Міністерство освіти і науки України

Національний університет «Львівська політехніка»



**ЗВІТ**

з лабораторної роботи №3

з дисципліни: “Кросплатформні засоби програмування”

на тему: “Спадкування та інтерфейси”

Виконав:

ст. гр. КІ-305

Шевчук М.О.

Прийняв:

Іванов Ю.С.

Львів – 2023

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

**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 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.

**28. Енергозберігаюча лампочка**

**Лістинг програми:**

**Lab3ShevchoukKI305.java**

package pack;  
  
import java.io.FileNotFoundException;  
  
*/\*\**  
 *\* Class App realize example program to lab work 3*  
 *\**   
 *\* @author Shevchouk Maxim*  
 *\* @version 1.1*  
 *\* @since version 1.0*  
 *\*/*  
public class Lab3ShevchoukKI305 {  
 */\*\**  
 *\* @param args*  
 *\* @throws FileNotFoundException*  
 *\*/*  
public static void main(String[] args) throws Exception {  
 EconomLamp lamp = new EconomLamp(23232232, 30);  
 lamp.getInfoLamp();  
 lamp.repaintLight("Red");  
 lamp.resizeLamp(7);  
 lamp.getInfoLamp();  
 lamp.getLightPower();  
 lamp.powerOn();  
 lamp.powerOff();  
 lamp.lampWorking(30);  
 lamp.getPower();  
 lamp.setPower(45);  
 lamp.getPower();  
 lamp.LampGlowing();  
 }  
}

**Lamp.java**

package pack;  
  
import java.io.\*;  
  
*/\*\**  
 *\* Class <code>StoveForCooking</code> implements stove for cooking*  
 *\*/*  
abstract class Lamp {  
 protected Type TypeOfLamp;  
 protected Light LightOfLamp;  
 protected Button PowerOn;  
 protected Button PowerOff;  
 protected PrintWriter fout;  
 static int *power*;  
  
 */\*\**  
 *\* Method change color of light lamp*  
 *\*/*  
public void repaintLight(String uCollor) {  
 TypeOfLamp.setColor(uCollor);  
 System.*out*.println("New color of light is " + uCollor + ".");  
 fout.println("New color of light is " + uCollor + ".");  
 fout.flush();  
 }  
  
 */\*\**  
 *\* Method change radius of lamp*  
 *\*/*  
public void resizeLamp(double uRadius) {  
 TypeOfLamp.setRadius(uRadius);  
 System.*out*.println("New radius of lamp is " + uRadius + " cm.");  
 fout.println("New radius of lamp is " + uRadius + " cm.");  
 fout.flush();  
 }  
  
 */\*\**  
 *\* Method provides information about the appearance of the lamp*  
 *\*/*  
public void getRadiusLamp() {  
 System.*out*.println("Radius of lanp: " + TypeOfLamp.getRadius() + " cm.");  
 fout.println("Radius of lanp: " + TypeOfLamp.getRadius() + " cm.");  
 fout.flush();  
 }  
  
 */\*\**  
 *\* Method provides information about the light power of Lamp*  
 *\*/*  
public void getLightPower() {  
 System.*out*.println("Light power is " + LightOfLamp.getPower() + " W.");  
 fout.println("Light power is " + LightOfLamp.getPower() + " W.");  
 fout.flush();  
 }  
  
  
 abstract void getInfoLamp();  
  
  
 */\*\**  
 *\* The method provides information about the remaining lamp life*  
 *\*/*  
public void PowerOnResource() {  
 System.*out*.println("The lamp will work for a while " + PowerOn.getResource() + " time units.");  
 fout.println("The lamp will work for a while " + PowerOn.getResource() + " time units.");  
 fout.flush();  
 }  
  
 */\*\**  
 *\* The method provides information about the remaining lamp life*  
 *\*/*  
public void PowerOffResource() {  
 System.*out*.println("The lamp will work for a while " + PowerOff.getResource() + " time units.");  
 fout.println("The lamp will work for a while " + PowerOff.getResource() + " time units.");  
 fout.flush();  
 }  
  
 */\*\**  
 *\* Method simulate the power on tap*  
 *\*/*  
public void powerOn() {  
 PowerOn.clickButton(true);  
 System.*out*.println("You tap the power on button.");  
 fout.println("You tap the power on button.");  
 fout.flush();  
 }  
  
