Міністерство освіти і науки України

Національний університет “Львівська політехніка ”

Кафедра ЕОМ



**ЗВІТ**

до лабораторної роботи №3

З дисципліни: «Кросплатформні засоби програмування»

На тему: «Спадкування та інтерфейси»

Варіант - 7

Виконала:

ст. гр. КІ-305

Дзера А. Р.

Прийняв:

доц. каф. ЕОМ

Іванов Ю. С.

Львів – 2023

**Мета роботи:** ознайомитися з спадкуванням та інтерфейсами у мові Java.

**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.

Варіант завдання – ноутбук

**Код програми**

**Файл Computer.java**

package KI305.Dzera.Lab3;  
  
import java.io.BufferedWriter;  
import java.io.File;  
import java.io.FileWriter;  
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.List;  
  
*/\*\*  
 \* Абстрактний клас `Computer` є базовим класом для представлення комп'ютера з різними компонентами  
 \** ***@author*** *Amina  
 \** ***@version*** *1.0  
 \*/*public class Computer{  
 private final CPU cpu;  
 private final RAM ram;  
 private final HardDrive hardDrive;  
 private Monitor monitor;  
 private final Mouse mouse;  
 private final Keyboard keyboard;  
 private List<Software> installedSoftware; // Додати змінну для зберігання програмного забезпечення  
  
 private BufferedWriter logFile;  
  
  
 */\*\*  
 \* Конструктор класу Computer без параметрів  
 \*  
 \** ***@throws*** *IOException виняток, який може виникнути при створенні лог-файлу  
 \*/* public Computer() throws IOException {  
 this.cpu = new CPU("Intel Core i5");  
 this.ram = new RAM(8);  
 this.hardDrive = new HardDrive(256);  
 this.monitor = new Monitor("1920x1080");  
 this.mouse = new Mouse("Logitech");  
 this.keyboard = new Keyboard("QWERTY");  
 this.installedSoftware = new ArrayList<>(); // Ініціалізувати список програмного забезпечення  
  
 try {  
 logFile = new BufferedWriter(new FileWriter("log.txt"));  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 */\*\*  
 \* Конструктор класу Computer з параметрами. Створює комп'ютер і ініціалізує його компоненти заданими значеннями  
 \*  
 \** ***@param*** *cpu об'єкт процесора  
 \** ***@param*** *ram об'єкт оперативної пам'яті  
 \** ***@param*** *hardDrive об'єкт жорсткого диска  
 \** ***@param*** *monitor об'єкт монітора  
 \** ***@param*** *mouse об'єкт миші  
 \** ***@param*** *keyboard об'єкт клавіатури  
 \** ***@throws*** *IOException виняток, який може виникнути при створенні файлу логу  
 \*/* public Computer(CPU cpu, RAM ram, HardDrive hardDrive, Monitor monitor, Mouse mouse, Keyboard keyboard) throws IOException {  
 this.cpu = cpu;  
 this.ram = ram;  
 this.hardDrive = hardDrive;  
 this.monitor = monitor;  
 this.mouse = mouse;  
 this.keyboard = keyboard;  
 installedSoftware = new ArrayList<>();  
  
 logFile = new BufferedWriter(new FileWriter("computer\_log.txt"));  
 }  
  
 */\*\*  
 \* Метод для увімкнення комп'ютера  
 \* Цей метод вмикає комп'ютер та виводить повідомлення про його ввімкнення  
 \*  
 \** ***@throws*** *IOException виняток, який може виникнути при записі в файл логу  
 \*/* public void turnOn() throws IOException {  
 System.*out*.println("The computer is on.");  
 writeToLog("The computer is on.");  
 }  
  
 */\*\*  
 \* Метод для вимкнення комп'ютера  
 \* Цей метод вимикає комп'ютер та виводить повідомлення про його вимкнення  
 \*  
 \** ***@throws*** *IOException виняток, який може виникнути при записі в файл логу  
 \*/* public void turnOff() throws IOException {  
 System.*out*.println("The computer is turned off.");  
 writeToLog("The computer is turned off.");  
 }  
  
 */\*\*  
 \* Метод для виконання обчислень на комп'ютері  
 \* Цей метод виводить повідомлення про початок обчислень на комп'ютері  
 \*  
 \** ***@throws*** *IOException виняток, який може виникнути при записі в файл логу  
 \*/* public void performCalculations() throws IOException {  
 System.*out*.println("Calculations are being performed...");  
 writeToLog("Calculations are being performed...");  
 }  
  
 */\*\*  
 \* Метод для зберігання інформації на жорсткому диску  
 \*  
 \** ***@param*** *data дані, які потрібно зберегти  
 \** ***@throws*** *IOException виняток, який може виникнути при записі в файл логу  
 \*/* public void storeData(String data) throws IOException {  
 hardDrive.saveData(data);  
 System.*out*.println("The data is saved on the hard disk.");  
 writeToLog("The data is saved on the hard disk.");  
 }  
  
 */\*\*  
 \* Метод для встановлення нового монітора для комп'ютера  
 \*  
 \** ***@param*** *monitor новий об'єкт монітора, який слід встановити  
 \** ***@throws*** *IOException виняток, який може виникнути при записі в файл логу  
 \*/* public void setMonitor(Monitor monitor) throws IOException {  
 this.monitor = monitor;  
 monitor.setMonitor();  
 System.*out*.println("Monitor changed to " + monitor.getResolution());  
 }  
  
 public void setBrightness(int brightness, Monitor monitor) {  
 monitor.setBrightness(brightness);  
 }  
  
  
 */\*\*  
 \* Метод для встановлення програмного забезпечення на комп'ютер  
 \*  
 \** ***@param*** *software об'єкт програмного забезпечення, яке слід встановити  
 \** ***@throws*** *IOException виняток, який може виникнути при записі в файл логу  
 \*/* public void installSoftware(Software software) throws IOException {  
 installedSoftware.add(software);  
 System.*out*.println("Installed software: " + software.getName());  
 writeToLog("Installed software: " + software.getName());  
 }  
  
 */\*\*  
 \* Метод для відображення списку встановленого програмного забезпечення на комп'ютері  
 \*  
 \** ***@throws*** *IOException виняток, який може виникнути при записі в файл логу  
 \*/* public void showInstalledSoftware() throws IOException {  
 System.*out*.println("Showing all installed software: ");  
 for (Software software : installedSoftware) {  
 System.*out*.println(software.getName());  
 writeToLog(software.getName());  
 }  
 }  
  
