**Звіт**

**за результатами виконання**

**Лабораторної роботи №1**

**З предмету «Алгоритми і структури даних»**

**Роботу виконали:**

**Студенти факультету інформатики**

Спеціальність: Інженерія програмного забезпечення

Франків Ярослав

Гречка Артем

**Студент факультету правничих наук**

Спеціальність: Право

Тембіковський Олександр

**Викладач практики:**

Пєчкурова Олена Миколаївна

Зміст

[ПОСТАНОВКА ЗАДАЧІ 3](#_Toc130058183)

[РОЗПОДІЛ РОЛЕЙ 4](#_Toc130058184)

[ОПИС УСІХ РЕАЛІЗОВАНИХ МОЖЛИВОСТЕЙ 4](#_Toc130058185)

[СТРУКТУРА ПРОГРАМИ 5](#_Toc130058186)

[ОПИС МЕТОДІВ 6](#_Toc130058187)

[ПРОБЛЕМИ У РОБОТІ 7](#_Toc130058188)

# ПОСТАНОВКА ЗАДАЧІ

Завдання – написання програми, що буде формувати список студентів та викладачів університету НаУКМА.

Мають бути реалізовані наступні можливості:

1. Створення/видалення/редагування факультетів;
2. Створення/видалення/редагування кафедр факультетів;
3. Додавання/видалення/редагування студента / викладачів до кафедри;
4. Пошук студентів/викладачів за ПІБ, курсом або групою;
5. Виведення всіх студентів, впорядкованих за курсами;
6. Виведення всіх студентів/викладачів факультету, впорядкованих за алфавітом;
7. Виведення всіх студентів кафедри, впорядкованих за курсами;
8. Виведення всіх студентів/викладачів кафедри, впорядкованих за алфавітом;
9. Виведення всіх студентів вказаного користувачем курсу;
10. Виведення всіх студентів кафедри, впорядкованих за алфавітом;

При цьому програма має відповідати наступним вимогам:

1. Необхідним є реалізація захисту від невірного введення даних або ж заборона таких дій;
2. При написанні коду програми повинні бути використані об’єкти та обмін даних між ними;
3. Необхідним є передбачення рівнім ієрархії класів;
4. Метод внесення даних у програму – ввід з клавіатури;
5. Доцільним є пропонування меню з певним обсягом дій для кінцевого користувача;
6. За результатами виконаної роботи повинен бути створений звіт.

# РОЗПОДІЛ РОЛЕЙ

Розподіл ролей був проведений під час обговорення сильних та слабких сторін кожного з членів групи. За результатами обговорення група дійшла висновку, що:

1. Артем: створення каркасу проекту, класів та вагомої частини методів. Створення спільного простору для роботи та контроль якості коду.
2. Ярослав: розробка інтерфейсу, його оформлення та доопрацювання.
3. Олександр: проведення фінальних тестувань, розробка бази даних, підготовка звіту.

