**Міністерство освіти і науки України**

**Національний університет «Львівська політехніка»**

**Кафедра ЕОМ**

Звіт



**Лабораторна робота № 3**

“ СПАДКУВАННЯ ТА ІНТЕРФЕЙСИ”

з курсу “Кросплатформні засоби програмування”

Варіант: 4

Виконав:

ст.гр.КІ-205

Воробець Тетяна

Прийняв:

доцент кафедри ЕОМ

Олексів М.В.

**Львів 2024**

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

**Теоретичні відомості:** Спадкування в ООП призначене для розширення функціональності існуючих класів шляхом утворення нових класів на базі вже існуючих. У Java реалізована однокоренева архітектура класів згідно якої всі класи мають єдиного спільного предка (кореневий клас в ієрархії класів) – клас Object. Решта класів мови Java утворюються шляхом успадковування даного класу. Будь-яке спадкування у мові Java є відкритим, при цьому аналогів захищеному і приватному спадкуванню мови С++ не існує. На відміну від С++ у Java можливе спадкування лише одного базового класу (множинне спадкування відсутнє). Спадкування реалізується шляхом вказування ключового слова class піcля якого вказується назва підкласу, ключове слово extends та назва суперкласу, що розширюється у новому підкласі.

Механізм поліморфізму забезпечує можливість присвоєння об’єктним змінним суперкласу об’єктів похідних класів та звертання з-під цих змінних до перевизначених у підкласі членів суперкласу. У Java всі об’єктні змінні є поліморфними. Поліморфізм реалізується за допомогою механізму динамічного (пізнього) зв’язування, який полягає у тому, що вибір методу, який необхідно викликати, відбувається не на етапі компіляції, а під час виконання програми.

У Java усі об’єктні змінні є типізовані. Механізми наслідування і поліморфізму дозволяють створювати нові типи (класи та інтерфейси) на базі вже існуючих та присвоювати об’єкти цих типів посиланням на об’єкти супертипу. В цьому випадку об’єкти підтипів мають ті самі елементи, що й об’єкти супертипу, тож таке висхідне приведення типів є безпечним і здійснюється компілятором автоматично. Проте присвоєння посиланню на об’єкт підтипу об’єкту супертипу не завжди є коректним, тому таке приведення вимагає явного приведення типів. При такому приведенні типів можливі дві ситуації:

• якщо посилання на об’єкт супертипу реально посилається на об’єкт підтипу, то приведення посилання на об’єкт супертипу до типу підтипу є коректним;

• якщо посилання на об’єкт супертипу посилається на об’єкт супертипу, то приведення посилання на об’єкт супертипу до типу підтипу викличе виключну ситуацію ClassCastExeption.

Абстрактні класи призначені бути основою для розробки ієрархій класів та не дозволяють створювати об’єкти свого класу. Вони реалізуються за допомогою ключового слова abstract. На відміну від звичайних класів абстрактні класи можуть містити абстрактні методи (а можуть і не містити). Абстрактні методи – це методи, що 6 оголошені з використанням ключового слова abstract і не місять тіла.

Інтерфейси вказують що повинен робити клас не вказуючи як саме він це повинен робити. Інтерфейси покликані компенсувати відсутність множинного спадкування у мові Java та гарантують визначення у класах оголошених у собі прототипів методів.

**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.

**Код програми:**

Experiment.java

package KI\_305.Vorobets.Lab3;

/\*

\* Interface contains methods for working with the main experiment cat

\*/

interface ShredingExperiment {

void Box();

void putPoison();

void outPoison();

}

/\*

\* Interface contains methods for working with the experiment cat

\*/

interface Experiment extends ShredingExperiment{

void checkExperiment();

}

Cat.java

/\*\*

\*

\*/

package KI\_305.Vorobets.Lab3;

import java.io.FileWriter;

import java.io.IOException;

/\*\*

\* Class <code>Cat</code> implements cat

\*/

public abstract class Cat {

protected String name;

protected Breed breed;

protected Master number, nameMaster;

protected Collar collar;

protected int energy;

protected int food;

protected String location;

/\*\*

\* Constructor

\*/

public Cat() {

this.collar = new Collar();

this.breed = new Breed();

this.number = new Master();

this.nameMaster = new Master();

logActivity("Cat's name: None "

+ "Cat's bread: None"+

"Cat's master: None, None"+

"Cat's colar: None");