 */\*\*  
 \* Метод для виведення на екран інформації про доступну оперативну пам'ять комп'ютера  
 \*  
 \** ***@throws*** *IOException виняток, який може виникнути при записі в файл логу  
 \*/* public void checkAvailableMemory() throws IOException {  
 System.*out*.println("Available RAM: " + ram.getAvailableMemory() + " ГБ");  
 writeToLog("Available RAM: " + ram.getAvailableMemory() + " ГБ");  
 }  
  
 */\*\*  
 \* Метод для запуску програмного додатку на комп'ютері  
 \*  
 \** ***@param*** *appName назва програмного додатку, який потрібно запустити  
 \** ***@throws*** *IOException виняток, який може виникнути при записі в файл логу  
 \*/* public void launchApplication(String appName) throws IOException {  
 System.*out*.println("The program starts: " + appName);  
 writeToLog("The program starts: " + appName);  
 }  
  
 */\*\*  
 \* Метод для перевірки стану жорсткого диска комп'ютера  
 \*  
 \** ***@return*** *`true`, якщо стан жорсткого диска в порядку, інакше `false`  
 \*/* public boolean isHardDriveHealthy() {  
 // Логіка перевірки стану жорсткого диска  
 return true; // Повертаємо true, якщо все в порядку  
 }  
  
 */\*\*  
 \* Перевіряє стан жорсткого диска і виводить відповідне повідомлення  
 \*  
 \** ***@throws*** *IOException виняток, який може виникнути при записі в файл логу  
 \*/* public void checkHardDriveStatus() throws IOException {  
 boolean isHealthy = isHardDriveHealthy();  
 if (isHealthy) {  
 System.*out*.println("Hard drive is in good condition.");  
 writeToLog("Hard drive is in good condition.");  
 } else {  
 System.*out*.println("Hard drive requires maintenance.");  
 writeToLog("Hard drive requires maintenance.");  
 }  
 }  
  
 */\*\*  
 \* Метод для зміни налаштувань монітора комп'ютера, таких як яскравість та контраст  
 \*  
 \** ***@param*** *brightness яскравість монітора (значення від 0 до 100)  
 \** ***@param*** *contrast контраст монітора (значення від 0 до 100)  
 \** ***@throws*** *IOException виняток, який може виникнути при записі в файл логу  
 \*/* public void changeMonitorSettings(int brightness, int contrast) throws IOException {  
 monitor.setBrightness(brightness);  
 //monitor.setContrast(contrast);  
 System.*out*.println("The monitor settings have been changed.");  
 writeToLog("The monitor settings have been changed.");  
 }  
  
 */\*\*  
 \* Метод для натискання лівої кнопки миші комп'ютера  
 \*  
 \** ***@throws*** *IOException виняток, який може виникнути при записі в файл логу  
 \*/* public void clickMouseLeftButton() throws IOException {  
 mouse.clickLeftButton();  
 writeToLog("Left mouse button clicked.");  
 }  
  
 */\*\*  
 \* Метод для натискання правої кнопки миші комп'ютера  
 \*  
 \** ***@throws*** *IOException виняток, який може виникнути при записі в файл логу  
 \*/* public void clickMouseRightButton() throws IOException {  
 mouse.clickRightButton();  
 writeToLog("Right mouse button clicked.");  
 }  
  
 */\*\*  
 \* Метод для прокручування миші вгору  
 \*  
 \** ***@throws*** *IOException виняток, який може виникнути при записі в файл логу  
 \*/* public void scrollMouseUp() throws IOException {  
 mouse.scrollUp();  
 writeToLog("Mouse scrolled up.");  
 }  
  
 */\*\*  
 \* Метод для прокручування миші вниз  
 \*  
 \** ***@throws*** *IOException виняток, який може виникнути при записі в файл логу  
 \*/* public void scrollMouseDown() throws IOException {  
 mouse.scrollDown();  
 writeToLog("Mouse scrolled down.");  
 }  
  
 */\*\*  
 \* Метод для натискання клавіші на клавіатурі  
 \*  
 \** ***@param*** *key символ, який представляє натиснуту клавішу  
 \*/* public void pressKeyOnKeyboard(char key) {  
 keyboard.pressKey(key);  
  
 }  
  
 */\*\*  
 \* Метод для введення тексту на клавіатурі  
 \*  
 \** ***@param*** *text рядок тексту, який буде введено на клавіатурі  
 \*/* public void typeTextOnKeyboard(String text) {  
 keyboard.typeText(text);  
  
 }  
  
 */\*\*  
 \* Метод для запису повідомлення в файл логу та додавання перехіду на новий рядок  
 \*  
 \** ***@param*** *message рядок повідомлення, який буде записаний в файл логу  
 \** ***@throws*** *IOException виняток, який може виникнути при записі в файл логу  
 \*/* public void writeToLog(String message) throws IOException{  
 logFile.write(message);  
 logFile.newLine(); //додати перехід на новий рядок  
 logFile.flush();  
 }  
  
 */\*\*  
 \* Метод для закриття файлу логу  
 \*  
 \** ***@throws*** *IOException виняток, який може виникнути при закритті файлу логу  
 \*/* public void closeLogFile() throws IOException{  
 logFile.close();  
 }  
  
 */\*\*  
 \* Метод для вимкнення комп'ютера та закриття файлу логу  
 \*  
 \** ***@throws*** *IOException виняток, який може виникнути при закритті файлу логу  
 \*/* public void shutdown() throws IOException{  
 writeToLog("Computer turned off.");  
 closeLogFile();  
 }  
}  
  
*/\*\*  
 \* Клас, що представляє центральний процесор (CPU) комп'ютера  
 \*/*class CPU {  
 private final String model;  
  
  
 */\*\*  
 \* Конструктор класу CPU ініціалізує модель процесора  
 \*  
 \** ***@param*** *model модель процесора  
 \*/* public CPU(String model) {  
 this.model = model;  
 }  
  
 */\*\*  
 \* Цей метод отримує модель процесора  
 \*  
 \** ***@return*** *модель процесора  
 \*/* public String getModel() {  
 return model;  
 }  
  
}  
  
*/\*\*  
 \* Клас, що представляє оперативну пам'ять комп'ютера (RAM)  
 \*/*class RAM {  
 private final int capacity; // oбсяг оперативної пам'яті в ГБ  
  
 */\*\*  
 \* Конструктор класу RAM ініціалізує обсяг оперативної пам'яті  
 \*  
 \** ***@param*** *capacity обсяг оперативної пам'яті в ГБ  
 \*/* public RAM(int capacity) {  
 this.capacity = capacity;  
 }  
  
 */\*\*  
 \* Отримує доступний обсяг оперативної пам'яті  
 \*  
 \** ***@return*** *обсяг доступної оперативної пам'яті в ГБ  
 \*/* public int getAvailableMemory() {  
 return capacity;  
 }  
}  
  
