Міністерство освіти і науки України

Національний університет «Києво-Могилянська академія»

кафедра математики



**Звіт із лабораторної роботи 1**

**Виконали:** студенти 1 курсу

Тітарчук Вероніка Олександрівна

Дзьоба Анастасія Юріївна

**Викладач:**

ст. викл. Кирієнко Оксана Валентинівна

Київ – 2020

Зміст

[Постановка задачі 2](#_Toc3152615)

[Розподіл ролей у групі 3](#_Toc3152616)

[Опис усіх реалізованих можливостей 4](#_Toc3152617)

[Структура програми з описом усіх написаних класів 5](#_Toc3152618)

[Структура даних 6](#_Toc3152619)

[Вирішення основних задач 7](#_Toc3152620)

[Проблеми в роботі 7](#_Toc3152621)

[Висновки 8](#_Toc3152622)

# Постановка задачі

**Завдання для лабораторної**

Лабораторна робота 1

Написати програму, що буде формувати список студентів та викладачів університету НаУКМА.

Відповідно мають бути реалізовані такі можливості роботи, як:

1. Створити/видалити/редагувати факультет.
2. Створити/видалити/редагувати кафедру факультета.
3. Додати/видалити/редагувати студента/викладача до кафедри.
4. Знайти студента/викладача за ПІБ, курсом або групою.
5. Вивести всіх студентів впорядкованих за курсами.
6. Вивести всіх студентів/викладачів факультета впорядкованих за алфавітом.
7. Вивести всіх студентів кафедри впорядкованих за курсами.
8. Вивести всіх студентів/викладачів кафедри впорядкованих за алфавітом.
9. Вивести всіх студентів кафедри вказаного курсу.
10. Вивести всіх студентів кафедри вказаного курсу впорядкованих за алфавітом.

Вимоги:

* Повинні бути реалізовані усілякі можливі варіанти захисту від невірного введення даних, або заборонених дій.
* При написанні програми необхідно обов’язково використовувати об'єкти і обмін даними між ними.
* Продумати ієрархію класів.
* Вся інформація вводиться з клавіатури.
* Для роботи користувача повинно пропонуватися меню з набором можливих дій.
* Робота може виконуватися в групі, максимум з 2 студентів.
* До роботи має бути доданий звіт про виконання лабораторної роботи з описом усіх написаних класів, а також реалізованих можливостей.
* При груповій роботі до звіта має бути доданий чіткий розподіл функцій та повноваженнь, що були реалізовані кожним учасником групи окремо.

# Розподіл ролей у групі

Дзьоба Анастасія:

* Продумування ієрархії класів та їх методів
* Реалізація основних методів роботи із студентами та викладачами
* Меню
* Інтерфейс користувача, ввід вивід даних
* Javadocs
* Документація

Тітарчук Вероніка:

* Продумування алгоритму сортування
* Тайм менеджент, та планування виконання проекту
* Реалізація основних методів роботи із студентами та викладачами
* Реалізація усіх видів сортування і виведення результатів
* Javadocs
* Документація

# Опис усіх реалізованих можливостей

Ви можете

1.1. Створити факультет

1.2. Видалити факультет

1.3. Змінити назву факультету

Редагувати факультет:

2.1. Створити кафедру

2.2. Видалити кафедру

2.3. Змінити назву кафедру

Редагувати кафедру:

2.1. Створити студента

2.2. Видалити студента

2.3. Змінити ім’я/прізвище/по батькові/курс/групу кафедру

2.1. Створити викладача

2.2. Видалити викладача

2.3. Змінити ім’я/прізвище/по батькові/курс/групу викладача

Пошук та сортування

3.1. Знайти студента/викладача за ПІБ, курсом або групою.

3.2. Вивести всіх студентів впорядкованих за курсами.

3.3. Вивести всіх студентів/викладачів факультета впорядкованих за алфавітом.

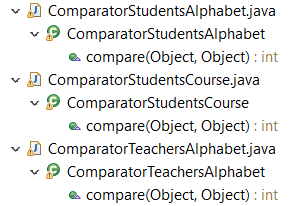
3.4. Вивести всіх студентів кафедри впорядкованих за курсами.

3.5. Вивести всіх студентів/викладачів кафедри впорядкованих за алфавітом.

3.6. Вивести всіх студентів кафедри вказаного курсу.

3.7. Вивести всіх студентів кафедри вказаного курсу впорядкованих за алфавітом.

# Структура програми з описом усіх написаних класів

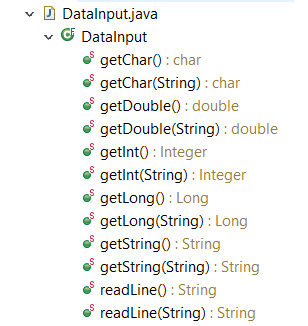


Класи для сортування

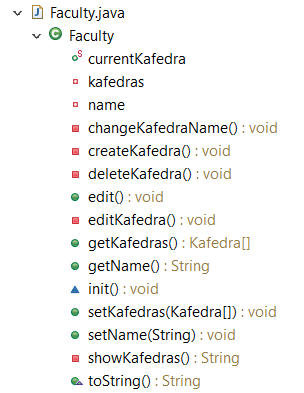
Студентів за алфавітом

Студентів за курсом

Викладачів за Алфавітом



Клас для зчитування введених символів із клавіатури

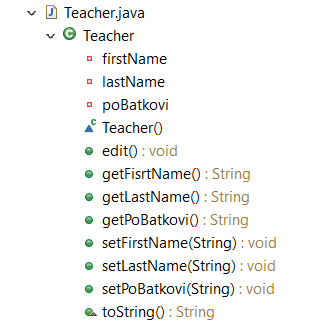


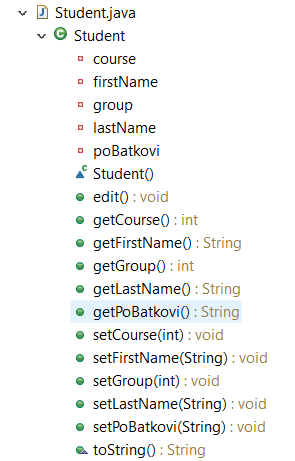
Клас для роботи із Факультетом та його кафедрами



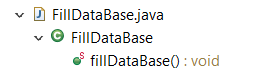
Клас для роботи із кафедрою та її студентами і викладачами

Клас для роботи із викладачем

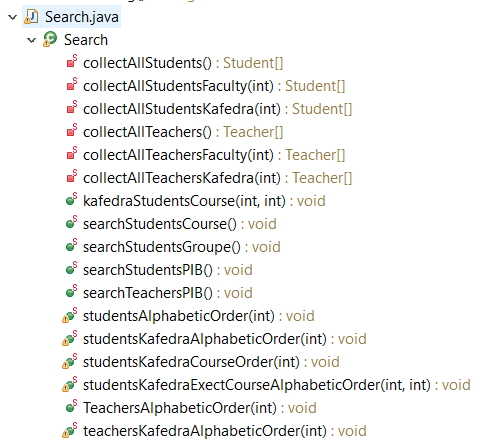




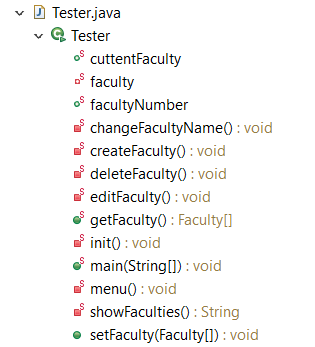
Клас для роботи зі студентом



Клас для заповнення бази даними про студентів із викладачів



Клас для пошуку та сортування викладачів і студентів



Клас для тестування роботи проекту

# Структура даних

**Tester**

Faculty[] faculty

public void man(String[] args)

**Faculty**

Kafedra[] kafedras

**ComparatorStudentsAlphabet**

**ComparatorTeachersAlphabet**

**Kafedra**

Teacher teachers[]

Student students[]

**ComparatorStudentsCourse**

FillDataBase

**Lecturer**

String lastName, firstName, poBatkovi

**Student**

String lastName, firstName, poBatkovi

int course,group

# Вирішення основних задач

Зрозуміти завдання, та сформувати чітку ієрархію класів та їх методи. Рішенням було - спочатку намалювати схему класі на папері та розписати їх методи, усвідомити який метод якого класу де і коли потрібно викликати.

