Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра ЕОМ



**Звіт**

з лабораторної роботи №4

з дисципліни: «Кросплатформенні засоби програмування»

на тему: «Спадкування та інтерфейси»

Виконав: ст.гр. КІ-34

Турчин П.Р.

Прийняв:

викл. каф. ЕОМ

Іванов Ю. С.

**Львів 2022**

**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 №3, для реалізації предметної області заданої варіантом. Суперклас, що реалізований у лабораторній роботі №3, зробити абстрактним. Розроблений підклас має забезпечувати механізми свого коректного функціонування та реалізовувати мінімум один інтерфейс. Програма має розміщуватися в пакеті Група.Прізвище.Lab4 та володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету.
2. Автоматично згенерувати документацію до розробленої програми.
3. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації.
4. Дати відповідь на контрольні запитання.

**Варіант 23(Газова Плита)**

**Лістинг програми:**

**Файл CookstoveApp.java**

*/\*\*  
 \* lab 4 package  
 \*/*package KI34.Turchyn.Lab4;  
  
import java.io.FileNotFoundException;  
*/\*\*  
 \* Gun Application class implements main method for Stove class possibilities  
 demonstration  
 \** ***@author*** *Pavlo Turchyn  
 \** ***@version*** *1.0  
 \*/*public class CookstoveApp {  
 */\*\*  
 \** ***@param*** *args function parameter  
 \** ***@throws*** *FileNotFoundException throw about non-existent file  
 \*/* public static void main(String[] args) throws FileNotFoundException {  
 GasStove object = new GasStove("Great", 10, true, "Bosch", 50, 70, true, true, 1, 10, 0);  
 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.statusCookstove(20);  
 object.Clean();  
  
 object.Extract();  
 object.dispose();  
 }  
}

**Файл CookStove.java**

*/\*\*  
 \* lab 4 package  
 \*/*package KI34.Turchyn.Lab4;  
  
import java.io.\*;  
*/\*\*  
 \* Class <code>Cookstove</code> implements cookstove  
 \* \* @author Pavlo Turchyn KI-34  
 \* \* @version 1.0  
 \*\*/*public abstract class CookStove {  
 protected String cookstovecondition;  
 protected int fire;  
 protected boolean status;  
 protected String name;  
 protected int lenght;  
 protected int width;  
 protected boolean gasbag;  
 protected boolean oven;  
 protected int burner;  
 protected 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();  
 }  
}

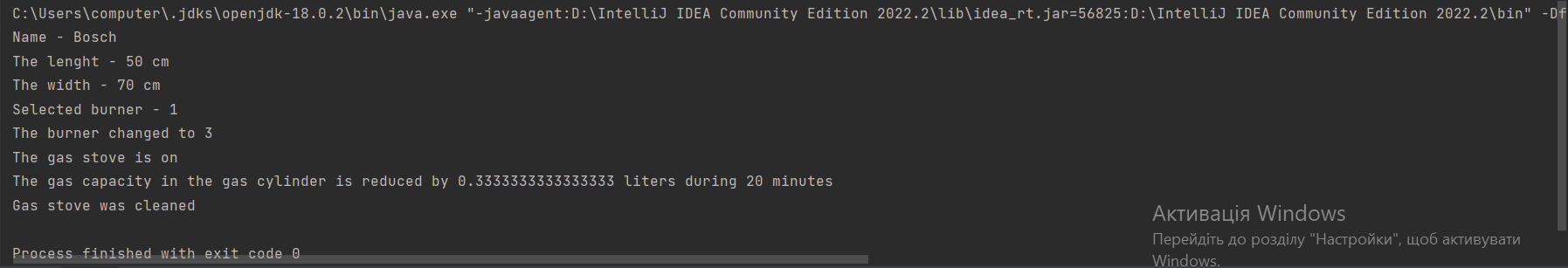
**Файл GasStove.java**

*/\*\*  
 \* lab 4 package  
 \*/*package KI34.Turchyn.Lab4;  
  
