МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «Програмна інженерія та інформаційні технології управління»

Звіт з лабораторної роботи №2

З предмету «Об’єктно-орієнтоване програмування»

Виконав

Студент групи КН-36а

Рубан Ю.Д.

Перевірив:

Козуля М.М.

Кізілов О.С.

Харків 2017

**Робота з винятками і файлами в Java**

**1 Завдання на лабораторну роботу**

**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.

**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.

**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.

**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.

**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.

**1 Хід виконання роботи**

**1.1 Індивідуальне завдання**

Спроектувано та реалізувано класи для представлення сутностей [попередньої лабораторної роботи](http://iwanoff.96.lt/oop_kn/LabTraining01.html). Створено два похідних класи від класу, який представляє основну сутність. Один клас доповнений можливостями читання даних з відповідно підготовленого текстового файлу та запису цих даних в інший файл після сортування. Другий клас реалізовує читання даних з XML-документу, зберігаются дані в структурах, які автоматично створюються за допомогою технології зв'язування даних, та записуются дані в інший XML-документ після сортування.

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

Код програми:

Клас Main:

import java.util.\*;  
public class Main {  
 public static void main(String[] args)  
 {  
 ArrayList<Day> days =new ArrayList<>();  
 Day d1=new Day(30,"день 1");  
 Day d2=new Day(25 ,"на след день получился большой коментарий");  
 Day d3=new Day(26 ,"ААААА");  
 days.add(d1);  
 ShowList s=new ShowList("выставка","автор",days);  
 s.pushDay(d2);  
 s.pushDay(d3);  
 s.sortViaVisitors();  
 s.readFromFile("file.xml");  
 s.writeInFile("file.xml");  
 System.out.println("LIST\n"+s);  
 Day[]arrDay = new Day[3];  
 arrDay[0] = new Day(23,"семен");  
 arrDay[1] = new Day(64,"тест");  
 arrDay[2] = new Day(11,"гламбург");  
 ShowArray sa = new ShowArray("коммуникация","ancient",arrDay);  
 sa.readFromFile("data.txt");  
 sa.sortViaABC();  
 sa.writeInFile("result.txt");  
 System.out.print("ARRAY\n"+sa);  
 }  
}

Клас AbsShow:

public abstract class AbsShow {  
 abstract void pushDay(Day d);  
 void setName(String n){name = n;}  
 String getName(){return name;}  
 void setTitle(String t){title = t;}  
 String getTitle(){return title;}  
 abstract int visitors();  
 abstract int maxVisitors();  
 abstract int dayWithLargeComment();  
 abstract void sortViaVisitors();  
 abstract void sortViaABC();  
 abstract boolean readFromFile(String fileName);  
 abstract boolean writeInFile(String fileName);  
 protected String title;  
 protected String name;  
  
}

Клас Day:

import javax.xml.bind.annotation.XmlElement;  
import javax.xml.bind.annotation.XmlRootElement;  
import javax.xml.bind.annotation.XmlType;  
public class Day  
{  
 Day(int count,String c)  
 {  
 this.comment=c;  
 this.countOfvisitors=count;  
 }  
 Day(){}  
 @Override  
 public String toString() {  
 return "Day{" +  
 "countOfvisitors=" + countOfvisitors +  
 ", comment='" + comment + '\'' +  
 "}"+'\n';  
 }  
 boolean equals(Day obj)  
 {  
 if(countOfvisitors == obj.countOfvisitors && comment == obj.comment)  
 return true;  
 else  
 return false;   
 }  
 public void setComment(String c) {  
 this.comment = c;  
 }  
 public void setCountOfvisitors(int count) {  
 this.countOfvisitors = count;  
 }  
 @XmlElement(name="count")  
 public int getCountOfvisitors() {  
 return this.countOfvisitors;  
 }  
 @XmlElement(name="comment")  
 public String getComment() {  
 return this.comment;  
 }  
 private int countOfvisitors;  
 private String comment;  
}

