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: «Спадкування та інтерфейси»

Виконав:
студент групи КІ-306
Щирба Д.В.
Перевірив:
доцент кафедри ЕОМ
Іванов Ю. С.

Львів – 2023

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

Завдання (варіант № 28) – похідний клас Енергозберігаюча лампочка

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 (багатофункціональний пристрій) 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.
5. Дати відповідь на контрольні запитання.

Вихідний код програми:

Файл Lightbulb.java

```
package KI306.Shchyrba.Lab3;

import java.io.FileNotFoundException;
import java.io.*;

/**
 * This class represents a Lightbulb with various attributes and functionality.
 */

public abstract class Lightbulb {

        private String model;
   private int energy_consumption;
   protected boolean isOn;
   private String color;
   protected PrintWriter logFile;

   /**
    * Default constructor initializes an object with default values.
    */

   public Lightbulb() throws FileNotFoundException
   {
        model = "Voltic";
        energy_consumption = 10;
      isOn = false;
      color = "white";
      logFile = new PrintWriter(new File("Lightbulb_Log.txt"));
   }

   /**
    * Constructor with parameters initializes an object with specified values.
    *
    * @param model                    The model of the Lightbulb.
    * @param energy_consumption       The energy  consumption of the Lighbulb (in watts).
    * @param isOn                     The state of Lightbulb.
    * @param color                    The color which Lightbulb emits.
    * @throws FileNotFoundException   Occurs when the log file cannot be created.
    */

   public Lightbulb(String model, int energy_consumption, String color) throws
FileNotFoundException
   {

        this.model = model;
           this.energy_consumption = energy_consumption;
           this.isOn = false; // Лампочка за замовчуванням вимкнена
           this.color = color;
           logFile = new PrintWriter(new File("Lightbulb_Log.txt"));
   }
```

```
/**
 * Turns on lightbulb and logs the change.
 *
 */

public void TurnOn ()
{
	this.isOn = true;
	logFile.println("Лампочку ввімкнено.");
	logFile.flush();
}

/**
 * Turns off lightbulb and logs the change.
 *
 */

public void TurnOff ()
{
	this.isOn = false;
	logFile.println("Лампочку ввимкненно.");
	logFile.flush();
}

/**
 * Changes the color of lightbulb and logs the change.
 *
 ** @param newColor New color of the lightbulb .
 */

public void changeColor(String newColor) {
   color = newColor;
   logFile.println("Колір світла змінено на " + newColor);
   logFile.flush();
}

/**
 * Changes the model of the lightbulb and logs the change.
 *
 ** @param newModel New model of the lightbulb .
 */

public void ChangeModel(String newModel) {
   model = newModel;
   logFile.println("Модель змінено на " + newModel);
   logFile.flush();
}

/**
 *Returns model of the lightbulb
 *
 *@return model
 */
```

```
public String GetModel() {
   return model;
}

/**
 *Returns energy consumption of the lightbulb
 *
 *@return energy consumption value
 */

public int GetEnergyConsumption() {
   return energy_consumption;
}

/**
 * Changes energy consumption of the lightbulb and logs the change.
 *
 ** @param newEnergyConsumption New energy consumption of the lightbulb .
 */

public void ChangeEnergyConsumption(int newEnergyConsumption) {
   energy_consumption = newEnergyConsumption;
   logFile.println("Потужність змінено на " + newEnergyConsumption + " ватт");
   logFile.flush();
}

/**
 * Defines luminous efficiency of the lightbulb and logs the stats.
 *
 * @return Luminous efficiency value
 */

public int LuminousEfficiency ()
{
      int efficiency = 10 * energy_consumption;
      logFile.println("Енергоефективність лампочки " + efficiency + "люмен/ватт");
            logFile.flush();
            return efficiency;
}

/**
 * Displays info about the current lightbulb
 */

public void GetInfo() {
   System.out.println("Модель: " + model);
   System.out.println("Потужність (ватт): " + energy_consumption);
   System.out.println("Стан: " + (isOn ? "увімкнена" : "вимкнена"));
   System.out.println("Колір світла: " + color);
}

/**
 * Closes log file.
 */
```

```
    public void CloseLogFile() {
        logFile.close();
    }



}
```

