Міністерство Освіти і Науки України

Національний технічний університет

«Харківський політехнічний інститут»

Кафедра «Програмної інженерії та інформаційних технологій управління»

**Лабораторна робота № 2**

# Робота з винятками і файлами в Java

**Виконав:**

Студент групи КН-36А

Антоненко В.В.

**Перевірив:**

Козуля М.М.

Харків 2017

**Завданная**

### 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.

**Хід роботи**

### 1.1 Індивідуальне завдання

*Код програми 1.1:*

**День**

**package** lab2.individual;  
  
**import** java.util.StringTokenizer;  
  
**public class** Day **implements** Comparable<Day> {  
 **private** String **data**;  
 **private double temperature**;  
 **private** String **comments**;  
  
  
 **public** Day() {  
 }  
  
 **public** Day(String data, **double** temperature, String comments) {  
 **this**.**data** = data;  
 **this**.**temperature** = temperature;  
 **this**.**comments** = comments;  
 }  
  
 @Override  
 **public** String toString() {  
 **return "Дата: "** + getData() + **"\nТемпература равна "** + getTemperature() +  
 **" градусов\nКомментарий: "** + getComments();  
 }  
  
 @Override  
 **public boolean** equals(Object obj) {  
 **if** (**this** == obj) {  
 **return true**;  
 }  
 **if** (obj == **null** || !(obj **instanceof** Day)) {  
 **return false**;  
 }  
 Day c = (Day) obj;  
 **return** c.getData() == getData() &&  
 c.getTemperature() == getTemperature() &&  
 c.getComments().equals(getComments());  
 }  
  
 **public** String getComments() {  
 **return comments**;  
 }  
  
 **public void** setComments(String comments) {  
 **this**.**comments** = comments;  
 }  
  
 **public double** getTemperature() {  
 **return temperature**;  
 }  
  
 **public void** setTemperature(**double** temperature) {  
 **this**.**temperature** = temperature;  
 }  
  
 **public** String getData() {  
 **return data**;  
 }  
  
 **public void** setDatd(String data) {  
 **this**.**data** = data;  
 }  
  
  
 **public int** kolvosim() {  
 **int** m = **comments**.length();  
 **return** m;  
 }  
  
 **public boolean** containsWord(String word) {  
 StringTokenizer st = **new** StringTokenizer(getComments());  
 String s;  
 **while** (st.hasMoreTokens()) {  
 s = st.nextToken();  
 **if** (s.equalsIgnoreCase(word)) {  
 **return true**;  
 }  
 }  
 **return false**;  
 }  
  
 **public boolean** containsSubstring(String substring) {  
 **return** getComments().toUpperCase().indexOf(substring.toUpperCase()) >= 0;  
 }  
  
 **private void** testWord(String word) {  
 **if** (containsWord(word)) {  
 System.***out***.println(**"Слово \""** + word + **"\" міститься у коментарі"**);  
 } **else** {  
 System.***out***.println(**"Слово \""** + word + **"\" не міститься у коментарі"**);  
 }  
 **if** (containsSubstring(word)) {  
 System.***out***.println(**"Текст \""** + word + **"\" міститься у коментарі"**);  
 } **else** {  
 System.***out***.println(**"Текст \""** + word + **"\" не міститься у коментарі"**);  
 }  
 }  
  
 @Override  
 **public int** compareTo(Day c) {  
 **return** Double.*compare*(getTemperature(), c.getTemperature());  
 }  
  
 **public static void** main(String[] args) {  
 Day days = **new** Day(**"20.05.17"**, 22.5,  
 **"Облачно с прояснениями"**);  
 days.testWord(**"Облачно"**);  
 days.testWord(**"Облако"**);  
 days.testWord(**"Ясно"**); }}

**Погода**

**package** lab2.individual;  
  
  
**import** java.util.Arrays;  
**import** java.util.Comparator;  
  
  
  