}

/\*\*

\* Constructor

\* **@param** <code>name</code> name cats

\* **@param** <code>breed</code> breed cats

\* **@param** <code>number</code> number master

\* **@param** <code>nameMaster</code> name master

\* **@param** <code>info\_collar</code> info\_collar

\*/

public Cat(String name, String breed, String number, String nameMaster, boolean info\_collar, String location) {

this.collar = new Collar();

this.breed = new Breed();

this.number = new Master();

this.nameMaster = new Master();

this.name = name;

this.energy = 5;

this.food = 5;

this.location = location;

this.collar.setPresence(info\_collar);

this.breed.setBreed(breed);

this.number.setNumber(number);

this.nameMaster.setNameMaster(nameMaster);

logActivity("Cat's name: "+name

+ ", Cat's bread: "+ breed +

", Number master: "+number + ", Name master: "+nameMaster+

", Cat's colar: " + info\_collar);

}

/\*\*

\* Method gets the name cat

\*/

public String getName() {

return name;

}

/\*\*

\* Method sets the name cat

\* **@param** <code>name</code> name cat

\*/

public void setName(String name) {

this.name = name;

logActivity("New name for cat's: " + name);

}

/\*\*

\* Method action cat play

\* **@param** <code>game</code> game for cat

\*/

public void play(String game) {

if (game == "mouse") {

this.energy = this.energy - 1;

this.food = this.food + 1;

logActivity(name+ " plays with mouse: energy-1, food+1");

}

else if(game == "bug") {

this.energy = this.energy + 1;

this.food = this.food - 1;

this.location = "bugs";

logActivity(name+ " plays in bug: energy+1, food-1");

}

else {

this.energy = this.energy - 2;

this.food = this.food - 1;

this.location = "outside";

logActivity(name+ " plays in outside: energy-2, food-1");

}

}

/\*\*

\* Method action cat sleep

\*/

public void sleep() {

this.energy = this.energy +1;

logActivity(name + " sleeps: energy+1");

}

/\*\*

\* Method action cat clean

\*/

public void clean() {

this.energy = this.energy - 1;

logActivity(name + " cleans: energy-1");

}

/\*\*

\* Method action cat night\_vision

\* **@param** <code>visor</code> on or off night vision

\*/

public void night\_vision(boolean visor) {

this.energy = this.energy - 1;

logActivity(name + " night vision "+visor+": energy-1");

}

/\*\*

\* Method action cat mew

\*/

public abstract void mew();

/\*\*

\* Method location cat

\*/

public void setPlace() {

logActivity(name + " in " + location);

}

/\*\*

\* Method action cat eat

\*/

public void eat(String food) {

if (food == "fish") {

this.food = this.food + 1;

logActivity(name+ " eats fish: food+1");

}

else if(food == "meat") {

this.food = this.food + 2;

logActivity(name+ " eats meat: food+2");

}

else {

this.food = this.food - 1;

logActivity(name+ " don't want to eat candy: food-1");

}

}

/\*\*

\* Method cat status

\*/

public void status() {

if (this.energy == 5 && this.food == 5) {

logActivity("Status: Nice");

}

else if (this.energy < 5 || this.food < 5) {

logActivity("Status: "+name+" need to sleep or eat");

}

else {

logActivity("Status: "+name+" want to play");

}

logActivity("Level energy: " + this.energy+ ", Level food: " + this.food);

}

/\*\*

\* Method information about cat

\*/

public void info\_cat() {

logActivity("Cat's name: "+name

+ ", Cat's bread: "+ breed.getBreed() +

", Number master: "+ number.getNumber() + ", Name master: "+nameMaster.getNameMaster()+

", Cat's colar: " + collar.getPresence() + ", Cat's energy: " + energy);

}

/\*\*

\* Method logging info

\*/

public void logActivity(String message) {

try (FileWriter fw = new FileWriter("cat\_activity.log", true)) {

fw.write(message + "\n");

System.***out***.println(message);

} catch (IOException e) {

System.***err***.println("Error: " + e.getMessage());

}

}

}

/\*\*

\* Class describing a breed of cat's

\*/

class Breed {

private String breed;

/\*\*

\* Method gets the name of breed

\*/

public String getBreed() {

return breed;

}

/\*\*

\* Method sets the name of room

\* **@param** <code>breed</code> name of room

\*/

public void setBreed(String breed) {

this.breed = breed;

}

}

/\*\*

\* Class describing a collar of cat's

\*/

class Collar {

private boolean collar;

/\*\*

\* Method gets the presence of a collar

\*/

public boolean getPresence() {

return collar;

}

/\*\*

\* Method sets the presence of a collar

\* **@param** <code>collar</code> the presence of a collar

\*/

public void setPresence(boolean collar) {

this.collar = collar;