Файл ES_Lightbulb_Interface.java

```
package KI306.Shchyrba.Lab3;

public interface ES_Lightbulb_Interface {

	public int getBrightnessLevel();
	public void setBrightnessLevel(int brightnessLevel);
	public void increaseBrightness(int amount);
	public void decreaseBrightness(int amount);

	public boolean hasDaylightSensor();
	public void toggleDaylightSensor();

	public void autoTurnOnAtBrightnessThreshold(int threshold);
	public void GetInfo();


}
```

Файл ES_Lightbulb.java

```
package KI306.Shchyrba.Lab3;

import java.io.FileNotFoundException;

/**
 * The {@code ES_Lightbulb} class represents an energysaving lightbulb, which extends the
{@code Lightbulb}
 * abstract class and implements the {@code ES_Lightbulb_Interface} interface. It adds specific
 * functionality for an office center, including office space allocation and office equipment
management.
 *
 * @author Danylo Shchyrba
 * @version 1.0
 */

public class ES_Lightbulb extends Lightbulb implements ES_Lightbulb_Interface {

	private int brightnessLevel; // Рівень яскравості (від 1 до 10)
	private boolean daylightSensor; // Наявність датчика світла


		/**
```

```
     * Default constructor initializes an object with default values.
     */
    public ES_Lightbulb() throws FileNotFoundException {
        super(); // Викликаємо конструктор за замовчуванням з базового класу
        brightnessLevel = 5; // За замовчуванням рівень яскравості - 5
        daylightSensor = false; // За замовчуванням датчик світла вимкнений
    }

    /**
     * Constructor with parameters initializes an object with specified values
     * @param brightnessLevel       The brigntess level of the ES_Lightbulb.
     * @param daylightSensor        The state of the daylight sensor of the ES_Lighbulb.
     **/
    public ES_Lightbulb(String model, int energy_consumption, String color, int
brightnessLevel, boolean daylightSensor) throws FileNotFoundException {
        super(model, energy_consumption, color);
        this.brightnessLevel = brightnessLevel;
        this.daylightSensor = daylightSensor;
    }

    /* Getter for brightnessLevel
     * @returns brightnessLevel Returns current brightness level of the ES_lightbulb
    **/

    public int getBrightnessLevel() {
        return brightnessLevel;
    }

    /* Setter for brightnessLevel
     * @param brightnessLevel Sets the brightness level of the ES_lightbulb
    **/

    public void setBrightnessLevel(int brightnessLevel) {
        this.brightnessLevel = brightnessLevel;
        // Логуємо зміну рівня яскравості
        logFile.println("Рівень яскравості змінено на " + brightnessLevel);
        logFile.flush();
    }

    /* Increases brightnessLevel of the ES_Lightbulb
     * @param amount Sets amount on which will be the brightness level of the
ES_lightbulb increased
    **/

    public void increaseBrightness(int amount) {
            if (brightnessLevel + amount <= 10)
        brightnessLevel += amount;
        // Логуємо зміну рівня яскравості
        logFile.println("Рівень яскравості збільшено на " + amount);
        logFile.flush();
    }

    /* Decreases brightnessLevel of the ES_Lightbulb
```

```
     * @param amount Sets amount on which will be the brightness level of the
ES_lightbulb decreased
     **/

    public void decreaseBrightness(int amount) {
        if (brightnessLevel - amount >= 1) {
            brightnessLevel -= amount;
            // Логуємо зміну рівня яскравості
            logFile.println("Рівень яскравості зменшено на " + amount);
            logFile.flush();
        }
    }

    // Метод для перевірки наявності датчика світла
    public boolean hasDaylightSensor() {
        return daylightSensor;
    }

    /*
     * Toggles daylightSensor state an logs the change
    **/

    public void toggleDaylightSensor() {
        daylightSensor = !daylightSensor;
        // Логуємо зміну стану датчика світла
        logFile.println("Датчик світла " + (daylightSensor ? "увімкнено" : "вимкнено"));
        logFile.flush();
    }

    /*
     * Turns on Lightbulb depending on environment brightness and Lightbulb brightness
an logs the change
     * @param threshold Environment brightness
    **/

    public void autoTurnOnAtBrightnessThreshold(int threshold) {
        if (brightnessLevel >= threshold && !isOn) {
            TurnOn();
        }
    }

    @Override
    public void GetInfo() {
        super.GetInfo(); // Викликаємо метод GetInfo() з базового класу
        System.out.println("Рівень яскравості: " + brightnessLevel);
        System.out.println("Датчик світла: " + (daylightSensor ? "присутній" :
"відсутній"));
    }
}
```

