Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра ЕОМ

Звіт

З лабораторної роботи №3

Варіант – 21

З дисципліни: «Кросплатформенні засоби програмування»

На тему: «Класи та пакети»

Виконав: ст. гр. КІ-35

Сухан Д. В.

Перевірив: доцент кафедри ЕОМ

Іванов Ю.С.

Львів 2022

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

Завдання (варіант № 21)

Варіант завдання : “ Пістолет ”

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 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.

Текст програми

Pistol.java

```
package KI35.Sukhan.Lab3;

import java.io.*;


/**
* Клас Pistol реалізує пістолет
*
```

```
* @author Denys Sukhan
* @version 1.0
* @since version 1.0
*
*/
    public class Pistol{
        private Trigger trigger;
        private Equipment equipment;
        private Cartridge cartridge;
        private PrintWriter fout;
        private boolean fuse;
        private String model,company;
        private double caliber,shoot_range;

          /* Constructor
           * @throws FileNotFoundException
          */

        public Pistol() throws FileNotFoundException {
            model ="Desert Eagle";
            company="American weapons.inc";
            caliber=7.62;
            shoot_range=2.2;
            fuse=true;
            trigger = new Trigger();
            equipment = new Equipment();
            cartridge = new Cartridge();
            fout = new PrintWriter(new File("KI35/Sukhan/Lab3/MyFile.txt"));
        }

        /* Constructor
         * @param model - модель пістолета       * @param  caliber - калібр
пістолета       * @param  sight - приціл
         * @param company - коспанія виробник    * @param  shoot_range - дальність
пострілу * @param  flashlight - ліхтарик
         * @throws FileNotFoundException         * @param  laser - лазер
        */
```

```
        public Pistol(String model,String company,double caliber,double
shoot_range,boolean laser,boolean sight, boolean flashlight) throws
FileNotFoundException {
            this.model =model;
            this.company=company;
            this.caliber=caliber;
            this.shoot_range=shoot_range;
            fuse=true;
            trigger = new Trigger();
            equipment = new Equipment(laser,sight,flashlight);
            cartridge = new Cartridge();
            fout = new PrintWriter(new File("KI35/Sukhan/Lab3/MyFile.txt"));
        }

        /**
          * Method змінює положення запобіжника
        */
        public void change_fuse_state(){
         fuse = fuse==false? true:false;
         if(fuse){
                System.out.println("Fuse is changed to [turn on state]\nPlease [turn
off] fuse state to make shot.");
                fout.println("Fuse is changed to [turn on state]\nPlease [turn off]
fuse state to make shot.");
                fout.flush();
            }
            else{
                System.out.println("Fuse is changed to [turn off state]");
                fout.println("Fuse is changed to [turn off state]");
                fout.flush();
            }
        }

        /**
          * Method змінює положення курка
        */
        public void change_push_state(){
```

```
            boolean push = trigger.get_push();
            trigger.set_push(!push);
             if(!push){
                    System.out.println("Push is changed to [turn on state]");
                    fout.println("Push is changed to [turn on state]");
                    fout.flush();
                }
                else{
                    System.out.println("Push is changed to [turn off state]\nPlease
[turn on] push state to make shot.");
                    fout.println("Push is changed to [turn off state]\nPlease [turn on]
push state to make shot.");
                    fout.flush();
                }

            }

        /**
          * Method виконує постріл
        */
        public void shoot(){
            boolean push = trigger.get_push();
            int cur_bullets = cartridge.get_cur_bullets();
            if (push == true && cur_bullets > 0 && fuse == false){
                cartridge.set_cur_bullets(cur_bullets-1);
                trigger.set_push(false);
                System.out.println("Shot");
                fout.println("Shot");
                fout.flush();
            }
            else{
                System.out.println("Pistol can't shoot!");
                fout.println("Pistol can't shoot!");
                fout.flush();
            }
        }
```

```
        /**
          * Method перезаряджає обойму
        */
        public void reload(){
            int cur_bullets = cartridge.get_cur_bullets();
            int max_bullets = cartridge.get_max_bullets();
            if (cur_bullets == max_bullets){