Клас ShowArray:

import java.io.\*;  
import java.util.Arrays;  
import java.util.StringTokenizer;  
  
public class ShowArray extends AbsShow {  
 private Day[] D;  
 ShowArray(String t, String n, Day[] d) {  
 this.title = t;  
 this.name = n;  
 this.D = d;  
 }  
  
 void pushDay(Day d) {  
 boolean check = false;  
 for (Day t : D) {  
 if (t.equals(d)) {  
 check = true;  
 }  
 }  
 if (!check) {  
 Day[] temp = new Day[D.length + 1];  
 System.*arraycopy*(D, 0, temp, 0, D.length);  
 temp[D.length] = d;  
 D = temp;  
 }  
 }  
  
 @Override  
 public String toString() {  
 return "Show{" +  
 "Days=" + show() +  
 ", title='" + title + '\'' +  
 ", name='" + name + '\'' +  
 '}'+'\n';  
 }  
 private String show()  
 {  
 String output="[";  
 for(int i = 0;i<D.length;i++)  
 {  
 output += D[i];  
 if(i<D.length-1)  
 {  
 output +=", ";  
 }  
 }  
 output+="]";  
 return output;  
 }  
 boolean equals(ShowList obj) {  
 if (name == obj.name && title == obj.title) {  
 return true;  
 } else {  
 return false;  
 }  
 }  
 public int visitors()  
 {  
 int count = 0;  
 for(Day d : D)  
 {  
 count+=d.getCountOfvisitors();  
 }  
 return count;  
 }  
 public int maxVisitors()  
 {  
 int max = 0;  
 for(Day d : D)  
 {  
 if(max<d.getCountOfvisitors())  
 {  
 max = d.getCountOfvisitors();  
 }  
 }  
 return max;  
 }  
 public int dayWithLargeComment()  
 {  
 int max = 0;  
 int index=0;  
 for(int i =0;i<D.length;i++)  
 {  
 if(max<D[i].getComment().length())  
 {  
 max = D[i].getComment().length();  
 index = i;  
 }  
 }  
 return index;  
 }  
 public void sortViaVisitors()  
 {  
 Arrays.*sort*(D,(Day o1, Day o2)->(Integer.*compare*(o2.getCountOfvisitors(),o1.getCountOfvisitors())));  
 }  
 public void sortViaABC()  
 {  
 Arrays.*sort*(D,(Day o1,Day o2)->o1.getComment().compareTo(o2.getComment()));  
 }  
 public boolean readFromFile(String fileName)  
 {  
 try(FileReader fr = new FileReader(fileName))  
 {  
 try(BufferedReader br = new BufferedReader(fr)){  
 String s;  
 Day temp;  
 while ((s=br.readLine())!=null){  
 temp = new Day();  
 StringTokenizer st = new StringTokenizer(s);  
 String comment;  
 comment = new String();  
 temp.setCountOfvisitors(Integer.*parseInt*(st.nextToken()));  
 while (st.countTokens() != 1) {  
 comment += st.nextToken()+" ";  
 }  
 comment+=st.nextToken();  
 temp.setComment(comment);  
 this.pushDay(temp);  
 }  
  
 }}catch (IOException ioe)  
 {  
 ioe.printStackTrace();  
 return false;  
 }  
 return true;  
 }  
 public boolean writeInFile(String fileName)  
 {  
  
 try(FileWriter fw= new FileWriter(fileName)){  
 PrintWriter pw = new PrintWriter(fw);  
 pw.println(this);  
 pw.close();  
  
 }catch(IOException ioe)  
 {  
 ioe.printStackTrace();  
 return false;  
 }  
 return true;  
 }  
}

Клас ShowList:

import java.io.\*;  
import javax.xml.bind.JAXBContext;  
import javax.xml.bind.Marshaller;  
import javax.xml.bind.Unmarshaller;  
import javax.xml.bind.annotation.XmlRootElement;  
import java.util.\*;  
  