Тяжким завданням стало створення зрозумілого і зручного меню для користувача. Рішенням було зробити текстовий інтерфейс у консолі, опції вибираються шляхом введення номерів опції в консоль із клавіатури.

# Проблеми в роботі

Спочатку було тяжко зрозуміти, як це все має функціонувати. Але розписавши на папері всі класи і методи, ми зрозуміли. Із сортуванням, спочатку довелося збирати всі необхідні об’єкти а тоді аж сортувати, був використаний власноруч написаний Bubble sort та class Comparator.

# Висновки

В ході цієї лабораторної роботи ми навчилися проектувати ієрархію класів, продумувати їх методи і структури даних. Також ми покращили навички комунікації, soft skill, та навички безпосередньо програмування hard skills.

Ми покращили навички роботи із масивами стрінгів, та дізналися більше про способи і методи сортування.

За нагоди отримані знання в ході цієї лабораторної роботи ми зможемо використати в подальшому житті.

**Код програми:**

public class Tester {

public static int cuttentFaculty;

private static Faculty[] faculty = new Faculty[6];

public static int facultyNumber;

public static void main(String[] args) {

init(); // Creates faculties in the array

FillDataBase.fillDataBase();

menu(); // main menu

}

/\*\*

\* main menu in the project

\*/

private static void menu() {

boolean go = true;

while (go) { // menu

System.out.println("Створити факультет введіть 1");

System.out.println("Переіменувати факультету введіть 2");

System.out.println("Редагувати факультет введіть 3");

System.out.println("Видалити факультет введіть 4");

System.out.println("Переглянути факультети введіть 5");

System.out.println("Для пошуку викладача за ПІБ введіть 6");

System.out.println("Для пошуку студента за ПІБ введіть 7");

System.out.println("Для пошуку стyдента за курсом введіть 8");

System.out.println("Для пошуку студента за групою введіть 9");

System.out.println("Для виходу введіть 0");

int menu = DataInput.getInt();

switch (menu) {

case 0:

go = false;

break;

case 1:

createFaculty();

break;

case 2:

changeFacultyName();

break;

case 3:

editFaculty();

break;

case 4:

deleteFaculty();

break;

case 5:

System.out.println(showFaculties());

break;

case 6:

Search.searchTeachersPIB();

break;

case 7:

Search.searchStudentsPIB();

break;

case 8:

Search.searchStudentsCourse();

case 9:

Search.searchStudentsGroupe();

break;

}

}

}

/\*\*

\* changes faculty name

\*/

private static void changeFacultyName() {

while (true) {

int facultyNumber = DataInput.getInt("Введіть номер факультету:\nНазад - введіть 0");

if (facultyNumber == 0)

break;

else if (facultyNumber > 0 && facultyNumber < faculty.length + 1) {

String newName = "";

try {

newName = DataInput.getString("Введіть нову назву: ");

} catch (Exception e) {

System.out.println("Помилка");

}

faculty[facultyNumber].setName(newName);

System.out.println("Факультет успішно переіменовано!");

} else

System.out.println("Помилка, факультету з таким номером немає");

}

}

/\*\*

\* Deletes faculty with input number;

\*/

private static void deleteFaculty() {

while (true) {

int facultyNumber = DataInput.getInt("Введіть номер факультету:\nНазад - введіть 0");

if (facultyNumber == 0)

break;

else if (facultyNumber > 0 && facultyNumber < faculty.length + 1) {

faculty[facultyNumber - 1] = new Faculty();

faculty[facultyNumber - 1].setName("Новий факультет");

System.out.println("Факультет успішно видалено!");

} else

System.out.println("Помилка, факультету з таким номером немає");

}

}

/\*\*

\* edits faculty

\*/

private static void editFaculty() {

while (true) {

facultyNumber = DataInput.getInt("Введіть номер факультету для редагування\nНазад 0");

if (facultyNumber == 0)

break;

else if (facultyNumber > 0 && facultyNumber < faculty.length + 1) {

cuttentFaculty = facultyNumber-1;

faculty[facultyNumber - 1].edit();

} else

System.out.println("Помилка, факультету з таким номером немає");

}

}

/\*\*

\* creates faculty

\*/

private static void createFaculty() {

while (true) {

int facultyNumber = DataInput.getInt("Введіть номер нового факультету (поточний видаляється)\nНазад 0");

if (facultyNumber == 0)

break;

else if (facultyNumber > 0 && facultyNumber < faculty.length + 1) {

faculty[facultyNumber - 1] = new Faculty();

String facultyName = "";

while (true) { // Для того, щоб не можна було зробити факультет із пустим ім'ям

try {

facultyName = DataInput.getString("Введіть назву факультету");

} catch (Exception e) {

System.out.println("Помилка");

}

if (facultyName.equals(""))

System.out.println("Помилка");

else

break;

}

faculty[facultyNumber - 1].setName(facultyName);

} else

System.out.println("Помилка, факультету з таким номером немає");

}

}

/\*\*

\* @return String with faculties` names

\*/

private static String showFaculties() {

String res = "";

for (int i = 0; i < faculty.length; i++) {

res += (i + 1) + ". " + faculty[i].getName() + "\n";

}

return res;

}

/\*\*

\* Creates faculties in the array

\*/

private static void init() {

for (int i = 0; i < faculty.length; i++) {

faculty[i] = new Faculty();

faculty[i].init();

faculty[i].setName("Новий факультет");

}

for (int i = 0; i < faculty.length; i++) {

for (int j = 0; Tester.getFaculty()[0].getKafedras().length > j; j++) {

Tester.getFaculty()[i].getKafedras()[j].init();

}

}

faculty[1].setName("sfg");

faculty[2].setName("FI");

faculty[3].setName("sfgdfhh");

}

public static Faculty[] getFaculty() {

return faculty;

}

public void setFaculty(Faculty[] faculty) {

Tester.faculty = faculty;

}

}

public class Faculty {

private String name;

private Kafedra[] kafedras = new Kafedra[4];

public static int currentKafedra;

public void edit() {

boolean go = true;

while (go) { // menu

System.out.println("Створити кафедру введіть 1");

System.out.println("Переіменувати факультету введіть 2");

System.out.println("Редагувати кафедру введіть 3");

System.out.println("Видалити кафедрут введіть 4");

System.out.println("Переглянути кафедри введіть 5");

System.out.println("Вивести всіх викладачів факультета впорядкованих за алфавітом введіть 6");

System.out.println("Вивести всіх студентів факультета впорядкованих за алфавітом введіть 7");

System.out.println("Для виходу введіть 0");

int menu = DataInput.getInt();

switch (menu) {

case 0:

go = false;

break;

case 1:

createKafedra();

break;

case 2:

changeKafedraName();

break;

case 3:

editKafedra();

break;

case 4:

deleteKafedra();

break;

case 5:

System.out.println(showKafedras());

break;

case 6:

Search.TeachersAlphabeticOrder(Tester.cuttentFaculty);// 3.3. Вивести всіх викладачів факультета

// впорядкованих за алфавітом.

break;

case 7:

Search.studentsAlphabeticOrder(Tester.cuttentFaculty); // 3.3. Вивести всіх студентів факультета

// впорядкованих за алфавітом.

break;

}

}

}

/\*\*

\* changes name of the kafedra

\* number of kafedra is input from the keyboard

\*/

private void changeKafedraName() {

while (true) {

int kafedraNumber = DataInput.getInt("Введіть номер кафедри:\nНазад - введіть 0");

if (kafedraNumber == 0)

break;

else if (kafedraNumber > 0 && kafedraNumber < kafedras.length + 1) {

String newName = "";

try {

newName = DataInput.getString("Введіть нову назву: ");

} catch (Exception e) {

System.out.println("Помилка");

}

kafedras[kafedraNumber].setName(newName);

System.out.println("Кафедру успішно переіменовано!");

} else

System.out.println("Помилка, кафедри з таким номером немає");

}

}

/\*\*

\* Creates faculties in the array

\*/

void init() {

for (int i = 0; i < kafedras.length; i++)

kafedras[i] = new Kafedra();

kafedras[0].setName("Математика");

kafedras[1].setName("Мультимедія");

kafedras[2].setName("sfgdfhh");

}

/\*\*

\* Deletes kafedra with input number;

\*/

private void deleteKafedra() {

while (true) {

int kafedraNumber = DataInput.getInt("Введіть номер кафедри:\nНазад - введіть 0");

if (kafedraNumber == 0)

break;

else if (kafedraNumber > 0 && kafedraNumber < kafedras.length + 1) {

kafedras[kafedraNumber - 1] = new Kafedra();

