Университет ИТМО

Факультет ФПИ и КТ

Лабораторная работа №5

Вариант 314467

Выполнил :

Иванов Александр Алексеевич

группа P3115

Преподаватель:

Пашнин Александр Денисович

г. Санкт-Петербург

2020 год

**Задание :**

**Разработанная программа должна удовлетворять следующим требованиям:**

* Класс, коллекцией экземпляров которого управляет программа, должен реализовывать сортировку по умолчанию.
* Все требования к полям класса (указанные в виде комментариев) должны быть выполнены.
* Для хранения необходимо использовать коллекцию типа java.util.Vector
* При запуске приложения коллекция должна автоматически заполняться значениями из файла.
* Имя файла должно передаваться программе с помощью: **аргумент командной строки**.
* Данные должны храниться в файле в формате xml
* Чтение данных из файла необходимо реализовать с помощью класса java.util.Scanner
* Запись данных в файл необходимо реализовать с помощью класса java.io.PrintWriter
* Все классы в программе должны быть задокументированы в формате javadoc.
* Программа должна корректно работать с неправильными данными (ошибки пользовательского ввода, отсутсвие прав доступа к файлу и т.п.).

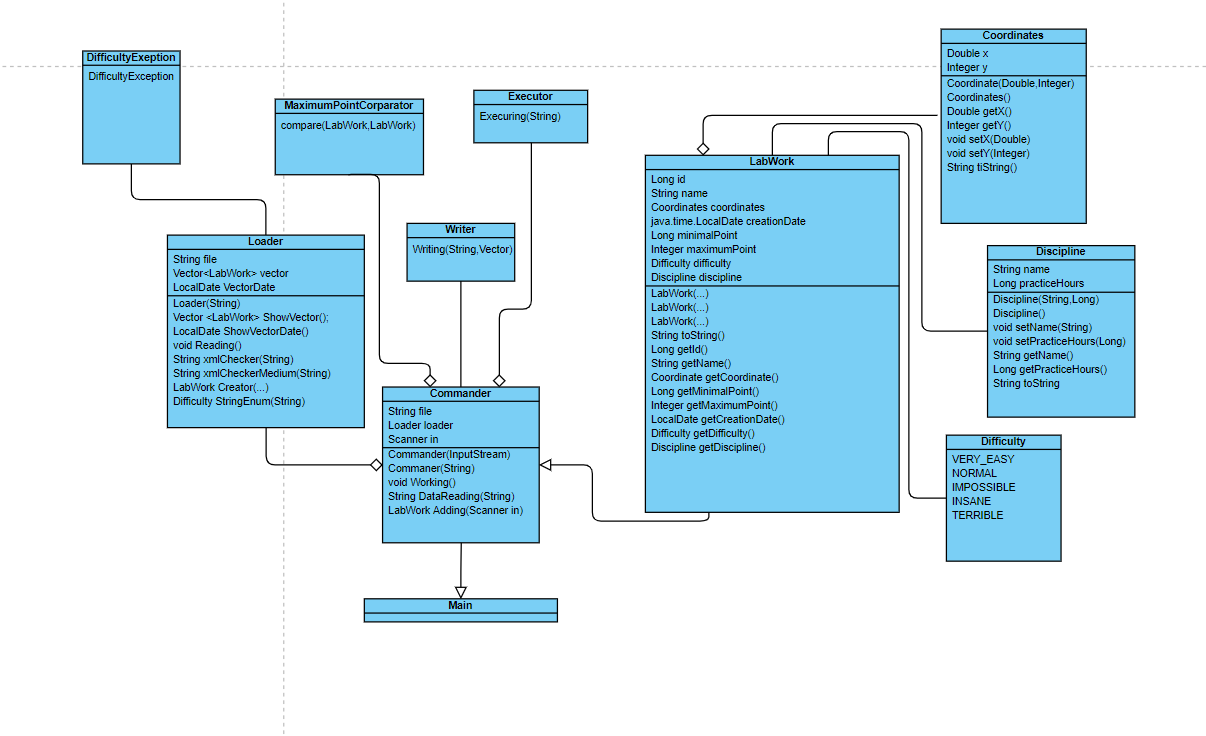
**В интерактивном режиме программа должна поддерживать выполнение следующих команд:**

