Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра ЕОМ

Звіт

з лабораторної роботи №3

з дисципліни: «Кросплатформенні засоби програмування»

на тему: «Класи та пакети»

Виконав: ст.гр. КІ-34

Степанов А. О.

Прийняв:

викл. каф. ЕОМ

Іванов Ю. С.

Львів 2022

Мета роботи: 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.Lab3;
   - 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. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації.
4. Дати відповідь на контрольні запитання.

Варіант 21

21. Пістолет

Лістинг програми:

Файл GunApp.java

```
/**
 * lab 3 package
 */
package KI34.Stepanov.Lab3;

import java.io.*;

/**
 * Gun Application class implements main method for Gun class possibilities
demonstration
 * @author Andriy Stepanov
 * @version 1.0
 */
public class GunApp {
    /**
     * @param args function parameter
     * @throws FileNotFoundException throw about non-existent file
     */
    public static void main(String[] args) throws FileNotFoundException {
        Gun gun = new Gun("Glock", 15, 5, 25, 75);
        gun.takeGun();
        gun.printInfo();
```

```
        gun.aim();
        gun.shoot();
        gun.shoot();
        gun.aim();
        gun.shoot();
        gun.reload();
        gun.aim();
        gun.aim();
        gun.shoot();
        gun.aim();
        gun.shoot();
        gun.increaseMaxAmmo(40);

        gun.printInfo();
        gun.decreaseMaxAmmo(15);
        gun.repair();
        gun.increaseMaxAmmo(30);
        gun.aim();
        gun.shoot();
        gun.putGun();

        gun.printInfo();
        gun.deleteInfo();

        gun.printInfo();
        gun.dispose();
    }
}
```