System.out.println("Кафедру успішно видалено!");

} else

System.out.println("Помилка, кафедри з таким номером немає");

}

}

/\*\*

\* goes to the kafedra menu

\* number of kafedra is input from the keyboard

\*/

private void editKafedra() {

while (true) {

int kafedraNumber = DataInput.getInt("Введіть номер кафедри для редагування\nНазад 0");

if (kafedraNumber == 0)

break;

else if (kafedraNumber > 0 && kafedraNumber < kafedras.length + 1) {

currentKafedra = kafedraNumber;

kafedras[kafedraNumber - 1].edit();

} else

System.out.println("Помилка, кафедри з таким номером немає");

}

}

/\*\*

\* create kafedra on the input cell in an array

\*/

private void createKafedra() {

while (true) {

int kafedraNumber = DataInput.getInt("Введіть номер нової кафедри (поточна видаляється)\nНазад 0");

if (kafedraNumber == 0)

break;

else if (kafedraNumber > 0 && kafedraNumber < kafedras.length + 1) {

kafedras[kafedraNumber - 1] = new Kafedra();

String kafedraName = "";

while (true) { // Для того, щоб не можна було зробити кафедру із пустим ім'ям

try {

kafedraName = DataInput.getString("Введіть назву кафедри");

} catch (Exception e) {

System.out.println("Помилка");

}

if (kafedraName.equals(""))

System.out.println("Помилка");

else

break;

}

kafedras[kafedraNumber - 1].setName(kafedraName);

} else

System.out.println("Помилка, кафедри з таким номером немає");

}

}

/\*\*

\* @return String with faculties` names

\*/

private String showKafedras() {

String res = "";

for (int i = 0; i < kafedras.length; i++) {

res += (i + 1) + ". " + kafedras[i].getName() + "\n";

}

return res;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public Kafedra[] getKafedras() {

return kafedras;

}

public void setKafedras(Kafedra[] kafedras) {

this.kafedras = kafedras;

}

@Override

public String toString() {

return "Faculty " + name + "Kafedras:\n" + Arrays.toString(kafedras);

}

}

import java.util.Arrays;

public class Kafedra {

private String name;

private Teacher teachers[] = new Teacher[5];

private Student students[] = new Student[15];

/\*\*

\* main menu in the kafedra

\*/

public void edit() {

boolean goMainMenu = true;

while (goMainMenu) { // menu

System.out.println("Меню студентів 1");

System.out.println("Меню викладачів 2");

System.out.println("Для виходу введіть 0");

int menu = DataInput.getInt();

switch (menu) {

case 0:

goMainMenu = false;

break;

case 1:

editStudents();

break;

case 2:

editTeachers();

break;

}

}

}

/\*\*

\* main menu for working with students in the kafedra

\*/

private void editTeachers() {

boolean go = true;

while (go) { // menu

System.out.println("Створити викладача введіть 1");

System.out.println("Редагувати викладача введіть 2");

System.out.println("Видалити викладачат введіть 3");

System.out.println("Переглянути викладачів введіть 4");

System.out.println("Вивести всіх викладачів кафедри впорядкованих за алфавітом введіть 5");

System.out.println("Для виходу введіть 0");

int menu = DataInput.getInt();

switch (menu) {

case 0:

go = false;

break;

case 1:

createTeacher();

break;

case 2:

editTeacher();

break;

case 3:

deleteTeacher();

break;

case 4:

showTeachers();

break;

case 5:

Search.teachersKafedraAlphabeticOrder(Faculty.currentKafedra);

}

}

}

/\*\*

\* Deletes Teacher with input number;

\*/

private void deleteTeacher() {

while (true) {

int TeacherNumber = DataInput.getInt("Введіть номер викладача:\nНазад - введіть 0");

if (TeacherNumber == 0)

break;

else if (TeacherNumber > 0 && TeacherNumber < teachers.length + 1) {

teachers[TeacherNumber - 1] = new Teacher();

System.out.println("викладача успішно видалено!");

} else

System.out.println("Помилка, викладача з таким номером немає");

}

}

/\*\*

\* edits teacher

\*/

private void editTeacher() {

while (true) {

int TeacherNumber = DataInput.getInt("Введіть номер викладача для редагування\nНазад 0");

if (TeacherNumber == 0)

break;

else if (TeacherNumber > 0 && TeacherNumber < teachers.length + 1) {

teachers[TeacherNumber - 1].edit();

} else

System.out.println("Помилка, викладача з таким номером немає");

}

}

/\*\*

\* creates teacher with the input information

\*/

private void createTeacher() {

while (true) {

int TeacherNumber = DataInput.getInt("Введіть номер нового викладача (поточна видаляється)\nНазад 0");

if (TeacherNumber == 0)

break;

else if (TeacherNumber > 0 && TeacherNumber < teachers.length + 1) {

teachers[TeacherNumber - 1] = new Teacher();

///////////////////////

String firstName = "";

while (true) { // Для того, щоб не можна було зробити кафедру із пустим ім'ям

try {

firstName = DataInput.getString("Введіть ім'я викладача");

} catch (Exception e) {

System.out.println("Помилка");

}

if (firstName.equals(""))

System.out.println("Помилка");

else

break;

}

teachers[TeacherNumber - 1].setFirstName(firstName);

//////////////////////

String lastName = "";

while (true) { // Для того, щоб не можна було зробити кафедру із пустим ім'ям

try {

lastName = DataInput.getString("Введіть прізвище викладача");

} catch (Exception e) {

System.out.println("Помилка");

}

if (lastName.equals(""))

System.out.println("Помилка");

else

break;

}

teachers[TeacherNumber - 1].setLastName(lastName);

/////////////////////////

String poBatkovi = "";

while (true) { // Для того, щоб не можна було зробити кафедру із пустим ім'ям

try {

poBatkovi = DataInput.getString("Введіть по батькі викладача");

} catch (Exception e) {

System.out.println("Помилка");

}

if (poBatkovi.equals(""))

System.out.println("Помилка");

else

break;

}

teachers[TeacherNumber - 1].setPoBatkovi(poBatkovi);

System.out.println("Викладач успішно створений!");

} else

System.out.println("Помилка, викладача з таким номером немає");

}

}

/\*\*

\* @return String with faculties` names

\*/

private void showTeachers() {

for (int i = 0; i < teachers.length; i++) {

System.out.println(teachers[i].toString());

}

}

/\*\*

\* main menu for working with students in the kafedra

\*/

private void editStudents() {

while (true) { // menu

System.out.println("Створити студента введіть 1");

System.out.println("Редагувати студента введіть 2");

System.out.println("Видалити студентат введіть 3");

System.out.println("Переглянути студентів введіть 4");

System.out.println("Вивести всіх студентів кафедри впорядкованих за алфавітом введіть 5");

System.out.println("Вивести всіх студентів кафедри впорядкованих за курсами введіть 6");

System.out.println("Вивести всіх студентів кафедри вказаного курсу введіть 7");

System.out.println("Вивести всіх студентів кафедри вказаного курсу впорядкованих за алфавітом введіть 8");

System.out.println("Для виходу введіть 0");

int menu = DataInput.getInt();

if (menu == 0)

break;

else if (menu == 1)

createStudent();

else if (menu == 2)

editStudent();

else if (menu == 3)

deleteStudent();

else if (menu == 4)

showStudents();

else if (menu == 5)

Search.studentsKafedraAlphabeticOrder(Faculty.currentKafedra);

else if (menu == 6)

Search.studentsKafedraCourseOrder(Faculty.currentKafedra);

else if (menu == 7)

studentsKafedraExectCourse(Faculty.currentKafedra);

else if (menu == 8)

studentsKafedraExectCourseAlphabeticOrder(Faculty.currentKafedra);

}

}

/\*\*

\* outputs students of the current kafedra in alphabetic order

\* @param currentKafedra kafedra you are working with

\*/

private void studentsKafedraExectCourseAlphabeticOrder(int currentKafedra) {

int course =0;

while(course<1||course>6) {

course = DataInput.getInt("Введіть курс, студентів, якого хочете вивести");

if(course<1||course>6)

System.out.println("Помилка, введіть значення [1-6]");

}

Search.studentsKafedraExectCourseAlphabeticOrder(currentKafedra, course);

