МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

факультет програмної інженерії та бізнесу

кафедра інженерії програмного забезпечення

**Лабораторна робота № 1**

з дисципліни « ООП на Java »

*назва дисципліни*

на тему: «ПРОЕКТУВАННЯ КЛАСІВ ТА СТВОРЕННЯ ОБ'ЄКТІВ ЗАСОБАМИ МОВИ JAVA»

Виконав: студент 2 курсу групи № 622п

освітньої програми

121 інженерія програмного забезпечення

(шифр і назва ОП)

Зайченко Ярослав Ігорович

(прізвище й ініціали студента)

Прийняв: доцент

Симбірський Генадій Дмитрович

(посада, науковий ступінь, прізвище й ініціали)

Кількість балів:

Харків – 2023

**ЗМІСТ**

[Постановка завдання 3](#_Toc158672321)

[Зміст звіту 3](#_Toc158672322)

[Порядок виконання роботи 3](#_Toc158672323)

[Виконання роботи 5](#_Toc158672324)

[Висновок 10](#_Toc158672325)

[Контрольні питання 11](#_Toc158672326)

Мета роботи: отримання практичних навичок із застосування класів та методів при проектуванні об'єктно-орієнтованих програмних додатків мовою Java.

Постановка завдання

Розробити та реалізувати мовою Java додаток для моделювання властивостей, поведінки та взаємодії об'єктів. Предметною областю для цього додатка можуть бути різні об'єкти запропоновані студентами. У додатку продемонструвати використання блоків ініціалізції, конструкторів та методів для зміни поведінки та стану об'єктів.

Зміст звіту

1. Постановка завдання.
2. Встановлення завдань, які вирішуються в лабораторній роботі
3. Опис концептуальної моделі для обраної категорії об’єктів.
4. Відображення структурних відносин між класами за допомогою діаграми класів UML.
5. Реалізація класів.
6. Реалізація класу для тестування додатку.
7. Результат компіляції за допомогою **javac.exe**.
8. Ручне тестування додатку.
9. Результат створення документації на розроблений додаток з допомогою **javadoc.exe.**
10. Висновки по роботі.

Порядок виконання роботи

1. Розробити концептуальну модель для заданої категорії об'єктів у вигляді словесного опису їх основних властивостей та поведінки.
2. Відобразити структурні відносини між спроектованими класами ми за допомогою діаграми класів UML.
3. Реалізувати програмний код класів на мові Java. Для кожного класу створити 2-3 конструктори. Поля класів визначати з модіфікатором private. Доступ до полів класів реалізувати за допомогою методів get і set (геттерів та сеттерів). Для кожного класу реалізувати декілька методів, які моделють поведінку об’єктів та змінюють їх стан.
4. При реалізації програмного коду програми обов'язково використовувати коментарі та дескриптори документування для опису класів, конструкторів, методів та полів.
5. Розробити тестуючий клас для демонстрації створення об'єктів класів та виклику методів, що змінюють їх властивості та поведінку.
6. Відкомпілювати розроблені класи програми за допомогою компілятора javac.exe.
7. Протестувати розроблений додаток.
8. Отримати документацію на розроблений додаток за допомогою утиліти javadoc.exe.
9. Зробити висновки щодо роботи.
10. Оформити звіт з лабораторної роботи відповідно до вимог, що пред'являються.

Завдання для варіанту 10: Створити об'єкт класу Цуценя, використовуючи класи Тварина, Собака. Методи: вивести на консоль ім'я, подати голос, стрибати, бігати, кусати.

Виконання роботи

Загальні відомості

Мова програмування – Java

Операційна система – Windows 11 Pro

Процесор – Apple Silicon M1 Pro with 6 CPU cores

Тип компілятора – IntelliJ IDEA 2024

Концептуальна модель об'єктів

**Клас Тварина**:

Властивості:

* Ім'я (name)

Поведінка:

* Виведення імені на консоль (void printName())
* Подати голос (void makeSound())

**Клас Собака** (наслідується від класу Тварина):

Поведінка:

* Стрибати (void jump())
* Бігати (void run())
* Кусати (void bite())

**Клас Цуценя** (наслідується від класу Собака):

* Зміна реалізації методу для подання звуку
* Метод створення рядка з повною інформацією про об’єкт

