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

Турчин П.Р.

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.

1. Автоматично згенерувати документацію до розробленої програми.
2. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації.
3. Дати відповідь на контрольні запитання.

**Лістинг програми:**

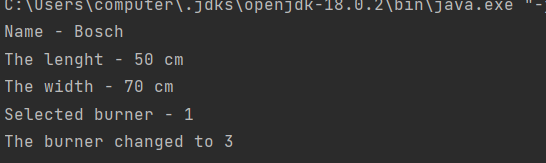
**Файл GunApp.java**

*/\*\*  
 \* lab 3 package  
 \*/*package KI34.Turchyn.Laba3;  
  
  
  
  
import static java.lang.System.*out*;  
import java.io.\*;  
*/\*\*  
 \* Cookstove Application class implements main method for Cookstove  
 class possibilities demonstration  
 \** ***@author*** *Pavlo Turchyn KI-34  
 \** ***@version*** *1.0  
 \*/*public class CookstoveApp {  
 */\*\*  
 \** ***@param*** *args param  
 \** ***@throws*** *FileNotFoundException  
 \*/* public static void main(String[] args) throws FileNotFoundException {  
  
 cookstove object = new cookstove("Great", 10, true, "Bosch", 50, 70, true, true, 1);  
  
 object.off\_onCookstove();  
 object.settings();  
 object.True\_falseOven();  
 object.True\_falseGasbag();  
  
 object.changeBurner(3);  
 object.FireStatusPlus(30);  
 object.FireStatusMinus(5);  
 object.Condition();  
 object.Condition2();  
  
 object.Extract();  
 object.dispose();  
 }  
  
  
  
}

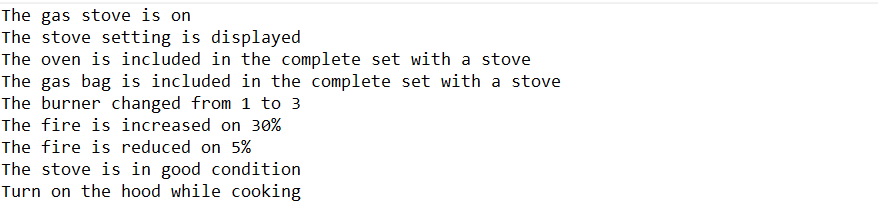
**Файл Gun.java**

*/\*\*  
 \* lab 3 package  
 \*/*package KI34.Turchyn.Laba3;  
  
import java.io.\*;  
*/\*\*  
 \* Class <code>Cookstove</code> implements cookstove  
 \* \* @author Pavlo Turchyn KI-34  
 \* \* @version 1.0  
 \*\*/*public class cookstove {  
 private String cookstovecondition;  
 private int fire;  
 private boolean status;  
 private String name;  
 private int lenght;  
 private int width;  
 private boolean gasbag;  
 private boolean oven;  
 private int burner;  
 private PrintWriter myWrite;  
  
 */\*\*  
 \* Constructor  
 \** ***@throws*** *FileNotFoundException param  
 \*/* public cookstove() throws FileNotFoundException {  
 cookstovecondition = "None";  
 fire = 0;  
 status = false;  
 name = "None";  
 lenght = 0;  
 width = 0;  
 gasbag = false;  
 oven = false;  
 burner = 0;  
 myWrite = new PrintWriter(new File("Log.txt"));  
 }  
  
