Міністерство освіти і науки України

Національний університет «Львівська політехніка»



ЗВІТ

Про виконання лабораторної роботи №4

З дисципліни “Кросплатформні засоби програмування ”

Варіант-25

**Виконала:**

студ. групи КІ-35

Янчинська Н. Р.

**Прийняв:**

Іванов Ю. С.

Львів 2022

СПАДКУВАННЯ ТА ІНТЕРФЕЙСИ

Мета: 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 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.

ВАРІАНТИ ЗАВДАННЬ

25. Пристрій кліматконтролю

Код програми:

ClimateControl.java

**package** KI35.Yanchynska.Lab4;

**import** java.io.\*;

/\*\*

\* **@author** Yanchynska Natalia

\* Class ClimateControl

\* **@version** 1.0

\*/

**interface** IClimateControl{

**void** changeHumidity(**int** humidity, **int** sec);

}

**public** **class** ClimateControl **extends** Conditioner **implements** IClimateControl{

Display humidityDisplay = **new** Display();

**int** currentHumidity = 0;

/\*\*

\* Constructor

\* **@param** temp

\* **@param** humidity

\* **@throws** FileNotFoundException

\*/

**public** ClimateControl(**int** temp, **int** humidity) **throws** FileNotFoundException

{

**super**(temp);

**this**.currentHumidity = humidity;

humidityDisplay.setTemp(humidity);

}

/\*\*

\* Constructor

\* **@param** temp

\* **@throws** FileNotFoundException

\*/

**public** ClimateControl(**int** temp) **throws** FileNotFoundException

{

**super**(temp);

**this**.currentHumidity = 70;

humidityDisplay.setTemp(70);

}

/\*\*

\* Method changes humidity

\* **@param** humidity

\* **@param** sec

\*/

**public** **void** changeHumidity(**int** humidity, **int** sec) {

**try**{

sendMessage("Changing humidity started... " + "(" + sec + " seconds)");

**for**(**int** i = 0; i < sec; i++)

{

**if**(currentHumidity < humidity)

{

currentHumidity += 1;

}

**else** **if**(currentHumidity > humidity)

{

currentHumidity -= 1;

}

System.***out***.println("Humidity: " + currentHumidity);

Thread.*sleep*(1000);

}

sendMessage("Changing humidity stopped... \n Humidity: " + currentHumidity);

}**catch** (Exception e){

System.***out***.println(e);

}

}

}

Conditioner.java

**package** KI35.Yanchynska.Lab4;

**import** java.io.\*;

/\*\*

\* **@author** Yanchynska Natalia

\* Class Conditioner

\* **@version** 1.0

\*/

**public** **abstract** **class** Conditioner {

Ventilator ventilator = **new** Ventilator();

Filter filter = **new** Filter();

Display display = **new** Display();

**int** currentTemp;

**private** PrintWriter fout;

/\*\*

\* Constructor

\* **@param** currentTemp

\* **@throws** FileNotFoundException

\*/

**public** Conditioner(**int** currentTemp) **throws** FileNotFoundException

{

**this**.currentTemp = currentTemp;

fout = **new** PrintWriter(**new** File("Log.txt"));

}

/\*\*

\* Constructor

\* **@throws** FileNotFoundException

\*/

**public** Conditioner() **throws** FileNotFoundException

{

currentTemp = 10;

fout = **new** PrintWriter(**new** File("Log.txt"));

}

/\*\*

\* Method

\* **@param** temp

\* **@param** sec

\*/

**public** **void** turnOn(**int** temp, **int** sec) {

**try** {

ventilator.setRotation(getRevolutionByTemp(temp));

sendMessage("Air conditioning started... " + "(" + sec + " seconds)");

**for**(**int** i = 0; i < sec; i++)

{

**try**{

**if**(currentTemp < temp)

{

currentTemp += 1;

}**else** **if**(currentTemp > temp){

currentTemp -= 1;

}

filter.tick();

display.setTemp(currentTemp);

System.***out***.println("Temperature: " + display.getTemp());

}

**catch** (Exception e){

System.***out***.println(e);

}

**finally** {

Thread.*sleep*(1000);

}

}

turnOff();

sendMessage("Air conditioning stopped... \n Temperature: " + currentTemp);

} **catch** (Exception e) {

System.***out***.println(e);

}

}

/\*\*

\* Method returns current temp

\*/

**public** **int** getTemp()

{

**return** display.getTemp();

}

/\*\*

\* Method cleans filter

\*/

**public** **void** cleanFilter()

{

filter.clean();

sendMessage("Filter cleaned!");

}

/\*\*

\* Method returns filter pollution

\*/

**public** **int** getFilterPollution()

{

**return** filter.pollutionStatus;

}

/\*\*

\* Method returns ventilator revolution

\*/

**public** **int** getVentilatorRevolution()

{

**return** ventilator.revolutionsPerSecond;

}

/\*\*

\* Method returns changes filter

\*/

**public** **void** changeFilter()

{

filter = **new** Filter();

