Лабораторна робота №10.

Об’єктно-орієнтована декомпозиція.
Основи введення/виведення Java SE

Мета

Використання об’єктно-орієнтованого підходу для розробки об’єкта предметної (прикладної) галузі. Оволодіння навичками управління введенням/виведенням даних з використанням класів Java SE.

Вимоги

1. Використовуючи об’єктно-орієнтований аналіз, реалізувати класи для представлення сутностей відповідно списку прикладних задач - domain-об’єктів (Прикладні задачі. Список №2. 20 варіантів)
2. Забезпечити та продемонструвати коректне введення та відображення кирилиці.
3. Продемонструвати можливість управління масивом domain-об’єктів.
4. Забезпечити можливість збереження і відновлення масива об’єктів рішення завдання з Прикладні задачі. Список №2. 20 варіантів .
5. Забороняється використання стандартного протокола серіалізації . docs.oracle.com/javase/8/docs/platform/serialization/spec/serialTOC.html
6. Продемонструвати використання моделі Long Term Persistence . docs.oracle.com/javase/tutorial/javabeans/advanced/longpersistence.html
7. Забезпечити діалог з користувачем у вигляді текстового меню.
8. При збереженні та відновленні даних забезпечити діалоговий режим вибору директорії з відображенням вмісту і можливістю переміщення по підкаталогах.

Рекомендації

1. Wiki: Decomposition .
2. Wiki: Plain old Java object .
3. Wiki: JavaBeans .
4. Wiki: Доменный объект .
5. Wiki: Object-oriented analysis and design .
6. Java Tutorials: Basic I/O .
7. Java SE API Specification: Package java.io .
8. Java IO Tutorial .
9. Java SE API Specification: Package java.beans .
10. Java Tutorials: JavaBeans Long Term Persistence .

1.1 Виконала:

студентка групи КН-108 Бокшо К.Е.

1.2 Загальне завдання

1) Забезпечити можливість збереження і відновлення масива об’єктів рішення завдання лабораторної роботи №10.

2) Забороняється використання стандартного протоколу серіалізації.
3) Продемонструвати використання моделі Long Term Persistence.
4) Забезпечити діалог з користувачем у вигляді простого текстового меню.
5) При збереженні та відновленні даних забезпечити діалоговий режим вибору директорії з відображенням вмісту і можливістю переміщення по підкаталогах.