@XmlRootElement  
public class ShowList extends AbsShow  
{  
 private List<Day>D;  
 ShowList(String t,String n,ArrayList<Day> d)  
 {  
 this.title = t;  
 this.name = n;  
 this.D=d;  
 }  
 ShowList(){}  
 void pushDay(Day d)  
 {  
 boolean check = false;  
 for (Day t:D)  
 {  
 if(t.equals(d))  
 {  
 check = true;  
 }  
 }  
 if(!check)  
 {  
 D.add(D.size(), d);  
 }  
 }  
 @Override  
 public String toString() {  
 return "Show{" +  
 "Days=" + D +  
 ", title='" + title + '\'' +  
 ", name='" + name + '\'' +  
 '}'+'\n';  
 }  
 boolean equals(ShowList obj)  
 {  
 if(name == obj.name && title == obj.title)  
 {  
 return true;  
 }  
 else  
 {  
 return false;  
 }  
 }  
 public int visitors()  
 {  
 int count = 0;  
 for(Day d : D)  
 {  
 count+=d.getCountOfvisitors();  
 }  
 return count;  
 }  
 public int maxVisitors()  
 {  
 int max = 0;  
 for(Day d : D)  
 {  
 if(max<d.getCountOfvisitors())  
 {  
 max = d.getCountOfvisitors();  
 }  
 }  
 return max;  
 }  
 public int dayWithLargeComment()  
 {  
 int max = 0;  
 int index=0;  
 for(int i =0;i<D.size();i++)  
 {  
 if(max<D.get(i).getComment().length())  
 {  
 max = D.get(i).getComment().length();  
 index = i;  
 }  
 }  
 return index;  
 }  
 public void sortViaVisitors()  
 {  
 Collections.sort(D,(Day o1,Day o2)->(Integer.compare(o2.getCountOfvisitors(),o1.getCountOfvisitors())));  
 }  
 public void sortViaABC()  
 {  
 Collections.sort(D,(Day o1,Day o2)->o1.getComment().compareTo(o2.getComment()));  
 }  
 public void addGroup(ArrayList<Day>group)  
 {  
 D.addAll(D.size(),group);  
 }  
 public boolean readFromFile(String fileName)  
 {  
 try {  
 JAXBContext context = JAXBContext.*newInstance*(Days.class);  
 Unmarshaller unmarshaller = context.createUnmarshaller();  
 FileInputStream xml = new FileInputStream(fileName);  
 Days temp = (Days) unmarshaller.unmarshal(xml);  
 this.addGroup( (ArrayList<Day>)temp.d);  
 }catch (Exception e)  
 {  
 e.printStackTrace();  
 return false;  
 }  
 return true;  
  
 }  
  
 @Override  
 String getName() {  
 return super.getName();  
 }  
  
 @Override  
 String getTitle() {  
 return super.getTitle();  
 }  
  
 @Override  
 void setName(String n) {  
 super.setName(n);  
 }  
  
 @Override  
 void setTitle(String t) {  
 super.setTitle(t);  
 }  
  
 public void setD(List<Day> d) {  
 D = d;  
 }  
  
 List<Day>getD(){return this.D;}  
 public boolean writeInFile(String fileName)  
 {  
 File file = new File(fileName);  
 Days t = new Days();  
 t.setD(D);  
  
 try {  
 JAXBContext context = JAXBContext.newInstance(Days.class);  
 Marshaller marshaller = context.createMarshaller();  
 marshaller.setProperty(Marshaller.JAXB\_FORMATTED\_OUTPUT, Boolean.TRUE);  
 marshaller.marshal(t, file);  
 }catch (Exception e)  
 {  
 e.printStackTrace();  
 return false;  
 }  
 return true;  
 }  
}

Клас Days:

import javax.xml.bind.annotation.XmlElement;  
import javax.xml.bind.annotation.XmlRootElement;  
import java.util.ArrayList;  
import java.util.List;  
  
@XmlRootElement(name="Days")  
public class Days {  
 @XmlElement(name = "DAY")  
 List<Day> d;  
  
 public Days() {}  
  
 public void setD(List<Day> d) {  
 this.d = d;  
 }  
}

**Результати виконання програми**

Консольне вікно:

LIST

Show{Days=[Day{countOfvisitors=30, comment='день 1'}

, Day{countOfvisitors=26, comment='ААААА'}

, Day{countOfvisitors=25, comment='на след день получился большой коментарий'}

, Day{countOfvisitors=30, comment='день 1'}

, Day{countOfvisitors=26, comment='ААААА'}

, Day{countOfvisitors=25, comment='на след день получился большой коментарий'}

, Day{countOfvisitors=30, comment='день 1'}

, Day{countOfvisitors=26, comment='ААААА'}

, Day{countOfvisitors=25, comment='на след день получился большой коментарий'}

, Day{countOfvisitors=30, comment='день 1'}

, Day{countOfvisitors=26, comment='d'}

, Day{countOfvisitors=25, comment='на след день получился большой коментарий'}

], title='выставка', name='автор'}

ARRAY

Show{Days=[Day{countOfvisitors=66, comment='day, when all begins'}

, Day{countOfvisitors=22, comment='just'}

, Day{countOfvisitors=11, comment='гламбург'}

, Day{countOfvisitors=23, comment='семен'}

, Day{countOfvisitors=64, comment='тест'}

], title='коммуникация', name='ancient'}

XML файл:

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>  
<Days>  
 <DAY>  
 <comment>день 1</comment>  
 <count>30</count>  
 </DAY>  
 <DAY>  
 <comment>ААААА</comment>  
 <count>26</count>  
 </DAY>  
 <DAY>  
 <comment>на след день получился большой коментарий</comment>  
 <count>25</count>  
 </DAY>  
 <DAY>  
 <comment>день 1</comment>  
 <count>30</count>  
 </DAY>  
 <DAY>  
 <comment>ААААА</comment>  
 <count>26</count>  
 </DAY>  
 <DAY>  
 <comment>на след день получился большой коментарий</comment>  
 <count>25</count>  
 </DAY>  
 <DAY>  
 <comment>день 1</comment>  
 <count>30</count>  
 </DAY>  
 <DAY>  
 <comment>ААААА</comment>  
 <count>26</count>  
 </DAY>  
 <DAY>  
 <comment>на след день получился большой коментарий</comment>  
 <count>25</count>  
 </DAY>  
 <DAY>  
 <comment>день 1</comment>  
 <count>30</count>  
 </DAY>  
 <DAY>  
 <comment>d</comment>  
 <count>26</count>  
 </DAY>  
 <DAY>  
 <comment>на след день получился большой коментарий</comment>  
 <count>25</count>  
 </DAY>  
</Days>

### 1.2 Сортування цілих

Реалізувано 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 реалізовано в окремій статичній функції. Для визначення порядку сортування створено класи, які реалізують інтерфейс Comparator.

Код програми:

Клас Main:

import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.FileWriter;  
import java.io.PrintWriter;  
import java.util.ArrayList;  
import java.util.Collections;  
import java.util.List;  
import java.util.StringTokenizer;  
public class Main {  
 public static void main(String[] args) {  
 try  
 {  
 FileReader fileReader = new FileReader("file.txt");  
 BufferedReader bufferedReader = new BufferedReader(fileReader);  
 String s;  
 StringTokenizer st;  
 Integer number=0;  
 List<Integer>numbers=new ArrayList<>();  
 while ((s=bufferedReader.readLine())!=null)  
 {  
 st = new StringTokenizer(s);  
 while(st.countTokens()!=0)  
 {  
 s = st.nextToken();  
 try {  
 number = Integer.*parseInt*(s);  
 }catch (Exception e){continue;}  
 if(number>0)  
 numbers.add(number);  
 }  
 }  
 Collections.*sort*(numbers,new Big());  
 PrintWriter printWriter = new PrintWriter(new FileWriter("Big.txt"));  
 printWriter.println(numbers);  
 printWriter.close();  
 Collections.*sort*(numbers,new Small());  
 printWriter = new PrintWriter(new FileWriter("Small.txt"));  
 printWriter.println(numbers);  
 printWriter.close();  
 }catch (Exception e)  
 {e.printStackTrace();}  
   
 }  
}

Клас Big:  
import java.util.Comparator;  
public class Big implements Comparator<Integer> {  
 @Override  
 public int compare(Integer t1, Integer t2) {  
 Integer sum1=0;  
 Integer sum2=0;  
 Integer number=0;  
 String o1 = t1.toString();  
 String o2 = t2.toString();  
 for(int i=0;i<o1.length();i++)  
 {  
 number = Character.*getNumericValue*(o1.charAt(i));  
 if(number<0){continue;}  
 sum1+= number;  
 }  
 number=0;  
 for(int i=0;i<o2.length();i++)  
 {  
 number=Character.*getNumericValue*(o2.charAt(i));  
 if(number<0){continue;}  
 sum2+=number;  
 }  
 return sum2.compareTo(sum1);  
 }  
}

