Міністерство освіти і науки України

Національний університет „Львівська політехніка”

Кафедра ЕОМ



**Звіт**

з лабораторної роботи №4

з дисципліни: “Кросплатформні засоби програмування”

на тему: “Спадкування та інтерфейси”

Виконав: ст. гр. КІ-34

Зубалій І.А.

Прийняв:

Іванов Ю.С.

Львів – 2022

**Мета:** ознайомитися зі спадкуванням та інтерфейсами у мовіJava.

**Індивідуальне завдання:** написати та налагодити програму на мові Java, що розширює клас, реалізований у лабораторній роботі №3, для реалізації предметної області заданої варіантом:

**5. Машина**

**Хід роботи:**

1. Запустив середовище Eclipse та написав програму згідно індивідуального завдання:

*TruckApp.java*

package lab4;

import java.io.FileNotFoundException;

/\*\*

\* TruckApp class implements main method for Truck class possibilities demonstration

\*

\* @author Zubalii

\* @version 1.0

\*/

public class TruckApp {

/\*\*

\* @param args

\* @throws FileNotFoundException

\*/

public static void main(String[] args) throws FileNotFoundException {

Truck truck = new Truck(5.5f, 8, 12, 3000);

truck.info();

System.out.println("----------------");

truck.loadCargo(5000);

truck.loadCargo(2000);

truck.go();

truck.shiftTo(3);

truck.accelerate();

truck.shiftTo(5);

truck.stopTheCar();

truck.unloadCargo(5000);

truck.unloadCargo(2000);

}

}

*Truck.java*

package lab4;

import java.io.FileNotFoundException;

/\*\*

\* Class Truck implements truck

\*

\* @author Zubalii

\* @version 1.0

\*/

public class Truck extends Car implements Trailer {

private int maxCargoWeight;

private int cargoWeight;

/\*\*

\* Constructor

\*

\* @throws FileNotFoundException

\*/

Truck() throws FileNotFoundException {

super();

setMaxCargoWeight(0);

}

/\*\*

\* Constructor

\*

\* @param engine capacity

\* @throws FileNotFoundException

\*/

Truck(float engineCapacity) throws FileNotFoundException {

super(engineCapacity);

setMaxCargoWeight(0);

}

/\*\*

\* Constructor

\*

\* @param engine capacity

\* @param number of gears

\* @throws FileNotFoundException

\*/

Truck(float engineCapacity, int gears) throws FileNotFoundException {

super(engineCapacity, gears);

setMaxCargoWeight(0);

}

/\*\*

\* Constructor

\*

\* @param engine capacity

\* @param number of gears

\* @param number of wheels

\* @throws FileNotFoundException

\*/

Truck(float engineCapacity, int gears, int wheelsQuantity) throws FileNotFoundException {

super(engineCapacity, gears, wheelsQuantity);

setMaxCargoWeight(0);

}

/\*\*

\* Constructor

\*

\* @param engine capacity

\* @param number of gears

\* @param number of wheels

\* @param max weight of cargo

\* @throws FileNotFoundException

\*/

Truck(float engineCapacity, int gears, int wheelsQuantity, int maxCargoWeight) throws FileNotFoundException {

super(engineCapacity, gears, wheelsQuantity);

setMaxCargoWeight(maxCargoWeight);

}

/\*\*

\* Setter for maxCargoWeight

\*/

@Override

public void setMaxCargoWeight(int weight) {

this.maxCargoWeight = weight;

}

/\*\*

\* Getter for maxCargoWeight

\*/

@Override

public int getMaxCargoWeight() {

return maxCargoWeight;

}

/\*\*

\* Method implements cargo loading

\*/

@Override

public void loadCargo(int weight) {

if (weight <= maxCargoWeight - cargoWeight) {

cargoWeight += weight;

System.out.println("The cargo is loaded");

super.fout.println("The cargo is loaded");

} else {

System.out.println("The cargo is not loaded. It's too heavy");

super.fout.println("The cargo is not loaded. It's too heavy");

}

}

/\*\*

\* Method implements cargo unloading

\*/

@Override

public void unloadCargo(int weight) {

if (weight <= cargoWeight) {

cargoWeight -= weight;

System.out.println("The cargo is unloaded");

super.fout.println("The cargo is unloaded");

} else {

System.out.println("You can't do it");

super.fout.println("You can't do it");

}

}

/\*\*

\* Getter for maxCargoWeight

\*/

public int getCargoWeight() {

return cargoWeight;

}

/\*\*

\* Method implements cargo weight showing

\*/

public void viewCargoWeight() {

System.out.println("Current weight of cargo is: " + cargoWeight);

super.fout.println("Current weight of cargo is: " + cargoWeight);