*/\*\*  
 \* Клас, що представляє жорсткий диск комп'ютера  
 \*/*class HardDrive {  
 private final int storageCapacity; // Обсяг жорсткого диска в ГБ  
  
 */\*\*  
 \* Конструктор класу HardDrive ініціалізує обсяг жорсткого диска  
 \*  
 \** ***@param*** *storageCapacity обсяг жорсткого диска в ГБ  
 \*/* public HardDrive(int storageCapacity) {  
 this.storageCapacity = storageCapacity;  
 }  
  
 */\*\*  
 \* Отримує обсяг жорсткого диска  
 \*  
 \** ***@return*** *обсяг жорсткого диска в ГБ  
 \*/* public int getStorageCapacity() {  
 return storageCapacity;  
 }  
  
 */\*\*  
 \* Зберігає дані на жорсткому диску  
 \*  
 \** ***@param*** *data дані, які потрібно зберегти  
 \*/* public void saveData(String data) {  
 try{  
 File file = new File("data.txt");  
 FileWriter fileWriter = new FileWriter(file);  
 BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);  
 bufferedWriter.write(data);  
 bufferedWriter.close();  
 System.*out*.println("Data has been saved to the hard drive.");  
 } catch (IOException e){  
 e.printStackTrace();  
 System.*err*.println("Error occurred while saving data to the hard drive.");  
 }  
 }  
}  
  
  
*/\*\*  
 \* Клас, що представляє монітор комп'ютера  
 \*/*class Monitor {  
 private final String resolution; //роздільна здатність монітора  
 private int brightness;  
 private int monitorSetMonitorCalls = 0; //лічильник викликів методу setMonitor()  
 private int monitorSetBrightnessCalls = 0; //лічильник викликів методу setBrightness()  
  
  
 */\*\*  
 \* Конструктор класу Monitor. Ініціалізує об'єкт монітора з заданою роздільною здатністю  
 \* та максимальною яскравістю екрану (100 за замовчуванням)  
 \*  
 \** ***@param*** *resolution роздільна здатність монітора (наприклад, "1920x1080")  
 \*/* public Monitor(String resolution) {  
 this.resolution = resolution;  
 this.brightness = 100;  
 }  
  
 */\*\*  
 \* Метод, що повертає роздільну здатність монітора  
 \*  
 \** ***@return*** *роздільна здатність монітора у форматі "ширинаxвисота"  
 \*/* public String getResolution() {  
 return resolution;  
 }  
  
 */\*\*  
 \* Метод, що встановлює яскравість екрану монітора на вказане значення  
 \*  
 \** ***@param*** *brightness нове значення яскравості екрану (від 0 до 100)  
 \*/* public void setBrightness(int brightness) {  
 this.brightness = brightness;  
 System.*out*.println("Monitor brightness set to " + brightness);  
 monitorSetBrightnessCalls++; //збільшує лічильник викликів методу setBrightness()  
 }  
  
 */\*\*  
 \* Метод, що позначає, що монітор був налаштований  
 \*/* public void setMonitor() {  
 System.*out*.println("Monitor set");  
 monitorSetMonitorCalls++; //збільшує лічильник викликів методу setMonitor()  
 }  
  
 */\*\*  
 \* Метод, що повертає кількість викликів методу setMonitor()  
 \*  
 \** ***@return*** *кількість викликів методу setMonitor()  
 \*/* public int getMonitorSetMonitorCalls() {  
 return monitorSetMonitorCalls;  
 }  
  
 */\*\*  
 \* Метод, що повертає кількість викликів методу setBrightness()  
 \*  
 \** ***@return*** *кількість викликів методу setBrightness()  
 \*/* public int getMonitorSetBrightnessCalls() {  
 return monitorSetBrightnessCalls;  
 }  
}

*/\*\*  
 \* Клас, що представляє комп'ютерну мишу  
 \*/*class Mouse {  
  
 public final String brand;  
  
 */\*\*  
 \* Конструктор класу Mouse ініціалізує бренд миші та виводить інформацію про бренд  
 \*  
 \** ***@param*** *brand бренд миші  
 \*/* public Mouse(String brand) {  
 this.brand = brand;  
 }  
  
 */\*\*  
 \* Метод виконує клік лівою кнопкою миші  
 \*/* public void clickLeftButton() {  
 System.*out*.println("Left mouse button clicked.");  
 }  
  
 */\*\*  
 \* Метод виконує клік правою кнопкою миші  
 \*/* public void clickRightButton() {  
 System.*out*.println("Right mouse button clicked.");  
 }  
  
 */\*\*  
 \* Метод прокручує мишу вгору  
 \*/* public void scrollUp() {  
 System.*out*.println("Mouse scrolled up.");  
 }  
  
 */\*\*  
 \* Метод прокручує мишу вниз  
 \*/* public void scrollDown() {  
 System.*out*.println("Mouse scrolled down.");  
 }  
  
}  
  
*/\*\*  
 \* Клас, що представляє комп'ютерну клавіатуру з певним розкладом  
 \*/*class Keyboard {  
 private final String layout;  
  
 */\*\*  
 \* Конструктор класу Keyboard ініціалізує розклад клавіатури  
 \*  
 \** ***@param*** *layout розклад клавіатури  
 \*/* public Keyboard(String layout) {  
 this.layout = layout;  
 }  
  
 */\*\*  
 \* Метод симулює натискання клавіші на клавіатурі та виводить повідомлення  
 \*  
 \** ***@param*** *key натискана клавіша  
 \*/* public void pressKey(char key) {  
 System.*out*.println("Key " + key + " pressed on " + layout + " keyboard.");  
 }  
  
 */\*\*  
 \* Метод симулює введення тексту на клавіатурі та виводить повідомлення  
 \*  
 \** ***@param*** *text введений текст  
 \*/* public void typeText(String text) {  
 System.*out*.println("Typing text: " + text + " on " + layout + " keyboard.");  
 }  
}  
  
  
*/\*\*  
 \* Клас, що представляє програмне забезпечення на комп'ютері  
 \*/*class Software {  
 private final String name;  
  
 */\*\*  
 \* Конструктор класу Software ініціалізує назву програмного забезпечення  
 \*  
 \** ***@param*** *name назва програмного забезпечення  
 \*/* public Software(String name) {  
 this.name = name;  
 }  
  
 */\*\*  
 \* Метод отримує назву програмного забезпечення  
 \*  
 \** ***@return*** *назва програмного забезпечення  
 \*/* public String getName() {  
 return name;  
 }  
}

**Файл Laptop.java**

package KI305.Dzera.Lab3;  
  
import java.io.IOException;  
  
