Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра ЕОМ

Звіт

з лабораторної роботи №3

з дисципліни: «Кросплатформенні засоби програмування»

на тему: «Класи та пакети»

Виконав: ст.гр. КІ-34

Лис Б. Л.

Прийняв:

викл. каф. ЕОМ

Іванов Ю. С.

Львів 2022

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

Завдання:

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.

Варіант: 11.

Монітор

Лістинг програми:

Файл MonitorApp.java

```
/**
 * lab 3 package
 */

package KI34.Lys.Lab3;

import static java.lang.System.out;
import java.io.*;
/**
 * Monitor Application class implements main method for Monitor
 class possibilities demonstration
 * @author Lys Bohdan
 * @version 1.0
 */

public class MonitorApp {
    /**
     * @param args param
     * @throws FileNotFoundException
     */
    public static void main(String[] args)  throws FileNotFoundException {

        Monitor object = new Monitor("MSI", 24, "LED", "1920 x 1080", 144, 100,
"16:9");
```

```
        object.off_onMonitor(Status.ON);
        object.getSettings();
        object.brightDisplayPlus(25);
        object.brightDisplayMinus(63);

        object.changeMatrix("IPS");
        object.changeHz(75);
        object.changeResolution("1080x720");
        object.changeMonitorFormat("4:3");
        object.getSettings();

        object.off_onMonitor(Status.OFF);
        object.dispose();
    }
}
```