Файл Gun.java

```
/**
 * lab 3 package
 */
package KI34.Stepanov.Lab3;

import java.io.*;

/**
 * Class <code>Gun</code> implements gun
 40
 * @author Andriy Stepanov
 * @version 1.0
 */
public class Gun {
    private String gunName;
    private int damage;
    private int ammo;
    private int maxAmmo;
    private boolean isAim;
    private boolean isTaken;
    private int exploitation;
    private PrintWriter fout;

    /**
     * Constructor
     * @param gunName Gun's Name
     * @param damage Received damage
     * @param ammo The count of ammo
     * @param maxAmmo Maximum count of ammo
     * @param exploitation Exploitation
     * @throws FileNotFoundException throw about non-existent file
     */
    public Gun(String gunName, int damage, int ammo, int maxAmmo, int
exploitation) throws FileNotFoundException {
        this.gunName = gunName;
```

```
        this.damage = damage;
        this.ammo = ammo;
        this.maxAmmo = maxAmmo;
        this.isAim = false;
        this.isTaken = false;
        this.exploitation = exploitation;
        fout = new PrintWriter(new File("Log.txt"));
    }

    /**
     * Method returns gun's name
     * @return Gun's name
     */
    public String getGunName() {
        return gunName;
    }

    /**
     * Method sets the new gun's name
     * @param gunName The name of the gun
     */
    public void setGunName(String gunName) {
        this.gunName = gunName;
    }

    /**
     * Method returns gun's damage
     * @return Gun's damage
     */
    public int getDamage() {
        return damage;
    }

    /**
     * Method sets the new gun's damage
     * @param damage The damage of the gun
     */
    public void setDamage(int damage) {
        this.damage = damage;
    }

    /**
     * Method returns the count of the ammo
     * @return the count of the ammo
     */
    public int getAmmo() {
        return ammo;
    }

    /**
     * Method sets the count of the ammo
     * @param ammo The count of the ammo
     */
    public void setAmmo(int ammo) {
        this.ammo = ammo;
    }

    /**
     * Method returns the count of the ammo
     * @return the count of the ammo
     */
    public int getMaxAmmo() {
        return maxAmmo;
    }

    /**
```

```
     * Method sets maximum count of ammo
     * @param maxAmmo Maximum count of ammo
     */
    public void setMaxAmmo(int maxAmmo) {
        this.maxAmmo = maxAmmo;
    }

    /**
     * Method returns aiming the gun
     * @return aiming the gun
     */
    public boolean isAim() {
        return isAim;
    }

    /**
     * Method sets aiming the gun
     * @param aim Aiming the gun
     */
    public void setAim(boolean aim) {
        isAim = aim;
    }

    /**
     * Method returns exploitation
     * @return exploitation
     */
    public int getExploitation() {
        return exploitation;
    }

    /**
     * Method sets exploitation the gun
     * @param exploitation Exploitation the gun
     */
    public void setExploitation(int exploitation) {
        this.exploitation = exploitation;
    }

    /**
     * Method returns taking gun
     * @return taking gun
     */
    public boolean isTaken() {
        return isTaken;
    }

    /**
     * Method sets taking gun
     * @param taken Taking gun
     */
    public void setTaken(boolean taken) {
        isTaken = taken;
    }

    /**
     * Method simulates gun's shoot
     */
    public void shoot() {
        if (isTaken) {
            if (isAim) {
                if (ammo == 0) {
                    System.out.println("Немає патронів");
                    fout.print("Немає патронів\n");
                    isAim = false;
                } else {
```

```
                    System.out.println("Вистріл");
                    fout.print("Вистріл\n");
                    ammo -= 1;
                    exploitation -= 1;
                    isAim = false;
                }
            } else {
                System.out.println("Спершу потрібно прицілитися");
                fout.print("Спершу потрібно прицілитися\n");
            }
        } else {
            System.out.println("Візьміть пістолет");
            fout.print("Візьміть пістолет\n");
        }
    }

    /**
     * Method simulates gun's reload
     */
    public void reload() {
        ammo = maxAmmo;
        System.out.println("Перезарядка");
        fout.print("Перезарядка\n");
    }

    /**
     * Method simulates gun's aim
     */
    public void aim() {
        if (isAim) {
            System.out.println("Ви і так прицілені");
            fout.print("Ви і так прицілені\n");
        } else {
            System.out.println("Ви прицілилися");
            fout.print("Ви прицілилися\n");
            isAim = true;
        }
    }

    /**
     * Method simulates gun's repair
     */
    public void repair() {
        if (exploitation == 100) {
            System.out.println("Пістолет повністю робочий");
            fout.print("Пістолет повністю робочий\n");
        } else {
            exploitation = 100;
            System.out.println("Пістолет відремонтовано");
            fout.print("Пістолет відремонтовано\n");
        }
    }

    /**
     * Method simulates gun's taking
     */
    public void takeGun() {
        System.out.println("Ви взяли пістолет до рук");
        fout.print("Ви взяли пістолет до рук\n");
        isTaken = true;
    }

    /**
     * Method simulates gun's putting
     */
    public void putGun() {
```

```
        System.out.println("Ви положили пістолет");
        fout.print("Ви положили пістолет\n");
        isTaken = false;
    }

    /**
     * Method simulates gun's increasing maximum count of the ammo
     * @param count the count of the maximum ammo
     */
    public void increaseMaxAmmo(int count) {
        if (count <= maxAmmo) {
            System.out.println("Магазин менший для збільшення");
            fout.print("Магазин менший для збільшення\n");
        } else {
            maxAmmo = count;
            ammo = count;
            System.out.println("Магазин збільшений до " + maxAmmo);
            fout.print("Магазин збільшений до " + maxAmmo + "\n");
        }
    }

    /**
     * Method simulates gun's decreasing maximum count of the ammo
     * @param count the count of the maximum ammo
     */
    public void decreaseMaxAmmo(int count) {
        if (count >= maxAmmo) {
            System.out.println("Магазин більший для зменшення");
            fout.print("Магазин більший для зменшення\n");
        } else {
            maxAmmo = count;
            ammo = count;
            System.out.println("Магазин зменшений до " + maxAmmo);
            fout.print("Магазин зменшений до " + maxAmmo + "\n");
        }
    }

    /**
     * Method simulates deleting information about gun
     */
    public void deleteInfo() {
        this.gunName = "None";
        this.damage = 0;
        this.ammo = 0;
        this.maxAmmo = 0;
        this.isAim = false;
        this.isTaken = false;
        this.exploitation = 0;
        System.out.println("Інформація про пістолет видалена");
        fout.print("Інформація про пістолет видалена");
    }

    /**
     * Method simulates printing information about gun
     */
    public void printInfo() {
        System.out.println("\n");
        System.out.println("Імя " + gunName);
        System.out.println("Шкода " + damage);
        System.out.println("Боєприпаси " + ammo);
        System.out.println("Максимальна кількість боєприпасів " + maxAmmo);
        System.out.println("Експлуатація " + exploitation + "%" + "\n\n");

        fout.print("\n");
        fout.print("Імя " + gunName + "\n");
        fout.print("Шкода " + damage + "\n");
```

```
        fout.print("Боєприпаси " + ammo + "\n");
        fout.print("Максимальна кількість боєприпасів " + maxAmmo + "\n");
        fout.print("Експлуатація " + exploitation + "%" + "\n\n");
    }

    /**
     * Method releases used recourses
     */
    public void dispose()
    {
        fout.flush();
        fout.close();
    }
}
```