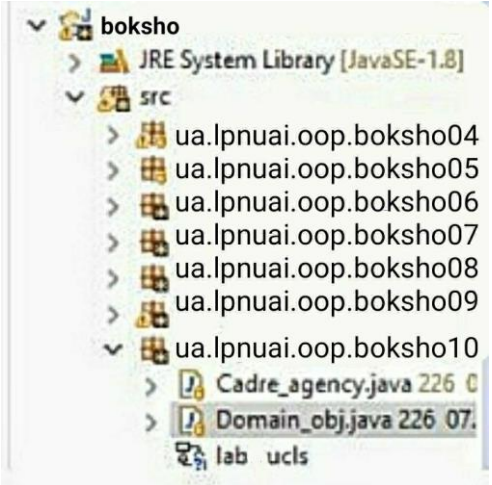
1.3 Прикладна задача

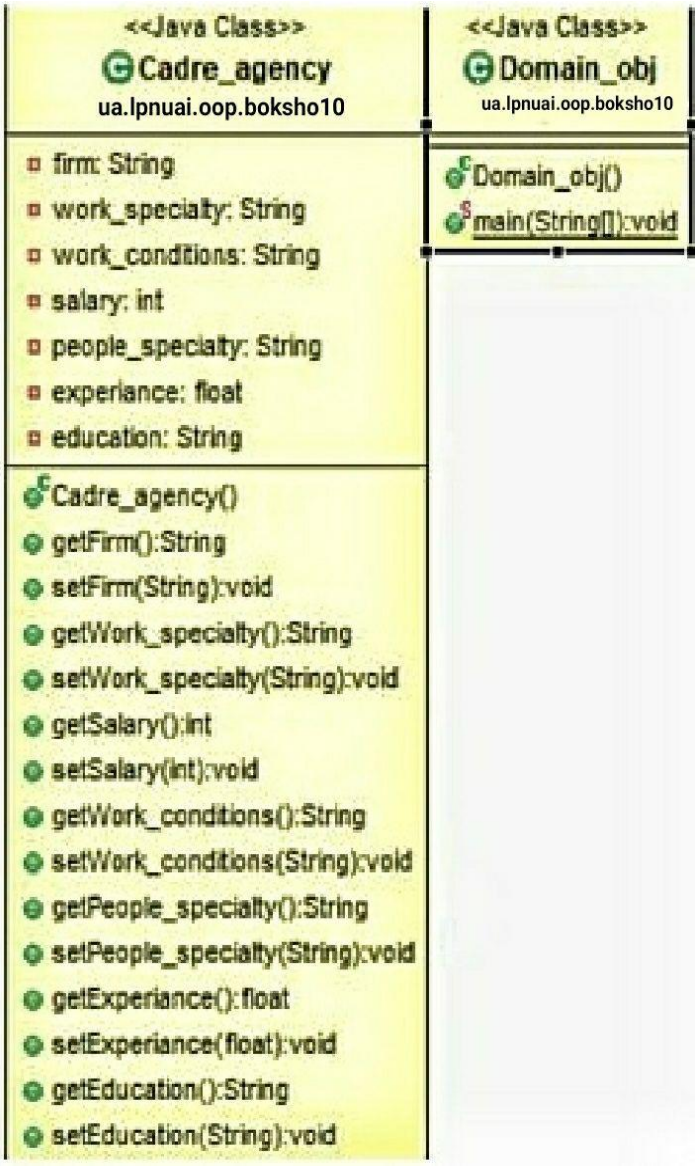
Бюро знайомств. Запис про клієнта: стать; реєстраційний номер; дата реєстрації; відомості про себе (довільний набір властивостей: ім’я, зріст, колір очей, дата народження, хобі тощо); вимоги до партнера (довільний набір властивостей).

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





Важливі фрагменти програми

```
/**
 * Реалізує клієнта бюро знайомств.
 *
 */
public class Client {

    /**
     * Стать
     */
    private String gender;
    /**
     * Реєстраційний номер
     */
    private int regNum;
    /**
     * Дата реєстрації
     */
    private String regDate;
    /**
     * Ім'я
```

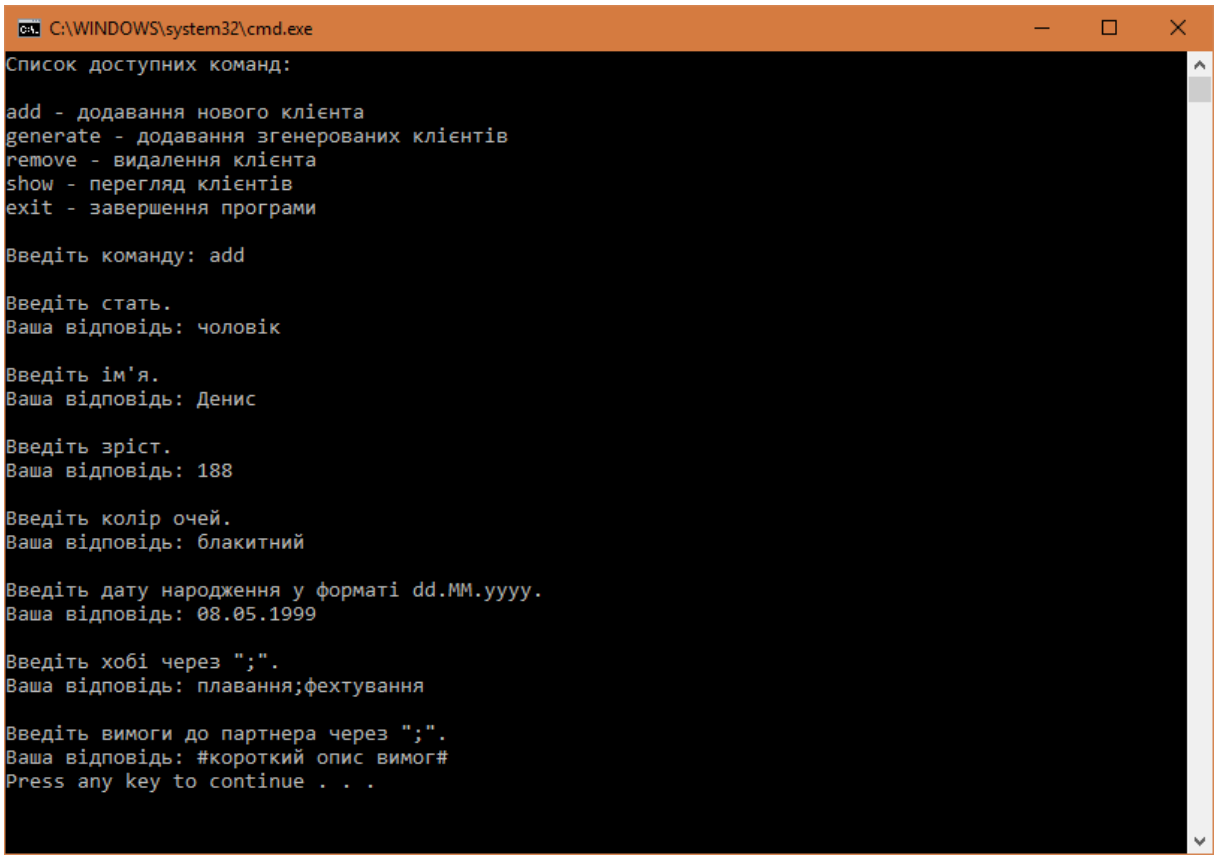
```
 */
private String name;
/**
 * Зріст
 */
private int height;
/**
 * Колір очей
 */
private String eyes;
/**
 * День народження
 */
private String birthday;
/**
 * Хобі
 */
private String[] hobby;
/**
 * Вимоги до партнера
 */
private String[] requirements;

/**
 * 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;
}
```