*/\*\*  
 \* Клас, що представляє ноутбук.  
 \* Ноутбук є підкласом класу Computer та імплементує інтерфейс LaptopInterface  
 \*  
 \** ***@author*** *Amina  
 \** ***@version*** *1.0  
 \*/*public class Laptop extends Computer implements LaptopInterface{  
 private Battery battery;  
  
 */\*\*  
 \* Конструктор для створення об'єкта ноутбука  
 \*  
 \** ***@throws*** *IOException виняток, який може виникнути при створенні об'єкта  
 \*/* public Laptop() throws IOException{  
 super();  
 this.battery = new Battery("Lithium-io");  
  
 }  
  
 */\*\*  
 \* Метод для включення ноутбука  
 \*  
 \** ***@throws*** *IOException виняток, який може виникнути при включенні ноутбука  
 \*/* @Override  
 public void turnOn() throws IOException{  
 super.turnOn();  
 System.*out*.println("Laptop is booting...");  
 }  
  
 */\*\*  
 \* Метод для вимикання ноутбука  
 \*  
 \** ***@throws*** *IOException виняток, який може виникнути при вимиканні ноутбука  
 \*/* @Override  
 public void turnOff() throws IOException {  
 super.turnOff();  
 System.*out*.println("Laptop is shutting down...");  
  
 }  
  
 */\*\*  
 \* Метод для використання батареї ноутбука.  
 \* Перевіряє рівень заряду батареї та споживає енергію в залежності від умов  
 \*/* @Override  
 public void useBattery() {  
 int currentBatteryLevel = battery.getCurrentLevel();  
 boolean isCharging = battery.isCharging();  
  
 if (!isCharging) {  
 //якщо ноутбук не заряджається, то споживайте енергію  
 int energyConsumption = calculateEnergyConsumption();  
 currentBatteryLevel -= energyConsumption;  
  
 System.*out*.println("Battery Level: " + currentBatteryLevel + "%");  
  
 // перевірка на те чи не стало заряду дуже низьким  
 if (currentBatteryLevel < 10) {  
 System.*out*.println("Low battery! Save your work and charge the laptop.");  
 }  
 }  
 }  
  
 */\*\*  
 \* Метод для зарядження батареї ноутбука  
 \*/* public void chargeBattery() {  
 battery.chargeBattery();  
 }  
  
 */\*\*  
 \* Метод для розрядки батареї ноутбука  
 \*  
 \** ***@param*** *energyConsumed кількість спожитої енергії  
 \*/* public void dischargeBattery(int energyConsumed) {  
 battery.dischargeBattery(energyConsumed);  
 }  
  
 */\*\*  
 \* Розрахунок споживаної енергії ноутбуком  
 \*  
 \** ***@return*** *кількість спожитої енергії  
 \*/* private int calculateEnergyConsumption() {  
 // споживання енергії в залежності від виконуваних завдань і активності (повертаємо припустиме значення споживаної енергії)  
 return 5;  
 }  
  
  
 */\*\*  
 \* Клас, що представляє батарею ноутбука  
 \*/* public class Battery {  
 private String type;  
 private int currentLevel; //поточний рівень заряду  
 private boolean charging; //заряджається комп чи ні  
  
 */\*\*  
 \* Конструктор для створення об'єкта батареї  
 \*  
 \** ***@param*** *type тип батареї  
 \*/* public Battery(String type) {  
 this.type = type;  
 this.currentLevel = 100;  
 this.charging = false; //на початку батарея не заряджається  
 }  
  
 */\*\*  
 \* Отримання типу батареї  
 \*  
 \** ***@return*** *тип батареї  
 \*/* public String getType() {  
 return type;  
 }  
  
 */\*\*  
 \* Отримання поточного рівня заряду батареї  
 \*  
 \** ***@return*** *поточний рівень заряду батареї в процентах  
 \*/* public int getCurrentLevel() {  
 return currentLevel;  
 }  
  
 */\*\*  
 \* Перевірка, чи батарея заряджається  
 \*  
 \** ***@return*** *`true` якщо батарея заряджається, `false` в іншому випадку  
 \*/* public boolean isCharging() {  
 return charging;  
 }  
  
 */\*\*  
 \* Метод для зарядження батареї.  
 \* Перевіряє, чи батарея не заряджається вже і чи не досягла вона максимального рівня заряду  
 \*/* public void chargeBattery() {  
 if (!charging && currentLevel < 100) {  
 charging = true;  
 System.*out*.println("Battery is now charging.");  
 }  
 }  
  
 */\*\*  
 \* Метод для розрядки батареї ноутбука.  
 \* Перевіряє, чи батарея не заряджається, і віднімає спожиту енергію від рівня заряду  
 \*  
 \** ***@param*** *energyConsumed кількість спожитої енергії  
 \*/* public void dischargeBattery(int energyConsumed) {  
 if (!charging) {  
 currentLevel -= energyConsumed;  
 if (currentLevel < 0) {  
 currentLevel = 0;  
 System.*out*.println("Battery is now fully discharged.");  
 }  
 }  
 }  
 }  
}

