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

Чаус Б.В.

**Прийняв:**

доцент кафедри ЕОМ

Іванов Ю. С.

Львів – 2023

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

**Завдання (варіант № 24)**

*24. Спорядження альпініста*

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.Lab2;

• 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 розробленої програми.

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.

**Вихідний код програми**

Файл AlpinistEquipmentApp.java

/\*\*

\*

\*/

package KI.Chaus.Lab2;

// import KI.Chaus.Lab2.\*;

/\*\*

\*

\*/

class Main {

public static void main(String[] args) {

AlpinistEquipment equipment = new AlpinistEquipment();

equipment.clearLogFile();

equipment.addItem("Climbing Rope", 2, 3.5);

equipment.addItem("Carabiner", 10, 0.15);

equipment.updateItem("Climbing Rope", 3, 4.0);

equipment.displayInventory();

System.out.println("Total Weight: " + equipment.getTotalWeight() + " kg");

System.out.println("Quantity of Climbing Rope: " + equipment.getQuantity("Climbing Rope"));

if (equipment.containsItem("Helmet")) {

System.out.println("Helmet is in inventory.");

} else {

System.out.println("Helmet is not in inventory.");

}

equipment.removeItem("Climbing Rope", 2);

equipment.displayInventory();

System.out.println("Total Weight: " + equipment.getTotalWeight() + " kg");

equipment.removeAllItems();

}

}

Файл AlpinistEquipment.java

package KI.Chaus.Lab2;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.PrintWriter;

import java.util.HashMap;

import java.util.Map;

/\*\*

\* This class represents an inventory system for alpinist equipment.

\* It allows adding, removing, updating, and displaying items in the inventory.

\* It also keeps a log of all operations in a file named "log.txt".

\*

\* @author Chaus Bohdan

\* @version 1.0

\*/

public class AlpinistEquipment {

private Map<String, Integer> quantities;

private Map<String, Double> weights;

/\*\*

\* Constructor to initialize the equipment inventory.

\*/

public AlpinistEquipment() {

quantities = new HashMap<>();

weights = new HashMap<>();

}

/\*\*

\* Adds a specified quantity of an item with its weight to the inventory.

\*

\* @param itemName The name of the item.

\* @param quantity The quantity of the item to be added.

\* @param weight The weight of a single item in kilograms.

\*/

public void addItem(String itemName, int quantity, double weight) {

quantities.put(itemName, quantities.getOrDefault(itemName, 0) + quantity);

weights.put(itemName, weight);

writeToLogFile("Added " + quantity + " " + itemName + "(s) with a total weight of " + (quantity \* weight) + " kg.");

}

/\*\*

\* Removes a specified quantity of an item from the inventory.

\*

\* @param itemName The name of the item.

\* @param quantity The quantity of the item to be removed.

\*/

public void removeItem(String itemName, int quantity) {

if (quantities.containsKey(itemName)) {

int currentQuantity = quantities.get(itemName);

if (currentQuantity >= quantity) {

quantities.put(itemName, currentQuantity - quantity);

writeToLogFile("Removed " + quantity + " " + itemName + "(s).");

} else {

System.out.println("Error: Not enough " + itemName + " in inventory.");

}

} else {

System.out.println("Error: " + itemName + " not found in inventory.");

}

}

/\*\*

\* Calculates and returns the total weight of all items in the inventory.

\*

\* @return The total weight of all items in kilograms.

\*/

public double getTotalWeight() {

double totalWeight = 0;

for (String itemName : quantities.keySet()) {

totalWeight += quantities.get(itemName) \* weights.get(itemName);

}

return totalWeight;

}

/\*\*

\* Displays the current inventory, including item names, quantities, and weights.

\*/

public void displayInventory() {

System.out.println("Inventory:");

for (String itemName : quantities.keySet()) {

System.out.println("Item Name: " + itemName);

System.out.println("Quantity: " + quantities.get(itemName));

System.out.println("Weight: " + weights.get(itemName) + " kg");

}

}

/\*\*

\* Gets the quantity of a specified item in the inventory.

\*

\* @param itemName The name of the item.

