Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра ЕОМ

Звіт

до лабораторної роботи №2

З дисципліни: «Кросплатформні засоби програмування»

На тему: «КЛАСИ ТА ПАКЕТИ»

Виконала: ст. гр. КІ-305

Романюк М. Р.

Прийняв: доцент кафедри ЕОМ

Іванов Ю. С.

Львів 2023

Мета: ознайомитися з процесом розробки класів та пакетів мовою Java.

ЗАВДАННЯ

1. Написати та налагодити програму на мові Java, що реалізує у вигляді класу предметну область згідно варіанту. Програма має задовольняти наступним вимогам:
   - програма має розміщуватися в пакеті `Група.Прізвище.Lab2`;
   - клас має містити мінімум 3 поля, що є об'єктами класів, які описують складові частини предметної області;
   - клас має містити кілька конструкторів та мінімум 10 методів;
   - для тестування і демонстрації роботи розробленого класу розробити клас-драйвер;
   - методи класу мають вести протокол своєї діяльності, що записується у файл;
   - розробити механізм коректного завершення роботи з файлом (не надіятися на метод `finalize()`);
   - програма має володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету.
2. Автоматично згенерувати документацію до розробленої програми.
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.

Мій варіант №22: Автомат

Код файлу Gun.java:

```
package KI305RomaniukLab2;

import java.io.FileWriter;
import java.io.IOException;

/**
 * Клас, що представляє гвинтівку в предметній області.
 */
public class Gun {
    private String model;
    private double caliber;
    private boolean safetyOn;
    private int ammunitionCount;
    private FileWriter logFile;

    /**
     * Конструктор класу Gun.
     *
     * @param model     Модель гвинтівки.
     * @param caliber   Калібр гвинтівки.
     */
    public Gun(String model, double caliber) {
        this.model = model;
        this.caliber = caliber;
        this.safetyOn = true;
        this.ammunitionCount = 0;

        try {
            this.logFile = new FileWriter("gun_log.txt");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    /**
     * Конструктор класу Gun з вказаною кількістю набоїв.
```

```
     *
     * @param model             Модель гвинтівки.
     * @param caliber           Калібр гвинтівки.
     * @param ammunitionCount   Початкова кількість набоїв.
     */
    public Gun(String model, double caliber, int ammunitionCount) {
        this(model, caliber);
        this.ammunitionCount = ammunitionCount;
    }

    /**
     * Завантажує набої в гвинтівку.
     *
     * @param bullets Кількість набоїв для завантаження.
     */
    public void loadAmmunition(int bullets) {
        if (!safetyOn) {
            ammunitionCount += bullets;
            log("Loaded " + bullets + " bullets. Total ammunition: " +
ammunitionCount);
        } else {
            log("Cannot load ammunition with safety on.");
        }
    }

    /**
     * Виконує вистріл з гвинтівки.
     */
    public void fire() {
        if (!safetyOn && ammunitionCount > 0) {
            log("Fired one round. Ammunition left: " + --ammunitionCount);
        } else if (safetyOn) {
            log("Cannot fire with safety on.");
        } else {
            log("Out of ammunition.");
        }
    }

    /**
     * Увімкнути або вимкнути безпеку на гвинтівці.
     */
    public void toggleSafety() {
        safetyOn = !safetyOn;
        if (safetyOn) {
            log("Safety is on.");
        } else {
            log("Safety is off.");
        }
    }

    // Приватний метод для ведення журналу подій.
    private void log(String message) {
        try {
            logFile.write(message + "\n");
            logFile.flush();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    /**
     * Закрити файл журналу.
     */
    public void closeLogFile() {
        try {
            logFile.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
```