* help : вывести справку по доступным командам
* info : вывести в стандартный поток вывода информацию о коллекции (тип, дата инициализации, количество элементов и т.д.)
* show : вывести в стандартный поток вывода все элементы коллекции в строковом представлении
* add {element} : добавить новый элемент в коллекцию
* update id {element} : обновить значение элемента коллекции, id которого равен заданному
* remove\_by\_id id : удалить элемент из коллекции по его id
* clear : очистить коллекцию
* save : сохранить коллекцию в файл
* execute\_script file\_name : считать и исполнить скрипт из указанного файла. В скрипте содержатся команды в таком же виде, в котором их вводит пользователь в интерактивном режиме.
* exit : завершить программу (без сохранения в файл)
* remove\_first : удалить первый элемент из коллекции
* add\_if\_max {element} : добавить новый элемент в коллекцию, если его значение превышает значение наибольшего элемента этой коллекции
* sort : отсортировать коллекцию в естественном порядке
* remove\_all\_by\_difficulty difficulty : удалить из коллекции все элементы, значение поля difficulty которого эквивалентно заданному
* filter\_by\_difficulty difficulty : вывести элементы, значение поля difficulty которых равно заданному
* filter\_starts\_with\_name name : вывести элементы, значение поля name которых начинается с заданной подстроки

**Формат ввода команд:**

* Все аргументы команды, являющиеся стандартными типами данных (примитивные типы, классы-оболочки, String, классы для хранения дат), должны вводиться в той же строке, что и имя команды.
* Все составные типы данных (объекты классов, хранящиеся в коллекции) должны вводиться по одному полю в строку.
* При вводе составных типов данных пользователю должно показываться приглашение к вводу, содержащее имя поля (например, "Введите дату рождения:")
* Если поле является enum'ом, то вводится имя одной из его констант (при этом список констант должен быть предварительно выведен).
* При некорректном пользовательском вводе (введена строка, не являющаяся именем константы в enum'е; введена строка вместо числа; введённое число не входит в указанные границы и т.п.) должно быть показано сообщение об ошибке и предложено повторить ввод поля.
* Для ввода значений null использовать пустую строку.
* Поля с комментарием "Значение этого поля должно генерироваться автоматически" не должны вводиться пользователем вручную при добавлении.

Диаграмма:



Код программы :

Main.java

import java.lang.\*;

/\*\*

\* {@author Ivanov Alexander P3115

\* \*/

/\*\*

\* {@param Main class which start read information from System.in

\*/

public class Main {

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

Commander commander = new Commander(System.in);

commander.Working();

}

}

import java.lang.\*;

import java.time.LocalDate;

/\*\*

\* {@param class of which vector consist

\*/

public class LabWork {

private Long id; //Поле не может быть null, Значение поля должно быть больше 0, Значение этого поля должно быть уникальным, Значение этого поля должно генерироваться автоматически

private String name; //Поле не может быть null, Строка не может быть пустой

private Coordinates coordinates; //Поле не может быть null

private java.time.LocalDate creationDate; //Поле не может быть null, Значение этого поля должно генерироваться автоматически

private Long minimalPoint; //Поле не может быть null, Значение поля должно быть больше 0

private Integer maximumPoint; //Поле не может быть null, Значение поля должно быть больше 0

private Difficulty difficulty; //Поле не может быть null

private Discipline discipline; //Поле может быть null

public LabWork(Long id , String name

, Coordinates coordinates , java.time.LocalDate creationDate,

Long minimalPoint , Integer maximumPoint, Difficulty difficulty , Discipline discipline){

this.id = id;

this.name =name;

this.coordinates = coordinates;

this.creationDate =creationDate;

this.minimalPoint =minimalPoint;

this.maximumPoint = maximumPoint;

this.difficulty = difficulty;

this.discipline =discipline;