import java.io.FileNotFoundException;  
*/\*\*  
 \* Class <code>GasStove</code> extends class CookStove  
 \** ***@author*** *Pavlo Turchyn  
 \** ***@version*** *1.0  
 \*/*public class GasStove extends CookStove implements Clean {  
 public double capacity;  
 public int pollution;  
 */\*\*  
 \* Constructor  
 \** ***@param*** *capacity The capacity of gas bag  
 \** ***@param*** *pollution The level of pollution in gas stove  
 \** ***@throws*** *FileNotFoundException throw about non-existent file  
 \*/* public GasStove(String cookstovecondition, int fire, boolean status, String name, int lenght, int width, boolean gasbag, boolean oven, int burner, double capacity, int pollution) throws FileNotFoundException {  
 super(cookstovecondition, fire, status, name, lenght, width, gasbag, oven, burner);  
 this.capacity = capacity;  
 this.pollution = pollution;  
 }  
  
  
 */\*\*  
 \* Method simulates level of capacity in gas bag during some time  
 \*/* public void statusCookstove(int time){  
 if (status){  
 myWrite.println("The gas stove is on");  
 System.*out*.println("The gas stove is on");  
 capacity -= (double) time/60;  
 myWrite.println("The gas capacity in the gas cylinder is reduced by " + (double) time/60 +" liters during " + time + " minutes");  
 System.*out*.println("The gas capacity in the gas cylinder is reduced by " + (double) time/60 +" liters during " + time + " minutes");  
  
 } else{  
 myWrite.println("The gas stove is off");  
 }  
 }  
 */\*\*  
 \* Method simulates cleaning of gas stove  
 \*/* public void Clean(){  
 pollution=0;  
 myWrite.println("Gas stove was cleaned");  
 System.*out*.println("Gas stove was cleaned");  
  
  
  
 }  
}

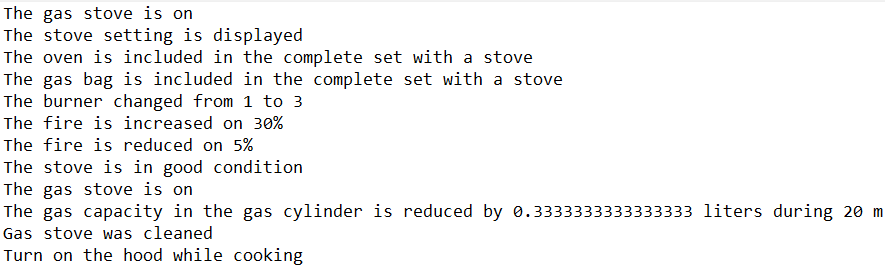
**Файл Clean.java**

*/\*\*  
 \* lab 4 package  
 \*/*package KI34.Turchyn.Lab4;  
  