**Структурні відносини між спроектованими класами за допомогою діаграми класів UML.**

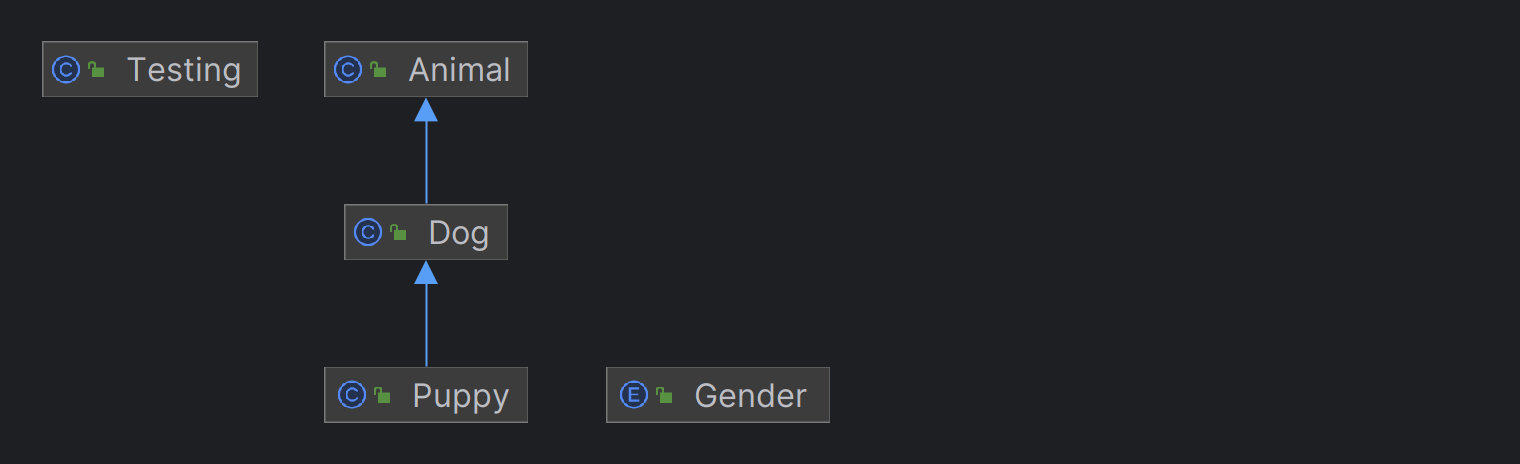


Рисунок 1 – згенерована діаграма у середовищі розробки

Реалізувати програмний код класів на мові Java.

При написанні коду були виконані умови: створення в класах 2 конструкторів, поля класів були створені з модифікатором **private**. Також для закритих полів створено методи гетерів та сетерів (**get/set**), у кожному класі присутні декілька різних методів, які моделюють поведінку об’єктів та змінюють їх стан. Фрагменти фунціоналу коду описані коментарями. Програмний код класів можна переглянути в додатках А-Д.

Клас для тестування

Створений клас демонструє створення різних екземплярів класу **Puppy** з використанням різноманітних конструкторів, та виклику методів класів.

package Classes;

import Classes.Puppy;

import javax.swing.\*;

public class Testing {

public static void StartTest(String puppyName){

// Створення об'єкта класу Цуценя

Puppy TestMYpuppy = new Puppy(puppyName);

Actions(TestMYpuppy);

Diff\_Constructors();

}

private static void Actions(Puppy object)

{

// Виклик методів для демонстрації

object.askName();

object.makeSound();

object.jump();

object.run();

object.bite();

}

public static void Diff\_Constructors(){

// Створення об'єкта собаки з використанням різних значень статі

Puppy p1 = new Puppy("Бобік", 3, "brown", Gender.MALE);

Puppy p2 = new Puppy("Ляля", 2);

System.out.println(p1.getInfo());

System.out.println(p2.getInfo());

}

public static void AllTest() {

//перевірка імені

try {

System.out.println("Правильне ім'я конструктора");

Puppy a = new Puppy("ПравильнийЦуцик");

System.out.println(a.getInfo());

System.out.println();

} catch (IllegalArgumentException e) {

System.out.println("Помилка: " + e.getMessage());

System.out.println();

}

try {

System.out.println("Мало літер ім'я конструктора");

Puppy a = new Puppy("По");

System.out.println(a.getInfo());

System.out.println();

} catch (IllegalArgumentException e) {

System.out.println("Помилка: " + e.getMessage());

System.out.println();

}

try {

System.out.println("Багато літер ім'я конструктора - 24шт");