Результат виконання програми:

```
Ви взяли пістолет до рук


Імя Glock
Шкода 15
Боєприпаси 5
Максимальна кількість боєприпасів 25
Експлуатація 75%


Ви прицілилися
Вистріл
Спершу потрібно прицілитися
Ви прицілилися
Вистріл
Перезарядка
Ви прицілилися
Ви і так прицілені
Вистріл
Ви прицілилися
Вистріл
Магазин збільшений до 40


Імя Glock
Шкода 15
Боєприпаси 40
Максимальна кількість боєприпасів 40
Експлуатація 71%
```

Протокол діяльності в консолі

```
Імя Glock
Шкода 15
Боєприпаси 40
Максимальна кількість боєприпасів 40
Експлуатація 71%


Магазин зменшений до 15
Пістолет відремонтовано
Магазин збільшений до 30
Ви прицілилися
Вистріл
Ви положили пістолет


Імя Glock
Шкода 15
Боєприпаси 29
Максимальна кількість боєприпасів 30
Експлуатація 99%
```

Протокол діяльності в консолі

```
Інформація про пістолет видалена


Імя None
Шкода 0
Боєприпаси 0
Максимальна кількість боєприпасів 0
Експлуатація 0%



Process finished with exit code 0
```

Протокол діяльності в консолі

```
Ви взяли пістолет до рук

Імя Glock
Шкода 15
Боєприпаси 5
Максимальна кількість боєприпасів 25
Експлуатація 75%

Ви прицілилися
Вистріл
Спершу потрібно прицілитися
Ви прицілилися
Вистріл
Перезарядка
Ви прицілилися
Ви і так прицілені
Вистріл
Ви прицілилися
Вистріл
Магазин збільшений до 40

Імя Glock
Шкода 15
Боєприпаси 40
Максимальна кількість боєприпасів 40
Експлуатація 71%

Магазин зменшений до 15
Пістолет відремонтовано
Магазин збільшений до 30
Ви прицілилися
Вистріл
Ви положили пістолет

Імя Glock
Шкода 15
Боєприпаси 29
Максимальна кількість боєприпасів 30
Експлуатація 99%

Інформація про пістолет видалена
Імя None
Шкода 0
Боєприпаси 0
Максимальна кількість боєприпасів 0
Експлуатація 0%
```

Протокол діяльності у файлі

Package KI34.Stepanov.Lab3

package KI34.Stepanov.Lab3

Classes

Class	Description
Gun	Class Gun implements gun 40
GunApp	Gun Application class implements main method for Gun class possibilities demonstration

Згенерована документація

Class Gun

java.lang.Object
KI34.Stepanov.Lab3.Gun

public class **Gun**
extends Object

Class Gun implements gun 40

Version:

1.0

Author:

Andriy Stepanov

Constructor Summary

Constructors

Constructor	Description
Gun(String gunName, int damage, int ammo, int maxAmmo, int exploitation)	Constructor

Інформація про клас Gun

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
void	aim()	Method simulates gun's aim
void	decreaseMaxAmmo(int count)	Method simulates gun's decreasing maximum count of the ammo
void	deleteInfo()	Method simulates deleting information about gun
void	dispose()	Method releases used recourses
int	getAmmo()	Method returns the count of the ammo
int	getDamage()	Method returns gun's damage
int	getExploitation()	Method returns exploitation
String	getGunName()	Method returns gun's name
int	getMaxAmmo()	Method returns the count of the ammo
void	increaseMaxAmmo(int count)	Method simulates gun's increasing maximum count of the ammo
boolean	isAim()	Method returns aiming the gun
boolean	isTaken()	Method returns taking gun
void	printInfo()	Method simulates printing information about gun
void	putGun()	Method simulates gun's putting
void	reload()	Method simulates gun's reload
void	repair()	Method simulates gun's repair
void	setAim(boolean aim)	Method sets aiming the gun
void	setAmmo(int ammo)	Method sets the count of the ammo
void	setDamage(int damage)	Method sets the new gun's damage
void	setExploitation(int exploitation)	Method sets exploitation the gun
void	setGunName(String gunName)	Method sets the new gun's name
void	setMaxAmmo(int maxAmmo)	Method sets maximum count of ammo
void	setTaken(boolean taken)	Method sets taking gun
void	shoot()	Method simulates gun's shoot
void	takeGun()	Method simulates gun's taking

