Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра ЕОМ



**Звіт**

З лабораторної роботи №3

Варіант№8

З дисципліни: «Кросплатформні засоби програмування»

На тему: «СПАДКУВАННЯ ТА ІНТЕРФЕЙСИ»

Виконав: ст. гр. КІ-304

Кісіль В. І.

Прийняв:

Олексів М.В.

Львів 2023

**Мета роботи:** ознайомитися з спадкуванням та інтерфейсами у мові Java.

**Теоретичні відомості**

**Спадкування**

Спадкування в ООП призначене для розширення функціональності існуючих класів шляхом утворення нових класів на базі вже існуючих. У Java реалізована однокоренева архітектура класів згідно якої всі класи мають єдиного спільного предка (кореневий клас в ієрархії класів) – клас Object. Решта класів мови Java утворюються шляхом успадковування даного класу. Будь-яке спадкування у мові Java є відкритим, при цьому аналогів захищеному і приватному спадкуванню мови С++ не існує. На відміну від С++ у Java можливе спадкування лише одного базового класу (множинне спадкування відсутнє). Спадкування реалізується шляхом вказування ключового слова class піcля якого вказується назва підкласу, ключове слово extends та назва суперкласу, що розширюється у новому підкласі. Синтаксис реалізації спадкування:

class Підклас extends Суперклас

{ Додаткові поля і методи }

В термінах мови Java базовий клас найчастіше називається суперкласом, а похідний клас – підкласом. Дана термінологія запозичена з теорії множин, де підмножина міститься у супермножині.

При наслідувані у Java дозволяється перевизначення (перевантаження) методів та полів. При цьому область видимості методу, що перевизначається, має бути не меншою, ніж область видимості цього методу у суперкласі, інакше компілятор видасть повідомлення, про обмеження привілеїв доступу до даних.

Перевизначення методу полягає у визначені у підкласі методу з сигнатурою методу суперкласу. При виклику такого методу з-під об’єкта підкласу викличеться метод цього підкласу. Якщо ж у підкласі немає визначеного методу, що викликається, то викличеться метод суперкласу. Якщо ж у суперкласі даний метод також відсутній, то згенерується повідомлення про помилку. Перевизначення у підкласах елементів суперкласів (полів або методів) призводить до їх приховування новими елементами. Бувають ситуації, коли у методах підкласу необхідно звернутися до цих прихованих елементів суперкласів. У цій ситуації слід використати ключове слово super, яке вказує, що елемент до якого йде звернення, розташовується у суперкласі, а не у підкласі. Синтаксис звертання до елементів суперкласу:

super.назваМетоду([параметри]); // виклик методу суперкласу

super.назваПоля // звертання до поля суперкласу

Використання ключового слова super у конструкторах підкласів має дещо інший сенс, ніж у методах. Тут воно застосовується для виклику конструктора суперкласу. Виклик конструктора суперкласу має бути першим оператором конструктора підкласу. Конкретний конструктор, який необхідно викликати, вибирається по переданим параметрам. Явний виклик конструктора суперкласу часто є необхідним, оскільки підкласи не мають доступу до приватних полів суперкласів. Тож ініціалізація їх полів значеннями відмінними від значень за замовчуванням без явного виклику відповідного конструктора суперкласу є неможливою. Якщо виклик конструктора суперкласу не вказаний явно у підкласі або суперклас не має конструкторів, тоді автоматично викликається конструктор за замовчуванням суперкласу. Синтаксис виклику конструктора суперкласу з конструктора підкласу:

public НазваПідкласу([параметри])

{ super([параметри]); оператори конструктора підкласу }

**ЗАВДАННЯ**

1. Написати та налагодити програму на мові Java, що розширює клас, що реалізований у

лабораторній роботі №2, для реалізації предметної області заданої варіантом.

Суперклас, що реалізований у лабораторній роботі №2, зробити абстрактним.

Розроблений підклас має забезпечувати механізми свого коректного функціонування

та реалізовувати мінімум один інтерфейс. Програма має розміщуватися в пакеті

Група.Прізвище.Lab3 та володіти коментарями, які дозволять автоматично

згенерувати документацію до розробленого пакету.

2. Автоматично згенерувати документацію до розробленого пакету.

