Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра ЕОМ

Звіт

з лабораторної роботи № 3

з дисципліни «Кросплатформенні засоби програмування»

на тему: «Спадкування та інтерфейси»

Виконав: ст. гр. КІ-302

Радевич-Винницький Я.А.

Перевірив: викладач

Майдан М.В.

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 №3, для реалізації предметної області заданої варіантом. Суперклас, що реалізований у лабораторній роботі №3, зробити абстрактним. Розроблений підклас має забезпечувати механізми свого коректного функціонування та реалізовувати мінімум один інтерфейс. Програма має розміщуватися в пакеті Група.Прізвище.Lab4 та володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету.
2. Автоматично згенерувати документацію до розробленого пакету.
3. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації.
4. Дати відповідь на контрольні запитання.

Варіант: 22. Штурмова гвинтівка.

Виконання завдання:

Код файлу assaultRifle.java, що містить абстрактний суперклас assaultRifle:

Лістинг 1

```
package KI302.RadevychVynnytskyi.Lab2;
import java.io.IOException;
import java.io.PrintWriter;
import java.io.File;
/**
 * Абстрактний клас `assaultRifle` представляє базовий клас для опису
автоматичних гвинтівок.
 * Цей клас містить загальні властивості і методи, які можуть бути у
спадкоємцях.
 */
abstract public class assaultRifle {
    private String name;
    private double cartridge;
    private double mass;
    private double length;
    private int firingRange;
    private int bulletsCapacity;
    private int bulletsCurrently;
    private int price;
    private boolean scope;
    private boolean stock;
    private boolean muffler;
    private PrintWriter fout;

    /**
     * Constructs an assaultRifle object with default values.
     *
     * @throws IOException if there is an error creating the log file.
     */

    public assaultRifle() throws IOException {
```

```
        name = null;
        cartridge = 0.0;
        mass = 0.0;
        length = 0.0;
        firingRange = 0;
        bulletsCapacity = 0;
        bulletsCurrently = 0;
        price = 0;
        scope = false;
        stock = false;
        muffler = false;
        fout = new PrintWriter(new File("Log.txt"));
    }

    public assaultRifle(String name, double cartridge, double mass, double
length, int firingRange, int bulletsCapacity, int bulletsCurrently, int price,
boolean scope, boolean stock, boolean muffler) throws IOException {
        this.name = name;
        this.cartridge = cartridge;
        this.mass = mass;
        this.length = length;
        this.firingRange = firingRange;
        this.bulletsCapacity = bulletsCapacity;
        this.bulletsCurrently = bulletsCurrently;
        this.price = price;
        this.scope = scope;
        this.stock = stock;
        this.muffler = muffler;
        fout = new PrintWriter(new File("Log.txt"));
    }

    public void getInfo() {
        System.out.println("Name: " + name);
        System.out.println("Cartridge: " + cartridge);
        System.out.println("Mass: " + mass + " kg");
        System.out.println("Length: " + length + " mm");
        System.out.println("FiringRange: " + firingRange + " m");
        System.out.println("Capacity of magazine: " + bulletsCapacity + "
bullets");
        System.out.println("Bullets Balance: " + bulletsCurrently);
        System.out.println("Price: " + price + " $");
        System.out.println("Scope: " + scope);
        System.out.println("Stock: " + stock);
        System.out.println("Muffler: " + muffler);
        logActivity("Info printed");
    }

    private void logActivity(String message) {
        fout.println(message);
        fout.flush();
    }

    /**
     * Closes the log file.
     */
    public void closeLogFile() {
        fout.close();
    }

    public void fullAutoFire() throws InterruptedException {
        for (int i = bulletsCurrently; i >= 0; i--) {
            System.out.println(i + " bullets left");
            Thread.sleep(50);
            setBulletsCurrently(i);
```

```
        }
        System.out.println("Run out of bullets!");
        logActivity("Auto Fire executed");
    }


    public void singleFire(int shots) throws InterruptedException {

        if (shots <= getBulletsCurrently()) {
            for (int i = getBulletsCurrently(); i >= getBulletsCurrently() -
shots; i--) {

                System.out.println(i + " bullets left");
                Thread.sleep(700);
            }
            setBulletsCurrently(getBulletsCurrently() - shots);
        } else {
            System.out.println(getBulletsCurrently() + " bullets left. You
can't do more shots than bullets left");
        }
        logActivity("Single Fire executed");
    }

    public void installMuffler(boolean muffler) {
        if (muffler == false) {
            muffler = true;
            setMuffler(muffler);
            System.out.println("Muffler is installed");
            logActivity("Muffler installed");
        } else {
            System.out.println("Muffler is alredy installed");
        }

    }

    public void unistallMuffler(boolean muffler) {
        if (muffler == true) {
            muffler = false;
            setMuffler(muffler);
            System.out.println("Muffler is uninstalled");
            logActivity("Muffler uninstalled");
        } else {
            System.out.println("There is no muffler on this gun");
        }
    }

    public void installScope(boolean scope) {
        if (scope == false) {
            scope = true;
            setScope(scope);
            System.out.println("Scope is installed");
            logActivity("Scope installd");
        } else {
            System.out.println("Scope is alredy installed");
        }
    }

    public void unistallScope(boolean scope) {
        if (scope == true) {
            scope = false;
            setScope(scope);
            System.out.println("Scope is uninstalled");
            logActivity("Scope uninstalled");
        } else {
```

```
            System.out.println("There is no scope on this gun");
        }
    }

    public void reloadGun() {
        if (getBulletsCurrently() != 30) {
            setBulletsCurrently(getBulletsCapacity());
            System.out.println("The assault rifle is reloaded");
            logActivity("Gun reload");
        } else {
            System.out.println("The magazine is full");
        }
    }

    public void compareGuns(assaultRifle a, assaultRifle b) {
        a.getInfo();
        System.out.println();
        b.getInfo();
        logActivity("Guns comparison");
    }


    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setCartridge(double cartridge) {
        this.cartridge = cartridge;
    }

    public void setMass(double mass) {
        this.mass = mass;
    }

    public void setLength(double length) {
        this.length = length;
    }

    public void setFiringRange(int firingRange) {
        this.firingRange = firingRange;
    }

    public void setBulletsCapacity(int bulletsCapacity) {
        this.bulletsCapacity = bulletsCapacity;
    }

    public int getBulletsCapacity() {
        return bulletsCapacity;
    }

    public int getBulletsCurrently() {
        return bulletsCurrently;
    }

    public void setBulletsCurrently(int bulletsCurrently) {
        this.bulletsCurrently = bulletsCurrently;
    }

    public void setPrice(int price) {
        this.price = price;
```

```
    }

    public void setStock(boolean stock) {
        this.stock = stock;
    }

    public boolean getMuffler() {
        return muffler;
    }

    public void setMuffler(boolean muffler) {
        this.muffler = muffler;
    }

    public boolean getScope() {
        return scope;
    }

    public void setScope(boolean scope) {
        this.scope = scope;
    }

}
```

Код файлу stormRifle.java, який містить підклас stormRifle, що наслідує суперклас assaultRifle та імплементує інтерфейс killPlayer:

Лістинг 2

```
package KI302.RadevychVynnytskyi.Lab2;

import java.io.IOException;
/**
 * Клас `stormRifle` представляє штурмову гвинтівку, яка є підкласом класу
`assaultRifle` і реалізує інтерфейс `bodyshotKill та headshotKill`.
 */
public class stormRifle extends assaultRifle implements killPlayer{
    private int headDamage;
    private int bodyDamage;
    public stormRifle(String name, double cartridge, double mass, double
length, int firingRange, int bulletsCapacity, int bulletsCurrently, int price,
boolean scope, boolean stock, boolean muffler, int headDamage, int bodyDamage)
throws IOException {
        super(name, cartridge, mass, length, firingRange, bulletsCapacity,
bulletsCurrently, price, scope, stock, muffler);
        this.headDamage = headDamage;
        this.bodyDamage = bodyDamage;
    }
    public void bodyShotKill(int xp, int bodyDamage) throws
InterruptedException {
        int count = 0;

        System.out.println("Player health points: " + xp + "%");
        System.out.println(getName() + " body damage: " + bodyDamage + "\n");

        for (int i = getBulletsCurrently() - 1; i >= 0; i--){
            setBulletsCurrently(i);
            xp -= bodyDamage;
            System.out.println(i + " bullets left");
            Thread.sleep(700);
```

```
            count++;
            if (xp > 0){
                System.out.println("Enemy's health: " + xp + "%");
            } else{
                System.out.println("Enemy's health: 0%");
                System.out.println("Player was killed in " + count  + "
shots");
                break;
            }
        }
    }
    public void headShotKill(int xp, int headDamage) throws
InterruptedException {
        int count = 0;
        System.out.println("Player health points: " + xp + "%");
        System.out.println(getName() + " head damage: " + headDamage + "\n");
        for (int i = getBulletsCurrently() - 1; i >= 0; i--){
            setBulletsCurrently(i);
            xp -= headDamage;
            System.out.println(i + " bullets left");
            Thread.sleep(700);
            count++;
            if (xp > 0){
                System.out.println("Enemy's health: " + xp + "%");
            } else{
                System.out.println("Enemy's health: 0%");
                System.out.println("Player was killed in " + count  + "
shot");
                break;
            }
        }
    }
}
```

Код файлу killPlayer.java, який містить інтерфейс killPlayer:

Лістинг 3

```
package KI302.RadevychVynnytskyi.Lab2;

public interface killPlayer {
    void bodyShotKill(int xp, int bodyDamage) throws InterruptedException;
    void headShotKill(int xp, int headDamage) throws InterruptedException;
}
```

Код файлу main.java, який містить клас-драйвер для тестування і демонстрації роботи програми:

Лістинг 4

```
package KI302.RadevychVynnytskyi.Lab2;

import java.io.IOException;
import java.util.Scanner;

/**
 *
```

```java
 */
public class main {

    /**
     * The main method where the program execution begins.
     *
     * @param args Unused command-line arguments.
     * @throws IOException          if there is an error related to file
input/output.
     * @throws InterruptedException if a thread is interrupted during
execution.
     */
    public static void main(String[] args) throws IOException,
InterruptedException {
        // TODO Auto-generated method stub
        stormRifle ak47 = new stormRifle("AK-47", 7.62, 4.2, 880, 350, 30, 30,
2700, false, true, false, 111, 34);
        mainMenu(ak47);
    }

    @SuppressWarnings("null")
    public static void mainMenu(stormRifle gun) throws InterruptedException,
IOException {
        while(true) {
            System.out.println("Main Menu: \n");
            System.out.println("1. Display Info");
            System.out.println("2. Automatic fire");
            System.out.println("3. Single fire");
            System.out.println("4. Install muffler");
            System.out.println("5. Remove muffler");
            System.out.println("6. Install scope");
            System.out.println("7. Remove scope");
            System.out.println("8. Reload the gun");
            System.out.println("9. Compare AK-47 with M4");
            System.out.println("10.Compare AK-47 with your gun");
            System.out.println("11.Shoot in the head");
            System.out.println("12.Shoot in the body");
            System.out.println("13.Exit\n");
            System.out.println("_________________");
            System.out.println("Select option: ");
            Scanner input = new Scanner(System.in);
            int choice = input.nextInt();
            int exit = 0;

            switch(choice) {
                case 1:
                    clearConsole();
                    gun.getInfo();
                    System.out.println("\nPress 0 - Back to Menu");
                    exit = input.nextInt();

                    if (exit == 0) {
                        clearConsole();
                        break;
                    }
                case 2:
                    clearConsole();
                    gun.fullAutoFire();
                    System.out.println("\nPress 0 - Back to Menu");
                    exit = input.nextInt();

                    if (exit == 0) {
                        clearConsole();
                        break;
```

```
                }
            case 3:
                clearConsole();
                System.out.println("How many shots should be done? ");
                int shots = input.nextInt();
                gun.singleFire(shots);
                System.out.println("\nPress 0 - Back to Menu");
                exit = input.nextInt();

                if (exit == 0) {
                    clearConsole();
                    break;
                }
            case 4:
                clearConsole();
                gun.installMuffler(gun.getMuffler());
                System.out.println("\nPress 0 - Back to Menu");
                exit = input.nextInt();

                if (exit == 0) {
                    clearConsole();
                    break;
                }
            case 5:
                clearConsole();
                gun.unistallMuffler(gun.getMuffler());
                System.out.println("\nPress 0 - Back to Menu");
                exit = input.nextInt();

                if (exit == 0) {
                    clearConsole();
                    break;
                }
            case 6:
                clearConsole();
                gun.installScope(gun.getScope());
                System.out.println("\nPress 0 - Back to Menu");
                exit = input.nextInt();

                if (exit == 0) {
                    clearConsole();
                    break;
                }
            case 7:
                clearConsole();
                gun.unistallScope(gun.getScope());
                System.out.println("\nPress 0 - Back to Menu");
                exit = input.nextInt();

                if (exit == 0) {
                    clearConsole();
                    break;
                }
            case 8:
                clearConsole();
                gun.reloadGun();
                System.out.println("\nPress 0 - Back to Menu");
                exit = input.nextInt();

                if (exit == 0) {
                    clearConsole();
                    break;
                }
            case 9:
```

```
                    clearConsole();
                    assaultRifle m4 = new stormRifle("M4", 5.56, 3.39, 840,
600, 30, 30, 3100, true, true, false, 92, 28);
                    m4.compareGuns(gun, m4);

                    System.out.println("\nPress 0 - Back to Menu");
                    exit = input.nextInt();

                    if (exit == 0) {
                        clearConsole();
                        break;
                    }
                case 10:
                    clearConsole();
                    Scanner gunName = new Scanner(System.in);
                    System.out.println("Enter name of your gun: ");
                    String name = "";
                    if (gunName.hasNextLine()){
                        name = gunName.nextLine();
                    } else {
                        System.out.println("Invalid input. Please enter a
string.");
                        break;
