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: «Класи та пакети»

Виконав:
студент групи КІ-34
Козлюк Д.С.
Прийняв:
доцент кафедри ЕОМ
Іванов Ю. С.

Львів – 2022

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

Завдання (варіант №8)

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 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.
   - Синтаксис визначення класу.
   - Синтаксис визначення методу.
   - Синтаксис оголошення поля.
   - Як оголосити та ініціалізувати константне поле?
   - Які є способи ініціалізації полів?
   - Синтаксис визначення конструктора.

- Синтаксис оголошення пакету.
- Як підключити до програми класи, що визначені в зовнішніх пакетах?
- В чому суть статичного імпорту пакетів?
- Які вимоги ставляться до файлів і каталогів при використанні пакетів?

Індивідуальне завдання: Фотоапарат

Текст програми

Camera.java

```
package KI34.Kozliuk.Lab3;

import java.io.*;
import java.text.DecimalFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.Scanner;

/**
 * Class <code>Camera</code> implements camera from lab3
 * @author Kozliuk Dmytro KI-34
 * @version 1.0
 */
public class Camera
{
    public boolean isCameraOn = true;
    Scanner scanner = new Scanner(System.in);
    private String name = "unknown";
    private String firm = "unknown";;
    private ZoomLever zoom;
    private Lens lens;
    private Flash flash;
    private Matrix matrix;
    private ArrayList<Photo> photo;
    private Memory memory;
    private PrintWriter fout;
    private SimpleDateFormat formatter;



    public void turnOffCamera()
    {
        isCameraOn = false;
        fout.write("Camera is turned off");
        fout.flush();
        fout.close();
    }
```

```
    /**
     * Default constructor to set params from keyboard to camera without
name and firm
     * @throws FileNotFoundException
     */

    public Camera() throws FileNotFoundException
    {

            fout = new PrintWriter(new File("MyLog.txt"));
            System.out.println("Enter params for zoom lever: ");

            System.out.print("max zooming >");
            var a = this.scanner.nextInt();
            System.out.print("name >");
            var b = this.scanner.next();
            fout.write("Zoom lever:");
            zoom = new ZoomLever(a, b);
            System.out.println("Enter params for lens:");
            System.out.println("aperture max, aperture min, focal
distance max , focal distance min (for new value separate with space)");
            System.out.print("> ");

            lens = new Lens(scanner.nextInt(), scanner.nextInt(),
scanner.nextInt(), scanner.nextInt());

            System.out.println("Enter params for matrix:");
            System.out.println("Name matrix and count of megapixels(for
new value separate with space)");
            System.out.print("> ");
            matrix = new Matrix(scanner.next(), scanner.nextInt());

            photo = new ArrayList<Photo>();

            System.out.println("Enter params for memory:");
            System.out.println("Size of memory and name memory(for new
value separate with space)");
            System.out.print("> ");
            memory = new Memory(scanner.nextInt(), scanner.next(),
photo);

            flash = new Flash();
        }

    /**
     * Constructor to read info about camera from txt file
     * @param name The file name
     * @throws IOException
     */
    public Camera(String name) throws IOException
    {
        /*this();
        this.name  = name;*/
```

```
        if(name.lastIndexOf(".txt") != -1 )
        {
            FileReader fr= new FileReader(name);
            Scanner scan = new Scanner(fr);
            ArrayList<String> arrayList = new ArrayList<String>();

            while (scan.hasNextLine()) {arrayList.add(scan.nextLine());}
            fr.close();

            this.firm = arrayList.get(0);
            this.name = arrayList.get(1);


            fout = new PrintWriter(new File("MyLog.txt"));

            zoom = new ZoomLever(Integer.parseInt(arrayList.get(2)),
arrayList.get(3));

            lens = new Lens
                    (
                    Integer.parseInt(arrayList.get(4)),
                    Integer.parseInt(arrayList.get(5)),
                    Integer.parseInt (arrayList.get(6)),
                    Integer.parseInt(arrayList.get(7))
                    );

            matrix = new Matrix(arrayList.get(8),
Integer.parseInt(arrayList.get(9)));

            photo = new ArrayList<Photo>();

            memory = new Memory(Integer.parseInt(arrayList.get(10)),
arrayList.get(11), photo);

            flash = new Flash();
        }else System.out.println("It is no format for txt file");
    }

    /**
     * Construct to set value to camera from keyboard
     * @param name The name of camera
     * @param firm The company that manufactures a camera
     * @throws FileNotFoundException
     */
    public Camera(String name, String firm) throws FileNotFoundException
    {
        this();
        this.name = name;
        this.firm = firm;
    }

    public String getName() {return name;}

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

