Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра ЕОМ

Звіт

до лабораторної роботи № 3

з дисципліни: «Кросплатформні засоби програмування»
на тему: «Класи та пакети»

Варіант 11

Виконав:

ст.гр. КІ-36

Басько С.І.

Прийняв:

Іванов Ю.С.

Львів 2022

ТЕОРЕТИЧНІ ВІДОМОСТІ

Класи

Мова Java є повністю об’єктно-орієнтованою мовою програмування, тому вона дозволяє писати програми лише з використанням об’єктно-орієнтованих парадигм програмування, що базуються на понятті класів.

Синтаксис оголошення простого класу в мові Java має наступний вигляд:

```
[public] class НазваКласу
{
	[конструктори]
	[методи]
	[поля]
}
```

Необов’язковий специфікатор доступу `public` робить клас загальнодоступним. У кожному файлі з кодом програми може бути лише один загальнодоступний клас, ім’я якого співпадає з назвою файлу, та безліч класів без специфікатора `public`. Створення об’єкту класу складається з двох етапів: оголошення та ініціалізації посилання на об’єкт. Оголошення посилання на об’єкт класу має синтаксис:

```
НазваКласу назваПосилання;
```

Приклад оголошення посилання на об’єкт класу `StartClass`:

```
StartClass obj;
```

Ініціалізація посилання на об’єкт класу здійснюється за допомогою оператора `new` і вказування конструктора, який має збудувати об’єкт. Одержаний в результаті цих операцій об’єкт розташується у області оперативної пам’яті що зветься *”куча”*. Ініціалізація посилання на об’єкт класу за допомогою конструктора за замовчуванням має такий синтаксис:

```
назваПосилання = new НазваКонструтора();
```

Приклад ініціалізації посилання на об’єкт класу `StartClass`:

```
obj = new StartClass();
```

При створенні об’єктів дозволяється суміщати оголошення та ініціалізацію об’єктів, а також створювати анонімні об’єкти. Якщо посилання на об’єкт не посилається на жоден об’єкт, то йому слід присвоїти значення `null`. На відміну від полів-посилань на об’єкти, локальні змінні-посилання на об’єкти не ініціалізуються значенням `null` при оголошенні. Для них ініціалізацію посилання слід проводити явно.

Методи

Метод – функція-член класу, яка призначена маніпулювати станом об’єкту класу. Методи можуть бути перевантаженими. Перевантаження методів відбувається шляхом вказування різної кількості параметрів та їх типів методам з однаковими назвами.

Синтаксис оголошення методу наступний:

```
[СпецифікаторДоступу] [static] [final] Тип назваМетоду([параметри]) [throws класи]
{
	[Тіло методу]
	[return [значення]];
}
```

Конструктори, методи, та поля класу можуть бути відкритими (`public`), закритими (`private`) та захищеними (`protected`), що визначається специфікатором
доступу.

Специфікатор доступу `public` робить елемент класу загальнодоступним в межах пакету (набору класів, з яких складається програма).

Специфікатор доступу `private` робить елемент класу закритим (недоступним) для всіх зовнішніх відносно даного класу елементів програми (включаючи похідні класи).

Специфікатор доступу `protected` робить елемент класу закритим (недоступним) для всіх зовнішніх відносно даного класу елементів програми, проте цей елемент буде загальнодоступним для похідних класів.

Якщо будь-який елемент класу не має специфікатора доступу, то цей елемент автоматично стає відкритим та видимим у межах пакету (не плутати з `public`).

Всі елементи класу, що оголошені без використання ключового слова `static`, належать об'єкту класу. Тобто, кожен об'єкт класу містить власну копію цих елементів класу. Ключове слово `static` робить поле або метод членом класу, а не об'єкту, тобто вони є спільними для всіх об'єктів класу.

Оскільки клас існує завжди, на відміну від об'єктів, які створюються в процесі роботи програми, то статичні елементи класу доступні навіть тоді, коли ще не створено жодного об'єкту класу. Цей підхід використовується при написанні методу `main` з якого починається виконання консольної програми, бо на момент її запуску ще не існує жодного об'єкту.

Метод може генерувати виключну ситуацію. Якщо виключна ситуація не опрацьовується у тілі методу, то вона повинна бути описана в оголошені методу після ключового слова `throws`. Якщо виключна ситуація опрацьовується у тілі методу, то цього робити не потрібно.

Передача параметрів у метод відбувається по значенню шляхом копіювання значень реальних параметрів у формальні параметри методу. Якщо ці значення є простими типами, то відбудеться копіювання значень. Якщо ці значення є посиланнями, то копіюватимуться не об'єкти, а посилання на об'єкти. Таким чином зміна значення посилання формального параметру в середині методу не вплине на значення посилання за його межами.