}

/\*\*

\* Setter for maxCargoWeight

\*/

public void setCargoWeight(int cargoWeight) {

this.cargoWeight = cargoWeight;

}

/\*\*

\* Method shows truck info

\*/

@Override

public void info() {

super.info();

System.out.println("Max weight of cargo is: " + maxCargoWeight);

super.fout.println("Max weight of cargo is: " + maxCargoWeight);

}

}

*Trailer.java*

package lab4;

/\*\*

\* Interface Trailer describes truck trailer

\*

\* @author Zubalii

\* @version 1.0

\*/

public abstract interface Trailer {

/\*\*

\* Method sets max weight of cargo

\*/

void setMaxCargoWeight(int weight);

/\*\*

\* Method gets max weight of cargo

\*/

int getMaxCargoWeight();

/\*\*

\* Method describes max cargo loading

\*/

void loadCargo(int weight);

/\*\*

\* Method describes max cargo unloading

\*/

void unloadCargo(int weight);

}

*Car.java*

package lab4;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.PrintWriter;

/\*\*

\* Class Car implements car

\*

\* @author Zubalii

\* @version 1.0

\*/

public abstract class Car {

private Engine engine;

private Gearbox gearbox;

private Wheels wheels;

protected PrintWriter fout;

/\*\*

\* Constructor

\*

\* @throws FileNotFoundException

\*/

Car() throws FileNotFoundException {

engine = new Engine();

gearbox = new Gearbox();

wheels = new Wheels();

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

}

/\*\*

\* Constructor

\*

\* @param engine capacity

\* @throws FileNotFoundException

\*/

Car(float engineCapacity) throws FileNotFoundException {

engine = new Engine(engineCapacity);

gearbox = new Gearbox();

wheels = new Wheels();

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

}

/\*\*

\* Constructor

\*

\* @param engine capacity

\* @param number of gears

\* @throws FileNotFoundException

\*/

Car(float engineCapacity, int gears) throws FileNotFoundException {

engine = new Engine(engineCapacity);

gearbox = new Gearbox(gears);

wheels = new Wheels();

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

}

/\*\*

\* Constructor

\*

\* @param engine capacity

\* @param number of gears

\* @param number of wheels

\* @throws FileNotFoundException

\*/

Car(float engineCapacity, int gears, int wheelsQuantity) throws FileNotFoundException {

engine = new Engine(engineCapacity);

gearbox = new Gearbox(gears);

wheels = new Wheels(wheelsQuantity);

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

}

/\*\*

\* Getter for Engine

\*/

public Engine getEngine() {

return engine;

}

/\*\*

\* Setter for Engine

\*/

public void setEngine(Engine engine) {

this.engine = engine;

}

/\*\*

\* Getter for Gearbox

\*/

public Gearbox getGearbox() {

return gearbox;

}

/\*\*

\* Setter for Gearbox

\*/

public void setGearbox(Gearbox gearbox) {

this.gearbox = gearbox;

}

/\*\*

\* Getter for Wheels

\*/

public Wheels getWheels() {

return wheels;

}

/\*\*

\* Setter for Wheels

\*/

public void setWheels(Wheels wheels) {

this.wheels = wheels;

}

/\*\*

\* Method implements car moving

\*/

public void go() {

engine.start();

gearbox.shiftGear(1);

engine.gas();

wheels.spin();

System.out.println("The car started to go...");

fout.println("The car started to go...");

}

/\*\*

\* Method implements car acceleration

\*/

public void accelerate() {

while (gearbox.getCurrentGear() != gearbox.getNumberOfGears()) {

gearbox.shiftGear(gearbox.getCurrentGear() + 1);

engine.gas();

}

}

/\*\*

\* Method implements car stopping

\*/

public void stopTheCar() {

wheels.brake();

gearbox.shiftGear(0);

System.out.println("The car is stoped...");

fout.println("The car is stoped...");

}

/\*\*

\* Method implements car reverse moving

\*/

public void goReverse() {

gearbox.shiftReverse();

engine.gas();

wheels.spin();

}

/\*\*

\* Method implements shifting to the required gear

\*

\* @param gear - required gear

\*/

public void shiftTo(int gear) {

gearbox.shiftGear(gear);

}

/\*\*

\* Method shows car info

\*/

public void info() {

System.out.println("Engine capacity is: " + engine.getCapacity());

fout.println("Engine capacity is: " + engine.getCapacity());

System.out.println("Number of gears is: " + gearbox.getNumberOfGears());

fout.println("Number of gears is: " + gearbox.getNumberOfGears());

