МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені І.І.Мечникова

Факультет математики, фізики та інформаційних технологій

**ІНДИВІДУАЛЬНА РОБОТА**

з дисципліни «Програмування»

на тему: Вищий навчальний заклад

Виконала: Кір’як А. В.

студентка І курсу ФМФІТ,

123 Комп’ютерна інженерія

Керівник: Антоненко О.С.

Одеса – 2023

ЗМІСТ

[1. Постановка задачі 3](#_Toc138268015)

[2. Алгоритм роботи програми 4](#_Toc138268016)

[3. Програмна реалізація 5](#_Toc138268017)

[Висновок 15](#_Toc138268018)

[Список використаної літератури 16](#_Toc138268019)

# Постановка задачі

Створення ієрархії класів на тему вищого навчального закладу.

Створити класи: «Людина» (ПІБ, вік, зріст), спадкоємці «Студент» (група, факультет, список оцінок за поточний курс), «Викладач» (додаткові поля: ступінь, предмет, який викладає), «Оцінка» (предмет , залік, диф. залік чи іспит, число, методи: оцінка за шкалою ECTS, оцінка словами: «відмінно», «добре» тощо).

Визначити метод – визначити стипендію студента за семестр (повернути підвищену, якщо все добре, і звичайну, якщо немає незадовільно або незалік і середня оцінка не нижча за «добре»).

# Алгоритм роботи програми

Даний код представляє собою університетську систему, яка має меню для роботи з викладачами і студентами. Головне меню включає опції "Викладачі", "Студенти" і "Вихід". Користувач може обрати одну з цих опцій.

У меню "Викладачі" користувач може переглянути інформацію про всіх або одного викладачів, додати нового викладача або видалити існуючого викладача.

У меню "Студенти" користувач може переглянути інформацію про всіх чи одного студентів, додати нового студента або видалити існуючого студента.

Програма працює у циклі, який триває доти, поки користувач не обере опцію "Вихід". Підменю для викладачів і студентів також працює у циклі, де користувач може обирати дії до тих пір, поки не обере опцію повернутися до головного меню.

# Програмна реалізація

Проект налічує в собі 5 класів:

1). Human;

2). Teacher;

3). Student;

4). Grade;

5). University;

Клас **Human:**

У програмі реалізовано базовий абстрактний клас Human. Цей клас включає в себе наступні закриті (private) поля:

**string** full\_name – повне ім’я (ПІБ) людини;

**unsigned int** age – вік людини (тип unsigned int, тому що вік може бути тільки більше 0);

**unsigned int** height – зріст людини;

Конструктори класу **Human**:

Human(const **string**& full\_name\_, const **unsigned int**& age\_, const **unsigned int**& height\_) – конструктор з параметрами;

Human(**const string**& full\_name\_, **const unsigned int**& age\_) – конструктор з параметрами;

Human() - конструктор за замовчуванням;

~Human() – деструктор класу, який не виконує додаткових дій.

Основні методи класу Human

Геттери:

* **string** getFull\_name() – повертає значення змінної full\_name;
* **int** getAge() – повертає значення змінної age;
* **int** getHeight() – повертає значення змінної height;

Сеттери:

* **void** setFull\_name(const **string**& full\_name\_) – метод для зміни значення full\_name;
* **void** setAge(const **int**& age\_) – метод для зміни значення age;
* **void** setHeight(const **int**& height\_) – метод, що змінює значення height;

**void** Info() – абстрактний метод для виводу інформації про поля класів-спадкоємців класу Human;

**void** IsValid() – абстрактний метод для перевірки коректного присвоєння значення полю age. При введенні віку більше 100, генерується виключення та відбувається закінчення роботи програми;

Клас **Teacher:**

Клас, який є спадкоємцем класу Human. Призначений для збереження інформації про викладачів. Включає в себе наступні поля:

**string** academic\_degree – наукова ступінь викладача;

**string** subject – предмет, який викладає.

Конструктори класу **Teacher**:

Teacher(**const string**& full\_name\_, **const unsigned** **int**& age\_, **const string**& degree\_, **const string**& sub\_)- конструктор з параметрами;

~Teacher()- деструктор, який не виконує додаткових дій.

Основні методи класу **Teacher**:

Геттери:

* **string** getAcDegree()- повертає значення змінної academic\_degree;
* **string** getSubject()- повертає значення змінної subject;

Сеттери:

* **void** setAcDegree(**const string**& degree\_) – призначений для зміни значення змінної academic\_degree.
* **void** setSubject(**const string**& sub\_)- призначений для зміни значеня змінної subject.

