Міністерство освіти і науки України
Національний університет «Львівська політехніка»

Кафедра «ЕОМ»

Звіт

до лабораторної роботи № 3

з дисципліни: «Кросплатформні засоби програмування»

На тему: «Спадкування та інтерфейси»

Виконав:
студент групи КІ-307
Возний А. О.

Перевірив:
доцент кафедри ЕОМ
Іванов Ю. С.

Львів – 2023

Мета роботи: ознайомитися з спадкуванням та інтерфейсами у мові Java.

Завдання:

1. Написати та налагодити програму на мові Java, що розширює клас, що реалізований у лабораторній роботі №2, для реалізації предметної області заданої варіантом. Суперклас, що реалізований у лабораторній роботі №2, зробити абстрактним. Розроблений підклас має забезпечувати механізми свого коректного функціонування та реалізовувати мінімум один інтерфейс. Програма має розміщуватися в пакеті Група.Прізвище.Lab3 та володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету.

2. Автоматично згенерувати документацію до розробленого пакету.

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: Піддослідний кіт

Лістинг програми:

Клас Cat

```
package ki307.voznyi.lab3;

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 кота.
     */
    private int numberOfPaws;

    /**
     * Стан здоров'я кота.
     */
    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 кота.
     * @param hearing    Наявність слуху у кота.
     * @param numOfPaws  Кількість лап у кота.
     * @param wellBeing  Стан здоров'я кота.
     */
    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 кота.
     *
     * @return Значення true, якщо у кота є зір, в іншому випадку - false.
     */
    public boolean hasVision() {
        return vision;
    }

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

    /**
     * Метод для отримання кількості лап у кота.
     *
```

```
     * @return Кількість лап у кота.
     */
    public int getNumberOfPaws() {
        return numberOfPaws;
    }

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

    /**
     * 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) {
        this.vision = vision;
        this.hearing = hearing;
        this.numberOfPaws = numOfPaws;
        this.wellBeing = wellBeing;
        logActivity("Updated health information.");
    }

    /**
     * Метод для виведення інформації про здоров'я кота 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 кота, такий як колір очей та окрас шерсті.
 */
class CatAppearance {

    /**
     * Колір очей кота.
     */
    private String eyeColor;
```

```java
    /**
     * Окрас шерсті кота.
     */
    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 коліра 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() {
```

```java
        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 кота з визначеним ім'ям, віком, здоров'ям та зовнішнім виглядом.
 */
public class Cat {

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

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

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

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

    /**
     * Конструктор за замовчуванням. Створює кота 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 кота.
     *
     * @param age Вік кота.
     */
    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.
     */
```

```java
    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 HouseCat

```
package ki307.voznyi.lab3;

interface Pet {
    void play();
}

interface Animal {
    void eat();
}

/**
 * Клас HouseCat представляє конкретний тип кота, який може бути домашнім улюбленцем
та твариною.
 * Він розширює клас Cat та реалізує інтерфейси Pet та Animal.
 *
 * @author Ваше Ім'я
 * @version 1.0
 * @since версія 1.0
 */
public class HouseCat extends Cat implements Pet, Animal {
    private boolean isPurring;

    /**
     * Конструктор за замовчуванням для класу HouseCat.
     * Ініціалізує кота значеннями за замовчуванням і встановлює purring в true.
     */
    public HouseCat() {
        super();
        this.isPurring = true;
    }
```

```java
    /**
     * Параметризований конструктор для класу HouseCat.
     *
     * @param name           Ім'я домашнього кота.
     * @param age            Вік домашнього кота.
     * @param catHealth      Інформація про здоров'я домашнього кота.
     * @param catAppearance  Інформація про зовнішній вигляд домашнього кота.
     * @param isPurring      Статус мурчання домашнього кота.
     */
    public HouseCat(String name, int age, CatHealth catHealth, CatAppearance
catAppearance, boolean isPurring) {
        super(name, age, catHealth, catAppearance);
        this.isPurring = isPurring;
    }

