МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

“ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

КАФЕДРА ПРОГРАМНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГИЙ УПРАВЛІННЯ

**Лабораторна робота 2**

Робота з винятками і файлами в Java

Виконала :

студентка групи Кн- 36б

Ликова Маргарита

Перевірили :

Кизилов О.С.

Козуля М.М.

Харків 2017

**Мета :** Покращити знання з JAVA, навчитися працювати з винятками в Java .

Хід роботи :

### 1.1 Індивідуальне завдання

Спроектувати та реалізувати класи для представлення сутностей [попередньої лабораторної роботи](http://iwanoff.96.lt/oop_kn/LabTraining01.html). Слід створити два похідних класи від класу, який представляє основну сутність. Один клас повинен бути доповненим можливостями читання даних з відповідно підготовленого текстового файлу та запису цих даних в інший файл після сортування. Другий клас повинен реалізовувати читання даних з XML-документу, зберігати дані в структурах, які автоматично створюються за допомогою технології зв'язування даних, та запис даних в інший XML-документ після сортування.

Окрім роботи з файлами повинно бути реалізоване виведення результатів у консольне вікно.

package seclab.f.ex;

import javax.xml.bind.annotation.XmlElement;

import javax.xml.bind.annotation.XmlRootElement;

public abstract class Doc {

private String name;

private String spec;

public Doc() { }

public Doc(String name, String spec) {

this.name = name; this.spec = spec; }

public String getName() {

return name; }

public void setName(String name) {

this.name = name; } public String getSpec() {

return spec; }

public void setSpec(String spec) {

this.spec = spec; }

abstract public Recept minOfVisitors(); abstract public int sumOfVisitors(); abstract public void sortByVisitors(); abstract public void sortByDay();}

**public** **class** Recept {

**private** **int** numOfVisit;

**private** String day;

**private** String shift;

**public** Recept() {}

**public** Recept(**int** numOfVisit, String day,String shift) {

**this**.numOfVisit = numOfVisit;

**this**.day = day;

**this**.shift = shift;

}

**public** **void** setDay(String day) {

**this**.day = day;

}

**public** **void** setNumOfVisit(**int** numOfVisit) {

**this**.numOfVisit = numOfVisit;

}

**public** **void** setShift(String shift) {

**this**.shift = shift;

}

**public** String getDay() {

**return** day;

}

**public** **int** getNumOfVisit() {

**return** numOfVisit;

}

**public** String getShift() {

**return** shift;

}

@Override

**public** String toString() {

**return** day + " , " + shift+","+numOfVisit;

}}

**public** **class** DocArr **extends** Doc {

**private** Recept[] recepts;

**public** DocArr() {}

**public** DocArr(String name,String spec, Recept[] recepts) {

**super**(name, spec);

**this**.recepts = recepts; }

**public** **void** setRecepts(Recept[] recepts) {

**this**.recepts = recepts; }

**public** Recept[] getRecepts() {

**return** recepts; }

@Override

**public** Recept minOfVisitors() {

Recept min = recepts[0];

**for** (Recept h : recepts) {

**if** (h.getNumOfVisit() < min.getNumOfVisit())

min = h;

}

**return** min;}

@Override

**public** **int** sumOfVisitors() {

**int** sum = 0;

**for** (Recept rec : recepts) {

sum += rec.getNumOfVisit();

}

**return** sum;}

@Override

**public** **void** sortByVisitors() {

Arrays.*sort*(recepts, (a, b) -> Integer.*compare*(a.getNumOfVisit(), b.getNumOfVisit()));}

@Override

**public** **void** sortByDay() {

Arrays.*sort*(recepts, (a, b) -> a.getDay().compareTo(b.getDay()));

}

@Override

**public** String toString() {

**return** "Doctor " + getName() + " " + getSpec() + "\n" + Arrays.*asList*(recepts);

}}

**public** **class** DocArrTXT **extends** DocArr {

**public** DocArrTXT() {

}

**public** DocArrTXT(String name, String spec, Recept[] recepts) {

**super**(name, spec, recepts);

}

**public** **void** toTxt(String name) {

**try** (FileWriter fReader = **new** FileWriter(name)) { fReader.write(**this**.getName() + "\n"); fReader.write(**this**.getSpec() + "\n");

**for** (Recept rec : getRecepts()) { fReader.write(rec.toString() + "\n");}

} **catch** (Exception e) {}

}

**public** DocArrTXT fromTxt() {

DocArrTXT sArr = **new** DocArrTXT();