**void** Info()**const override** – перевизначений метод, реалізований для виведення інформації про поля класу Teacher;

**void** IsValid() **const override** – перевизначений метод для перевірки коректності вводу змінної age. При введенні віку більше 100, генерується виключення та відбувається закінчення роботи програми;

Клас **Student**:

Клас, який є спадкоємцем класу Human. Призначений для збереження інформації про студентів. Включає в себе наступні поля:

**unsigned int** group – номер групи, до якої входить студент (може бути тільки додатнє);

**string** faculty – назва факультету, на якому навчаеться студент;

**unsigned int** course – номер курсу стедента (тільки додатнє значення);

**vector<Grade>** gradesVector – vector, призначений для зберігання елементів типу Grade, тобто об'єктів класу, який містить оцінки студента.

Конструктори класу **Student**:

Student(**const string**& full\_name\_, **const unsigned int**& age\_, **const string**& fac\_, **const unsigned int**& course\_ , **const unsigned int**& group\_, **const vector<Grade>**& gradesVector\_)- конструктор з параметрами;

Student() – конструктор без параметрів;

~Student() – деструктор класу, який не виконує додаткових дій;

Основні методи класу **Student**:

Геттери:

* **unsigned int** getGroup() – повертає значення змінної group;
* **string** getFaculty()- повертає значення змінної faculty;
* **unsigned int** getCourse()-повертає значення змінної course;

Сеттери:

* **void** setGroup(**const unsigned int**& group\_)- призначений для зміни значення змінної group;
* **void** setFaculty(**const string**& fac\_) - призначений для зміни значення змінної faculty;
* **void** setCourse(**const unsigned int**& course\_)- призначений для зміни значення змінної course;

**void** Info()**const override** – перевизначений метод, реалізований для виведення інформації про поля класу Students;

**void** IsValid() **const override** – перевизначений метод для перевірки коректності вводу змінної age. При введенні віку більше 100, генерується виключення та відбувається закінчення роботи програми;

**int** CalculateAverageGrade() **const** – в даному методі створюється змінна cout, якій присвоється значення кількості елементів у векторі **vector<Grade>** gradesVector. Опрацьовується цикл розрахунку суми всіх елементів вектора(тобто сума оцінок), а потім знаходиться їх середнє арифметичне: сума оцінок ділиться на змінну count;

**int** CalculateScholarship() **const** – у цьому методі розраховується стипендія студента. Якщо сєреднє арифметичне більше чи дорівнює 90 і метод hasFailingGrade() повертає false, то даний метод повертає значення «3000» (підвищена стипендія), а якщо середнє аріфметичне в діапазоні від 75 до 90 балів і немає незадовільних оцінок, то повертається значення «1900» (звичайна стипендія);

**bool** hasFailingGrade() **const** - перевіряє, чи має студент хоча б одну незадовільну оцінку (тобто меньше за 60 балів), перебираючи всі значення gradeVector, і повертає true, якщо така оцінка є, а false, якщо немає.

Клас **Grade**:

Клас, який призначений для збереження інформації про оцінки студента. Включає в себе наступні поля:

**string** subject – назва предмету, за який виставляється оцінка;

**bool** passed – зберігає true, якщо у студента залік;

**unsigned int** grade – збегірає оцінку;

**char** ects - ECTS формат оцінки;

**string** word\_grade – текстовий формат оцінки («відмінно», «добре» тощо);

Конструктори класу **Grade**:

Grade(**const string**& subject\_, **const int&** grade\_) - конструктор з параметрами;

Grade(**const int**& grade\_) - конструктор з параметрами;

~Grade() - деструктор класу, який не виконує додаткових дій;

Основні методи класу **Grade** :

Гетерри:

* **int** getGrade() - повертає отриману оцінку;
* **string** getSubject() - повертає предмет, до якого відноситься оцінка;

Сетери:

* **void** setGrade(const int& grade\_) - призначений для зміни значення змінної grade;
* **void** setSubject(const string& subject\_) - призначений для зміни значення змінної subject;

**char** FormECTS(int grade)- функція, що отримує оцінку (grade) як вхідний аргумент і повертає символ, що представляє відповідний рівень ECTS (European Credit Transfer and Accumulation System).

**string** FormWordGrade(int grade)– функція, що отримує оцінку (grade) як вхідний аргумент і повертає рядок, що представляє оцінку у словесному форматі.;

Головна функція **University:**

Функція main є точкою входу у програму. Вона керує потоком виконання та служить як головне меню для взаємодії з системою університету.