}

/\*\*

\* Method for setting a new temperature

\*/

**public** **void** changeCurrentTemp(**int** temp)

{

currentTemp = temp;

}

/\*\*

\* Method turnoff the conditioner

\*/

**private** **void** turnOff()

{

ventilator.turnOff();

}

**public** **void** dispose()

{

fout.close();

}

**protected** **void** sendMessage(String message)

{

System.***out***.println(message);

fout.println(message);

}

**private** **int** getRevolutionByTemp(**int** temp)

{

**return** Math.*abs*(temp) \* 10;

}

}

Display.java

**package** KI35.Yanchynska.Lab4;

/\*\*

\* **@author** Yanchynska Natalia

\* Class Display

\* **@version** 1.0

\*/

**public** **class** Display {

**int** temp = 0;

/\*\*

\* Method for setting temperature on display

\* **@param** temp

\*/

**public** **void** setTemp(**int** temp)

{

**this**.temp = temp;

}

/\*\*

\* Method for getting temperature from display

\*/

**public** **int** getTemp()

{

**return** temp;

}

}

Filter.java

**package** KI35.Yanchynska.Lab4;

/\*\*

\* **@author** Yanchynska Natalia

\* Class Filter

\* **@version** 1.0

\*/

**public** **class** Filter {

**int** pollutionStatus = 0;

**int** MAX\_POLLUTION = 1000;

/\*\*

\* Method is called every second

\*/

**public** **int** tick() **throws** Exception

{

**if**(pollutionStatus > MAX\_POLLUTION)

{

**throw** **new** Exception("Filter needs to be cleaned!");

}

pollutionStatus += 1;

**return** pollutionStatus;

}

/\*\*

\* Method for cleaning filter

\*/

**public** **void** clean()

{

pollutionStatus = 0;

}

}

Main.java

**package** KI35.Yanchynska.Lab4;

/\*\*

\* **@author** Yanchynska Natalia

\* Class Main

\* **@version** 1.0

\*/

**public** **class** Main {

/\*\*

\* **@param** args

\*/

**public** **static** **void** main(String[] args)

{

**try**{

**int** START\_TEMP = 43;

System.***out***.println("Temperature: " + START\_TEMP);

ClimateControl conditioner = **new** ClimateControl(START\_TEMP, 70);

conditioner.turnOn(35, 10);

System.***out***.println("Filter pollution: " + conditioner.getFilterPollution());

conditioner.cleanFilter();

System.***out***.println("Temperature: " + conditioner.getTemp());

conditioner.changeHumidity(80, 10);

conditioner.dispose();

}

**catch** (Exception e)

{

System.***out***.println(e);

}

}

}

Ventilator.java

**package** KI35.Yanchynska.Lab4;

/\*\*

\* **@author** Yanchynska Natalia

\* Class Ventilator

\* **@version** 1.0

\*/

**public** **class** Ventilator {

**int** revolutionsPerSecond = 0;

**int** MAX\_REVOLUTIONS = 1000;

/\*\*

\* Method for setting ventilator revolution

\* **@param** rotation

\*/

**public** **void** setRotation(**int** rotation) **throws** Exception

{

**if**(rotation > MAX\_REVOLUTIONS || rotation < 0)

{

**throw** **new** Exception("Invalid revolution");

}

revolutionsPerSecond = rotation;

}

/\*\*

\* Method turn off the ventilator

\*/

**public** **void** turnOff()

{

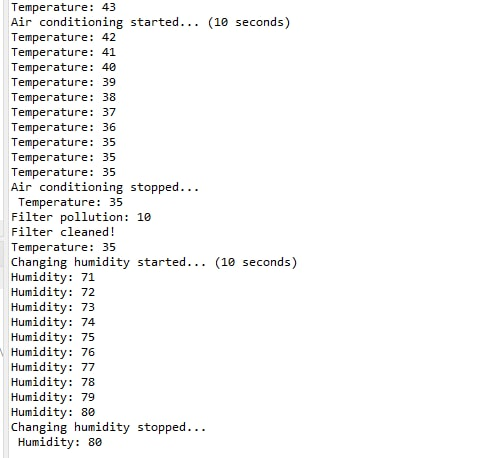
revolutionsPerSecond = 0;

}

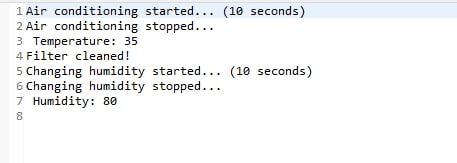
}

Результат виконання програми:

Консоль:



Текстовий файл:



Висновок:

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