System.out.println("Number of wheels is: " + wheels.getNumberOfWheels());

fout.println("Number of wheels is: " + wheels.getNumberOfWheels());

}

/\*\*

\* Method releases used recourses

\*/

public void dispose() {

fout.close();

}

/\*\*

\* Class Engine implements car engine

\*

\* @author Zubalii

\* @version 1.0

\*/

class Engine {

private float capacity;

/\*\*

\* Constructor

\*/

public Engine() {

this.capacity = 1.0f;

}

/\*\*

\* Constructor

\*

\* @param engine capacity

\*/

public Engine(float capacity) {

this.capacity = capacity;

}

/\*\*

\* Getter for capacity

\*/

public float getCapacity() {

return capacity;

}

/\*\*

\* Setter for capacity

\*/

public void setCapacity(float capacity) {

this.capacity = capacity;

}

/\*\*

\* Method implements engine start

\*/

public void start() {

System.out.println("Engine is started...");

fout.println("Engine is started...");

}

/\*\*

\* Method implements the operation of the accelerator pedal

\*/

public void gas() {

System.out.println("Wroom...wroom...");

fout.println("Wroom...wroom...");

}

}

/\*\*

\* Class Engine implements car gearbox

\*

\* @author Zubalii

\* @version 1.0

\*/

class Gearbox {

private int currentGear = 0;

private int numberOfGears;

/\*\*

\* Constructor

\*/

public Gearbox() {

this.numberOfGears = 4;

}

/\*\*

\* Constructor

\*

\* @param number of gears

\*/

public Gearbox(int numberOfGears) {

this.numberOfGears = numberOfGears;

}

/\*\*

\* Getter for numberOfGears

\*/

public int getNumberOfGears() {

return numberOfGears;

}

/\*\*

\* Setter for numberOfGears

\*/

public void setNumberOfGears(int numberOfGears) {

this.numberOfGears = numberOfGears;

}

/\*\*

\* Method implements gear shifting

\*

\* @param required gear

\*/

public void shiftGear(int gear) {

if ((gear >= 0) && (gear <= numberOfGears)) {

currentGear = gear;

System.out.println("Shifted " + currentGear + " gear");

fout.println("Shifted " + currentGear + " gear");

} else if (gear == 0) {

System.out.println("Shifted neutral gear");

fout.println("Shifted neutral gear");

} else {

System.out.println("Krhrhrh...This gear doesn't exist");

fout.println("Krhrhrh...This gear doesn't exist");

}

}

/\*\*

\* Method implements reverse gear

\*/

public void shiftReverse() {

currentGear = -1;

System.out.println("Shifted into reverse");

fout.println("Shifted into reverse");

}

/\*\*

\* Method returns current gear

\*

\* @return current gear

\*/

public int getCurrentGear() {

return currentGear;

}

}

/\*\*

\* Class Wheels implements car wheels

\*

\* @author Zubalii

\* @version 1.0

\*/

class Wheels {

private int numberOfWheels;

/\*\*

\* Constructor

\*/

public Wheels() {

this.numberOfWheels = 4;

}

/\*\*

\* Constructor

\*

\* @param number of wheels

\*/

public Wheels(int numberOfWheels) {

this.numberOfWheels = numberOfWheels;

}

/\*\*

\* Getter for numberOfWheels

\*/

public int getNumberOfWheels() {

return numberOfWheels;

}

/\*\*

\* Setter for numberOfWheels

\*/

public void setNumberOfWheels(int numberOfWheels) {

this.numberOfWheels = numberOfWheels;

}

/\*\*

\* Method implements the rotation of the wheels

\*/

public void spin() {

System.out.println("Wheels are spinning...");

fout.println("Wheels are spinning...");

}

/\*\*

\* Method implements braking

\*/

public void brake() {

System.out.println("Wheels are braking...");

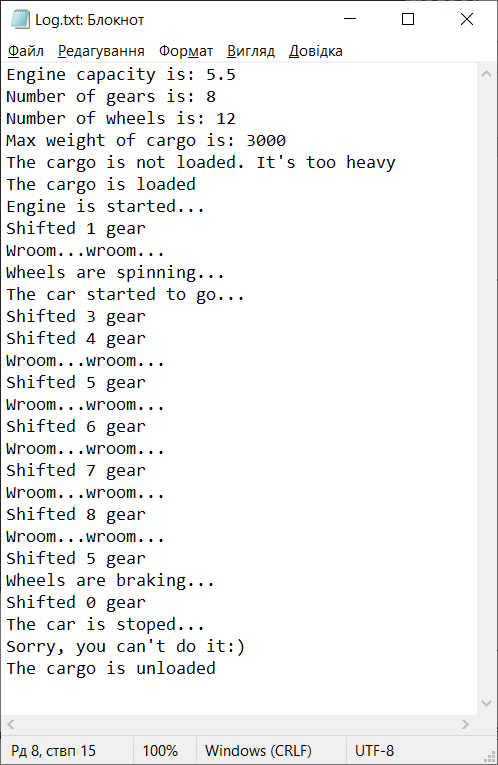
fout.println("Wheels are braking...");