**public abstract class** AbstractPogoda {  
 **abstract public** String getSezon();  
 **abstract public void** setSezon(String sezon);  
 **abstract public** String getCom();  
 **abstract public void** setCom(String com);  
 **abstract public void** addDays(Day day);  
 **abstract public void** sortByTemp();  
 **abstract public void** sortByComments();  
 **abstract public void** clearDay();  
 **abstract public int** daysCount();  
 **abstract public** Day getDays(**int** index);  
 **abstract public void** setDay(**int** i, Day day);  
 **abstract public boolean** readFromFile(String fileName);  
 **abstract public boolean** writeToFile(String fileName);  
  
 **public static** Day[] addToArray(Day[] arr, Day item) {  
 Day[] newArr;  
 **if** (arr != **null**) {  
 newArr = **new** Day[arr.**length** + 1];  
 System.*arraycopy*(arr, 0, newArr, 0, arr.**length**);  
 }  
 **else** {  
 newArr = **new** Day[1];  
 }  
 newArr[newArr.**length** - 1] = item;  
 **return** newArr;  
 }  
  
 **public double** sered\_temp() {  
 **double** sum = 0;  
 **for** (**int** i = 0; i < daysCount(); i++) {  
 sum += getDays(i).getTemperature();  
 }  
 **return** sum / daysCount();  
 }  
  
  
 **public** String maxTemp() {  
 Day max = getDays(0);  
 **for** (**int** i = 1; i < daysCount(); i++) {  
 **if** (max.getTemperature() < getDays(i).getTemperature()) {  
 max = getDays(i);  
 }  
 }  
 **return** max.getData();  
 }  
  
 **public** String maxComment() {  
 Day max = getDays(0);  
 **for** (**int** i = 1; i < daysCount(); i++) {  
 **if** (max.kolvosim() < getDays(i).kolvosim()) {  
 max = getDays(i);  
 }  
 }  
 **return** max.getData();  
 }  
  
 **public** Day[] findWord(String word) {  
 Day[] result = **null**;  
 **for** (**int** i = 0; i < daysCount(); i++) {  
 Day days = getDays(i);  
 **if** (days.containsWord(word)) {  
 result = *addToArray*(result, days);  
 }  
 }  
 **return** result;  
 }  
  
 **public void** showData(String title) {  
 System.***out***.println(title);  
 System.***out***.println(getSezon() + **" "** + getCom());  
 **for** (**int** i = 0; i < daysCount(); i++) {  
 System.***out***.println(getDays(i).getData() + **" "** + getDays(i).getTemperature() + **" "** + getDays(i).getComments());  
 }  
 System.***out***.println();  
 }  
  
  
 **protected void** test(String inFileName, String outFileName) {  
 readFromFile(inFileName);  
 showData(**"Выходные данные:"**);  
 System.***out***.println(**"Максимальная температура: "** + maxTemp());  
 System.***out***.println(**"Самый длинный комментарий: "** + maxComment());  
 sortByTemp();  
 showData(**"Сортировка по температуре:"**);  
 sortByComments();  
 showData(**"Сотрировка за алфавитом комментария:"**);  
 writeToFile(outFileName);  
 }  
}

*Клас TextFilePogoda*

**package** lab2.individual;  
  
**import** java.io.\*;  
**import** java.util.Arrays;  
**import** java.util.Scanner;  
**import** java.util.Comparator;  
  
**public class** TextFilePogoda **extends** AbstractPogoda {  
 **private** String **sezon**;  
 **private** String **com**;  
 **private** Day[] **days**;  
  
  
 @Override  
 **public** String getSezon() {  
 **return sezon**;  
 }  
  
 @Override  
 **public void** setSezon(String sezon) {  
 **this**.**sezon** = sezon;  
 }  
  
 @Override  
 **public** String getCom() {  
 **return com**;  
 }  
  
 @Override  
 **public void** setCom(String com) {  
 **this**.**com** = com;  
 }  
  