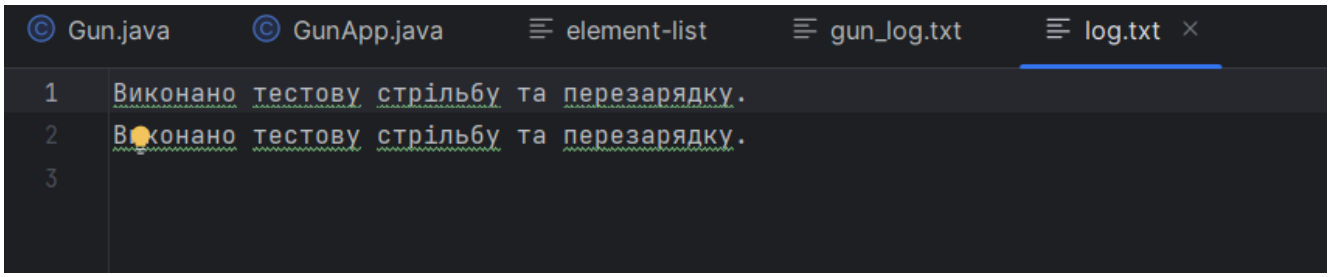
```
    }
}
```

Код файлу GunApp.java:

```
package KI305RomaniukLab2;

/**
 * Клас-драйвер для демонстрації роботи класу Gun.
 */

public class GunApp {
    public static void main(String[] args) {
        Gun ak47 = new Gun("AK-47", 7.62, 30);

        System.out.println("Safety is ON");
        ak47.toggleSafety();
        System.out.println("Safety is OFF");
        ak47.toggleSafety();

        System.out.println("Loading ammunition...");
        ak47.loadAmmunition(10);

        System.out.println("Firing...");
        ak47.fire();
        ak47.fire();
        ak47.fire();

        System.out.println("Loading more ammunition...");
        ak47.loadAmmunition(20);

        System.out.println("Firing...");
        ak47.fire();

        ak47.closeLogFile();
    }
}
```

Результат виконання програми:

```
"C:\Program Files\Java\jdk-20\bin\java.exe" "-java
Safety is ON
Safety is OFF
Loading ammunition...
Firing...
Loading more ammunition...
Firing...

Process finished with exit code 0
```




```
1  Safety is off.
2  Safety is on.
3  Cannot load ammunition with safety on.
4  Cannot fire with safety on.
5  Cannot fire with safety on.
6  Cannot fire with safety on.
7  Cannot load ammunition with safety on.
8  Cannot fire with safety on.
```

Згенерована документація:

allclasses-index.html	26.09.2023 16:24	Chrome HTML Do...	4 КБ
allpackages-index.html	26.09.2023 16:24	Chrome HTML Do...	3 КБ
copy.svg	26.09.2023 16:24	Microsoft Edge HT...	1 КБ
element-list	26.09.2023 16:24	Файл	1 КБ
gun_log.txt	26.09.2023 16:27	Текстовий докум...	1 КБ
help-doc.html	26.09.2023 16:24	Chrome HTML Do...	9 КБ
index.html	26.09.2023 16:24	Chrome HTML Do...	2 КБ
lab 2.iml	25.09.2023 18:36	Файл IML	1 КБ
log.txt	26.09.2023 1:27	Текстовий докум...	1 КБ
member-search-index.js	26.09.2023 16:24	JavaScript File	1 КБ
module-search-index.js	26.09.2023 16:24	JavaScript File	1 КБ
overview-tree.html	26.09.2023 16:24	Chrome HTML Do...	4 КБ
package-search-index.js	26.09.2023 16:24	JavaScript File	1 КБ
script.js	26.09.2023 16:24	JavaScript File	10 КБ
search.html	26.09.2023 16:24	Chrome HTML Do...	4 КБ
search.js	26.09.2023 16:24	JavaScript File	16 КБ
search-page.js	26.09.2023 16:24	JavaScript File	11 КБ
stylesheet.css	26.09.2023 16:24	Каскадна таблиц...	32 КБ
tag-search-index.js	26.09.2023 16:24	JavaScript File	1 КБ
type-search-index.js	26.09.2023 16:24	JavaScript File	1 КБ

Package KI305RomaniukLab2

package KI305RomaniukLab2

Classes

Class	Description
Gun	Клас, що представляє гвинтівку в предметній області.
GunApp	Клас-драйвер для демонстрації роботи класу Gun.

Package KI305RomaniukLab2

Class Gun

java.lang.Object
    KI305RomaniukLab2.Gun

public class **Gun**
extends Object

Клас, що представляє гвинтівку в предметній області.

Constructor Summary

Constructors

Constructor	Description
`Gun(String model, double caliber)`	Конструктор класу Gun.
`Gun(String model, double caliber, int ammunitionCount)`	Конструктор класу Gun з вказаною кількістю набоїв.

