

Міністерство освіти і науки України
Національний університет «Львівська політехніка»

Кафедра «ЕОМ»



Звіт

до лабораторної роботи № 2

з дисципліни: «Кросплатформні засоби програмування»

На тему: «Класи та пакети»

Виконав:

студент групи КІ-307

Возний А. О.

Перевірив:

доцент кафедри ЕОМ

Іванов Ю. С.

Львів – 2023

Мета роботи: ознайомитися з процесом розробки класів та пакетів мовою Java.

Завдання:

1. Написати та налагодити програму на мові Java, що реалізує у вигляді класу предметну область згідно варіанту. Програма має задовольняти наступним вимогам:
 - програма має розміщуватися в пакеті `Група.Прізвище.Lab2`;
 - клас має містити мінімум 3 поля, що є об'єктами класів, які описують складові частини предметної області;
 - клас має містити кілька конструкторів та мінімум 10 методів;
 - для тестування і демонстрації роботи розробленого класу розробити клас-драйвер;
 - методи класу мають вести протокол своєї діяльності, що записується у файл;
 - розробити механізм коректного завершення роботи з файлом (не надіятися на метод `finalize()`);
 - програма має володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету.
2. Автоматично згенерувати документацію до розробленої програми.
3. Завантажити код на GitHub згідно методичних вказівок по роботі з GitHub.
4. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації та завантажити його у ВНС.
5. Дати відповідь на контрольні запитання.

Варіант завдання: Кіт

Лістинг програми:

Клас Cat

```
package ki307.voznyi.lab2;

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;

/**
 * Клас, що представляє інформацію про здоров'я кота.
 */
class CatHealth {

    /**
     * Значення, яке показує, чи є у кота зір.
     */
    private boolean vision;

    /**
     * Значення, яке показує, чи є у кота слух.
     */
    private boolean hearing;

    /**
```

```

    * Кількість лап у kota.
    */
    private int numberOfPaws;

    /**
     * Стан здоров'я kota.
     */
    private String wellBeing;

    /**
     * Конструктор за замовчуванням. Створює об'єкт із заданими значеннями за замовчуванням.
     */
    public CatHealth() {
        this.vision = true;
        this.hearing = true;
        this.numberOfPaws = 4;
        this.wellBeing = "Healthy";
    }

    /**
     * Конструктор, який дозволяє встановити значення параметрів.
     *
     * @param vision    Наявність зіру у kota.
     * @param hearing   Наявність слуху у kota.
     * @param numOfPaws Кількість лап у kota.
     * @param wellBeing Стан здоров'я kota.
     */
    public CatHealth(boolean vision, boolean hearing, int numOfPaws, String wellBeing) {
        this.vision = vision;
        this.hearing = hearing;
        this.numberOfPaws = numOfPaws;
        this.wellBeing = wellBeing;
    }

    /**
     * Метод для перевірки наявності зіру у kota.
     *
     * @return Значення true, якщо у kota є зір, в іншому випадку - false.
     */
    public boolean hasVision() {
        return vision;
    }

    /**
     * Метод для перевірки наявності слуху у kota.
     *
     * @return Значення true, якщо у kota є слух, в іншому випадку - false.
     */
    public boolean hasHearing() {
        return hearing;
    }

    /**
     * Метод для отримання кількості лап у kota.
     *
     * @return Кількість лап у kota.
     */
    public int getNumberOfPaws() {
        return numberOfPaws;
    }
}

```

```

/**
 * Метод для отримання стану здоров'я kota.
 *
 * @return Стан здоров'я kota.
 */
public String getWellBeing() {
    return wellBeing;
}

/**
 * Метод для оновлення інформації про здоров'я kota.
 *
 * @param vision      Наявність зіру у kota.
 * @param hearing     Наявність слуху у kota.
 * @param numOfPaws   Кількість лап у kota.
 * @param wellBeing   Стан здоров'я kota.
 */
public void updateHealth(boolean vision, boolean hearing, int numOfPaws, String wellBeing)
{
    this.vision = vision;
    this.hearing = hearing;
    this.numberOfPaws = numOfPaws;
    this.wellBeing = wellBeing;
    logActivity("Updated health information.");
}

/**
 * Метод для виведення інформації про здоров'я kota на консоль.
 */
public void displayHealthInfo() {
    System.out.println("Vision: " + hasVision());
    System.out.println("Hearing: " + hasHearing());
    System.out.println("Number of Paws: " + getNumberOfPaws());
    System.out.println("Well-being: " + getWellBeing());
}

/**
 * Приватний метод для записування діяльності у файл журналу.
 *
 * @param activity Діяльність для запису.
 */
private void logActivity(String activity) {
    // реалізація методу logActivity
}
}

/**
 * Клас, що представляє зовнішній вигляд kota, такий як колір очей та окрас шерсті.
 */
class CatAppearance {

    /**
     * Колір очей kota.
     */
    private String eyeColor;

    /**
     * Окрас шерсті kota.
     */