    /**
     * Отримує статус мурчання домашнього кота.
     *
     * @return true, якщо кіт мурчить, false в іншому випадку.
     */
    public boolean isPurring() {
        return isPurring;
    }


    /**
     * Встановлює статус мурчання домашнього кота.
     *
     * @param isPurring true, якщо кіт мурчить, false в іншому випадку.
     */
    public void setPurring(boolean isPurring) {
        this.isPurring = isPurring;
    }


    /**
     * Реалізує поведінку гри для домашнього кота.
     * Якщо кіт мурчить, виводить, що кіт грає і мурчить.
     * В іншому випадку виводить, що кіт грає.
     */
    @Override
    public void play() {
        if (isPurring) {
            System.out.println("The cat is playing and purring.");
        } else {
            System.out.println("The cat is playing.");
        }
    }


    /**
     * Реалізує поведінку їжі для домашнього кота.
     * Виводить, що кіт їсть.
     */
    @Override
    public void eat() {
        System.out.println("The cat is eating.");
    }

    /**
     * Виводить інформацію про домашнього кота, включаючи його статус мурчання.
     */
```

```
    @Override
    public void displayInfo() {
        super.displayInfo();
        System.out.println("Purring: " + isPurring());
    }
}
```

Клас MainClass

```
package ki307.voznyi.lab3;

/**
 * Головний клас для демонстрації функціоналу класів Cat та HouseCat.
 * Виводить інформацію про об'єкти кота та домашнього кота, а також викликає методи
play та eat для домашнього кота.
 */
public class MainClass {
    /**
     * Точка входу в програму.
     * Створює об'єкти кота та домашнього кота, виводить їхню інформацію та викликає
методи play та eat для домашнього кота.
     *
     * @param args Параметри командного рядка (не використовуються в даному випадку).
     */
    public static void main(String[] args) {

       // Створення об'єкта класу Cat за допомогою анонімного класу
        Cat cat = new Cat() {};

        System.out.println("Інформація про кота:");
        cat.displayInfo();
        System.out.println();

        // Створення об'єкта класу HouseCat з параметрами
        HouseCat houseCat = new HouseCat("Fluffy", 2, new CatHealth(true, true, 4,
"Healthy"), new CatAppearance("Blue", "White"), true);

        System.out.println("Інформація про домашнього кота:");
        houseCat.displayInfo();

        // Виклик методів play та eat для домашнього кота
        houseCat.play();
        houseCat.eat();
    }
}
```

Результат виконання програми:

```
Інформація про кота:
Name: Unknown
Age: 0
Eye Color: Unknown
Coat Color: Unknown
Vision: true
Hearing: true
Number of Paws: 4
Well-being: Healthy