}

/\*\*

\* outputs students of the current kafedra of the input course

\* @param currentKafedra kafedra you are working with

\*/

private void studentsKafedraExectCourse(int currentKafedra) {

int course =0;

while(course<1||course>6) {

course = DataInput.getInt("Введіть курс, студентів, якого хочете вивести");

if(course<1||course>6)

System.out.println("Помилка, введіть значення [1-6]");

}

Search.kafedraStudentsCourse(currentKafedra, course);

}

/\*\*

\* Deletes student with input number;

\*/

private void deleteStudent() {

while (true) {

int studentNumber = DataInput.getInt("Введіть номер студента:\nНазад - введіть 0");

if (studentNumber == 0)

break;

else if (studentNumber > 0 && studentNumber < students.length + 1) {

students[studentNumber - 1] = new Student();

System.out.println("Студента успішно видалено!");

} else

System.out.println("Помилка, студента з таким номером немає");

}

}

/\*\*

\* edits student

\*/

private void editStudent() {

while (true) {

int studentNumber = DataInput.getInt("Введіть номер студента для редагування\nНазад 0");

if (studentNumber == 0)

break;

else if (studentNumber > 0 && studentNumber < students.length + 1) {

students[studentNumber - 1].edit();

} else

System.out.println("Помилка, студента з таким номером немає");

}

}

/\*\*

\* creates student with the input information

\*/

private void createStudent() {

while (true) {

int studentNumber = DataInput.getInt("Введіть номер нового студента (поточна видаляється)\nНазад 0");

if (studentNumber == 0)

break;

else if (studentNumber > 0 && studentNumber < students.length + 1) {

students[studentNumber - 1] = new Student();

///////////////////////

String firstName = "";

while (true) { // Для того, щоб не можна було зробити кафедру із пустим ім'ям

try {

firstName = DataInput.getString("Введіть ім'я студента");

} catch (Exception e) {

System.out.println("Помилка");

}

if (firstName.equals(""))

System.out.println("Помилка");

else

break;

}

students[studentNumber - 1].setFirstName(firstName);

//////////////////////

String lastName = "";

while (true) { // Для того, щоб не можна було зробити кафедру із пустим ім'ям

try {

lastName = DataInput.getString("Введіть прізвище студента");

} catch (Exception e) {

System.out.println("Помилка");

}

if (lastName.equals(""))

System.out.println("Помилка");

else

break;

}

students[studentNumber - 1].setLastName(lastName);

/////////////////////////

String poBatkovi = "";

while (true) { // Для того, щоб не можна було зробити кафедру із пустим ім'ям

try {

poBatkovi = DataInput.getString("Введіть по батькі студента");

} catch (Exception e) {

System.out.println("Помилка");

}

if (poBatkovi.equals(""))

System.out.println("Помилка");

else

break;

}

students[studentNumber - 1].setPoBatkovi(poBatkovi);

//////////////////////////////

int course = 0;

while (course < 1 || course > 4) {

course = DataInput.getInt("Введіть курс студента [1-4] : ");

if (course < 1 || course > 4)

System.out.println("Помилка, введіть цифру 1-4");

}

students[studentNumber - 1].setCourse(course);

////////////////////////////////

int group = 0;

while (group < 1 || group > 6) {

group = DataInput.getInt("Введіть групу студента [1-6] : ");

if (group < 1 || group > 6)

System.out.println("Помилка, введіть цифру 1-6");

}

students[studentNumber - 1].setGroup(group);

System.out.println("Студент успішно створений!");

} else

System.out.println("Помилка, студента з таким номером немає");

}

}

/\*\*

\* @return String with faculties` names

\*/

private void showStudents() {

for (int i = 0; i < students.length; i++) {

System.out.println(students[i].toString());

}

}

/\*\*

\* initializes teachers and students

\*/

public void init() {

for (int i = 0; i < teachers.length; i++)

teachers[i] = new Teacher();

for (int i = 0; i < students.length; i++)

students[i] = new Student();

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public Teacher[] getTeachers() {

return teachers;

}

public void setTeachers(Teacher[] teachers) {

this.teachers = teachers;

}

public Student[] getStudents() {

return students;

}

public void setStudents(Student[] students) {

this.students = students;

}

@Override

public String toString() {

return "Kafedra " + name + "\nTeachers:\n" + Arrays.toString(teachers) + "\nStudents:\n"

+ Arrays.toString(students);

}

}

public class Student {

Student(){

setFirstName("");

setLastName("");

setPoBatkovi("");

}

private String firstName;

private String lastName;

private String poBatkovi;

private int course;

private int group;

/\*\*

\* edits student fields

\*/

public void edit() {

while (true) { // menu

System.out.println("Редагувати ім'я 1");

System.out.println("Редагувати прізвище 2");

System.out.println("Редагувати по батькові 3");

System.out.println("Редагувати курс 4");

System.out.println("Редагувати групу 5");

System.out.println("Для виходу введіть 0");

int menu = DataInput.getInt();

if (menu == 0)

break;

if (menu == 1) {

String firstName = "";

while (true) { // Для того, щоб не можна було зробити кафедру із пустим ім'ям

try {

firstName = DataInput.getString("Введіть ім'я студента");

} catch (Exception e) {

System.out.println("Помилка");

}

if (firstName.equals(""))

System.out.println("Помилка");

else

break;

}

setFirstName(firstName);

}

if (menu == 2) {

String lastName = "";

while (true) { // Для того, щоб не можна було зробити кафедру із пустим ім'ям

try {

lastName = DataInput.getString("Введіть прізвище студента");

} catch (Exception e) {

System.out.println("Помилка");

}

if (lastName.equals(""))

System.out.println("Помилка");

else

break;

}

setLastName(lastName);

}

if(menu==3) {

String poBatkovi = "";

while (true) { // Для того, щоб не можна було зробити кафедру із пустим ім'ям

try {

poBatkovi = DataInput.getString("Введіть по батькові студента");

} catch (Exception e) {

System.out.println("Помилка");

}

if (poBatkovi.equals(""))

System.out.println("Помилка");

else

break;

}

setPoBatkovi(poBatkovi);

}

if(menu==4) {

int course = 0;

while (course < 1 || course > 4) {

course = DataInput.getInt("Введіть курс студента [1-4] : ");

if (course < 1 || course > 4)

System.out.println("Помилка, введіть цифру 1-4");

}

setCourse(course);

}

if(menu==5) {

int group = 0;

while (group < 1 || group > 6) {

group = DataInput.getInt("Введіть групу студента [1-6] : ");

if (group < 1 || group > 6)

System.out.println("Помилка, введіть цифру 1-6");

}

setGroup(group);

}

}

}

@Override

public String toString() {

return "Student " + lastName + " " + firstName + " " + poBatkovi + " Course " + course + " Group " + group;

}

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

public String getPoBatkovi() {

return poBatkovi;

}

public void setPoBatkovi(String poBatkovi) {

this.poBatkovi = poBatkovi;

}

public int getCourse() {

return course;

}

public void setCourse(int course) {

this.course = course;

}

public int getGroup() {

return group;

}

public void setGroup(int group) {

this.group = group;

}

}

public class Teacher {

Teacher(){

setFirstName("");

setLastName("");

setPoBatkovi("");

}

private String firstName;

private String lastName;

private String poBatkovi;

/\*\*

\* edits teacher fields

\*/

public void edit() {

while (true) { // menu

System.out.println("Редагувати ім'я 1");

System.out.println("Редагувати прізвище 2");

System.out.println("Редагувати по батькові 3");

System.out.println("Для виходу введіть 0");

int menu = DataInput.getInt();

if (menu == 0)

break;

if (menu == 1) {

String firstName = "";

while (true) { // Для того, щоб не можна було зробити кафедру із пустим ім'ям

try {

firstName = DataInput.getString("Введіть ім'я студента");

} catch (Exception e) {

System.out.println("Помилка");

}

if (firstName.equals(""))

System.out.println("Помилка");

else

break;

}

setFirstName(firstName);

}

if (menu == 2) {

String lastName = "";

while (true) { // Для того, щоб не можна було зробити кафедру із пустим ім'ям

try {

lastName = DataInput.getString("Введіть прізвище студента");

} catch (Exception e) {

System.out.println("Помилка");

}

if (lastName.equals(""))

System.out.println("Помилка");

else

break;

}

setLastName(lastName);

}

if(menu==3) {

String poBatkovi = "";

while (true) { // Для того, щоб не можна було зробити кафедру із пустим ім'ям

try {

poBatkovi = DataInput.getString("Введіть по батькові студента");