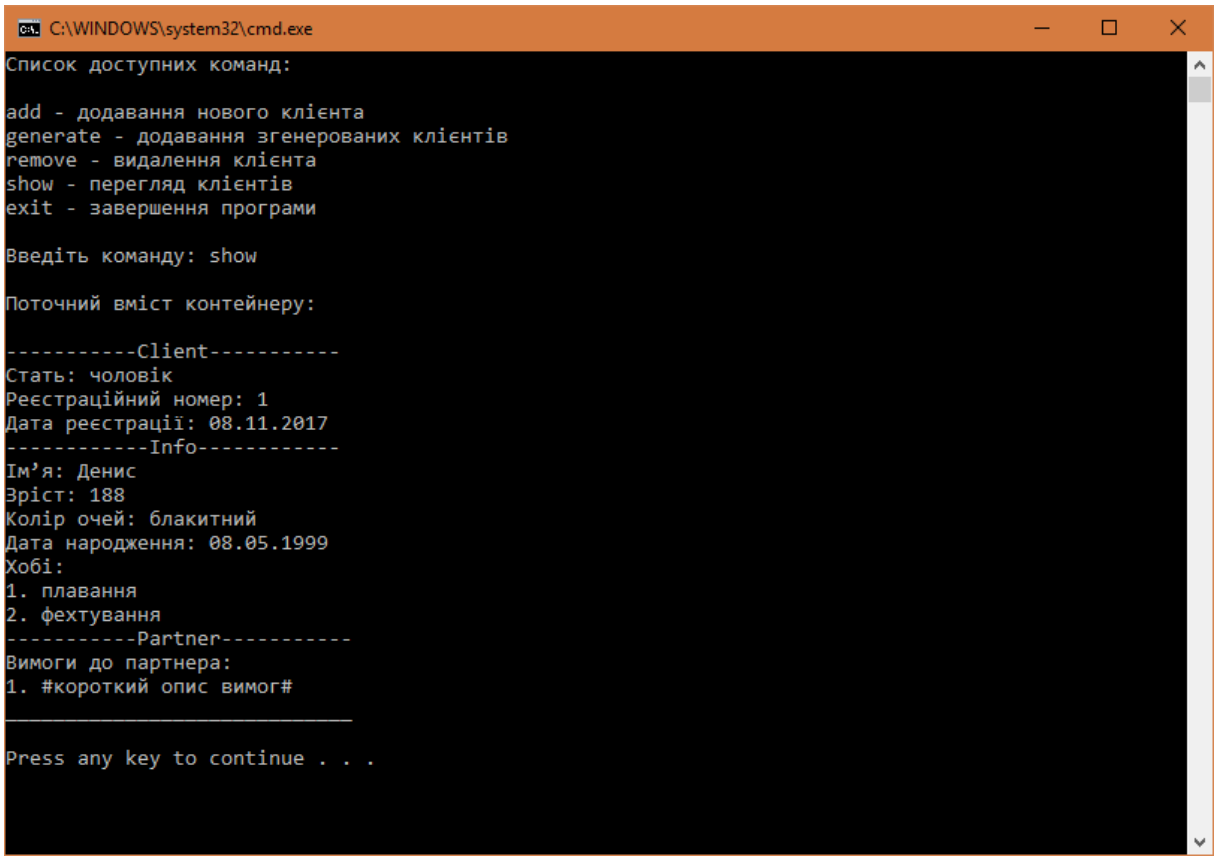
```
/**
 * @return the height
 */
public int getHeight() {
	return height;
}

/**
 * @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
 */
```