# ОПИС УСІХ РЕАЛІЗОВАНИХ МОЖЛИВОСТЕЙ

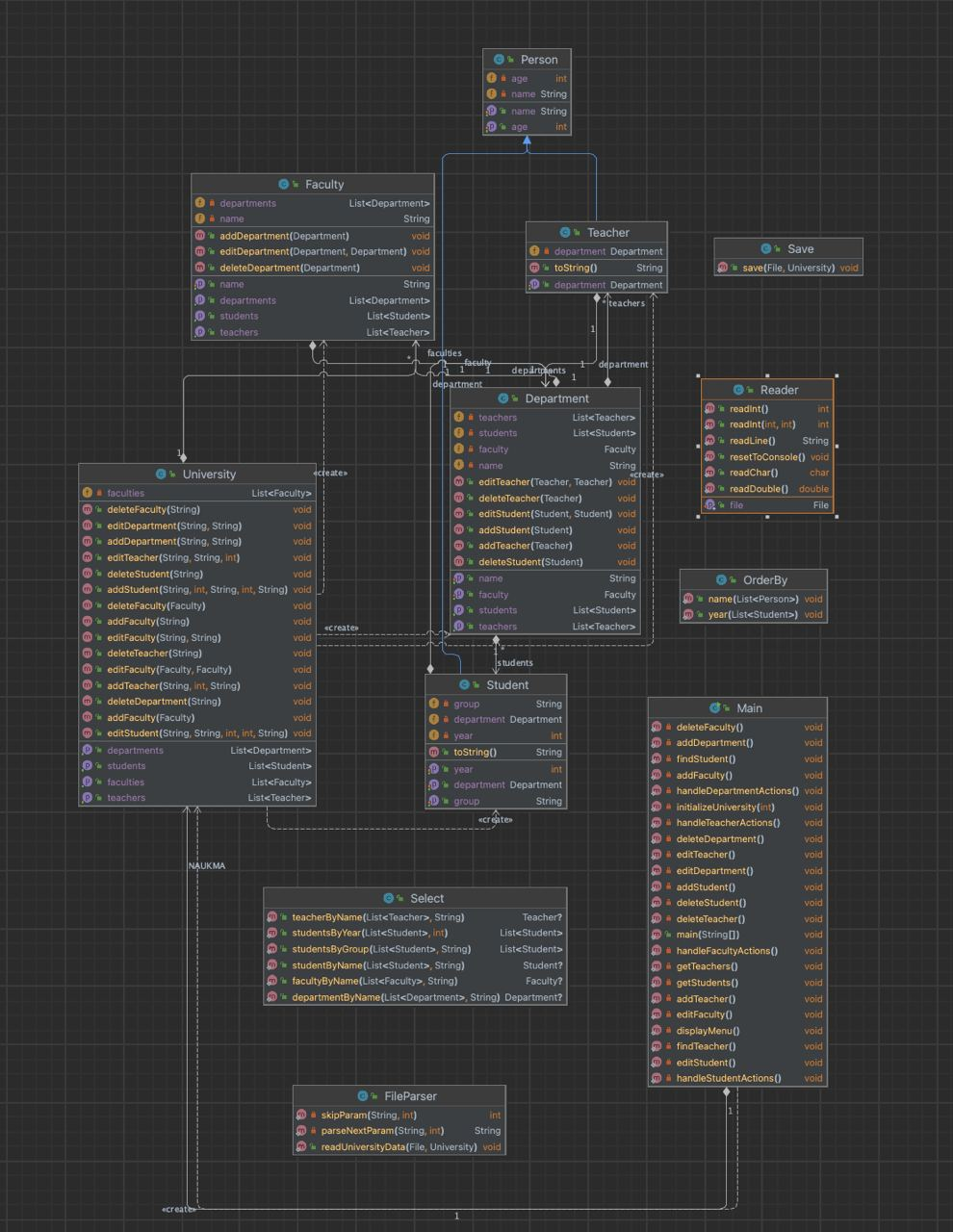
Написана нами програма має наступні реалізовані можливості:

1. Створення/видалення/редагування факультетів;
2. Створення/видалення/редагування кафедр факультетів;
3. Додавання/видалення/редагування студента / викладачів до кафедри;
4. Пошук студентів/викладачів за ПІБ, курсом або групою;
5. Виведення всіх студентів, впорядкованих за курсами;
6. Виведення всіх студентів/викладачів факультету, впорядкованих за алфавітом;
7. Виведення всіх студентів кафедри, впорядкованих за курсами;
8. Виведення всіх студентів/викладачів кафедри, впорядкованих за алфавітом;
9. Виведення всіх студентів вказаного користувачем курсу;
10. Виведення всіх студентів кафедри, впорядкованих за алфавітом;

В програмі також було окремо реалізовано можливість роботи з базами даних, котрими у даному випадку є файли. Таким чином робота кінцевого користувача з програмою є значною мірою полегшеною не тільки через можливість зберігати свою роботу для подальшої її продовження, але й почати свою роботу з цією програмою не з пустого місця.

# СТРУКТУРА ПРОГРАМИ

**Діаграма**



У ході роботи були сформовані наступні елементи програми:

1. Початкове меню інтерфейсу програми. На початку роботи з програмою кінцевому користувачу пропонується почати роботу з однією з базою даних: такими базами даних є база даних за замовчуванням, котра вже є встановленою; власна база даних користувача, шлях до котрої він повинен встановити самостійно; А також можливість не працювати з Приклад початку роботи програми
   базою даних та почати роботу з нуля.