Файл Monitor.java

```
/**
 * lab 3 package
 */
package KI34.Lys.Lab3;

import java.io.*;

/**
 * Class <code>Monitor</code> implements monitor
 * * @author Lys Bohdan
 *  * @version 1.0
 **/

public class Monitor {
    private double hz;
    private String name;
    private String resolution;
    private double diagonal;
    private String matrix;
    private double BrightDisplay;
    private Status status;
    private PrintWriter myWrite;
    private String monitorFormat;

    /**
     * Constructor
     * @throws FileNotFoundException param
     */

    public Monitor() throws FileNotFoundException {
        matrix = "None";
        diagonal = 0;
        name = "None";
        resolution = "None";
        hz = 0;
        BrightDisplay = 0;
        monitorFormat = "None";
        myWrite = new PrintWriter("Log.txt");
    }

    /**
     * Constructor
     * @param name <code>name</code> Name of monitor
     * @param diagonal <code>diagonal</code> Monitor diagonal
     * @param matrix <code>matrix</code> Monitor matrix
     * @param resolution <code>resolution</code> Monitor resolution
     * @param hz <code>hz</code> Monitor refresh rate
     * @param BrightDisplay <code>BrightDisplay</code> The brightness of the
monitor display
```

```
     * @param monitorFormat <code>monitorFormat</code> Monitor screen format
     * @throws FileNotFoundException
     */

    public Monitor(String name, double diagonal, String matrix, String
resolution, double hz, double BrightDisplay, String monitorFormat) throws
FileNotFoundException {
        this.matrix = matrix;
        this.diagonal = diagonal;
        this.name = name;
        this.resolution = resolution;
        this.hz = hz;
        this.BrightDisplay = BrightDisplay;
        this.monitorFormat = monitorFormat;
        myWrite = new PrintWriter("Log.txt");
    }

    /**
     * Method returns monitor`s refresh rate
     * @return monitor`s refresh rate
     */
    public double getHz() {
        return hz;
    }

    /**
     * Method returns monitor`s name
     * @return monitor`s name
     */
    public String getName() {
        return name;
    }

    /**
     * Method returns monitor`s resolution
     * @return monitor`s resolution
     */
    public String getResolution() {
        return resolution;
    }

    /**
     * Method returns monitor`s diagonal
     * @return monitor`s diagonal
     */
    public double getDiagonal() {
        return diagonal;
    }

    /**
     * Method returns monitor`s matrix type
     * @return monitor`s matrix type
     */
    public String getMatrix() {
        return matrix;
    }

    /**
     * Method returns monitor`s display brightness
     * @return monitor`s display brightness
     */
    public double getBrightDisplay() {
        return BrightDisplay;
    }

    /**
```

```
     * Method sets the new monitor`s refresh rate
     * @param hz <code>hz</code> monitor`s refresh rate
     */
    public void setHz(double hz) {
        this.hz = hz;
    }

    /**
     * Method sets the new monitor`s name
     * @param name <code>name</code> monitor`s name
     */
    public void setName(String name) {
        this.name = name;
    }

    /**
     * Method sets the new monitor`s resolution and format
     * @param resolution <code>resolution</code> monitor`s resolution
     * @param monitorFormat <code>monitorFormat</code> monitor`s screen format
     */
    public void setResolution(String resolution, String monitorFormat) {
        this.resolution = resolution;
        this.monitorFormat = monitorFormat;
    }

    /**
     * Method sets the new monitor`s diagonal
     * @param diagonal <code>diagonal</code> monitor`s diagonal
     */
    public void setDiagonal(double diagonal) {
        this.diagonal = diagonal;
    }

    /**
     * Method sets the new monitor`s matrix type
     * @param matrix <code>matrix</code> monitor`s matrix type
     */
    public void setMatrix(String matrix) {
        this.matrix = matrix;
    }

    /**
     * Method sets the new monitor`s display brightness
     * @param brightDisplay <code>brightDisplay</code> monitor`s display
brightness
     */
    public void setBrightDisplay(double brightDisplay) {
        BrightDisplay = brightDisplay;
    }

    /**
     * Method simulates monitor`s off/on
     */
    public void off_onMonitor(Status comand) {
        if (comand == Status.ON) {
            myWrite.println("Монітор включено.");
            status = Status.ON;
        } else {
            status = Status.OFF;
            myWrite.println("Монітор виключено.");
        }
    }

    /**
     * Method return monitor`s settings which we set
     */
```

```
    public void getSettings() {
        if (status == Status.ON) {
            System.out.println("Name - " + name);
            System.out.println("Diagonal - " + diagonal);
            System.out.println("Resolution - " + resolution);
            System.out.println("Hz - " + hz);
            System.out.println("Matrix - " + matrix);
            System.out.println("Bright - " + BrightDisplay);
            myWrite.println("Налаштування монітора виведенно.");
        } else {
            System.out.println("Your monitor is off!!! ");
            myWrite.println("Налаштування монітора не вивиеденно, та як монітор
виключенно.");
        }

    }

    /**
     * Method simulates increment monitor`s display brightness
     */
    public void brightDisplayPlus(int change) {
        BrightDisplay = BrightDisplay + change;
        myWrite.println("Яскравість монітора збільшенна.");
    }

    /**
     * Method simulates decrement monitor`s display brightness
     */
    public void brightDisplayMinus(int change) {
        BrightDisplay = BrightDisplay - change;
        myWrite.println("Яскравість монітора зменшенна.");
    }

    /**
     * Method simulates changing monitor`s matrix type
     */
    public void changeMatrix(String mat) {
        System.out.println("Selected matrix - " + matrix);
        myWrite.println("Матриця монітора зміненна з " + matrix + " на " + mat);
        matrix = mat;
        System.out.println("Matrix changed - " + matrix);
    }

    /**
     * Method simulates changing monitor`s refresh rate
     */
    public void changeHz(int hz) {
        if (status == Status.ON) {
            System.out.println("Selected Hz - " + this.hz);
            myWrite.println("Частота оновлення монітура зміненна з " + this.hz +
" на " + hz);
            this.hz = hz;
            System.out.println("Hz changed - " + hz);

        } else {
            System.out.println("Your monitor is off!!! ");
            myWrite.println("Частота оновлення монітора не зміненна, та як
монітор виключенно.");
        }
    }

    /**
     * Method simulates changing monitor`s name
     */
    public void changeName(String name) {
        System.out.println("Selected name - " + name);
```

```
        myWrite.println("Ноазва монітора зміненна з " + this.name + " на " +
name);
        this.name = name;
        System.out.println("Name changed - " + name);

    }

    /**
     * Method simulates changing monitor`s resolution
     */
    public void changeResolution(String resolution) {
        if (status == Status.ON) {
            System.out.println("Selected resolution - " + this.resolution);
            myWrite.println("Розширення монітура зміненно з " + this.resolution
+ " на " + resolution);
            this.resolution = resolution;
            System.out.println("Resolution changed - " + this.resolution);

        } else {
            System.out.println("Your monitor is off!!! ");
            myWrite.println("Розширення монітора не зміненно, та як монітор
виключенно.");
        }
    }

    /**
     * Method simulates changing monitor`s screen format
     */
    public void changeMonitorFormat(String monitorFormat) {
        if (monitorFormat == "16:9") {
            this.monitorFormat = monitorFormat;
            System.out.println("Monitor format changed - " + monitorFormat);
            System.out.println("Max resolution for this format is 2560 x
1440.");
            myWrite.println("Формат монітора зміненно на - " + monitorFormat);
        } else if (monitorFormat == "4:3") {
            this.monitorFormat = monitorFormat;
            System.out.println("Monitor format changed - " + monitorFormat);
            System.out.println("Max resolution for this format is 1440x1080.");
            myWrite.println("Формат монітора зміненно на - " + monitorFormat);
        } else if (monitorFormat == "16:10") {
            this.monitorFormat = monitorFormat;
            System.out.println("Monitor format changed - " + monitorFormat);
            System.out.println("Max resolution for this format is 1920x1200.");
            myWrite.println("Формат монітора зміненно на - " + monitorFormat);
        } else {
            System.out.println("Input incorrect monitor format!");
            myWrite.println("Формат монітора не зміненно, некоректне
введення.");
        }
    }

    /**
     * Method releases used recourses
     */
    public void dispose()
    {
        myWrite.flush();
        myWrite.close();
    }
}
```