}

public LabWork(Long time, Object mass0, Coordinates coordinates, Object mass3, Object mass4, Difficulty difficulty, Discipline discipline) {

this.id = time;

this.name = (String) mass0;

this.coordinates = coordinates;

this.creationDate = (java.time.LocalDate.now());

this.minimalPoint = (long)mass3;

this.maximumPoint = (Integer) mass4;

this.difficulty = difficulty;

this.discipline = discipline;

}

public LabWork(LabWork labWork, Long id) {

this.id = id;

this.name = labWork.getName();;

this.coordinates = labWork.getCoordinates();

this.discipline = labWork.getDiscipline();

this.difficulty = labWork.getDifficulty();

this.maximumPoint = labWork.maximumPoint;

this.minimalPoint = labWork.minimalPoint;

}

@Override

public String toString() {

return "LabWork{" +

"id = " + id +

", name = " + name +

", coordinates = " + coordinates +

", creationDate = " + creationDate +

", minimalPoint = " + minimalPoint +

", maximumPoint = " + maximumPoint +

", difficulty = " + difficulty +

", discipline = " + discipline.toString() +

'}';

}

public LabWork(){}

public Long getId(){return id;}

public String getName(){return name;}

public Coordinates getCoordinates() {

return coordinates;

}

public Long getMinimalPoint() {

return minimalPoint;

}

public Integer getMaximumPoint() {

return maximumPoint;

}

public LocalDate getCreationDate() {

return creationDate;

}

public Difficulty getDifficulty() {

return difficulty;

}

public Discipline getDiscipline() {

return discipline;

}

public void setId(Long id) {

this.id = id;

}

}

import java.io.IOException;

import java.nio.file.NoSuchFileException;

import java.nio.file.Paths;

import java.time.LocalDate;

import java.util.Scanner;

import java.util.Vector;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

/\*\*

\* {@param class works with Vector : loading from File, transform and contain

\*/

public class Loader implements Reading {

private String file;

private Vector <LabWork> vector = new Vector<LabWork>(0);

private java.time.LocalDate VectorDate;

public Loader(String file){

this.file = file;

}

// вывод вектора в формате строки

public Vector<LabWork> ShowVector(){

return vector;

}

/\*\*

\* {@return Vector which was read from File

\*/

public java.time.LocalDate ShowVectorDate(){

return VectorDate;

}

/\*\*

\* {@return time when Vector was read

\*/

@Override

// чтение данных из файла data.xml

public void Reading() throws IOException {

try {

Scanner scanner = new Scanner(Paths.get(file));

Long id = 0L;

Difficulty difficulty = Difficulty.VERY\_EASY;

java.time.LocalDate creationDate = LocalDate.now();

String name = "";

Long minimalPoint = 0L;

Integer maximumPoint = 0;

Discipline discipline = new Discipline();

Coordinates coordinates = new Coordinates(0.0, 0);

if (scanner.hasNext()) {

String line = scanner.nextLine();

while (!line.contains("LabWork") && scanner.hasNext()) {

if (line.contains("<VectorDate>")) {

VectorDate = LocalDate.parse(xmlCheckerMedium(line));

}

line = scanner.nextLine();

}

while (scanner.hasNext() && !line.contains("</LabWork>")) {

line = scanner.nextLine();

name = "";

minimalPoint = 0L;

maximumPoint = 0;

discipline = new Discipline();

coordinates = new Coordinates(0.0, 0);

if (!line.contains("</LabWork>")) {

id++;

String WorkName = xmlChecker(line);