Methods inherited from class java.lang.Object

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

Інформація про клас Gun

Constructor Details

Gun

```
public Gun(String gunName,
           int damage,
           int ammo,
           int maxAmmo,
           int exploitation)
    throws FileNotFoundException
```

Constructor

Parameters:

`gunName` - Gun's Name

`damage` - Received damage

`ammo` - The count of ammo

`maxAmmo` - Maximum count of ammo

`exploitation` - Exploitation

Throws:

`FileNotFoundException` - throw about non-existent file

Інформація про клас Gun

Class GunApp

java.lang.Object
    KI34.Stepanov.Lab3.GunApp

```
public class GunApp
extends Object
```

Gun Application class implements main method for Gun class possibilities demonstration

Version:

1.0

Author:

Andriy Stepanov

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

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Інформація про клас GunApp

Constructor Details

GunApp

```
public GunApp()
```

Method Details

main

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

Parameters:

args - function parameter

Throws:

FileNotFoundException - throw about non-existent file

Інформація про клас GunApp

Відповіді на контрольні запитання:

1. **Синтаксис визначення класу?**

Синтаксис оголошення простого класу в мові Java має наступний вигляд:

```
[public] class НазваКласу
{
        [конструктори]
        [методи]
        [поля]
}
```

2. **Синтаксис визначення методу?**

Синтаксис оголошення методу наступний:

[СпецифікаторДоступу] [static] [final] Тип назваМетоду([параметри]) [throws класи]

{

[Тіло методу]

[return [значення]];

}

3. **Синтаксис оголошення поля?**

Синтаксис оголошення поля наступний:

[СпецифікаторДоступу] [static] [final] Тип НазваПоля [=ПочатковеЗначення];

4. **Як оголосити та ініціалізувати константне поле?**

Синтаксис оголошення та ініціалізування константного поля наступний:

[СпецифікаторДоступу] [final] Тип НазваПоля [= ПочатковеЗначення];

5. **Які є способи ініціалізації полів?**

Ініціалізацію полів при створенні об’єкту можна здійснювати трьома способами:

- у конструкторі;
- явно при оголошені поля;
- у блоці ініціалізації (виконується перед виконанням конструктора).

6. **Синтаксис визначення конструктора?**

Синтаксис оголошення конструктора:

[СпецифікаторДоступу] НазваКласу([параметри])
{
Тіло конструктора
}

7. **Синтаксис оголошення пакету?**

Синтаксис оператора package:

package НазваПакету{.НазваПідпакету};

8. **Як підключити до програми класи, що визначені в зовнішніх пакетах?**

Доступ до класів з інших пакетів можна отримати двома шляхами:

1. вказуючи повне ім’я пакету перед іменем кожного класу.
2. використовуючи оператор import, що дозволяє підключати як один клас так і всі загальнодоступні класи пакету, позбавляючи необхідності записувати імена класів з вказуванням повної назви пакету перед ними.

9. **В чому суть статичного імпорту пакетів?**

Статичний імпорт дозволяє не вживати явно назву класу при звертанні до статичного поля або методу класу.

10. **Які вимоги ставляться до файлів і каталогів при використанні пакетів?**

Використання пакетів вимагає, щоб файли і каталоги проекту та їх ієрархія були строго структурованими. Так назви пакету і його підпакетів мають співпадати з назвами каталогів, де вони розміщуються. Назви загальнодоступних класів мають співпадати з назвами файлів, де вони розміщуються. Ієрархія каталогів і файлів проекту має співпадати з ієрархією пакетів. Після компіляції ієрархія каталогів, де містяться файли класів, співпадає з ієрархією каталогів проекту.

Висновок:

На цій лабораторній роботі я ознайомився з процесом розробки класів та пакетів мовою Java