 */\*\**  
 *\* Method simulate the power off tap*  
 *\*/*  
public void powerOff() {  
 PowerOff.clickButton(false);  
 System.*out*.println("You tap the power off button.");  
 fout.println("You tap the power off button.");  
 fout.flush();  
 }  
  
 */\*\**  
 *\* The method provides information about the number of lamp switching on and off*  
 *\*/*  
public void OnOffCount() {  
 System.*out*.println("You tap the on/off button " + PowerOn.getCountClicks());  
 fout.println("You tap the on/off button " + PowerOn.getCountClicks());  
 fout.flush();  
 }  
}  
  
  
  
  
  
  
  
*/\*\**  
 *\* Class <code>Type</code> implements type of lamp*  
 *\**   
 *\* @author Shevchouk Maxim*  
 *\* @version 1.0*  
 *\*/*  
class Type {  
 private String LightColor;  
 private double radius;  
  
 */\*\**  
 *\* Constructor*  
 *\*/*  
public Type() {  
 LightColor = "White";  
 radius = 5;  
 }  
  
 */\*\**  
 *\* Method sets new light color of lamp*  
 *\**   
 *\* @param <code>newColor</code> The color value*  
 *\*/*  
public void setColor(String newLightColor) {  
 LightColor = newLightColor;  
 }  
  
 */\*\**  
 *\* Method sets new radius of type of lamp*  
 *\**   
 *\* @param <code>newSize</code> The Size value*  
 *\*/*  
public void setRadius(double newRadius) {  
 radius = newRadius;  
 }  
  
 */\*\**  
 *\* Method gets light color of type of lamp*  
 *\*/*  
public String getColor() {  
 return LightColor;  
 }  
  
 */\*\**  
 *\* Method gets radius of type of radius in cm*  
 *\*/*  
public double getRadius() {  
 return radius;  
 }  
  
 */\*\**  
 *\* Method gets size of type of lamp in cm^2*  
 *\*/*  
public double getSize() {  
 return 3.14 \* radius \* radius;  
 }  
}  
  
  
  
  
  
  
  
  
*/\*\**  
 *\* Class <code>Light</code> implements Light of stove for cooking*  
 *\**   
 *\* @author Shevchouk Maxim*  
 *\* @version 1.0*  
 *\*/*  
class Light {  
 private double power;  
  
 */\*\**  
 *\* Constructor*  
 *\*/*  
public Light() {  
 power = 20;  
 }  
  
 */\*\**  
 *\* Constructor*  
 *\**   
 *\* @param <code>uPower</code> The power value*  
 *\*/*  
public Light(int uPower) {  
 power = uPower;  
 }  
  
 */\*\**  
 *\* Method returns power*  
 *\**   
 *\*/*  
public double getPower() {  
 return power;  
 }  
  
 */\*\**  
 *\* Method sets new power of lamp*  
 *\**   
 *\* @param <code>newPower</code> The power value*  
 *\*/*  
public void setPower(double newPower) {  
 power = newPower;  
 }  
}  
  
  
  
  
  
  
*/\*\**  
 *\* Class <code>Button</code> implements buttons of audio player*  
 *\**   
 *\* @author Shevchouk Maxim*  
 *\* @version 1.0*  
 *\*/*  
class Button {  
 *// remaining work resource*  
private int remainingWork, count;  
  
 */\*\**  
 *\* Constructor*  
 *\*/*  
public Button() {  
 remainingWork = 99999999;  
 count = 0;  
 }  
  
 */\*\**  
 *\* Constructor*  
 *\**   
 *\* @param <code>res</code> remaining work resource*  
 *\*/*  
public Button(int res) {  
 remainingWork = res;  
 count = 0;  
 }  
  
 */\*\**  
 *\* Method simulates Button clicking*  
 *\*/*  
public void clickButton(boolean type) {  
 if (type) {  
 remainingWork--;  
 count++;  
 } else {  
 remainingWork--;  
 count--;  
 }  
 }  
  
 */\*\**  
 *\* Method returns resource of lamp*  
 *\**   
 *\*/*  
public int getResource() {  
 return remainingWork;  
 }  
  
 */\*\**  
 *\* Method returns how many times the user clicked on the button at one time*  
 *\**   
 *\* @return Button clicks*  
 *\*/*  
public int getCountClicks() {  
 return count;  
 }  
  
