Державний вищий навчальний заклад

«Прикарпатський національний університет імені Василя Стефаника»

Кафедра комп’ютерних наук та інформаційних систем

**ЛАБОРАТОРНА РОБОТА №3**

з предмету «Програмування Java»

Тема: «Потоки введення-виведення»

Виконав:

студент групи КН-2

Здендяк В.Б.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_2020р

ППрийняв:

к.т.н. доц. Горєлов В.О.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_2020р

Івано-Франківськ

2020

**Мета**: сформувати теоретичний базис та навички практичного використання потоків Java.

**Завдання**: опрацювати поняття потоку. Ознайомитися та навчитися використовувати поширені класи потоків вводу-виведення Java: буферизовані та небуферизовані. Навчитися використовувати комбінації потоків вводу-виведення.

1. Напишіть клас-модель Model, що містить id, вік людини, її зріст, ім’я та прізвище.

Id — int, число у діапазоні від 1 до 100000. Вік — double, зріст — int у см, ім’я та прізвище — String.

2. Створіть колекцію посилань на об’єкти класу Model, заповніть її довільним чином 10 елементами.

3. Відсортуйте елементи колекції (див. матеріали до лабораторної роботи по масивах) за полем id.

4. Запишіть вміст колекції до текстового файла із розширенням \*.txt. Дані оформіть у вигляді таблички. Скористайтеся символами “|” та “-”.

При перегляді у редакторі з моноширинним шрифтом отримаємо таблицю зі стовпчиками з однаковою шириною (передбачте вирівнювання за правою межею при записі до файла).

Запишіть даний файл з кодуванням UTF-8.

5. Прочитайте створений файл (програмно) і заповніть колекцію елементів типу Model. Виведіть у консоль інформацію, що містить файл, у вигляді аналогічної таблиці..

**Git-hub:** <https://github.com/vzdendyak/Java-labs/tree/Lab6-IO_Streams>

**Код:**

**Файл Main.java**

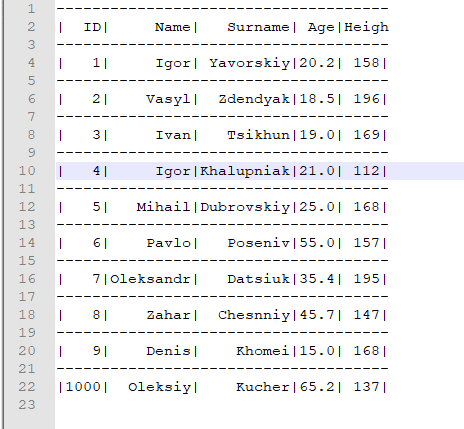
package com.company;  
  
import java.io.\*;  
import java.nio.charset.StandardCharsets;  
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.List;  
  
public class Main {  
  
 public static void main(String[] args) {  
 // заповнюємо масив моделей  
 Model[] userList = new Model[10];  
 userList[0] = new Model(2, 18.5, 196, "Vasyl", "Zdendyak");  
 userList[1] = new Model(3, 19, 169, "Ivan", "Tsikhun");  
 userList[2] = new Model(1, 20.2, 158, "Igor", "Yavorskiy");  
 userList[3] = new Model(4, 21, 112, "Igor", "Khalupniak");  
 userList[4] = new Model(5, 25, 168, "Mihail", "Dubrovskiy");  
 userList[5] = new Model(7, 35.4, 195, "Oleksandr", "Datsiuk");  
 userList[6] = new Model(8, 45.7, 147, "Zahar", "Chesnniy");  
 userList[7] = new Model(6, 55, 157, "Pavlo", "Poseniv");  
 userList[8] = new Model(1000, 65.2, 137, "Oleksiy", "Kucher");  
 userList[9] = new Model(9, 15, 168, "Denis", "Khomei");  
 // сортуємо їх  
 Arrays.*sort*(userList);  
 // Операція запису у файл, передаємо масив і цифру 1/2. 1 - запис у файл, 2 - вивід в консоль  
 *writeOperation*(userList, 1);  
 // Отримуємо масив обєктів з файлу  
 var users = *readFromFile*();  
 // Переводимо List в Array  
 Model[] arrayUserList = new Model[users.size()];  
 users.toArray(arrayUserList);  
 // Операція виводу в консоль, передаємо масив і цифру 1/2. 1 - запис у файл, 2 - вивід в консоль  
 *writeOperation*(arrayUserList, 2);  
 }  
  