Method Summary

All Methods **Instance Methods** **Concrete Methods**

Modifier and Type	Method	Description
void	`closeLogFile()`	Закрити файл журналу.
void	`fire()`	Виконує вистріл з гвинтівки.
void	`loadAmmunition(int bullets)`	Завантажує набої в гвинтівку.
void	`toggleSafety()`	Увімкнути або вимкнути безпеку на гвинтівці.

void	toggleSafety()	Увімкнути або вимкнути безпеку на гвинтівці.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Details

Gun

```
public Gun(String model,
           double caliber)
```

Конструктор класу Gun.

Parameters:

`model` - Модель гвинтівки.

`caliber` - Калібр гвинтівки.

Gun

```
public Gun(String model,
           double caliber,
           int ammunitionCount)
```

Конструктор класу Gun з вказаною кількістю набоїв.

Parameters:

Конструктор класу Gun з вказаною кількістю набоїв.

Parameters:

`model` - Модель гвинтівки.

`caliber` - Калібр гвинтівки.

`ammunitionCount` - Початкова кількість набоїв.

Method Details

loadAmmunition

```
public void loadAmmunition(int bullets)
```

Завантажує набої в гвинтівку.

Parameters:

`bullets` - Кількість набоїв для завантаження.

fire

```
public void fire()
```

Виконує вистріл з гвинтівки.

toggleSafety

```
public void toggleSafety()
```

Увімкнути або вимкнути безпеку на гвинтівці.

closeLogFile

```
public void closeLogFile()
```

Закрити файл журналу.

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Package KI305RomaniukLab2

Class GunApp

java.lang.Object
    KI305RomaniukLab2.GunApp

```
public class GunApp
extends Object
```

Клас-драйвер для демонстрації роботи класу Gun.

Constructor Summary

Constructors

Constructor	Description
GunApp()	

Method Summary

All Methods | **Static Methods** | **Concrete Methods**

Modifier and Type	Method	Description
static void	main(String[] args)	

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Details

GunApp

```
public GunApp()
```

Method Details

main

```
public static void main(String[] args)
```

Відповіді на контрольні запитання

1. Синтаксис визначення класу.
Відповідь: Визначення класу в Java виглядає так:

```
public class ClassName {
    // Тіло класу
}
```

2. Синтаксис визначення методу.
Відповідь: Визначення методу в Java виглядає так:

```
access_modifier return_type method_name(parameters) {
    // Тіло методу
}
```

3. Синтаксис оголошення поля.
Відповідь: Оголошення поля в Java виглядає так:
access_modifier data_type field_name;

4. Як оголосити та ініціалізувати константне поле?
Відповідь: Константне поле оголошується з використанням ключового слова final, ініціалізується під час оголошення або в конструкторі класу. Наприклад:
```
public class MyClass {
   final int myConstantField = 10;
}
```

5. Які є способи ініціалізації полів?
Відповідь: Поля в Java можна ініціалізувати під час оголошення, в конструкторі класу або в блоці ініціалізації.

6. Синтаксис визначення конструктора.
Відповідь: Визначення конструктора в Java виглядає так:
```
public ClassName(constructor_parameters) {
   // Тіло конструктора
}
```

7. Синтаксис оголошення пакету.
Відповідь: Оголошення пакету в Java вказується на початку файлу і виглядає так:
package package_name;

8. Як підключити до програми класи, що визначені в зовнішніх пакетах?
Відповідь: Для підключення класів з інших пакетів використовують імпорт. Наприклад, import package_name.ClassName;.

9. В чому суть статичного імпорту пакетів?
Відповідь: Статичний імпорт дозволяє використовувати статичні методи і поля з імпортованих класів без звернення до їх імені класу. Наприклад, import static package_name.ClassName.*;.

10. Які вимоги ставляться до файлів і каталогів при використанні пакетів?
Відповідь: Файли і каталоги повинні мати імена, що відповідають ієрархії пакетів. Наприклад, якщо пакет має ім'я com.example, то класи цього пакету повинні зберігатися у каталозі com/example/.

Висновок: у ході виконання лабораторної роботи я ознайомилася з процесом розробки класів та пакетів мовою Java.