Інформація про домашнього кота:
Name: Fluffy
Age: 2
Eye Color: Blue
Coat Color: White
Vision: true
Hearing: true
Number of Paws: 4
Well-being: Healthy
Purring: true
The cat is playing and purring.
The cat is eating.
```

Рис. 1. Результат виконання програми

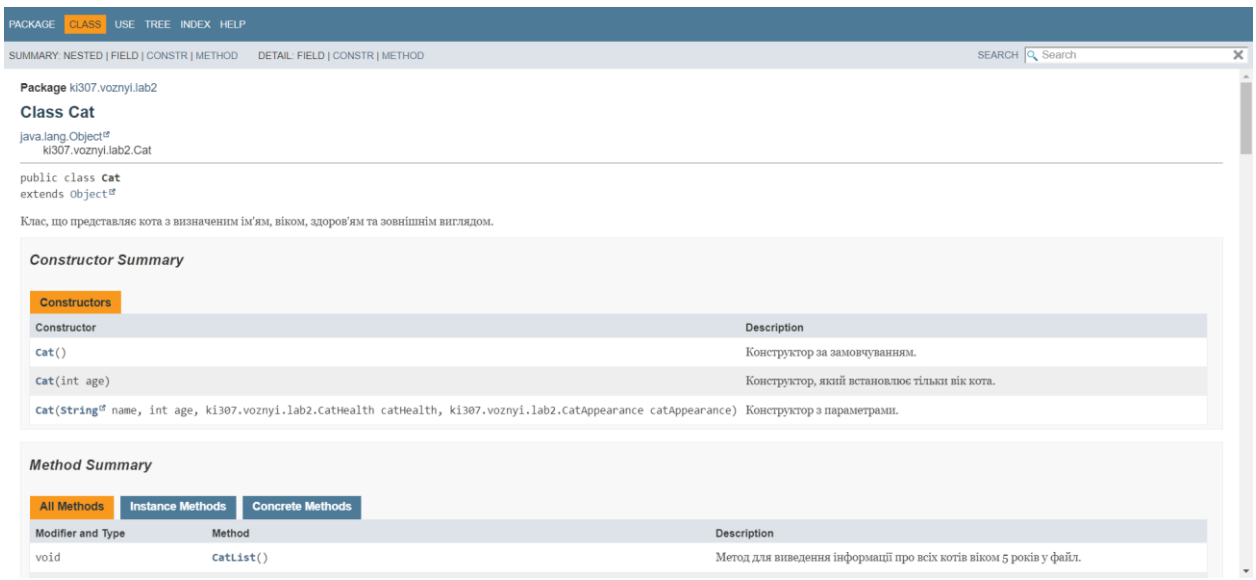

Рис. 2. Згенерована документація до класу Cat

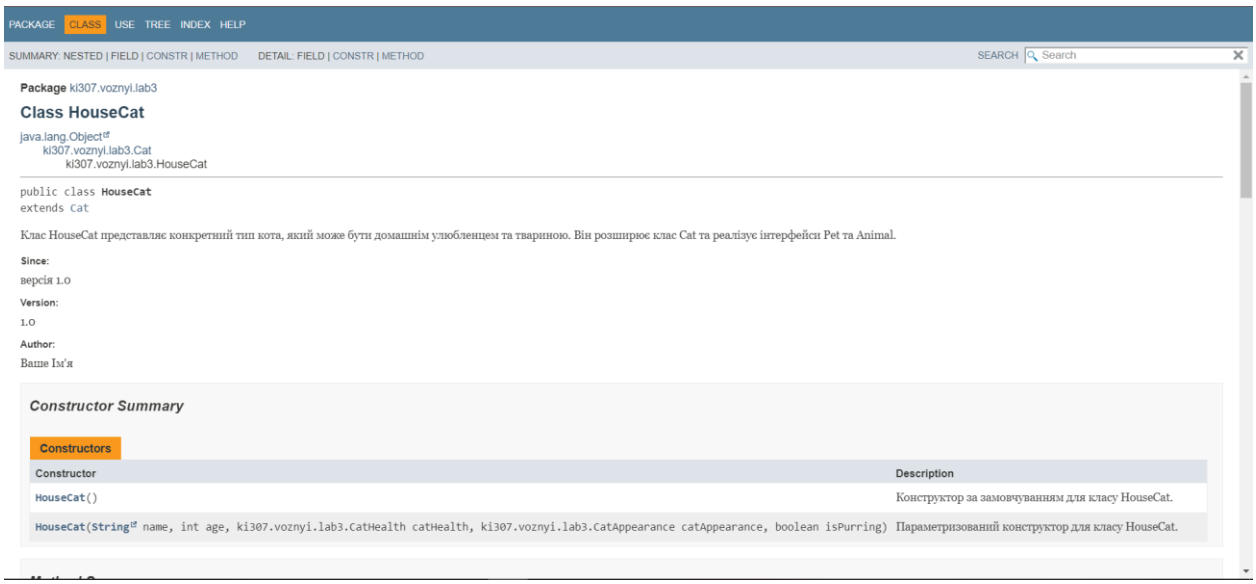

Рис. 3. Згенерована документація до класу HouseCat

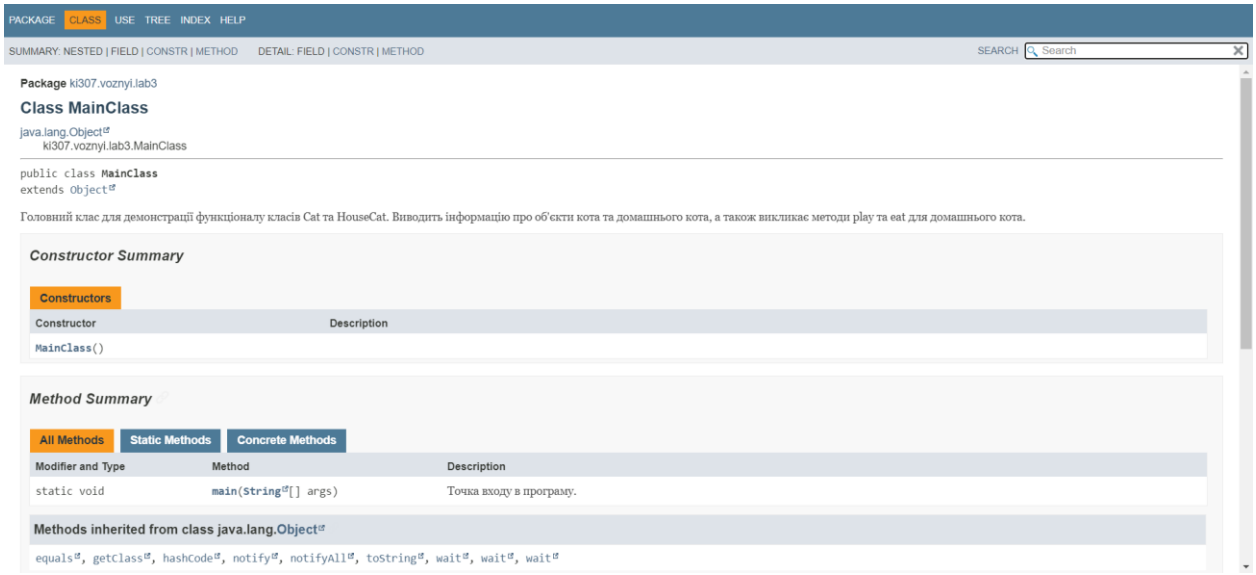

Рис. 4. Згенерована документація до класу MainClass

Відповіді на контрольні питання:

Синтаксис реалізації спадкування.

Синтаксис реалізації спадкування: class ChildClass extends ParentClass { ... }`.

Що таке суперклас та підклас?

Суперклас - це клас, від якого інший клас (підклас) успадковує властивості і методи. Підклас - це клас, який успадковує властивості і методи від суперкласу.

Як звернутися до членів суперкласу з підкласу?

До членів суперкласу з підкласу можна звертатися, використовуючи ключове слово `super`.

Коли використовується статичне зв’язування при виклику методу?

Статичне зв’язування використовується при виклику статичних методів або методів, які компілятор може визначити на етапі компіляції за типом посилання.

Як відбувається динамічне зв’язування при виклику методу?

Динамічне зв’язування відбувається під час виконання програми, коли викликається метод на об'єкті, і вибір методу залежить від типу об'єкта на етапі виконання.

Що таке абстрактний клас та як його реалізувати?

Абстрактний клас - це клас, який не може бути створений безпосередньо, і він містить абстрактні методи. Для його реалізації використовується ключове слово `abstract`.

Для чого використовується ключове слово instanceof?

Ключове слово `instanceof` використовується для перевірки, чи об'єкт є екземпляром певного класу або підкласу.

Як перевірити чи клас є підкласом іншого класу?

Для перевірки, чи клас є підкласом іншого класу, можна використовувати ключове слово `instanceof` або порівняння типів об'єктів.

Що таке інтерфейс?

Інтерфейс - це контракт, який визначає набір методів, які клас повинен реалізувати. Він не містить реалізації методів, тільки їхні сигнатури.

Як оголосити та застосувати інтерфейс?

Для оголошення інтерфейсу використовується ключове слово `interface`, і класи реалізують інтерфейс за допомогою ключового слова `implements`.

Висновок: під час виконання цієї лабораторної роботи я ознайомився зі спадкуванням та інтерфейсами у мові Java.