Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Кафедра «Електронних обчислювальних машин»

Звіт
з лабораторної роботи № 2
з дисципліни: «Кросплатформенні засоби програмування»
на тему: «Класи та пакети»

Виконав:
студент групи *КІ-306*
Ярема Максим
Прийняв:
доцент кафедри ЕОМ
Іванов Ю. С.

Львів – 2023

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

Завдання (варіант № 29)

1. Написати та налагодити програму на мові Java, що реалізує у вигляді класу предметну область згідно варіанту(29. Протигаз). 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

Вихідний код програми

Файл GasMaskApplication.java

```
package ki306.yarema.lab3;

import ki306.yarema.lab3.models.Filter;
import ki306.yarema.lab3.models.GasMask;

import java.io.IOException;

/**
* Gas Mask Application class implements main method for GasMask
class possibilities demonstration
*
* @author Yarema Maksym
* @version 1.0
* @since version 1.0
*/
```

```
public class GasMaskApplication {
/**
* @param args
*/
public static void main(String[] args) {
GasMask gasMask = new GasMask();

gasMask.checkStatus();

gasMask.breathe();
gasMask.breathe();

gasMask.sealMask();

gasMask.breathe();

gasMask.turnNightVisionOn();

gasMask.breathe();

gasMask.replaceFilter(new Filter("A1B1E1K1"));

gasMask.checkStatus();

gasMask.cleanAndReplaceFilter();

try {
gasMask.dispose();
} catch (IOException e) {
e.printStackTrace();
}
}
}
```

Файл GasMask.java

```
package ki306.yarema.lab3.models;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
/**
* Class Lab2YaremaKI306 implements laboratory work №3
* A class representing a gas mask and allowing various operations with it.
* The gas mask has a type, a filter, a state (on or off), a usage count,
* a sealing state, and a cleanliness state.
*
* @author Yarema Maksym
* @version 1.0
* @since version 1.0
*
*/
public class GasMask {
private final String type;
private Filter filter;
private boolean isNightVisionOn;
private int usageCount;
private boolean isSealed;
private boolean isClean;
private String logFileName;
private FileWriter logFileWriter;

/**
* Constructor for creating a gas mask with default values.
*  Sets the mask type to "EN14387," the filter type to "A2B2E2K2," and other initial
values.
*/
public GasMask() {
type = "EN14387";
filter = new Filter("A2B2E2K2");
```

```
isNightVisionOn = false;
usageCount = 0;
isSealed = false;
isClean = true;
logFileName = "GasMaskLog.txt";

try {
logFileWriter = new FileWriter(new File(logFileName));
} catch (IOException e) {
e.printStackTrace();
}
}

/**
* Constructor for creating a gas mask with specified parameters.
* @param maskType The type of the gas mask.
* @param filterType The type of the filter.
*/
public GasMask(String maskType, String filterType) {
type = maskType;
filter = new Filter(filterType);
isNightVisionOn = false;
usageCount = 0;
isSealed = false;
isClean = true;
logFileName = "GasMaskLog.txt";

try {
logFileWriter = new FileWriter(new File(logFileName));
} catch (IOException e) {
e.printStackTrace();
}
}

/**
* Turn on night vision mode for the gas mask.
*/
public void turnNightVisionOn() {
isNightVisionOn = true;
log("Night vision mode is turned on.");
}

/**
* Turn off night vision mode for the gas mask.
*/
public void turnNightVisionOff() {
isNightVisionOn = false;
log("Night vision mode is turned off.");
}

/**
* Replace the filter of the gas mask with a new one.
* @param newFilter The new filter.
*/
public void replaceFilter(Filter newFilter) {
filter = newFilter;
log("Filter replaced with a new one.");
}

/**
* Breathe through the gas mask.
*/
public void breathe() {
if (isSealed && filter.isEffective()) {
incrementUsage();
log("Breathed safely using the gas mask.");
} else {
log("Gas mask is not sealed properly or the filter is not effective.");
}
}

/**
* Seal the gas mask.
*/
```

```
public void sealMask() {
isSealed = true;
log("Gas mask is sealed.");
}

/**
* Unseal the gas mask.
*/
public void unsealMask() {
isSealed = false;
log("Gas mask is unsealed.");
}

/**
* Clean the gas mask.
*/
public void cleanMask() {
isClean = true;
log("Gas mask is cleaned.");
}

/**
* Increment the usage count of the gas mask.
*/
public void incrementUsage() {
usageCount++;
}

/**
* Clean the gas mask and replace the filter with a new one.
*/
public void cleanAndReplaceFilter() {
if (!isClean) {
cleanMask();
}
replaceFilter(new Filter("A2B2E2K2"));
log("Gas mask is cleaned and the filter is replaced.");
}

/**
* Check the status of the gas mask and display status information.
*/
public void checkStatus() {
String status = "Gas Mask Status:\n";
status += "Type: " + type + "\n";
status += "Filter Type: " + filter.getType() + "\n";
status += "Is Night Vision On: " + isNightVisionOn + "\n";
status += "Usage Count: " + usageCount + "\n";
status += "Is Sealed: " + isSealed + "\n";
status += "Is Clean: " + isClean + "\n";
log(status);
}

/**
* Write a message to the log file.
* @param message The message to be logged.
*/
public void log(String message) {
try {
logFileWriter.write(message + "\n");
logFileWriter.flush();
} catch (IOException e) {
e.printStackTrace();
}
}

/**
* Dispose of the gas mask and close the log file.
* @throws IOException Thrown in case of input/output errors.
*/
public void dispose() throws IOException {
logFileWriter.close();
}
}
```

