) {

client.setBirthday(birthday);

} else {

return null;

}

System.out.format("\nВведіть хобі через \";\"");

final String hobby = DialogHelper.getInput();

final String[] hobbies = hobby.split(";");

if (checkArray(hobbies)) {

client.setHobbies(hobbies);

} else {

return null;

}

System.out.format("\nВведіть вимоги до партнера через \";\"");

final String reqs = DialogHelper.getInput();

final String[] requirements = reqs.split(";");

if (checkArray(requirements)) {

client.setRequirements(requirements);

} else {

return null;

}

final String regDate = ft.format(today);

if (check(REGEX.date, regDate)) {

client.setRegDate(regDate);

} else {

return null;

}

} catch (final Exception e) {

System.out.println(e.getMessage());

}

return client;

}

/\*\*

\* Генерує нове бюро знайомств.

\*

\* @param size

\* кількість клієнтів

\* @return bureau бюро знайомств

\*/

public static LinkedList<Client> buildClients(int size) {

final LinkedList<Client> clients = new LinkedList<>();

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

final Client temp = ClientUtil.buildTest();

temp.setRegNum(i);

clients.addLast(temp);

}

return clients;

}

/\*\*

\* Генерує нового клієнта бюро знайомств.

\*

\* @return client

\*/

public static Client buildTest() {

final Client client = new Client();

final Date today = new Date();

final SimpleDateFormat ft = new SimpleDateFormat("dd.MM.yyyy");

client.setBirthday(BIRTHDAY[randNum(BIRTHDAY.length)]);

client.setEyes(EYES[randNum(EYES.length)]);

client.setGender(GENDER[randNum(GENDER.length)]);

if (client.getGender().equals("Чоловік")) {

client.setName(MALE\_NAMES[randNum(MALE\_NAMES.length)]);

} else {

client.setName(FEMALE\_NAMES[randNum(FEMALE\_NAMES.length)]);

}

client.setHeight(randNum(MAX\_HEIGHT - MIN\_HEIGHT, MIN\_HEIGHT));

final String hobbies = HOBBIES[randNum(HOBBIES.length)];

final String[] hobby = hobbies.split(";");

client.setHobbies(hobby);

client.setRegDate(ft.format(today));

final String reqs = REQUIREMENTS[randNum(REQUIREMENTS.length)];

final String[] requirements = reqs.split(";");

client.setRequirements(requirements);

return client;

}

/\*\*

\* @param key

\* @param data

\* @return

\*/

public static boolean check(REGEX key, String data) {

final Pattern pattern = Pattern.compile(filtrs[key.ordinal()]);

final Matcher matcher = pattern.matcher(data);

return matcher.matches();

}

/\*\*

\* @param array

\* @return

\*/

public static boolean checkArray(String[] array) {

boolean safe = false;

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

if (check(REGEX.string, array[i])) {

safe = true;

} else {

safe = false;

}

}

return safe;

}

/\*\*

\* @param client

\* @return

\*/

public static String clientToData(Client client) {

String data = "";

data += client.getGender() + "#";

data += client.getRegNum() + "#";

data += client.getRegDate() + "#";

data += client.getName() + "#";

data += client.getHeight() + "#";

data += client.getEyes() + "#";

data += client.getBirthday() + "#";

final int hobbCount = client.getHobbies().length;

if (hobbCount > 1) {

for (int i = 0; i < hobbCount - 1; i++) {

data += client.getHobbies()[i] + ",";

}

}

data += client.getHobbies()[hobbCount - 1] + "#";

final int reqsCount = client.getRequirements().length;

if (reqsCount > 1) {

for (int i = 0; i < reqsCount - 1; i++) {

data += client.getRequirements()[i] + ",";

}

}

data += client.getRequirements()[reqsCount - 1];

return data;

}

/\*\*

\* стать; реєстраційний номер; дата реєстрації; відомості про себе

\* (довільний набір властивостей: ім’я, зріст, колір очей, дата народження,

\* хобі тощо); вимоги до партнера (довільний набір властивостей).