```
    public String getFirm() {return firm;}

    public void setFirm(String firm) {this.firm = firm;}

    /**
     * Method to change lens on camera
     */
    void changeLens()
    {
        System.out.println("Enter params for lens " );
        System.out.print("focal distance max >");
        lens.setMaxFocalDistance(scanner.nextInt());
        System.out.print("aperture max >");
        lens.setMaxAperture(scanner.nextInt());
        System.out.print("aperture max >");
        lens.setMinAperture(scanner.nextInt());
        System.out.print("focal distance min >");
        lens.setMinFocalDistance(scanner.nextInt());
        System.out.println("Lens successfully changed");
    }

    void increaseMemory(int size)
{memory.setSizeOfMemory(memory.getSizeOfMemory() + size);}

    /**
     * Method to take photo
     */
    public void takePhoto()
    {
        photo.add(new Photo(new Date(), (1+ Math.random() * 15), "jpg"));
        memory.updateMemory(photo.get(photo.size()-1));
        if (memory.getInUsageMemory() > memory.getSizeOfMemory())
        {
            System.out.println("Can not save photo because of leak
memory");
            fout.write("Can not save photo because of leak memory\n");
            photo.remove(photo.get(photo.size()-1));
        }else
        {
            if(flash.isOff())
            {
            System.out.println("Photo is taken with on flash");
            fout.write("Photo is taken with on flash\n");
            }
            else
            {
                System.out.println("Photo is taken with off flash");
                fout.write("Photo is taken with off flash\n");
            }

        }
        fout.flush();
    }
```

```
    /**
     * Method to turn on flash on camera
     */
    public void turnONFlash()
    {
        flash.setOff(true);
        System.out.println("Flash is set on");
        fout.write("Flash is on\n");
        fout.flush();
    }

    /**
     * Method to turn off flash on camera
     */
    public void turnOffFlash()
    {
        flash.setOff(false);
        System.out.println("Flash is off");
        fout.write("Flash is  off\n");
        fout.flush();
    }

    /**
     * Method to increase zoom on camera
     * @param zoom The how much we want to increase zoom
     */
    public void increaseZoom(int zoom)
    {
        if(zoom + this.zoom.getZoom() > this.zoom.getMaxZoom() )
        {
            System.out.println("Can not to increase zoom because is out
of bound" + "\nThe max value is " + this.zoom.getMaxZoom());
            fout.write("Can not to increase zoom because is out of bound"
+ "\nThe max value is " + this.zoom.getMaxZoom() + '\n');
        }
        else
        {
            this.zoom.increaseZoom(zoom);
            fout.write("Zoom increased on " + zoom +  "x\n");
        }

        fout.flush();
    }

    /**
     * Method to decrease zoom on camera
     * @param zoom The how much we want to decrease zoom
     */
    public void decreaseZoom(int zoom)
    {
        if(this.zoom.getZoom() -  zoom  < 0) {
            System.out.println("Can not to decrease zoom because is out
of bound" + "\nThe min value is 1");
```

```
            fout.write("Can not to decrease zoom because is out of bound"
+ "\nThe min value is 1\n");
        }
        else {
            this.zoom.decreaseZoom(zoom);
            fout.write("Zoom decreased on " + zoom +  "x\n");
        }
    }

    /**
     * Set zoom to default value (1)
     */
    public void setZoomDefault()
    {
        zoom.setZoom(zoom.zoomDefault);
        fout.write("Setting up zoom to default value -> 1x\n");
        fout.flush();
    }

    /**
     * Method to get status of flash
     * @return The flash is on/off
     */
    public boolean getFlashInfo() {return flash.isOff();}

    /**
     * Method get info about current zoom
     */
    public void getZoomInfo()
    {
        System.out.println("Zoom is " +  zoom.getInfoZoom() +
                "and now have " + zoom.getZoom() + "x " + ". Max " +
zoom.getMaxZoom() + "x" );
        fout.write("Zoom is " +  zoom.getInfoZoom() +
                "and now have " + zoom.getZoom() + "x " + ". Max " +
zoom.getMaxZoom() + "x\n" );
        fout.flush();
    }

    /**
     *  Method to view all list of photos that have been taken by the
camera
     */
    public void viewListPhotos()
    {
        DecimalFormat dec = new DecimalFormat("#0.00");
        formatter = new SimpleDateFormat("dd-MM-yyyy HH:mm:ss");
        fout.write("Viewing list of photos: \n");
        System.out.println("Name    \t" + "Date                \t" +
"Size(Mb)\t" + "    Format\t");
        fout.write("Name    \t" + "Date                \t" +
"\tSize(Mb)\t" + "    Format\t\n");
        for (var photo : photo) {
```

```
            System.out.println(photo.name + '\t' +
formatter.format(photo.date) + "\t\t" + dec.format(photo.sizeMb) +
"\t\t\t" + photo.format);
            fout.write(photo.name + '\t' + formatter.format(photo.date) +
"\t\t" + dec.format(photo.sizeMb) + "\t\t\t" + photo.format + '\n');
        }

        fout.flush();
    }

    /**
     * Method to get status about cameras memory
     */
    public void getMemoryStatus()
    {
        System.out.println("Total memory: " + memory.getSizeOfMemory() +
"Mb");
        System.out.println("In Usage memory: " +
memory.getInUsageMemory() + "Mb");
        System.out.println("Remaining memory: " +
memory.getRemainMemory() + "Mb");
        fout.write("Total memory: " + memory.getSizeOfMemory() + "Mb\n");
        fout.write("In Usage memory: " + memory.getInUsageMemory() +
"Mb\n");
        fout.write("Remaining memory: " + memory.getRemainMemory() +
"Mb\n");
        fout.flush();

