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-35

Береговський І. С.

**Прийняв:**

доцент кафедри ЕОМ

Іванов Ю. С.

Львів – 2022

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

**Завдання (варіант № 28)**

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.

• 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?

****

**Текст програми**

**LampApp.java**

package ki\_35.berehovskyi.lab3;

import java.io.\*;

/\*\*

\*

\* @author Ihor Berehovskyi

\*

\*/

public class LampApp {

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

// TODO Auto-generated method stub

Lamp lamp1 = new Lamp();

lamp1.getPower();

lamp1.turnOnLamp();

lamp1.printResource();

lamp1.changeColor("blue");

lamp1.turnOffLamp();

lamp1.printResource();

lamp1.changeColor("blue");

lamp1.displayInfo();

lamp1.dispose();

}

}

**Lamp.java**

package ki\_35.berehovskyi.lab3;

import java.io.\*;

public class Lamp {

private Filament filament;

private Flask flask;

private Base base;

private boolean isTurnedOn;

private PrintWriter fout;

/\*\*

\* Constructor

\* @throws FileNotFoundException

\*/

public Lamp() throws FileNotFoundException

{

filament = new Filament();

flask = new Flask();

base = new Base();

fout = new PrintWriter(new File("lab3\_lamp.txt"));

}

/\*\*

\* Constructor

\* @param operatingVoltage

\* @param resistance

\* @param material

\* @param color

\* @param type

\* @param size

\* @throws FileNotFoundException

\*/

public Lamp(int resistance) throws FileNotFoundException{

filament = new Filament(resistance);

flask = new Flask();

base = new Base();

fout = new PrintWriter(new File("lab3\_lamp.txt"));

}

public Lamp(int operatingVoltage, int resistance, Flask.Material material, String color, Base.Type type, double size) throws FileNotFoundException

{

filament = new Filament(operatingVoltage, resistance);

flask = new Flask(material, color);

base = new Base(type, size);

fout = new PrintWriter(new File("lab3\_lamp.txt"));

}

/\*\*

\* Method returns the OperatingVoltage

\* @return OperatingVoltage

\*/

public int getLampOperatingVoltage() {

return filament.getOperatingVoltage();

}

/\*\*

\* Method returns the Resistance

\* @return Resistance

\*/

public int getLampResistance() {

return filament.getResistance();

}

/\*\*

\* Method returns the power

\* @returnPower

\*/

public int getPower() {

return (int)(Math.pow(getLampOperatingVoltage(), 2) / getLampResistance());

}

/\*\*

\* Method returns the Color

\* @return Color

\*/

public String getLampColor() {

return flask.getColor();

}

/\*\*

\* Method returns the Type

\* @return type

\*/

public Base.Type getLampType(){

return base.getType();

}

/\*\*

\* Method simulates turning on

\*

\*/

public void turnOnLamp(){

filament.action();

isTurnedOn = true;

System.out.println("Now u can enjoy " + getLampColor() + " light");

fout.println("Now u can enjoy " + getLampColor() + " light");

}

/\*\*

\* Method simulates turning off

\*

\*/

public void turnOffLamp() {

filament.action();

isTurnedOn = false;

System.out.println("Oh dude, it's so dark now");

fout.println("Now u can enjoy " + getLampColor() + " light");

}

/\*\*

\* Method returns the Resource

\* @return Resource

\*/

public int getLampResource() {

return filament.getResource();

}

/\*\*

\* Method outputs the Resource

\*

\*/

public void printResource() {

System.out.println("Resource of lamp is " + getLampResource());

fout.println("Resource of lamp is " + getLampResource());

}

/\*\*

\* Method simulates changing color

\*

\*/

public void changeColor(String color) {

if(isTurnedOn) {

System.out.println("Firstly u must turn off your lamp");

fout.println("Firstly u must turn off your lamp");

}

else {

flask.setColor(color);

}

}

/\*\*

\* Method gets info about type

\*@return type

\*/

public String decipherType(Base.Type type) {

if(type == Base.Type.B)

return "pin socket";

else if(type == Base.Type.E)

return "Edison screw socket";

else if(type == Base.Type.G)

return "pined socket";

else if(type == Base.Type.R)

return "socket with recessed contacts";

else if(type == Base.Type.S)

return "soffit socket";

else if(type == Base.Type.P)

return " ";

else if(type == Base.Type.T)

return "telephone socket";

else

return "cable socket";

