Лабораторна робота №7.

Об’єктно-орієнтована декомпозиція

**Мета роботи:**

* Використання об’єктно-орієнтованого підходу для розробки об’єкта предметної (прикладної) галузі.

**1. Вимоги до лабораторної роботи**

* 1. **Розробник:**

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

* 1. **Загальне завдання**

1. Використовуючи об’єктно-орієнтований аналіз, реалізувати класи для представлення сутностей відповідно прикладної задачі - domain-об’єктів.
2. Забезпечити та продемонструвати коректне введення та відображення кирилиці.
3. Продемонструвати можливість управління масивом domain-об’єктів.
   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.

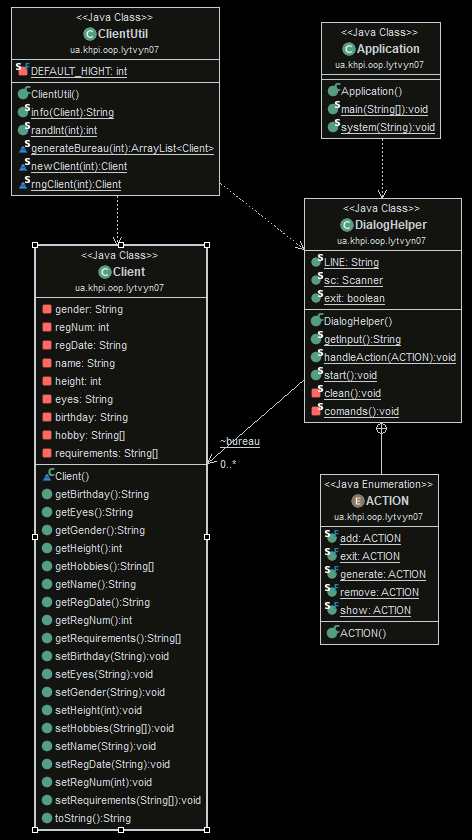
**2.1 Ієрархія та структура класів**

Рисунок 1 « Ієрархія та структура класів»

**2.2 Важливі фрагменти програми**

/\*\*

\* Реалізує клієнта бюро знайомств.

\*

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

\*/

**public** **class** **Client** {

/\*\*

\* Стать

\*/

**private** **String** gender;

/\*\*

\* Реєстраційний номер

\*/

**private** **int** regNum;

/\*\*

\* Дата реєстрації

\*/

**private** **String** regDate;

/\*\*

\* Ім'я

\*/

**private** **String** name;

/\*\*

\* Зріст

\*/

**private** **int** height;

/\*\*

\* Колір очей

\*/

**private** **String** eyes;

/\*\*

\* День народження

\*/

**private** **String** birthday;

/\*\*

\* Хобі

\*/

**private** **String**[] hobby;

/\*\*

\* Вимоги до партнера

\*/

**private** **String**[] requirements;

/\*\*

\* Конструктор за замовчуванням

\*/

**Client**() {

gender = **null**;

regNum = 0;

regDate = **null**;

name = **null**;

height = 0;

eyes = **null**;

birthday = **null**;

hobby = **null**;

requirements = **null**;