```

```

    */
    private String coatColor;

    /**
     * Конструктор за замовчуванням. Створює об'єкт із заданими значеннями за замовчуванням.
     */
    public CatAppearance() {
        this.eyeColor = "Unknown";
        this.coatColor = "Unknown";
    }

    /**
     * Конструктор, який дозволяє встановити значення параметрів.
     *
     * @param eyeColor    Колір очей кота.
     * @param coatColor   Окрас шерсті кота.
     */
    public CatAppearance(String eyeColor, String coatColor) {
        this.eyeColor = eyeColor;
        this.coatColor = coatColor;
    }

    /**
     * Метод для отримання кольору очей кота.
     *
     * @return Колір очей кота.
     */
    public String getEyeColor() {
        logActivity("Checked eye color.");
        return eyeColor;
    }

    /**
     * Метод для отримання окрасу шерсті кота.
     *
     * @return Окрас шерсті кота.
     */
    public String getCoatColor() {
        logActivity("Checked coat color.");
        return coatColor;
    }

    /**
     * Метод для оновлення інформації про зовнішній вигляд кота.
     *
     * @param newEyeColor    Новий колір очей кота.
     * @param newCoatColor   Новий окрас шерсті кота.
     */
    public void updateAppearance(String newEyeColor, String newCoatColor) {
        this.eyeColor = newEyeColor;
        this.coatColor = newCoatColor;
        logActivity("Updated appearance information.");
    }

    /**
     * Метод для відображення інформації про зовнішній вигляд кота на консоль.
     */
    public void displayAppearanceInfo() {
        System.out.println("Eye Color: " + getEyeColor());
        System.out.println("Coat Color: " + getCoatColor());
    }

```

```

    }

    /**
     * Приватний метод для записування діяльності у файл журналу.
     *
     * @param activity Діяльність для запису.
     */
    private void logActivity(String activity) {
        // реалізація методу logActivity
    }
}

/**
 * Клас, що представляє kota з визначеним ім'ям, віком, здоров'ям та зовнішнім виглядом.
 */
public class Cat {

    /**
     * Ім'я kota.
     */
    private String name;

    /**
     * Вік kota.
     */
    private int age;

    /**
     * Об'єкт, що представляє здоров'я kota.
     */
    private CatHealth catHealth;

    /**
     * Об'єкт, що представляє зовнішній вигляд kota.
     */
    private CatAppearance catAppearance;

    /**
     * Конструктор за замовчуванням. Створює kota з ім'ям "Unknown", віком 0, здоров'ям і
     зовнішнім виглядом за замовчуванням.
     */
    public Cat() {
        this.name = "Unknown";
        this.age = 0;
        this.catHealth = new CatHealth();
        this.catAppearance = new CatAppearance();
    }

    /**
     * Конструктор, який встановлює тільки вік kota.
     *
     * @param age Вік kota.
     */
    public Cat(int age) {
        this.age = age;
    }

    /**
     * Конструктор з параметрами.