                    }

                    System.out.println("Enter cartridge: ");
                    double cart = 0.0;
                    if (input.hasNextDouble()){
                        cart = input.nextDouble();
                    } else {
                        System.out.println("Invalid input. Please enter a
double.");
                        break;
                    }

                    System.out.println("Enter mass: ");
                    double mass = 0.0;
                    if (input.hasNextDouble()) {
                        mass = input.nextDouble();
                    } else {
                        System.out.println("Invalid input. Please enter a
double.");
                        break;
                    }

                    System.out.println("Enter length: ");
                    double length = 0.0;
                    if (input.hasNextDouble()) {
                        length = input.nextDouble();
                    } else {
                        System.out.println("Invalid input. Please enter a
double.");
                        break;
                    }

                    System.out.println("Enter firing range: ");
                    int firingRange = 0;
                    if (input.hasNextInt()) {
                        firingRange = input.nextInt();
                    } else {
                        System.out.println("Invalid input. Please enter an
integer.");
                        break;
                    }
```

```
                    System.out.println("Enter bullets capacity: ");
                    int bulletsCapacity = 0;
                    if (input.hasNextInt()) {
                        bulletsCapacity = input.nextInt();
                    } else {
                        System.out.println("Invalid input. Please enter an
integer.");
                        break;
                    }

                    System.out.println("Enter price: ");
                    int price = 0;
                    if (input.hasNextInt()) {
                        price = input.nextInt();
                    } else {
                        System.out.println("Invalid input. Please enter an
integer.");
                        break;
                    }

                    System.out.println("Enter if there is a scope: ");
                    boolean scope = false;
                    if (input.hasNextBoolean()) {
                        scope = input.nextBoolean();
                    } else {
                        System.out.println("Invalid input. Please enter a
boolean.");
                        break;
                    }

                    System.out.println("Enter if there is a stock: ");
                    boolean stock = false;
                    if (input.hasNextBoolean()) {
                        stock = input.nextBoolean();
                    } else {
                        System.out.println("Invalid input. Please enter a
boolean.");
                        break;
                    }

                    System.out.println("Enter if there is a muffler: ");
                    boolean muffler = false;
                    if (input.hasNextBoolean()){
                        muffler = input.nextBoolean();
                    } else {
                        System.out.println("Invalid input. Please enter a
bool.");
                        break;
                    }

                    assaultRifle userGun = new stormRifle(name, cart, mass,
length, firingRange, bulletsCapacity, bulletsCapacity, price, scope, stock,
muffler, 92, 28);
                    clearConsole();
                    userGun.compareGuns(gun, userGun);

                    System.out.println("\nPress 0 - Back to Menu");
                    exit = input.nextInt();

                    if (exit == 0) {
                        clearConsole();
                        break;
                    }
```

```
                case 11:
                    clearConsole();
                    gun.headShotKill(100,  111);

                    System.out.println("\nPress 0 - Back to Menu");
                    exit = input.nextInt();

                    if (exit == 0) {
                        clearConsole();
                        break;
                    }
                case 12:
                    clearConsole();
                    gun.bodyShotKill(100, 34);

                    System.out.println("\nPress 0 - Back to Menu");
                    exit = input.nextInt();

                    if (exit == 0) {
                        clearConsole();
                        break;
                    }

            }
            if (choice == 13) {
                input.close();
                gun.closeLogFile();
                clearConsole();
                System.out.println("Exit...");
                break;
            }
        }
    }

    @SuppressWarnings("deprecation")
    public static void clearConsole() {
        try {
            final String os = System.getProperty("os.name");

            if (os.contains("Windows")) {
                // If the OS is Windows, use the "cls" command
                new ProcessBuilder("cmd", "/c",
"cls").inheritIO().start().waitFor();
            } else {
                // If the OS is not Windows (e.g., Linux or macOS), use
"clear"
                Runtime.getRuntime().exec("clear");
            }
        } catch (final Exception e) {
            // Handle exceptions here
            e.printStackTrace();
        }
    }
}
```