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

**Варіант завдання:**

**8. Цифрова відеокамера**

**Код програми:** Button.java

package ki\_304.Kisil.Lab3;

/\*\*

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

\*/

public class Button

{

final int STAND\_RES\_BTN = 100000;

// Button clicks resource

private int btnResource;

/\*\*

\* Constructor

\*/

public Button()

{

btnResource = STAND\_RES\_BTN;

}

/\*\*

\* 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;

}

}

Camera.java

package ki\_304.Kisil.Lab3;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.PrintWriter;

/\*\*

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

\*/

public abstract 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 abstract void distanceUp();

/\*\*

\* Method simulate increasing focus distance

\*/

public abstract void distanceDown();

/\*\*

\* Method simulate checking camera focus distance

\*/

public abstract void viewDistance();

/\*\*

\* Method simulate checking camera focus distance parameters

\*/

public abstract void viewFocusParameters();

/\*\*

\* Method simulate photographing

\*/

public abstract void makeAPhoto();

/\*\*

\* Method simulate photographing many times

\*/

public abstract void makeAPhoto(int amount);

/\*\*

\* Method simulate deleting photo

\*/

public abstract void deleteAPhoto();

/\*\*

\* Method simulate checking camera storage

\*/

public abstract void viewStorage();

/\*\*

\* Method simulate checking camera storage

\*/

public abstract void viewStorageSpace();

/\*\*

\* Method simulate checking button resources

\*/

public abstract void viewButtonRes();

/\*\*

\* Method releases used recourses

\*/

public abstract void dispose();

}

FocusControl.java

package ki\_304.Kisil.Lab3;

/\*\*

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

\*/

public class FocusControl

{

final int STANDART\_LENGTH = 15;

final int MIN\_STANDART\_LENGTH = 1;

final int MAX\_STANDART\_LENGTH = 30;

private int length;

private int maxlength;

private int minlength;

/\*\*

\* Constructor

\*/

public FocusControl()

{

length = STANDART\_LENGTH;

minlength = MIN\_STANDART\_LENGTH;;

maxlength = MAX\_STANDART\_LENGTH;

}

/\*\*

\* @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 = MIN\_STANDART\_LENGTH;

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;

}

}

IRecord.java

package ki\_304.Kisil.Lab3;

public interface IRecord {

void makeVideo();

}

Storage.java

package ki\_304.Kisil.Lab3;

public class Storage

{

final int STANDART\_NUMB\_PHOTO = 1000;

final int NO\_PHOTO = 0;

int photoNumber;

int maxNumber;

/\*\*

\* Constructor

\*/

public Storage()

{

photoNumber = NO\_PHOTO;

maxNumber = STANDART\_NUMB\_PHOTO;

}

/\*\*

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

\*/

public Storage(int num)

{

photoNumber = num;

maxNumber = STANDART\_NUMB\_PHOTO;

}

/\*\*

\* @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;

}

}

public boolean makeVid()

{if(photoNumber<maxNumber)

{

photoNumber=photoNumber+100;

if(photoNumber>maxNumber)

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

return false;}

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;

}

}

VideoCamera.java

package ki\_304.Kisil.Lab3;

import java.io.\*;

/\*\*

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

\*/

public class VideoCamera extends Camera implements IRecord {

private Button PhotoButton;

private Storage PhotoStorage;

private FocusControl PhotoFocus;

private PrintWriter fout;

/\*\*

\* Constructor

\* @throws FileNotFoundException

\*/

public VideoCamera() 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 VideoCamera (int photo) throws FileNotFoundException

{

PhotoButton = new Button();

PhotoStorage = new Storage(photo);

PhotoFocus = new FocusControl();

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

}

public void makeVideo()

{

if( PhotoButton.clickButton()==true)

{

if (PhotoStorage.makeVid()==true)

{

System.out.print("You record a video"+"\n");

fout.print("You record a video"+"\n");

}

}

}

/\*\*

\* 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("You make a photo"+"\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("You make a photo"+"\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();

}

}

VideoCameraAPP.java

package ki\_304.Kisil.Lab3;

import java.io.\*;

public class VideoCameraApp {

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

{

VideoCamera Photocamera = new VideoCamera();

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.makeVideo();

Photocamera.viewStorage();

Photocamera.dispose();

}

}

**Результат роботи програми:**

You make a photo

You make a photo

There are 2 photoes in storage

Photo deleted

Photo deleted

Storage Error

There are 0 photoes in storage

There are 99998 resources of button left

There are 1000 space for photoes in storage

Storage Error

There are 1000 space for photoes in storage

Distance now 15

Focus distance increased by 1 point

Focus distance increased by 1 point

Focus distance increased by 1 point

Focus distance increased by 1 point

Focus distance increased by 1 point

Focus distance increased by 1 point

Focus distance increased by 1 point

Focus distance increased by 1 point

Focus distance increased by 1 point

Focus distance increased by 1 point

Focus distance increased by 1 point

Focus distance increased by 1 point

Focus distance increased by 1 point

Focus distance increased by 1 point

Focus distance increased by 1 point

Error (min length)

Error (min length)

Distance now 30

Max focus distance 30

Min focus distance 1

You make a photo

You make a photo

You make a photo

You make a photo

You make a photo

You make a photo

You make a photo

You make a photo

You make a photo

You make a photo

You make a photo

You make a photo

You make a photo

You make a photo

You make a photo

You record a video

There are 115 photoes in storage

**Висновок:** Я ознайомився з спадкуванням та інтерфейсами у мові Java.