                System.out.println("Pistol doesn't require reloading!");
                fout.println("Pistol doesn't require reloading!");
                fout.flush();
            }
            else{
               cartridge.set_cur_bullets(max_bullets);
               System.out.println("The gun is reloaded !");
               fout.println("The gun is reloaded !");
               fout.flush();
            }
        }

        /**
          * Method змінює комплектування лазером
        */
        public void switch_laser(){
            boolean laser = equipment.get_laser();
            System.out.println(laser==false? "Laser is [turned on] ": "Laser is [turned
on]");
            fout.println(laser==false? "Laser is [turned on] ": "Laser is [turned
on]");
            fout.flush();
            equipment.set_laser(!laser);
        }

        /**
          * Method змінює комплектування прицілом
        */
        public void switch_sight(){
            boolean sight = equipment.get_sight();
```

```
            System.out.println(sight==false? "Sight is [turned on] ": "Sight is [turned
on]");
            fout.println(sight==false? "Sight is [turned on] ": "Sight is [turned
on]");
            fout.flush();
            equipment.set_sight(!sight);
        }

        /**
          * Method змінює комплектування ліхтариком
        */
        public void switch_flashlight(){
            boolean flashlight = equipment.get_flashlight();
            System.out.println(flashlight==false? "Flashlight is [turned on] ":
"Flashlight is [turned on]");
            fout.println(flashlight==false? "Flashlight is [turned on] ": "Flashlight
is [turned on]");
            fout.flush();
            equipment.set_flashlight(!flashlight);
        }

        /**
          * Method об'єм магазина патронів
        */
        public void change_cartridge_capacity(int capacity){
            cartridge.set_max_bullets(capacity);
            cartridge.set_cur_bullets(capacity);
            System.out.println("Cartridge capacity was changed to :" + capacity);
            fout.println("Cartridge capacity was changed to :" + capacity);
            fout.flush();
        }

        /**
          * Method виводить інформацію про пістолет
        */
        public void get_info(){
```

```
            System.out.println("Pistol model (" + model+")\nMade in company (" +
company+")\nCaliber(" + caliber+")\nShoot range(" + shoot_range+")\nCurrent bullets(" +
cartridge.get_cur_bullets()+")\nCatrige capacity(" +
cartridge.get_max_bullets()+")\n");
            fout.println("Pistol model (" + model+")\nMade in company (" +
company+")\nCaliber(" + caliber+")\nShoot range(" + shoot_range+")\nCurrent bullets(" +
cartridge.get_cur_bullets()+")\nCatrige capacity(" +
cartridge.get_max_bullets()+")\n");
            fout.flush();
        }



    }

    class Trigger{
        private boolean push;
         /*
          * Constructor
         */
        public Trigger(){
            push = false;
        }
        public void set_push(boolean push){
            this.push = push;
         }

         public boolean get_push(){
            return push;
         }

    }

    class Equipment{
          private boolean laser,sight,flashlight;
           /*
            * Constructor
           */
```

```
        public Equipment(){
          laser=false;
          sight=false;
         flashlight=false;
        }
          /* Constructor
          * @param  shoot_range - дальність пострілу
          * @param  flashlight - ліхтарик
          * @param  laser - лазер
          */
        public Equipment(boolean laser,boolean sight, boolean flashlight){
          this.laser=laser;
          this.sight=sight;
          this.flashlight=flashlight;
        }

        public void set_laser(boolean laser){
          this.laser = laser;
       }
       public void set_sight(boolean sight){
          this.sight = sight;
       }
       public void set_flashlight(boolean flashlight){
          this.flashlight = flashlight;
       }
       public boolean get_laser(){
          return laser;
       }
       public boolean get_sight(){
          return sight;
       }
       public boolean get_flashlight(){
          return flashlight;
       }

}
```

```
class Cartridge {
    private int max_bullets;
    private int cur_bullets;
    /*
    * Constructor
    */
    public Cartridge(){
      max_bullets = 15;
      cur_bullets= max_bullets;
   }
 public void  set_max_bullets(int max_bullets){
    this.max_bullets = max_bullets;
 }

 public void set_cur_bullets(int cur_bullets){
    this.cur_bullets = cur_bullets;
 }

 public int get_max_bullets(){
    return max_bullets;
 }

 public int get_cur_bullets(){
    return cur_bullets;
 }

}
```

