Міністерство освіти і науки України

Національний університет "Львівська політехніка"

Звіт з лабораторної роботи №3

з курсу “Кросплатформні засоби програмування”

Спадкування та інтерфейси

Виконав: студент гр. KI-306

Шаповал Віталій

Прийняв: к.т.н. Олексів М.В.

Львів 2024 р.

ТЕОРЕТИЧНІ ВІДОМОСТІ

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 C++ не існує. На відміну від C++ у Java можливе спадкування лише одного базового класу (множинне спадкування відсутнє). Спадкування реалізується шляхом вказування ключового слова class після якого вказується назва підкласу, ключове слово extends та назва суперкласу, що розширюється у новому підкласі.

Варіант № 28 Смартлампа

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.

Файл ./ce306/shapoval/lab3/Switchable.java:

```
package ce306.shapoval.lab3;

/**
 * Interface <code>Switchable</code> defines methods to switch the lamp on and
off.
 */
public interface Switchable {

    /**
     * Method to switch the lamp on.
     */
    void turnOn();
```

```
        /**
         * Method to switch the lamp off.
         */
        void turnOff();

        /**
         * Method to check if the lamp is currently on.
           * @return <code>true</code> if the lamp is on, <code>false</code>
otherwise.
         */
        boolean isOn();
    }
```

Файл ./ce306/shapoval/lab3/Lamp.java:

```
    /**
    * lab 3 package
    */
    package ce306.shapoval.lab3;

    import java.io.*;

    /**
    * Class <code>Lamp</code> implements lamp details
    */
    public abstract class Lamp {
        //private Object battery_;
        protected LampBattery battery;
        protected LampSocket socket;
        protected PrintWriter fout;

        /**
        * Default constructor
        * @throws FileNotFoundException
        */
        public Lamp() throws FileNotFoundException {
            battery = null;
            socket = new LampSocket();
            fout = new PrintWriter(new File("lamp_log.txt"));
        }

        /**
        * Parameterized constructor
        * @param battery <code>LampBattery</code> object
        * @param type <code>LampType</code> object
        * @param socket <code>LampSocket</code> object
        * @throws FileNotFoundException
        */
              public Lamp(LampBattery battery, LampSocket socket) throws
FileNotFoundException {
            this.battery = battery;
            this.socket = socket;
```

```
            fout = new PrintWriter(new File("lamp_log.txt"));
        }

        /**
        * Method returns lamp's information
        * @return Lamp's information
        */
        public String getLampInfo() {
            String info = "Лампа:\n";
            if (battery != null) {
                info += battery.checkStatus() + "\n";
            }
            info += socket.getSocketInfo();

            fout.println(info);
            fout.flush();
            return info;
        }

        /**
        * Method charges the battery
        * @param amount The amount to charge
             *  @return  <code>true</code>  if  the  charge  was  successful,
<code>false</code> otherwise
        */
        public boolean chargeBattery(double amount) {
            if(battery != null) {
                battery.charge(amount);
                fout.printf("Батарея була заряджена на %.2f мАг.\n", amount);
                fout.flush();
                return true;
            }
            fout.println("Батарея не існує");
            fout.flush();
            return false;
        }

        /**
        * Method discharges the battery
        * @param amount The amount to discharge
            *  @return  <code>true</code>  if  the  discharge  was  successful,
<code>false</code> otherwise
        */
        public boolean dischargeBattery(double amount) {
            if (battery != null) {
                battery.discharge(amount);
                fout.printf("Батарея була розряджена на %.2f мАг.\n", amount);
                fout.flush();
                return true;
            }
            fout.println("Батарея не існує");
            fout.flush();
            return false;
```

```
        }

        /**
        * Method returns the current battery capacity
        * @return Current battery capacity
        */
        public double getCurrentBatteryCapacity() {
            if (battery == null) {
                fout.println("Батарея не існує");
                fout.flush();
                return 0;
            }

                            fout.println("Значення потужності батареєю: " +
battery.getCurrentCapacity());
            fout.flush();
            return battery.getCurrentCapacity();
        }

        /**
        * Method checks the status of the lamp
        * @return Current status of the lamp
        */
        public String checkLampStatus() {
            if (battery == null) {
                fout.println("Батарея не існує");
                fout.flush();
                return "Батарея не існує";
            }

            String status = battery.checkStatus();
            fout.println("Стан лампи перевірено: " + status);
            fout.flush();
            return status;
        }

        /**
        * Method releases used resources
        */
        public void dispose() {
            fout.close();
        }

    }
```