 */\*\*  
 \* Constructor  
 \** ***@param*** *cookstovecondition <code>cookstovecondition</code> Condition of cookstove  
 \** ***@param*** *fire <code>fire</code> Condition of fire  
 \** ***@param*** *status <code>status</code> On/Off mode of cookstove  
 \** ***@param*** *name <code>name</code> Name of cookstove  
 \** ***@param*** *lenght <code>lenght</code> Lenght of cookstove  
 \** ***@param*** *width <code>width</code> Width of cookstove  
 \** ***@param*** *gasbag <code>gasbag</code> Availability of gasbag  
 \** ***@param*** *oven <code>oven</code> Availability of oven  
 \** ***@param*** *burner <code>burner</code> The work of switching burner  
 \** ***@throws*** *FileNotFoundException  
 \*/* public cookstove( String cookstovecondition,int fire,boolean status,String name, int lenght, int width, boolean gasbag, boolean oven, int burner) throws FileNotFoundException {  
 this.cookstovecondition=cookstovecondition;  
 this.fire = fire;  
 this.status = status;  
 this.name = name;  
 this.lenght = lenght;  
 this.width = width;  
 this.gasbag = gasbag;  
 this.oven = oven;  
 this.burner = burner;  
 myWrite = new PrintWriter(new File("Log.txt"));  
  
 }  
 */\*\*  
 \* Method returns cookstoves' condtion  
 \** ***@return*** *cookstoves' condtion  
 \*/* public String getCookstovecondition() {  
 return cookstovecondition;  
 }  
 */\*\*  
 \* Method sets the cookstoves' condtion  
 \** ***@param*** *Cookstovecondition <code>cookstovecondition</code> cookstoves' condtion  
 \*/* public void setCookstovecondition(String Cookstovecondition) {  
 this.cookstovecondition = cookstovecondition;  
 }  
 */\*\*  
 \* Method returns fire status  
 \** ***@return*** *status of fire  
 \*/* public int getFire() {  
 return fire;  
 }  
 */\*\*  
 \* Method sets the fire status  
 \** ***@param*** *fire <code>fire</code> fire status  
 \*/* public void setFire(int fire) {  
 this.fire = fire;  
 }  
 */\*\*  
 \* Method returns off/on mode of cookstove  
 \** ***@return*** *off/on mode of cookstove  
 \*/* public boolean getStatus() {  
 return status;  
 }  
 */\*\*  
 \* Method sets the off/on status of cookstove  
 \** ***@param*** *status <code>fire</code> off/on mode of cookstove  
 \*/* public void setStatus(boolean status) {  
 this.status = status;  
 }  
 */\*\*  
 \* Method returns company's name of cookstove  
 \** ***@return*** *company's name of cookstove  
 \*/* public String getName() {  
 return name;  
 }  
 */\*\*  
 \* Method sets the company's name of cookstove  
 \** ***@param*** *name <code>name</code> company's name of cookstove  
 \*/* public void setName(String name) {  
 this.name = name;  
 }  
 */\*\*  
 \* Method returns lenght of cookstove  
 \** ***@return*** *lenght of cookstove  
 \*/* public int getLenght() {  
 return lenght;  
 }  
 */\*\*  
 \* Method sets the lenght of cookstove  
 \** ***@param*** *lenght <code>lenght</code> lenght of cookstove  
 \*/* public void setLenght(int lenght) {  
 this.lenght = lenght;  
 }  
 */\*\*  
 \* Method returns width of cookstove  
 \** ***@return*** *width of cookstove  
 \*/* public int getWidth() {  
 return width;  
 }  
 */\*\*  
 \* Method sets the width of cookstove  
 \** ***@param*** *width <code>width</code> width of cookstove  
 \*/* public void setWidth(int width) {  
 this.width = width;  
 }  
 */\*\*  
 \* Method returns avaiability of gas bag  
 \** ***@return*** *avaiability of gas bag  
 \*/* public boolean getGasbag() {  
 return gasbag;  
 }  
 */\*\*  
 \* Method sets the avaiability of gas bag  
 \** ***@param*** *gasbag <code>gasbag</code> avaiability of gas bag  
 \*/* public void setGasbag(boolean gasbag) {  
 this.gasbag = gasbag;  
 }  
 */\*\*  
 \* Method returns avaiability of oven  
 \** ***@return*** *avaiability of oven  
 \*/* public boolean getOven() {  
 return oven;  
 }  
 */\*\*  
 \* Method sets the avaiability of oven  
 \** ***@param*** *oven <code>oven</code> avaiability of oven  
 \*/* public void setOven(boolean oven) {  
 this.oven = oven;  
 }  
 */\*\*  
 \* Method returns execution process with burner  
 \** ***@return*** *execution process with burner  
 \*/* public int getBurner() {  
 return burner;  
 }  
 */\*\*  
 \* Method sets the execution process with burner  
 \** ***@param*** *burner <code>burner</code> execution process with burner  
 \*/* public void setBurner(int burner) {  
 this.burner = burner;  
 }  
  