Puppy a = new Puppy("ПоПоПоПоПоПоПоПоПоПоПоПо");

System.out.println(a.getInfo());

System.out.println();

} catch (IllegalArgumentException e) {

System.out.println("Помилка: " + e.getMessage());

System.out.println();

}

try {

System.out.println("Не літери ім'я конструктора");

Puppy a = new Puppy("12\_34");

System.out.println(a.getInfo());

System.out.println();

} catch (IllegalArgumentException e) {

System.out.println("Помилка: " + e.getMessage());

System.out.println();

}

//перевірка віку

try {

System.out.println("Правильний вік");

Puppy a = new Puppy("Мінік",2);

System.out.println(a.getInfo());

System.out.println();

} catch (IllegalArgumentException e) {

System.out.println("Помилка: " + e.getMessage());

System.out.println();

}

try {

System.out.println("Вік нуль");

Puppy a = new Puppy("Мінік",0);

System.out.println(a.getInfo());

System.out.println();

} catch (IllegalArgumentException e) {

System.out.println("Помилка: " + e.getMessage());

System.out.println();

}

try {

System.out.println("Вік меньше нуля");

Puppy a = new Puppy("Мінік",-2);

System.out.println(a.getInfo());

System.out.println();

} catch (IllegalArgumentException e) {

System.out.println("Помилка: " + e.getMessage());

System.out.println();

}

}

}

Компіляція коду

Для того, щоб відкомпілювати розроблені класи програми за допомогою компілятора javac.exe необхідно відкрити командний рядок (cmd) і використати команду **javac**, яка поставляється разом з JDK (Java Development Kit). Файли проєкту зберігаються в папці src, ми можемо скомпілювати їх такою командою:

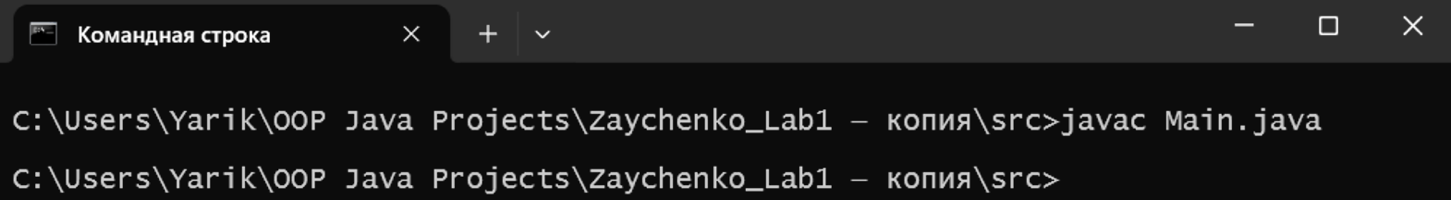
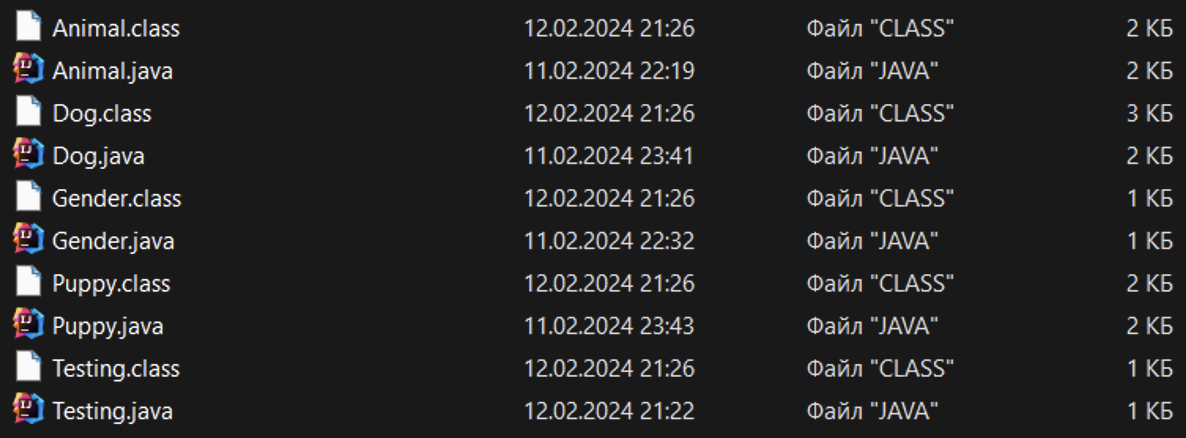