Вихід та повернення значення з методу відбувається за допомогою оператора `return`. Якщо метод не повертає значення, то оператор `return` можна опустити. Перед поверненням з методу значення об'єкту, що може змінювати свій стан, слід обов'язково скористатися методом `clone` об'єкту, який створює його копію.

Синтаксис виклику нестатичного методу:

```
НазваОб'єкту.назваМетоду([параметри]);
```

Синтаксис виклику статичного методу має 2 види:

```
НазваОб'єкту.назваМетоду([параметри]); // через об'єкт класу
НазваКласу.назваМетоду([параметри]); // через назву класу
```

Хід роботи

Варіант 11

Монітор (Display)

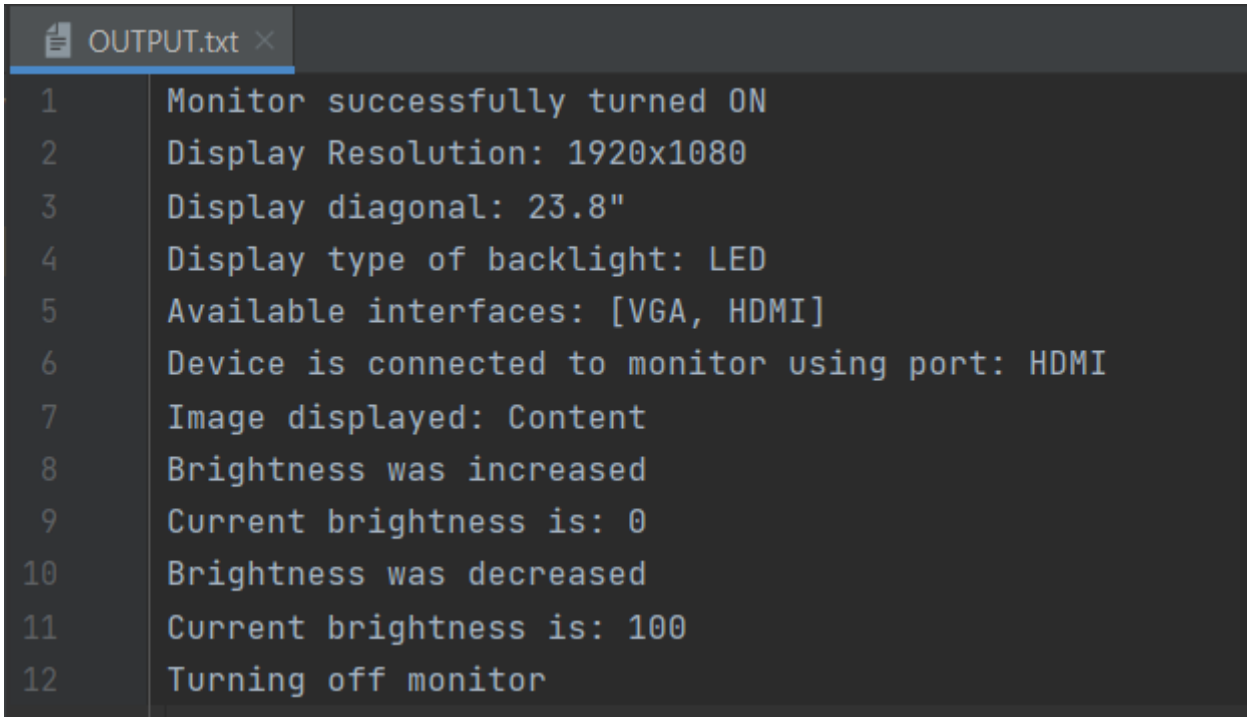
```
package vitkor.ki.objects;

import vitkor.ki.enums.BacklightType;
import vitkor.ki.enums.Color;
import vitkor.ki.enums.MonitorInterface;
import vitkor.ki.enums.MonitorStatus;

import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.util.Set;

public class Monitor {

    private static final String OUTPUT_FILE = "OUTPUT.txt";

    private BacklightType backlightType;
    private Resolution displayResolution;
    private Double diagonal;
    private Color frameColor;
    private Color standColor;
    private Set<MonitorInterface> availablePorts;

    private Brightness brightness;
    private MonitorInterface currentUsedPort;
    private MonitorStatus currentStatus = MonitorStatus.ON;

    private PrintWriter printWriter;

    public Monitor(Resolution displayResolution, Double diagonal,
                   BacklightType backlightType, Set<MonitorInterface>
availablePorts,
                   Brightness brightness) {
        this.displayResolution = displayResolution;
        this.diagonal = diagonal;
        this.backlightType = backlightType;
        this.availablePorts = availablePorts;
        this.frameColor = Color.BLACK;
        this.standColor = Color.BLACK;
        this.brightness = brightness;
```

```
    }

    public Monitor(Resolution displayResolution, Double diagonal,
                   BacklightType backlightType, Set<MonitorInterface>
availablePorts,
                   Brightness brightness,
                   Color frameColor, Color standColor) {
        this.backlightType = backlightType;
        this.displayResolution = displayResolution;
        this.diagonal = diagonal;
        this.frameColor = frameColor;
        this.standColor = standColor;
        this.availablePorts = availablePorts;
        this.brightness = brightness;
    }


    /**
     * Turning on monitor, status moved to ON
     */
    public void turnOnMonitor() {
        try {
            printWriter = new PrintWriter(OUTPUT_FILE);
            this.currentStatus = MonitorStatus.ON;
            printWriter.println("Monitor successfully turned ON");
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
    }

    /**
     * Turning off monitor, status moved to OFF
     */
    public void turnOffMonitor() {
        if (MonitorStatus.ON.equals(currentStatus)) {
            printWriter.println("Turning off monitor");
            printWriter.flush();
            printWriter.close();
            currentStatus = MonitorStatus.OFF;
        }
    }

    /**
     * Connecting to monitor using specified port if it available
     *
     * @param port used port
     */
    public void connectToMonitorUsingPort(MonitorInterface port) {
        if (availablePorts.contains(port)) {
            currentUsedPort = port;
            printWriter.println("Device is connected to monitor using port: " +
port);
        } else {
            printWriter.println("Device could not connect to monitor because