PistolAPP.java

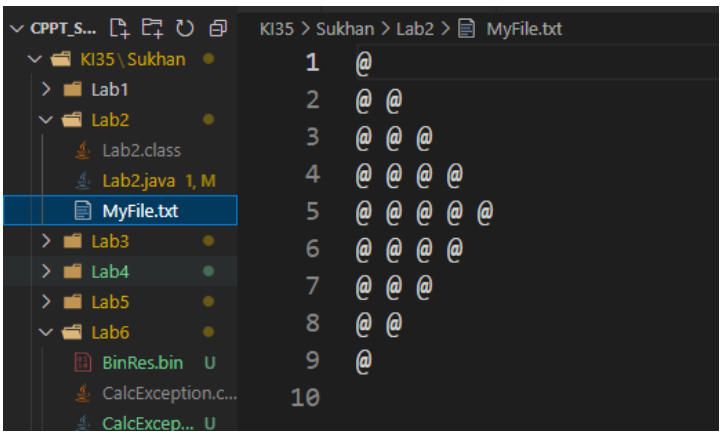
```
package KI35.Sukhan.Lab3;
import java.io.*;
import java.util.Scanner;
/**
* Клас Pistol реалізує програму-драйвер до пістолета
*
* @author Denys Sukhan
* @version 1.0
* @since version 1.0
*
*/
public class PistolAPP
{
  public static void main(String[] args) throws FileNotFoundException
   {
    String model,company;
    double caliber,shoot_range;
    boolean laser,sight,flashlight;
    Scanner in = new Scanner(System.in);

    System.out.print(" Enter the model of pistol:");
    model = in.nextLine();
    System.out.print(" Enter the name of the company that manufactured
the gun:");
    company = in.nextLine();

    System.out.print(" Enter the caliber of the gun: ");
    caliber = in.nextDouble();
    System.out.print(" Enter the firing range of the gun: ");
    shoot_range = in.nextDouble();

    System.out.print(" You want to add an accessory to your gun?  Chouse
the true or false : ");
    laser= in.nextBoolean();
    sight = laser ; flashlight = sight;
```

```
  Pistol pistol = new Pistol(model,company,caliber,shoot_range,laser,sight,flashlight);
    pistol.change_fuse_state();
    pistol.change_push_state();
    pistol.shoot();
    pistol.get_info();
    pistol.change_push_state();
    pistol.shoot();
    pistol.change_push_state();
    pistol.shoot();
    pistol.get_info();
    pistol.change_cartridge_capacity(25);