}

/\*\*

\* **@return** the birthday

\*/

**public** **String** **getBirthday**() {

**return** birthday;

}

/\*\*

\* **@return** the eyes

\*/

**public** **String** **getEyes**() {

**return** eyes;

}

/\*\*

\* **@return** the gender

\*/

**public** **String** **getGender**() {

**return** gender;

}

/\*\*

\* **@return** the height

\*/

**public** **int** **getHeight**() {

**return** height;

}

/\*\*

\* **@return** the hobbies

\*/

**public** **String**[] **getHobbies**() {

**return** hobby;

}

/\*\*

\* **@return** the name

\*/

**public** **String** **getName**() {

**return** name;

}

/\*\*

\* **@return** the regDate

\*/

**public** **String** **getRegDate**() {

**return** regDate;

}

/\*\*

\* **@return** the regNum

\*/

**public** **int** **getRegNum**() {

**return** regNum;

}

/\*\*

\* **@return** the requirements

\*/

**public** **String**[] **getRequirements**() {

**return** requirements;

}

/\*\*

\* **@param** birthday

\* the birthday to set

\*/

**public** **void** **setBirthday**(**String** birthday) {

**if** (birthday == **null** || birthday.equals("")) {

**throw** **new** IllegalArgumentException(birthday);

}

**this**.birthday = birthday;

}

/\*\*

\* **@param** eyes

\* the eyes to set

\*/

**public** **void** **setEyes**(**String** eyes) {

**if** (eyes == **null** || eyes.equals("")) {

**throw** **new** IllegalArgumentException(eyes);

}

**this**.eyes = eyes;

}

/\*\*

\* **@param** gender

\* the gender to set

\*/

**public** **void** **setGender**(**String** gender) {

**if** (gender == **null** || gender.equals("")) {

**throw** **new** IllegalArgumentException(eyes);

}

**this**.gender = gender;

}

/\*\*

\* **@param** height

\* the height to set

\*/

**public** **void** **setHeight**(**int** height) {

**if** (height <= 0) {

**throw** **new** IllegalArgumentException("" + height);

}

**this**.height = height;

}

/\*\*

\* **@param** hobby

\* the hobbies to set

\*/

**public** **void** **setHobbies**(**String**[] hobby) {

**if** (hobby.length == 0) {

**throw** **new** IllegalArgumentException(hobby.toString());

}

**this**.hobby = hobby;

}

/\*\*

\* **@param** name

\* the name to set

\*/

**public** **void** **setName**(**String** name) {

**if** (name == **null** || name.equals("")) {

**throw** **new** IllegalArgumentException(name);

}

**this**.name = name;

}

/\*\*

\* **@param** regDate

\* the regDate to set

\*/

**public** **void** **setRegDate**(**String** regDate) {

**if** (regDate == **null** || regDate.equals("")) {

**throw** **new** IllegalArgumentException(regDate);

}

**this**.regDate = regDate;

}

/\*\*

\* **@param** regNum

\* the regNum to set

\*/

**public** **void** **setRegNum**(**int** regNum) {

**if** (regNum <= 0) {

**throw** **new** IllegalArgumentException("" + regNum);

}

**this**.regNum = regNum;

}

/\*\*

\* **@param** requirements

\* the requirements to set

\*/

**public** **void** **setRequirements**(**String**[] requirements) {

**if** (requirements.length == 0) {

**throw** **new** IllegalArgumentException(requirements.toString());

}

**this**.requirements = requirements;

}

/\*

\* (non-Javadoc)

\*

\* @see java.lang.Object#toString()

\*/

***@Override***

**public** **String** **toString**() {

**return** **super**.toString();

}

}

1. **РЕЗУЛЬТАТ РОБОТИ**

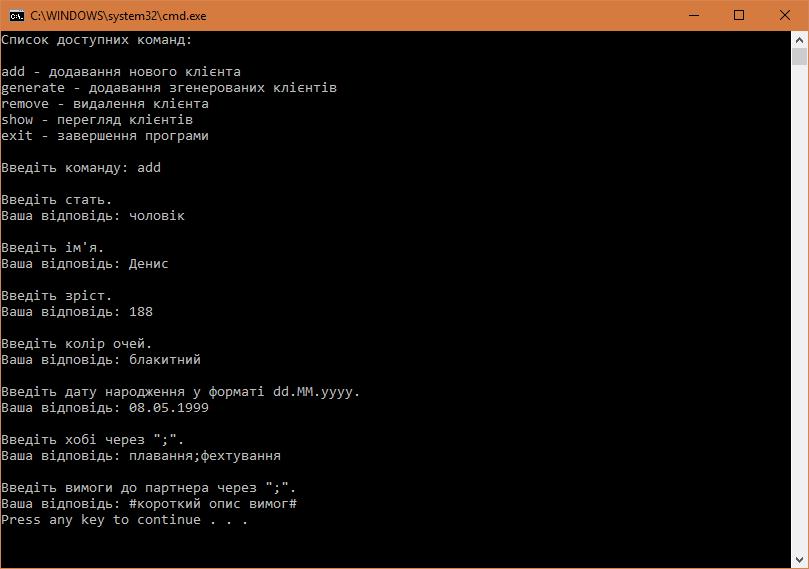
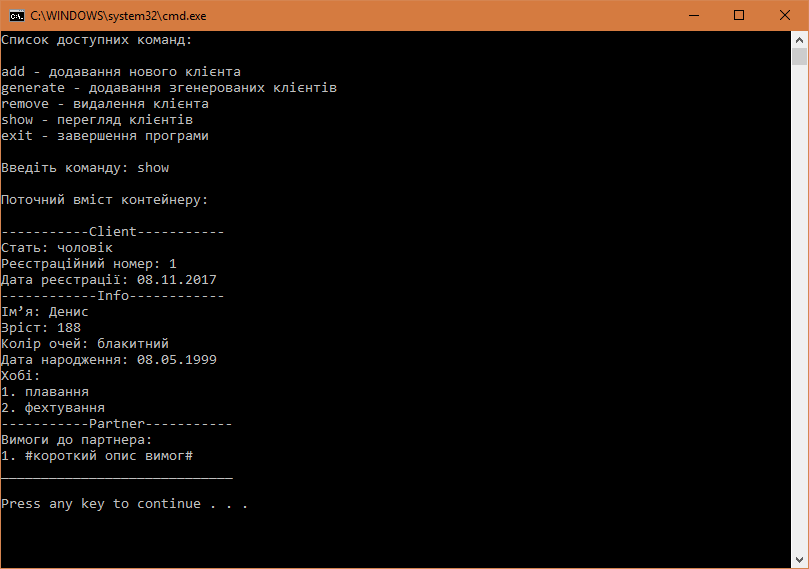
**Для налагодження роботи програми було успішно проведено її тестування.

Рисунок 2 "Результати"

Рисунок 2 "Результати"

**ВИСНОВКИ**

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

*Було отримано і вдосконалено навички у використанні об’єктно-орієнтованого підходу для розробки об’єкта предметної (прикладної) галузі.*