Файл Filter.java

```
package ki306.yarema.lab3.models;

/**
* A class representing a filter used in a gas mask and allowing various operations with
it.
* The filter has a type and an effectiveness state.
*
* @author Yarema Maksym
* @version 1.0
* @since version 1.0
*/
public class Filter {
private final String type;
private boolean isEffective;

/**
* Constructor for creating a filter with default values.
* Sets the filter type to "A2B2E2K2" and initializes it as effective.
*/
public Filter() {
type = "A2B2E2K2";
isEffective = true;
}

/**
* Constructor for creating a filter with a specified type.
* @param type The type of the filter.
*/
public Filter(String type) {
this.type = type;
isEffective = true;
}

/**
* Make the filter ineffective.
*/
public void makeIneffective() {
isEffective = false;
}

/**
* Check if the filter is effective.
* @return true if the filter is effective, false otherwise.
*/
public boolean isEffective() {
return isEffective;
}

/**
* Get the type of the filter.
* @return The type of the filter.
*/
public String getType() {
return type;
}
}
```

Результат виконання програми

GasMaskLog.txt:

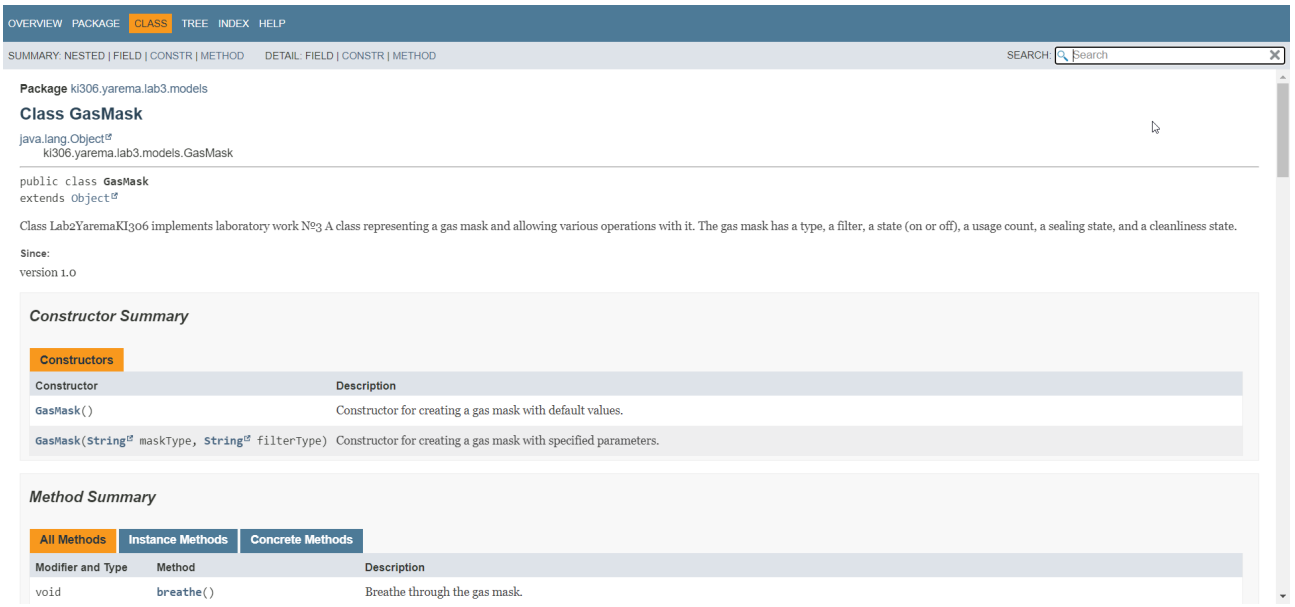
Main.java | GasMaskLog.txt

```
Gas mask is turned on.
Gas Mask Status:
Type: EN14387
Filter Type: A2B2E2K2
Is On: true
Usage Count: 0
Is Sealed: false
Is Clean: true

Gas mask is not sealed properly or the filter is not effective.
Gas mask is not sealed properly or the filter is not effective.
Gas mask is sealed.
Breathed safely using the gas mask.
Breathed safely using the gas mask.
Filter replaced with a new one.
Gas Mask Status:
Type: EN14387
Filter Type: A1B1E1K1
Is On: true
Usage Count: 2
Is Sealed: true
Is Clean: true

Filter replaced with a new one.
Gas mask is cleaned and the filter is replaced.
```