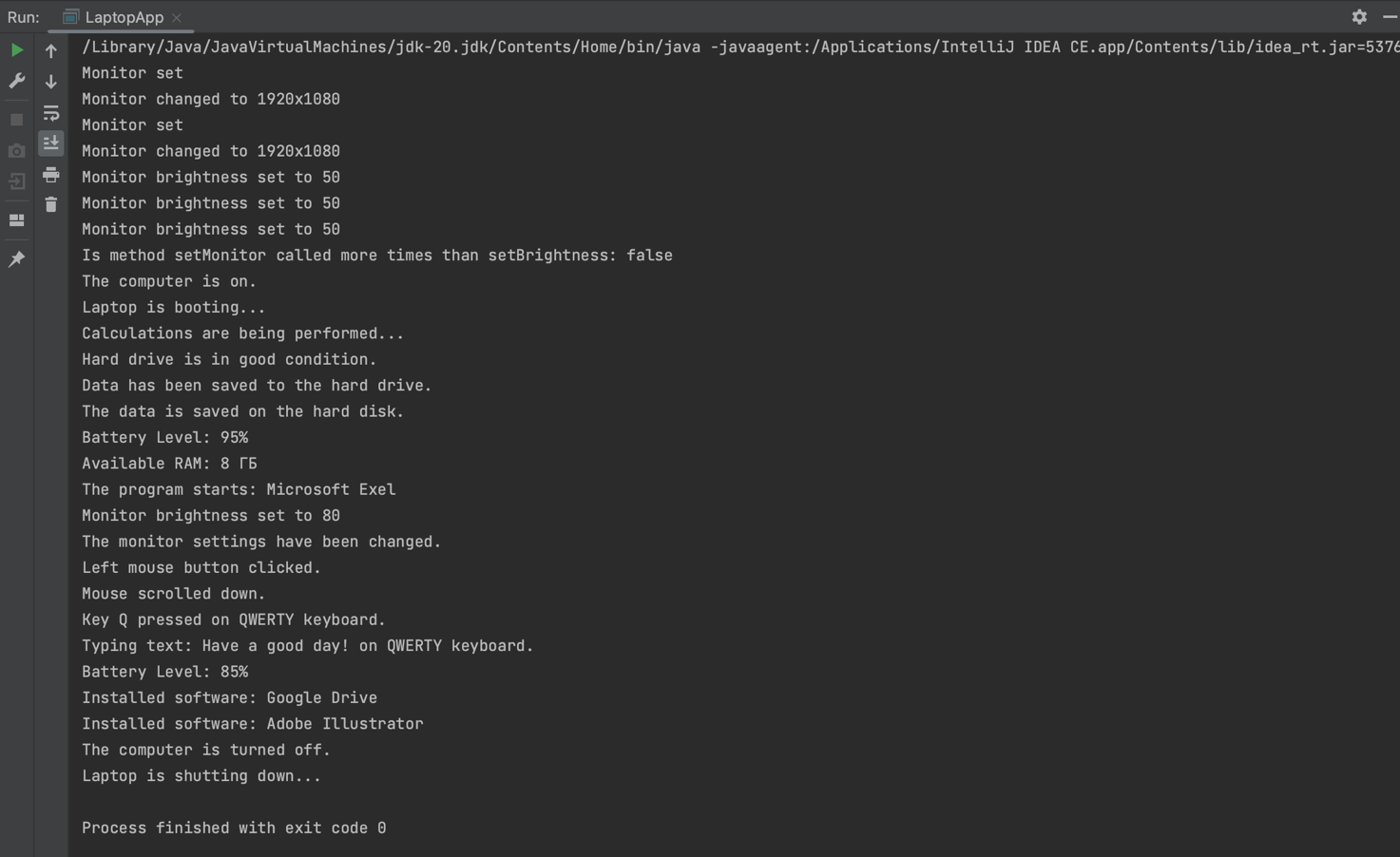
**Файл LaptopApp.java**

package KI305.Dzera.Lab3;  
  
import java.io.IOException;  
  
*/\*\*  
 \* Клас `LaptopApp` є демонстраційним додатком, який ілюструє використання функціональності ноутбука.  
 \*  
 \* Даний додаток створений для тестування та виконання різних операцій на ноутбуці  
 \*/*public class LaptopApp {  
 public static void main(String[] args) throws IOException {  
 Computer computer = new Computer();  
 Monitor monitor = new Monitor("1920x1080");  
  
 computer.setMonitor(monitor);  
 computer.setMonitor(monitor);  
 computer.setBrightness(50, monitor);  
 computer.setBrightness(50, monitor);  
 computer.setBrightness(50, monitor);  
  
  
 boolean monitorCalledMoreThanBrightness = monitor.getMonitorSetMonitorCalls() > monitor.getMonitorSetBrightnessCalls();  
 System.*out*.println(monitorCalledMoreThanBrightness);  
  
 try{  
 Laptop laptop = new Laptop();  
  
 laptop.turnOn();  
 laptop.performCalculations();  
 laptop.isHardDriveHealthy();  
 laptop.checkHardDriveStatus();  
 laptop.storeData("Some important data");  
 laptop.useBattery();  
 laptop.chargeBattery();  
 laptop.checkAvailableMemory();  
 laptop.launchApplication("Microsoft Exel");  
 laptop.changeMonitorSettings(80,70);  
 laptop.clickMouseLeftButton();  
 laptop.scrollMouseDown();  
 laptop.pressKeyOnKeyboard('Q');  
 laptop.typeTextOnKeyboard("Have a good day!");  
 laptop.dischargeBattery(10);  
 laptop.useBattery();  
  
 Software software1 = new Software("Google Drive");  
 Software software2 = new Software("Adobe Illustrator");  
 laptop.installSoftware(software1);  
 laptop.installSoftware(software2);  
  
 laptop.turnOff();  
 laptop.closeLogFile();  
  