```

```

*
* @param name      Ім'я кота.
* @param age       Вік кота.
* @param catHealth  Здоров'я кота.
* @param catAppearance Зовнішній вигляд кота.
*/
public Cat(String name, int age, CatHealth catHealth, CatAppearance catAppearance) {
    this.name = name;
    this.age = age;
    this.catHealth = catHealth;
    this.catAppearance = catAppearance;
}

/**
 * Метод для зміни імені кота.
 *
 * @param newName Нове ім'я кота.
 */
public void changeName(String newName) {
    this.name = newName;
    logActivity("Name changed to " + newName);
}

/**
 * Метод для виведення інформації про всіх котів віком 5 років у файл.
 */
public void CatList() {
    try (BufferedWriter writer = new BufferedWriter(new FileWriter("catList.log", true))) {
        writer.write(this.name + "\n");
    } catch (IOException e) {
        e.printStackTrace();
    }
}

/**
 * Метод для зміни віку кота.
 *
 * @param newAge Новий вік кота.
 */
public void changeAge(int newAge) {
    this.age = newAge;
    logActivity("Age changed to " + newAge);
}

/**
 * Метод для виведення інформації про кота на консоль.
 */
public void displayInfo() {
    System.out.println("Name: " + name);
    System.out.println("Age: " + age);
    catAppearance.displayAppearanceInfo();
    catHealth.displayHealthInfo();
}

/**
 * Приватний метод для запису діяльності в файл журналу.
 *
 * @param activity Діяльність для запису.
 */
private void logActivity(String activity) {

```

```

        try (BufferedWriter writer = new BufferedWriter(new FileWriter("cat_activity.log",
true))) {
            writer.write(activity + "\n");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    /**
     * Метод, що повертає ім'я кота.
     *
     * @return Ім'я кота.
     */
    public String getName() {
        return name;
    }

    /**
     * Метод для встановлення нового імені кота.
     *
     * @param name Нове ім'я кота.
     */
    public void setName(String name) {
        this.name = name;
        logActivity("Name changed to " + name);
    }

    /**
     * Метод, що повертає вік кота.
     *
     * @return Вік кота.
     */
    public int getAge() {
        return age;
    }

    /**
     * Метод для встановлення нового віку кота.
     *
     * @param age Новий вік кота.
     */
    public void setAge(int age) {
        this.age = age;
        logActivity("Age changed to " + age);
    }

    /**
     * Метод для оновлення інформації про здоров'я кота.
     *
     * @param vision    Наявність зіру у кота.
     * @param hearing   Наявність слуху у кота.
     * @param numOfPaws Кількість лап у кота.
     * @param wellBeing Стан здоров'я кота.
     */
    public void updateHealth(boolean vision, boolean hearing, int numOfPaws, String wellBeing)
    {
        catHealth.updateHealth(vision, hearing, numOfPaws, wellBeing);
        logActivity("Updated health information.");
    }
}

```



```

/**
 * Метод для оновлення інформації про зовнішній вигляд кота.
 *
 * @param eyeColor   Новий колір очей кота.
 * @param coatColor  Новий окрас шерсті кота.
 */
public void updateAppearance(String eyeColor, String coatColor) {
    catAppearance.updateAppearance(eyeColor, coatColor);
    logActivity("Updated appearance information.");
}

/**
 * Метод для виведення інформації про здоров'я кота на консоль.
 */
public void displayHealthInfo() {
    catHealth.displayHealthInfo();
}

/**
 * Метод для виведення інформації про зовнішній вигляд кота на консоль.
 */
public void displayAppearanceInfo() {
    catAppearance.displayAppearanceInfo();
}
}

```

Клас CatApp

```

package ki307.voznyi.lab2;

/**
 * Ця програма виводить в консоль та у файл рисунок згідно до 4-го варіанту
 *
 * @author Voznyi Andrii CI-307
 * @version 1.0
 * @since version 1.0
 */
public class CatApp {
    /**
     * Точка входу в програму. Створює об'єкти котів, відображає їхню інформацію та
     * виконує деякі зміни в інформації про котів.
     *
     * @param args Аргументи командного рядка (не використовуються в цій програмі).
     */
    public static void main(String[] args) {
        // Створення об'єкта кота за допомогою конструктора за замовчуванням
        Cat cat1 = new Cat();
        cat1.displayInfo();

        // Створення об'єкта кота з параметрами
        CatHealth health = new CatHealth(true, true, 4, "Healthy");
        CatAppearance appearance = new CatAppearance("Green", "Striped");
        Cat cat2 = new Cat("Whiskers", 4, health, appearance);
        cat2.displayInfo();

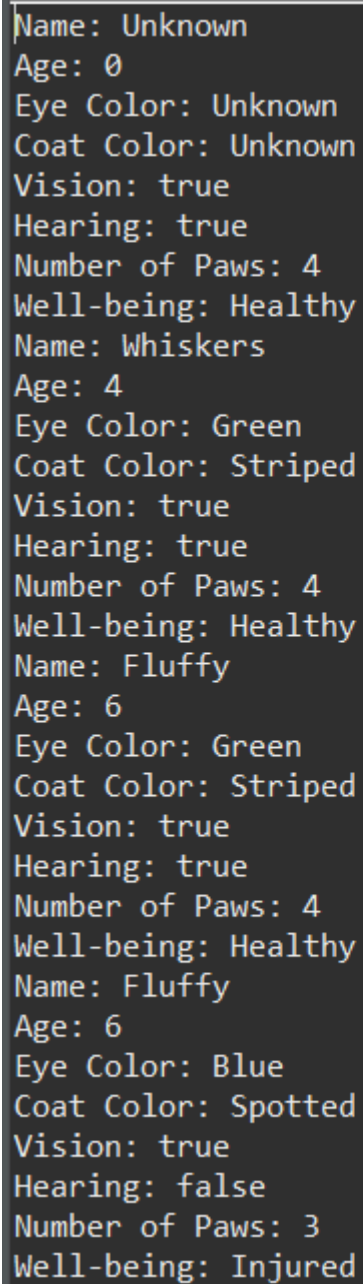
        // Зміна імені та віку кота
    }
}