Рисунок 2 – використання команди компіляції файлів \*.java



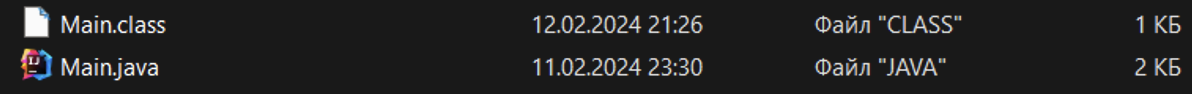


Рисунок 3 – результат використання команди для компіляції

Тестування програми

Тестувати будемо правильними та неправильними значеннями. Всі тестування були виконані за допомогою окремого класу, кожен тест підписаний. Результат на рисунку 4.

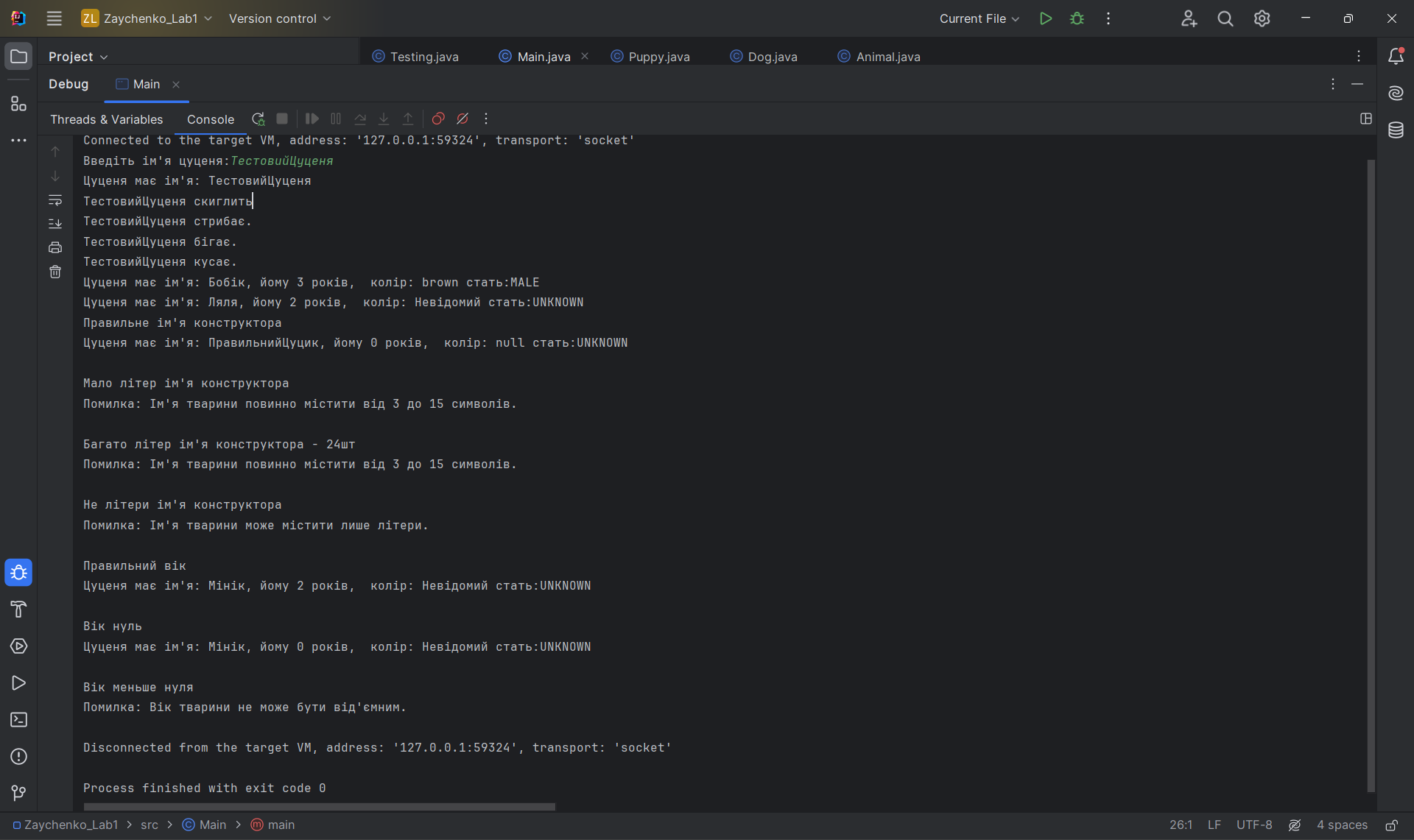


Рисунок 4 – виконання всіх тестових випадків

Генерація документації на розроблений додаток

Для цього необхідно скористатись командою **javadoc**. В даному випадку вона буде написана таким чином: ***javadoc -d docs Main.java***. Результат виконання команди для створення документації зображено на рисунку 5.