  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 }  
}

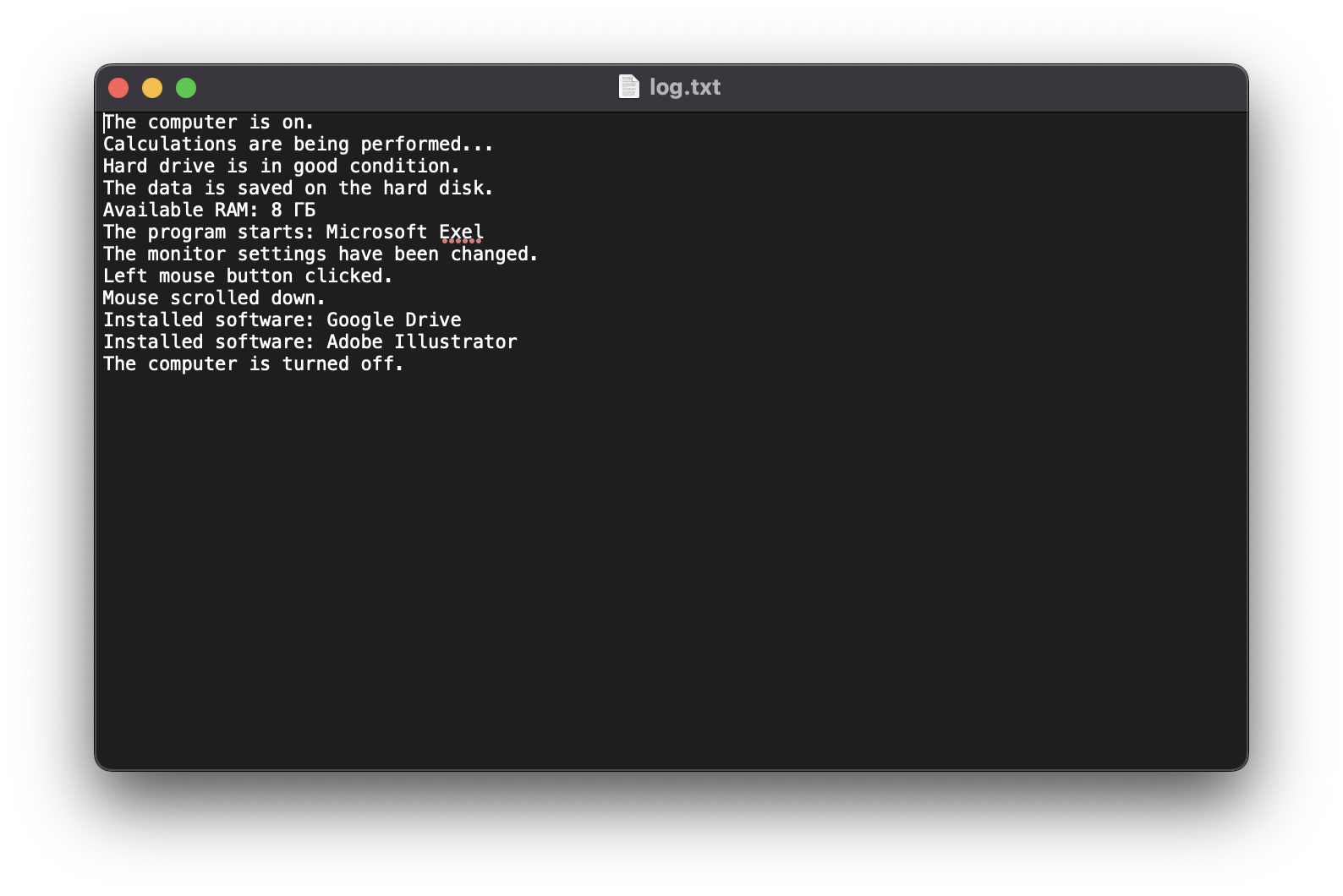
**Файл LaptopInterface.java**

package KI305.Dzera.Lab3;  
  
*/\*\*  
 \* Інтерфейс `LaptopInterface` представляє специфікацію функціональності, яку повинні реалізовувати класи, що представляють ноутбуки або інші пристрої з батареєю  
 \*/*public interface LaptopInterface {  
 */\*\*  
 \* Метод `useBattery()` визначає функціональність використання батареї.  
 \* Класи, що імплементують цей інтерфейс, повинні реалізувати цей метод для забезпечення конкретної реакції на використання батареї пристроєм  
 \*/* void useBattery();  
  