 @Override  
 **public void** addDays(Day day) {  
 **days** = *addToArray*(**days**, day);  
 }  
  
 @Override  
 **public void** sortByTemp() {  
 Arrays.*sort*(**days**, **new** CompareByTemp());  
 }  
  
 @Override  
 **public void** sortByComments() {  
 Arrays.*sort*(**days**, **new** CompareByComments());  
 }  
  
 @Override  
 **public void** clearDay() {  
 **days** = **new** Day[0];  
 }  
  
 @Override  
 **public int** daysCount() {  
 **return days**.**length**;  
 }  
  
 @Override  
 **public** Day getDays(**int** index) {  
 **return days**[index];  
 }  
  
 @Override  
 **public void** setDay(**int** i, Day day) {  
 **days**[i] = day;  
 }  
  
 @Override  
 **public boolean** readFromFile(String fileName) {  
 **try** (Scanner scanner = **new** Scanner(**new** FileReader(fileName))) {  
 setSezon(scanner.next());  
 setCom(scanner.nextLine());  
 **while** (scanner.hasNext()) {  
 String data = scanner.next();  
 **double** temperature = scanner.nextDouble();  
 String com = scanner.nextLine();  
 addDays(**new** Day(data, temperature, com));  
 }  
 }  
 **catch** (IOException ex) {  
 **return false**;  
 }  
 **return true**;  
 }  
  
 @Override  
 **public boolean** writeToFile(String fileName) {  
 **try** (PrintWriter out = **new** PrintWriter(**new** FileWriter(fileName))) {  
 out.println(getSezon() + **" "** + getCom());  
 **for** (Day day : **days**) {  
 out.print(day.getData() + **" "** + day.getTemperature());  
 out.println(day.getComments());  
 }  
 }  
 **catch** (IOException e) {  
 **return false**;  
 }  
 **return true**;  
 }  
  
 **class** CompareByTemp **implements** Comparator<Day> {  
  
 **public int** compare(Day c1, Day c2) {  
 **return** Double.*compare*(c1.getTemperature(), c2.getTemperature());  
 }  
  
 }  
  
 **class** CompareByComments **implements** Comparator<Day> {  
  
 **public int** compare(Day c1, Day c2) {  
 **return** c1.getComments().compareTo(c2.getComments());  
 }  
  
 }  
  
 **public static void** main(String[] args) {  
 **new** TextFilePogoda().test(**"Summer.txt"**, **"Sorted.txt"**);  
 }  
}

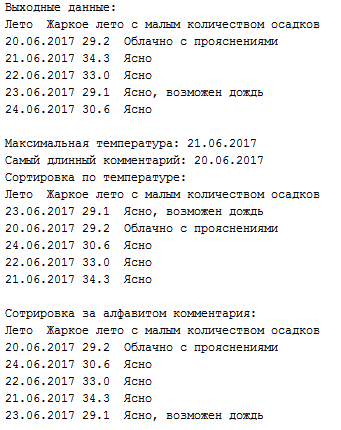
*Файл Summer.txt містить такі дані:*

Лето Жаркое лето с малым количеством осадков  
20.06.2017 29,2 Облачно с прояснениями  
21.06.2017 34,3 Ясно  
22.06.2017 33,0 Ясно  
23.06.2017 29,1 Ясно, возможен дождь  
24.06.2017 30,6 Ясно

*Після запуску програми в файл Sorted.txt записуються такі дані:*

Лето Жаркое лето с малым количеством осадков  
20.06.2017 29.2 Облачно с прояснениями  
24.06.2017 30.6 Ясно  
22.06.2017 33.0 Ясно  
21.06.2017 34.3 Ясно  
23.06.2017 29.1 Ясно, возможен дождь