}

}

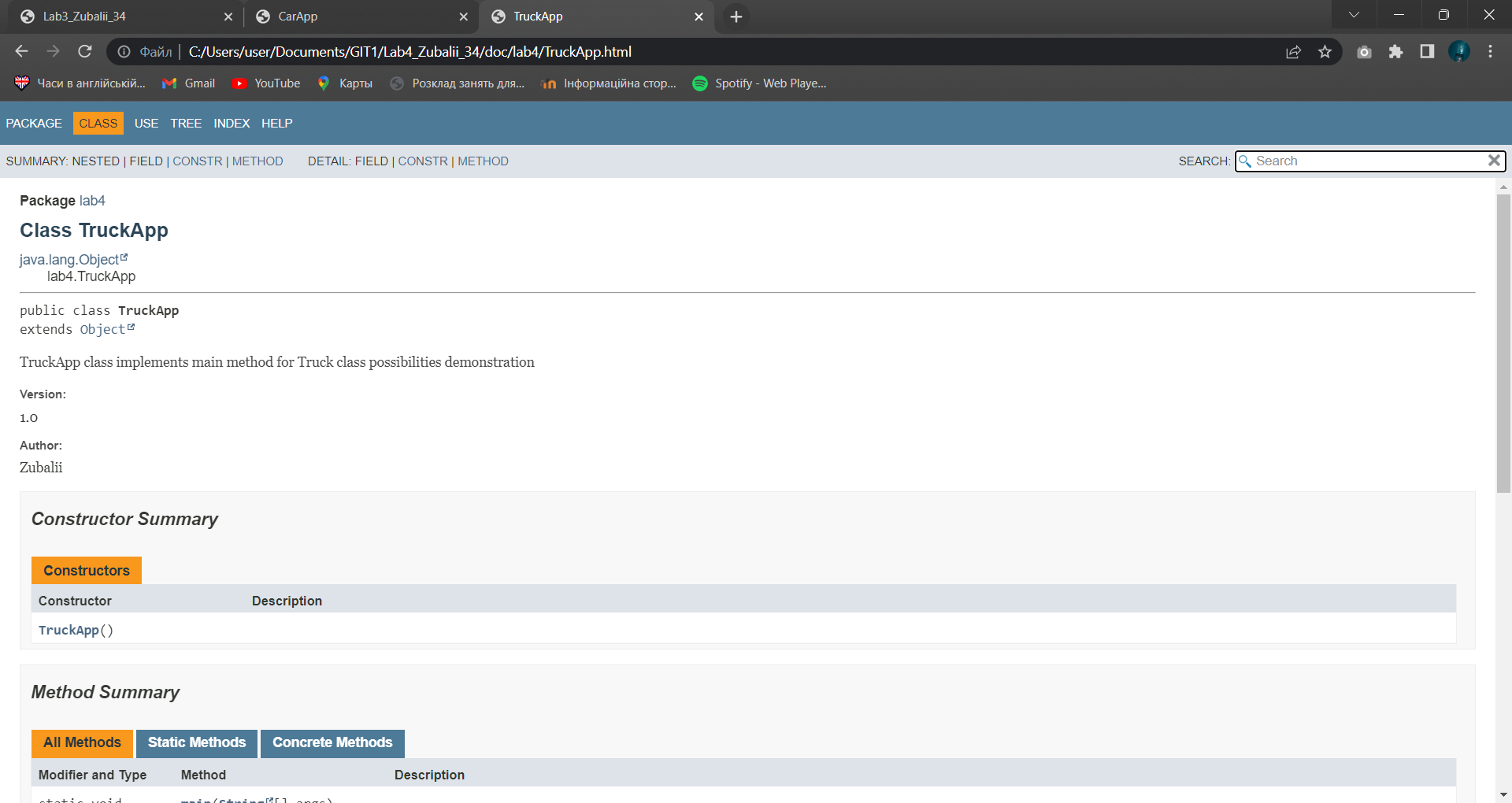
}

1. Після виконання програми переглянув створений файл Log.txt:



*Рис.1. Результат виконання програми*

1. Згенерував документацію



*Рис.3. Згенерований html-файл*

**Висновок:**

На даній лабораторній роботі ознайомився зі спадкуванням та інтерфейсами у мові Java.