Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра ЕОМ



**Звіт**

З лабораторної роботи №2

Варіант№8

З дисципліни: «Кросплатформні засоби програмування»

На тему: «Класи та пакети»

Виконав: ст. гр. КІ-304

Кісіль В. І.

Прийняв:

Олексів М.В.

Львів 2023

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

**Теоретичні відомості**

**Зубчаті масиви**

Мова Java є повністю об’єктно-орієнтованою мовою програмування, тому вона дозволяє писати програми лише з використанням об’єктно-орієнтованих парадигм програмування, що базуються на понятті класів. Синтаксис оголошення простого класу в мові Java має наступний вигляд:

[public] class НазваКласу

{

[конструктори] [методи] [поля]

}

Метод – функція-член класу, яка призначена маніпулювати станом об’єкту класу. Методи можуть бути перевантаженими. Перевантаження методів відбувається шляхом вказування різної кількості параметрів та їх типів методам з однаковими назвами. Синтаксис оголошення методу наступний: [СпецифікаторДоступу] [static] [final] Тип назваМетоду([параметри]) [throws класи]

{

[Тіло методу] [return [значення]];

}

Конструктор – спеціальний метод класу, який не повертає значення, має ім’я класу та призначений для початкової ініціалізації об’єктів класу. Синтаксис оголошення конструктора:

[СпецифікаторДоступу] НазваКласу([параметри])

{

Тіло конструктора

}

Поле (властивість) – це дані-члени класу, що призначені для зберігання стану об’єкту. Поле може бути статичним (в цьому випадку воно називається полем класу), незмінним (константне поле), простим типом чи об’єктом та мати різні рівні доступу, що визначаються специфікатором доступу. Допускається ініціалізація поля в місці оголошення. Синтаксис оголошення поля наступний:

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

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.

1. Автоматично згенерувати документацію до розробленої програми.
2. Завантажити код на GitHub згідно методичних вказівок по роботі з GitHub.
3. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації та завантажити його у ВНС.
4. Дати відповідь на контрольні запитання.

**Варіант завдання:**

**8. Фотоапарат**

**Код програми:**

package ki304.Kisil.Lab2;

import java.io.\*;

public class CameraApp {

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

{

Camera Photocamera = new Camera();

Photocamera.makeAPhoto();

Photocamera.makeAPhoto();

Photocamera.viewStorage();

Photocamera.deleteAPhoto();

Photocamera.deleteAPhoto();

Photocamera.deleteAPhoto();

Photocamera.viewStorage();

Photocamera.viewButtonRes();

Photocamera.viewStorageSpace();

Photocamera.deleteAPhoto();

Photocamera.viewStorageSpace();

Photocamera.viewDistance();

Photocamera.distanceUp();

Photocamera.distanceUp();

Photocamera.distanceUp();

Photocamera.distanceUp();

Photocamera.distanceUp();

Photocamera.distanceUp();

Photocamera.distanceUp();

Photocamera.distanceUp();

Photocamera.distanceUp();

Photocamera.distanceUp();

Photocamera.distanceUp();

Photocamera.distanceUp();

Photocamera.distanceUp();

Photocamera.distanceUp();

Photocamera.distanceUp();

Photocamera.distanceUp();

Photocamera.distanceUp();

Photocamera.viewDistance();

Photocamera.viewFocusParameters();

Photocamera.makeAPhoto(15);

Photocamera.viewStorage();

Photocamera.dispose();

}

}

package ki304.Kisil.Lab2;

import java.io.\*;

/\*\*

\* Class <code>Camera</code> implements camera

\*/

public class Camera {

private Button PhotoButton;

private Storage PhotoStorage;

private FocusControl PhotoFocus;

private PrintWriter fout;

/\*\*

\* Constructor

\* @throws FileNotFoundException

\*/

public Camera() throws FileNotFoundException

{

PhotoButton = new Button();

PhotoStorage = new Storage();

PhotoFocus = new FocusControl();

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

}

/\*\*

\* Constructor

\* @param <code>resource</code> Button clicks photo

\* @throws FileNotFoundException

\*/

public Camera (int photo) throws FileNotFoundException

{

PhotoButton = new Button();

PhotoStorage = new Storage(photo);

PhotoFocus = new FocusControl();

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

}

/\*\*

\* Method simulate increasing focus distance

\*/

public void distanceUp()

{

if(PhotoFocus.scrollUp() == true)

{

System.out.print("Focus distance increased by 1 point"+"\n");

fout.print("Focus distance increased by 1 point"+"\n");

}

}

/\*\*

\* Method simulate increasing focus distance

\*/

public void distanceDown()

{

if(PhotoFocus.scrollDown() == true)

{

System.out.print("Focus distance decreased by 1 point"+"\n");

fout.print("Focus distance decreased by 1 point"+"\n");