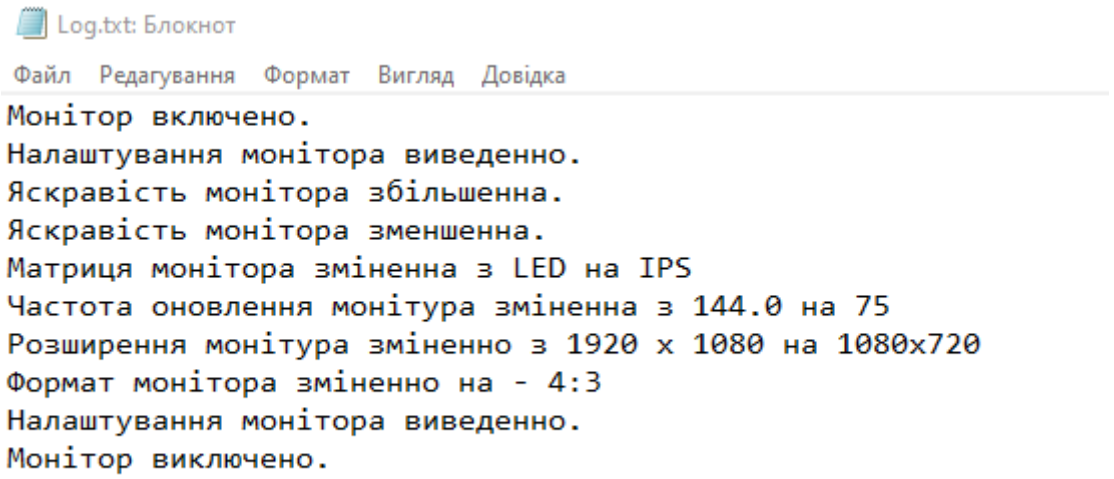
Файл Status.java

```java
/**
 * lab 3 package
 */
package KI34.Lys.Lab3;
/**
 * Method simulates off/on
 */
public enum Status {
    ON, OFF
}
```

Результат виконання програми:

```
C:\Users\mrpet\.jdks\openjdk-18.0.2\bin\java.exe "-javaagent:C:\Program Fil
Name - MSI
Diagonal - 24.0
Resolution - 1920 x 1080
Hz - 144.0
Matrix - LED
Bright - 100.0
Selected matrix - LED
Matrix changed - IPS
Selected Hz - 144.0
Hz changed - 75
Selected resolution - 1920 x 1080
Resolution changed - 1080x720
Monitor format changed - 4:3
Max resolution for this format is 1440x1080.
Name - MSI
Diagonal - 24.0
Resolution - 1080x720
Hz - 75.0
Matrix - IPS
Bright - 62.0

Process finished with exit code 0
```



Протокол діяльності

Package KI34.Lys.Lab3

package KI34.Lys.Lab3

All Classes and Interfaces	Classes	Enum Classes

Class	Description
Monitor	Class Monitor implements monitor * @author student * @version 1.0
MonitorApp	Monitor Application class implements main method for Monitor class possibilities demonstration
Status	Method simulates off/on

Згенерована документація

Package KI34.Lys.Lab3

Class Monitor

java.lang.Object
  KI34.Lys.Lab3.Monitor

public class **Monitor**
extends Object

Class Monitor implements monitor * @author student * @version 1.0

Constructor Summary

Constructors	
Constructor	Description
Monitor()	Constructor
Monitor(String name, double diagonal, String matrix, String resolution, double hz, double BrightDisplay, String monitorFormat)	Constructor

