Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут комп'ютерних технологій, автоматики та метрології

Кафедра ЕОМ



**Звіт**

**До лабораторної роботи №3**

# З дисципліни: «Кросплатформні засоби програмування»

На тему «Основи розробки програм мовою Java»

# Варіант №30

Виконала: ст. гр. КІ-36

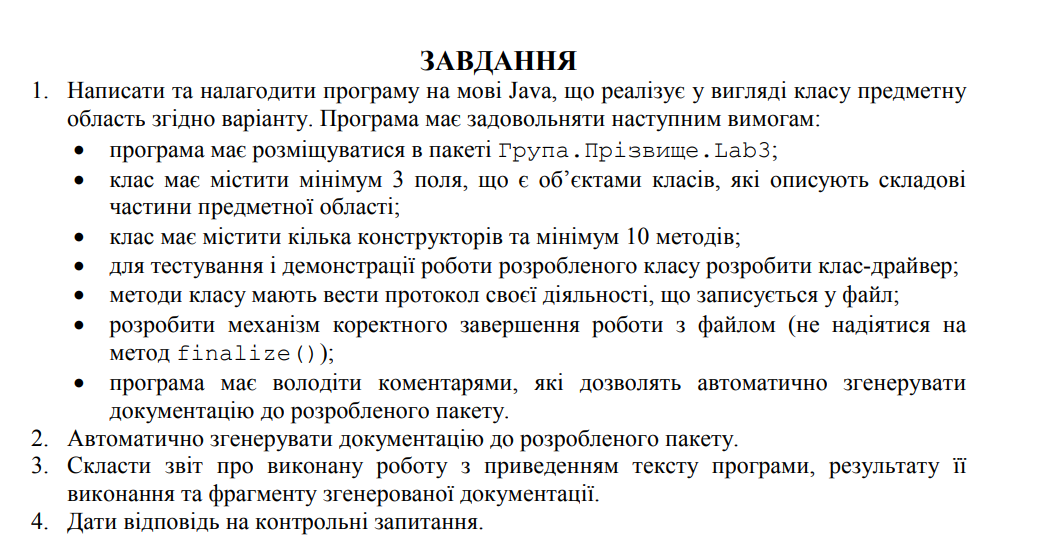
Завада В.С.

Прийняв:

Іванов Ю.С.

Львів – 2022

**Мета:** ознайомитися з процесом розробки класів та пакетів мовою Java.





**Хід роботи:**

**Лістинг прогами:**

public class Wheel  
{  
 private double size;  
 private String type;  
  
 public Wheel(double size, String type) {  
 this.size = size;  
 this.type = type;  
 }  
  
 public double getSize() {  
 return size;  
 }  
  
 public void setSize(double size) {  
 this.size = size;  
 }  
  
 public String getType() {  
 return type;  
 }  
  
 public void setType(String type) {  
 this.type = type;  
 }  
  
 @Override  
 public String toString() {  
 return "Wheel{ " +  
 "size = " + size +  
 ", type = '" + type + '\'' +  
 " }";  
 }  
}

public class Motor  
{  
 private double power;  
 private double fuelConsumption;  
  
 public Motor(double power, double fuelConsumption) {  
 this.power = power;  
 this.fuelConsumption = fuelConsumption;  
 }  
  
 public double getPower() {  
 return power;  
 }  
  
 public void setPower(double power) {  
 this.power = power;  
 }  
  
 public double getFuelConsumption() {  
 return fuelConsumption;  
 }  
  
 public void setFuelConsumption(double fuelConsumption) {  
 this.fuelConsumption = fuelConsumption;  
 }  
  
 @Override  
 public String toString() {  
 return "Motor{ " +  
 "power = " + power +  
 ", fuelConsumption = " + fuelConsumption +  
 " }";  
 }  
}

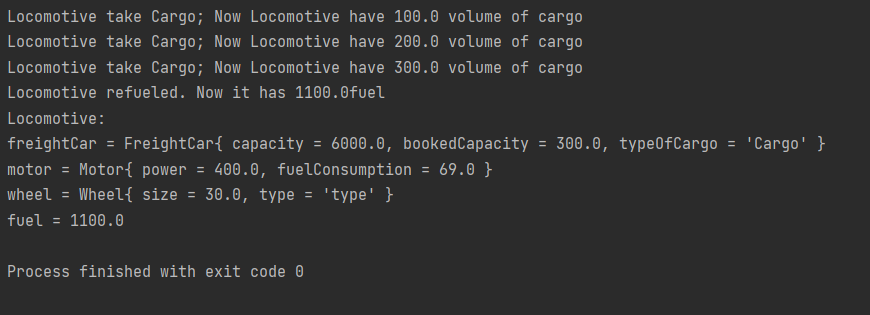
import java.io.\*;  
import java.text.SimpleDateFormat;  
import java.util.\*;  
  
*/\*\*  
 \* Class Logger. Was created to log information, errors and warnings. Also there was implemented Singelton  
 \** ***@author*** *\** ***@version*** *1.0  
 \*/*public class Logger  
{  
 private static Logger *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) {  
 Locomotive locomotive = new Locomotive(  
 new FreightCar(6000, 0, "type"),  
 new Motor(400, 69),  
 new Wheel(30, "type"),  
 1000  
 );  
  
 locomotive.TakeCargo(100, "Cargo");  
 locomotive.TakeCargo(100, "Cargo");  
 locomotive.TakeCargo(100, "Cargo");  
 locomotive.Refuel(100);  
  
 System.*out*.println(locomotive);  
 }  
}

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 + '\'' +  
 " }";  
 }  
}

*/\*\*  
 \* Class Locomotive  
 \** ***@author*** *\** ***@version*** *1.0  
 \*/*public class Locomotive  
{  
 private FreightCar freightCar;  
 private Motor motor;  
 private Wheel wheel;  
 private double fuel;  
 private 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 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");  
 }  
  
 */\*\*  
 \* TakeOff Cargo method  
 \** ***@param*** *capacity  
 \** ***@param*** *type  
 \*/* 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");  
 }  
  
 */\*\*  
 \* Method to refuel  
 \** ***@param*** *fuel  
 \*/* 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");  
 }  
  
 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;  
 }  
}

**Результат:**

****

**Висновок:** у ході данної лабораторної роботи я ознайомився з процесом розробки класів та пакетів мовою Java.