\*

\* @param client

\* клієнт бюро знайомств

\*

\* @return info

\*/

public static String info(Client client) {

String hobbies = "";

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

hobbies += "" + (i + 1) + ". " + client.getHobbies()[i] + "\n";

}

String requirements = "";

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

requirements += "" + (i + 1) + ". "

+ client.getRequirements()[i] + "\n";

}

final String info = "-----------Client-----------\n" +

"Стать: " + client.getGender() + "\n" +

"Реєстраційний номер: " + client.getRegNum() + "\n" +

"Дата реєстрації: " + client.getRegDate() + "\n" +

"------------Info------------\n" +

"Ім’я: " + client.getName() + "\n" +

"Зріст: " + client.getHeight() + "\n" +

"Колір очей: " + client.getEyes() + "\n" +

"Дата народження: " + client.getBirthday() + "\n" +

"Хобі:\n" + hobbies +

"-----------Partner-----------\n" +

"Вимоги до партнера:\n" + requirements +

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

return info;

}

/\*\*

\* @param list

\* @return

\*/

public static Client[] listToArray(LinkedList<Client> list) {

final Client[] array = new Client[list.size()];

int i = 0;

for (final Client client : list) {

array[i++] = client;

}

return array;

}

/\*\*

\* @param data

\* @return

\*/

public static Client parse(String data) {

final String[] temp = data.split("#");

final Client client = new Client();

final String[] hobby = temp[7].split(",");

final String[] reqs = temp[8].split(",");

boolean safe = check(REGEX.gender, temp[0]);

if (safe) {

safe = check(REGEX.date, temp[2]);

if (safe) {

safe = check(REGEX.name, temp[3]);

if (safe) {

safe = check(REGEX.height, temp[4]);

if (safe) {

safe = check(REGEX.string, temp[5]);

if (safe) {

safe = check(REGEX.date, temp[6]);

if (safe) {

safe = checkArray(hobby);

if (safe) {

safe = checkArray(reqs);

}

}

}

}

}

}

}

if (safe) {

client.setGender(temp[0]);

client.setRegNum(Integer.parseInt(temp[1]));

client.setRegDate(temp[2]);

client.setName(temp[3]);

client.setHeight(Integer.parseInt(temp[4]));

client.setEyes(temp[5]);

client.setBirthday(temp[6]);

client.setHobbies(hobby);

client.setRequirements(reqs);

} else {

return null;

}

return client;

}

/\*\*

\* Повертає псевдо-рандомне число.

\*

\* @param border

\* допустима границя чисел

\*

\* @return число типу Integer

\* @see java.util.Random#nextInt(int)

\*/

public static int randNum(int border) {

if (border < 0) {

throw new IllegalArgumentException();

}

final Random rand = new Random();

final int randomNum = rand.nextInt(border);

return randomNum;

}

/\*\*

\* Повертає псевдо-рандомне число.

\*

\* @param border

\* допустима границя чисел

\*

\* @return число типу Integer

\* @see java.util.Random#nextInt(int)

\*/

public static int randNum(int border, int bottom) {

if (border < 0) {

throw new IllegalArgumentException();

}

final Random rand = new Random();

final int randomNum = rand.nextInt(border) + bottom;

return randomNum;

}

/\*\*

\* Сортування контейнеру за допомогою компаратора

\*

\* @param list

\*

\* @param clientComparator

\* компаратор для сортування

\*/

public static void sort(LinkedList<Client> list,

ClientComparator clientComparator) {

final Client[] temp = listToArray(list);

MergeSort.sort(temp, clientComparator);

list.clear();

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

list.addLast(temp[i]);

