Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра «Електронних обчислювальних машин»



Звіт

з лабораторної роботи № 3

з дисципліни: «Кросплатформенні засоби програмування»

на тему: «Спадкування та інтерфейси»

**Виконав:**

студент групи КІ-306

Фодор Андрій

**Прийняв:**

доцент кафедри ЕОМ

Іванов Ю. С.

Львів – 2023

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

**Завдання (варіант № 21)**

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 21 – ВОДЯНИЙ ПІСТОЛЕТ

Для того, аби виконати спадкування, виконав такі кроки:

1. Перед словом class в оголошенні класу Pistol написав ключове слово abstract, що дозволило зробити цей клас абстрактним
2. У вкладці File>Project Structure обрав вкладку Artifacts, додав головний клас, в якому міститься main, тим самим налаштувавши Artifacts
3. У вкладці Build обрав Build Artefacts…, що дозволило згенерувати jar файл, в якому містяться всі класи з цього проекту
4. Подібним способом додай цей файл як бібліотеку до проекту третьої ЛР

Це дозволило в проекті користуватися цим класом Pistol, а саме успадкувати від нього новий клас WaterPistol

**Вихідний код програми**

**Файл WaterPistol.java**

package KI306.Fodor.Lab3;

import KI306.Fodor.Lab2.Pistol;

import java.io.FileNotFoundException;

/\*\*

\* Interface waterable describes the ability to use water shots

\*/

interface waterable {

String realizeWaterAble();

boolean waterAble = true;

}

/\*\*

\* Interface waterPressure extends waterable and describes the ability to calculate water pressure in weapon

\*/

interface waterPressure extends waterable {

int calculateWP();

int k = 2;

}

//завдяки такій організації класу ми отримаємо альтернативу множинного наслідування з С++, оскільки можемо реалізовувати скільки завгодно інтерфейсів

/\*\*

\* Class WaterPistol inherits from the Pistol class and implements the interface waterPressure

\*/

public class WaterPistol extends Pistol implements waterPressure {

/\*\*

\* Default constructor. Call superclass default constructor

\* @throws FileNotFoundException

\*/

public WaterPistol() throws FileNotFoundException {

super();

this.holeArea = (int)(Math.random()\*15+5);

}

/\*\*

\* Constructor with parameters. Call superclass constructor with parameters

\* @param bulletCount - count of bullets you want

\* @param model - model of pistol you want

\* @param holeArea - hole area size you want in mm

\* @throws FileNotFoundException

\*/

public WaterPistol(int bulletCount, String model, int holeArea) throws FileNotFoundException{

super(bulletCount, model);

if(holeArea <= 20){

this.holeArea = holeArea;

}else{

this.holeArea = 20;

}

}

/\*\*

\* Interface`s implemented method

\* @return changed pistol model

\*/

@Override

public String realizeWaterAble() {

this.water = waterAble;

return this.getGunsModel();

}

/\*\*

\* Interface`s realised method

\* @return water pressure

\*/

@Override

public int calculateWP() {

this.pressure = k\*50/this.holeArea;

return this.pressure;

}

//перевизначення методів батьківського класу

/\*\*

\* Overridden superclass method, that call inside itself appropriate method

\* @param howMany - count of shoots tou want

\* @param distance - distance for shooting

\*/

public void makeSomeShoots(int howMany, int distance){

if(this.water && this.pressure != 0){

super.makeSomeShoots(howMany, distance\*(20/this.pressure));

System.out.println("Water shooting is successful");

}else{

System.out.println("Water shooting is cannot be started because you don`t realized waterAble or don`t calculate pressure");

}

}

/\*\*

\* Overridden method, that change return logic

\* @return changed model of pistol

\*/

public String getGunsModel(){

return super.getGunsModel() + " water version";

}

//власний метод, що розширює функціонал підкласу, не є реалізацією інтерфейсу

/\*\*

\* This class own independent method, that realize necessary initialization

\* @return boolean value that indicate whether initialization is successful

\*/

public boolean makeSettings(){

this.calculateWP();

this.realizeWaterAble();

return true;

}

private int pressure;

private int holeArea;

private boolean water;

}

/\*\*

\* Demonstrative class

\*/

class MAIN {

public static void main(String[] args) throws FileNotFoundException {

WaterPistol wp = new WaterPistol(/\*5, "Just a toy", 15\*/);

if(wp.makeSettings()){

wp.makeSomeShoots(3, 5);

// Pistol p = new Pistol();