*Виведення даних у консольне вікно:*



*Клас XMLPogoda*

**package** lab2.individual;  
  
**import** lab2.individual.xml.PogodaData;  
**import** lab2.individual.xml.PogodaData.DayData;  
  
**import** javax.xml.bind.JAXBContext;  
**import** javax.xml.bind.JAXBException;  
**import** javax.xml.bind.Marshaller;  
**import** javax.xml.bind.Unmarshaller;  
**import** java.io.FileInputStream;  
**import** java.io.FileNotFoundException;  
**import** java.io.FileWriter;  
**import** java.io.IOException;  
**import** java.util.Arrays;  
**import** java.util.Comparator;  
  
**public class** XMLPogada **extends** AbstractPogoda {  
 **private** PogodaData **cd**;  
 @Override  
 **public** String getSezon() {  
 **return cd**.getSezon();  
 }  
  
 @Override  
 **public void** setSezon(String sezon) {  
 **cd**.setSezon(sezon);  
 }  
  
 @Override  
 **public** String getCom() {  
 **return cd**.getCom();  
 }  
  
 @Override  
 **public void** setCom(String com) {  
 **cd**.setCom(com);  
 }  
  
 @Override  
 **public void** addDays(Day day) {  
 DayData dayData = **new** DayData();  
 dayData.setData(day.getData());  
 dayData.setTemperature(day.getTemperature());  
 dayData.setComments(day.getComments());  
 **cd**.getDayData().add(dayData);  
 }  
  
 @Override  
 **public void** sortByTemp() {  
 Day[] arr = **new** Day[daysCount()];  
 **for** (**int** i = 0; i < arr.**length**; i++) {  
 arr[i] = getDays(i);  
 }  
 Arrays.*sort*(arr, **new** CompareByTemp());  
 **for** (**int** i = 0; i < arr.**length**; i++) {  
 setDay(i, arr[i]);  
 }  
 }  
  
 @Override  
 **public void** sortByComments() {  
 Day[] arr = **new** Day[daysCount()];  
 **for** (**int** i = 0; i < arr.**length**; i++) {  
 arr[i] = getDays(i);  
 }  
 Arrays.*sort*(arr, **new** CompareByComments());  
 **for** (**int** i = 0; i < arr.**length**; i++) {  
 setDay(i, arr[i]);  
 }  
 }  
  
 @Override  
 **public void** clearDay() {  
 **cd**.getDayData().clear();  
 }  
  
 @Override  
 **public int** daysCount() {  
 **return cd**.getDayData().size();  
 }  
  
 @Override  
 **public** Day getDays(**int** index) {  
 DayData dayData = **cd**.getDayData().get(index);  
 **return new** Day(dayData.getData(), dayData.getTemperature(), dayData.getComments());  
 }  
  
 @Override  
 **public void** setDay(**int** i, Day day) {  
 DayData dayData = **new** DayData();  
 dayData.setData(day.getData());  
 dayData.setTemperature(day.getTemperature());  
 dayData.setComments(day.getComments());  
 **cd**.getDayData().set(i, dayData);  
 }  
  
 @Override  
 **public boolean** readFromFile(String fileName) {  
 **try** {  
 JAXBContext jaxbContext = JAXBContext.*newInstance*(**"lab2.individual.xml"**);  
 Unmarshaller unmarshaller = jaxbContext.createUnmarshaller();  
 **cd** = (PogodaData) unmarshaller.unmarshal(**new** FileInputStream(fileName));  
 **return true**;  
 }  
 **catch** (FileNotFoundException | JAXBException e) {  
 **return false**;  
 }  
 }  
  
 @Override  
 **public boolean** writeToFile(String fileName) {  
 **try** {  
 JAXBContext jaxbContext = JAXBContext.*newInstance*(**"lab2.individual.xml"**);  
 Marshaller marshaller = jaxbContext.createMarshaller();  
 marshaller.setProperty(Marshaller.***JAXB\_FORMATTED\_OUTPUT***, Boolean.***TRUE***);  
 marshaller.marshal(**cd**, **new** FileWriter(fileName));  
 **return true**;  
 }  
 **catch** (JAXBException | IOException e) {  
 **return false**;  
 }  
 }  
  