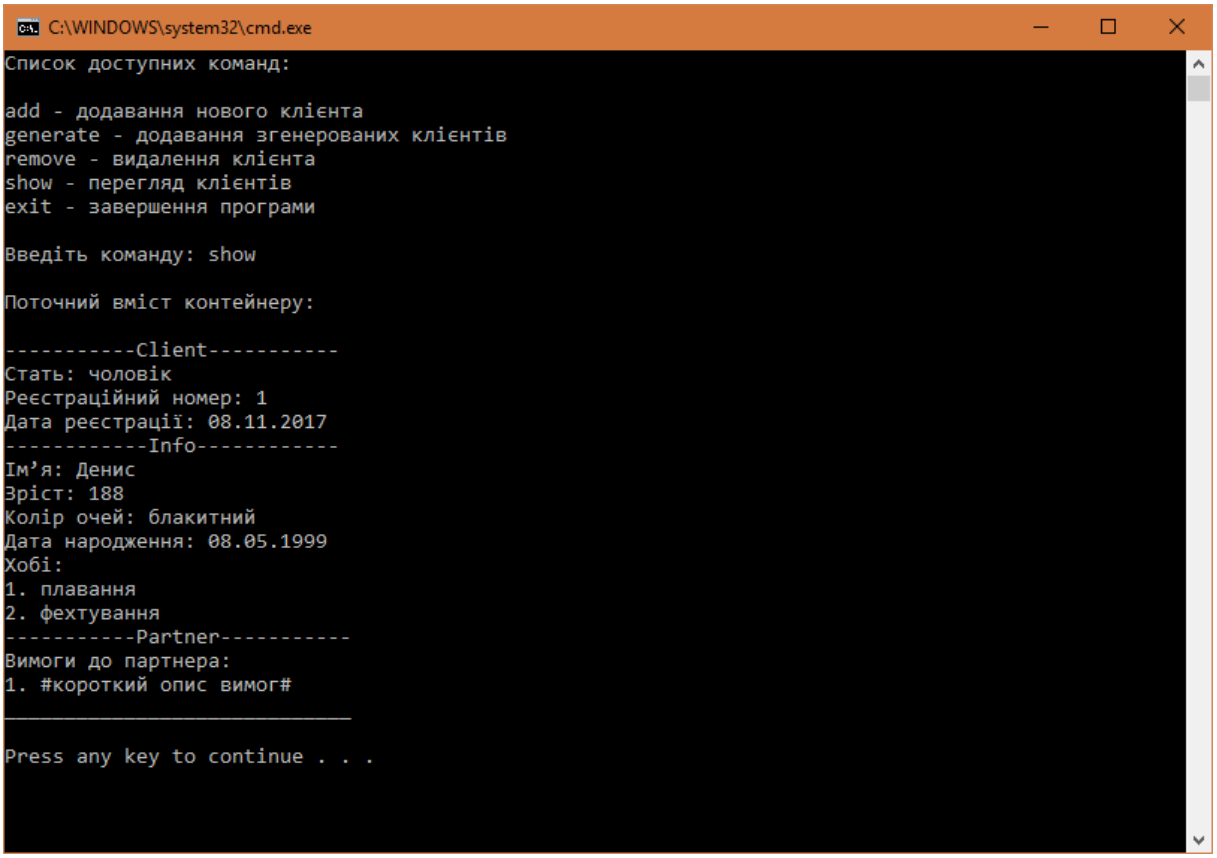
```
public void setEyes(String eyes) {
	if (eyes == null || eyes.equals("")) {
		throw new IllegalArgumentException(eyes);
	}
	this.eyes = eyes;
}

/**
 * @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();
	}
}
```

3. РЕЗУЛЬТАТ РОБОТИ







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