//java.lang.System.out.println(" WorkName =" + WorkName);

while (scanner.hasNext() && !line.contains("/" + WorkName)) {

line = scanner.nextLine();

line = line.replace(" ", "");

String data = xmlChecker(line);

//java.lang.System.out.println(" line =" + line);

//java.lang.System.out.println(" data =" + data);

switch (data) {

case "creationDate":

creationDate = LocalDate.parse(xmlCheckerMedium(line));

break;

case "name": {

name = xmlCheckerMedium(line);

//java.lang.System.out.println("name =" + name);

}

break;

case "coordinates": {

while (!line.contains("/coordinates")) {

if (line.contains("x")) {

line = xmlCheckerMedium(line);

//java.lang.System.out.println(line);

coordinates.setX(Double.parseDouble(line));

}

if (line.contains("y")) {

line = xmlCheckerMedium(line);

coordinates.setY(Integer.parseInt(line));

//java.lang.System.out.println(line);

}

line = scanner.nextLine();

}

}

break;

case "minimalPoint": {

line = xmlCheckerMedium(line);

minimalPoint = Long.parseLong(line);

//java.lang.System.out.println(line);

}

break;

case "maximumPoint": {

line = xmlCheckerMedium(line);

maximumPoint = Integer.parseInt(line);

//java.lang.System.out.println(line);

}

break;

case "difficulty": {

switch (xmlCheckerMedium(line)) {

case "VERY\_EASY":

difficulty = Difficulty.VERY\_EASY;

break;

case "NORMAL":

difficulty = Difficulty.NORMAL;

break;

case "IMPOSSIBLE":

difficulty = Difficulty.IMPOSSIBLE;

break;

case "INSANE":

difficulty = Difficulty.INSANE;

break;

case "TERRIBLE":

difficulty = Difficulty.TERRIBLE;

break;

}

//java.lang.System.out.println("difficulty =" + difficulty);

}

break;

case "discipline": {

while (!line.contains("/discipline")) {

if (line.contains("name")) {

discipline.setName(xmlCheckerMedium(line));

//java.lang.System.out.println(discipline.getName());

}

if (line.contains("practiceHours")) {

line = xmlCheckerMedium(line);

discipline.setPracticeHours(Long.parseLong(line));

//java.lang.System.out.println(discipline.getPracticeHours());

}

line = scanner.nextLine();

}

}

break;

default:

}

}

vector.add(Creator(id, name, coordinates, LocalDate.now(), minimalPoint, maximumPoint, difficulty, discipline));

}

}

}

} catch (NoSuchFileException e) {

System.out.println("Не обнаружен файл для хранения data.xml");

}

}

// нахождение первого слова

@Override

public String xmlChecker(String line) {

Pattern pattern = Pattern.compile("<.+?>");

Matcher matcher = pattern.matcher(line);

if (matcher.find()) {

line = line.substring(matcher.start(), matcher.end());

}

line =line.replace("<","");

line =line.replace(">","");

return line;

/\*\*

\* {@return return first word from line

\*/

}

// нахождение второго слова

@Override

public String xmlCheckerMedium(String line) {

Pattern pattern = Pattern.compile(">.+?<");

Matcher matcher = pattern.matcher(line);

if (matcher.find()) {

line = line.substring(matcher.start(), matcher.end());

}

line =line.replace("<","");

line =line.replace(">","");

return line;

/\*\*

\* {@return return second word from line

\*/

}

// создание Vector по данным из файла

@Override

public LabWork Creator(Long id, String name, Coordinates coordinates, LocalDate creationDate, Long minimalPoint, Integer maximumPoint, Difficulty difficulty, Discipline discipline) {

return new LabWork(id,name,coordinates,creationDate,minimalPoint,maximumPoint,difficulty,discipline);