Клас Small:  
import java.util.Comparator;  
public class Small implements Comparator<Integer> {  
 @Override  
 public int compare(Integer t1, Integer t2)  
 {  
 Integer sum1=0;  
 Integer sum2=0;  
 Integer number=0;  
 String o1=t1.toString();  
 String o2=t2.toString();  
 for(int i=0;i<o1.length();i++)  
 {  
 number = Character.*getNumericValue*(o1.charAt(i));  
 if(number<0){continue;}  
 sum1+= number;  
 }  
 number=0;  
 for(int i=0;i<o2.length();i++)  
 {  
 number=Character.*getNumericValue*(o2.charAt(i));  
 if(number<0){continue;}  
 sum2+=number;  
 }  
 return sum1.compareTo(sum2);  
 }  
}

Файл для читання file.txt

11 34 50 -54 0.5 0,6 fdfs

Файл Big.txt

[34, 50, 11]

Файл Small.txt

[11, 50, 34]

### 1.3 Реалізація серіализації й десеріализації

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

Код програми:

Клас Main:  
import java.beans.XMLDecoder;  
import java.beans.XMLEncoder;  
import java.io.\*;  
  
public class Main {  
 public static void main(String[] args) {  
 Group group = new Group(2);  
 group.setStudent(0,new Student("bob",325654));  
 group.setStudent(1,new Student("meh",325754));  
 try  
 {  
 ObjectOutputStream objectOutputStream = new ObjectOutputStream(new FileOutputStream("File.obj"));  
 XMLEncoder xmlEncoder = new XMLEncoder(new BufferedOutputStream(new FileOutputStream("File.xml")));  
 xmlEncoder.writeObject(group);  
 xmlEncoder.flush();  
 xmlEncoder.close();  
 objectOutputStream.writeObject(group);  
 }catch (Exception e)  
 {  
 e.printStackTrace();  
 }  
 try {  
 Group temp;  
 ObjectInputStream objectInputStream = new ObjectInputStream(new FileInputStream("File.obj"));  
 temp = (Group) objectInputStream.readObject();  
 System.*out*.println(temp);  
 Group xmlTemp;  
 XMLDecoder xmlDecoder = new XMLDecoder(new FileInputStream("File.xml"));  
 xmlTemp = (Group)xmlDecoder.readObject();  
 System.*out*.println("XML\n"+xmlTemp);  
 }catch (ClassNotFoundException e)  
 {  
 e.printStackTrace();  
 }catch (ArrayIndexOutOfBoundsException b)  
 {  
 System.*out*.print("bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb");  
 }catch (IOException e)  
 {  
 e.printStackTrace();  
 }  
 }  
}

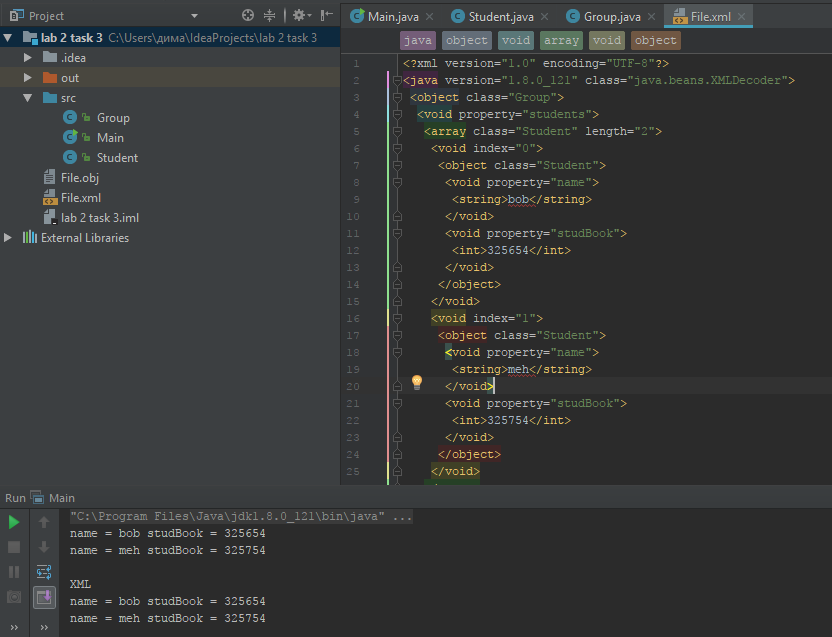
Клас Student:

import java.io.Serializable;  
  
public class Student implements Serializable{  
 private String name;  
 private int studBook;  
 public Student(){}  
 public Student(String name, int studBook)  
 {  
 this.name=name;  
 this.studBook=studBook;  
 }  
 public int getStudBook() {  
 return studBook;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public void setStudBook(int studBook) {  
 this.studBook = studBook;  
 }  
 @Override  
 public String toString()  
 {  
 return "name = "+name + " studBook = " + studBook+"\n";  
 }  
}