**try** (Scanner scanner = **new** Scanner(**new** FileReader("Doctor.txt"))) { sArr.setName(scanner.nextLine()); sArr.setSpec(scanner.nextLine());

List<Recept> sList = **new** ArrayList<>(); **while** (scanner.hasNext()) { String[] string = scanner.nextLine().split(","); Integer integer = Integer.*parseInt*(string[2].replaceAll(" ", "")); sList.add(**new** Recept(integer, string[0], string[1]));} sArr.setRecepts(sList.toArray(**new** Recept[sList.size()]));

} **catch** (Exception e) {

e.printStackTrace();

}

**return** sArr;

}

}

**public** **class** DocList **extends** Doc{

**private** List<Recept> list;

**public** DocList() {

}

**public** DocList(String name, String spec, List<Recept> list) {

**super**(name, spec);

**this**.list = list;

}

**public** List<Recept> getList() {

**return** list;

}

**public** **void** setList(List<Recept> list) {

**this**.list = list;

}

@Override

**public** Recept minOfVisitors() {

Recept h = list.get(0);

**for** (Recept spec : list) {

**if** (h.getNumOfVisit() > spec.getNumOfVisit())

h = spec;}

**return** h;}

@Override

**public** **int** sumOfVisitors() {

**int** sum = 0;

**for** (Recept spec : list) {

sum += spec.getNumOfVisit();}

**return** sum;}

@Override

**public** **void** sortByVisitors() {

Collections.*sort*(list, (a, b) -> Integer.*compare*(a.getNumOfVisit(), b.getNumOfVisit()));}

@Override

**public** **void** sortByDay() {

Collections.*sort*(list, (a, b) -> a.getDay().compareTo(b.getDay()));

}

@Override

**public** String toString() {

**return** "Doctor " + getName() + " " + getSpec() + "\n" + Arrays.*asList*(list.toArray());

}}

@XmlRootElement(name= "Doctor")

**public** **class** DocListXML **extends** DocList{

**public** DocListXML() {

}

**public** DocListXML(String name, String spec, List<Recept> list) {

**super**(name, spec, list);

}

**public** DocListXML fromXml() {

**try** {

JAXBContext jaxbContext=JAXBContext.*newInstance*(DocListXML.**class**);

Unmarshaller unmarshaller=jaxbContext.createUnmarshaller();

DocListXML sXml=(DocListXML) unmarshaller.unmarshal(**new** File("Doctor.xml"));

**return** sXml;

} **catch** (Exception e) {

e.printStackTrace();

}

**return** **null**;

}

**public** **void** toXml(String name) {

**try** {

JAXBContext jaxbContext = JAXBContext.*newInstance*(DocListXML.**class**);

Marshaller marshaller = jaxbContext.createMarshaller(); marshaller.setProperty(Marshaller.***JAXB\_FORMATTED\_OUTPUT***, Boolean.***TRUE***);

marshaller.marshal(**this**, **new** FileWriter(name));

} **catch** (Exception e) {

e.printStackTrace();

}}}

**public** **class** Test {

**public** **static** **void** main(String[] Args) {

DocArrTXT sArr = **new** DocArrTXT();

sArr = sArr.fromTxt();

sArr.sortByDay();

sArr.toTxt("sortbyDay.txt");

sArr.sortByVisitors();

sArr.toTxt("sortbyVisitors.txt");

DocListXML list = **new** DocListXML();

list = list.fromXml();

list.toXml("Doctor.xml");

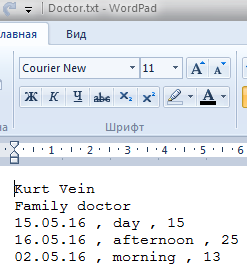
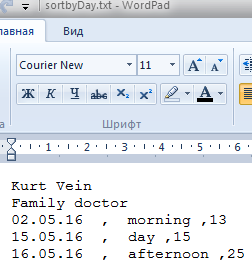
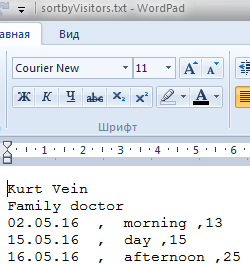
list.sortByDay();