// MAIN.ShowInfoAboutWeapon(p);

ShowInfoAboutWeapon(wp);

System.out.println("Pressure = " + wp.calculateWP());

} else{

System.out.println("Something has wrong");

}

wp.dispose();

}

//тут виконано демонстрацію динамічного (пізнього) зв'язування та поліморфізм в цілому

public static void ShowInfoAboutWeapon(Pistol p){

System.out.println("--- Weapon info ---\nModel = " + p.getGunsModel() +

";\nServiceability = " + p.getGunsServiceability() +

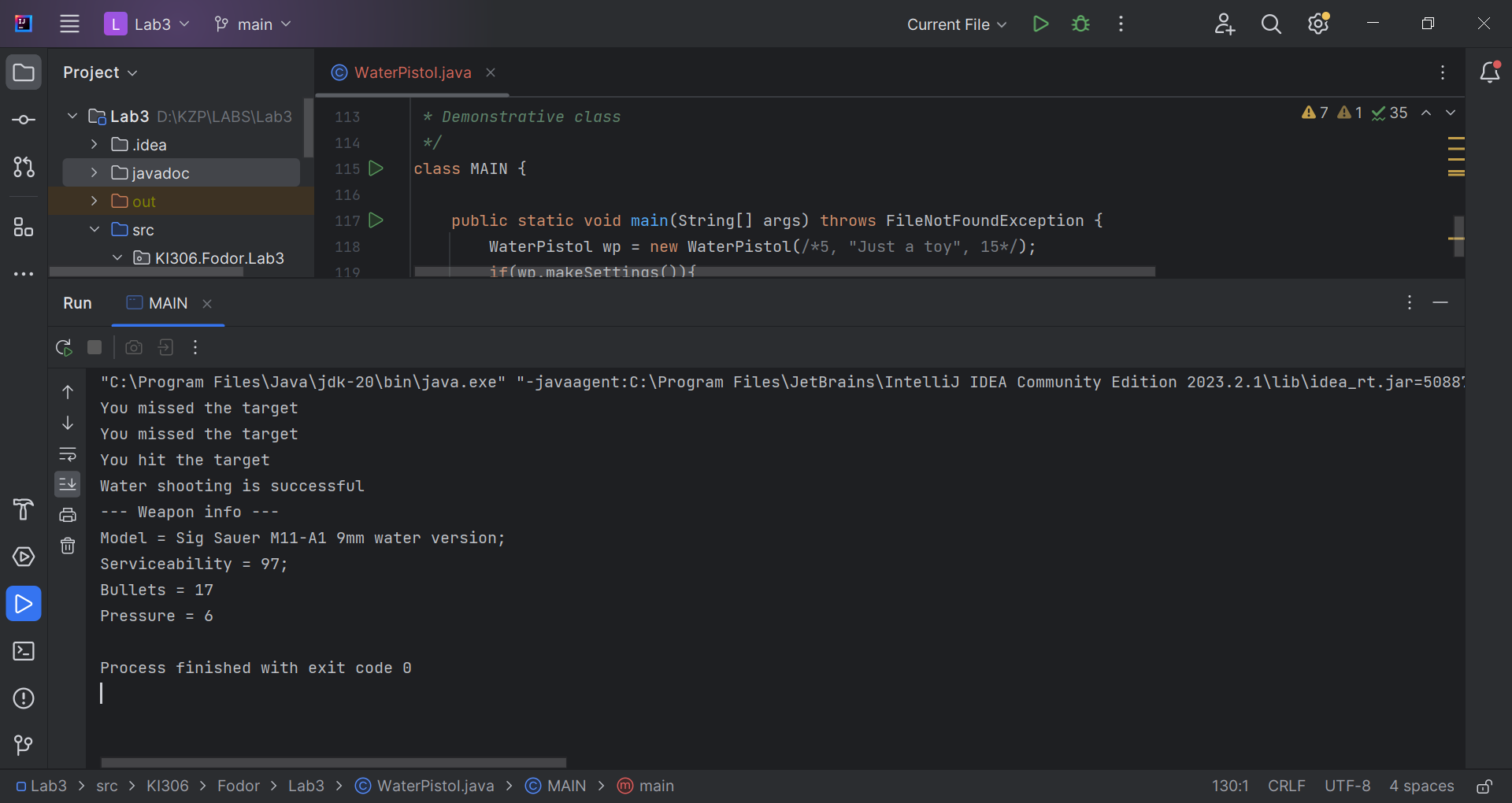
";\nBullets = " + p.getGunsBulletCount());

