Міністерство освіти і науки України

Національний університет „Львівська політехніка” Кафедра “Електронних обчислювальних машин”



Звіт

з лабораторної роботи №4

з дисципліни «Кросплатформні засоби програмування»

на тему: «Спадкування та інтерфейси»

Виконав:ст.гр.КІ-34

Шкраба Р. І.

Прийняв:

Іванов Ю.С.

**Львів 2022**

**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.

**Варіант:** 24

|  |  |
| --- | --- |
| 24. | Спорядження військового альпініста |

*Лістинг*

public class Grapnel  
{  
 private double maxWeight;  
 private double length;  
  
 public Grapnel(double maxWeight, double length) {  
 this.maxWeight = maxWeight;  
 this.length = length;  
 }  
  
 public double getMaxWeight() {  
 return maxWeight;  
 }  
  
 public void setMaxWeight(double maxWeight) {  
 this.maxWeight = maxWeight;  
 }  
  
 public double getLength() {  
 return length;  
 }  
  
 public void setLength(double length) {  
 this.length = length;  
 }  
  
 @Override  
 public String toString() {  
 return "Grapnel{ " +  
 "maxWeight = " + maxWeight +  
 ", length = " + length +  
 '}';  
 }  
}

import java.io.File;  
import java.io.FileWriter;  
import java.io.IOException;  
import java.text.SimpleDateFormat;  
import java.util.Date;  
  
*/\*\*  
 \* Class Logger. Was created to log information, errors and warnings. Also there was implemented Singelton  
 \** ***@author*** *\** ***@version*** *1.0  
 \*/*public class Logger  
{  
 private static Logger *logger*;  
 private final String fileName;  
  
 protected final String infoFlag = new String("[INFO] ");  
 protected final String errorFlag = new String("[ERROR] ");  
 protected final String warningFlag = new String("[WARNING] ");  
  
 */\*\*  
 \* Constructor  
 \** ***@param*** *fileName  
 \*/* private Logger(String fileName)  
 {  
 this.fileName = fileName;  
 File loggerFile = null;  
 FileWriter fout = null;  
 try  
 {  
 loggerFile = new File(fileName);  
 fout = new FileWriter(loggerFile, true);  
 SimpleDateFormat formatter= new SimpleDateFormat("yyyy-MM-dd 'at' HH:mm:ss z");  
 Date date = new Date(System.*currentTimeMillis*());  
 fout.write("[" + formatter.format(date) + "] " + "Logger start to work\n");  
 }  
 catch (IOException e)  
 {  
 System.*err*.println("Something wrong with log file" + e.getMessage());  
 System.*exit*(1);  
 }  
 finally  
 {  
 try  
 {  
 fout.flush();  
 fout.close();  
 }  
 catch (IOException e)  
 {  
 System.*out*.println(e.getMessage());  
 }  
 }  
 }  
  
 */\*\*  
 \* Method to do logging  
 \** ***@param*** *massege  
 \*/* public void log(String massege)  
 {  
 File loggerFile = null;  
 FileWriter fout = null;  
 try  
 {  
 loggerFile = new File(this.fileName);  
 fout = new FileWriter(loggerFile, true);  
 SimpleDateFormat formatter= new SimpleDateFormat("yyyy-MM-dd 'at' HH:mm:ss z");  
 Date date = new Date(System.*currentTimeMillis*());  
 fout.write("[" + formatter.format(date) + "] " + massege + "\n");  
 }  
 catch (IOException e)  
 {  
 System.*err*.println("Something wrong with log file" + e.getMessage());  
 System.*exit*(1);  
 }  
 finally  
 {  
 try  
 {  
 fout.flush();  
 fout.close();  
 }  
 catch (IOException | NullPointerException e)  
 {  
 System.*out*.println(e.getMessage());  
 }  
 }  
 }  
  
 */\*\*  
 \* Singleton implementation  
 \** ***@param*** *fileName  
 \** ***@return*** *\*/* public static Logger getLogger(String fileName)  
 {  
 if (*logger* == null)  
 {  
 *logger* = new Logger(fileName);  
 }  
 return *logger*;  
 }  
  
 */\*\*  
 \* Getter for logger  
 \** ***@return*** *logger  
 \*/* public static Logger getLogger()  
 {  
 return *logger*;  
 }  
  
}

public class Main {  
 public static void main(String[] args) {  
 MilitaryMountaineeringEquipment mountaineeringEquipment = new MilitaryMountaineeringEquipment(  
 new Grapnel(121.3, 45),  
 new Piolet("Brand", 32.1),  
 new Rope(37.4, 2), "New Brand", 4500);  
  
 mountaineeringEquipment.IsEquipmentHoldUp(130);  
 mountaineeringEquipment.ThrowTheHook(37);  
 System.*out*.println(mountaineeringEquipment);  
 }  
}

public class MilitaryMountaineeringEquipment extends MountaineeringEquipment implements TakeWeapon  
{  
  
 */\*\*  
 \* Constructor  
 \*  
 \** ***@param*** *grapnel  
 \** ***@param*** *piolet  
 \** ***@param*** *rope  
 \** ***@param*** *brand  
 \** ***@param*** *price  
 \*/* public MilitaryMountaineeringEquipment(Grapnel grapnel, Piolet piolet, Rope rope, String brand, double price) {  
 super(grapnel, piolet, rope, brand, price);  
 logger.log(logger.infoFlag + "MilitaryMountaineeringEquipment was called");  
 }  
  