/\*\*

\* {@return LabWork transformed from data

\*/

}

public Difficulty StringtoEnum(String line) throws DifficultyExeption {

Difficulty difficulty =Difficulty.VERY\_EASY;

switch (line) {

case "VERY\_EASY":

break;

case "NORMAL":

difficulty = Difficulty.NORMAL;

break;

case "IMPOSSIBLE":

difficulty = Difficulty.IMPOSSIBLE;

break;

case "INSANE":

difficulty = Difficulty.INSANE;

break;

case "TERRIBLE":

difficulty = Difficulty.TERRIBLE;

break;

default:{

throw new DifficultyExeption();

}

}

return difficulty;

/\*\*

\* {@return Difficulty transformed from String

\*/

}

}

import java.io.IOException;

import java.io.InputStream;

import java.io.PrintWriter;

import java.nio.file.NoSuchFileException;

import java.nio.file.Paths;

import java.util.Scanner;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

/\*\*

\* {@param Commander which works with commands from InputSystem and Files

\*/

public class Commander implements Working{

final String file = "data.xml";

final Loader loader = new Loader(file);

Scanner in = new Scanner(System.in);

public Commander(InputStream system){

in = new Scanner(system);

}

public Commander(String file) throws IOException {

in = new Scanner(Paths.get(file));

}

@Override

public void Working() throws Exception {

System.out.println("ВВедите команду ( команда help - справка по командам ) :");

loader.Reading();

String command = in.next();

while(!command.equals("exit")) {

switch (command) {

case "help" :

{

PrintWriter writer = new PrintWriter(System.out);

writer.println(

"help : вывести справку по доступным командам\n" +

"info : вывести в стандартный поток вывода информацию о коллекции (тип, дата инициализации, количество элементов и т.д.)\n" +

"show : вывести в стандартный поток вывода все элементы коллекции в строковом представлении\n" +

"add {element} : добавить новый элемент в коллекцию\n" +

"update id {element} : обновить значение элемента коллекции, id которого равен заданному\n" +

"remove\_by\_id id : удалить элемент из коллекции по его id\n" +

"clear : очистить коллекцию\n" +

"save : сохранить коллекцию в файл\n" +

"execute\_script file\_name : считать и исполнить скрипт из указанного файла. В скрипте содержатся команды в таком же виде, в котором их вводит пользователь в интерактивном режиме.\n" +

"exit : завершить программу (без сохранения в файл)\n" +

"remove\_first : удалить первый элемент из коллекции\n" +

"add\_if\_max {element} : добавить новый элемент в коллекцию, если его значение превышает значение наибольшего элемента этой коллекции\n" +

"sort : отсортировать коллекцию в естественном порядке\n" +

"remove\_all\_by\_difficulty difficulty : удалить из коллекции все элементы, значение поля difficulty которого эквивалентно заданному\n" +

"filter\_by\_difficulty difficulty : вывести элементы, значение поля difficulty которых равно заданному\n" +

"filter\_starts\_with\_name name : вывести элементы, значение поля name которых начинается с заданной подстроки\n");

writer.flush();

}

break;

case "info" :{

PrintWriter writer = new PrintWriter(System.out,true);

writer.println(

"Type : Vector; \n"+" Initialization date : "+loader.ShowVectorDate()+"\n"+

"Elements :"+loader.ShowVector().capacity()+"\n"+

"Type: LabWork;\n"+

"LabWork's data:" +

" private Long id; \n" +

" private String name; \n" +

" private Coordinates coordinates; \n" +

" private java.time.LocalDate creationDate; \n" +

" private Long minimalPoint; \n" +

" private Integer maximumPoint; \n" +

" private Difficulty difficulty; \n" +

" private Discipline discipline; \n" +

"LabWork's Methods :"+

" toString();\n" +

" LabWork();\n" +

" getId();\n" +

" getName();\n" +

" getCoordinates();\n" +

" getMinimalPoint(); \n" +

" getMaximumPoint(); \n" +

" getCreationDate();\n" +

" getDifficulty(); \n" +

" getDiscipline(); \n"+

"Type: Difficulty;\n"+

" VERY\_EASY;\n"+

" NORMAL;\n"+

" IMPOSSIBLE;\n"+

" INSANE;\n"+

" TERRIBLE;"