} catch (Exception e) {

System.out.println("Помилка");

}

if (poBatkovi.equals(""))

System.out.println("Помилка");

else

break;

}

setPoBatkovi(poBatkovi);

}

}

}

@Override

public String toString() {

return "Teacher " + lastName + " " + firstName + " " + poBatkovi;

}

public String getFisrtName() {

return firstName;

}

public void setFirstName(String fisrtName) {

this.firstName = fisrtName;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

public String getPoBatkovi() {

return poBatkovi;

}

public void setPoBatkovi(String poBatkovi) {

this.poBatkovi = poBatkovi;

}

}

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

public final class DataInput {

/\*\*

\*

\* @return long value of the input string;

\* @throws IOException

\*/

public static Long getLong(String st) throws Exception {

String s = getString(st);

Long value = Long.valueOf(s);

return value;

}

/\*\*

\*

\* @return long value of the input string;

\* @throws IOException

\*/

public static Long getLong() throws Exception {

String s = getString();

Long value = Long.valueOf(s);

return value;

}

/\*\*

\*

\* @return long value of the input string;

\* @throws IOException

\*/

public static char getChar(String st) throws Exception {

String s = getString(st);

return s.charAt(0);

}

/\*\*

\*

\* @return long value of the input string;

\* @throws IOException

\*/

public static char getChar() throws Exception {

String s = getString();

return s.charAt(0);

}

public static double getDouble(String st) {

String s = "";

while (true) {

try {

s = getString(st);

double value = Double.valueOf(s);

return value;

} catch (Exception e) {

System.out.println("Помилка");

}

}

}

public static double getDouble(){

String s = "";

while (true) {

try {

s = getString();

double value = Double.valueOf(s);

return value;

} catch (Exception e) {

System.out.println("Помилка");

}

}

}

/\*\*

\*

\* @return long value of the input string;

\* @throws IOException

\*/

public static Integer getInt(String st) {

String s = "";

while (true) {

try {

s = getString(st);

Integer value = Integer.valueOf(s);

return value;

} catch (Exception e) {

System.out.println("Помилка");

}

}

}

/\*\*

\*

\* @return long value of the input string;

\* @throws IOException

\*/

public static Integer getInt() {

String s = "";

while (true) {

try {

s = getString();

Integer value = Integer.valueOf(s);

return value;

} catch (Exception e) {

System.out.println("Помилка");

}

}

}

/\*\*

\*

\* @return input string;

\* @throws IOException

\*/

public static String readLine(String st) throws Exception {

System.out.println(st);

return new BufferedReader(new InputStreamReader(System.in)).readLine();

}

/\*\*

\*

\* @return input string;

\* @throws IOException

\*/

public static String readLine() throws Exception {

return new BufferedReader(new InputStreamReader(System.in)).readLine();

}

/\*\*

\*

\* @return input string;

\* @throws IOException

\*/

public static String getString(String st) throws Exception {

System.out.println(st);

InputStreamReader isr = new InputStreamReader(System.in);

BufferedReader br = new BufferedReader(isr);

String s = br.readLine();

return s;

}

/\*\*

\*

\* @return input string;

\* @throws IOException

\*/

public static String getString() throws Exception {

InputStreamReader isr = new InputStreamReader(System.in);

BufferedReader br = new BufferedReader(isr);

String s = br.readLine();

return s;

}

}

import java.util.Arrays;

public class Search {

/\*\*

\* outputs teachers with the last name, first name or по батькові that matches

\* the input string

\*/

public static void searchTeachersPIB() {

String search = "";

try {

search = DataInput.getString("Введіть ім'я, прізвище або по батькові викладача: ");

} catch (Exception e) {

System.out.println("Помилка");

}

Teacher[] toSearch = collectAllTeachers();

System.out.println("Результати пошуку: \n");

for (int i = 0; i < toSearch.length; i++) {

if (search.equals(toSearch[i].getFisrtName()) || search.equals(toSearch[i].getLastName())

|| search.equals(toSearch[i].getPoBatkovi()))

System.out.println(toSearch[i].toString());

}

System.out.println();

}

/\*\*

\*

\* @return array with all teachers in all faculties in all kafedras

\*/

private static Teacher[] collectAllTeachers() {

Teacher[] allTeachers = new Teacher[Tester.getFaculty().length \* Tester.getFaculty()[0].getKafedras().length

\* Tester.getFaculty()[0].getKafedras()[0].getTeachers().length];

Faculty[] toColect = Tester.getFaculty().clone();

int number = 0;

for (int i = 0; Tester.getFaculty().length > i; i++) {

for (int j = 0; Tester.getFaculty()[i].getKafedras().length > j; j++) {

for (int t = 0; t < Tester.getFaculty()[i].getKafedras()[j].getTeachers().length; t++) {

allTeachers[number++] = toColect[i].getKafedras()[j].getTeachers()[t];

}

}

}

return allTeachers;

}

/\*\*

\*

\* @param faculty

\* faculty to collect

\* @return array with all teachers in all kafedras

\*/

private static Teacher[] collectAllTeachersFaculty(int faculty) {

Teacher[] allTeachers = new Teacher[Tester.getFaculty()[0].getKafedras().length

\* Tester.getFaculty()[0].getKafedras()[0].getTeachers().length];

Faculty[] toColect = Tester.getFaculty().clone();

int number = 0;

for (int j = 0; Tester.getFaculty()[faculty].getKafedras().length > j; j++) {

for (int t = 0; t < Tester.getFaculty()[faculty].getKafedras()[j].getTeachers().length; t++) {

allTeachers[number++] = toColect[faculty].getKafedras()[j].getTeachers()[t];

}

}

return allTeachers;

}

/\*\*

\*

\* @param kafedra

\* kafedra to collect

\* @return array with all teachers in kafedra

\*/

private static Teacher[] collectAllTeachersKafedra(int kafedra) {

Teacher[] allTeachers = new Teacher[Tester.getFaculty()[0].getKafedras()[0].getTeachers().length];

Kafedra[] toColect = Tester.getFaculty()[Tester.cuttentFaculty].getKafedras().clone();

int number = 0;

for (int i = 0; Tester.getFaculty()[Tester.cuttentFaculty].getKafedras()[kafedra]

.getTeachers().length > i; i++) {

allTeachers[number++] = toColect[kafedra].getTeachers()[i];

}

return allTeachers;

}

/\*\*

\*

\* @param kafedra

\* kafedra to collect

\* @return array with all students in kafedra

\*/

private static Student[] collectAllStudentsKafedra(int kafedra) {

Student[] allStudents = new Student[Tester.getFaculty()[0].getKafedras()[0].getStudents().length];

allStudents = Tester.getFaculty()[Tester.cuttentFaculty].getKafedras()[Faculty.currentKafedra].getStudents()

.clone();

return allStudents;

}

/\*\*

\* outputs students with the last name, first name or по батькові that matches

\* the input string

\*/

public static void searchStudentsPIB() {

String search = "";

try {

search = DataInput.getString("Введіть ім'я, прізвище або по батькові студента: ");

} catch (Exception e) {

System.out.println("Помилка");

}

Student[] toSearch = collectAllStudents();

System.out.println("Результати пошуку: \n");

for (int i = 0; i < toSearch.length; i++) {

if (search.equals(toSearch[i].getFirstName()) || search.equals(toSearch[i].getLastName())

|| search.equals(toSearch[i].getPoBatkovi()))

System.out.println(toSearch[i].toString());

}

System.out.println();

}

/\*\*

\* outputs students with the course that matches the input int

\*/

public static void searchStudentsCourse() {

int searchCourse = -1;

try {

searchCourse = DataInput.getInt("Введіть курс студента: ");

} catch (Exception e) {

System.out.println("Помилка");

}

if (searchCourse < 1 || searchCourse > 4)

System.out.println("Помилка");

else {

Student[] toSearch = collectAllStudents();

System.out.println("Результати пошуку: \n");

for (int i = 0; i < toSearch.length; i++) {

if (searchCourse == toSearch[i].getCourse())

System.out.println(toSearch[i].toString());

}

System.out.println();

}

}

/\*\*

\* outputs students with the group that matches the input int

\*/

public static void searchStudentsGroupe() {

int searchGroup = -1;

try {

searchGroup = DataInput.getInt("Введіть групу студента: ");

} catch (Exception e) {

System.out.println("Помилка");