}

}

/\*\*

\* Class describing a master of cat's

\*/

class Master {

private String nameMaster;

private String number;

/\*\*

\* Method sets the presence of a collar

\* **@param** <code>collar</code> the presence of a collar

\*/

public String getNameMaster() {

return nameMaster;

}

/\*\*

\* Method sets the presence of a collar

\* **@param** <code>collar</code> the presence of a collar

\*/

public void setNameMaster(String nameMaster) {

this.nameMaster = nameMaster;

}

/\*\*

\* Method gets the number master

\*/

public String getNumber() {

return number;

}

/\*\*

\* Method sets the number master

\* **@param** <code>number</code> number master

\*/

public void setNumber(String number) {

this.number = number;

}

}

ExperimentCat.java

/\*\*

\*

\*/

package KI\_305.Vorobets.Lab3;

import java.util.Random;

/\*\*

\* ExperimentCat implements Experiment

\*/

public class ExperimentCat extends Cat implements Experiment {

private int poison = 0;

/\*\*

\* Constructor

\*/

public ExperimentCat() {

this.name = null;

this.energy = 0;

this.food = 0;

this.location = null;

this.collar.setPresence(false);

this.breed.setBreed("");

this.number.setNumber("");

this.nameMaster.setNameMaster("");

logActivity("ExperimentCat's name: None "

+ "ExperimentCat's bread: None"+

"ExperimentCat's master: None, None"+

"ExperimentCat's colar: None");

}

/\*\*

\* Constructor

\* **@param** <code>name</code> name cats

\* **@param** <code>breed</code> breed cats

\* **@param** <code>number</code> number master

\* **@param** <code>nameMaster</code> name master

\* **@param** <code>info\_collar</code> info\_collar

\*/

public ExperimentCat(String name, String breed, String number, String nameMaster, boolean info\_collar, String location) {

super(name, breed, number, nameMaster, info\_collar, location);

this.name = name;

this.energy = 5;

this.food = 5;

this.location = location;

this.collar.setPresence(info\_collar);

this.breed.setBreed(breed);

this.number.setNumber(number);

this.nameMaster.setNameMaster(nameMaster);

logActivity("ExperimentCat's name: "+name

+ ", ExperimentCat's bread: "+ breed +

", Number master: "+number + ", Name master: "+nameMaster+

", ExperimentCat's colar: " + info\_collar);

}

/\*\*

\* Method action put cat in the box

\*/

*@Override*

public void Box() {

// **TODO** Auto-generated method stub

this.location = "Box";

logActivity(name + " in " + location);

}

/\*\*

\* Method action put poison

\*/

*@Override*

public void putPoison() {

// **TODO** Auto-generated method stub

poison = 1;

logActivity("Poison in Box");

}

/\*\*

\* Method action out poison

\*/

public void outPoison() {

// **TODO** Auto-generated method stub

poison = 0;

logActivity("Poison out Box");

}

/\*\*

\* Method check result experiment about Shreding Cat`s

\*/

*@Override*

public void checkExperiment() {

// **TODO** Auto-generated method stub

Random random = new Random();

int randomNumber = random.nextInt(2);

if (poison == 1 && location=="Box") {

if (randomNumber == 0) {

logActivity(name + " is dead in experimet");

}

else {

logActivity(name + " is live in experiment");

}

}

else {

logActivity("Conditional for experiment is not done");

}

}

/\*\*

\* Method mew() from abstract class Cat

\*/

public void mew() {

logActivity(name+" said mew");

}

}

ExperimentCatApp.java

/\*\*

\*

\*/

package KI\_305.Vorobets.Lab3;

/\*\*

\* ExperimentCatApp is main Class

\*/

public class ExperimentCatApp {

/\*\*

\* **@param** args

\*/

public static void main(String[] args) {

// **TODO** Auto-generated method stub

ExperimentCat ShredingCat = new ExperimentCat("Shred", "british", "0987654321", "Shreding", true, "room");

ShredingCat.status();

ShredingCat.mew();

ShredingCat.setPlace();

ShredingCat.putPoison();

ShredingCat.checkExperiment();

ShredingCat.Box();

ShredingCat.outPoison();

ShredingCat.checkExperiment();