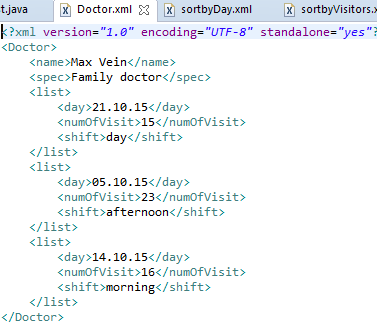
list.toXml("sortbyDay.xml");

list.sortByVisitors();

list.toXml("sortbyVisitors.xml");

}}

**1.2 Сортування цілих**

Реалізувати програму читання з текстового файлу цілих додатних значень (числа розділені пробілами, слід читати до кінця файлу), занесення цих чисел у масив, сортування за зменшенням та за збільшенням суми цифр та зберігання обох результатів у двох нових текстових файлах. Перелічені дії реалізувати в окремій статичній функції. Для визначення порядку сортування створити класи, які реалізують інтерфейс Comparator.

**public** **class** Int {

**public** **static** Comparator<Integer> comareByAbsValues() {

**class** LocalComparator **implements** Comparator<Integer> {

@Override

**public** **int** compare(Integer d1, Integer d2) {

**return** -Integer.*compare*(*abs*(d1), *abs*(d2));

} }

**return** **new** LocalComparator(); }

**public** **static** Comparator<Integer> comareByNorValues() {

**class** LocalComparator **implements** Comparator<Integer> {

@Override

**public** **int** compare(Integer d1, Integer d2) {

**return** Integer.*compare*(*abs*(d1), *abs*(d2));

} }

**return** **new** LocalComparator();

}

**void** readWrite() {

**try** (BufferedReader reader = **new** BufferedReader(**new** FileReader("in.txt"))) {

String line;ArrayList<Integer> numbers = **new** ArrayList<>();Integer[] arr = {};

**while** ((line = reader.readLine()) != **null**){

Scanner scanner = **new** Scanner(line);

**while** (scanner.hasNextInt()){ numbers.add(scanner.nextInt());

}

**int** i=numbers.size()-1;

**while**(i>=0){

**if**(numbers.get(i)<0){

numbers.remove(i); }

i--;} }

arr = numbers.toArray(**new** Integer[numbers.size()]);

PrintWriter firstWriter = **new** PrintWriter(**new** FileWriter("out1.txt"));

PrintWriter secondWriter = **new** PrintWriter(**new** FileWriter("out2.txt"));

**try** { Arrays.*sort*(arr,*comareByNorValues*());

**for** (**int** x : arr)

firstWriter.print(x + " ");

Arrays.*sort*(arr, *comareByAbsValues*());

**for** (**int** x : arr)

secondWriter.print(x + " ");

}

**finally** {

firstWriter.close();

secondWriter.close();}

}

**catch** (IOException ex) {

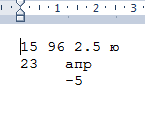
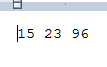
ex.printStackTrace();

} }

**public** **static** **void** main(String[] args) {

**new** Int().readWrite();

}}

**1.3 Реалізація серіализації й десеріализації**

Описати класи Студент і Академічна група (з полем - масивом студентів). Створити об'єкти, здійснити їх бінарну серіалізацію й десеріалізацію, а також серіалізацію й десеріалізацію в XML.

**public** **class** Student **implements** Serializable {

**private** **static** **final** **long** ***serialVersionUID*** = -8744549235976038849L;

**public** String name;

**public** String lastname;

**public** **int** yearsold;

**public** Student(String name,String lastname,**int** yearsold) {

**this**.name = name;

**this**.lastname=lastname;

**this**.yearsold=yearsold; }

**public** Student() {}

@Override

**public** String toString() {

String s=name+" "+lastname+" "+yearsold;

**return** s;

}}

**public** **class** Group **implements** Serializable {

**private** **static** **final** **long** ***serialVersionUID*** = -5942871511728120386L;

**public** Student[] students;

**public** String groupName;

**public** Group() {}

**public** Group(Student[] students,String groupName) {

**this**.students = students;

**this**.groupName=groupName;

}

}

**public** **class** SerializationDeserialisation {

**public** **static** **void** main(String[] args) {

Student[] students = { **new** Student("Rita","Lykova",18), **new** Student("Kolya","Garbozyuk",19), **new** Student("Maks","Kozachenko",18) };

Group group = **new** Group(students,"KN-36b");

**try** (ObjectOutputStream out = **new** ObjectOutputStream(**new** FileOutputStream("temp.dat"))) {

out.writeObject(group);

} **catch** (IOException e) {

e.printStackTrace();