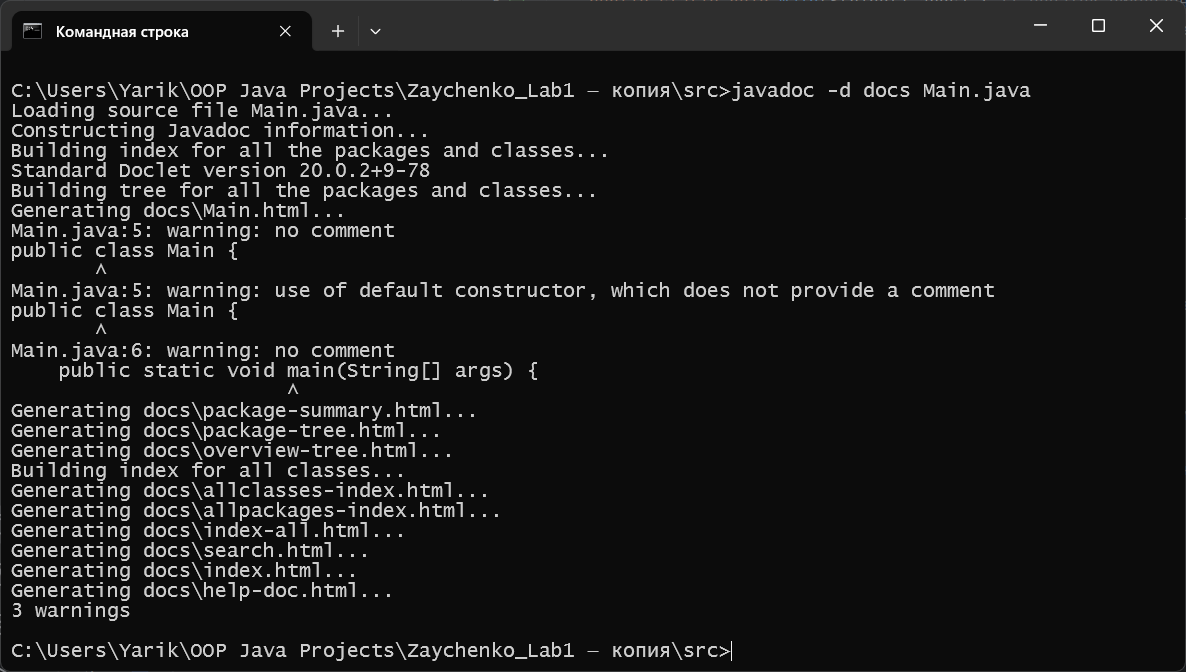


Рисунок 5 – створення документації додатку стандартними командами JRE

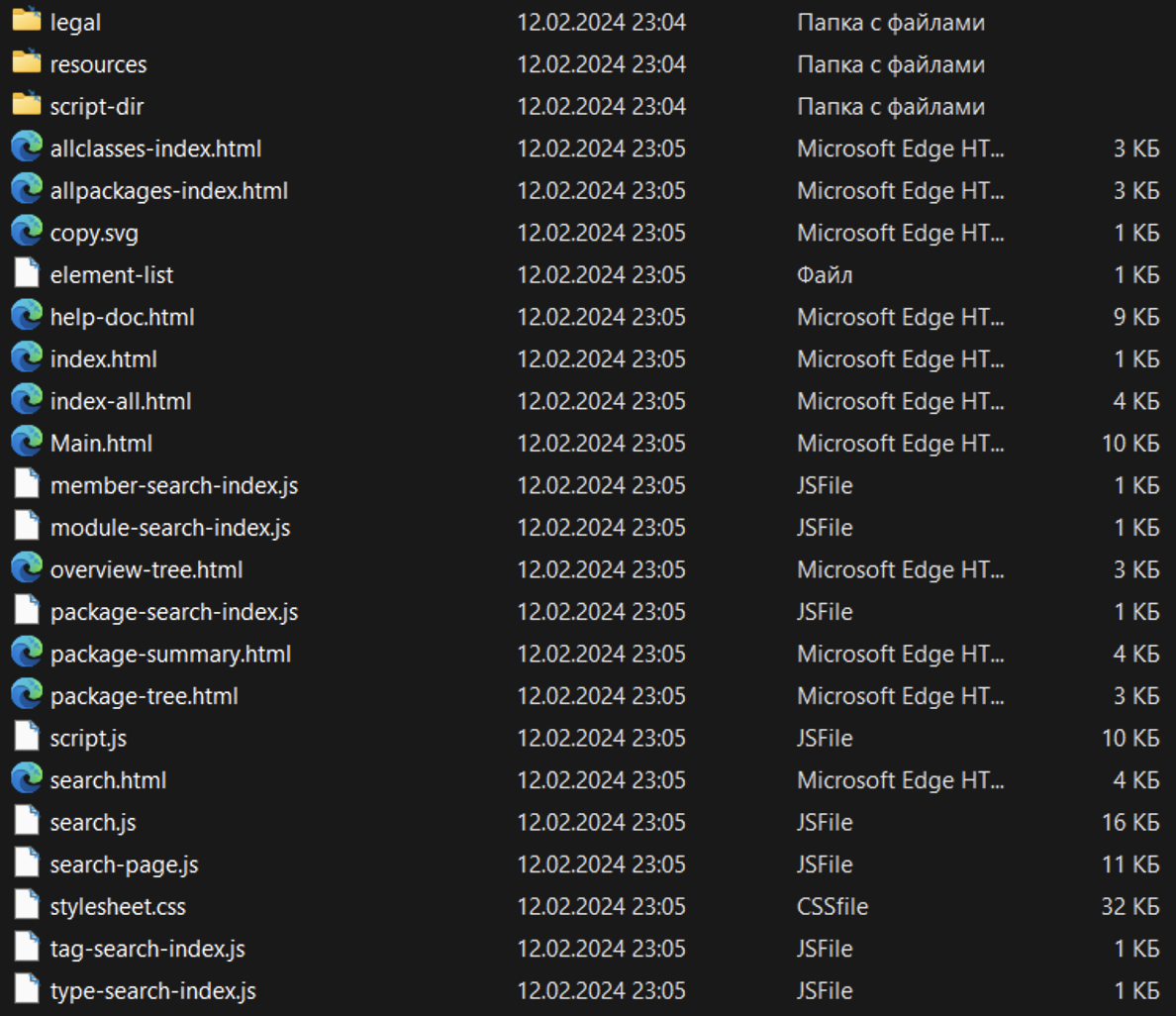


Рисунок 6 – отримані файли документації на додаток

Висновок

У ході виконання лабораторної роботи було розроблено програмний додаток мовою Java для моделювання властивостей, поведінки та взаємодії об'єктів. Було використано концепти об'єктно-орієнтованого програмування, такі як класи, конструктори, методи, геттери та сеттери.

Під час розробки програмного коду було враховано постановку завдання, в якій вказано необхідність реалізації конструкторів та методів для зміни поведінки та стану об'єктів. Кожен клас був ретельно розроблений з урахуванням його властивостей та поведінки.

Для підтримки зрозумілості та документації коду, було додано коментарі та дескриптори документування для опису класів, конструкторів, методів та полів. Це сприяє зручнішому розумінню функціоналу кожного класу та його взаємодії з іншими.

Також було реалізовано тестуючий клас для демонстрації створення об'єктів класів та виклику методів, що змінюють їх властивості та поведінку. Цей етап дозволив перевірити коректність роботи розробленого додатку та виявити та виправити можливі помилки.

Крім того, було створено документацію на розроблений додаток за допомогою утиліти javadoc.exe. Це дозволяє отримати зрозумілу та структуровану інформацію про функціонал програми та класів, що її складають.

Отже, виконання лабораторної роботи над даним проектом дало можливість отримати практичні навички з проектування та реалізації об'єктно-орієнтованих програмних додатків мовою Java.