 **class** CompareByTemp **implements** Comparator<Day> {  
  
 **public int** compare(Day c1, Day c2) {  
 **return** Double.*compare*(c1.getTemperature(), c2.getTemperature());  
 }  
  
 }  
  
 **class** CompareByComments **implements** Comparator<Day> {  
  
 **public int** compare(Day c1, Day c2) {  
 **return** c1.getComments().compareTo(c2.getComments());  
 }  
  
 }  
  
 **public static void** main(String[] args) {  
 **new** XMLPogada().test(**"Summer.xml"**, **"Sorted.xml"**);  
 }  
  
}

*Файл Pogoda.xsd*

*<?***xml version="1.0" encoding="UTF-8"***?>*<**xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"**>  
 <**xs:element name="PogodaData"**>  
 <**xs:complexType**>  
 <**xs:sequence**>  
 <**xs:element maxOccurs="unbounded" name="DayData"**>  
 <**xs:complexType**>  
 <**xs:attribute name="Data" type="xs:string" use="required"** />  
 <**xs:attribute name="Temperature" type="xs:double" use="required"** />  
 <**xs:attribute name="Comments" type="xs:string" use="required"** />  
 </**xs:complexType**>  
 </**xs:element**>  
 </**xs:sequence**>  
 <**xs:attribute name="Sezon" type="xs:string" use="required"** />  
 <**xs:attribute name="Com" type="xs:string" use="required"** />  
 </**xs:complexType**>  
 </**xs:element**>  
</**xs:schema**>

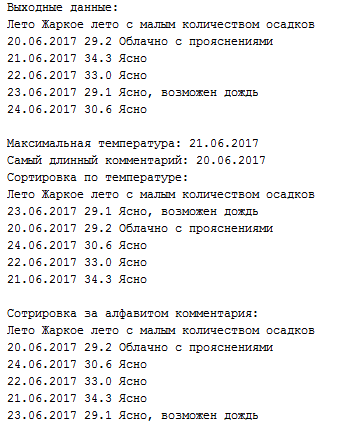
*Файл Summer.xml*

*<?***xml version="1.0" encoding="UTF-8"***?>*<**PogodaData Sezon="Лето" Com="Жаркое лето с малым количеством осадков"** >  
 <**DayData Data="20.06.2017" Temperature="29.2" Comments="Облачно с прояснениями"** />  
 <**DayData Data="21.06.2017" Temperature="34.3" Comments="Ясно"** />  
 <**DayData Data="22.06.2017" Temperature="33.0" Comments="Ясно"** />  
 <**DayData Data="23.06.2017" Temperature="29.1" Comments="Ясно, возможен дождь"** />  
 <**DayData Data="24.06.2017" Temperature="30.6" Comments="Ясно"** />  
</**PogodaData**>

*Файл Sorted.xml після запуску програми*

*<?***xml version="1.0" encoding="UTF-8"***?>*<**PogodaData Sezon="Лето" Com="Жаркое лето с малым количеством осадков"** >  
 <**DayData Data="20.06.2017" Temperature="29.2" Comments="Облачно с прояснениями"** />  
 <**DayData Data="21.06.2017" Temperature="34.3" Comments="Ясно"** />  
 <**DayData Data="22.06.2017" Temperature="33.0" Comments="Ясно"** />  
 <**DayData Data="23.06.2017" Temperature="29.1" Comments="Ясно, возможен дождь"** />  
 <**DayData Data="24.06.2017" Temperature="30.6" Comments="Ясно"** />  
</**PogodaData**>