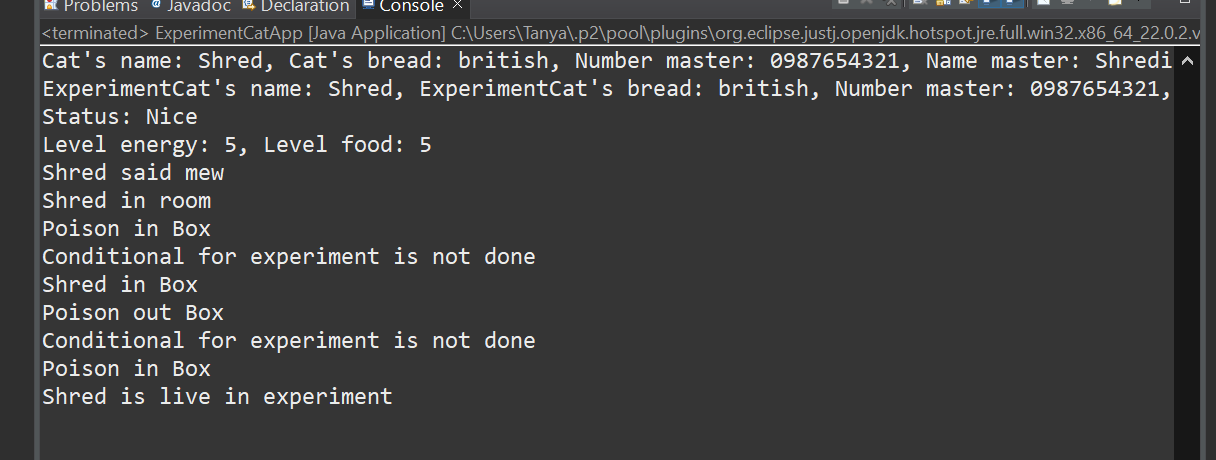
ShredingCat.putPoison();

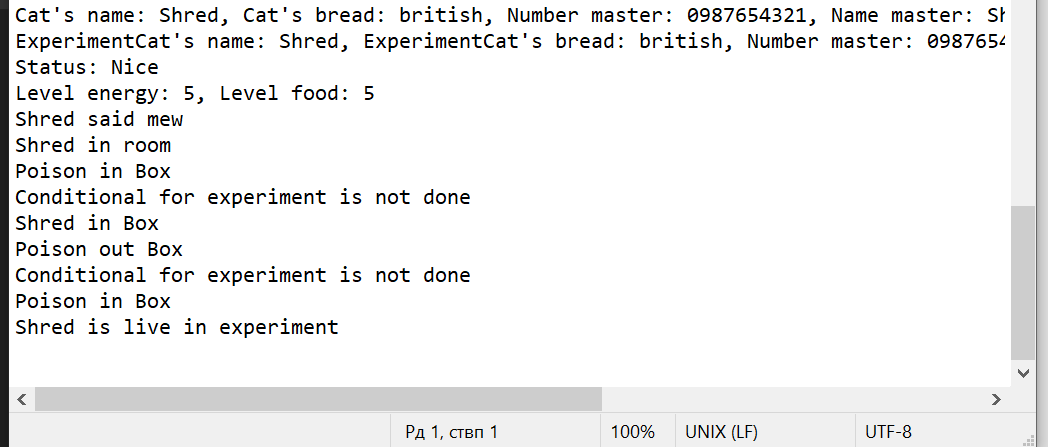
ShredingCat.checkExperiment();