Файл ./ce306/shapoval/lab3/LampBattery.java:

```
    package ce306.shapoval.lab3;

    /**
     * The <code>LampBattery</code> class represents the battery used in a lamp.
     * It includes fields for nominal voltage, nominal capacity, and current
capacity,
```

```
     * and provides methods to charge and discharge the battery, as well as to
check its status.
     */
    public class LampBattery {
       private double nominalVoltage;   // The nominal voltage of the battery (in
volts)
        private double nominalCapacity;   // The nominal capacity of the battery
(in mAh)
        private double currentCapacity;   // The current capacity of the battery
(in mAh)

        /**
         * Default constructor that initializes the battery with default values.
         * Nominal voltage is set to 3.7V and capacity to 1500mAh.
         * The current capacity is set to be fully charged.
         */
        public LampBattery() {
            nominalVoltage = 3.7;
            nominalCapacity = 1500;
            currentCapacity = nominalCapacity;
        }

        /**
         * Constructor that initializes the battery with specified nominal voltage
and capacity.
           * The current capacity is set to match the nominal capacity (fully
charged).
         *
         * @param nominalVoltage The nominal voltage of the battery (in volts)
         * @param nominalCapacity The nominal capacity of the battery (in mAh)
         */
        public LampBattery(double nominalVoltage, double nominalCapacity) {
            this.nominalVoltage = nominalVoltage;
            this.nominalCapacity = nominalCapacity;
            this.currentCapacity = nominalCapacity;
        }

        /**
         * Constructor that initializes the battery with specified nominal voltage,
         * nominal capacity, and current capacity.
         *
         * @param nominalVoltage The nominal voltage of the battery (in volts)
         * @param nominalCapacity The nominal capacity of the battery (in mAh)
         * @param currentCapacity The current capacity of the battery (in mAh)
         */
        public LampBattery(double nominalVoltage, double nominalCapacity, double
currentCapacity) {
            this.nominalVoltage = nominalVoltage;
            this.nominalCapacity = nominalCapacity;
            this.currentCapacity = currentCapacity;
        }

        /**
```

```
         * Charges the battery by the specified amount.
         * The battery cannot be overcharged beyond its nominal capacity.
         *
         * @param amount The amount to charge the battery by (in mAh)
         */
        public void charge(double amount) {
            if (amount < 0) {
                  System.out.println("Cannot charge the battery with a negative
value.");
                return;
            }
            currentCapacity += amount;
            if (currentCapacity > nominalCapacity) {
              currentCapacity = nominalCapacity; // Limit the battery capacity to
the nominal value
            }
        }

        /**
         * Discharges the battery by the specified amount.
         * The battery cannot have a negative capacity.
         *
         * @param amount The amount to discharge the battery by (in mAh)
         */
        public void discharge(double amount) {
            if (amount < 0) {
                System.out.println("Cannot discharge the battery with a negative
value.");
                return;
            }
            currentCapacity -= amount;
            if (currentCapacity < 0) {
                currentCapacity = 0; // Prevent the capacity from becoming negative
            }
        }

        /**
         * Returns the nominal voltage of the battery.
         *
         * @return The nominal voltage (in volts)
         */
        public double getNominalVoltage() {
            return nominalVoltage;
        }

        /**
         * Returns the nominal capacity of the battery.
         *
         * @return The nominal capacity (in mAh)
         */
        public double getNominalCapacity() {
            return nominalCapacity;
        }
```

```
        /**
         * Returns the current capacity of the battery.
         *
         * @return The current capacity (in mAh)
         */
        public double getCurrentCapacity() {
            return currentCapacity;
        }

        /**
         * Checks the status of the battery based on its current capacity.
         *
         * @return A string indicating the current status of the battery:
         *         - "Battery is discharged" if the capacity is 0
        *        - "Battery is almost discharged" if the capacity is less than 20%
of nominal
         *         - "Battery is in normal condition" otherwise
         */
        public String checkStatus() {
            if (currentCapacity == 0) {
                return "Battery is discharged.";
            } else if (currentCapacity < nominalCapacity * 0.2) {
                return "Battery is almost discharged.";
            } else {
                return "Battery is in normal condition.";
            }
        }
    }
```