*Виведення даних у консольне вікно:*



### 1.2 Сортування цілих

*Код програми 1.2:*

**package** lab2.sort;  
  
**import** java.io.\*;  
**import** java.util.\*;  
  
**public class** SortInt {  
  
**public static class** IntegerValueException **extends** Exception {  
 **private double wrongValue**;  
  
 **public** IntegerValueException(**double** wrongValue) {  
 **this**.**wrongValue** = wrongValue;  
 }  
  
 **public double** getWrongValue() {  
 **return wrongValue**;  
 }  
  
 }  
**public static** Comparator<Integer> comareByAbsValues() {  
 **class** LocalComparator **implements** Comparator<Integer> {  
 @Override  
 **public int** compare(Integer d1, Integer d2) {  
 **return** -Integer.*compare*(d1, d2);  
 }  
 }  
 **return new** LocalComparator();  
 }  
  
 **public static void** sortDoubles(String inFileName, String firstOutFileName,  
 String secondOutFileName) **throws** IntegerValueException, IOException,  
 InputMismatchException {  
 Integer[] arr = {};  
 **try** (BufferedReader reader = **new** BufferedReader(**new** FileReader(inFileName));  
 Scanner scanner = **new** Scanner(reader)) {  
 **while** (scanner.hasNext()) {  
 **int** d = scanner.nextInt();  
 **if** (d < 0) {  
 **throw new** IntegerValueException(d);  
 }  
 Integer[] arr1 = **new** Integer[arr.**length** + 1];  
 System.*arraycopy*(arr, 0, arr1, 0, arr.**length**);  
 arr1[arr.**length**] = d;  
 arr = arr1;  
 }  
 }  
 PrintWriter firstWriter = **new** PrintWriter(**new** FileWriter(firstOutFileName));  
 PrintWriter secondWriter = **new** PrintWriter(**new** FileWriter(secondOutFileName));  
 **try** {  
 Arrays.*sort*(arr);  
 **for** (Integer x : arr)  
 firstWriter.print(x + **" "**);  
 firstWriter.print(**"\n"**);  
 Arrays.*sort*(arr, *comareByAbsValues*());  
 **for** (Integer x : arr)  
 firstWriter.print(x + **" "**);  
 **int** p = 0;  
 **for** (Integer x : arr)  
 p += x;  
 secondWriter.print(p);  
 }  
 **finally** {  
 firstWriter.close();  
 secondWriter.close();  
 }  
 }  
  
 **public static void** main(String[] args) {  
 **try** {  
 *sortDoubles*(**"in.txt"**, **"out1.txt"**, **"out2.txt"**);  
 }  
 **catch** (IntegerValueException e) {  
 e.printStackTrace();  
 System.***err***.println(**"Wrong value: "** + e.getWrongValue());  
 }  
 **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 **catch** (InputMismatchException e) {  
 e.printStackTrace();  
 }  
 }  
  
}

*Файл in.txt(містить вхідні дані)*

2 9 3 1 9 8 5

*Файл out1.txt(сортування за зменшенням та за збільшенням )*

1 2 3 5 8 9 9   
9 9 8 5 3 2 1

*Файл out2.txt(сума цифр)*

37

### 1.3 Реалізація серіализації й десеріализації

*Код програми 1.3:*

**Студент**

**package** lab2.seranddes;  
  
**import** java.io.Serializable;  
  
**public class** Student **implements** Serializable {  
 **private static final long *serialVersionUID*** = -6755942443306500892L;  
 **private** String **name**;  
 **private int age**;  
  
 **public** Student(String name, **int** age) {  
 **this**.**name** = name;  
 **this**.**age** = age;  
 }  
  
 **public** Student(){  
 }  
  
 **public** String getName() {  
 **return name**;  
 }  
  
 **public void** setName(String name) {  
 **this**.**name** = name;  
 }  
  
  
 **public int** getAge() {  
 **return age**;  
 }  
  
 **public void** setAge(**int** age) {  
 **this**.**age** = age;  
 }  
  
}

**Академічна група**

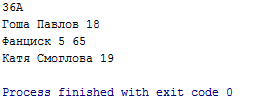
**package** lab2.seranddes;  
  