Клас Group:

import java.io.Serializable;  
  
public class Group implements Serializable{  
 private Student[] students;  
 public Group(){}  
 public Group(int n)  
 {  
 students = new Student[n];  
 }  
 public Student getStudent(int i)  
 {  
 return students[i];  
 }  
  
 public void setStudents(Student[] students) {  
 this.students = students;  
 }  
  
 public void setStudent(int i, Student st)  
 {  
 students[i]=st;  
 }  
  
 public Student[] getStudents() {  
 return students;  
 }  
  
 @Override  
 public String toString()  
 {  
 String res="";  
 for (Student s:students)  
 {  
 res+=s;  
 }  
 return res;  
 }  
}

Результати виконання програми:



### 1.4 Список файлів усіх підкаталогів

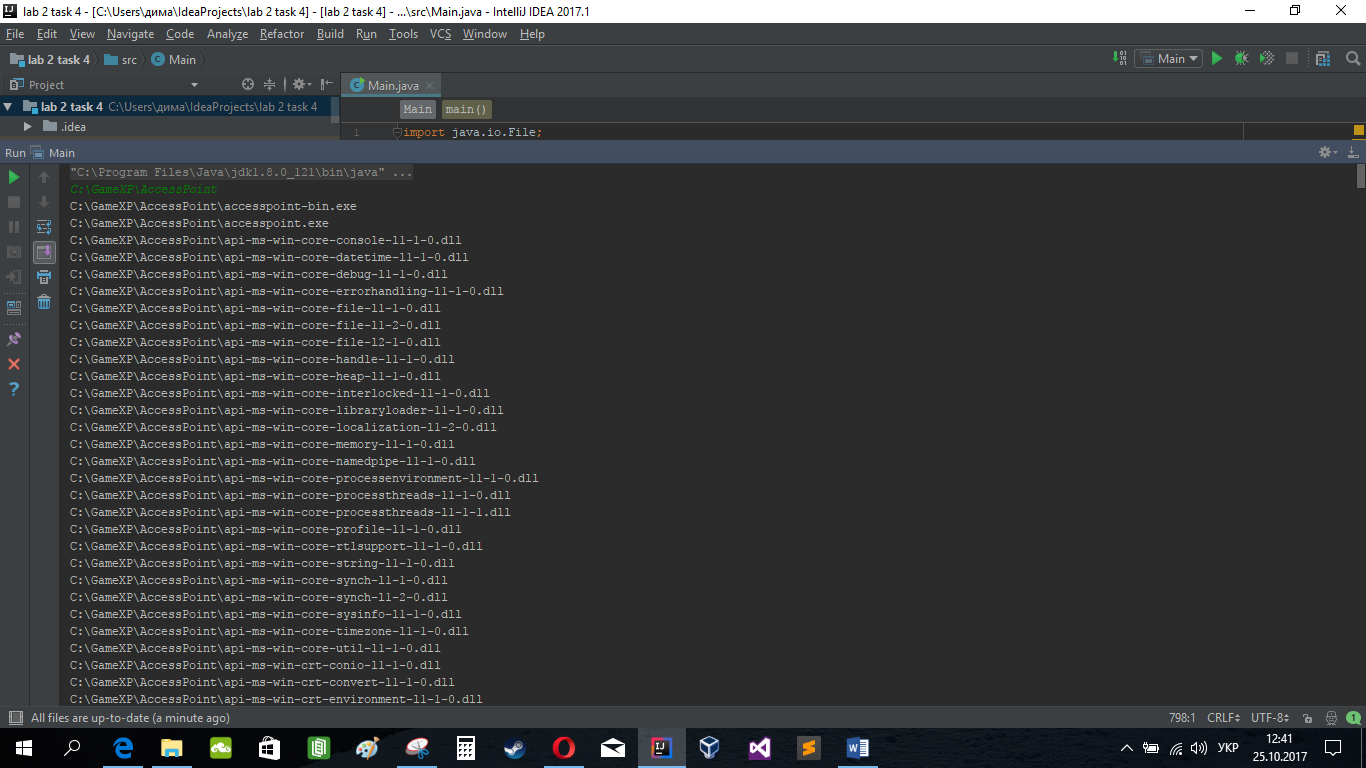
Введено з клавіатури ім'я певної теки. Виведено на екран імена усіх файлів цієї теки, а також усіх файлів підкаталогів, їхніх підкаталогів тощо. Реалізувано виведення через рекурсивну функцію. Якщо тека не існує, виведено повідомлення про помилку.