}

}

/\*\*

\* Method simulate checking camera focus distance

\*/

public void viewDistance()

{

System.out.print("Distance now " + PhotoFocus.getFocusDistance() +"\n");

fout.print ("Distance now " + PhotoFocus.getFocusDistance() +"\n");

fout.flush();

}

/\*\*

\* Method simulate checking camera focus distance parameters

\*/

public void viewFocusParameters()

{

System.out.print("Max focus distance " + PhotoFocus.getMaxFocusDistance()

+"\n"+"Min focus distance " + PhotoFocus.getMinFocusDistance() +"\n");

fout.print ("Max focus distance " + PhotoFocus.getMaxFocusDistance()

+"\n"+"Min focus distance " + PhotoFocus.getMinFocusDistance() +"\n");

fout.flush();

}

/\*\*

\* Method simulate photographing

\*/

public void makeAPhoto()

{

if( PhotoButton.clickButton()==true)

{

if (PhotoStorage.makePhoto()==true)

{

System.out.print("You make a photo"+"\n");

fout.print("Photo deleted"+"\n");

}

}

}

/\*\*

\* Method simulate photographing many times

\*/

public void makeAPhoto(int amount)

{

for(int i = 0; i<amount;i++)

{

if( PhotoButton.clickButton()==true)

{

if (PhotoStorage.makePhoto()==true)

{

System.out.print("You make a photo"+"\n");

fout.print("Photo deleted"+"\n");

}

}

}

}

/\*\*

\* Method simulate deleting photo

\*/

public void deleteAPhoto()

{

if (PhotoStorage.deletePhoto() == true)

{

System.out.print("Photo deleted"+"\n");

fout.print("Photo deleted"+"\n");

}

}

/\*\*

\* Method simulate checking camera storage

\*/

public void viewStorage()

{

System.out.print("There are " + PhotoStorage.getPhotoNumber() +" photoes in storage"+"\n");

fout.print ("There are " + PhotoStorage.getPhotoNumber() +" photoes in storage"+"\n");

fout.flush();

}

/\*\*

\* Method simulate checking camera storage

\*/

public void viewStorageSpace()

{

System.out.print("There are " + PhotoStorage.getPhotoNumberLeft() +" space for photoes in storage"+"\n");

fout.print ("There are " + PhotoStorage.getPhotoNumberLeft() +" space for photoes in storage"+"\n");

fout.flush();

}

/\*\*

\* Method simulate checking button resources

\*/

public void viewButtonRes()

{

System.out.print("There are " + PhotoButton.getButtonResource() +" resources of button left"+"\n");

fout.print ("There are " + PhotoButton.getButtonResource() +" resources of button left "+"\n");

fout.flush();

}

/\*\*

\* Method releases used recourses

\*/

public void dispose()

{

fout.close();

}

}

package ki304.Kisil.Lab2;

/\*\*

\* Class <code>Button</code> implements camera button

\*/

public class Button

{

// Button clicks resource

private int btnResource;

/\*\*

\* Constructor

\*/

public Button()

{

btnResource = 100000;

}

/\*\*

\* Constructor

\* @param <code>res</code> Button clicks resource

\*/

public Button(int res)

{

btnResource = res;

}

/\*\*

\* Method simulates Button clicking

\*/

public boolean clickButton()

{if(btnResource>0)

{

btnResource--;

return true;

}

else {

System.out.print("Broken button"+"\n");

return false;}

}

/\*\*

\* Method returns button clicks resource available

\* @return Button clicks resource available

\*/

public int getButtonResource()

{

return btnResource;

}

}

package ki304.Kisil.Lab2;

/\*\*

\* Class <code>FocusControl</code> implement focus adjustment scroller

\*/

public class FocusControl

{

private int length;

private int maxlength;

private int minlength;

/\*\*

\* Constructor

\*/

public FocusControl()

{

length = 15;

minlength = 1;

maxlength = 30;

}

/\*\*

\* @param <code>len</code> Focus distance

\*/

public FocusControl(int len)

{

length = len;

minlength = 1;

maxlength = 30;

}

/\*\*

\* @param <code>len</code> Focus distance

\* @param <code>max</code> Max Focus distance

\*/

public FocusControl(int len, int max)

{

length = len;

minlength = 1;

maxlength = max;

}

/\*\*

\* Method simulate increasing focus distance

\*/

public boolean scrollUp()

{

if(length == maxlength)

{

System.out.print("Error (min length)"+"\n");

return false;}

else

{

length=length+1;

return true;

}

}

/\*\*

\* Method simulate decreasing focus distance

\*/

public boolean scrollDown()

{

if(length == minlength)

{

System.out.print("Error (min length)"+"\n");

return false;