**import** java.io.Serializable;  
  
**public class** AcademGroup **implements** Serializable {  
 **private static final long *serialVersionUID*** = 8433147861334322335L;  
 **private** String **name**;  
 **private** Student[] **students**;  
  
 **public** AcademGroup(String name, Student... students) {  
 **this**.**name** = name;  
 **this**.**students** = students;  
 }  
  
 **public** AcademGroup() {  
  
 }  
  
 **public** String getName() {  
 **return name**;  
 }  
  
 **public void** setName(String name) {  
 **this**.**name** = name;  
 }  
  
 **public** Student[] getStudents() {  
 **return students**;  
 }  
  
 **public void** setStudents(Student... students) {  
 **this**.**students** = students;  
 }  
  
}

**Бінарна серіализація та десеріалізація**

**package** lab2.seranddes;  
  
**import** java.io.\*;  
  
**public class** DataSerialization {  
 **public static void** main(String[] args) {  
 AcademGroup c = **new** AcademGroup(**"36А"**,  
 **new** Student(**"Гоша Павлов"**, 18),  
 **new** Student(**"Фанциск 5"**, 65),  
 **new** Student(**"Катя Смоглова"**, 19)  
 );  
 **try** (ObjectOutputStream out = **new** ObjectOutputStream(**new** FileOutputStream(**"AcademGroups.dat"**))) {  
 out.writeObject(c);  
 }  
 **catch** (IOException e) {  
 e.printStackTrace();  
 };  
 }  
}

**package** lab2.seranddes;  
  
**import** java.io.\*;  
  
**public class** DataDeserialization {  
  
 **public static void** main(String[] args) **throws** ClassNotFoundException {  
 **try** (ObjectInputStream in = **new** ObjectInputStream(**new** FileInputStream(**"AcademGroups.dat"**))) {  
 AcademGroup academgroup = (AcademGroup) in.readObject();  
 System.***out***.println(academgroup.getName());  
 **for** (Student c : academgroup.getStudents()) {  
 System.***out***.println(c.getName() + **" "** + c.getAge());  
 }  
 }  
 **catch** (IOException e) {  
 e.printStackTrace();  
 };  
 }  
  
}

*Виведення даних у консольне вікно:*



### Серіализація та десеріалізація в XML

**package** lab2.seranddes;  
  
**import** java.beans.XMLEncoder;  
**import** java.io.\*;  
  
  
**public class** XMLSerialization {  
  
 **public static void** main(String[] args) {  
 AcademGroup c = **new** AcademGroup();  
 c.setName(**"36A"**);  
 Student a = **new** Student();  
 a.setName(**"Василий Волсиков"**);  
 a.setAge(18);  
 Student b = **new** Student();  
 b.setName(**"Юля Попова"**);  
 b.setAge(20);  
 c.setStudents(a, b);  
  
 **try** (XMLEncoder xmlEncoder = **new** XMLEncoder(**new** FileOutputStream(**"AcademGroup.xml"**))) {  
 xmlEncoder.writeObject(a);  
 xmlEncoder.writeObject(b);  
 xmlEncoder.writeObject(c);  
 xmlEncoder.flush();  
 }  
 **catch** (IOException e) {  
 e.printStackTrace();  
 }  
  
 }  
  
}

**package** lab2.seranddes;  
  
**import** java.beans.XMLDecoder;  
**import** java.io.\*;  
  
**public class** XMLDeserialization {  
  
 **public static void** main(String[] args) {  
 **try** (XMLDecoder xmlDecoder = **new** XMLDecoder(**new** FileInputStream(**"AcademGroup.xml"**))) {  
 Student a = (Student) xmlDecoder.readObject();  
 Student b = (Student) xmlDecoder.readObject();  
 AcademGroup c = (AcademGroup) xmlDecoder.readObject();  
 System.***out***.println(c.getName());  
 **for** (Student j : c.getStudents()) {  
 System.***out***.println(j.getName() + **" "** + j.getAge());  
 }  
 }  
 **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
}