Файл ./ce306/shapoval/lab3/LampSocket.java:

```
    package ce306.shapoval.lab3;

    /**
     * The <code>LampSocket</code> class represents the socket (or base) of a
lamp,
     * which connects the lamp to the electrical supply.
     * It includes attributes such as the type, shape, and diameter of the socket.
     */
    public class LampSocket {

        // Fields to store the properties of the socket
        private String type;      // The type of the socket (e.g., E5, G24, B22d)
       private String shape;    // The shape or connection type (e.g., threaded,
bi-pin)
        private double diameter;  // The diameter of the socket in millimeters

        /**
        * Default constructor initializing with default values (type E5, threaded
connection).
         * The diameter is set to 5mm.
         */
        public LampSocket() {
```

```
            this.type = "E5";
            this.diameter = 5;
            this.shape = "threaded connection";
        }

        /**
         * Constructor to initialize the socket with specific type, shape, and
diameter.
         *
         * @param type The type of the socket (e.g., E27, G24)
        * @param shape The connection type or shape of the socket (e.g., threaded,
bi-pin)
         * @param diameter The diameter of the socket in millimeters
         */
        public LampSocket(String type, String shape, double diameter) {
            this.type = type;
            this.diameter = diameter;
            this.shape = shape;
        }

        /**
         * Returns the type of the socket.
         *
         * @return A string representing the socket type
         */
        public String getType() {
            return type;
        }

        /**
         * Returns the diameter of the socket in millimeters.
         *
         * @return The diameter of the socket
         */
        public double getDiameter() {
            return diameter;
        }

        /**
         * Returns the shape or connection type of the socket.
         *
         * @return A string representing the socket's shape
         */
        public String getShape() {
            return shape;
        }

        /**
         * Sets the type of the socket.
         *
         * @param type The new type of the socket
         */
        public void setType(String type) {
```

```
            this.type = type;
        }

        /**
         * Sets the diameter of the socket in millimeters.
         *
         * @param diameter The new diameter of the socket
         */
        public void setDiameter(double diameter) {
            this.diameter = diameter;
        }

        /**
         * Sets the shape or connection type of the socket.
         *
         * @param shape The new shape of the socket
         */
        public void setShape(String shape) {
            this.shape = shape;
        }

        /**
          * Displays detailed information about the socket, including its type,
shape, and diameter.
         *
         * @return A formatted string containing socket details
         */
        public String getSocketInfo() {
            return "Socket Type: " + type + "\n" +
                    "Socket Shape: " + shape + "\n" +
                    "Socket Diameter: " + diameter + " mm\n";
        }
    }
```

Файл ./ce306/shapoval/lab3/EnergyLamp.java:

```
    package ce306.shapoval.lab3;

    import java.io.FileNotFoundException;

    /**
      *  The  <code>EnergySavingLamp</code>  class  extends  the  abstract  class
<code>Lamp</code>
     * and implements the <code>Switchable</code> interface.
     */
    public final class EnergySavingLamp extends Lamp implements Switchable {
        // Energy efficiency factor (e.g., 0.8 means 80% efficiency)
        private final double energyEfficiency;
        private boolean isOn;

        /**
           * Constructor that initializes the energy-saving lamp with a given
battery, socket, and energy efficiency.
         * @param battery Lamp battery
```

```
     * @param socket Lamp socket
     * @param energyEfficiency Efficiency factor (1.0 means 100% efficiency)
     * @throws FileNotFoundException
     */
     public EnergySavingLamp(LampBattery battery, LampSocket socket, double
energyEfficiency) throws FileNotFoundException {
        super(battery, socket);
        this.energyEfficiency = energyEfficiency;
        this.isOn = false; // Lamp is initially off
    }

    /**
         * Implementation of the <code>turnOn</code> method from the
<code>Switchable</code> interface.
     * Turns the lamp on.
     */
    @Override
    public void turnOn() {
        isOn = true;
        fout.println("Лампу увімкнено.");
        fout.flush();
    }

    /**
         * Implementation of the <code>turnOff</code> method from the
<code>Switchable</code> interface.
     * Turns the lamp off.
     */
    @Override
    public void turnOff() {
        isOn = false;
        fout.println("Лампу вимкнено.");
        fout.flush();
    }

    /**
         * Implementation of the <code>isOn</code> method from the
<code>Switchable</code> interface.
        * @return <code>true</code> if the lamp is on, <code>false</code>
otherwise.
     */
    @Override
    public boolean isOn() {
        return isOn;
    }

    /**
     * Method that overrides <code>getLampInfo</code> to provide information
about the energy-saving lamp.
     * @return Information about the lamp, including its energy efficiency and
status.
     */
    @Override
```

```
        public String getLampInfo() {
            String info = super.getLampInfo();
                  info += String.format("Енергоефективність лампи: %.2f%%\n",
energyEfficiency * 100);
            info += isOn ? "Лампа увімкнена\n" : "Лампа вимкнена\n";

            fout.println(info);
            fout.flush();
            return info;
        }

        /**
         * Method to get the energy efficiency of the lamp.
         * @return The energy efficiency factor.
         */
        public double getEnergyEfficiency() {
            return energyEfficiency;
        }
    }
```