 public static void writeOperation(Model[] users, int writeType) {  
 try {  
 // Змінні для збереження максимальної довжини стовпця  
 int idLenghtMax = 0;  
 int ageLenghtMax = 0;  
 int heightLenghtMax = 0;  
 int nameLenghtMax = 0;  
 int surnameLenghtMax = 0;  
 int lineLenght;  
 // Змінні для збереження даних стовпця  
 String id;  
 String age;  
 String height;  
 String firstName;  
 String lastName;  
 // Об\*єкт який записує у файл  
 OutputStreamWriter writer;  
 // Для кожної моделі перебираємо її поля, і дізнаємось найдовше значення стопця по якому будуть рівнятись інші  
 for (Model tempModel : users) {  
 // дістаємо значення з моделі  
 int idLenght = Integer.*toString*(tempModel.getId()).length();  
 int ageLenght = Double.*toString*(tempModel.getAge()).length();  
 int heightLenght = Integer.*toString*(tempModel.getHeight()).length();  
 int firstNameLenght = tempModel.getFirstName().length();  
 int lastNameLenght = tempModel.getLastName().length();  
 // порівнюємо значення з попереднім збереженим (Схоже на сортування бульбашкою)  
 if (idLenght > idLenghtMax) idLenghtMax = idLenght;  
 if (ageLenght > ageLenghtMax) ageLenghtMax = ageLenght;  
 if (heightLenght > heightLenghtMax) heightLenghtMax = heightLenght;  
 if (firstNameLenght > nameLenghtMax) nameLenghtMax = firstNameLenght;  
 if (lastNameLenght > surnameLenghtMax) surnameLenghtMax = lastNameLenght;  
 }  
 // Дізнаємось максимальну довжину рядка  
 lineLenght = idLenghtMax + ageLenghtMax + heightLenghtMax + nameLenghtMax + surnameLenghtMax + 7;  
 // Якщо запис у файл - то відкриваємо файл і перезаписуємо весь вміст  
 if (writeType == 1) {  
 writer = new OutputStreamWriter(new FileOutputStream("info.txt"), StandardCharsets.*UTF\_8*);  
 }  
 // Якщо запис у косноль - то відкриваємо файл з параметром append:true щоб не перезаписати весь вміст  
 else {  
 writer = new OutputStreamWriter(new FileOutputStream("info.txt", true), StandardCharsets.*UTF\_8*);  
 }  
 for (int i = -1; i < users.length; i++) {  
 // Починаємо з -1, щоб записати нульовий рядок з позначеннями, якщо більше то записуємо дані з обєкту  
 if (i == -1) {  
 id = "ID";  
 age = "Age";  
 height = "Height";  
 firstName = "Name";  
 lastName = "Surname";  
 } else {  
 id = Integer.*toString*(users[i].getId());  
 age = Double.*toString*(users[i].getAge());  
 height = Integer.*toString*(users[i].getHeight());  
 firstName = users[i].getFirstName();  
 lastName = users[i].getLastName();  
 }  
 // Створюємо розділювач  
 String strPerson = "|";  
  
 // Підрівнюємо кожне значення до відповідної довжини стовпця  
 for (int t = 0; t < idLenghtMax - id.length(); t++)  
 strPerson += " ";  
 strPerson += id + "|";  
 for (int t = 0; t < nameLenghtMax - firstName.length(); t++)  
 strPerson += " ";  
 strPerson += firstName + "|";  
 for (int t = 0; t < surnameLenghtMax - lastName.length(); t++)  
 strPerson += " ";  
 strPerson += lastName + "|";  
 for (int t = 0; t < ageLenghtMax - age.length(); t++)  
 strPerson += " ";  
 strPerson += age + "|";  
 for (int t = 0; t < heightLenghtMax - height.length() + 1; t++)  
 strPerson += " ";  
 strPerson += height + "|";  
  