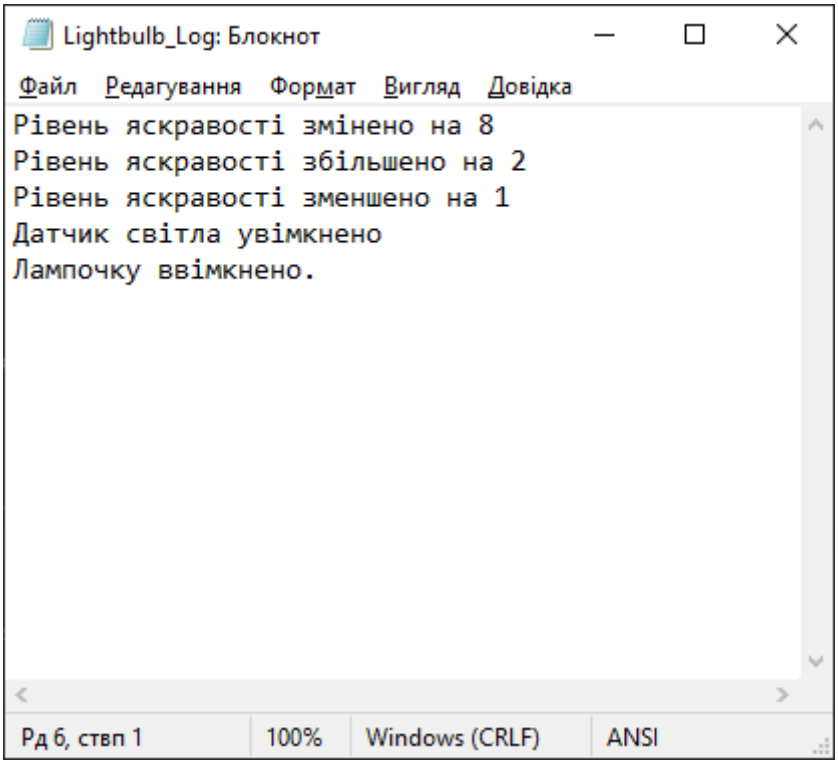
Файл LightbulbApp.java

```
package KI306.Shchyrba.Lab3;
import java.io.*;

/**
 * The {@code LightbulbApp} class provides a simple driver program to test the Lightbulb and ES_Lightbulb classes.
 *
 * @author Danylo Shchyrba
 * @version 1.0
 */

public class LightbulbApp {

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

		ES_Lightbulb E77 = new ES_Lightbulb();
		E77.getBrightnessLevel();
		E77.setBrightnessLevel(8);
		E77.increaseBrightness(2);
		E77.decreaseBrightness(1);

		E77.hasDaylightSensor();
		E77.toggleDaylightSensor();

		E77.autoTurnOnAtBrightnessThreshold(7);

		E77.GetInfo();
	}
}
```