\* @return The quantity of the item, or 0 if not found.

\*/

public int getQuantity(String itemName) {

return quantities.getOrDefault(itemName, 0);

}

/\*\*

\* Updates the quantity and weight of a specified item in the inventory.

\*

\* @param itemName The name of the item.

\* @param newQuantity The new quantity of the item.

\* @param newWeight The new weight of a single item in kilograms.

\*/

public void updateItem(String itemName, int newQuantity, double newWeight) {

quantities.put(itemName, newQuantity);

weights.put(itemName, newWeight);

writeToLogFile("Updated " + itemName + " to " + newQuantity + " quantity with a weight of " + newWeight + " kg.");

}

/\*\*

\* Removes all items from the inventory.

\*/

public void removeAllItems() {

quantities.clear();

weights.clear();

writeToLogFile("All items removed from inventory.");

}

/\*\*

\* Checks if a specified item is present in the inventory.

\*

\* @param itemName The name of the item.

\* @return true if the item is present, false otherwise.

\*/

public boolean containsItem(String itemName) {

return quantities.containsKey(itemName);

}

/\*\*

\* Clears the contents of the log file.

\*/

public void clearLogFile() {

File logFile = new File("log.txt");

try {

PrintWriter writer = new PrintWriter(logFile);

writer.close();

} catch (FileNotFoundException e) {

e.printStackTrace();

}

}

/\*\*

\* Writes a message to the log file.

\*

\* @param message The message to be written to the log file.

\*/

private void writeToLogFile(String message) {

try (PrintWriter writer = new PrintWriter(new FileOutputStream(new File("log.txt"), true))) {

writer.println(message);

} catch (FileNotFoundException e) {

e.printStackTrace();

}

}

}

**Результат виконання програми**

Inventory:

Item Name: Climbing Rope

Quantity: 3

Weight: 4.0 kg

Item Name: Carabiner

Quantity: 10

Weight: 0.15 kg

Total Weight: 13.5 kg

Quantity of Climbing Rope: 3

Helmet is not in inventory.

Inventory:

Item Name: Climbing Rope

Quantity: 1

Weight: 4.0 kg

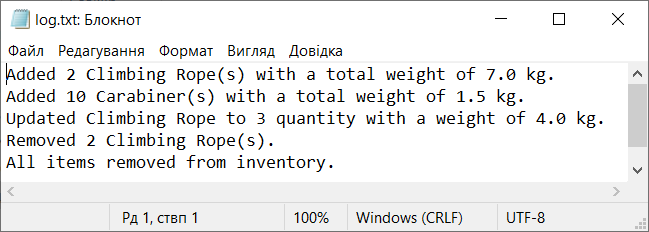
Item Name: Carabiner

Quantity: 10

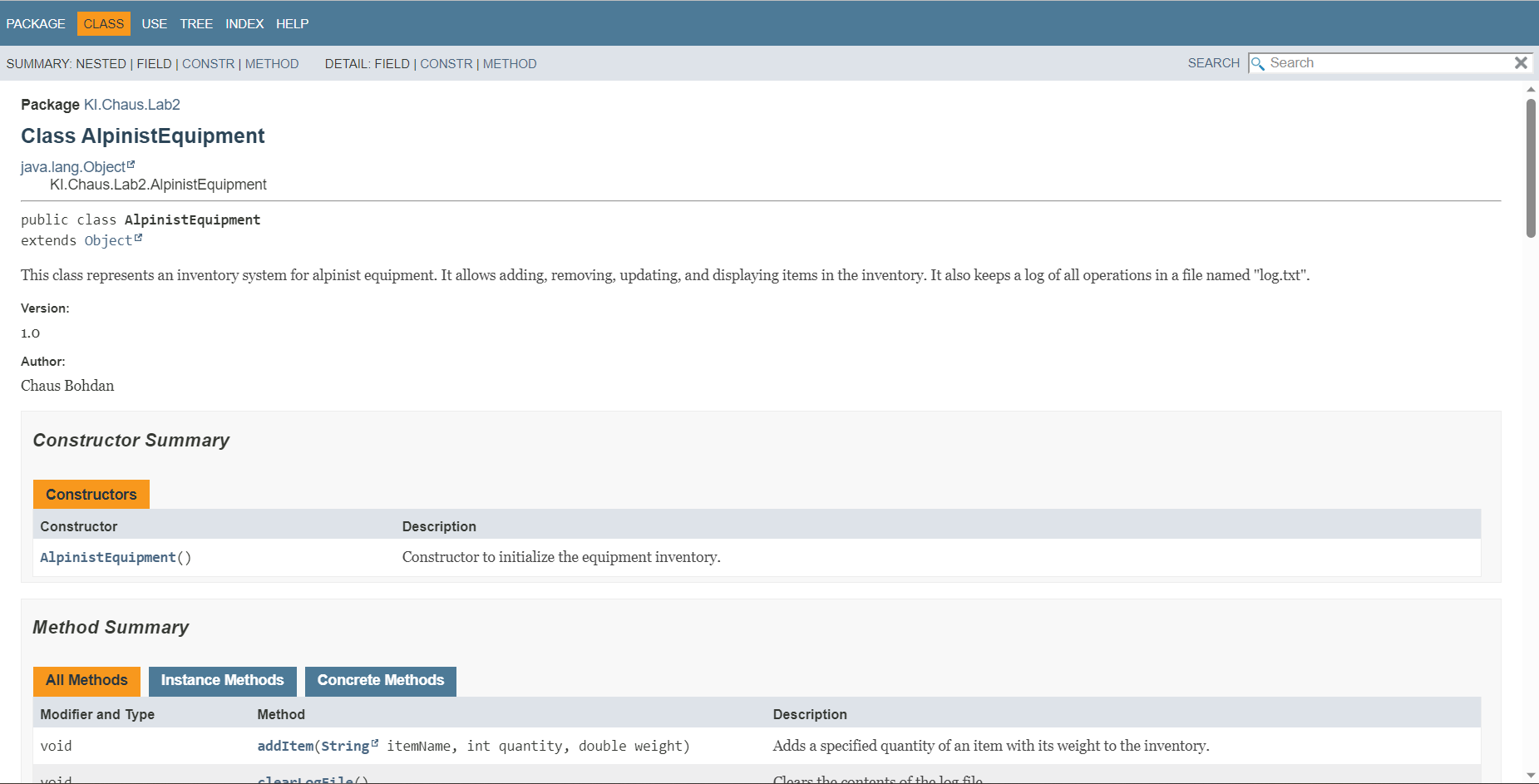
Weight: 0.15 kg

Total Weight: 5.5 kg

**Текстовий файл з результатом виконання програми**



**Фрагмент згенерованої документації**



**Відповіді на контрольні запитання**

1. Синтаксис визначення класу:

public class ClassName { // тіло класу }

1. Синтаксис визначення методу:

public returnType methodName(parameterType parameterName) { // тіло методу return returnValue; // (якщо метод повертає значення) }

1. Синтаксис оголошення поля:

accessModifier dataType fieldName;

1. Як оголосити та ініціалізувати константне поле:

public static final dataType CONSTANT\_NAME = value;

1. Способи ініціалізації полів:
   * Ініціалізація при оголошенні:

dataType fieldName = value;

* + В конструкторі класу.
  + В блоку ініціалізації.

1. Синтаксис визначення конструктора:

public ClassName(parameterType parameterName) { // тіло конструктора }

1. Синтаксис оголошення пакету:

package packageName;

1. Як підключити до програми класи, що визначені в зовнішніх пакетах:

import packageName.ClassName;

1. Суть статичного імпорту пакетів полягає в тому, що можна імпортувати конкретне статичне поле чи метод з класу, і використовувати його без зазначення імені класу.
2. Вимоги до файлів та каталогів при використанні пакетів:

* Файли класів повинні бути організовані в підкаталогах, які відповідають ієрархії пакетів.
* Назви файлів повинні співпадати з назвами класів.
* Шлях до кореневого каталогу пакету повинен збігатися з оголошеним package в класі.

**Висновок**

Під час лабораторної роботи, я ознайомився з процесом розробки класів та пакетів мовою Java.