}

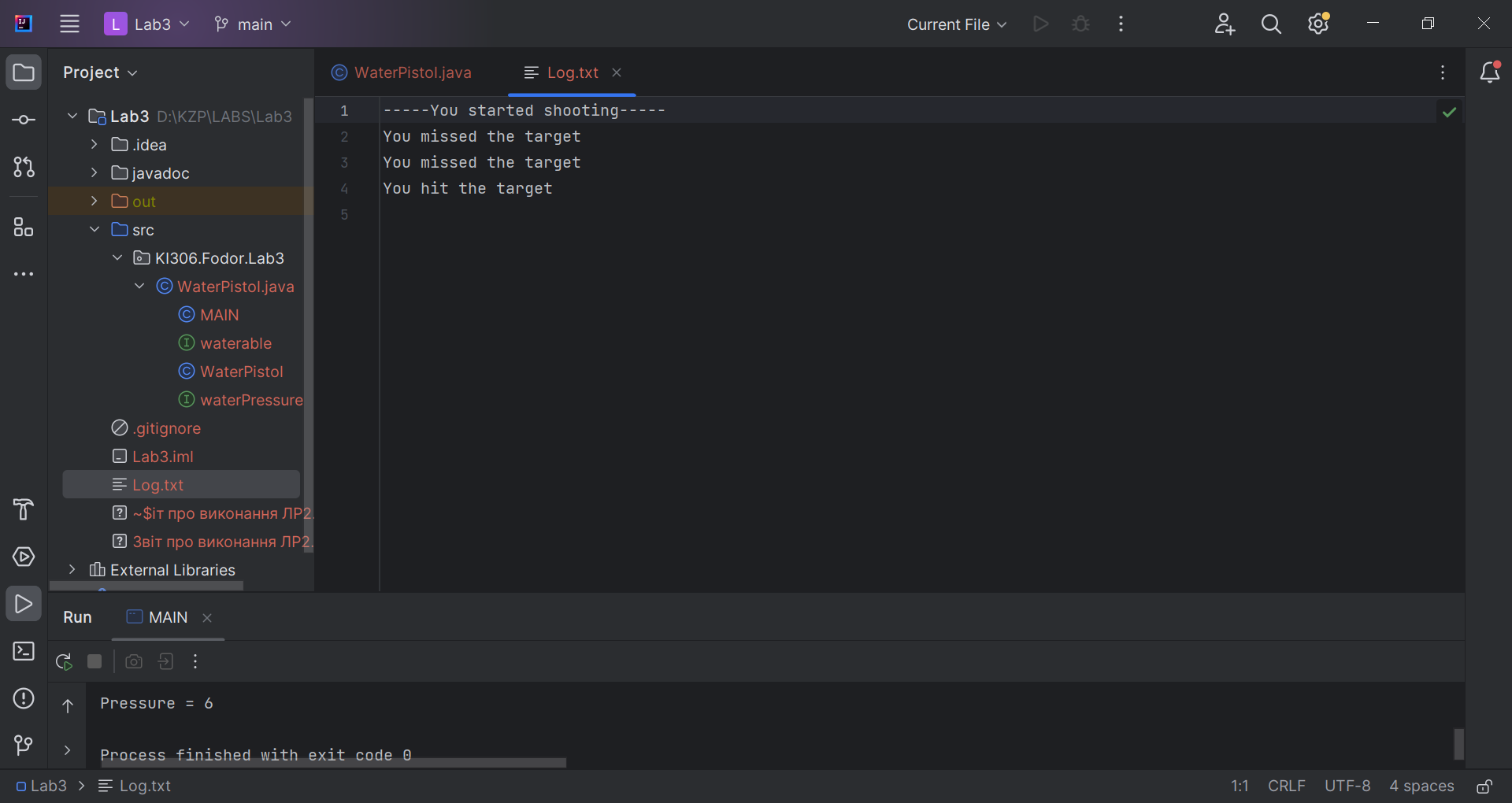
}

Виконав невелику демонстрацію програми, аби наглядно побачити реалізацію поліморфізму, спадкування, реалізації інтерфейсів та неможливість створення об’єкту абстрактного класу