Файл ./EnergySavingLampApp.java:

```
    import ce306.shapoval.lab3.*;
    import java.io.FileNotFoundException;
    import java.util.Scanner;

    /**
     * The <code>EnergySavingLampApp</code> class is a driver class that tests the
functionality
       * of the <code>EnergySavingLamp</code> class, which extends the
<code>Lamp</code> class.
     * It provides a console menu for users to interact with both the base class
functionalities
       * (like charging and discharging the battery) and the extended
functionalities (turning the
     * lamp on and off, checking energy efficiency).
     */
    public class EnergySavingLampApp {
        /**
              * The entry point of the application. It initializes the
<code>EnergySavingLamp</code> object
          * and displays a menu for user interaction. The user can choose from
various options such as
         * turning the lamp on/off, charging/discharging the battery, checking the
lamp status,
         * and getting lamp information, including its energy efficiency.
         *
         * @param args Command line arguments (not used in this application)
         */
        public static void main(String[] args) {
            // Scanner to read user input
            Scanner scanner = new Scanner(System.in);

            // Creating EnergySavingLamp object
```

```
            EnergySavingLamp lamp = null;
            try {
                lamp = new EnergySavingLamp(new LampBattery(), new LampSocket(),
0.8); // 80% efficiency
            } catch (FileNotFoundException e) {
                   System.out.println("Error: Unable to create file for logging
data.");
                return;
            }

            boolean exit = false;

            while (!exit) {
                // Displaying menu options
                System.out.println("===== Energy-Saving Lamp Menu =====");
                System.out.println("1. Turn lamp on");
                System.out.println("2. Turn lamp off");
                System.out.println("3. Check if the lamp is on");
                System.out.println("4. Get lamp information");
                System.out.println("5. Charge battery");
                System.out.println("6. Discharge battery");
                System.out.println("7. Check battery status");
                System.out.println("8. Check current battery capacity");
                System.out.println("9. Exit");
                System.out.print("Enter your choice: ");

                int choice = scanner.nextInt();
                switch (choice) {
                    case 1:
                        lamp.turnOn();
                        System.out.println("The lamp is now on.");
                        break;
                    case 2:
                        lamp.turnOff();
                        System.out.println("The lamp is now off.");
                        break;
                    case 3:
                       System.out.println(lamp.isOn() ? "The lamp is on." : "The lamp
is off.");
                        break;
                    case 4:
                        System.out.println(lamp.getLampInfo());
                        break;
                    case 5:
                        System.out.print("Enter charge amount: ");
                        double chargeAmount = scanner.nextDouble();
                        if (lamp.chargeBattery(chargeAmount)) {
                            System.out.println("Battery charged.");
                        } else {
                            System.out.println("Failed to charge battery.");
                        }
                        break;
                    case 6:
```

```
                        System.out.print("Enter discharge amount: ");
                        double dischargeAmount = scanner.nextDouble();
                        if (lamp.dischargeBattery(dischargeAmount)) {
                            System.out.println("Battery discharged.");
                        } else {
                            System.out.println("Failed to discharge battery.");
                        }
                        break;
                    case 7:
                                        System.out.println("Battery  status:  "  +
lamp.checkLampStatus());
                        break;
                    case 8:
                             System.out.println("Current battery capacity: " +
lamp.getCurrentBatteryCapacity());
                        break;
                    case 9:
                        System.out.println("Exiting.");
                        exit = true;
                        break;
                    default:
                        System.out.println("Invalid choice.");
                }
            }

            // Releasing resources
            lamp.dispose();
            scanner.close();
        }
    }
```