*Файл AcademGroup.xml*

*<?***xml version="1.0" encoding="UTF-8"***?>*<**java version="1.8.0\_144" class="java.beans.XMLDecoder"**>  
 <**object class="lab2.seranddes.Student" id="Student0"**>  
 <**void property="age"**>  
 <**int**>18</**int**>  
 </**void**>  
 <**void property="name"**>  
 <**string**>Василий Волсиков</**string**>  
 </**void**>  
 </**object**>  
 <**object class="lab2.seranddes.Student" id="Student1"**>  
 <**void property="age"**>  
 <**int**>20</**int**>  
 </**void**>  
 <**void property="name"**>  
 <**string**>Юля Попова</**string**>  
 </**void**>  
 </**object**>  
 <**object class="lab2.seranddes.AcademGroup"**>  
 <**void property="name"**>  
 <**string**>36A</**string**>  
 </**void**>  
 <**void property="students"**>  
 <**array class="lab2.seranddes.Student" length="2"**>  
 <**void index="0"**>  
 <**object idref="Student0"**/>  
 </**void**>  
 <**void index="1"**>  
 <**object idref="Student1"**/>  
 </**void**>  
 </**array**>  
 </**void**>  
 </**object**>  
</**java**>

*Виведення даних у консольне вікно:*



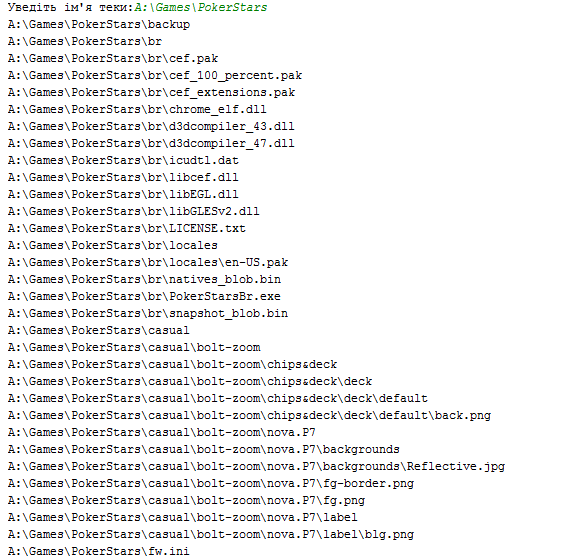
### 1.4  Список файлів усіх підкаталогів

*Код програми 1.4:*

**package** lab2.list;  
**import** java.io.File;  
**import** java.io.FilenameFilter;  
**import** java.io.IOException;  
**import** java.util.Scanner;  
  
**public class** ListFile {  
 **public static void** directory(String dirName) **throws** IOException, WrongArgumentException {  
 File dir = **new** File(dirName);  
 **if** (!dir.exists()) {  
 **throw new** WrongArgumentException();  
 }  
 **if** (!dir.isDirectory()) {  
 **return**;  
 }  
 File[] list = dir.listFiles();  
 **for**(File file : list) {  
 System.***out***.println(file.getCanonicalPath());  
 *directory*(file+**""**);  
 }  
  
 }  
 **public static void** main(String[] args) **throws** IOException, WrongArgumentException {  
 Scanner scanner = **new** Scanner(System.***in***);  
 System.***out***.print(**"Уведіть ім\'я теки:"**);  
 String dirName = scanner.next();  
 **try** {  
 *directory*(dirName);  
 }  
 **catch** (WrongArgumentException e){  
 e.wrong();  
 }  
 }  
}

**package** lab2.list;  
  
**public class** WrongArgumentException **extends** Throwable {  
 **public void** wrong(){  
 System.***out***.println(**"Не существует такая тека"**);  
 }  
}

*Виведення частки результату у консольне вікно:*



***Висновок:***

У даній роботі я за допомогою мови програмування Java та використовуючи попередній досвід я розробив різного роду програми, під час розробки яких я навчився працювати з винятками та файлами в Java