**Результат виконання програми**

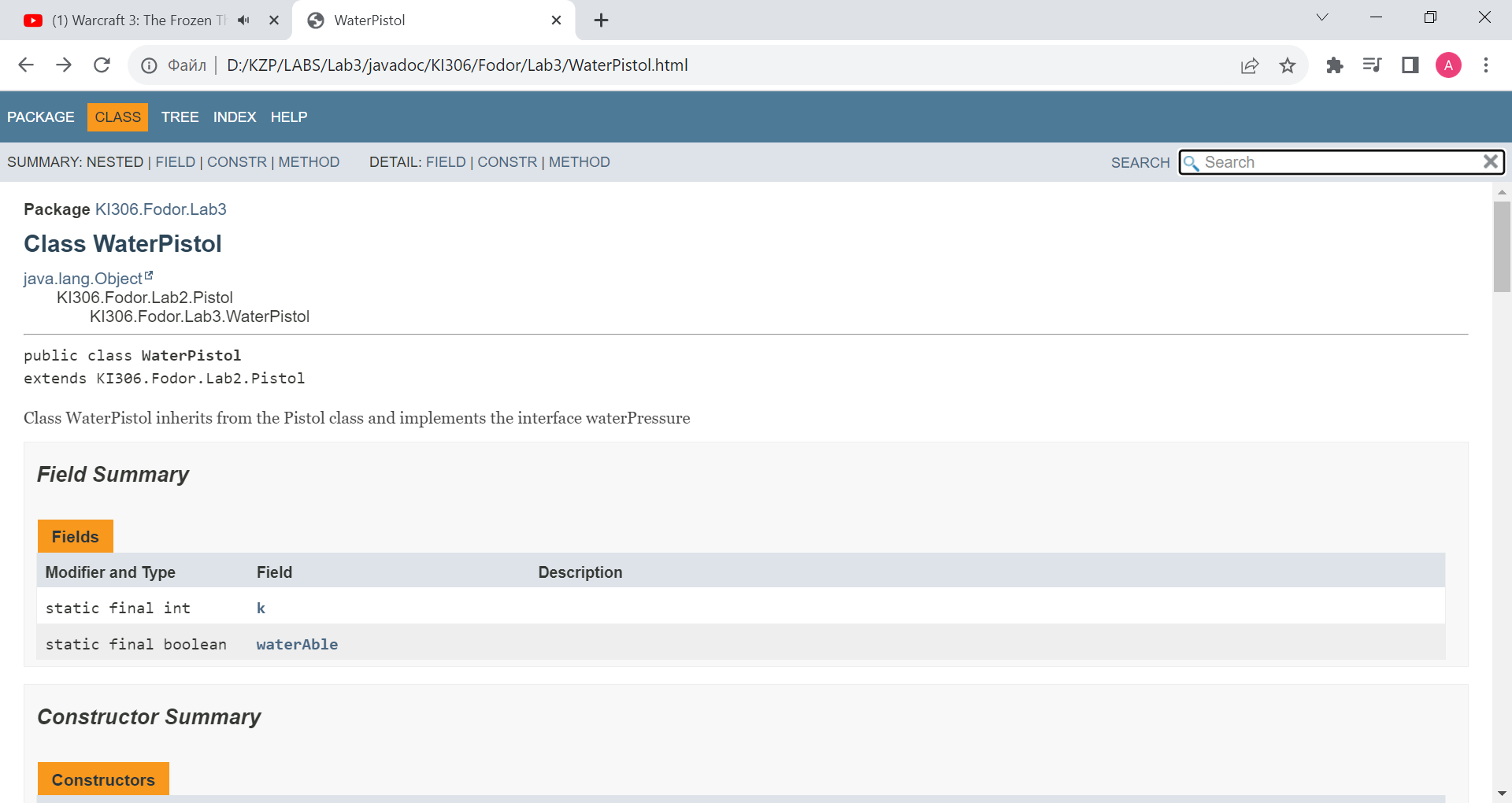


Вміст файлу Log.txt після виконання вказаного фрагменту коду:



Варто звернути увагу, що оскільки тиск води в пістолеті сформувався доволі малий, то точність прямопропорційно зменшується (досить велике поле holeArea – площа отвору пістолета). Весь функціонал, який був доступний в класі Pistol, залишається доступним і в класі WaterPistol. Недоступними стали поля класу Pistol, оскільки вони мали модифікатор доступу private.

**Фрагмент згенерованої документації**



**Висновок**

На цій лабораторній роботі я ознайомився із спадкуванням та інтерфейсами у мові Java. Закріпив отримані знання на прикладі виконання індивідуального завдання. Переконався у принципах роботи вивчених механізмів, реалізувавши інтерфейс, спадкування та поліморфізм.