  
  
 */\*\*  
 \* Method show us the status of cookstove(off/on)  
 \*/* public void off\_onCookstove(){  
 if (status){  
 myWrite.println("The gas stove is on");  
 } else{  
 myWrite.println("The gas stove is off");  
 }  
 }  
  
 */\*\*  
 \* Method show us the settings of our cookstove  
 \*/* public void settings(){  
 System.*out*.println("Name - " + name);  
 System.*out*.println("The lenght - " + lenght + " cm");  
 System.*out*.println("The width - " + width + " cm");  
 myWrite.println("The stove setting is displayed");  
 }  
  
 */\*\*  
 \* Method demonstrate avaiablity of oven  
 \*/* public void True\_falseOven(){  
 if (oven){  
 myWrite.println("The oven is included in the complete set with a stove");  
 }else{  
 myWrite.println("The stove is not included in the complete set with a stove");  
 }  
 }  
  
  
 */\*\*  
 \* Method demonstrate avaiablity of gasbag  
 \*/* public void True\_falseGasbag(){  
 if (gasbag){  
 myWrite.println("The gas bag is included in the complete set with a stove");  
 }else{  
 myWrite.println("The stove is not included in the complete set with a stove\n");  
 }  
 }  
  
 */\*\*  
 \* Method simulates changing burner in cookstove  
 \*/* public void changeBurner(int bur){  
 System.*out*.println("Selected burner - " + burner);  
 myWrite.println("The burner changed from " + burner + " to " + bur);  
 System.*out*.println("The burner changed to " + bur);  
 }  
  
 */\*\*  
 \* Method simulates decrement fire in cookstove  
 \*/* public void FireStatusPlus(int change){  
 fire = fire + change;  
 myWrite.println("The fire is increased on " + change + "% ");  
 }  
 */\*\*  
 \* Method simulates increment fire in cookstove  
 \*/* public void FireStatusMinus(int change){  
 fire = fire - change;  
 myWrite.println("The fire is reduced on " + change + "% ");  
 }  
 */\*\*  
 \* Method show us the great condition of cookstove  
 \*/* public void Condition(){  
 if(cookstovecondition == "Great"){  
 myWrite.println("The stove is in good condition");  
 }  
 }  
  
 */\*\*  
 \* Method show us the bad condition of cookstove  
 \*/* public void Condition2(){  
 if(cookstovecondition == "Bad"){  
 myWrite.println("The stove is in bad condition, it needs to be repaired");  
 }  
 }  
  
 */\*\*  
 \* Method makes comments while cooking while the stove is on  
 \*/* public void Extract() {  
 if(status){  
 myWrite.println("Turn on the hood while cooking");  
 }  
 }  
  
 */\*\*  
 \* Method releases used recourses  
 \*/* public void dispose()  
 {  
 myWrite.flush();  
 myWrite.close();  
 }  
}