}

else

{

length=length-1;

return true;

}

}

/\*\*

\* Method returns focus distance at the moment

\* @returns current focus distance

\*/

public int getFocusDistance()

{

return length;

}

/\*\*

\* Method returns max focus distance

\* @returns current focus distance

\*/

public int getMaxFocusDistance()

{

return maxlength;

}

/\*\*

\* Method returns min focus distance

\* @returns current focus distance

\*/

public int getMinFocusDistance()

{

return minlength;

}

}

package ki304.Kisil.Lab2;

/\*\*

\* <code>Storage</code> implement storage of camera

\*/

public class Storage

{

int photoNumber;

int maxNumber;

/\*\*

\* Constructor

\*/

public Storage()

{

photoNumber = 0;

maxNumber = 1000;

}

/\*\*

\* @param <code>num</code> numbers of photos

\*/

public Storage(int num)

{

photoNumber = num;

maxNumber = 10000;

}

/\*\*

\* @param <code>num</code> numbers of photos

\* @param<code>max</code> max numbers of photo

\*/

public Storage(int num, int max)

{

photoNumber = num;

maxNumber = max;

}

/\*\*

\* Method simulate deleting photo

\*/

public boolean deletePhoto()

{

if(photoNumber>0) {

photoNumber--;

return true;}

else

{System.out.print("Storage Error"+"\n");

return false;}

}

/\*\*

\* Method simulate making photo

\*/

public boolean makePhoto()

{if(photoNumber<maxNumber)

{

photoNumber++;

return true;

}

else

{

System.out.print("Storage Error"+"\n");

return false;

}

}

/\*\*

\* Method returns numbers of photos in storage

\* @return current number of photos

\*/

public int getPhotoNumber()

{

return photoNumber;

}

/\*\*

\* Method returns numbers of photos in storage

\* @return current number of photos

\*/

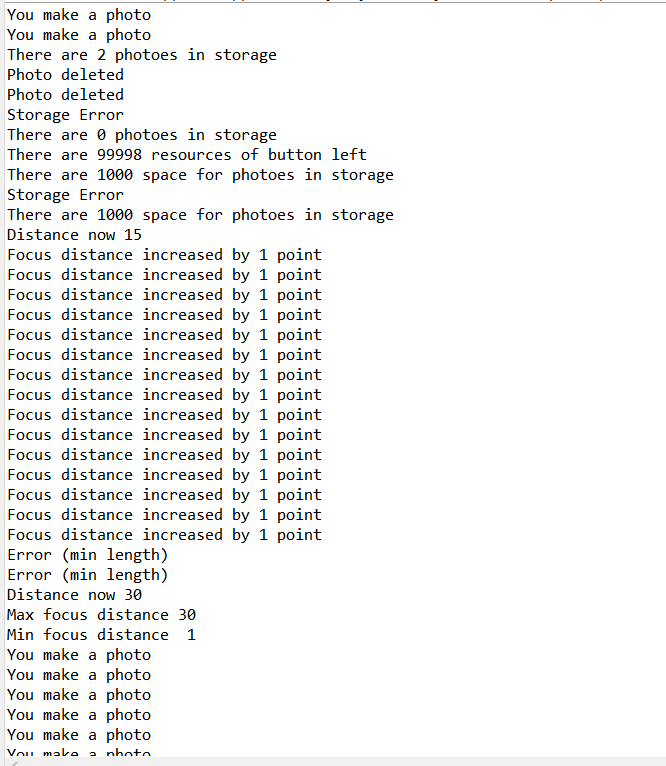
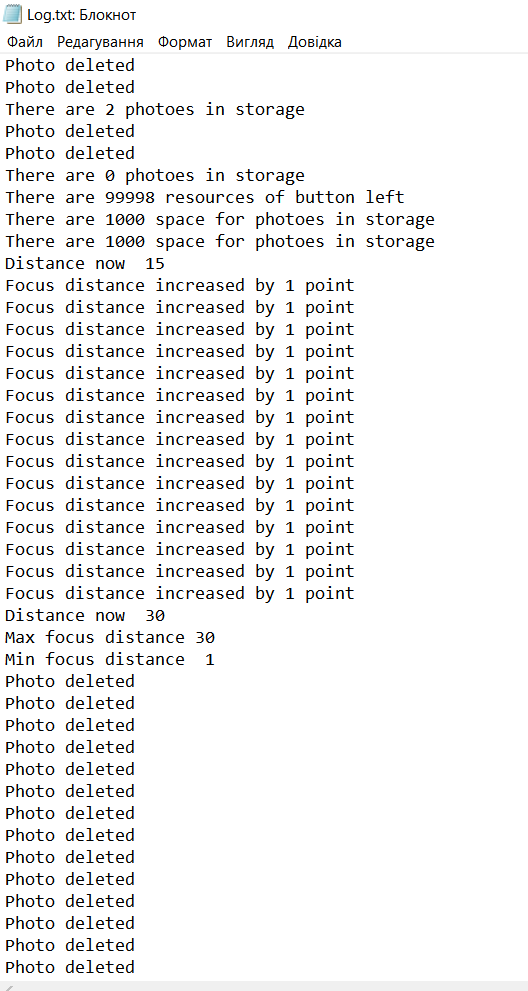
public int getPhotoNumberLeft()