Контрольні питання

*Додаток А*Машинний лістинг файлу **Main.java**

import Classes.Testing;

import java.util.Scanner;

public class Main { // клас мейн

public static void main(String[] args) { // початок виконання програми в методі мейн

// Створення об'єкта Scanner для зчитування введеного тексту з консолі

Scanner scanner = new Scanner(System.in);

boolean end = false;

while (!end) {

try {

// Виведення повідомлення, щоб користувач ввів текст

System.out.print("Введіть ім'я цуценя:");

String inputName = scanner.nextLine(); // Зчитування введеного тексту

Testing.StartTest(inputName); //виклик методу з класу для тестів

end = true; //все пройшло без помилок - можна виходити

} catch (IllegalArgumentException ex) {

System.out.println(ex.getMessage()); // виведення тексту помилки

}

}

Testing.AllTest();

}

}

*Додаток Б*Машинний лістинг файлу **Animal.java**

package Classes;

//загальні властивості тварин.

public class Animal {

private String name;

private int Age;

// конструктор з параметром ім'я.

public Animal(String NAME) {

setName(NAME);

}

public Animal(String NAME, int AGE) {

setName(NAME);

setAge(AGE);

}

// отримання імені тварини.

public String getName() {

return this.name;

}

public int getAge() {

return this.Age;

}

// Метод-сеттер для імені тварини з перевіркою на відповідність заданим критеріям

public void setName(String name) throws IllegalArgumentException {

// Розбиваємо рядок на символи і перевіряємо кожен символ

for (char c : name.toCharArray()) {

if (!Character.isLetter(c)) {

throw new IllegalArgumentException("Ім'я тварини може містити лише літери.");

}

}

if (name.length() < 3 || name.length() > 20) {

throw new IllegalArgumentException("Ім'я тварини повинно містити від 3 до 15 символів.");

}

this.name = name;

}

public void setAge(int AGE) throws IllegalArgumentException {

if (AGE < 0) throw new IllegalArgumentException("Вік тварини не може бути від'ємним.");

this.Age=AGE;

}

// метод звуку

public void makeSound() {

System.out.println("Тварина видає звук.");

}

}

*Додаток В*Машинний лістинг файлу **Dog.java**

package Classes;

import Classes.Puppy;

import javax.swing.\*;

public class Testing {

public static void StartTest(String puppyName){

// Створення об'єкта класу Цуценя

Puppy TestMYpuppy = new Puppy(puppyName);

Actions(TestMYpuppy);

Diff\_Constructors();

}

private static void Actions(Puppy object)

{

// Виклик методів для демонстрації

object.askName();

object.makeSound();

object.jump();

object.run();

object.bite();

}

public static void Diff\_Constructors(){

// Створення об'єкта собаки з використанням різних значень статі

Puppy p1 = new Puppy("Бобік", 3, "brown", Gender.MALE);

Puppy p2 = new Puppy("Ляля", 2);

System.out.println(p1.getInfo());

System.out.println(p2.getInfo());

}

public static void AllTest() {

//перевірка імені

try {

System.out.println("Правильне ім'я конструктора");

Puppy a = new Puppy("ПравильнийЦуцик");

System.out.println(a.getInfo());

System.out.println();

} catch (IllegalArgumentException e) {

System.out.println("Помилка: " + e.getMessage());

System.out.println();

}

try {

System.out.println("Мало літер ім'я конструктора");

Puppy a = new Puppy("По");

System.out.println(a.getInfo());

System.out.println();

} catch (IllegalArgumentException e) {

System.out.println("Помилка: " + e.getMessage());

System.out.println();

}

try {

System.out.println("Багато літер ім'я конструктора - 24шт");

Puppy a = new Puppy("ПоПоПоПоПоПоПоПоПоПоПоПо");

System.out.println(a.getInfo());

System.out.println();

} catch (IllegalArgumentException e) {

System.out.println("Помилка: " + e.getMessage());

System.out.println();

}

try {

System.out.println("Не літери ім'я конструктора");

Puppy a = new Puppy("12\_34");

System.out.println(a.getInfo());

System.out.println();

} catch (IllegalArgumentException e) {

System.out.println("Помилка: " + e.getMessage());

System.out.println();

}

//перевірка віку

try {

System.out.println("Правильний вік");

Puppy a = new Puppy("Мінік",2);

System.out.println(a.getInfo());

System.out.println();

} catch (IllegalArgumentException e) {

System.out.println("Помилка: " + e.getMessage());

System.out.println();

}

try {

System.out.println("Вік нуль");

Puppy a = new Puppy("Мінік",0);

System.out.println(a.getInfo());

System.out.println();

} catch (IllegalArgumentException e) {

System.out.println("Помилка: " + e.getMessage());

System.out.println();