Результат роботи програми:

```
Main Menu:

1. Display Info
2. Automatic fire
3. Single fire
4. Install muffler
5. Remove muffler
6. Install scope
7. Remove scope
8. Reload the gun
9. Compare AK-47 with M4
10.Compare AK-47 with your gun
11.Shoot in the head
12.Shoot in the body
13.Exit

-----------------
Select option:

11
Player health points: 100%
AK-47 head damage: 111

29 bullets left
Enemy's health: 0%
Player was killed in 1 shot

Press 0 - Back to Menu
```

```
12
Player health points: 100%
AK-47 body damage: 34

28 bullets left
Enemy's health: 66%
27 bullets left
Enemy's health: 32%
26 bullets left
Enemy's health: 0%
Player was killed in 3 shots

Press 0 - Back to Menu
```

```
1
Name: AK-47
Cartridge: 7.62
Mass: 4.2 kg
Length: 880.0 mm
FiringRange: 350 m
Capacity of magazine: 30 bullets
Bullets Balance: 26
Price: 2700 $
Scope: false
Stock: true
Muffler: false

Press 0 - Back to Menu
```

Рис. 1 – Скріншот консолі після використання методів `headShotKill` *та* `bodyShotKill`

Висновок: у ході виконання лабораторної роботи було вивчено поняття спадкування та інтерфейсів у мові Java. Було розроблено дочірній клас stormRiffle та імплементовано у ньому оголошені в інтерфейсі killPlayer методи headShotKill та bodyShotKill.