}

**try** (ObjectInputStream in = **new** ObjectInputStream(**new** FileInputStream("temp.dat"))) {

Group group2 = (Group) in.readObject();

System.***out***.println(Arrays.*asList*(group2.students));} **catch** (Exception e) {

e.printStackTrace();}

**try** (XMLEncoder xmlEncoder = **new** XMLEncoder(**new** FileOutputStream("group.xml"))) {

xmlEncoder.writeObject(group);

xmlEncoder.flush();

}

**catch** (IOException e) {

e.printStackTrace();

}**try** (XMLDecoder xmlDecoder = **new** XMLDecoder(**new** FileInputStream("group.xml"))) {

Group group2 = (Group)xmlDecoder.readObject();

System.***out***.println(Arrays.*asList*(group2.students));

}

**catch** (IOException e) {

e.printStackTrace();

} }}

Group.xml

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<java version=*"1.8.0\_151"* class=*"java.beans.XMLDecoder"*>

<object class=*"seclab.third\_ex.Group"* id=*"Group0"*>

<void class=*"seclab.third\_ex.Group"* method=*"getField"*>

<string>students</string>

<void method=*"set"*>

<object idref=*"Group0"*/>

<array class=*"seclab.third\_ex.Student"* length=*"3"*>

<void index=*"0"*>

<object class=*"seclab.third\_ex.Student"* id=*"Student0"*>

<void class=*"seclab.third\_ex.Student"* method=*"getField"*>

<string>name</string>

<void method=*"set"*>

<object idref=*"Student0"*/>

<string>Rita</string>

</void>

</void>

<void class=*"seclab.third\_ex.Student"* method=*"getField"*>

<string>lastname</string>

<void method=*"set"*>

<object idref=*"Student0"*/>

<string>Lykova</string>

</void>

</void>

<void class=*"seclab.third\_ex.Student"* method=*"getField"*>

<string>yearsold</string>

<void method=*"set"*>

<object idref=*"Student0"*/>

<int>18</int>

</void>

</void>

</object>

</void>

<void index=*"1"*>

<object class=*"seclab.third\_ex.Student"* id=*"Student1"*>

<void class=*"seclab.third\_ex.Student"* method=*"getField"*>

<string>name</string>

<void method=*"set"*>

<object idref=*"Student1"*/>

<string>Kolya</string>

</void>

</void>

<void class=*"seclab.third\_ex.Student"* method=*"getField"*>

<string>lastname</string>

<void method=*"set"*>

<object idref=*"Student1"*/>

<string>Garbozyuk</string>

</void>

</void>

<void class=*"seclab.third\_ex.Student"* method=*"getField"*>

<string>yearsold</string>

<void method=*"set"*>

<object idref=*"Student1"*/>

<int>19</int>

</void>

</void>

</object>

</void>

<void index=*"2"*>

<object class=*"seclab.third\_ex.Student"* id=*"Student2"*>

<void class=*"seclab.third\_ex.Student"* method=*"getField"*>

<string>name</string>

<void method=*"set"*>

<object idref=*"Student2"*/>

<string>Maks</string>

</void>

</void>

<void class=*"seclab.third\_ex.Student"* method=*"getField"*>

<string>lastname</string>

<void method=*"set"*>

<object idref=*"Student2"*/>

<string>Kozachenko</string>

</void>

</void>

<void class=*"seclab.third\_ex.Student"* method=*"getField"*>

<string>yearsold</string>

<void method=*"set"*>

<object idref=*"Student2"*/>

<int>18</int>

</void>

</void>

</object>

</void>

</array>

</void>

</void>

<void class=*"seclab.third\_ex.Group"* method=*"getField"*>

<string>groupName</string>

<void method=*"set"*>

<object idref=*"Group0"*/>

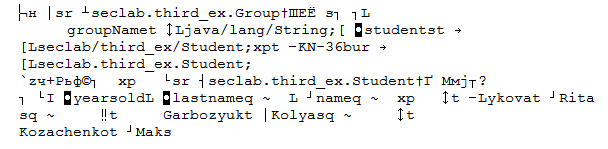
<string>KN-36b</string>

</void>

</void>

</object>

</java>



**1.4 Список файлів усіх підкаталогів**

Увести з клавіатури ім'я певної теки. Вивести на екран імена усіх файлів цієї теки, а також усіх файлів підкаталогів, їхніх підкаталогів тощо. Реалізувати виведення через рекурсивну функцію. Якщо тека не існує, вивести повідомлення про помилку.

