Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут комп'ютерних технологій, автоматики та метрології

Кафедра ЕОМ



**Звіт**

**До лабораторної роботи №4**

# З дисципліни: «Кросплатформні засоби програмування»

На тему «Спадкування та інтерфейси»

# Варіант №30

Виконав: ст. гр. КІ-35

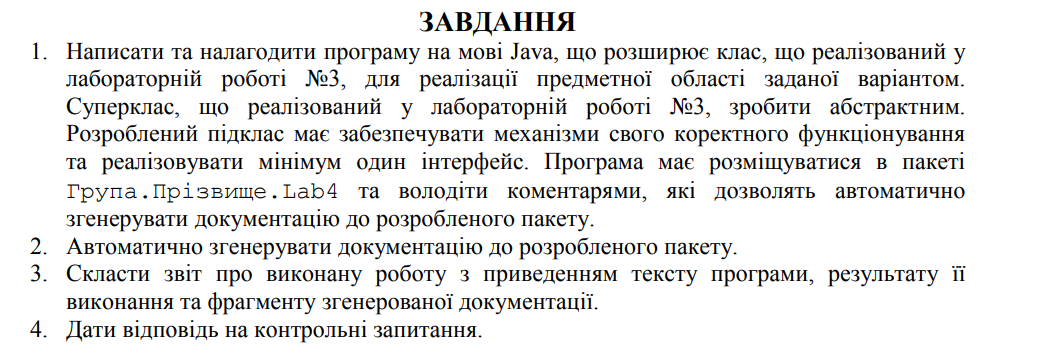
Войнагій Р.В.

Прийняв:

Іванов Ю.С.

Львів – 2022

**Мета:** ознайомитися з спадкуванням та інтерфейсами у мові Java.





**Виконання:**

**Лістинг програми:**

public class ElectricTrain extends Locomotive implements Refuel  
{  
 */\*\*  
 \* Constructor  
 \*  
 \** ***@param*** *freightCar  
 \** ***@param*** *motor  
 \** ***@param*** *wheel  
 \** ***@param*** *fuel  
 \*/* public ElectricTrain(FreightCar freightCar, Motor motor, Wheel wheel, double fuel) {  
 super(freightCar, motor, wheel, fuel);  
 logger.log(logger.infoFlag + "ElectricTrain constructor called");  
 }  
  
 @Override  
 public void TakeCargo(double capacity, String type) {  
 if (freightCar.getBookedCapacity() + capacity <= freightCar.getCapacity())  
 {  
 freightCar.setBookedCapacity(freightCar.getBookedCapacity() + capacity);  
 freightCar.setTypeOfCargo(type);  
 System.*out*.println("Locomotive take " + type + "; Now Locomotive have " + freightCar.getBookedCapacity() + " volume of cargo");  
 }  
 else  
 {  
 System.*out*.println("Locomotive do not have enough space");  
 }  
 logger.log(logger.infoFlag + "Locomotive TakeCargo method called");  
 }  
  
 @Override  
 public void TakeOffCargo(double capacity, String type) {  
 if (freightCar.getBookedCapacity() - capacity >= 0)  
 {  
 freightCar.setBookedCapacity(freightCar.getBookedCapacity() - capacity);  
 System.*out*.println("Locomotive take off " + type + "; Now Locomotive have " + freightCar.getBookedCapacity() + " volume of cargo");  
 }  
 else  
 {  
 System.*out*.println("Locomotive do not have enough space");  
 }  
 logger.log(logger.infoFlag + "Locomotive TakeOffCargo method called");  
 }  
  
 @Override  
 public void Refuel(double fuel) {  
 this.fuel += fuel;  
 System.*out*.println("Locomotive refueled. Now it has " + this.fuel + "fuel");  
 logger.log(logger.infoFlag + "Locomotive Refuel method called");  
  
 }  
  
 @Override  
 public String toString() {  
 return "ElectricTrain:" +  
 "\nfreightCar = " + freightCar +  
 "\nmotor = " + motor +  
 "\nwheel = " + wheel +  
 "\nfuel = " + fuel;  
 }  
}