}

}

/\*\*

\* @param client

\* @return

\*/

@Deprecated

public static boolean validate(Client client) {

boolean safe = false;

safe = check(REGEX.string, client.getName());

safe = check(REGEX.date, client.getBirthday());

safe = check(REGEX.string, client.getEyes());

safe = check(REGEX.string, client.getGender());

safe = check(REGEX.height, Integer.toString(client.getHeight()));

safe = checkArray(client.getHobbies());

safe = checkArray(client.getRequirements());

safe = check(REGEX.date, client.getRegDate());

return safe;

}

}

1. **РЕЗУЛЬТАТ РОБОТИ**

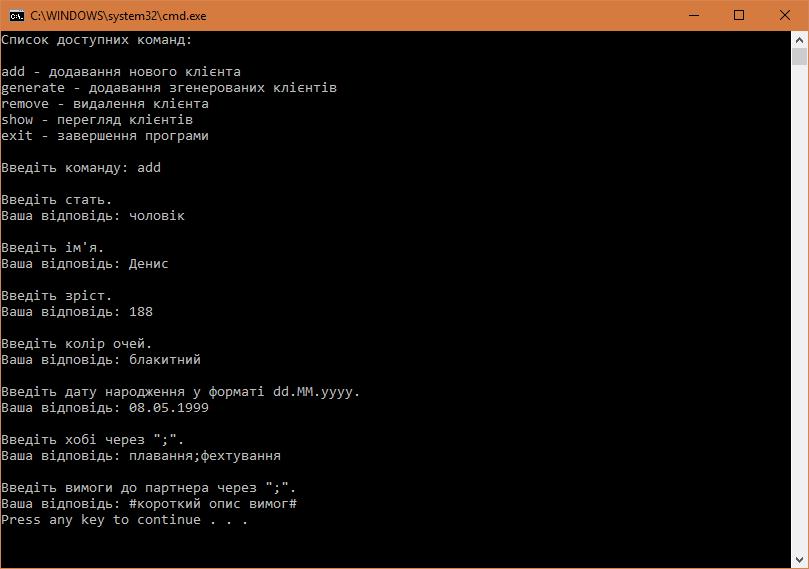
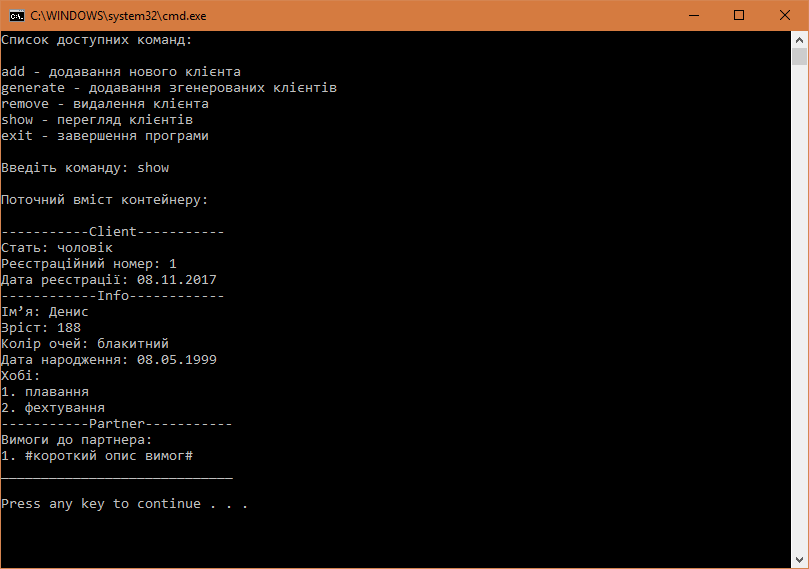
**Для налагодження роботи програми було успішно проведено її тестування.

Рисунок 3 "Результати"

Рисунок 4 "Результати"

**ВИСНОВКИ**

*Створено і налагоджено програму, що повністю виконую поставлене індивідуальне завдання та відповідає вимогам.*

*Було отримано і вдосконалено навички у використанні регулярних виразів для перевірки рядка на відповідність шаблону.*