public interface Clean {  
 void Clean();  
}

**Результат виконання програми:**

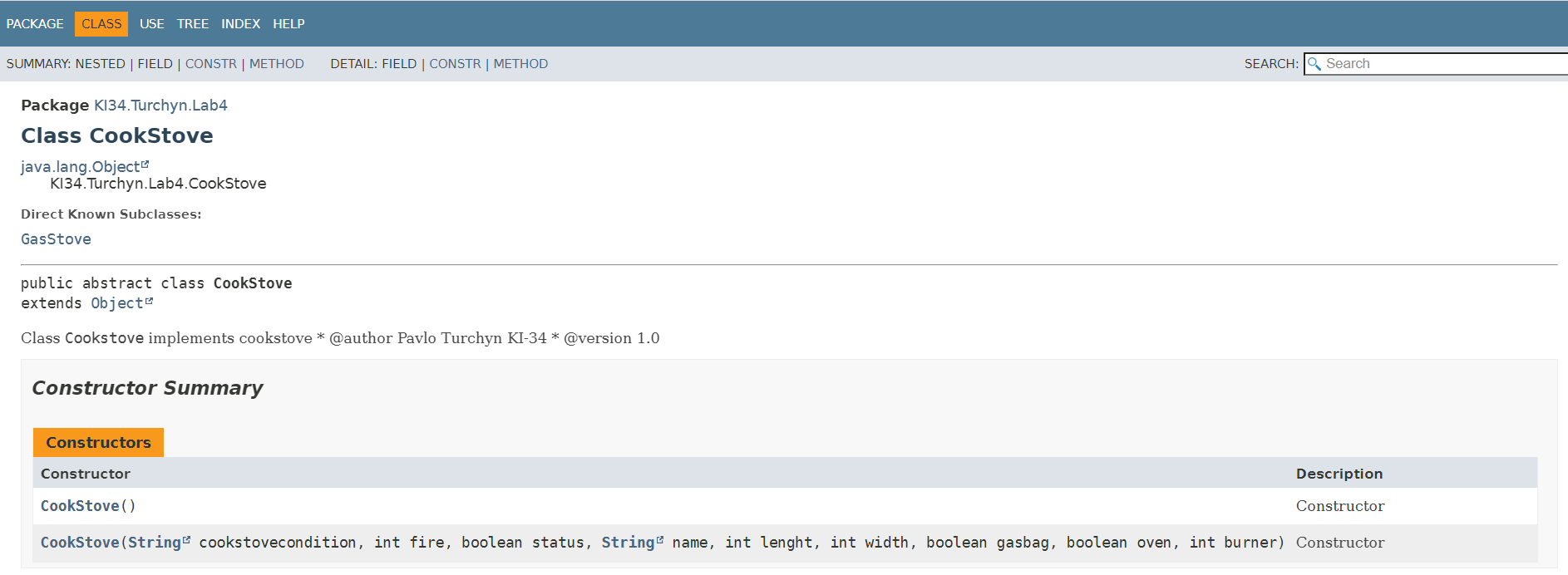
**

*Протокол діяльності в консолі*

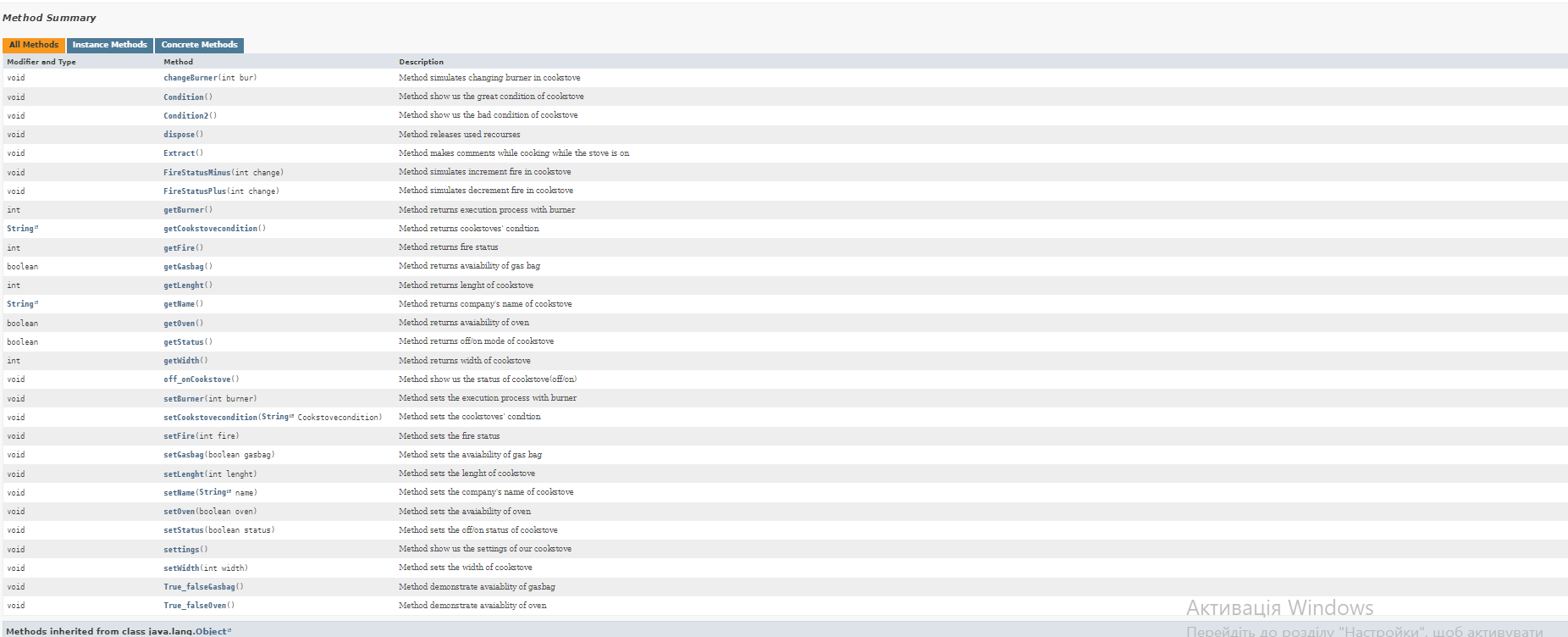
**

*Протокол діяльності в консолі*

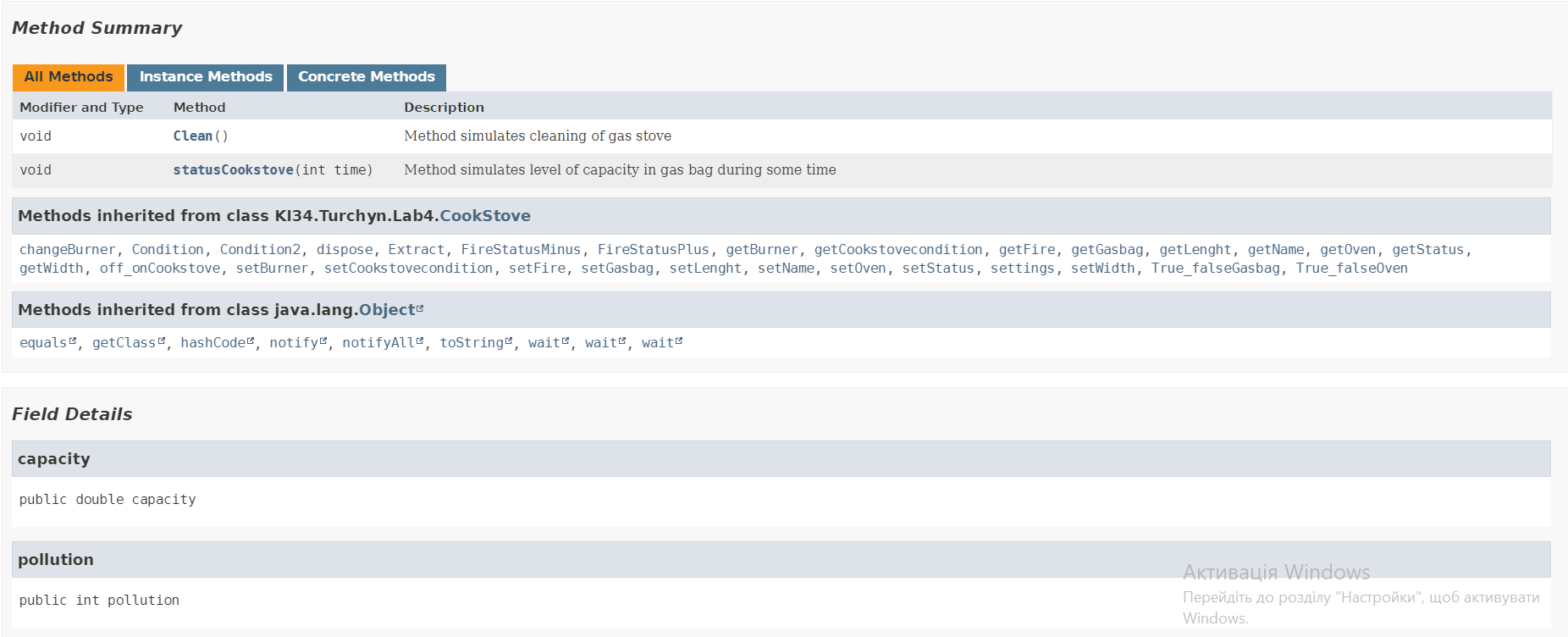
*Згенерована документація*

**

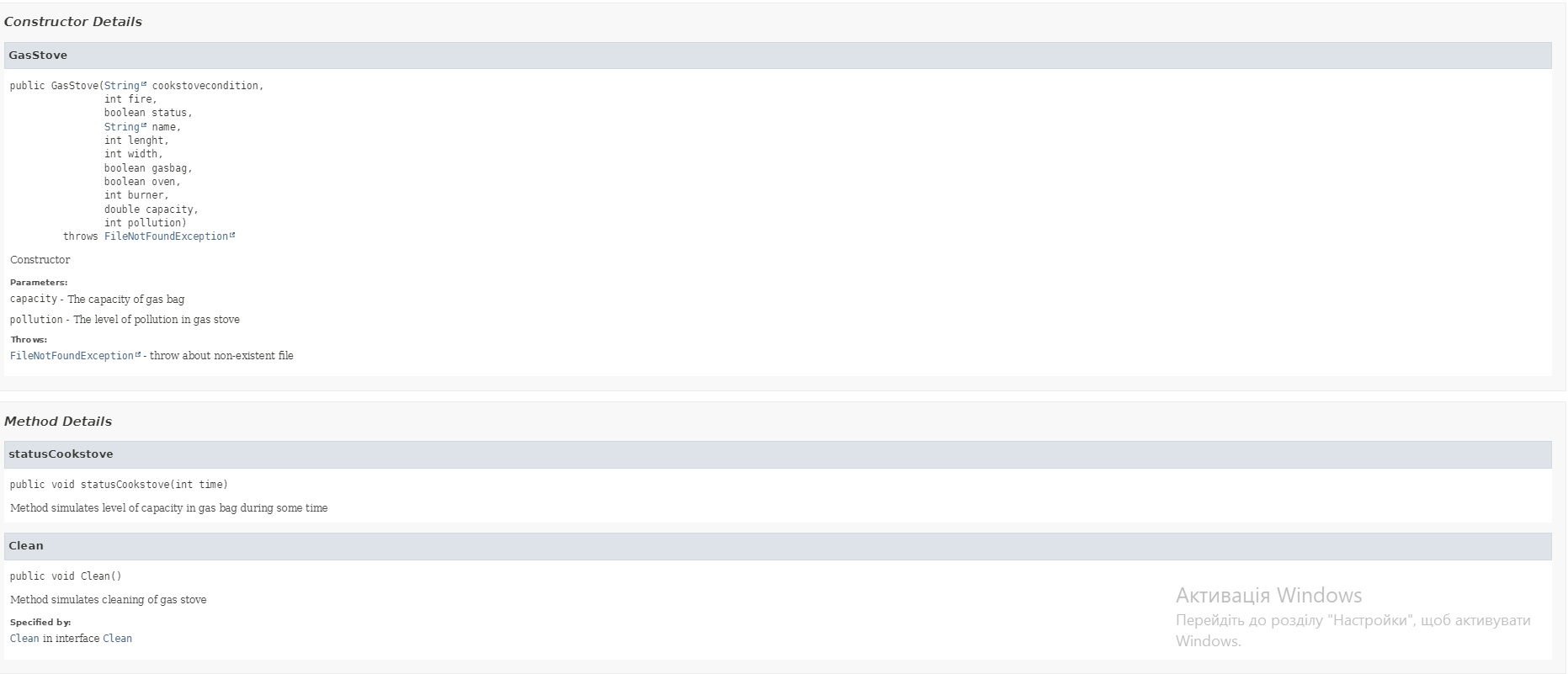
*Інформація про клас CookStove*

**

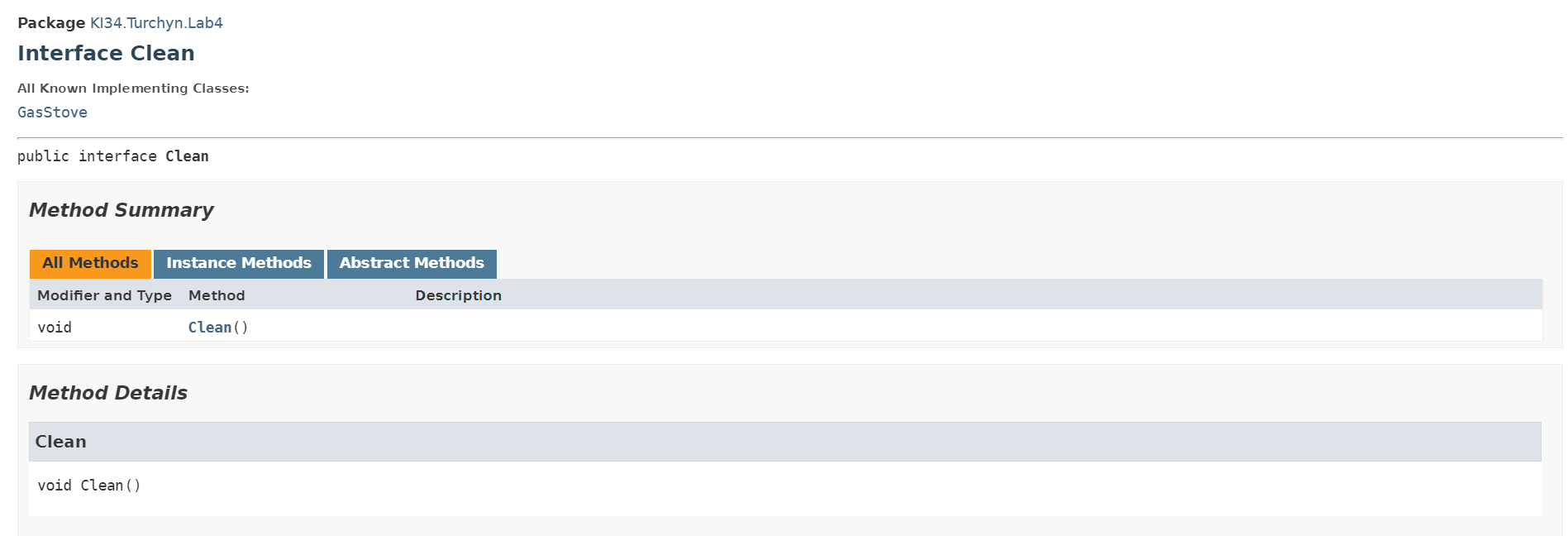
*Інформація про клас CookStove*

**

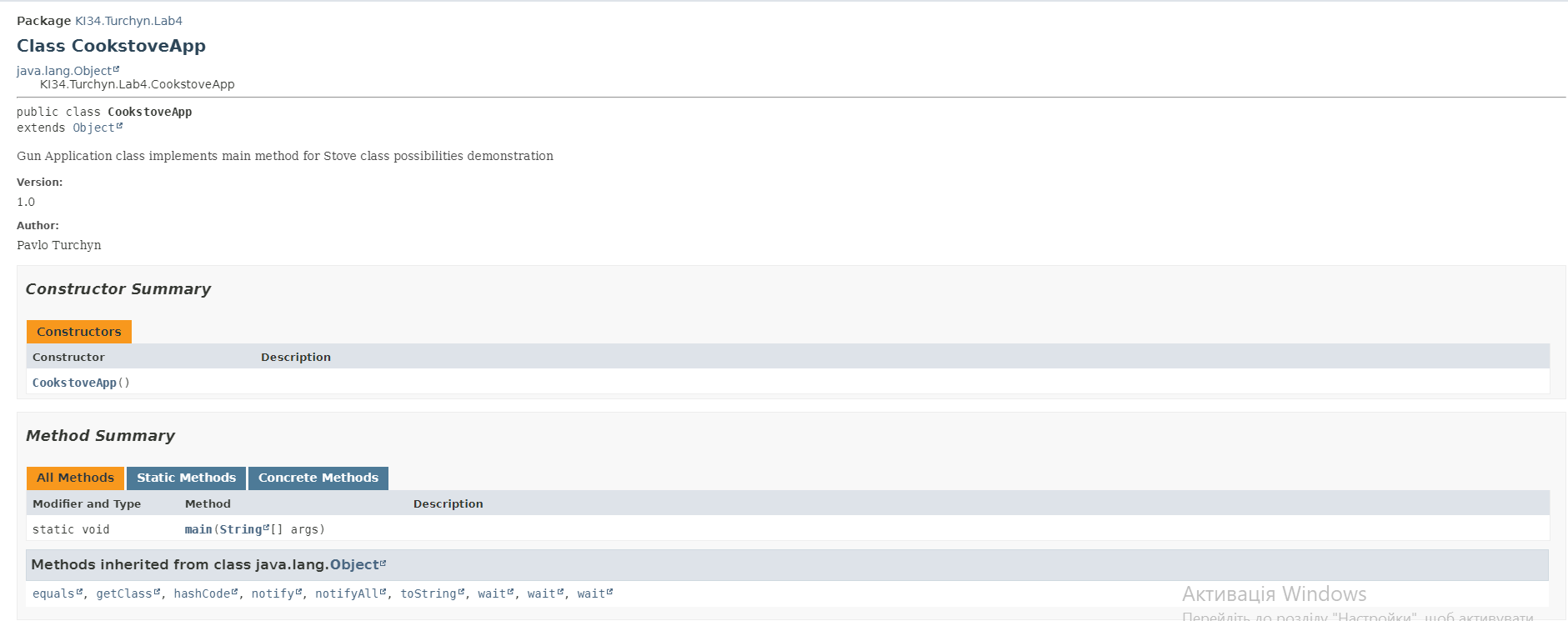
*Інформація про клас GasStove*

**

*Інформація про клас GasStove*

**

*Інформація про інтерфейс Clean*