 // Якщо запис у файл то робимо розділювач ---- після кожного запису і перехід на новий рядок  
 if (writeType == 1) {  
 for (int j = 0; j < lineLenght; j++) {  
 writer.write('-');  
 }  
 writer.write('\n');  
 for (int t = 0; t < lineLenght; t++) {  
 writer.write(strPerson.charAt(t));  
 }  
 writer.write('\n');  
 }  
 // Якщо запис у консоль то таке саме  
 else {  
 for (int j = 0; j < lineLenght; j++) {  
 System.*out*.print('-');  
 }  
 System.*out*.print('\n');  
 for (int t = 0; t < lineLenght; t++) {  
 System.*out*.print(strPerson.charAt(t));  
 }  
 System.*out*.println();  
 }  
 }  
 writer.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public static List<Model> readFromFile() {  
 List<Model> res = new ArrayList<Model>();  
 try {  
 // Відкриваємо потік для зчитування з файлу  
 FileInputStream is = new FileInputStream("info.txt");  
 InputStreamReader reader = new InputStreamReader(is, StandardCharsets.*UTF\_8*);  
 // Створюємо змінні шоб утримувати значення кожного стовпців які зчитаємо, для створення обєкта  
 int tempId = 0, tempHeight = 0;  
 String tempFirstName = null, tempLastName = null;  
 double tempAge = 0;  
 char symbol;  
 int counter = 0;  
 // Рядок який містить поточне зчитане значення від одного символу | до іншого |  
 String tempString = "";  
 // Перескакуємо перші 114 символів, це наш верхній рядок з описом  
 for (int i = 0; i < 114; i++)  
 reader.read();  
 // Зчитаний символ це не кінець файлу, то читаємл  
 while ((symbol = (char) reader.read()) != '\uFFFF') {  
 // Якщо символ | то витягуємо зчитані дані  
 if (symbol == '|') {  
 try {  
 // Обрізаємо лишні пробіли  
 tempString = tempString.trim();  
 // Беремо остачу від ділення на 5, щоб знати який це з наших 5 стовпців : Id, Name, Surname, Age, Height і поступово по них ітеруватись  
 int tempCounter = counter % 5;  
 // Визначаємо що це за поле, парсимо і записуємо в створенні вверху змінні  
 switch (tempCounter) {  
 case 0:  
 tempId = Integer.*parseInt*(tempString);  
 break;  
 case 1:  
 tempFirstName = tempString;  
 break;  
 case 2:  
 tempLastName = tempString;  
 break;  
 case 3:  
 tempString = tempString.replace(',', '.');  
 tempAge = Double.*parseDouble*(tempString);  
 break;  
 case 4:  
 tempHeight = Integer.*parseInt*(tempString);  
 break;  
 }  
 } catch (Exception e) {  
  
 }  
 // Якщо в нас НЕ пустий рядок або рядок який НЕ містить - розділювач, то збільшуємо лічильник і йдем далі.  
 if (!tempString.equals("") && !tempString.contains("-"))  
 counter++;  
 // Обнуляємо зчитані дані  
 tempString = "";  
 }  
 // Якщо дойшли до кінця рядка то записуємо все що зчитали в обєкт , і додаємо його в масив  
 else if (symbol == '\n') {  
 if (tempId != 0 && tempAge != 0 && tempHeight != 0 && tempFirstName != null && tempLastName != null) {  
 Model tempModel = new Model(tempId, tempAge, tempHeight, tempFirstName, tempLastName);  
 res.add(tempModel);  
 }  
 }  
 // Якщо розділювач то пропускаємо 37 байт(Довжина розділювача) і йдем далі  
 else if (symbol == '-') {  
 for (int i = 0; i < 37; i++) reader.read();  
  
 }  
 // Якщо все ок, то записуємо в наш рядок по одному байту, і коли ми дойдемо до | то з цього рядка зчитаємо що записали  
 else {  
 tempString += symbol;  
 }  
 }  
 reader.close();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return res;  
 }  
  
}

**Файл Model.java**

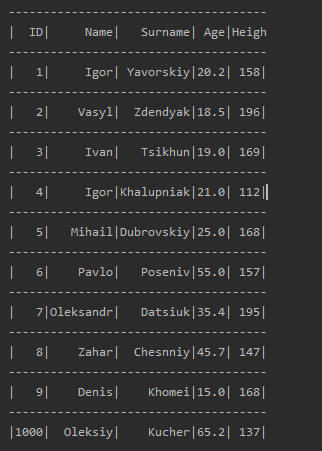
package com.company;  
  
public class Model implements Comparable<Model>{  
 private int id;  
 private double age;  
 private int height;  
 private String firstName;  
 private String lastName;  
  
 Model(int id, double age, int height, String name, String surname){  
 this.id=id;  
 this.age=age;  
 this.height=height;  
 this.firstName=name;  
 this.lastName=surname;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public double getAge() {  
 return age;  
 }  
  
 public int getHeight() {  
 return height;  
 }  
  
 public String getFirstName() {  
 return firstName;  
 }  
  
 public String getLastName() {  
 return lastName;  
 }  
  
  
 @Override  
 public int compareTo(Model model) {  
 return this.id-model.id;  
 }  
}

**Результати виконання роботи**

Результат запису у файл:



Результат зчитування з файлу і переводу в консоль:



**Висновки**: В цій лабораторній роботі було опрацьовано та набуто базові навички роботи та використання потоків Java.

Як було досліджено, Потік – це об’єкт з якого можна зчитати або в нього записати послідовність байтів.   
 В Java для роботи з потоками існує багато різних класів. Зокрема

FileInputStream, FileOutputStream, BufferedInput/OutputStream. Як було досліджено, потоки поділяються на буферизовані та небуферизовані.

Буферизовані доцільно використовувати наприклад коли ми записуємо часто невеликі обсяги даних.

Було досліджено роботу з класів OutputStreamWriter та Input та створено програму яка вміє записувати у файл дані з масиву у вигляді таблиці, вирівнюючи по правому краю, а також зчитувати ці дані з файлу і переводити їх у консоль.