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
void	brightDisplayMinus(int change)	Method simulates decrement monitor`s display brightness
void	brightDisplayPlus(int change)	Method simulates increment monitor`s display brightness
void	changeHz(int hz)	Method simulates changing monitor`s refresh rate
void	changeMatrix(String mat)	Method simulates changing monitor`s matrix type
void	changeMonitorFormat(String monitorFormat)	Method simulates changing monitor`s screen format
void	changeName(String name)	Method simulates changing monitor`s name
void	changeResolution(String resolution)	Method simulates changing monitor`s resolution
void	dispose()	Method releases used recourses
double	getBrightDisplay()	Method returns monitor`s display brightness
double	getDiagonal()	Method returns monitor`s diagonal
double	getHz()	Method returns monitor`s refresh rate
String	getMatrix()	Method returns monitor`s matrix type
String	getName()	Method returns monitor`s name
String	getResolution()	Method returns monitor`s resolution
void	getSettings()	Method return monitor`s settings which we set
void	off_onMonitor(Status comand)	Method simulates monitor`s off/on
void	setBrightDisplay(double brightDisplay)	Method sets the new monitor`s display brightness
void	setDiagonal(double diagonal)	Method sets the new monitor`s diagonal
void	setHz(double hz)	Method sets the new monitor`s refresh rate
void	setMatrix(String matrix)	Method sets the new monitor`s matrix type
void	setName(String name)	Method sets the new monitor`s name

Інформація про клас Monitor

Package KI34.Lys.Lab3

Class MonitorApp

java.lang.Object
    KI34.Lys.Lab3.MonitorApp

```
public class MonitorApp
extends Object
```

Monitor Application class implements main method for Monitor class possibilities demonstration

Version:

1.0

Author:

student

Constructor Summary

Constructors

Constructor	Description
MonitorApp()	

Method Summary

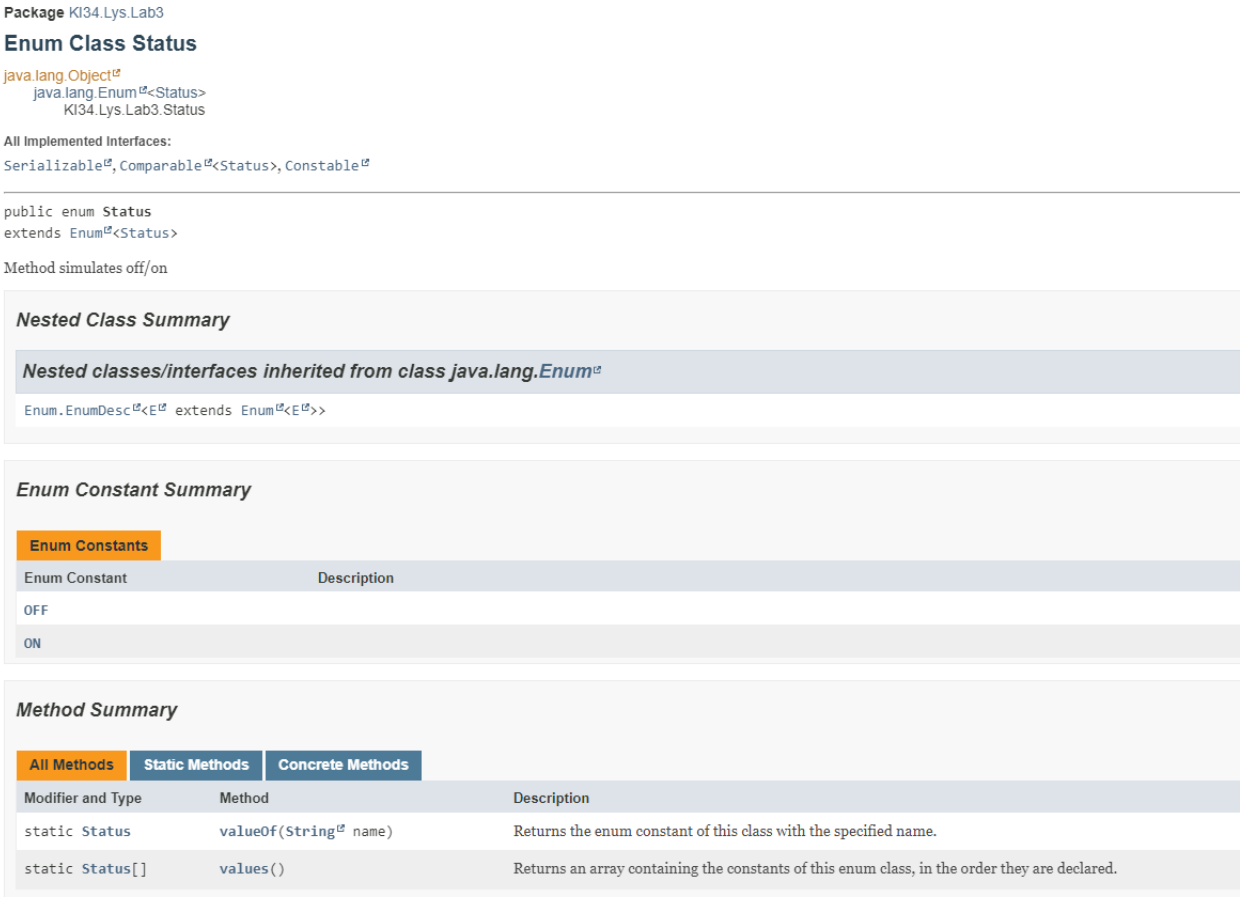
All Methods | Static Methods | Concrete Methods