}

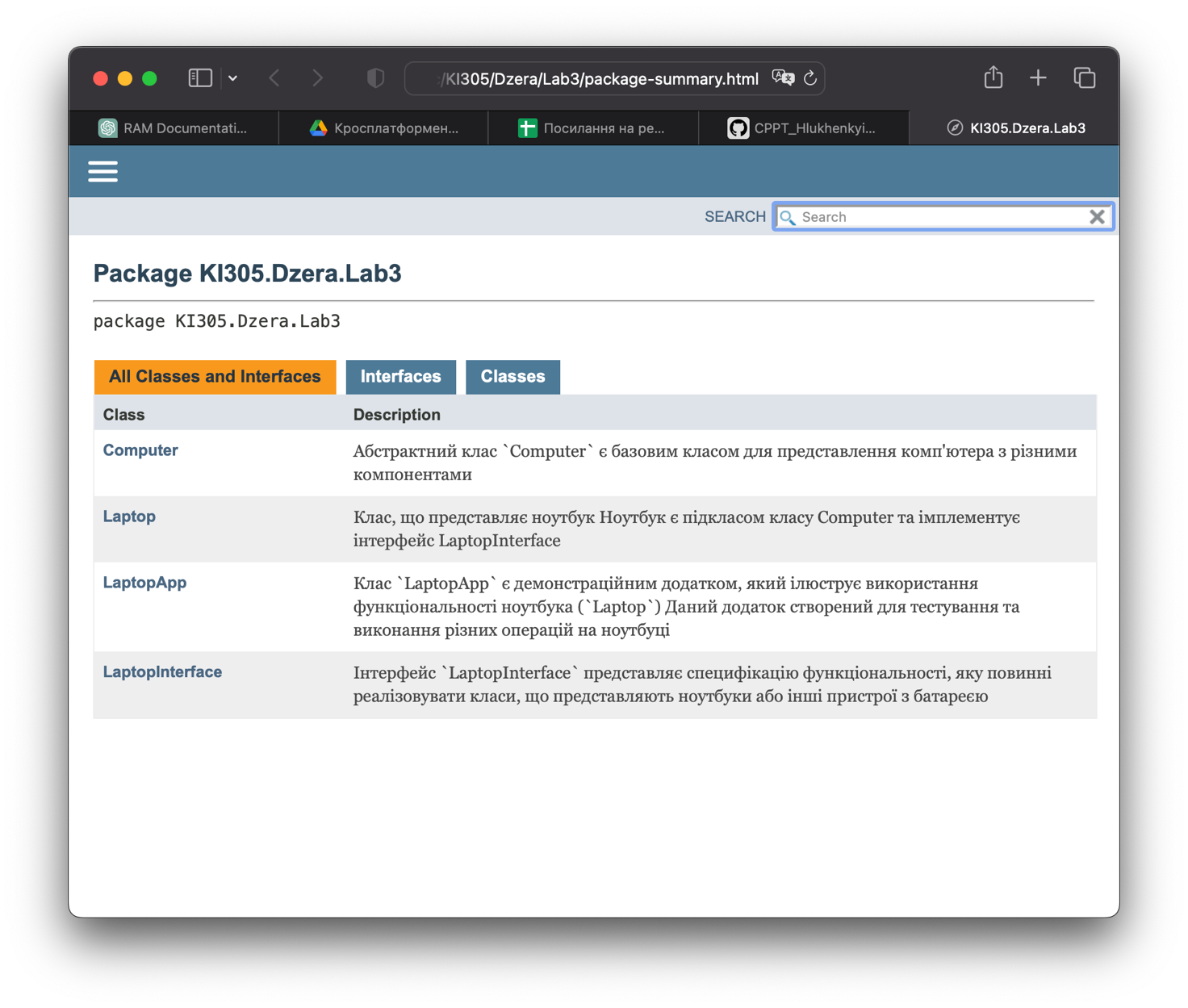
**Результати роботи програми**

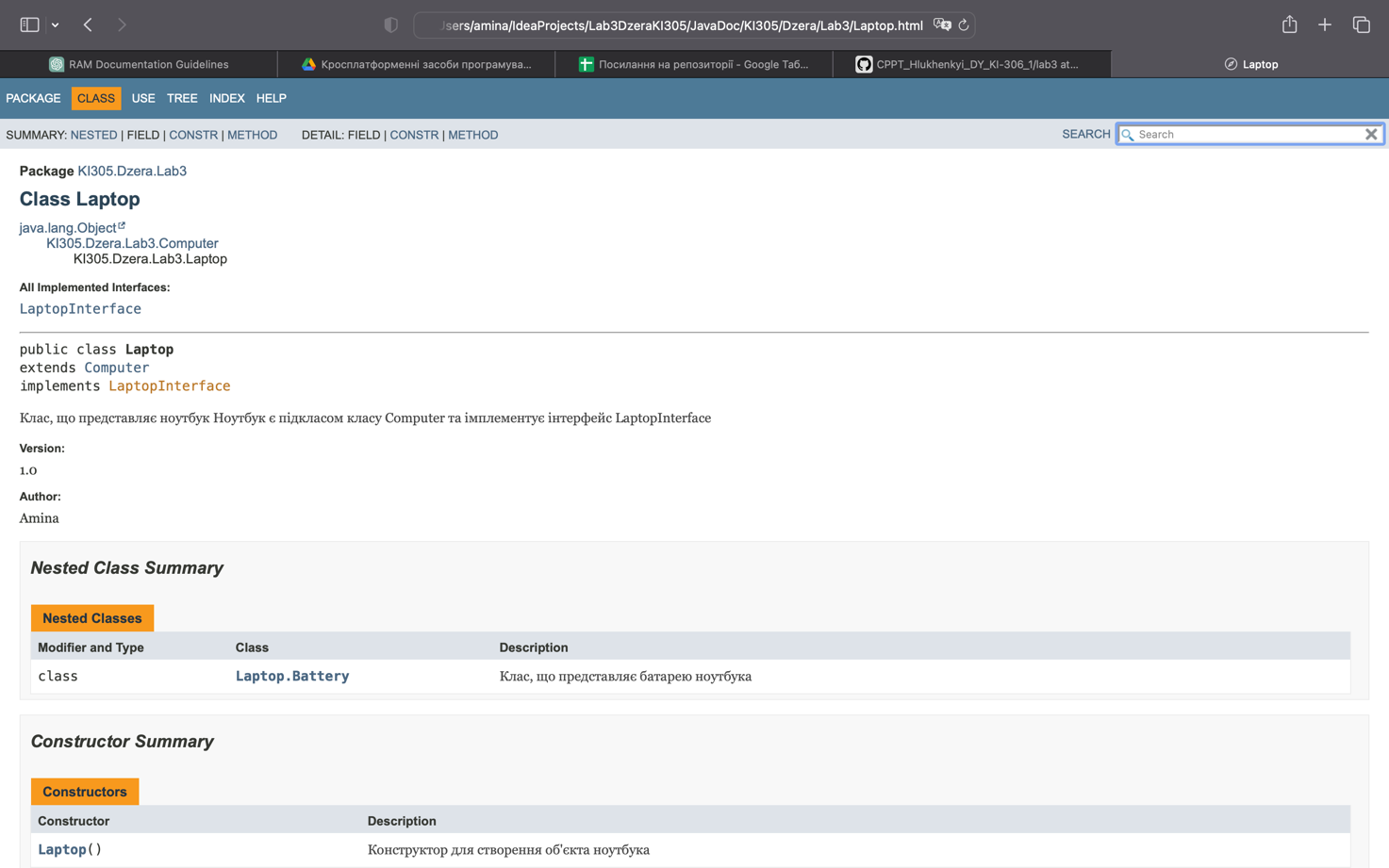


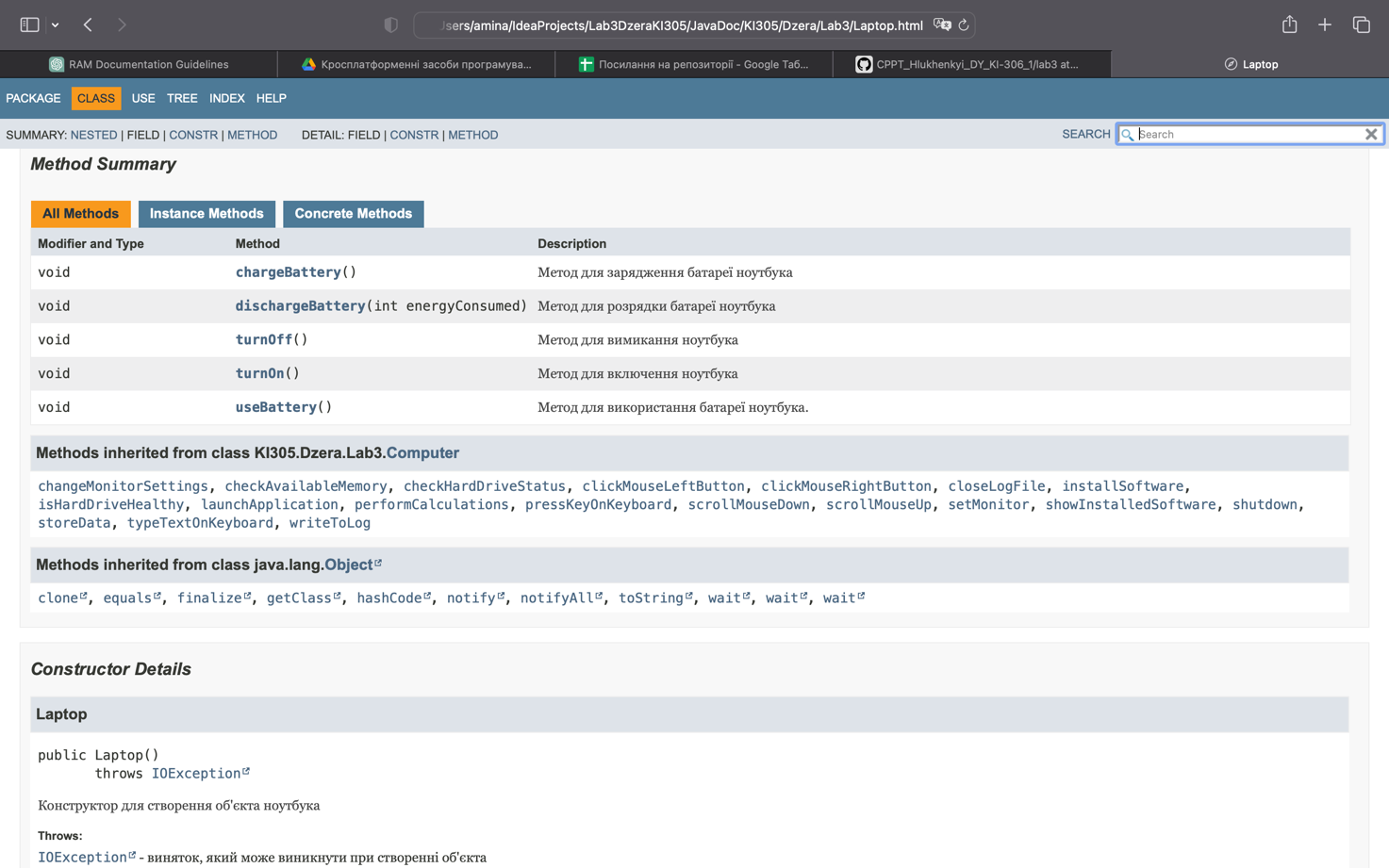
**Вміст файлу log.txt**

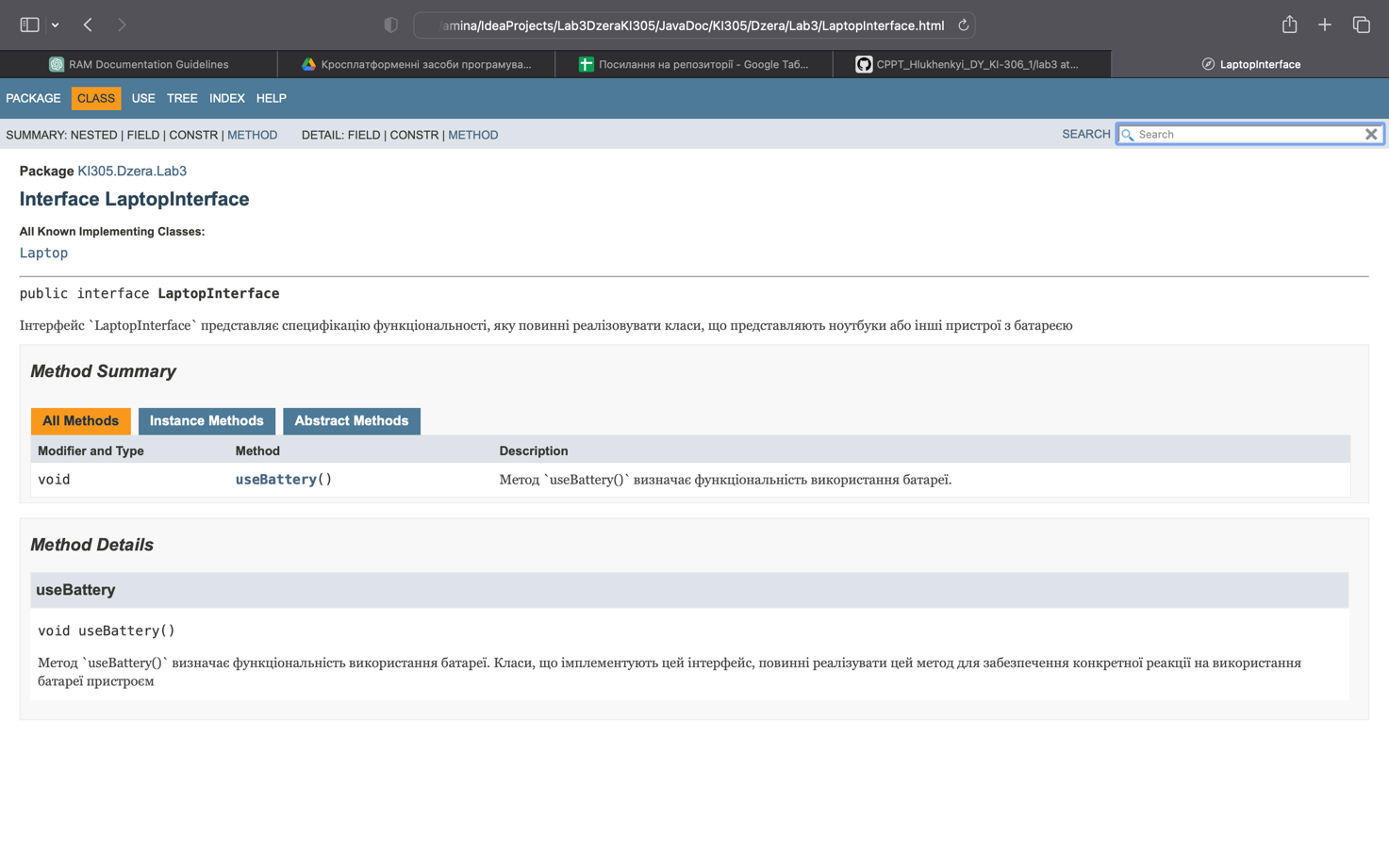


**Фрагмент згенерованої документації**









**Контрольні питання**

1. Синтаксис реалізації спадкування.

class Subclass extends Superclass {

// Тіло підкласу

}

1. Що таке суперклас та підклас?

*Суперклас* ⎯ це клас, від якого інший клас успадковує властивості та методи. Він також відомий як базовий клас або батьківський клас.

*Підклас* ⎯ це клас, який успадковує властивості та методи з іншого класу. Він також відомий як похідний клас або дочірній клас.

1. Як звернутися до членів суперкласу з підкласу?

Потрібно використати ключове слово “*super*” для звернення до членів суперкласу з підкласу.

super.methodName();

super.variableName;

1. Коли використовується статичне зв’язування при виклику методу?

Статичне зв’язування відбувається під час компіляції. Це відбувається, коли метод, який буде викликаний, визначається на основі типу посилання.

1. Як відбувається динамічне зв’язування при виклику методу?

Динамічне зв’язування відбувається під час виконання програми. Метод, який буде викликаний, визначається на основі типу об’єкта в час виконання.

1. Що таке абстрактний клас та як його реалізувати?