Виконання програми:

Термінал:

===== Energy-Saving Lamp Menu =====

1. Turn lamp on

2. Turn lamp off

3. Check if the lamp is on

4. Get lamp information

5. Charge battery

6. Discharge battery

7. Check battery status

8. Check current battery capacity

9. Exit

Enter your choice: 1

The lamp is now on.

===== Energy-Saving Lamp Menu =====

1. Turn lamp on

2. Turn lamp off

3. Check if the lamp is on

4. Get lamp information

5. Charge battery

6. Discharge battery

7. Check battery status

8. Check current battery capacity

9. Exit

Enter your choice: 3

The lamp is on.

===== Energy-Saving Lamp Menu =====

1. Turn lamp on

2. Turn lamp off

3. Check if the lamp is on

4. Get lamp information

5. Charge battery

6. Discharge battery

7. Check battery status

8. Check current battery capacity

9. Exit

Enter your choice: 2

The lamp is now off.

===== Energy-Saving Lamp Menu =====

1. Turn lamp on

2. Turn lamp off

3. Check if the lamp is on

4. Get lamp information

5. Charge battery

6. Discharge battery

7. Check battery status

8. Check current battery capacity

9. Exit

Enter your choice: 3

The lamp is off.

===== Energy-Saving Lamp Menu =====

1. Turn lamp on

2. Turn lamp off

3. Check if the lamp is on

4. Get lamp information

5. Charge battery

6. Discharge battery

7. Check battery status

8. Check current battery capacity

9. Exit

Enter your choice: 4

Лампа:

Battery is in normal condition.

Socket Type: E5

Socket Shape: threaded connection

Socket Diameter: 5.0 mm

Енергоефективність лампи: 80.00%

Лампа вимкнена

===== Energy-Saving Lamp Menu =====

1. Turn lamp on

2. Turn lamp off

3. Check if the lamp is on

4. Get lamp information

5. Charge battery

6. Discharge battery

7. Check battery status

8. Check current battery capacity

9. Exit

Enter your choice: 9

Exiting.

Вміст Лог файлу:

Лампу увімкнено.

Лампу вимкнено.

Лампа:

Battery is in normal condition.

Socket Type: E5

Socket Shape: threaded connection

Socket Diameter: 5.0 mm

Лампа:

Battery is in normal condition.

Socket Type: E5

Socket Shape: threaded connection

Socket Diameter: 5.0 mm

Енергоефективність лампи: 80.00%

Лампа вимкнена

Висновок

На цій лабораторній роботі я поглибив свої знання з об'єктно-орієнтованого програмування на мові Java, зокрема навчився розширювати

класи, що вже були реалізовані у попередніх лабораторних роботах, шляхом додавання нових функцій та реалізації інтерфейсів. Я зробив суперклас з лабораторної роботи №2 абстрактним, що дозволило створити більш гнучку та розширювану архітектуру. Розроблений підклас забезпечив коректне функціонування лампи з можливістю зміни кольору та роботи з батареєю, а також реалізував інтерфейс для зміни кольору лампи.

Особливу увагу було приділено організації коду в окремому пакеті, відповідно до структури пакунків Група.Прізвище.Lab3, а також доданню коментарів, які дали змогу автоматично згенерувати документацію до пакету. Це дало змогу практично закріпити навички написання самодокументованого коду та роботи з інструментами генерації документації в Java.

Окрім того, я вдосконалив свої навички роботи з системами контролю версій Git, завантаживши проект на платформу GitHub згідно методичних вказівок. Це допомогло покращити вміння співпраці над проектами та ведення спільної розробки з використанням сучасних інструментів для контролю версій.