}

if (searchGroup < 1 || searchGroup > 6)

System.out.println("Помилка");

else {

Student[] toSearch = collectAllStudents();

System.out.println("Результати пошуку: \n");

for (int i = 0; i < toSearch.length; i++) {

if (searchGroup == toSearch[i].getGroup())

System.out.println(toSearch[i].toString());

}

System.out.println();

}

}

/\*\*

\* outputs teachers of the current faculty in alphabetic order

\*

\* @param currentFaculty

\* faculty to take teachers from

\*/

public static void TeachersAlphabeticOrder(int currentFaculty) {

Teacher[] toSortAlphabet = collectAllTeachersFaculty(currentFaculty);

boolean uncomplete = true;

while (uncomplete) {

uncomplete = false;

for (int i = 0; i < toSortAlphabet.length - 1; i++) {

for (int j = 0; j < Math.min(toSortAlphabet[i].getLastName().length(), // довжина

toSortAlphabet[i + 1].getLastName().length()); j++) {

if (toSortAlphabet[i].getLastName().charAt(j) > toSortAlphabet[i].getLastName().charAt(j)) {

Teacher temp = toSortAlphabet[i];

toSortAlphabet[i] = toSortAlphabet[i + 1];

toSortAlphabet[i + 1] = temp;

uncomplete = true;

}

}

}

}

for (int i = 0; i < toSortAlphabet.length; i++)

System.out.println(toSortAlphabet[i].toString());

}

/\*\*

\*

\* @param currentFaculty

\* faculty to work with

\* @return array with all the students of the all kafedras in the current

\* faculty

\*/

private static Student[] collectAllStudentsFaculty(int currentFaculty) {

Student[] allStudents = new Student[Tester.getFaculty()[0].getKafedras().length

\* Tester.getFaculty()[0].getKafedras()[0].getStudents().length];

Faculty[] toColect = Tester.getFaculty().clone();

int number = 0;

for (int j = 0; Tester.getFaculty()[currentFaculty].getKafedras().length > j; j++) {

for (int t = 0; t < Tester.getFaculty()[currentFaculty].getKafedras()[j].getStudents().length; t++) {

allStudents[number++] = toColect[currentFaculty].getKafedras()[j].getStudents()[t];

}

}

return allStudents;

}

/\*\*

\*

\* @param currentFaculty

\* faculty to work with

\* prints array with all the students of the all kafedras in the current

\* faculty in alphabetic order

\*/

public static void studentsAlphabeticOrder(int currentFaculty) {

Student[] toSortAlphabet = collectAllStudentsFaculty(currentFaculty);

ComparatorStudentsAlphabet compStudAlph = new ComparatorStudentsAlphabet();

Arrays.sort(toSortAlphabet, compStudAlph);

for (int i = 0; i < toSortAlphabet.length; i++)

System.out.println(toSortAlphabet[i].toString());

}

/\*\*

\*

\* @param currentKafedra

\* kafedra to work with

\* prints all the teachers of the all kafedras in the current

\* faculty in alphabetic order

\*/

public static void teachersKafedraAlphabeticOrder(int currentKafedra) {

Teacher[] toSortAlphabet = collectAllTeachersKafedra(currentKafedra);

ComparatorTeachersAlphabet compTeachAlph = new ComparatorTeachersAlphabet();

Arrays.sort(toSortAlphabet, compTeachAlph);

for (int i = 0; i < toSortAlphabet.length; i++)

System.out.println(toSortAlphabet[i].toString());

}

/\*\*

\*

\* @param currentKafedra

\* kafedra to work with

\* prints all the students of the current kafedras in the current

\* faculty in alphabetic order

\*/

public static void studentsKafedraAlphabeticOrder(int currentKafedra) {

Student[] toSortAlphabet = collectAllStudentsKafedra(currentKafedra);

ComparatorStudentsAlphabet compStudAlph = new ComparatorStudentsAlphabet();

Arrays.sort(toSortAlphabet, compStudAlph);

for (int i = 0; i < toSortAlphabet.length; i++)

System.out.println(toSortAlphabet[i].toString());

}

/\*\*

\*

\* @param currentFaculty

\* faculty to work with

\* prints all the students of the all kafedras in the current

\* faculty in course order

\*/

public static void studentsKafedraCourseOrder(int currentKafedra) {

Student[] toSortAlphabet = collectAllStudentsKafedra(currentKafedra);

ComparatorStudentsCourse compStudCourse = new ComparatorStudentsCourse();

Arrays.sort(toSortAlphabet, compStudCourse);

for (int i = 0; i < toSortAlphabet.length; i++)

System.out.println(toSortAlphabet[i].toString());

}

/\*\*

\*

\* @param currentFaculty

\* faculty to work with

\* @param course course that you want to print students of

\* prints all the students of the all kafedras in the current

\* faculty of the input course

\*/

public static void kafedraStudentsCourse(int currentKafedra, int course) {

Student[] kafedraStudents = collectAllStudentsKafedra(currentKafedra);

for (int i = 0; i < kafedraStudents.length; i++)

if (kafedraStudents[i].getCourse() == course)

System.out.println(kafedraStudents[i].toString());

}

/\*\*

\* @return all the students of all faculties of all kafedras

\*/

private static Student[] collectAllStudents() {

Student[] allStudents = new Student[Tester.getFaculty().length \* Tester.getFaculty()[0].getKafedras().length

\* Tester.getFaculty()[0].getKafedras()[0].getStudents().length];

Faculty[] toColect = Tester.getFaculty().clone();

int number = 0;

for (int i = 0; Tester.getFaculty().length > i; i++) {

for (int j = 0; Tester.getFaculty()[i].getKafedras().length > j; j++) {

for (int t = 0; t < Tester.getFaculty()[i].getKafedras()[j].getStudents().length; t++) {

allStudents[number++] = toColect[i].getKafedras()[j].getStudents()[t];

}

}

}

return allStudents;

}

/\*\*

\*

\* @param currentFaculty

\* faculty to work with

\* @param course course that you want to print students of

\* prints all the students of the all kafedras in the current

\* faculty of the input course in alphabetic order

\*/

public static void studentsKafedraExectCourseAlphabeticOrder(int currentKafedra, int course) {

Student[] kafedraStudents = collectAllStudentsKafedra(currentKafedra);

ComparatorStudentsAlphabet compStudAlph = new ComparatorStudentsAlphabet();

Arrays.sort(kafedraStudents, compStudAlph);

for (int i = 0; i < kafedraStudents.length; i++)

if (kafedraStudents[i].getCourse() == course)

System.out.println(kafedraStudents[i].toString());

}

}

public class FillDataBase {

/\*\*

\* fills arrays with some information about students and teachers

\*/

public static void fillDataBase() {

Tester.getFaculty()[0].getKafedras()[0].getTeachers()[0].setFirstName("Andriy");

Tester.getFaculty()[0].getKafedras()[0].getTeachers()[0].setLastName("Petrov");

Tester.getFaculty()[0].getKafedras()[0].getTeachers()[0].setPoBatkovi("Andriovych");

Tester.getFaculty()[0].getKafedras()[0].getTeachers()[1].setFirstName("Petro");

Tester.getFaculty()[0].getKafedras()[0].getTeachers()[1].setLastName("Synelskyi");

Tester.getFaculty()[0].getKafedras()[0].getTeachers()[1].setPoBatkovi("Grygorovych");

Tester.getFaculty()[0].getKafedras()[0].getTeachers()[2].setFirstName("Vasyl");

Tester.getFaculty()[0].getKafedras()[0].getTeachers()[2].setLastName("Koratov");

Tester.getFaculty()[0].getKafedras()[0].getTeachers()[2].setPoBatkovi("Andriyovych");

Tester.getFaculty()[0].getKafedras()[0].getTeachers()[3].setFirstName("Anatolyi");

Tester.getFaculty()[0].getKafedras()[0].getTeachers()[3].setLastName("Gogol");

Tester.getFaculty()[0].getKafedras()[0].getTeachers()[3].setPoBatkovi("Tarasovych");

Tester.getFaculty()[0].getKafedras()[0].getTeachers()[4].setFirstName("Inna");

Tester.getFaculty()[0].getKafedras()[0].getTeachers()[4].setLastName("Romanenko");

Tester.getFaculty()[0].getKafedras()[0].getTeachers()[4].setPoBatkovi("Andriivna");

Tester.getFaculty()[0].getKafedras()[1].getTeachers()[0].setFirstName("Maria");