);

writer.flush();

}

break;

case "show" :{

PrintWriter a = new PrintWriter(System.out);

a.println(loader.ShowVector().toString());

a.flush();}

break;

case "add" :{

try{loader.ShowVector().add(Adding(in));}catch (NumberFormatException e){

System.out.println("ВВеденное значение не соответствует формату");

}

}

break;

case "update" :{

command = in.next();

int index = Integer.parseInt(command);

loader.ShowVector().remove(index);

try{loader.ShowVector().insertElementAt(new LabWork(Adding(in), (long) index),index);}

catch (NumberFormatException e){

System.out.println("ВВеденное значение не соответствует формату");

}

}

break;

case "remove\_by\_id" :{

try{command = in.next();

Long index = Long.parseLong(command);

for (int i = 0 ; i <loader.ShowVector().capacity();++i) {

if(loader.ShowVector().get(i).getId().equals(index)){

loader.ShowVector().remove(i);

}

}}catch (NumberFormatException e){

System.out.println("После команды remove\_by\_id необходимо ввести число с типом Long");

}

}

break;

case "clear" :{

loader.ShowVector().clear();

PrintWriter a = new PrintWriter("data.xml");

a.write("");

a.flush();

loader.ShowVector().removeAll(loader.ShowVector());

}

break;

case "save" :{

Writer writer = new Writer();

writer.Writing("data.xml", loader.ShowVector());

}

break;

case "execute\_script" :{

try{

command = in.next();

Executer executer = new Executer();

executer.Executing(command);

} catch(NoSuchFileException e){

System.out.println("Файл не найден или не может быть открыт");

}

}

break;

case "remove\_first" :{

loader.ShowVector().remove(0);

}

break;

case "add\_if\_max" :{

try{

try{LabWork labWork = Adding(in);

int MAX = 0;

for(int i = 0 ; i < loader.ShowVector().capacity();++i){

if(loader.ShowVector().get(i).getMaximumPoint()>MAX){

MAX = loader.ShowVector().get(i).getMaximumPoint();

}

}

if(labWork.getMaximumPoint()>MAX){

loader.ShowVector().add(labWork);

}}

catch (NumberFormatException e){

System.out.println("Введенный LabWork не может быть создан");

}}

catch (NumberFormatException a){

System.out.println("ВВеденное значение не соответствует формату");

}

}

break;

case "sort" :{

loader.ShowVector().sort(new MaximumPointComparator());

}

break;

case "remove\_all\_by\_difficulty" :{

try{command =in.next();

PrintWriter writer = new PrintWriter(System.out,true);

for(int i = 0 ; i < loader.ShowVector().capacity();++i){

if(loader.ShowVector().get(i).getDifficulty() == loader.StringtoEnum(command)){

loader.ShowVector().remove(loader.ShowVector().get(i));

}

}

writer.flush();}

catch (DifficultyExeption e){

System.out.println("Неправильное значение difficulty");

}

}

break;

case "filter\_by\_difficulty" :{

try{

command =in.next();

PrintWriter writer = new PrintWriter(System.out,true);

for(int i = 0 ; i < loader.ShowVector().capacity();++i){

if(loader.ShowVector().get(i).getDifficulty() == loader.StringtoEnum(command)){

writer.println(loader.ShowVector().get(i));

}

}

writer.flush();}

catch (DifficultyExeption e){

System.out.println("Неправильное значение difficulty");

}

}

break;

case "filter\_starts\_with\_name" :{

command =in.next();

PrintWriter writer = new PrintWriter(System.out,true);

for(int i = 0 ; i < loader.ShowVector().capacity();++i){

if(loader.ShowVector().get(i).getName().contains(command)){

writer.println(loader.ShowVector().get(i));