    pistol.change_push_state();
    pistol.shoot();
    pistol.get_info();
    pistol.reload();
    pistol.get_info();
    }
}
```

Результат виконання програми

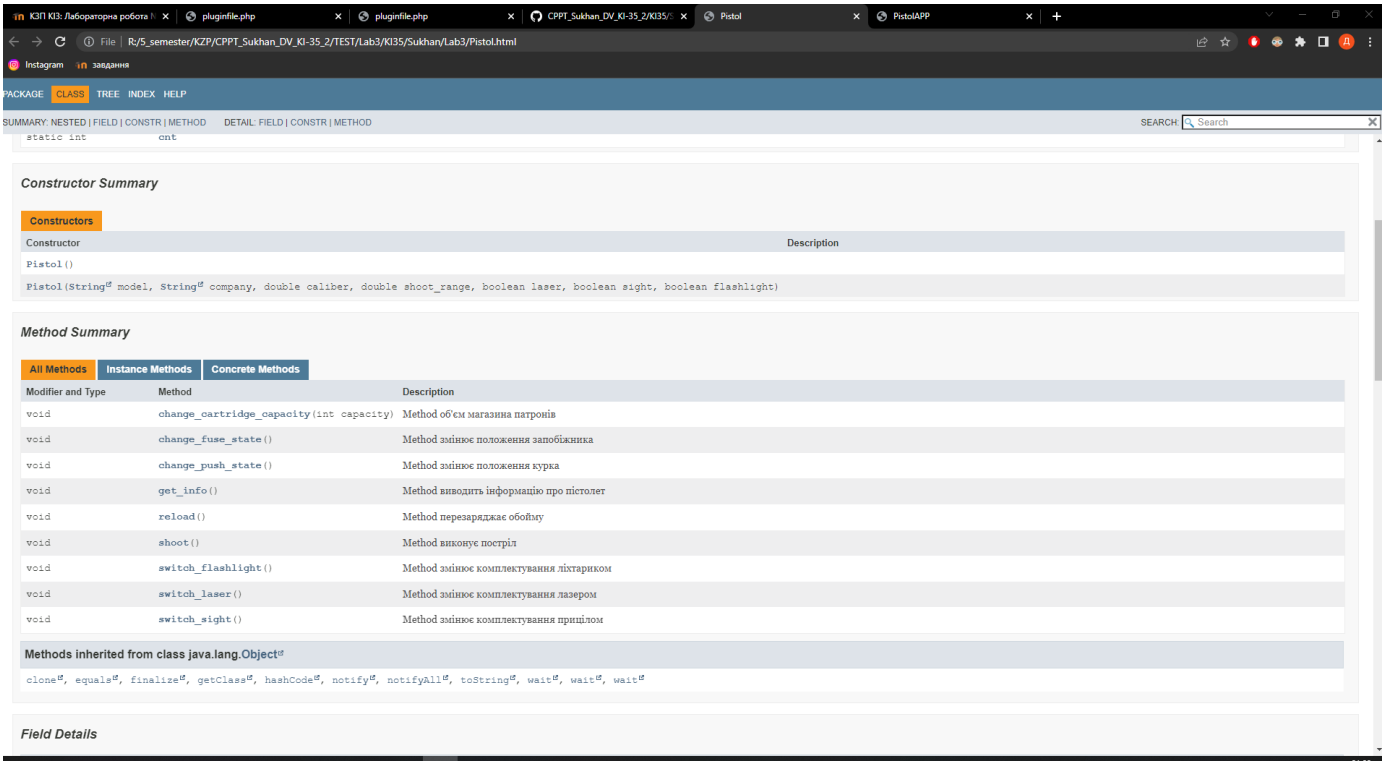
```
PS R:\5_semester\KZP\CPPT_Sukhan_DV_KI-35_2>  r:; cd 'r:\5_semester\KZP\CPPT_Sukhan_DV_KI-35_2'; &
\KZP\CPPT_Sukhan_DV_KI-35_2' 'KI35.Sukhan.Lab3.PistolAPP'
 Enter the model of pistol:Glock
 Enter the name of the company that manufactured the gun:American.WEAPONS.inc.
 Enter the caliber of the gun: 7,65
 Enter the firing range of the gun: 1,5
 You want to add an accessory to your gun?  Chouse the true or false : true
Fuse is changed to [turn off state]
Push is changed to [turn on state]
Shot
Pistol model (Glock)
Made in company (American.WEAPONS.inc.)
Caliber(7.65)
Shoot range(1.5)
Current bullets(14)
Catrige capacity(15)

Push is changed to [turn on state]
Shot
Push is changed to [turn on state]
Shot
Pistol model (Glock)
Made in company (American.WEAPONS.inc.)
Caliber(7.65)
Shoot range(1.5)
Current bullets(12)
Catrige capacity(15)

Cartridge capacity was changed to :25
Push is changed to [turn on state]
Shot
Pistol model (Glock)
Made in company (American.WEAPONS.inc.)
Caliber(7.65)
Shoot range(1.5)
Current bullets(24)
Catrige capacity(25)

The gun is reloaded !
Pistol model (Glock)
Made in company (American.WEAPONS.inc.)
Caliber(7.65)
Shoot range(1.5)
Current bullets(25)
Catrige capacity(25)
```