Tester.getFaculty()[0].getKafedras()[1].getTeachers()[0].setLastName("Renchka");

Tester.getFaculty()[0].getKafedras()[1].getTeachers()[0].setPoBatkovi("Tarasivna");

Tester.getFaculty()[0].getKafedras()[1].getTeachers()[1].setFirstName("Oksana");

Tester.getFaculty()[0].getKafedras()[1].getTeachers()[1].setLastName("Bandera");

Tester.getFaculty()[0].getKafedras()[1].getTeachers()[1].setPoBatkovi("Stepanovych");

Tester.getFaculty()[0].getKafedras()[1].getTeachers()[2].setFirstName("Natalia");

Tester.getFaculty()[0].getKafedras()[1].getTeachers()[2].setLastName("Sahaidachna");

Tester.getFaculty()[0].getKafedras()[1].getTeachers()[2].setPoBatkovi("Teodorivna");

Tester.getFaculty()[0].getKafedras()[1].getTeachers()[3].setFirstName("Ksenia");

Tester.getFaculty()[0].getKafedras()[1].getTeachers()[3].setLastName("Petrova");

Tester.getFaculty()[0].getKafedras()[1].getTeachers()[3].setPoBatkovi("Andriovych");

Tester.getFaculty()[0].getKafedras()[1].getTeachers()[4].setFirstName("Marusia");

Tester.getFaculty()[0].getKafedras()[1].getTeachers()[4].setLastName("Kuratova");

Tester.getFaculty()[0].getKafedras()[1].getTeachers()[4].setPoBatkovi("Petrivna");

Tester.getFaculty()[0].getKafedras()[2].getTeachers()[0].setFirstName("Alina");

Tester.getFaculty()[0].getKafedras()[2].getTeachers()[0].setLastName("Levchuk");

Tester.getFaculty()[0].getKafedras()[2].getTeachers()[0].setPoBatkovi("Sviatoslvovych");

Tester.getFaculty()[0].getKafedras()[2].getTeachers()[1].setFirstName("Kate");

Tester.getFaculty()[0].getKafedras()[2].getTeachers()[1].setLastName("Myloslavska");

Tester.getFaculty()[0].getKafedras()[2].getTeachers()[1].setPoBatkovi("Andriovych");

Tester.getFaculty()[0].getKafedras()[2].getTeachers()[2].setFirstName("Andriy");

Tester.getFaculty()[0].getKafedras()[2].getTeachers()[2].setLastName("Lipovych");

Tester.getFaculty()[0].getKafedras()[2].getTeachers()[2].setPoBatkovi("Mykytovych");

Tester.getFaculty()[0].getKafedras()[2].getTeachers()[3].setFirstName("Taras");

Tester.getFaculty()[0].getKafedras()[2].getTeachers()[3].setLastName("Burdovych");

Tester.getFaculty()[0].getKafedras()[2].getTeachers()[3].setPoBatkovi("Youriyovych");

Tester.getFaculty()[0].getKafedras()[2].getTeachers()[4].setFirstName("Volodymyr");

Tester.getFaculty()[0].getKafedras()[2].getTeachers()[4].setLastName("Tarasov");

Tester.getFaculty()[0].getKafedras()[2].getTeachers()[4].setPoBatkovi("Tarasovych");

Tester.getFaculty()[0].getKafedras()[3].getTeachers()[0].setFirstName("Pavlo");

Tester.getFaculty()[0].getKafedras()[3].getTeachers()[0].setLastName("Hudenko");

Tester.getFaculty()[0].getKafedras()[3].getTeachers()[0].setPoBatkovi("Valeriyovych");

Tester.getFaculty()[0].getKafedras()[3].getTeachers()[1].setFirstName("Stepan");

Tester.getFaculty()[0].getKafedras()[3].getTeachers()[1].setLastName("Orlov");

Tester.getFaculty()[0].getKafedras()[3].getTeachers()[1].setPoBatkovi("Oleksiyovych");

Tester.getFaculty()[0].getKafedras()[3].getTeachers()[2].setFirstName("Daria");

Tester.getFaculty()[0].getKafedras()[3].getTeachers()[2].setLastName("Lutsenko");

Tester.getFaculty()[0].getKafedras()[3].getTeachers()[2].setPoBatkovi("Artemovych");

Tester.getFaculty()[0].getKafedras()[3].getTeachers()[3].setFirstName("Artem");

Tester.getFaculty()[0].getKafedras()[3].getTeachers()[3].setLastName("Lubimov");

Tester.getFaculty()[0].getKafedras()[3].getTeachers()[3].setPoBatkovi("Anatoliyovych");

Tester.getFaculty()[0].getKafedras()[3].getTeachers()[4].setFirstName("Anton");

Tester.getFaculty()[0].getKafedras()[3].getTeachers()[4].setLastName("Grytsenko");

Tester.getFaculty()[0].getKafedras()[3].getTeachers()[4].setPoBatkovi("Andriovych");

Tester.getFaculty()[0].getKafedras()[0].getStudents()[0].setFirstName("Andriy");

Tester.getFaculty()[0].getKafedras()[0].getStudents()[0].setLastName("Petrov");

Tester.getFaculty()[0].getKafedras()[0].getStudents()[0].setPoBatkovi("Andriovych");

Tester.getFaculty()[0].getKafedras()[0].getStudents()[0].setCourse(2);

Tester.getFaculty()[0].getKafedras()[0].getStudents()[0].setGroup(1);

Tester.getFaculty()[0].getKafedras()[0].getStudents()[1].setFirstName("Petro");

Tester.getFaculty()[0].getKafedras()[0].getStudents()[1].setLastName("Synelskyi");

Tester.getFaculty()[0].getKafedras()[0].getStudents()[1].setPoBatkovi("Grygorovych");

Tester.getFaculty()[0].getKafedras()[0].getStudents()[1].setCourse(1);

Tester.getFaculty()[0].getKafedras()[0].getStudents()[1].setGroup(6);

Tester.getFaculty()[0].getKafedras()[0].getStudents()[2].setFirstName("Vasyl");

Tester.getFaculty()[0].getKafedras()[0].getStudents()[2].setLastName("Oleksiiv");

Tester.getFaculty()[0].getKafedras()[0].getStudents()[2].setPoBatkovi("Andriyovych");

Tester.getFaculty()[0].getKafedras()[0].getStudents()[2].setCourse(1);

Tester.getFaculty()[0].getKafedras()[0].getStudents()[2].setGroup(5);

Tester.getFaculty()[0].getKafedras()[0].getStudents()[3].setFirstName("Anatolyi");

Tester.getFaculty()[0].getKafedras()[0].getStudents()[3].setLastName("Gogol");

Tester.getFaculty()[0].getKafedras()[0].getStudents()[3].setPoBatkovi("Tarasovych");

Tester.getFaculty()[0].getKafedras()[0].getStudents()[3].setCourse(2);

Tester.getFaculty()[0].getKafedras()[0].getStudents()[3].setGroup(3);

Tester.getFaculty()[0].getKafedras()[0].getStudents()[4].setFirstName("Inna");

Tester.getFaculty()[0].getKafedras()[0].getStudents()[4].setLastName("Romanenko");

Tester.getFaculty()[0].getKafedras()[0].getStudents()[4].setPoBatkovi("Andriivna");

Tester.getFaculty()[0].getKafedras()[0].getStudents()[4].setCourse(4);

Tester.getFaculty()[0].getKafedras()[0].getStudents()[4].setGroup(3);

Tester.getFaculty()[0].getKafedras()[1].getStudents()[0].setFirstName("Maria");

Tester.getFaculty()[0].getKafedras()[1].getStudents()[0].setLastName("Renchka");

Tester.getFaculty()[0].getKafedras()[1].getStudents()[0].setPoBatkovi("Tarasivna");

Tester.getFaculty()[0].getKafedras()[1].getStudents()[0].setCourse(1);

Tester.getFaculty()[0].getKafedras()[1].getStudents()[0].setGroup(2);

Tester.getFaculty()[0].getKafedras()[1].getStudents()[1].setFirstName("Oksana");

Tester.getFaculty()[0].getKafedras()[1].getStudents()[1].setLastName("Bandera");

Tester.getFaculty()[0].getKafedras()[1].getStudents()[1].setPoBatkovi("Stepanovych");

Tester.getFaculty()[0].getKafedras()[1].getStudents()[1].setCourse(3);