    }

    /**
     * Method to delete all photo and clear memory
     */
    public void clearMemory()
    {
        memory.setInUsageMemory(0);
        photo.clear();
        fout.write("Memory full cleaned up\n");
        fout.flush();
    }

    /**
     * Method for deleting picture by name
     * @param namePicture The picture name that we want to delete
     */
    public boolean deletePicture(String namePicture)
    {
        if (photo.removeIf(photos -> photos.name.equals(namePicture))){
            fout.write("Photo " + namePicture + " was removed\n");
            fout.flush();
            return true;
        }else
        {
            fout.write("Photo " + namePicture + " was not removed\n");
```

```
                fout.flush();
                return false;
            }

        }

        /**
         * Method to view current lens parameters
         */
        public void viewLens()
        {
            System.out.println("Now the camera has a lens with the following
parameters:");
            fout.write("Now the camera has a lens with the following
parameters:");
            System.out.println(
                    "Aperture max -> " + lens.getMaxAperture() +
                    "Aperture min -> " + lens.getMinAperture() +
                    "Focal distance max" + lens.getMaxFocalDistance() +
                    "Focal distance min" + lens.getMinFocalDistance());
            fout.write("Aperture max -> " + lens.getMaxAperture() +
                    "Aperture min -> " + lens.getMinAperture() +
                    "Focal distance max" + lens.getMaxFocalDistance() +
                    "Focal distance min" + lens.getMinFocalDistance());
            fout.flush();
        }


        /**
         * Method to get full memory size
         * @return Full size of memory in megabytes
         */
        int getFullMemory() {return memory.getSizeOfMemory();}

        /**
         * Method to get memory that used
         * @return Memory that used by photo
         */
        int getMemoryInUsage() {return memory.getInUsageMemory();}

        /**
         * Method to get how megapixels the camera have
         * @return Num of megapixels the camera
         */
        int getMegapixels() {return matrix.getMegaPixels();}

        String getMatrixName(){return matrix.getName();}
    }

    /**
     * Class <code>Lens</code> implements cameras lens from lab3
     * @author Kozliuk Dmytro KI-34
     * @version 1.0
     */
```

```
class Lens
{
    private double maxFocalDistance;
    private double minFocalDistance;
    private double minAperture;
    private double maxAperture;

    /**
     * Default constructor
     */
    public Lens()
    {
        System.out.println("Camera have not Lens. Please put on it");
    }

    /**
     * Constructor to set parameters for lens of camera
     * @param maxAperture
     * @param minAperture
     * @param maxFocalDistance
     * @param minFocalDistance
     */
    public Lens(double maxAperture, double minAperture, double
maxFocalDistance, double minFocalDistance)
    {
        this.maxAperture = maxAperture;
        this.minAperture = minAperture;
        this.maxFocalDistance = maxFocalDistance;
        this.minFocalDistance = minFocalDistance;
    }

    /**
     * Method to get max focal distance
     * @return
     */
    public double getMaxFocalDistance() {
        return maxFocalDistance;
    }

    /**
     * Method to set max focal distance
     * @param maxFocalDistance
     */
    public void setMaxFocalDistance(double maxFocalDistance) {
        this.maxFocalDistance = maxFocalDistance;
    }

    /**
     * Method to get min focal distance
     * @return
     */
    public double getMinFocalDistance() {
        return minFocalDistance;
    }
```

```java
    /**
     * Method to get min focal distance
     * @param minFocalDistance
     */
    public void setMinFocalDistance(double minFocalDistance) {
        this.minFocalDistance = minFocalDistance;
    }

    public double getMinAperture() {
        return minAperture;
    }

    public void setMinAperture(double minAperture) {
        this.minAperture = minAperture;
    }

    public double getMaxAperture() {
        return maxAperture;
    }

    public void setMaxAperture(double maxAperture) {
        this.maxAperture = maxAperture;
    }
}

/**
 * Class <code>Zoom</code> implements cameras zoom from lab3
 * @author Kozliuk Dmytro KI-34
 * @version 1.0
 */
class ZoomLever
{
    public int zoomDefault = 1;
    private int zoom;
    private int maxZoom;
    private String infoZoom;

    /**
     * Constructor to set default value to zoom
     */
    public ZoomLever()
    {
        this.zoom = zoomDefault;
        this.maxZoom = zoomDefault;
        infoZoom = "noname";
    }

    /**
     * Constructor to set max zooming level
     * @param zoom max zooming level
     */
    public ZoomLever(int zoom)
    {
```

```
        this.maxZoom = zoom;
        this.zoom = zoomDefault;
        infoZoom = "noname";
    }