Фрагмент згенерованої документації



OVERVIEW PACKAGE CLASS TREE INDEX HELP

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

Package ki306.yarema.lab3.models

Class GasMask

java.lang.Object
ki306.yarema.lab3.models.GasMask

```
public class GasMask
extends Object
```

Class Lab2YaremaKI306 implements laboratory work №3 A class representing a gas mask and allowing various operations with it. The gas mask has a type, a filter, a state (on or off), a usage count, a sealing state, and a cleanliness state.

Since:
version 1.0

Constructor Summary

Constructors

Constructor	Description
GasMask()	Constructor for creating a gas mask with default values.
GasMask(String maskType, String filterType)	Constructor for creating a gas mask with specified parameters.

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
void	breathe()	Breathe through the gas mask.

Відповіді на контрольні запитання

1. Синтаксис визначення класу.
   - public class ClassName {
     // Class members (fields, methods, constructors)
     }

2. Синтаксис визначення методу.

- public returnType methodName(parameters) {
  // Method body
  }

3. Синтаксис оголошення поля.
   - accessModifier dataType fieldName;

4. Як оголосити та ініціалізувати константне поле?
   - public static final dataType CONSTANT_NAME = initial_value;

5. Які є способи ініціалізації полів?
   - Явна ініціалізація при оголошенні поля.
   - Ініціалізація у конструкторі класу.
   - Ініціалізація у блоку ініціалізації (конструкторі, статичному або звичайному).

6. Синтаксис визначення конструктора.
   - public ClassName(parameters) {
     // Constructor body
     }

7. Синтаксис оголошення пакету.
   - package packageName.subpackage;

8. Як підключити до програми класи, що визначені в зовнішніх пакетах?
   - Вказати повне ім'я класу перед використанням (наприклад, java.util.Date today = new java.util.Date();).
   - Використовувати оператор import для підключення класів з інших пакетів, щоб уникнути повторення повного імені класу.

9. В чому суть статичного імпорту пакетів?
   - Статичний імпорт дозволяє підключити статичні методи і поля класів без повного імені класу.

- Завдяки статичному імпорту, можна використовувати статичні члени класу, не додаваючи перед ними ім'я класу.

10. Які вимоги ставляться до файлів і каталогів при використанні пакетів?
    - Назви пакетів повинні відповідати структурі каталогів.
    - Назви загальнодоступних класів повинні співпадати з назвами файлів, де вони розміщені.
    - Після компіляції ієрархія каталогів проекту повинна відповідати ієрархії пакетів.
    - Для компіляції та запуску програми слід використовувати шляхи до файлів та пакетів.

Висновок

У ході виконання даної лабораторної роботи, я отримав цінні навички розробки класів та пакетів у мові програмування Java. Ця лабораторна робота надала мені можливість ознайомитися з базовими конструкціями Java, такими як оголошення класів, методів та полів. Я навчився правильно структурувати свій код, визначати доступ до класів та їх членів, а також використовувати модифікатори доступу для керування видимістю.