Tester.getFaculty()[0].getKafedras()[1].getStudents()[1].setGroup(6);

Tester.getFaculty()[0].getKafedras()[1].getStudents()[2].setFirstName("Natalia");

Tester.getFaculty()[0].getKafedras()[1].getStudents()[2].setLastName("Sahaidachna");

Tester.getFaculty()[0].getKafedras()[1].getStudents()[2].setPoBatkovi("Teodorivna");

Tester.getFaculty()[0].getKafedras()[1].getStudents()[2].setCourse(4);

Tester.getFaculty()[0].getKafedras()[1].getStudents()[2].setGroup(5);

Tester.getFaculty()[0].getKafedras()[1].getStudents()[3].setFirstName("Ksenia");

Tester.getFaculty()[0].getKafedras()[1].getStudents()[3].setLastName("Petrova");

Tester.getFaculty()[0].getKafedras()[1].getStudents()[3].setPoBatkovi("Andriovych");

Tester.getFaculty()[0].getKafedras()[1].getStudents()[3].setCourse(3);

Tester.getFaculty()[0].getKafedras()[1].getStudents()[3].setGroup(2);

Tester.getFaculty()[0].getKafedras()[1].getStudents()[4].setFirstName("Marusia");

Tester.getFaculty()[0].getKafedras()[1].getStudents()[4].setLastName("Tsvetaeva");

Tester.getFaculty()[0].getKafedras()[1].getStudents()[4].setPoBatkovi("Petrivna");

Tester.getFaculty()[0].getKafedras()[1].getStudents()[4].setCourse(2);

Tester.getFaculty()[0].getKafedras()[1].getStudents()[4].setGroup(1);

Tester.getFaculty()[0].getKafedras()[2].getStudents()[0].setFirstName("Alina");

Tester.getFaculty()[0].getKafedras()[2].getStudents()[0].setLastName("Levchuk");

Tester.getFaculty()[0].getKafedras()[2].getStudents()[0].setPoBatkovi("Sviatoslvovych");

Tester.getFaculty()[0].getKafedras()[2].getStudents()[0].setCourse(1);

Tester.getFaculty()[0].getKafedras()[2].getStudents()[0].setGroup(2);

Tester.getFaculty()[0].getKafedras()[2].getStudents()[1].setFirstName("Kate");

Tester.getFaculty()[0].getKafedras()[2].getStudents()[1].setLastName("Mykhayliv");

Tester.getFaculty()[0].getKafedras()[2].getStudents()[1].setPoBatkovi("Andriovych");

Tester.getFaculty()[0].getKafedras()[2].getStudents()[1].setCourse(3);

Tester.getFaculty()[0].getKafedras()[2].getStudents()[1].setGroup(3);

Tester.getFaculty()[0].getKafedras()[2].getStudents()[2].setFirstName("Andriy");

Tester.getFaculty()[0].getKafedras()[2].getStudents()[2].setLastName("Zinchenko");

Tester.getFaculty()[0].getKafedras()[2].getStudents()[2].setPoBatkovi("Mykytovych");

Tester.getFaculty()[0].getKafedras()[2].getStudents()[2].setCourse(2);

Tester.getFaculty()[0].getKafedras()[2].getStudents()[2].setGroup(5);

Tester.getFaculty()[0].getKafedras()[2].getStudents()[3].setFirstName("Taras");

Tester.getFaculty()[0].getKafedras()[2].getStudents()[3].setLastName("Datsenko");

Tester.getFaculty()[0].getKafedras()[2].getStudents()[3].setPoBatkovi("Youriyovych");

Tester.getFaculty()[0].getKafedras()[2].getStudents()[3].setCourse(4);

Tester.getFaculty()[0].getKafedras()[2].getStudents()[3].setGroup(3);

Tester.getFaculty()[0].getKafedras()[2].getStudents()[4].setFirstName("Volodymyr");

Tester.getFaculty()[0].getKafedras()[2].getStudents()[4].setLastName("Buksha");

Tester.getFaculty()[0].getKafedras()[2].getStudents()[4].setPoBatkovi("Tarasovych");

Tester.getFaculty()[0].getKafedras()[2].getStudents()[4].setCourse(2);

Tester.getFaculty()[0].getKafedras()[2].getStudents()[4].setGroup(1);

Tester.getFaculty()[0].getKafedras()[3].getStudents()[0].setFirstName("Pavlo");

Tester.getFaculty()[0].getKafedras()[3].getStudents()[0].setLastName("Sergeev");

Tester.getFaculty()[0].getKafedras()[3].getStudents()[0].setPoBatkovi("Valeriyovych");

Tester.getFaculty()[0].getKafedras()[3].getStudents()[0].setCourse(1);

Tester.getFaculty()[0].getKafedras()[3].getStudents()[0].setGroup(1);

Tester.getFaculty()[0].getKafedras()[3].getStudents()[1].setFirstName("Stepan");

Tester.getFaculty()[0].getKafedras()[3].getStudents()[1].setLastName("Lubimov");

Tester.getFaculty()[0].getKafedras()[3].getStudents()[1].setPoBatkovi("Oleksiyovych");

Tester.getFaculty()[0].getKafedras()[3].getStudents()[1].setCourse(1);

Tester.getFaculty()[0].getKafedras()[3].getStudents()[1].setGroup(6);

Tester.getFaculty()[0].getKafedras()[3].getStudents()[2].setFirstName("Daria");

Tester.getFaculty()[0].getKafedras()[3].getStudents()[2].setLastName("Petrenko");

Tester.getFaculty()[0].getKafedras()[3].getStudents()[2].setPoBatkovi("Artemovych");

Tester.getFaculty()[0].getKafedras()[3].getStudents()[2].setCourse(2);

Tester.getFaculty()[0].getKafedras()[3].getStudents()[2].setGroup(4);

Tester.getFaculty()[0].getKafedras()[3].getStudents()[3].setFirstName("Artem");

Tester.getFaculty()[0].getKafedras()[3].getStudents()[3].setLastName("Nikonov");

Tester.getFaculty()[0].getKafedras()[3].getStudents()[3].setPoBatkovi("Anatoliyovych");

Tester.getFaculty()[0].getKafedras()[3].getStudents()[3].setCourse(3);

Tester.getFaculty()[0].getKafedras()[3].getStudents()[3].setGroup(2);

Tester.getFaculty()[0].getKafedras()[3].getStudents()[4].setFirstName("Anton");

Tester.getFaculty()[0].getKafedras()[3].getStudents()[4].setLastName("Grytsenko");

Tester.getFaculty()[0].getKafedras()[3].getStudents()[4].setPoBatkovi("Andriovych");

Tester.getFaculty()[0].getKafedras()[3].getStudents()[4].setCourse(1);

Tester.getFaculty()[0].getKafedras()[3].getStudents()[4].setGroup(1);

}

}

import java.text.Collator;

import java.util.Comparator;

import java.util.Locale;

public class ComparatorTeachersAlphabet implements Comparator {

@Override

public int compare(Object o1, Object o2) {

Teacher tch1 = (Teacher) o1;

Teacher st2 = (Teacher) o2;

Locale locale = new Locale("uk","UA");

Collator collator = Collator.getInstance(locale);

int res = collator.compare(tch1.getLastName(),st2.getLastName());

if(res==0)

res = collator.compare(tch1.getFisrtName(),st2.getFisrtName());

if(res==0)

res = collator.compare(tch1.getPoBatkovi(),st2.getPoBatkovi());

return res;

}

}

import java.text.Collator;

import java.util.Comparator;

import java.util.Locale;

public class ComparatorStudentsAlphabet implements Comparator {

@Override

public int compare(Object o1, Object o2) {

Student st1 = (Student) o1;

Student st2 = (Student) o2;

Locale locale = new Locale("uk","UA");

Collator collator = Collator.getInstance(locale);

int res = collator.compare(st1.getLastName(),st2.getLastName());

if(res==0)

res = collator.compare(st1.getFirstName(),st2.getFirstName());

if(res==0)

res = collator.compare(st1.getPoBatkovi(),st2.getPoBatkovi());

return res;

}

}

import java.text.Collator;

import java.util.Comparator;

import java.util.Locale;

public class ComparatorStudentsCourse implements Comparator {

@Override

public int compare(Object o1, Object o2) {

Student st1 = (Student) o1;

Student st2 = (Student) o2;

int res = st1.getCourse() - st2.getCourse();

return res;

}

}