Код програми:

Клас Main:

import java.io.File;  
import java.io.IOException;  
import java.util.Scanner;  
  
public class Main {  
 public static void show(File dir)  
 {  
 for (File temp:dir.listFiles()) {  
 if(temp.isDirectory()) {  
 *show*(temp);  
 }  
 System.*out*.println(temp);  
 }  
 }  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 String dirName = sc.nextLine();  
 File dir = new File(dirName);  
 if(!dir.isDirectory())  
 {  
 System.*out*.print("wrong name");  
 }  
 *show*(dir);  
 }  
}

Результат виконання програми:



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

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

Код програми:

import java.io.DataInputStream;  
import java.io.DataOutputStream;  
import java.io.FileInputStream;  
import java.io.FileOutputStream;  
import java.util.zip.ZipEntry;  
import java.util.zip.ZipInputStream;  
import java.util.zip.ZipOutputStream;  
  
public class Main {  
 public static void main(String[] args) {  
 Group group = new Group(2);  
 group.setStudent(0,new Student("bob",43532));  
 group.setStudent(1,new Student("john",2353453));  
 try {  
 ZipOutputStream zipOutputStream = new ZipOutputStream(new FileOutputStream("file.zip"));  
 DataOutputStream dataOutputStream = new DataOutputStream(zipOutputStream);  
 for(Student student:group.getStudents())  
 {  
 ZipEntry zipEntry = new ZipEntry(student.getName());  
 zipOutputStream.putNextEntry(zipEntry);  
 dataOutputStream.writeInt(student.getStudBook());  
 zipOutputStream.closeEntry();  
 }  
 }catch (Exception e)  
 {  
 e.printStackTrace();  
 }  
 try {  
 ZipInputStream zipInputStream = new ZipInputStream(new FileInputStream("file.zip"));  
 DataInputStream dataInputStream = new DataInputStream(zipInputStream);  
 ZipEntry zipEntry;  
 while ((zipEntry=zipInputStream.getNextEntry())!=null)  
 {  
 System.*out*.println("name = "+ zipEntry.getName());  
 System.*out*.println("studBook = "+ dataInputStream.readInt());  
 zipInputStream.closeEntry();  
 }  
 }catch (Exception e)  
 {  
 e.printStackTrace();  
 }  
 }  
}

Клас Group:

import java.io.Serializable;  
  
public class Group implements Serializable{  
 private Student[] students;  
 Group(){}  
 Group(int n)  
 {  
 students = new Student[n];  
 }  
 public Student getStudent(int i)  
 {  
 return students[i];  
 }  
 public void setStudent(int i, Student st)  
 {  
 students[i]=st;  
 }  
 @Override  
 public String toString()  
 {  
 String res="";  
 for (Student s:students)  
 {  
 res+=s;  
 }  
 return res;  
 }  
  
 public Student[] getStudents() {  
 return students;  
 }  
}

Клас Student:

import java.io.Serializable;  
  
public class Student implements Serializable{  
 private String name;  
 private int studBook;  
 Student(){}  
 Student(String name, int studBook)  
 {  
 this.name=name;  
 this.studBook=studBook;  
 }  
 public int getStudBook() {  
 return studBook;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public void setStudBook(int studBook) {  
 this.studBook = studBook;  
 }  
 @Override  
 public String toString()  
 {  
 return "name = "+name + " studBook = " + studBook+"\n";  
 }  
}

Результат виконання програми:

name = bob

studBook = 43532

name = john

studBook = 2353453

**Висновки:**

В даній лабораторній роботі було засвоєно теоретичний матеріал з використання XML технологій, а також, роботи з файлами та архівами.