Modifier and Type	Method	Description
static void	main(String[] args)	

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Інформація про клас MonitorApp



Package KI34.Lys.Lab3

Enum Class Status

java.lang.Object
  java.lang.Enum<Status>
    KI34.Lys.Lab3.Status

All Implemented Interfaces:

Serializable, Comparable<Status>, Constable

```
public enum Status
extends Enum<Status>
```

Method simulates off/on

Nested Class Summary

Nested classes/interfaces inherited from class java.lang.Enum

Enum.EnumDesc<E extends Enum<E>>

Enum Constant Summary

Enum Constants

Enum Constant	Description
OFF	
ON	

Method Summary

All Methods | **Static Methods** | **Concrete Methods**

Modifier and Type	Method	Description
static Status	valueOf(String name)	Returns the enum constant of this class with the specified name.
static Status[]	values()	Returns an array containing the constants of this enum class, in the order they are declared.

Інформація про клас EnumStatus

Відповіді на контрольні запитання:

1. **Синтаксис визначення класу?**

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

2. **Синтаксис визначення методу?**

[СпецифікаторДоступу] [static] [final] Тип назваМетоду([параметри]) [throws класи]

{

    [Тіло методу]

    [return [значення]];

}

3. **Синтаксис оголошення поля?**

   [СпецифікаторДоступу] [static] [final] Тип НазваПоля [= ПочатковеЗначення];

4. **Як оголосити та ініціалізувати константне поле?**

   [СпецифікаторДоступу] [final] Тип НазваПоля [= ПочатковеЗначення];

5. **Які є способи ініціалізації полів?**
   - у конструкторі;
   - явно при оголошені поля;
   - у блоці ініціалізації (виконується перед виконанням конструктора).
6. **Синтаксис визначення конструктора?**

   [СпецифікаторДоступу] НазваКласу([параметри])
   {
   Тіло конструктора
   }

7. **Синтаксис оголошення пакету?**

   package НазваПакету{.НазваПідпакету};

8. **Як підключити до програми класи, що визначені в зовнішніх пакетах?**

   Доступ до класів з інших пакетів можна отримати двома шляхами:

1. вказуючи повне ім’я пакету перед іменем кожного класу.
2. використовуючи оператор import, що дозволяє підключати як один клас так і всі загальнодоступні класи пакету, позбавляючи необхідності записувати імена класів з вказуванням повної назви пакету перед ними.

9. **В чому суть статичного імпорту пакетів?**

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

10. Які вимоги ставляться до файлів і каталогів при використанні пакетів?

Використання пакетів вимагає, щоб файли і каталоги проекту та їх ієрархія були строго структурованими. Так назви пакету і його підпакетів мають співпадати з назвами каталогів, де вони розміщуються. Назви загальнодоступних класів мають співпадати з назвами файлів, де вони розміщуються. Ієрархія каталогів і файлів проекту має співпадати з ієрархією пакетів. Після компіляції ієрархія каталогів, де містяться файли класів, співпадає з ієрархією каталогів проекту.

Висновок:

На цій лабораторній роботі, я ознайомитися з процесом розробки класів та пакетів мовою Java.