}

/\*\*

\* Method displaying info

\*

\*/

public void displayInfo() {

System.out.println("You choose the " + flask.getMaterial() + " lamp, with " + decipherType(base.getType()) + " and " + getPower()+"A");

fout.println("You choose the " + flask.getMaterial() + " lamp, with " + decipherType(base.getType()) + " and " + getPower()+"A");

}

/\*\*

\* Method releases used recourses

\*/

public void dispose(){

fout.close();

}

}

class Base{

enum Type {E, G , R, B, S, P, T, K, W}

private Type type;

private double size;

/\*\*

\* Constructor

\*/

public Base(){

type = Type.E;

size = 0.005;

}

/\*\*

\* Constructor

\* @param type

\* @param size

\*/

public Base(Type type, double size){

this.type = type;

this.size = size;

}

/\*\*

\* Method returns the Type

\* @return type

\*/

public Type getType() {

return type;

}

/\*\*

\* Method returns the Size

\* @return size

\*/

public double getSize() {

return size;

}

/\*\*

\* Method sets type

\* @param type

\*/

public void setType(Type type) {

this.type = type;

}

/\*\*

\* Method sets size

\* @param size

\*/

public void setSize(double size) {

this.size = size;

}

}

class Flask{

enum Material {GLASS, PLASTIC}

private Material material;

private String color;

/\*\*

\* Constructor

\*/

public Flask() {

material = Material.GLASS;

color = "white";

}

/\*\*

\* Constructor

\* @param material

\* @param color

\*/

public Flask(Material material, String color) {

this.material = material;

this.color = color;

}

/\*\*

\* Method returns the material

\* @return material

\*/

public Material getMaterial() {

return material;

}

/\*\*

\* Method returns the color

\* @return color

\*/

public String getColor() {

return color;

}

/\*\*

\* Method sets color

\* @param color

\*/

public void setColor(String color) {

this.color = color;

}

/\*\*

\* Method sets material

\* @param material

\*/

public void setMaterial(Material material) {

this.material = material;

}

}

class Filament {

private int operatingVoltage, resistance, Resource;

/\*\*

\* Constructor

\*/

public Filament() {

operatingVoltage = 220;

resistance = 880;

Resource = 1000000;

}

public Filament(int resistance) {

this.resistance = resistance;

}

/\*\*

\* Constructor

\* @param operatingVoltage

\* @param resistance

\*/

public Filament(int operatingVoltage, int resistance) {

this.operatingVoltage = operatingVoltage;

this.resistance = resistance;

Resource = 1000000;

}

/\*\*

\* Method returns the operatingVoltage

\* @return operatingVoltage

\*/

public int getOperatingVoltage() {

return operatingVoltage;

}

/\*\*

\* Method returns the resistance

\* @return resistance

\*/

public int getResistance() {

return resistance;

}

/\*\*

\* Method returns the Resource

\* @return Resource

\*/

public int getResource() {

return Resource;

}

/\*\*

\* Method sets operating\_voltage

\* @param operating\_voltage

\*/

public void setVoltage(int operating\_voltage) {

this.operatingVoltage = operating\_voltage;

}

/\*\*

\* Method sets resistance

\* @param resistance

\*/

public void setResistance(int resistance) {

this.resistance = resistance;

}

/\*\*

\* Method sets Resource

\* @param Resource

\*/

public void setResource(int resource) {

this.Resource = resource;

}

/\*\*

\* Method decrements resource

\*/

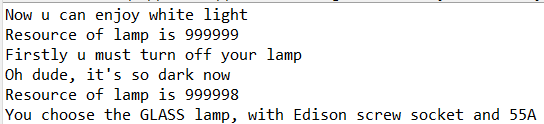
public void action() {

Resource--;

}

}

**Результат виконання програми**



**Фрагмент згенерованої документації**



**Відповіді на контрольні запитання**

• Синтаксис оголошення пакету.

package НазваПакету{.НазваПідпакету};

• В чому суть статичного імпорту пакетів?

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

**Висновок**

На даній лабораторній роботі я написав реалізацію класу, який містить 3 поля, що є об’єктами класів, які описують складові частини предметної області. В кожному класі створив кілька конструкторів та методів, а також, для тестування і демонстрації роботи розробленого класу розробив клас-драйвер. Це дало змогу ознайомитися з процесом розробки класів та пакетів мовою Java.