this port: " + port + " is not available");
        }
    }

    /**
     * Increasing of brightness value
     *
     * @param value how much to increase
     */
    public void increaseBrightness(Integer value) {
        int newBrightnessValue = brightness.getCurrentValue() + value;
        int brightnessMaxValue = brightness.getMaxValue();
```

```
        brightness.setCurrentValue(Math.min(brightnessMaxValue,
newBrightnessValue));
        printWriter.println("Brightness was increased");
        printWriter.println("Current brightness is: " +
brightness.getCurrentValue());
    }

    /**
     * Decreasing of brightness value
     *
     * @param value how much to decrease
     */
    public void decreaseBrightness(Integer value) {
        int newBrightnessValue = brightness.getCurrentValue() - value;
        int brightnessMinValue = brightness.getMinValue();
        brightness.setCurrentValue(Math.max(brightnessMinValue,
newBrightnessValue));
        printWriter.println("Brightness was decreased");
        printWriter.println("Current brightness is: " +
brightness.getCurrentValue());
    }

    /**
     * Displaying of an image
     * @param image - image with content
     */
    public void displayImage(Image image) {
        printWriter.println("Image displayed: " + image.getContent());
    }

    /**
     * Show monitor information
     */
    public void viewMonitorInformation(){
        if (MonitorStatus.ON.equals(currentStatus)) {
            printWriter.println("Display Resolution: " +
displayResolution.getWidth() + "x" + displayResolution.getHeight());
            printWriter.println("Display diagonal: " + diagonal + "\"");
            printWriter.println("Display type of backlight: " + backlightType);
            printWriter.println("Available interfaces: " + availablePorts);
        }
    }

    //GETTERS & SETTERS

    public Resolution getDisplayResolution() {
        return displayResolution;
    }

    public void setDisplayResolution(Resolution displayResolution) {
        this.displayResolution = displayResolution;
    }

    public Double getDiagonal() {
        return diagonal;
    }

    public void setDiagonal(Double diagonal) {
        this.diagonal = diagonal;
    }

    public BacklightType getBacklightType() {
        return backlightType;
    }

    public void setBacklightType(BacklightType backlightType) {
```

```
        this.backlightType = backlightType;
    }

    public Color getFrameColor() {
        return frameColor;
    }

    public void setFrameColor(Color frameColor) {
        this.frameColor = frameColor;
    }

    public Color getStandColor() {
        return standColor;
    }

    public void setStandColor(Color standColor) {
        this.standColor = standColor;
    }

    public Set<MonitorInterface> getAvailablePorts() {
        return availablePorts;
    }

    public void setAvailablePorts(Set<MonitorInterface> availablePorts) {
        this.availablePorts = availablePorts;
    }

    public MonitorStatus getCurrentStatus() {
        return currentStatus;
    }

    public void setCurrentStatus(MonitorStatus currentStatus) {
        this.currentStatus = currentStatus;
    }

    public MonitorInterface getCurrentUsedPort() {
        return currentUsedPort;
    }

    public Brightness getBrightness() {
        return brightness;
    }

    public void setBrightness(Brightness brightness) {
        this.brightness = brightness;
    }
}
```

Вміст файлу:



Висновок: Я ознайомився з класами і пакетами на мові програмування Java.