Початок роботи програми

1. Після вибору однієї з пропозицій для кінцевого користувача стає доступним основне меню інтерфейсу, котре надає можливість йому почати роботу з:
   1. Факультетами
   2. Кафедрами
   3. Студентами
   4. Викладачами

Також користувачу пропонується припинити роботу з програмою, якщо у нього є на те бажання.

Зображення, що містить текст

Автоматично згенерований опис

Приклад продовження роботи з програмою

1. Використання будь-якої можливості – окрім виходу з програми - для початку роботи приведе користувача до одного з меню, у котрому користувачу залежного від його вибору надається можливість додавати, видаляти або ж редагувати вже існуючі факультети та кафедри; додавати або ж видаляти у списки студентів певних осіб, встановивши для кожного з них певні значення у полях інформації, таких як: ПІБ, вік, кафедру студента, групу студента та рік навчання студента. Такий самий процес є створеним й для роботи з викладачами, за виключенням того, що викладачам непотрібно призначати інформацію про номер групи та рік навчання.

Приклад додавання факультету до бази даних

Зображення, що містить текст

Автоматично згенерований опис

1. Після проведення роботи з додавання/видалення/редагування інформації, користувачу надається можливість вивести на екран інформацію про наявні факультети, кафедри, студентів та викладачів.

Зображення, що містить текст

Автоматично згенерований опис

Виведення інформації кінцевим користувачем інформації, що знаходиться у базі даних

# ОПИС МЕТОДІВ

1. **Main**Клас відповідає за виклик методів, що є задіяні для зчитування даних від кінцевого користувача, в ньому також знаходиться інтерфейс програми, з яким працює сам кінцевий користувач.
2. **Department**   
   Клас реалізує механіку кафедри. На кафедрі зберігаються студенти та викладачі, а також присутні методи для додавання/видалення/редагування студентів та викладачів.
3. **Faculty**Клас реалізує механіку факультета. На факультеті зберігаються кафедри, а також присутні методи для додавання/видалення/редагування кафедр.
4. **FileParser**Клас, як і стає відомим з його назви, проводить основну роботу з отримання, зчитування та систематизації інформації, що є заданою у базі даних, що є представленою окремим файлом.
5. **Order**

Клас є відповідальним за набір можливих налаштувань для сортування.

1. **OrderBy**  
   Клас проводить роботу з сортування наявної інформації, що отримується від колекцій.
2. **Person**  
   Клас є задіяний у встановленні/призначенні необхідної для роботи програми інформації для об’єктів, що є позначеними як персони. У даному випадку такими персонами є студенти так викладачі. Відтак, призначенням класу є встановлення ПІБ та віку тощо для таких об’єктів.
3. **Reader**

Клас є задіяним в проведенні роботи зі зчитування інформації, що надає кінцевий користувач, та перевірки такої інформації на коректність вводу.

1. **Save**

Клас є відповідальним за внесення отриманої та опрацьованої кінцевим користувачем інформації у іншу базу даних.

1. **Select**Клас є задіяним у пошуку інформації за певними завчасно встановленими критеріями: вік, назва, ім’я, група тощо.
2. **Student/Teacher**Ці класи реалізують студентів та викладачів.
3. **University**

Клас містить велику кількість методів, що є задіяними у інших класах для додавання, редагування та видалення факультетів, кафедр, студентів та викладачів, виводу отриманої інформації на екран. Фактично – це є місцезнаходження переважної кількості методів, що є задіяними для роботи з інформацією, що є зчитаною та перевіреною.

# ПРОБЛЕМИ У РОБОТІ

Однією з суттєвих проблем, що виникла під час роботи у групі, була правильна структуризація та організація коректної роботи програми з масивами даних та їх коректне відображення.   
Проблематичним також було створення та налагодження роботи інтерфейсу програми.   
Проблемою також була організація та структуризація коректного спільного використання створених методів в межах проекту.