    /**
     * @param zoom The max zooming level
     * @param infoZoom The info about zoom
     */
    public ZoomLever(int zoom, String infoZoom)
    {
        this(zoom);
        this.infoZoom = infoZoom;
    }

    public int getZoom()
    {
        return zoom;
    }

    public void setZoom(int zoom)
    {
        this.zoom = zoom;
    }

    /**
     * Method to increase zoom level
     * @param zoom
     */
    public void increaseZoom(int zoom)
    {
        this.zoom += zoom;
    }

    /**
     * Method to decrease zoom level
     * @param zoom
     */
    public void decreaseZoom(int zoom)
    {
        this.zoom -= zoom;
    }

    public String getInfoZoom()
    {
        return infoZoom;
    }

    public void setInfoZoom(String infoZoom)
    {
        this.infoZoom = infoZoom;
    }

    public int getMaxZoom() {
```

```
        return maxZoom;
    }

    public void setMaxZoom(int maxZoom) {
        this.maxZoom = maxZoom;
    }
}

/**
 * Class <code>Flash</code> implements cameras flashlight from lab3
 * @author Kozliuk Dmytro KI-34
 * @version 1.0
 */
class Flash
{
    //is flash off ?
    private boolean isOff;

    /**
     * Default constructor
     * By default flash is off
     */
    Flash() {isOff = false;}

    /**
     * Constructor
     * @param isOff The status of flashlight
     */
    Flash(boolean isOff) {this.isOff = isOff;}

    public boolean isOff() {return isOff;}

    public void setOff(boolean off) {isOff = off;}

}

/**
 * Class <code>Matrix</code> implements cameras matrix from lab3
 * @author Kozliuk Dmytro KI-34
 * @version 1.0
 */
class Matrix
{
    //type of cameras matrix
    private String name;
    private int megaPixels;

    /**
     * Constructor to set default value to matrix
     */
    public Matrix()
    {
        this.megaPixels = 0;
        this.name = null;
```

```
    }
    /**
     * Constructor to set type and count of megapixels of camera
     * @param name Type of matrix (CCD/CMOS)
     * @param megaPixels
     */
    public Matrix(String name, int megaPixels)
    {
        this.name = name;
        this.megaPixels = megaPixels;
    }

    public String getName() {return name;}

    public void setName(String name) {this.name = name;}
    public int getMegaPixels() {return megaPixels;}

    public void setMegaPixels(int megaPixels) {this.megaPixels =
megaPixels;}
}

/**
 * Class <code>Photo</code> implements cameras photo from lab3
 * @author Kozliuk Dmytro KI-34
 * @version 1.0
 */
class Photo
{
    //store count of taken photo
    private static Integer photoCount = 0;
    Date date;
    String format;
    Double sizeMb;
    String name;

    /**
     * Constructor
     * The default initialization
     */
    Photo()
    {
        date = new Date();
        format = null;
        sizeMb = (double) 0;
        name = null;
    }

    /**
     * Constrictor that initialize info about taken photo and increase
count of photo
     * @param date The data of photo taken
     * @param sizeMb The size of photo
     * @param format The format of photo (png, jpg, etc...)
     */
```

```
    Photo(Date date, Double sizeMb,  String format)
    {
        this.format = format;
        this.date = date;
        this.sizeMb = sizeMb;
        this.name = "untitled" + photoCount.toString();
        photoCount ++;
    }

}
/**
 * Class <code>Memory</code> implements cameras memory from lab3
 * @author Kozliuk Dmytro KI-34
 * @version 1.0
 */
class Memory
{
    private int sizeOfMemory;

    private int inUsageMemory;

    private String name;

    /**
     * Default constructor
     */
    public Memory()
    {
        this.sizeOfMemory = 0;
        this.name = null;
        inUsageMemory = 0;
    }

    /**
     * Constructor for filling memory by photos
     * @param sizeOfMemory The size of photo
     * @param name The name of photo
     * @param photoList The list of all photo that taken
     */
    public Memory(int sizeOfMemory, String name, ArrayList<Photo>
photoList)
    {
        this.name =  name;
        this.sizeOfMemory = sizeOfMemory;
        if(!photoList.isEmpty())
            for (var photo : photoList)
                inUsageMemory += photo.sizeMb;
        else inUsageMemory = 0;
    }

    public int getSizeOfMemory() {return sizeOfMemory;}

    public void setSizeOfMemory(int sizeOfMemory) {this.sizeOfMemory =
sizeOfMemory;}
```

```
    public String getName() {return name;}

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

    public int getInUsageMemory() {return inUsageMemory;}

    public void setInUsageMemory(int inUsageMemory) {this.inUsageMemory =
inUsageMemory;}

    /**
     * Method to update memory with new photo size
     * @param photo The object of class photo
     */
    public void updateMemory(Photo photo) {inUsageMemory  +=
photo.sizeMb;}

    /**
     * Method to get remaining memory in mb
     * @return remain capacity of memory in mb
     */
    public int getRemainMemory() {return this.sizeOfMemory -
this.inUsageMemory;}
}
```

CameraApp.java