Абстрактний клас - це клас, який не може бути інстанційований (тобто, не можна створити об'єкт цього класу) і може містити абстрактні методи (методи без реалізації). Для оголошення абстрактного класу використовується ключове слово *abstract*. Приклад:

abstract class AbstractClass {

abstract void abstractMethod();

}

1. Для чого використовується ключове слово instanceof?

*instanceof* ⎯ це оператор, який перевіряє, чи об’єкт є екземпляром певного класу або інтерфейсу. Приклад використання:

if (object instanceof ClassName) {

// Виконати дії, якщо object є екземпляром ClassName

}

1. Як перевірити чи клас є підкласом іншого класу?

Потрібно використовувати ключове слово *instanceof* або метод *isAssignableFrom()* класу Class. Приклад використання:

if (object instanceof Superclass) {

// Виконати дії, якщо object є підкласом Superclass

}

1. Що таке інтерфейс?

*Інтерфейс* ⎯ це сховище абстрактних методів, які клас повинен реалізувати. Інтерфейси визначають, які методи повинні бути реалізовані класами, що реалізують цей інтерфейс.

1. Як оголосити та застосувати інтерфейс?

Оголошення інтерфейсу:

interface InterfaceName {

void methodName();

}

Реалізація інтерфейсу у класі:

class ClassName implements InterfaceName {

public void methodName() {

// Реалізація методу з інтерфейсу

}

}

**Висновок**

Під час виконання даної лабораторної роботи я ознайомилася та отримала навички роботи з такими концепціями як спадкування та інтерфейси у мові Java.