Вміст MyFile.txt



Фрагмент згенерованої документації

Pistol.html



Pistol

PACKAGE CLASS TREE INDEX HELP

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

static int cnt

Constructor Summary

Constructors

Constructor	Description
Pistol()	
Pistol(String model, String company, double caliber, double shoot_range, boolean laser, boolean sight, boolean flashlight)	

Method Summary

All Methods | Instance Methods | Concrete Methods

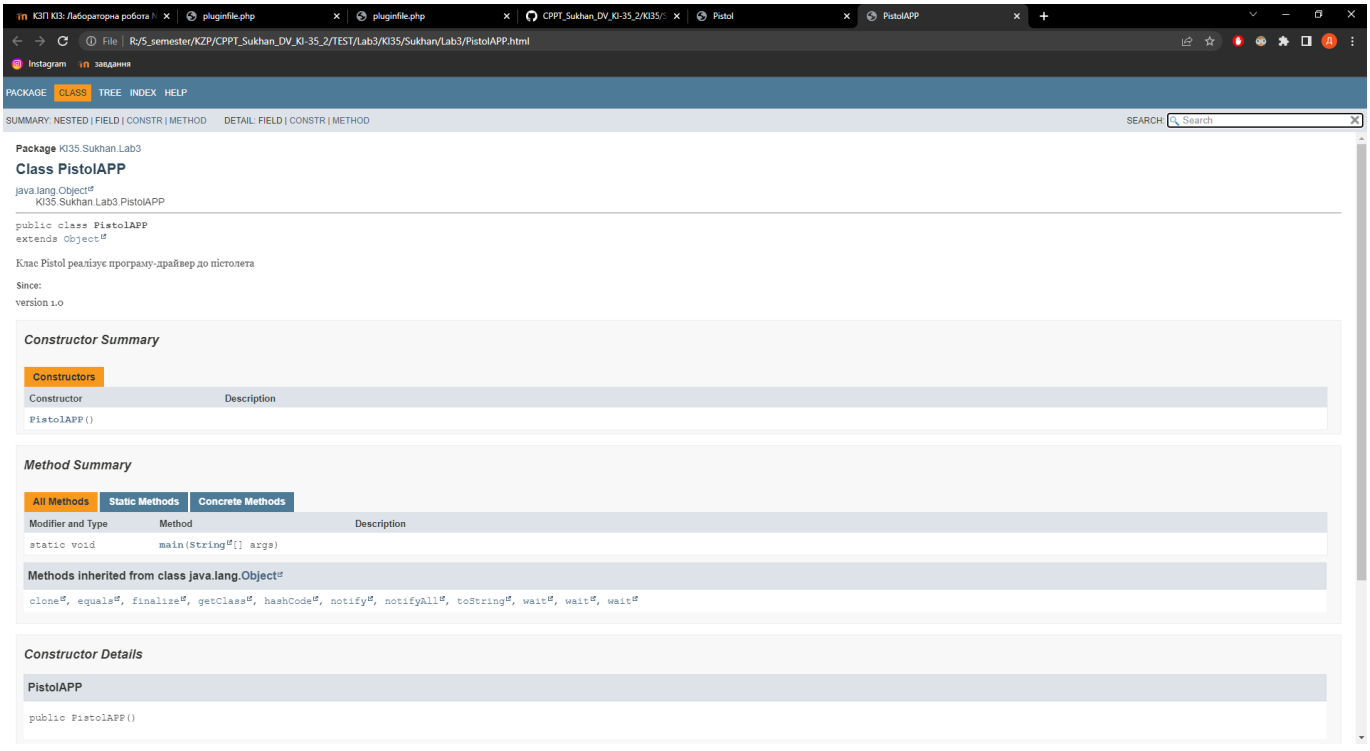
Modifier and Type	Method	Description
void	change_cartridge_capacity(int capacity)	Method об'єм магазина патронів
void	change_fuse_state()	Method змінює положення запобіжника
void	change_push_state()	Method змінює положення курка
void	get_info()	Method виводить інформацію про пістолет
void	reload()	Method перезаряджає обойму
void	shoot()	Method виконує постріл
void	switch_flashlight()	Method змінює комплектування ліхтариком
void	switch_laser()	Method змінює комплектування лазером
void	switch_sight()	Method змінює комплектування прицілом

Methods inherited from class java.lang.Object

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

Field Details

PistolAPP.html



PistolAPP

PACKAGE CLASS TREE INDEX HELP

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

Package KI35.Sukhan.Lab3

Class PistolAPP

java.lang.Object
KI35.Sukhan.Lab3.PistolAPP

public class PistolAPP
extends Object

Клас Pistol реалізує програму-драйвер до пістолета

Since:
version 1.0

Constructor Summary

Constructors

Constructor	Description
PistolAPP()	

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

PistolAPP

public PistolAPP()

Відповідь на контрольні запитання:

1. Синтаксис визначення класу.

Відповідь:

[public] class НазваКласу

{

[конструктори]

[методи]

[поля]

}

2. Синтаксис визначення методу.

Відповідь:

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

{

[Тіло методу]

[return [значення]];

}

3. Синтаксис оголошення поля.

Відповідь:

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

4. Як оголосити та ініціалізувати константне поле?

Відповідь:

[СпецифікаторДоступу] static final Тип НазваПоля = Значення;

- явно при оголошені поля класу;

- у статичному блоці ініціалізації.

5. Які є способи ініціалізації полів?

Віповідь:

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.

Відповідь:

[СпецифікаторДоступу] НазваКласу([параметри])

{

Тіло конструктора

}

7. Синтаксис оголошення пакету.

Відповідь:

package НазваПакету{.НазваПідпакету};

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

Відповідь:

вказуючи повне ім’я пакету перед іменем кожного класу або використовуючи оператор **import**

9. В чому суть статичного імпорту пакетів?

Відповідь:

Можливість імпортувати окремі статичні методи або поля класу

- **import static** НазваПакету{.НазваПідпакету}.НазваКласу.НазваСтатичногоМетодуАбоПоля;

- **import static** НазваПакету{.НазваПідпакету}.*;

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

Відповідь:

Для уникнення конфліктів імен не зловживати імпортом пакетів.

Висновок:

Під час виконання даної лабораторної роботи я на практиці вивчив синтаксис визначення класів, їх методів та полів і як краще ініціалізовувати їх. Вперше для себе стикнувся з поняттям пакети, в свою чергу вивчив їх створення та принцип роботи. Та дізнався про можливість статичного імпорту методів та полів красу, за допомогою ключового слова “static”.