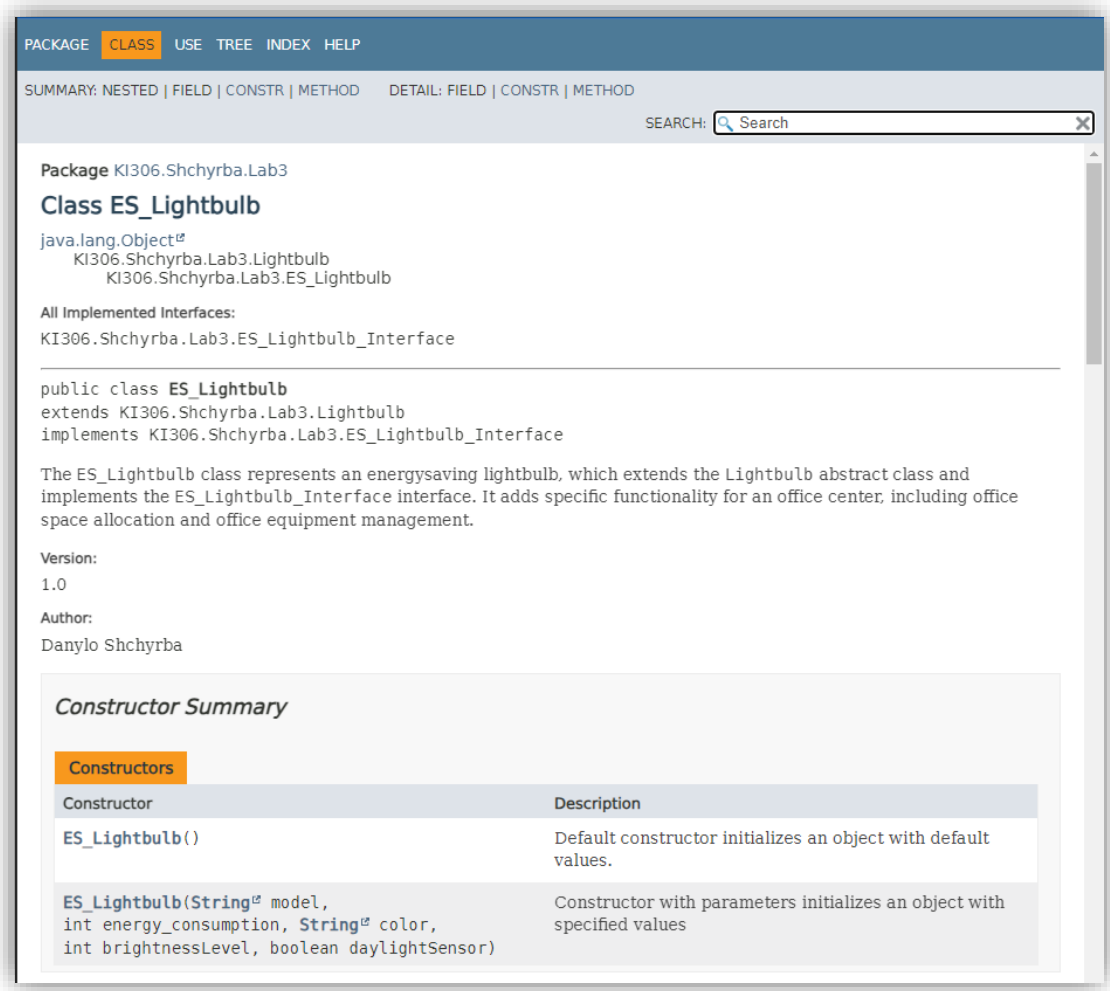
Результат виконання програми:

```
Модель: Voltic
Потужність (ватт): 10
Стан: увімкнена
Колір світла: white
Рівень яскравості: 9
Датчик світла: присутній
```

Текстовий файл з результатом виконання програми:



Фрагмент згенерованої документації:



PACKAGE CLASS USE TREE INDEX HELP

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

SEARCH: Search

Package KI306.Shchyrba.Lab3

Class ES_Lightbulb

java.lang.Object
KI306.Shchyrba.Lab3.Lightbulb
KI306.Shchyrba.Lab3.ES_Lightbulb

All Implemented Interfaces:

`KI306.Shchyrba.Lab3.ES_Lightbulb_Interface`

```
public class ES_Lightbulb
extends KI306.Shchyrba.Lab3.Lightbulb
implements KI306.Shchyrba.Lab3.ES_Lightbulb_Interface
```

The `ES_Lightbulb` class represents an energysaving lightbulb, which extends the `Lightbulb` abstract class and implements the `ES_Lightbulb_Interface` interface. It adds specific functionality for an office center, including office space allocation and office equipment management.

Version:

1.0

Author:

Danylo Shchyrba

Constructor Summary

Constructors

Constructor	Description
`ES_Lightbulb()`	Default constructor initializes an object with default values.
`ES_Lightbulb(String model, int energy_consumption, String color, int brightnessLevel, boolean daylightSensor)`	Constructor with parameters initializes an object with specified values

Відповіді на контрольні запитання:

1. Синтаксис реалізації спадкування.
   - class МійКлас implements Інтерфейс {
     // тіло класу
     }

2. Що таке суперклас та підклас?
   - суперклас - це клас, від якого інший клас успадковує властивості та методи. Підклас - це клас, який успадковує властивості та методи від суперкласу.

3. Як звернутися до членів суперкласу з підкласу?
   - super.назваМетоду([параметри]); // виклик методу суперкласу
     super.назваПоля; // звернення до поля суперкласу

4. Коли використовується статичне зв'язування при виклику методу?
   - Статичне зв'язування використовується, коли метод є приватним, статичним, фінальним або конструктором. В таких випадках вибір методу відбувається на етапі компіляції.

5. Як відбувається динамічне зв'язування при виклику методу?
   - вибір методу для виклику відбувається під час виконання програми на основі фактичного типу об'єкта.

6. Що таке абстрактний клас та як його реалізувати?
   - це клас, який має один або більше абстрактних методів (методів без реалізації). Щоб створити абстрактний клас, використовується ключове слово abstract.
     Приклад:
     abstract class АбстрактнийКлас {
     abstract void абстрактнийМетод();
     }

7. Для чого використовується ключове слово instanceof?
   - для перевірки, чи об'єкт належить до певного класу або інтерфейсу.
     Синтаксис:
     if (об'єкт instanceof Клас) {
     // код, який виконується, якщо об'єкт належить до класу
     }

8. Як перевірити чи клас є підкласом іншого класу?
   - В Java використовується ключове слово extends, щоб вказати, що клас є підкласом іншого класу. Перевірити, чи один клас є підкласом іншого класу можна шляхом аналізу ієрархії успадкування.

9. Що таке інтерфейс?
   - це абстрактний тип даних, який визначає набір методів, але не надає їх реалізацію. Всі методи інтерфейсу є загальнодоступними та автоматично є public. Інтерфейси використовуються для створення контрактів, які класи повинні реалізувати.

10. Як оголосити та застосувати інтерфейс?
    - Для оголошення інтерфейсу використовується ключове слово interface.
      Синтаксис:
      ```
      interface Інтерфейс {
          // оголошення методів та констант
      }
      ```

    - Для застосування інтерфейсу в класі використовується ключове слово implements.
      Синтаксис:
      ```
      class МійКлас implements Інтерфейс {
          // реалізація методів інтерфейсу
      }
      ```

Висновок:

У ході виконання даної лабораторної роботи, я отримав навички роботи з концепціями спадкування та інтерфейсами в мові програмування Java. Ознайомившись з цими важливими аспектами об'єктно-орієнтованого програмування, я зрозумів їх роль у створенні більш структурованих і гнучких програм.