 */\*\*  
 \* Overrided method to throw hook  
 \** ***@param*** *length  
 \*/* @Override  
 public void ThrowTheHook(double length) {  
 logger.log(logger.infoFlag + "MilitaryMountaineeringEquipment method throw hook was called");  
 if(length <= rope.getLength())  
 {  
 System.*out*.println("You have enough rope to throw hook");  
 }  
 else  
 {  
 System.*out*.println("Your rope is short to throw hook");  
 }  
 }  
  
 @Override  
 public void IsEquipmentHoldUp(double weight) {  
 logger.log(logger.infoFlag + "MilitaryMountaineeringEquipment method IsEquipmentHoldUp was called");  
 if(grapnel.getMaxWeight() >= weight)  
 {  
 System.*out*.println("Equipment hold up you");  
 }  
 else  
 {  
 System.*out*.println("Equipment don't hold up you");  
 }  
  
 }  
  
 */\*\*  
 \* Implemented intarface  
 \** ***@param*** *weight  
 \** ***@return*** *\*/* @Override  
 public double TakeWeapon(double weight) {  
 logger.log(logger.infoFlag + "MilitaryMountaineeringEquipment method TakeWeapon was called");  
 return weight + 20;  
 }  
  
 @Override  
 public String toString() {  
 return "MilitaryMountaineeringEquipment: \n" +  
 "grapnel=" + grapnel +  
 "\npiolet=" + piolet +  
 "\nrope=" + rope +  
 "\nbrand='" + brand + '\'' +  
 "\nprice=" + price;  
 }  
}

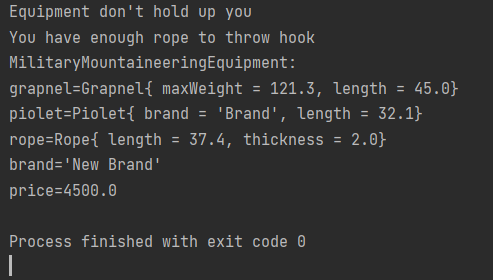
*/\*\*  
 \* Class  
 \** ***@author*** *\** ***@version*** *1.0  
 \*/*public abstract class MountaineeringEquipment  
{  
 protected Grapnel grapnel;  
 protected Piolet piolet;  
 protected Rope rope;  
 protected String brand;  
 protected double price;  
 protected Logger logger = Logger.*getLogger*("logs.txt");  
  
 */\*\*  
 \* Constructor  
 \** ***@param*** *grapnel  
 \** ***@param*** *piolet  
 \** ***@param*** *rope  
 \** ***@param*** *brand  
 \** ***@param*** *price  
 \*/* public MountaineeringEquipment(Grapnel grapnel, Piolet piolet, Rope rope, String brand, double price) {  
 this.grapnel = grapnel;  
 this.piolet = piolet;  
 this.rope = rope;  
 this.brand = brand;  
 this.price = price;  
 }  
  
 */\*\*  
 \* Method to throw hook  
 \** ***@param*** *length  
 \*/* public abstract void ThrowTheHook(double length);  
  
 */\*\*  
 \* Method to check is equipment hold up  
 \** ***@param*** *weight  
 \*/* public abstract void IsEquipmentHoldUp(double weight);  
  
 public Grapnel getGrapnel() {  
 return grapnel;  
 }  
  
 public void setGrapnel(Grapnel grapnel) {  
 this.grapnel = grapnel;  
 }  
  
 public Piolet getPiolet() {  
 return piolet;  
 }  
  
 public void setPiolet(Piolet piolet) {  
 this.piolet = piolet;  
 }  
  
 public Rope getRope() {  
 return rope;  
 }  
  
 public void setRope(Rope rope) {  
 this.rope = rope;  
 }  
  
 public String getBrand() {  
 return brand;  
 }  
  
 public void setBrand(String brand) {  
 this.brand = brand;  
 }  
  
 public double getPrice() {  
 return price;  
 }  
  
 public void setPrice(double price) {  
 this.price = price;  
 }  
  
 @Override  
 public String toString() {  
 return "MountaineeringEquipment: \n" +  
 "grapnel = " + grapnel +  
 "\npiolet = " + piolet +  
 "\nrope = " + rope +  
 "\nbrand = '" + brand + '\'' +  
 "\nprice = " + price;  
 }  
}

public class Piolet  
{  
 private String brand;  
 private double length;  
  
 public Piolet(String brand, double length) {  
 this.brand = brand;  
 this.length = length;  
 }  
  
 public String getBrand() {  
 return brand;  
 }  
  
 public void setBrand(String brand) {  
 this.brand = brand;  
 }  
  
 public double getLength() {  
 return length;  
 }  
  
 public void setLength(double length) {  
 this.length = length;  
 }  
  
 @Override  
 public String toString() {  
 return "Piolet{ " +  
 "brand = '" + brand + '\'' +  
 ", length = " + length +  
 '}';  
 }  
}

public class Rope  
{  
 private double length;  
 private double thickness;  
  
 public Rope(double length, double thickness)  
 {  
 this.length = length;  
 this.thickness = thickness;  
 }  
  
 public double getLength() {  
 return length;  
 }  
  
 public void setLength(double length) {  
 this.length = length;  
 }  
  
 public double getThickness() {  
 return thickness;  
 }  
  
 public void setThickness(double thickness) {  
 this.thickness = thickness;  
 }  
  
 @Override  
 public String toString() {  
 return "Rope{ " +  
 "length = " + length +  
 ", thickness = " + thickness +  
 '}';  
 }  
}

public interface TakeWeapon  
{  
 double TakeWeapon(double weight);  
}

**Результати:**

****

**Висновок:** виконавши цю лабораторну роботу, я ознайомився з спадкуванням та інтерфейсами у мові Java.