```
package KI34.Kozliuk.Lab3;
import static java.lang.System.out;
import java.io.*;
import java.util.Scanner;

public class CameraApp
{
    public static void main(String[] args) throws IOException
    {
        Human Vasya = new Human();
        Vasya.takeCamera(new Camera("camerainfo.txt"));
    }
}


class Human
{
    private Scanner scanner;

    public void takeCamera(Camera camera)
    {
        scanner = new Scanner(System.in);
        while (camera.isCameraOn)
        {
            out.println("Please select what do you want to do");
            out.println("1. Take photo");
            out.println("2. Change lens");
```

```
                out.println("3. Turn on flash");
                out.println("4. Turn off flash");
                out.println("5. Increase zoom");
                out.println("6. Decrease zoom");
                out.println("7. Set default zoom");
                out.println("8. Is flash off?");
                out.println("9. What zoom level is now? ");
                out.println("10. View list of photo");
                out.println("11. What is my memory status?");
                out.println("12. Clear memory");
                out.println("13. Delete some picture by name");
                out.println("14. Get full size of memory");
                out.println("15. Get memory in usage");
                out.println("16. Get camera info");
                out.println("17. Get lens info");
                out.println("18. Turn off camera");

                out.print("Enter you choice > ");
                int choice = scanner.nextInt();
                switch (choice) {
                    case 1 -> {
                        camera.takePhoto();
                        out.println("\n-----------------------------------
");
                    }
                    case 2 -> {
                        camera.changeLens();
                        out.println("\n-----------------------------------
");
                    }
                    case 3 -> {
                        camera.turnONFlash();
                        out.println("\n-----------------------------------
");
                    }
                    case 4 -> {
                        camera.turnOffFlash();
                        out.println("\n-----------------------------------
");
                    }
                    case 5 -> {
                        out.print("How much to increase the zoom ? \nEnter
>");
                        camera.increaseZoom(scanner.nextInt());
                        out.println("\n-----------------------------------
");
                    }
                    case 6 -> {
                        out.print("How much to decrease the zoom ? \nEnter
>");
                        camera.decreaseZoom(scanner.nextInt());
                        out.println("\n-----------------------------------
");
                    }
```

```
                    case 7 -> {
                        camera.setZoomDefault();
                        out.println("\n-----------------------------------
");
                    }
                    case 8 -> {
                        if(camera.getFlashInfo()){
                            out.println("Flash is in");
                        }else
                        {
                            out.println("Flash is off");
                        }
                        out.println("\n-----------------------------------
");
                    }
                    case 9 -> {
                        camera.getZoomInfo();
                        out.println("\n-----------------------------------
");
                    }
                    case 10 -> {
                        camera.viewListPhotos();
                        out.println("\n-----------------------------------
");
                    }
                    case 11 -> {
                        camera.getMemoryStatus();
                        out.println("\n-----------------------------------
");
                    }
                    case 12 -> {
                        camera.clearMemory();
                        out.println("\n-----------------------------------
");
                    }
                    case 13 -> {
                        out.print("Picture name that you want to delete ?
\nEnter >");
                        var namePhoto = scanner.next();
                        if(camera.deletePicture(namePhoto))
                            out.println("Photo " + namePhoto + " was
successfully deleted");
                        else
                            out.println("Photo " + namePhoto + " was failed
to delete. Incorrect name");
                        out.println("\n-----------------------------------
");
                    }
                    case 14 -> {
                        int mem = camera.getFullMemory();
                        out.println("Full memory is " + mem + " mb");
                        out.println("\n-----------------------------------
");
                    }
```

```
                    case 15 -> {
                        int memInUse = camera.getMemoryInUsage();
                        out.println("\n-----------------------------------
");
                        out.println(memInUse + " mb");
                    }
                    case 16 -> {
                        String name = camera.getName();
                        String firm = camera.getFirm();
                        int mgPx = camera.getMegapixels();
                        var matrixName = camera.getMatrixName();
                        out.println("Name is " + name + "\nFirm is " + firm +
"\nMegapixels is " + mgPx + "\nMatrix is " + matrixName);
                        out.println("\n-----------------------------------
");
                    }
                    case 17 -> {
                        camera.viewLens();
                    }
                    case 18 -> {
                        camera.turnOffCamera();
                        out.println("\n------Camera is off---------");
                        out.println("\n------GOODBYE---------");
                    }