}

**EconomLamp.java**

package pack;  
  
import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.PrintWriter;  
  
interface Glow {  
 void LampGlowing();  
}  
  
*/\*\**  
 *\* Class <code>EconomLamp</code> implements Econom Lamp*  
 *\*/*  
public class EconomLamp extends Lamp implements Glow {  
 */\*\**  
 *\* Constructor*  
 *\**   
 *\* @throws FileNotFoundException*  
 *\*/*   
private boolean TheLampIsIntact;  
 private int PowerLight;  
   
 public EconomLamp() throws FileNotFoundException {  
 TypeOfLamp = new Type();  
 LightOfLamp = new Light();  
 PowerOn = new Button();  
 PowerOff = new Button();  
  
 TheLampIsIntact = true;  
 PowerLight = 30;  
 fout = new PrintWriter(new File("Log.txt"));  
 }  
  
 */\*\**  
 *\* Constructor*  
 *\**   
 *\* @param <code>resource</code> Button clicks resource*  
 *\* @throws FileNotFoundException*  
 *\*/*  
public EconomLamp(int resource, int uPowerLight) throws FileNotFoundException {  
 TypeOfLamp = new Type();  
 LightOfLamp = new Light();  
 PowerOn = new Button(resource);  
 PowerOff = new Button(resource);  
  
 TheLampIsIntact = true;  
 PowerLight = uPowerLight;  
 fout = new PrintWriter(new File("Log.txt"));  
 }  
  
 */\*\**  
 *\* Method provides information about the appearance of the Lamp*  
 *\*/*  
public void getInfoLamp() {  
 System.*out*.println("Light color is " + TypeOfLamp.getColor());  
 System.*out*.println("Lamp size is " + TypeOfLamp.getSize());  
 fout.println("Light color is " + TypeOfLamp.getColor());  
 fout.println("Lamp size is " + TypeOfLamp.getSize());  
 fout.flush();  
 }  
  
  
 */\*\**  
 *\* Method of modeling the lamp glowing*  
 *\*/*  
public void LampGlowing() {  
 System.*out*.println("The lamp is glow up.");  
 fout.println("The lamp is glow up.");  
 fout.flush();  
 }   
   
  
  
 */\*\**  
 *\* Method change the power light*  
 *\**   
 *\* @param <code>additionalPow</code> The value of the power light*  
 *\*/*  
public void setPower(int additionalPow) {  
 PowerLight = additionalPow;  
 }  
  
 */\*\**  
 *\* Method how the lamp works*  
 *\*/*  
public void lampWorking(int time) {  
 if (TheLampIsIntact) {  
 System.*out*.println("The lamp is working " + time +" min.");  
 fout.println("The lamp is working " + time +" min.");  
 } else {  
 System.*out*.println("The lamp is NOT working.");  
 fout.println("The lamp is NOT working.");  
 }  
 fout.flush();  
 }  
  
 */\*\**  
 *\* Method print light power*  
 *\*/*  
public void getPower() {  
 System.*out*.println("Power light: " + PowerLight);  
 fout.println("Power light: " + PowerLight);  
 fout.flush();  
 }  
  