```

```
cat2.changeName("Fluffy");
cat2.changeAge(6);
cat2.displayInfo();

// Зміна здоров'я та зовнішнього вигляду кота
cat2.updateHealth(true, false, 3, "Injured");
cat2.updateAppearance("Blue", "Spotted");
cat2.displayInfo();
}
}
```

Результат виконання програми:



```
Name: Unknown
Age: 0
Eye Color: Unknown
Coat Color: Unknown
Vision: true
Hearing: true
Number of Paws: 4
Well-being: Healthy
Name: Whiskers
Age: 4
Eye Color: Green
Coat Color: Striped
Vision: true
Hearing: true
Number of Paws: 4
Well-being: Healthy
Name: Fluffy
Age: 6
Eye Color: Green
Coat Color: Striped
Vision: true
Hearing: true
Number of Paws: 4
Well-being: Healthy
Name: Fluffy
Age: 6
Eye Color: Blue
Coat Color: Spotted
Vision: true
Hearing: false
Number of Paws: 3
Well-being: Injured
```

Рис. 1. Результат виконання програми

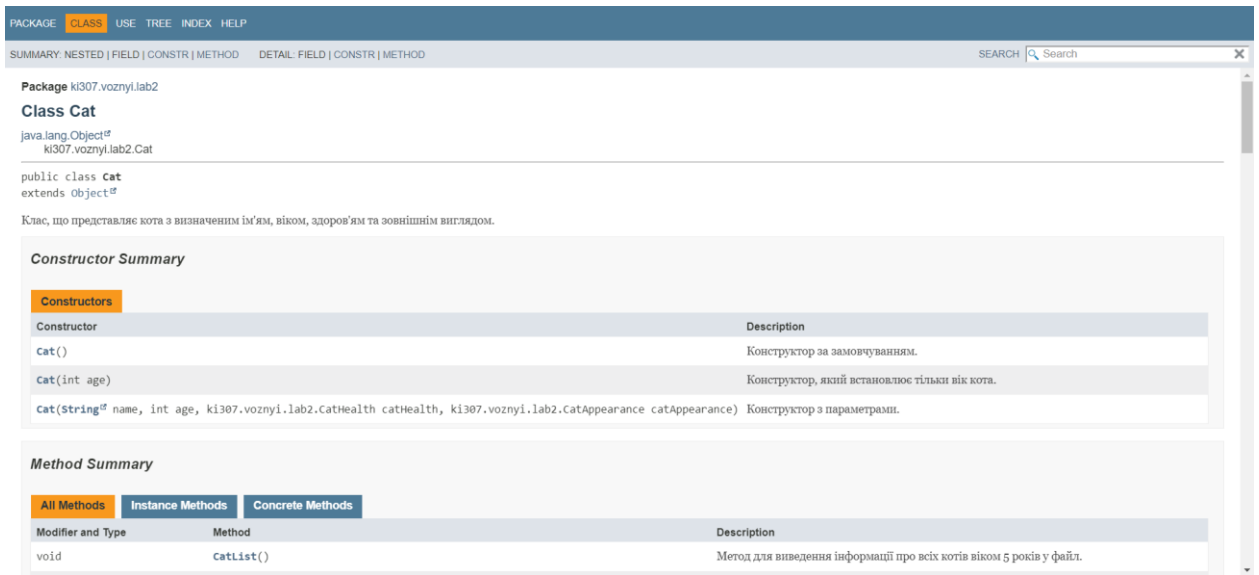


Рис. 2. Згенерована документація до класу Cat

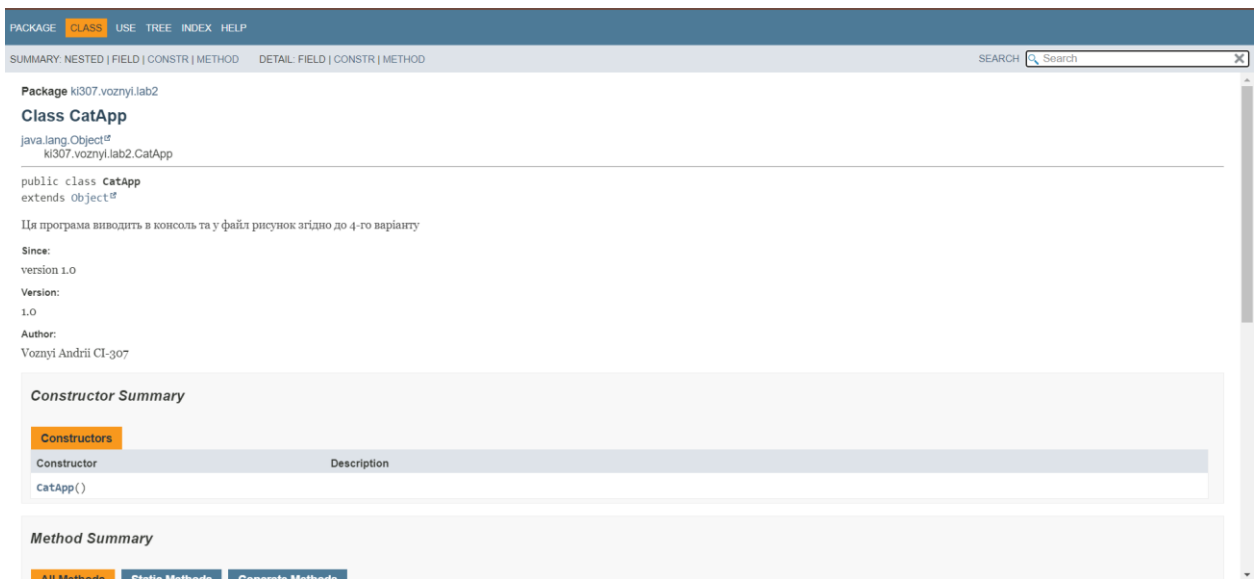


Рис. 3. Згенерована документація до класу CatApp

Відповіді на контрольні питання:

1. Синтаксис визначення класу:

```
public class MyClass {
    // тіло класу
}
```

2. Синтаксис визначення методу:

```
public returnType methodName(parameter1Type parameter1Name,  