                    default -> {
                        out.println("Unknown command! Try again! ");
                        out.println("\n-----------------------------------
");
                    }
                }
            }
        }
}
```

camerainfo.txt

SONY
FX3 Body
20
Optical
1
5
18
55
CMOS
60
10000
MicroSD

Результат виконання програми

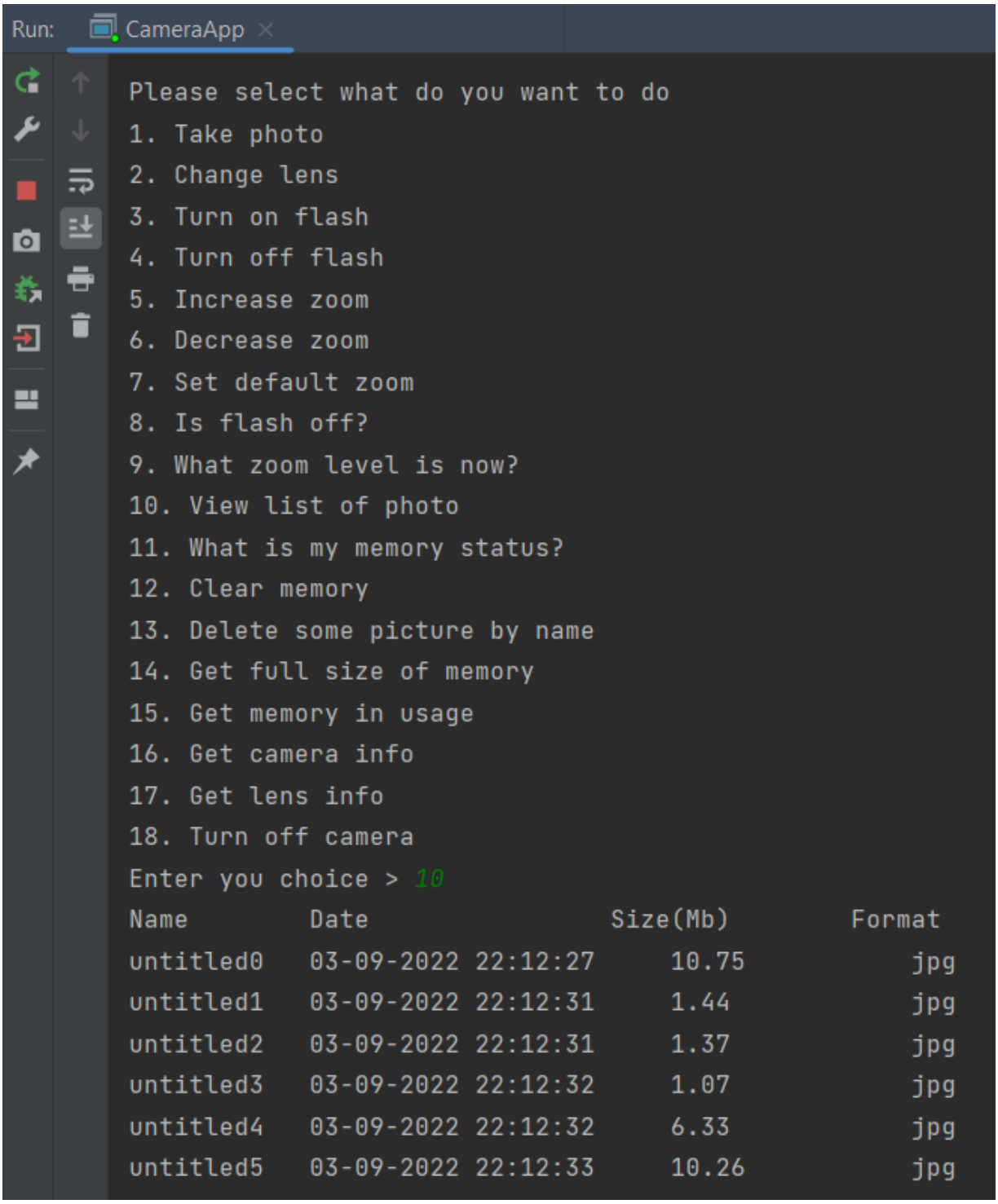


Рис.1.Фрагмент із результату виконання програми

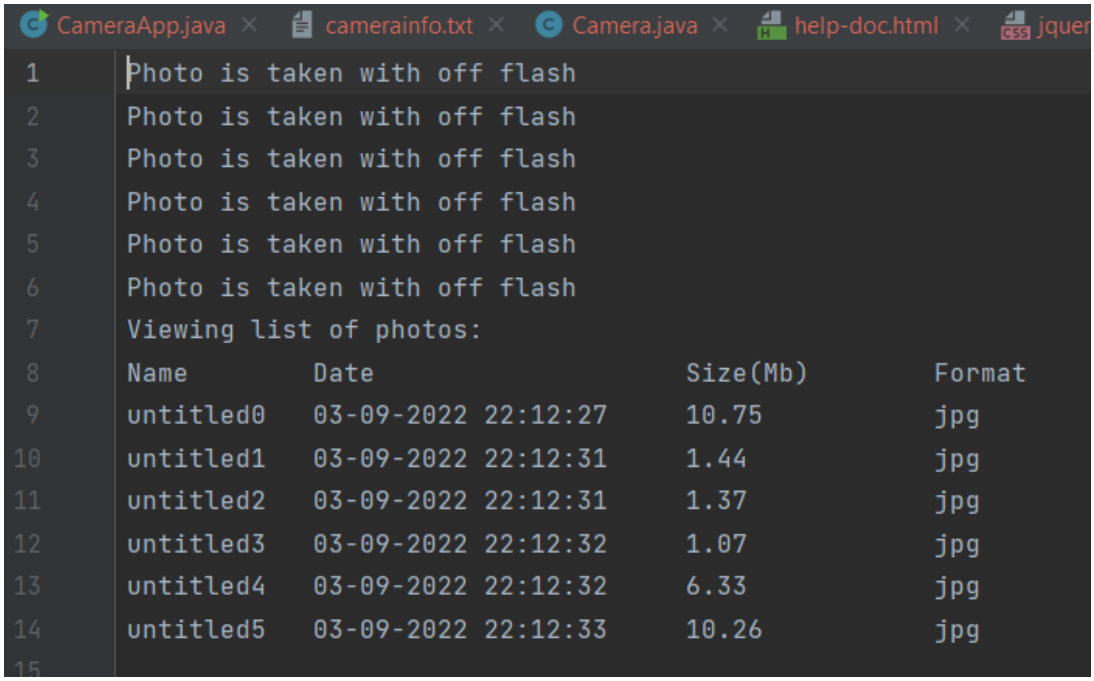


Рис.2.Результат введення протоколу діяльності виконання програми що записаний у файл

Фрагмент згенерованої документації

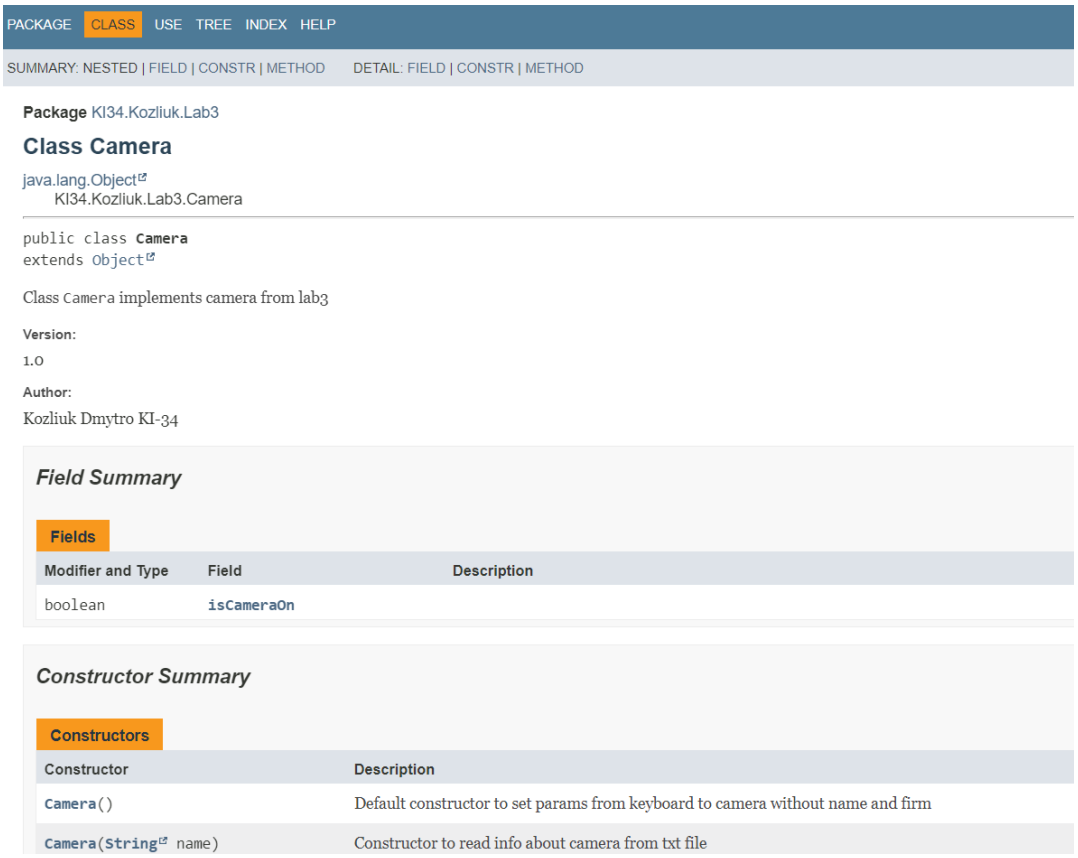
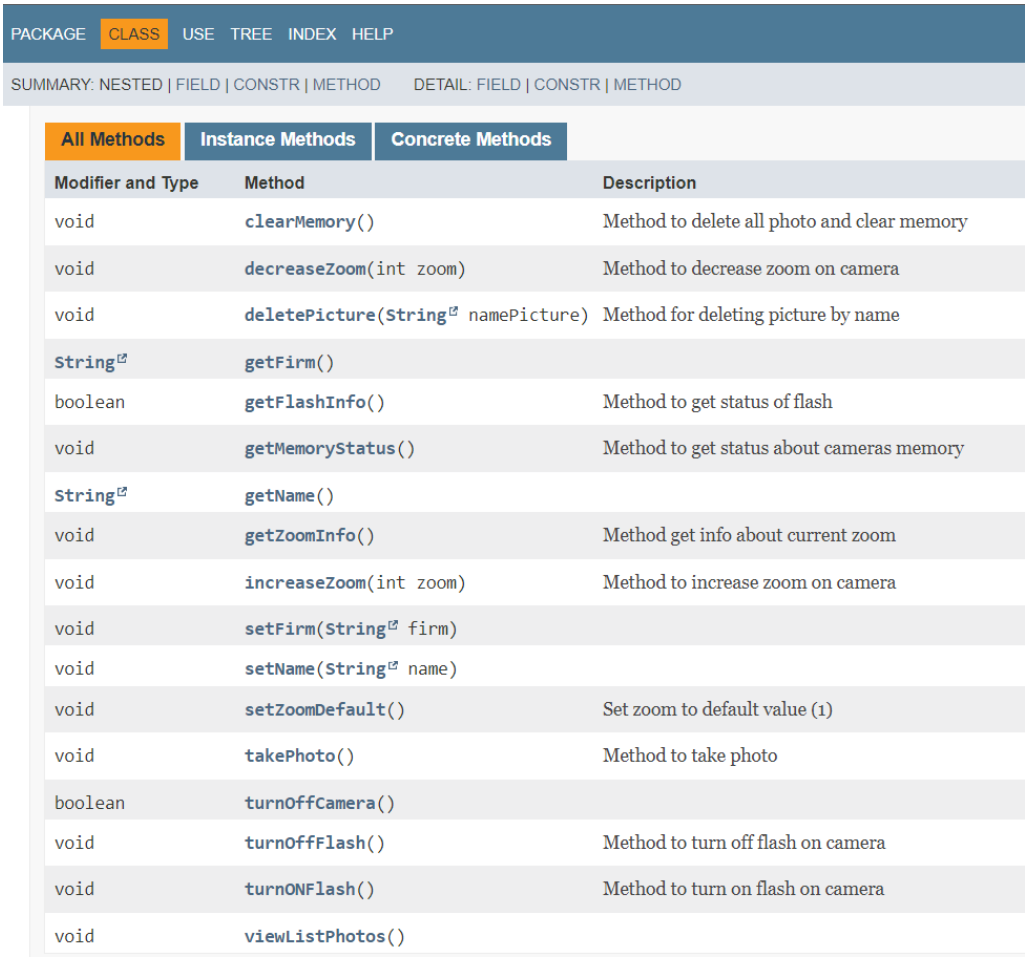


Рис.3.Фрагмент згенерованої документації №1



PACKAGE CLASS USE TREE INDEX HELP

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

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
void	clearMemory()	Method to delete all photo and clear memory
void	decreaseZoom(int zoom)	Method to decrease zoom on camera
void	deletePicture(String namePicture)	Method for deleting picture by name
String	getFirm()	