}

}

writer.flush();

}

break;

default:{

System.out.println("Такой команды не существует, обратитесь к команде help для вывода справки по командам");

}

}

command = in.next();

}

in.close();

}

public String DataReading(String line) {

Pattern pattern = Pattern.compile(".+?,");

Matcher matcher = pattern.matcher(line);

if (matcher.find()) {

line = line.substring(matcher.start(), matcher.end());

}

line =line.replace(",","");

line =line.replace(" ","");

return line;

}

public LabWork Adding(Scanner in){

String command;

Object[] mass = new Object[8];

mass[1] = (double) 1;

for(int i = 0 ; i < 8 ; ++i ){

command = in.next();

if(command.equals(",")){i--;}

else {

command = command.replace(",", "");

command = command.replace(" ", "");

if(i == 0){

mass[0] = command;

}

if(i == 1){

mass[1] = Double.parseDouble(command);

}

if(i == 2){

mass[2] = Integer.parseInt(command);

}

if(i == 3){

mass[3] = Long.parseLong(command);

}

if(i == 4){

mass[4] = Integer.parseInt(command);

}

if(i == 5){

mass[5] = command;

}

if(i == 6){

mass[6] = command;

}

if(i == 7){

mass[7] = Long.parseLong(command);

}

}

}

try{Discipline discipline = new Discipline((String)mass[6],(Long)mass[7]);

Difficulty difficulty = loader.StringtoEnum((String) mass[5]);

Coordinates coordinates = new Coordinates((Double)mass[1],(Integer) mass[2]);

return new LabWork((long) (loader.ShowVector().capacity() + 1),mass[0],coordinates, mass[3],(Integer)mass[4],

difficulty,discipline);}

catch (DifficultyExeption e){

System.out.println("ВВеденное значение difficulty не является Enum и заменено на VERY\_EAST");

}

Coordinates coordinates = new Coordinates((Double)mass[1],(Integer) mass[2]);

Discipline discipline = new Discipline((String)mass[6],(Long)mass[7]);

return new LabWork((long) (loader.ShowVector().capacity() + 1),mass[0],coordinates, mass[3],(Integer)mass[4],

Difficulty.VERY\_EASY,discipline);

/\*\*

\* {@return class LabWork which assembled by string data

\*/

}

}

import java.util.Comparator;

/\*\*

\* {@param Comparator class which compairs LabWork's maximumPoints

\*/

class MaximumPointComparator implements Comparator<LabWork>{

public int compare(LabWork labWork1, LabWork labWork2){

if(labWork1.getMaximumPoint() == labWork2.getMaximumPoint()) return 0;

if(labWork1.getMaximumPoint() > labWork2.getMaximumPoint()) return 1;

else return -1;

/\*\*

\* {@return 0,1 or -1 to manage the compair between LabWork's pair

\*/

}

}

/\*\*

\* {@param class which is a part of LabWork class

\*/

public class Coordinates {

private Double x = 0.0; //Поле не может быть null

private Integer y = 1; //Значение поля должно быть больше -957, Поле не может быть null

public Coordinates(Double x, Integer y){

this.x = x;

this.y = y;

}

public Coordinates(){

this.x = 0.0;

this.y = 0;

}

public Double getX(){return this.x;}

public Integer getY(){return this.y;}

public void setX(Double x){this.x = x;}

public void setY(Integer y){this.y =y;}

@Override

public java.lang.String toString() {

return "Coordinates{" +

"x=" + x +

", y=" + y +

'}';

}

}

/\*\*

\* {@param enum which is a part of LabWork class

\*/

public enum Difficulty {

VERY\_EASY,

NORMAL,

IMPOSSIBLE,

INSANE,

TERRIBLE;

}

import java.lang.\*;