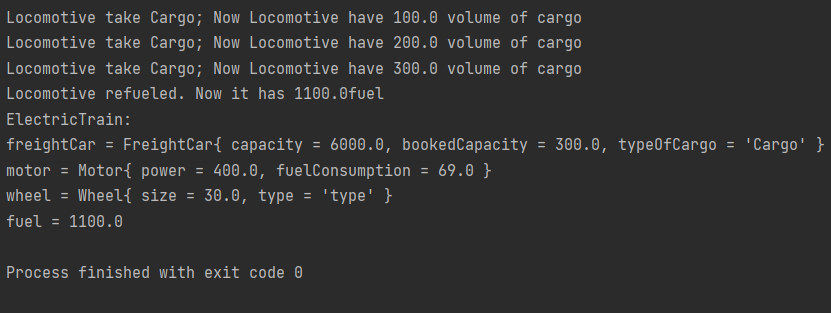
public class Main {  
 public static void main(String[] args) {  
 ElectricTrain electricTrain = new ElectricTrain(  
 new FreightCar(6000, 0, "type"),  
 new Motor(400, 69),  
 new Wheel(30, "type"),  
 1000  
 );  
  
 electricTrain.TakeCargo(100, "Cargo");  
 electricTrain.TakeCargo(100, "Cargo");  
 electricTrain.TakeCargo(100, "Cargo");  
 electricTrain.Refuel(100);  
  
 System.*out*.println(electricTrain);  
 }  
}

*/\*\*  
 \* Class Locomotive  
 \** ***@author*** *\** ***@version*** *1.0  
 \*/*public abstract class Locomotive  
{  
 protected FreightCar freightCar;  
 protected Motor motor;  
 protected Wheel wheel;  
 protected double fuel;  
 protected Logger logger = Logger.*getLogger*("logs.txt");  
  
 */\*\*  
 \* Constructor  
 \** ***@param*** *freightCar  
 \** ***@param*** *motor  
 \** ***@param*** *wheel  
 \*/* public Locomotive(FreightCar freightCar, Motor motor, Wheel wheel, double fuel) {  
 this.freightCar = freightCar;  
 this.motor = motor;  
 this.wheel = wheel;  
 this.fuel = fuel;  
 logger.log(logger.infoFlag + "Locomotive constructor called");  
 }  
  
 */\*\*  
 \* Take Cargo method  
 \** ***@param*** *capacity  
 \** ***@param*** *type  
 \*/* public abstract void TakeCargo(double capacity, String type);  
  
 */\*\*  
 \* TakeOff Cargo method  
 \** ***@param*** *capacity  
 \** ***@param*** *type  
 \*/* public abstract void TakeOffCargo(double capacity, String type);  
  
 public FreightCar getFreightCar() {  
 return freightCar;  
 }  
  
 public void setFreightCar(FreightCar freightCar) {  
 this.freightCar = freightCar;  
 }  
  
 public Motor getMotor() {  
 return motor;  
 }  
  
 public void setMotor(Motor motor) {  
 this.motor = motor;  
 }  
  
 public Wheel getWheel() {  
 return wheel;  
 }  
  
 public void setWheel(Wheel wheel) {  
 this.wheel = wheel;  
 }  
  
 public double getFuel() {  
 return fuel;  
 }  
  
 public void setFuel(double fuel) {  
 this.fuel = fuel;  
 }  
  
 @Override  
 public String toString() {  
 return "Locomotive:" +  
 "\nfreightCar = " + freightCar +  
 "\nmotor = " + motor +  
 "\nwheel = " + wheel +  
 "\nfuel = " + fuel;  
 }  
}

public class FreightCar  
{  
 private double capacity;  
 private double bookedCapacity;  
 private String typeOfCargo;  
  
 public FreightCar(double capacity, double bookedCapacity, String typeOfCargo) {  
 this.capacity = capacity;  
 this.bookedCapacity = bookedCapacity;  
 this.typeOfCargo = typeOfCargo;  
 }  
  
 public double getCapacity() {  
 return capacity;  
 }  
  
 public void setCapacity(double capacity) {  
 this.capacity = capacity;  
 }  
  
 public double getBookedCapacity() {  
 return bookedCapacity;  
 }  
  
 public void setBookedCapacity(double bookedCapacity) {  
 this.bookedCapacity = bookedCapacity;  
 }  
  
 public String getTypeOfCargo() {  
 return typeOfCargo;  
 }  
  
 public void setTypeOfCargo(String typeOfCargo) {  
 this.typeOfCargo = typeOfCargo;  
 }  
  
 @Override  
 public String toString() {  
 return "FreightCar{ " +  
 "capacity = " + capacity +  
 ", bookedCapacity = " + bookedCapacity +  
 ", typeOfCargo = '" + typeOfCargo + '\'' +  
 " }";  
 }  
}

public interface Refuel  
{  
 void Refuel(double fuel);  
}

**Результати:**

****

**Висновок:** ознайомився з спадкуванням та інтерфейсами у мові Java.