boolean	getFlashInfo()	Method to get status of flash
void	getMemoryStatus()	Method to get status about cameras memory
String	getName()	
void	getZoomInfo()	Method get info about current zoom
void	increaseZoom(int zoom)	Method to increase zoom on camera
void	setFirm(String firm)	
void	setName(String name)	
void	setZoomDefault()	Set zoom to default value (1)
void	takePhoto()	Method to take photo
boolean	turnOffCamera()	
void	turnOffFlash()	Method to turn off flash on camera
void	turnONFlash()	Method to turn on flash on camera
void	viewListPhotos()	

Рис.4.Фрагмент згенерованої документації №2

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];

5. Як оголосити та ініціалізувати константне поле?

Оголосити константне поле можна за допомогою ключового слова final:

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

private **final** int i = 5;

6. 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).

7. Синтаксис визначення конструктора.

Синтаксис оголошення конструктора:

[СпецифікаторДоступу] НазваКласу([параметри])

{

Тіло конструктора

}

8. Синтаксис оголошення пакету.

Синтаксис оператора package:

package НазваПакету{.НазваПідпакету};

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

Для підключення одного загальнодоступного класу пакету необхідно за допомогою оператора import через крапку вказати повну ієрархію пакету та назву класу, який має бути імпортовано, наприклад,

```
import java.util.Date
Date today = new Date();
```

Для підключення всіх загальнодоступних класів пакету необхідно за допомогою оператора import через крапку вказати повну ієрархію пакету та символ зірочка (*), наприклад,

```
import java.util.*
Date today = new Date();
```

10. В чому суть статичного імпорту пакетів?

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

11. 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

Під час виконання даної лабораторної роботи я ознайомився з процесом розробки класів та пакетів мовою Java. Розробив предметну область згідно свого варіанту (фотоапарат), розбивши її на 6 складових частин, ознайомився з одним із принципів ООП – інкапсуляція даних. Зрозумів наскільки ООП є корисним інструментом при підримці проєкту та його розширенні.