}

try {

System.out.println("Вік меньше нуля");

Puppy a = new Puppy("Мінік",-2);

System.out.println(a.getInfo());

System.out.println();

} catch (IllegalArgumentException e) {

System.out.println("Помилка: " + e.getMessage());

System.out.println();

}

}

}

*Додаток Г*Машинний лістинг файлу **Gender.java**

package Classes;

// Оголошення перерахування для статі тварини

public enum Gender {

MALE,

FEMALE,

UNKNOWN;

}

*Додаток Ґ*Машинний лістинг файлу **Puppy.java**

package Classes;

//властивості та поведінкa цуценяти.

public class Puppy extends Dog {

private String color;

public Puppy(String name)

{

super(name);

}

// Конструктор з параметрами ім'я, вік, колір та стать

public Puppy(String name, int age, String color, Gender gender) {

super(name, age, gender);

this.color = color;

}

// Конструктор з параметрами ім'я та вік (колір і стать встановлюються за замовчуванням)

public Puppy(String name, int age) {

super(name, age, Gender.UNKNOWN);

this.color = "Невідомий";

}

// Метод для отримання інформації про ім'я та колір цуценяти

public String getInfo() {

String result = ("Цуценя має ім'я: " + getName() +", йому "+getAge()+" років, "+" колір: " + color + " стать:"+getGender().toString());

return result;

}

public void askName() {

System.out.println( "Цуценя має ім'я: " + super.getName());

}

@Override

public void makeSound() {

System.out.println(super.getName() +" скиглить");

}

}

*Додаток Д*Машинний лістинг файлу **Testing.java**

package Classes;

import Classes.Puppy;

import javax.swing.\*;

public class Testing {

public static void StartTest(String puppyName){

// Створення об'єкта класу Цуценя

Puppy TestMYpuppy = new Puppy(puppyName);

Actions(TestMYpuppy);

Diff\_Constructors();

}

private static void Actions(Puppy object)

{

// Виклик методів для демонстрації

object.askName();

object.makeSound();

object.jump();

object.run();

object.bite();

}

public static void Diff\_Constructors(){

// Створення об'єкта собаки з використанням різних значень статі

Puppy p1 = new Puppy("Бобік", 3, "brown", Gender.MALE);

Puppy p2 = new Puppy("Ляля", 2);

System.out.println(p1.getInfo());

System.out.println(p2.getInfo());

}

public static void AllTest() {

//перевірка імені

try {

System.out.println("Правильне ім'я конструктора");

Puppy a = new Puppy("ПравильнийЦуцик");

System.out.println(a.getInfo());

System.out.println();

} catch (IllegalArgumentException e) {

System.out.println("Помилка: " + e.getMessage());

System.out.println();

}

try {

System.out.println("Мало літер ім'я конструктора");

Puppy a = new Puppy("По");

System.out.println(a.getInfo());

System.out.println();

} catch (IllegalArgumentException e) {

System.out.println("Помилка: " + e.getMessage());

System.out.println();

}

try {

System.out.println("Багато літер ім'я конструктора - 24шт");

Puppy a = new Puppy("ПоПоПоПоПоПоПоПоПоПоПоПо");

System.out.println(a.getInfo());

System.out.println();

} catch (IllegalArgumentException e) {

System.out.println("Помилка: " + e.getMessage());

System.out.println();

}

try {

System.out.println("Не літери ім'я конструктора");

Puppy a = new Puppy("12\_34");

System.out.println(a.getInfo());

System.out.println();

} catch (IllegalArgumentException e) {

System.out.println("Помилка: " + e.getMessage());

System.out.println();

}

//перевірка віку

try {

System.out.println("Правильний вік");

Puppy a = new Puppy("Мінік",2);

System.out.println(a.getInfo());

System.out.println();

} catch (IllegalArgumentException e) {

System.out.println("Помилка: " + e.getMessage());

System.out.println();

}

try {

System.out.println("Вік нуль");

Puppy a = new Puppy("Мінік",0);

System.out.println(a.getInfo());

System.out.println();

} catch (IllegalArgumentException e) {

System.out.println("Помилка: " + e.getMessage());

System.out.println();

}

try {

System.out.println("Вік меньше нуля");

Puppy a = new Puppy("Мінік",-2);

System.out.println(a.getInfo());

System.out.println();

} catch (IllegalArgumentException e) {

System.out.println("Помилка: " + e.getMessage());

System.out.println();

}

}

}