**public** **class** FileTest {

**public** **static** **void** findDir(String path) **throws** IOException {

File file = **new** File(path);

//File[]list=file.listFiles();

**for** (File f : file.listFiles()) {

**if** (f.list() != **null**) { System.***out***.println(f.getAbsolutePath()); *findDir*(f.getAbsolutePath());}

**else** System.***out***.println(f.getAbsolutePath());

**try** { Thread.*sleep*(500);

} **catch** (InterruptedException e) { e.printStackTrace();

}}}

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

Scanner s=**new** Scanner(System.***in***);

System.***out***.println("Введите имя папки,которую хотите найти: ");

String dirName=s.next();

File dir = **new** File(dirName);

**if**(!dir.isDirectory()){

**do**{ System.***out***.println("Такой папки не существует!Введите имя папки еще раз:");

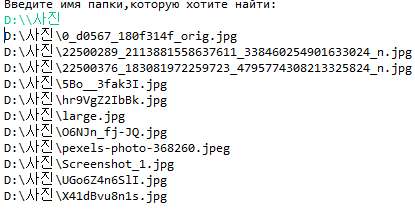
dirName=s.next();

dir = **new** File(dirName);

}**while**(!dir.isDirectory());}

*findDir*(dirName);

}}



**1.5 Робота з ZIP-архівом (додаткове завдання)**

Описати класи Студент і Академічна група (з полем - масивом студентів). Створити об'єкти, здійснити запис даних про студентів академічної групи в архів. В іншій програмі здійснити читання з архіву.

**public** **class** StudentInf {

**public** String name;

**public** String lastname;

**public** **int** yearsold;

**public** String getName() {

**return** name; }

@Override

**public** String toString() {

**return** "Students name is " + name + "\nStudents lastname is " + lastname

+ "\nYearsold: " + yearsold;}

**public** StudentInf(String name, String lastname, **int** yearsold) {

**super**();

**this**.name = name;

**this**.lastname = lastname;

**this**.yearsold = yearsold; }

**public** **void** setName(String name) {

**this**.name = name; }

**public** String getLastname() {

**return** lastname; }

**public** **void** setLastname(String lastname) {

**this**.lastname = lastname;}

**public** **int** getYearsold() {

**return** yearsold;}

**public** **void** setYearsold(**int** yearsold) {

**this**.yearsold = yearsold;}}

**public** **class** Groupp {

**public** StudentInf[] students;

**public** Groupp(StudentInf[] students) {

**super**();

**this**.students = students;}

**public** StudentInf[] getStudents() {

**return** students;

}

**public** **void** setStudents(StudentInf[] students) {

**this**.students = students;

}

@Override

**public** String toString() {

**return** "Group members:\n" + Arrays.*toString*(students);}}

**public** **class** ZiP {

**public** **static** **void** main(String[] args) {

StudentInf[] students = { **new** StudentInf("Rita","Lykova",18), **new** StudentInf("Kolya","Garbozyuk",19), **new** StudentInf("Maks","Kozachenko",18) };

Groupp groupp = **new** Groupp(students);

**try** (ZipOutputStream zOut = **new** ZipOutputStream(**new** FileOutputStream("36bGroup.zip"));

DataOutputStream out = **new** DataOutputStream(zOut)) {

**for** (StudentInf student : groupp.getStudents()) {

ZipEntry zipEntry = **new** ZipEntry(student.getLastname());

zOut.putNextEntry(zipEntry);

out.writeUTF(student.getName());

out.writeInt(student.getYearsold());

zOut.closeEntry();

}

FileWriter fw = **new** FileWriter("results.txt");

}

**catch** (IOException e) {

e.printStackTrace();

}**try** (ZipInputStream zIn = **new** ZipInputStream(**new** FileInputStream("36bGroup.zip"));DataInputStream in = **new** DataInputStream(zIn)) {

ZipEntry entry;

**while** ((entry = zIn.getNextEntry()) != **null**) {System.***out***.println("Last name: " + entry.getName());System.***out***.println("Name: " + in.readUTF()); System.***out***.println("Years old: " + in.readInt()); System.***out***.println(); zIn.closeEntry();

}}

**catch** (IOException e) {

e.printStackTrace();

}}

}

# 

# Висновок : Виконуючи лабораторну роботу ми навчилися працювати з винятками та файлами в Java. Реалізовали серіфлізацію та десеріалізацію. Навчилися працювати з Zip-архівом.