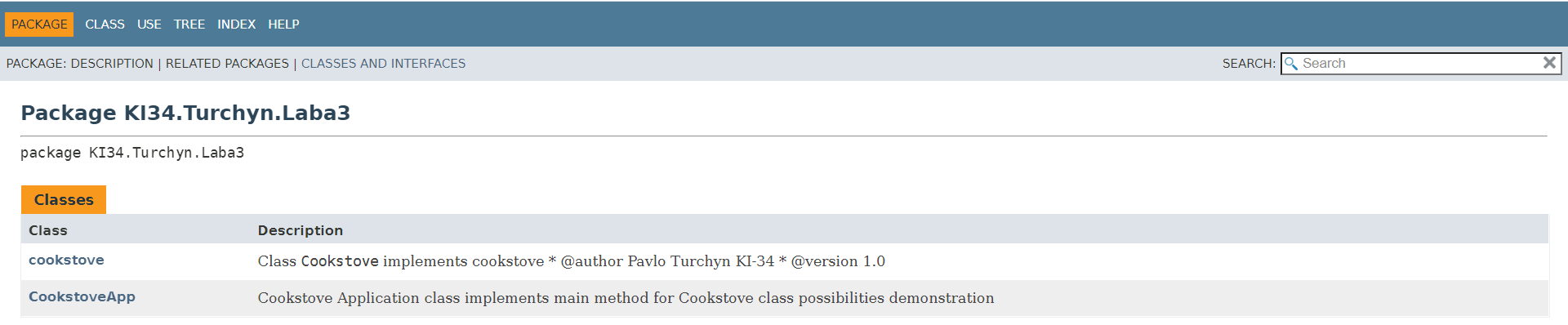
**Результат виконання програми:**



*Протокол діяльності в консолі*

**

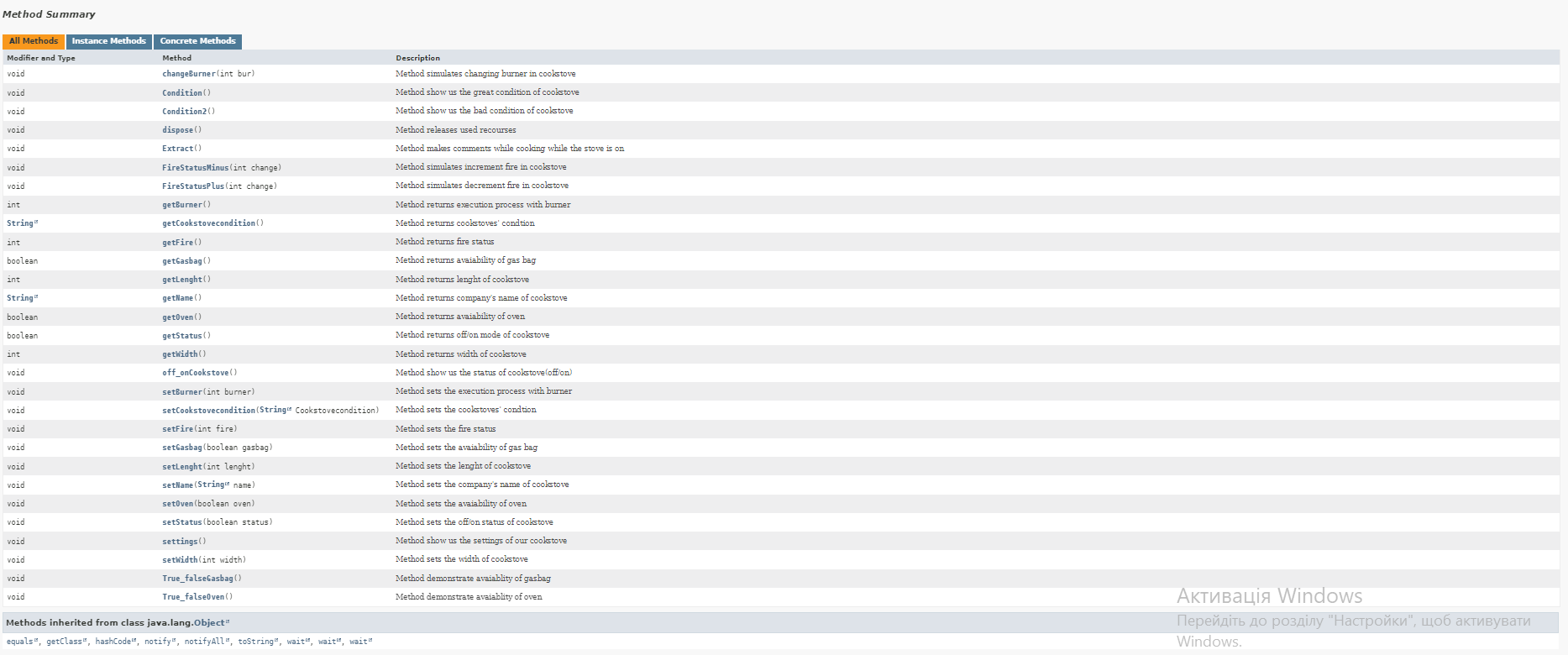
*Протокол діяльності у файлі*

**

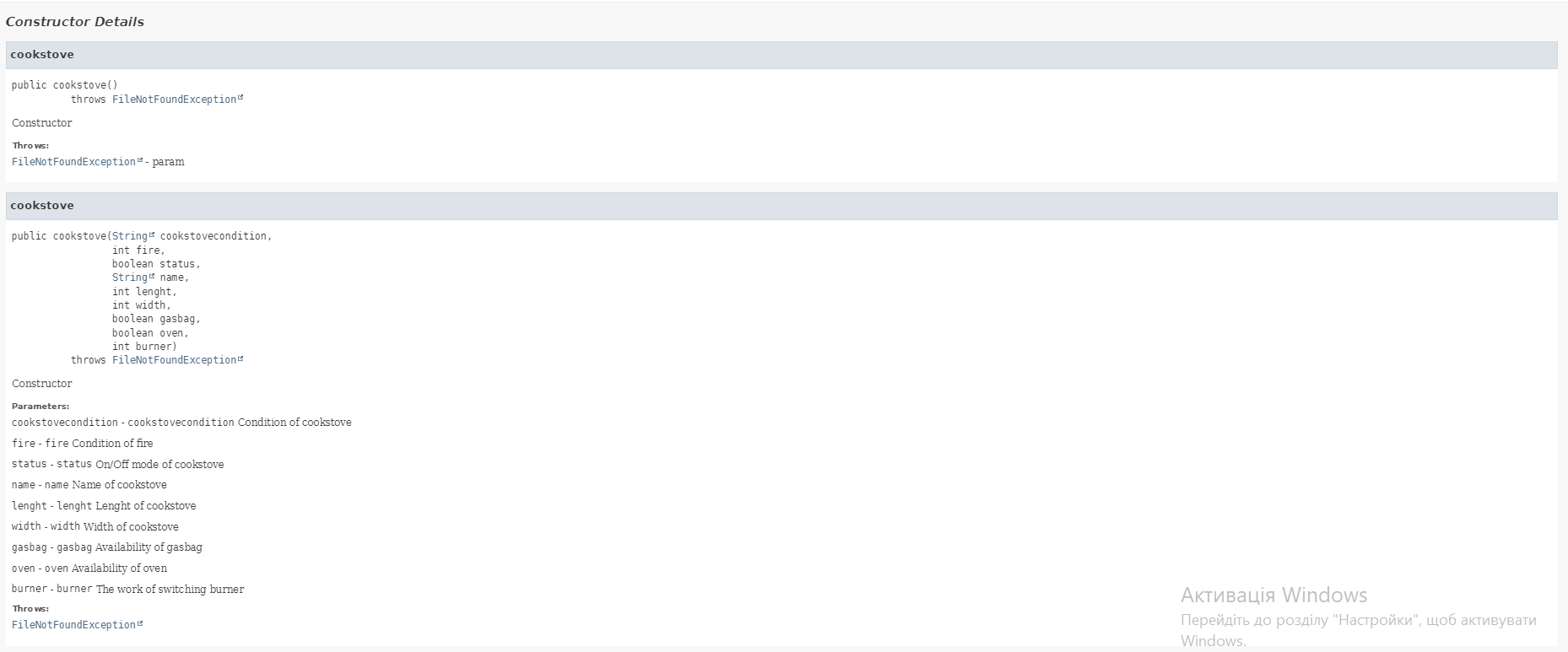
*Згенерована документація*

**

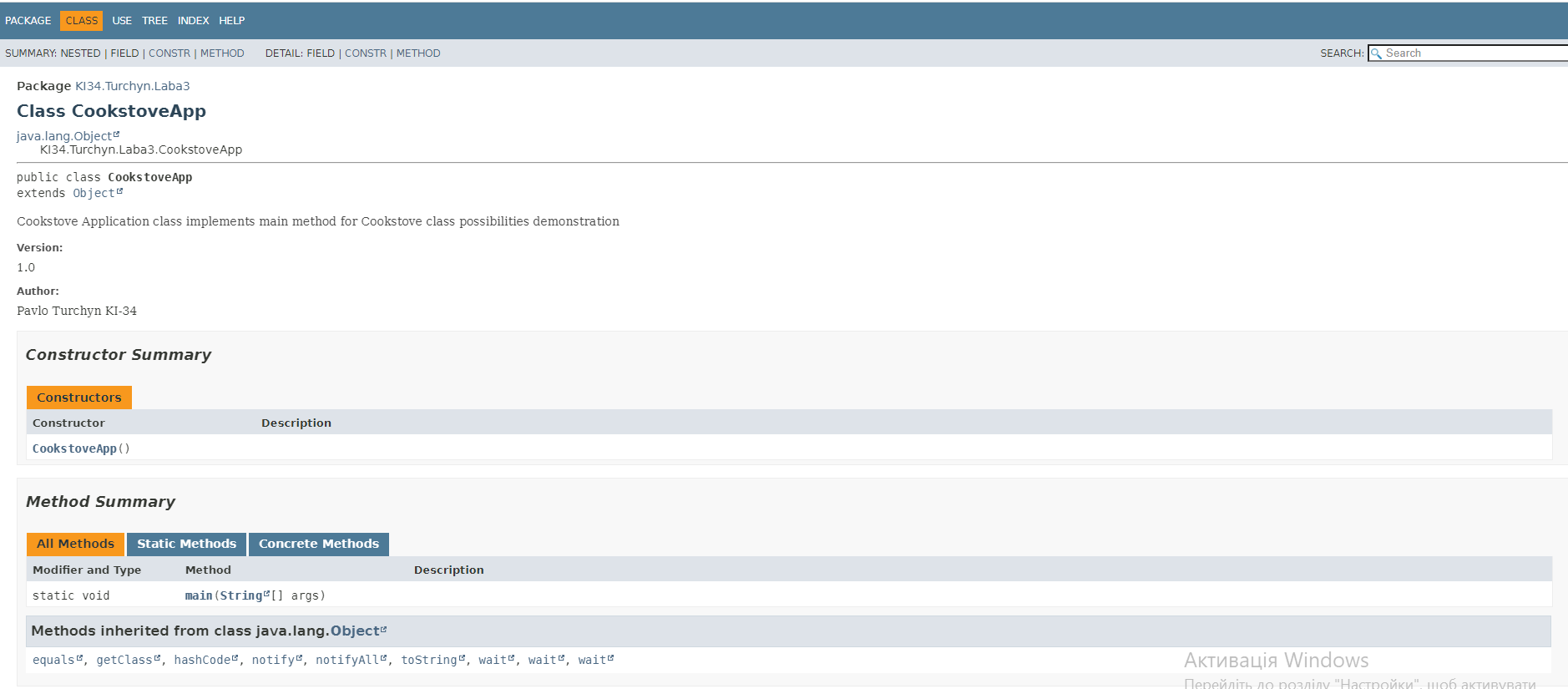
*Інформація про клас Cookstove*

**

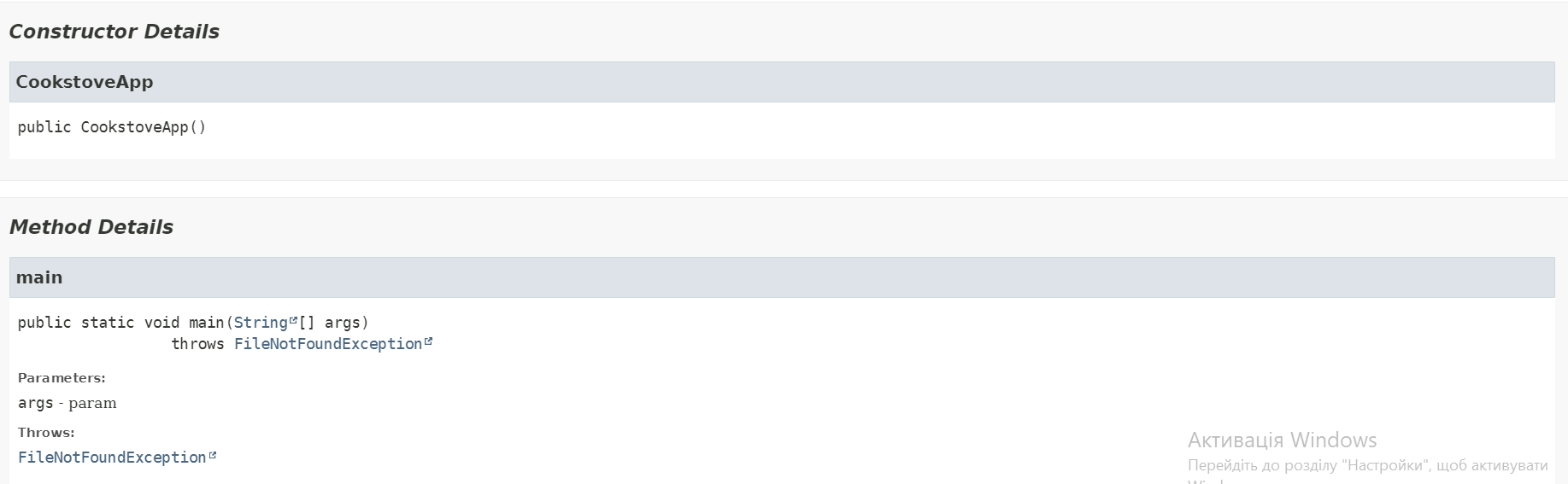
*Інформація про клас Cookstove*

**

*Інформація про клас Cookstove*

**

*Інформація про клас CookstoveApp*