/\*\*

\* {@param enum which is a part of LabWork class

\*/

public class Discipline<practiceHours> {

private String name; //Поле не может быть null, Строка не может быть пустой

private Long practiceHours; //Поле не может быть null

public Discipline(String name, Long practiceHours){

this.name = name;

this.practiceHours = practiceHours;

}

public Discipline(){}

void setName(String name){this.name =name;}

void setPracticeHours(Long practiceHours){this.practiceHours = practiceHours; }

String getName(){return name;}

Long getPracticeHours(){return practiceHours;}

@Override

public String toString() {

return "Discipline{" +

"name = " + name +

", practiceHours = " + practiceHours +

'}';

}

}

import java.io.IOException;

/\*\*

\* {@param class which execute commands from File by Commander class

\*/

public class Executer implements Executing {

@Override

public void Executing(String file) throws Exception {

Commander commander = new Commander(file);

commander.Working();

}

}

import java.io.IOException;

public interface Executing {

void Executing(String file) throws Exception;

}

/\*\*

\* {@param Exception class which control Difficulty data

\*/

class DifficultyExeption extends Exception{

public DifficultyExeption() {

super("ВВеденное значение difficulty не является Enum и заменено на VERY\_EAST");

}

}

import java.io.IOException;

import java.util.Scanner;

interface Reading {

void Reading() throws IOException;

String xmlChecker(String file);

String xmlCheckerMedium(String file);

LabWork Creator(Long id , String name

, Coordinates coordinates , java.time.LocalDate creationDate,

Long minimalPoint , Integer maximumPoint, Difficulty difficulty , Discipline discipline);

}

import java.io.IOException;

public interface Working {

void Working() throws Exception;

}

import java.io.FileNotFoundException;

import java.io.OutputStream;

import java.io.PrintWriter;

import java.nio.file.NoSuchFileException;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.time.LocalDate;

import java.util.Vector;

/\*\*

\* {@param class which write vector to File by PrintWriter

\*/

public class Writer implements Writing{

@Override

public void Writing(String file, Vector vector) {

try {

PrintWriter writer = new PrintWriter(file);

writer.println("<?xml version=\"1.0\"?>");

writer.println("\t<VectorDate>" + LocalDate.now() + "</VectorDate>");

writer.println("\t<LabWork>");

for (int i = 0; i < vector.capacity(); ++i) {

LabWork a = (LabWork) vector.get(i);

writer.println("\t\t<ElementNumber" + a.getId() + ">");

writer.println("\t\t\t<id>" + a.getId() + "</id>");

writer.println("\t\t\t<name>" + a.getName() + "</name>");

writer.println("\t\t\t<coordinates>\n\t\t\t\t<x>" + a.getCoordinates().getX() + "</x>\n\t\t\t\t<y>" +

a.getCoordinates().getY() + "</y>\n" + "\t\t\t</coordinates>");

writer.println("\t\t\t<creationDate>" + a.getCreationDate() + "</creationDate>");

writer.println("\t\t\t<minimalPoint>" + a.getMinimalPoint() + "</minimalPoint>");

writer.println("\t\t\t<maximumPoint>" + a.getMaximumPoint() + "</maximumPoint>");

writer.println("\t\t\t<difficulty>" + a.getDifficulty() + "</difficulty>");

writer.println("\t\t\t<discipline>\n" + "\t\t\t\t<practiseHours>" + a.getDiscipline().getPracticeHours() + "</practiseHours>\n"

+ "\t\t\t\t<name>" + a.getDiscipline().getName() + "</name>\n" + "\t\t\t</discipline>");

writer.println("\t\t</ElementNumber" + a.getId() + ">");

}

writer.println("\t</LabWork>");

writer.close();

} catch (FileNotFoundException e){

System.out.println("Файл для записи data.xml не обнаружен и был создан");

}

}

}

import java.io.FileNotFoundException;

import java.util.Vector;

public interface Writing {

void Writing(String file, Vector vector) throws FileNotFoundException;

}