}

**Результат виконання програми:**

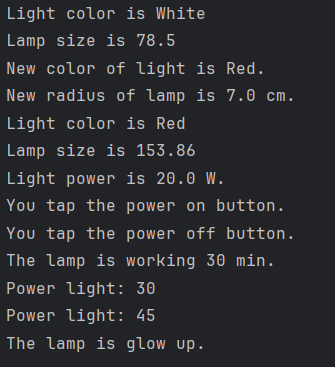
****

Рис.1.Результат виводу на консоль

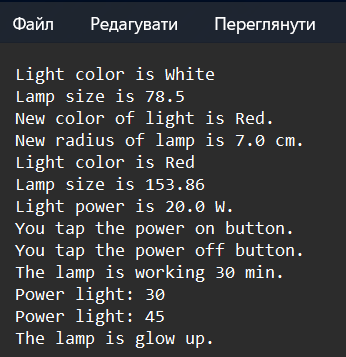


Рис.2.Результат виводу у файл

**Згенерована документація:**

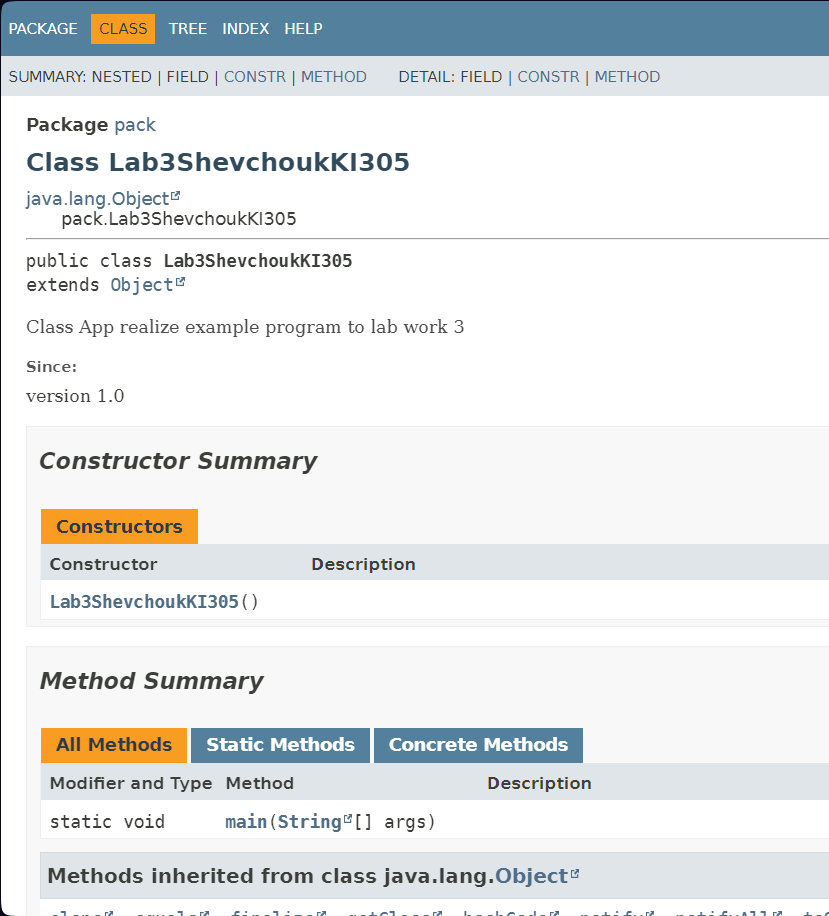
****

Рис.3. Згенерована документація

**Контрольні запитання:**

1. **Синтаксис реалізації спадкування.**

Синтаксис реалізації спадкування:

class Підклас extends Суперклас

{

Додаткові поля і методи

}

1. **Що таке суперклас та підклас??**

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

1. **Як звернутися до членів суперкласу з підкласу?**

Виклик методу суперкласу:

*super.назваМетоду([параметри]);*

Звертання до поля суперкласу:

*super.назваПоля*

1. **Коли використовується статичне зв’язування при виклику методу?**

Статичне зв’язування використовується коли метод є приватним, статичним, фінальним або конструктором.

1. **Як відбувається динамічне зв’язування при виклику методу?**

Віртуальна машина повинна викликати версію методу, що відповідає фактичному типу об’єкту на який посилається об’єктна змінна.

Оскільки на пошук необхідного методу потрібно багато часу, то віртуальна машина заздалегідь створює для кожного класу таблицю методів, в якій перелічуються сигнатури всіх методів і фактичні методи, що підлягають виклику. При виклику методу віртуальна машина просто переглядає таблицю методів.

1. **Що таке абстрактний клас та як його реалізувати?**

Абстрактний клас – це клас, для якого не можна створити об’єкти, призначений бути основою для розробки ієрархії класів.

Реалізується за допомогою ключового слова abstract.

1. **Для чого використовується ключове слово instanceof?**

Оператор instanceof дозволяє визначити, чи вказаний об’єкт належить до заданого типу.

1. **Як перевірити чи клас є підкласом іншого класу?**

Щоб перевірити чи клас є підкласом іншого класу, потрібно за допомогою instanceof порівняти, чи дійсно посилання на об’єкт супертипу посилається на об’єкт підтипу.

1. **Що таке інтерфейс?**

Це абстрактний тип, який використовується для визначення поведінки, яку класи повинні реалізовувати. Інтерфейси покликані компенсувати відсутність множинного спадкування у мові Java та гарантують визначення у класах оголошених у собі прототипів методів.

1. **Як оголосити та застосувати інтерфейс?**

Синтаксис оголошення інтерфейсів:

[public] interface НазваІнтерфейсу

{

Прототипи методів та оголошення констант інтерфейсу

}

**Висновок:**

На цій лабораторній роботі я ознайомився з спадкуванням та інтерфейсами у мові Java.