parameter2Type parameter2Name) {  
    // тіло методу  
}
```

3. Синтаксис оголошення поля:

```
accessModifier dataType fieldName;
```

4. Як оголосити та ініціалізувати константне поле?

```
public static final int MY_CONSTANT = 42;
```

5. Які є способи ініціалізації полів?

- Через конструктор.
- Прямо при оголошенні поля.
- У блоках ініціалізації.

6. Синтаксис визначення конструктора.

```
public MyClass(parameter1Type parameter1Name, parameter2Type  
parameter2Name) {  
    // тіло конструктора  
}
```

7. Синтаксис оголошення пакету.

```
package mypackage;
```

8. Як підключити до програми класи, що визначені в зовнішніх пакетах?

```
import packageName.ClassName;
```

9. В чому суть статичного імпорту пакетів?

```
import static packageName.ClassName.staticMethod;
```

10. Які вимоги ставляться до файлів і каталогів при використанні пакетів?

- Файли повинні бути розташовані у відповідних каталогах, які відображають структуру пакетів.
- Ім'я файлу повинно відповідати імені класу з урахуванням регістру.

- Внутрішні каталоги пакетів повинні бути частинами шляху до файлу класу.

Висновок: під час виконання цієї лабораторної роботи я ознайомився з процесом розробки класів та пакетів мовою Java.