**

*Інформація про клас CookstoveApp*

**Відповіді на контрольні запитання:**

1. **Синтаксис визначення класу?**

Синтаксис оголошення простого класу в мові Java має наступний вигляд:

[public] class НазваКласу

{

[конструктори]

[методи]

[поля]

}

1. **Синтаксис визначення методу?**

Синтаксис оголошення методу наступний:

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

{

[Тіло методу]

[return [значення]];

}

1. **Синтаксис оголошення поля?**

Синтаксис оголошення поля наступний:

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

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

Синтаксис оголошення та ініціалізування константного поля наступний:

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

1. **Які є способи ініціалізації полів?**

Ініціалізацію полів при створенні об’єкту можна здійснювати трьома способами:

* у конструкторі;
* явно при оголошені поля;
* у блоці ініціалізації (виконується перед виконанням конструктора).

1. **Синтаксис визначення конструктора?**

Синтаксис оголошення конструктора:

[СпецифікаторДоступу] НазваКласу([параметри])

{

Тіло конструктора

}

1. **Синтаксис оголошення пакету?**

Синтаксис оператора package:

package НазваПакету{.НазваПідпакету};

1. **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.
3. **В чому суть статичного імпорту пакетів?**

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

1. **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 ознайомився з процесом розробки класів та пакетів мовою Java