}

}

**Результат:**

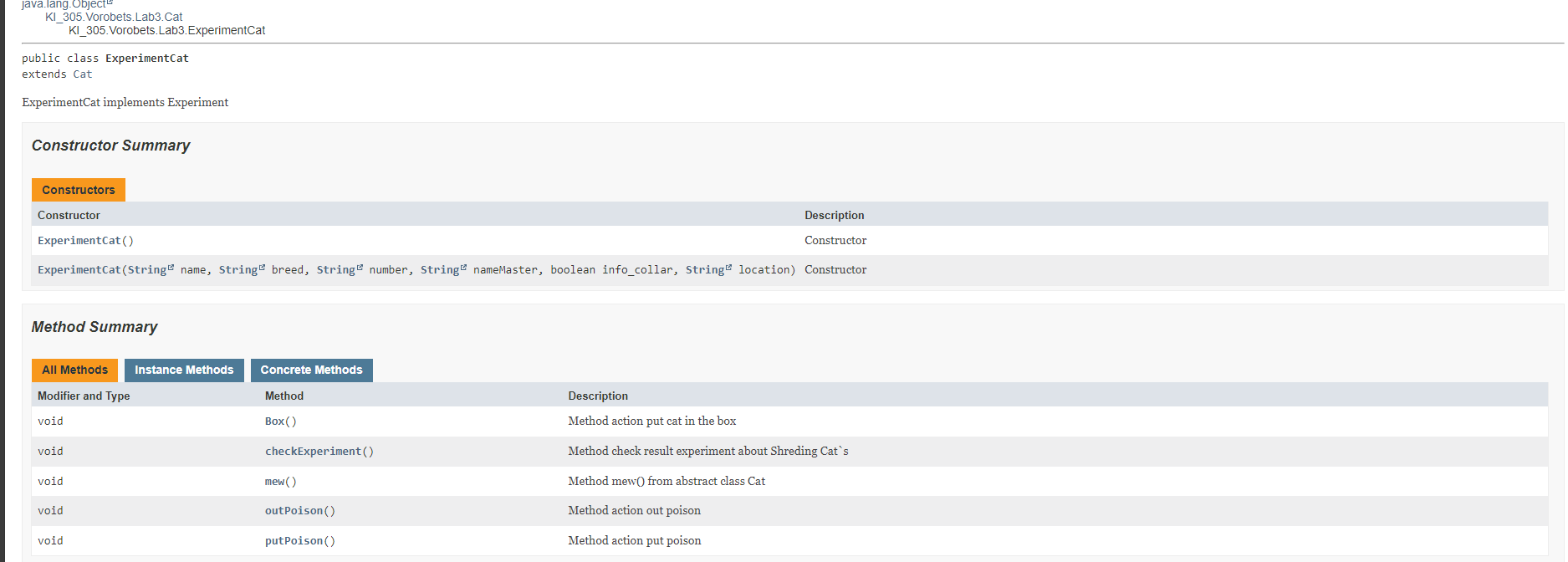
****

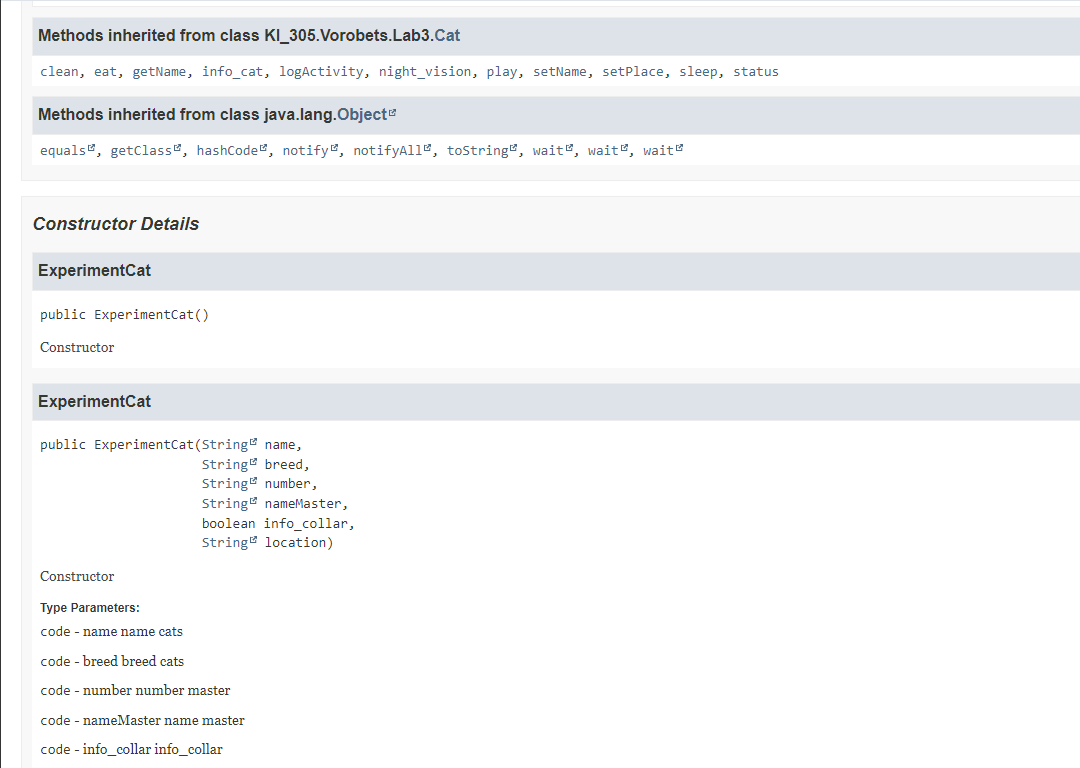
****

**Посилання на репозиторій:**

<https://github.com/NikaDe7/CPPT_Vorobets_TI_KI-35_1.git>

**Документація:**





**Висновок:** ознайомилася з спадкуванням та інтерфейсами у мові Java.