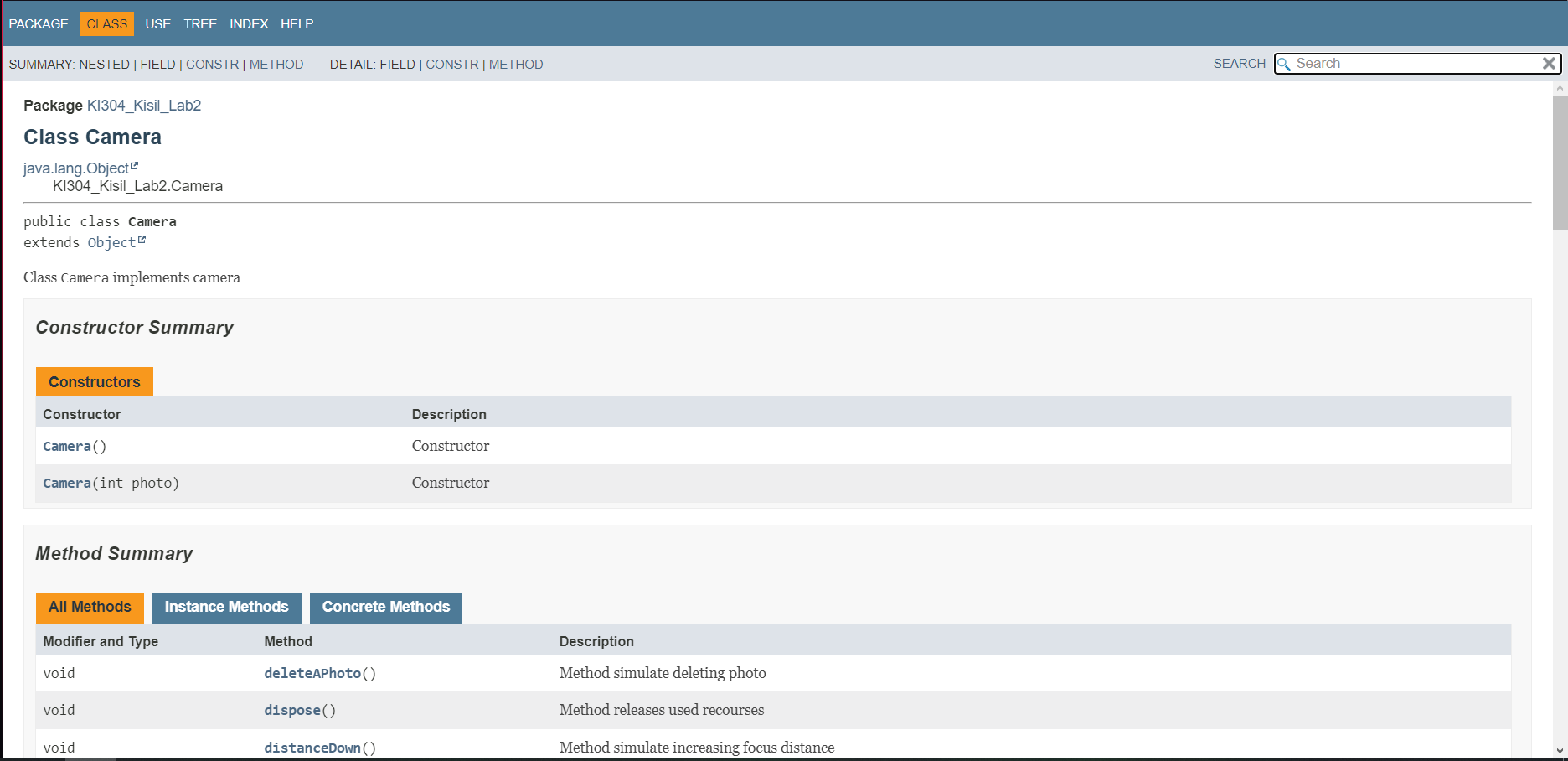
{

return maxNumber-photoNumber;

}

}

**Результат роботи програми:** ****

**Фрагмент згенерованої документації**

**Відповіді на контрольні запитання**

**Висновок:** Я ознайомився з процесом розробки класів та пакетів мовою Java.