**

*Інформація про клас CookStoveApp*

**Відповіді на контрольні запитання:**

1. **Синтаксис реалізації спадкування.**

Синтаксис реалізації спадкування:

class Підклас extends Суперклас

{

Додаткові поля і методи

}

1. **Що таке суперклас та підклас??**

В термінах мови Java базовий клас найчастіше називається суперкласом, а похідний клас – підкласом. Дана термінологія запозичена з теорії множин, де підмножина міститься у супермножині.

1. **Як звернутися до членів суперкласу з підкласу?**

Виклик методу суперкласу:

*super.назваМетоду([параметри]);*

Звертання до поля суперкласу:

*super.назваПоля*

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

Статичне зв’язування використовується коли метод є приватним, статичним, фінальним або конструктором.

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

Віртуальна машина повинна викликати версію методу, що відповідає фактичному типу об’єкту на який посилається об’єктна змінна.

Оскільки на пошук необхідного методу потрібно багато часу, то віртуальна машина заздалегідь створює для кожного класу таблицю методів, в якій перелічуються сигнатури всіх методів і фактичні методи, що підлягають виклику. При виклику методу віртуальна машина просто переглядає таблицю методів.

1. **Що таке абстрактний клас та як його реалізувати?**

Абстрактний клас – це клас, для якого не можна створити об’єкти, призначений бути основою для розробки ієрархії класів.

Реалізується за допомогою ключового слова abstract.

1. **Для чого використовується ключове слово instanceof?**

Оператор instanceof дозволяє визначити, чи вказаний об’єкт належить до заданого типу.

1. **Як перевірити чи клас є підкласом іншого класу?**

Щоб перевірити чи клас є підкласом іншого класу, потрібно за допомогою instanceof порівняти, чи дійсно посилання на об’єкт супертипу посилається на об’єкт підтипу.

1. **Що таке інтерфейс?**

Це абстрактний тип, який використовується для визначення поведінки, яку класи повинні реалізовувати. Інтерфейси покликані компенсувати відсутність множинного спадкування у мові Java та гарантують визначення у класах оголошених у собі прототипів методів.

1. **Як оголосити та застосувати інтерфейс?**

Синтаксис оголошення інтерфейсів:

[public] interface НазваІнтерфейсу

{

Прототипи методів та оголошення констант інтерфейсу

}

**Висновок:**

На цій лабораторній роботі я ознайомився з спадкуванням та інтерфейсами у мові Java.