Класи:

**Teacher** - представляє вчителя і зберігає інформацію про його ім'я, вік, науковий ступінь та викладаний предмет;

**Student** - представляє студента і зберігає інформацію про його ім'я, вік, факультет, курс, групу та оцінки;

**Grade** - представляє оцінку і зберігає інформацію про її значення.

Функції:

**Main\_Menu()-** відображає головне меню програми;

**Menu\_Teachers() -** відображає меню для роботи з вчителями;

**Menu\_Students()** - відображає меню для роботи зі студентами;

**CreateTeachers()** - створення початкового списку викладачів університету. Вона ініціалізує деякі початкові дані викладачів, щоб забезпечити наявність даних для подальших дій у програмі ;

**CreateStudents() -**  створення початкового списку студентів університету. Вона ініціалізує деякі початкові дані студентів, щоб забезпечити наявність даних для подальших дій у програмі ;

**ShowOneStudent(int number)** – функція призначена для відображення інформації про конкретного студента. Вона перевіряє, чи індекс студента відповідає переданому номеру і виводить інформацію про цього студента за допомогою методу Info().

**ShowListOfStudents()** - призначена для відображення списку всіх студентів. Вона проходить по кожному студенту у векторі students і викликає метод Info(), щоб вивести інформацію про кожного студента.

**ShowListOfTeachers()-** призначена для відображення списку всіх викладачів. Вона проходить по кожному викладачу у векторі teachers і виводить на екран інформацію про них, включаючи повне ім'я, предмет та науковий ступінь. Кожен викладач представляється у вигляді порядкового номера, після якого слідує його повне ім'я, предмет та науковий ступінь. Відображаються всі викладачі у відповідному форматі, розділяючи їх порожнім рядком після кожного викладача.

**ShowOneTeacher(int number)** - призначена для відображення інформації про одного викладача за вказаним номером. Вона проходить по всіх викладачах у векторі teachers і перевіряє, чи співпадає номер викладача з вказаним номером мінус один (так як вектори індексуються з нуля). Якщо номери збігаються, викликається метод Info() для відображення інформації про викладача на екрані. Ця функція призначена для виведення інформації про одного викладача зі списку, наприклад, коли користувач обирає конкретного викладача для перегляду.

**AddTeacher()** - додає нового вчителя до списку викладачів. Ця функція запитує користувача про інформацію про нового викладача, як-от full\_name, age, degree, sub. Після отримання цих даних, функція створює об'єкт вчителя та додає його до списку teachers ;

**RemoveTeacher()**- видаляє вчителя зі списку teachers. Функція запитує користувача про інформацію про видаляємого викладача таку як повне ім'я або ідентифікатор. Потім вона шукає в списку в teachers і видаляє його зі списку, якщо такий вчитель знайдений.;

**AddStudent()**- додає нового студента до списку студенті. Функція запитує користувача про інформацію про нового студента, таку як full\_name, age, fac, cours , group, **vector<Grade>**. Після отримання цих даних, функція створює об'єкт студента та додає його до списку студентів ;

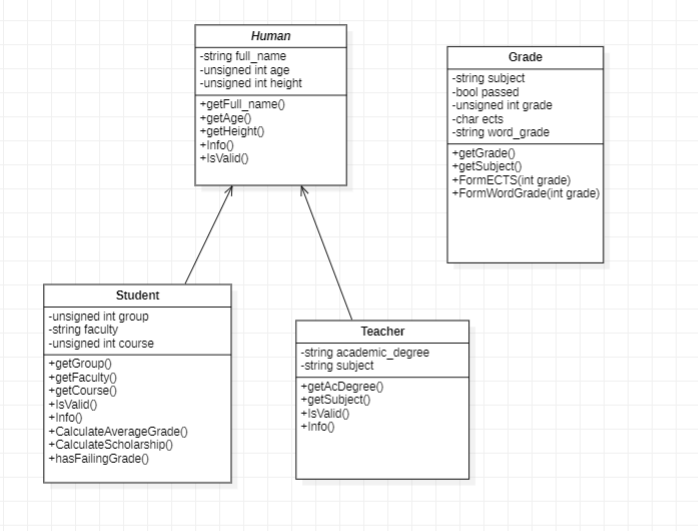
**RemoveStudent()-** видаляє студента зі списку студентів. Функція запитує користувача про інформацію про видаляємого студента, наприклад, його повне ім'я або ідентифікатор. Потім вона шукає студента в списку студентів і видаляє його зі списку, якщо такий студент знайдений ;

**main()-** головна функція програми , що виконує основну логіку програми. Вона координує виконання усіх інших функцій і дій програми. Основна робота функції main() включає наступні кроки, які виконуються з використанням циклів:

* Створення викладачів і студентів: За допомогою функції CreateTeachers() виконується створення викладачів. Цикл може використовуватися для повторення процесу створення викладачів досягнення певної кількості або задовольнення певної умови. Аналогічно, функція CreateStudents() використовує цикл для створення студентів.
* Виведення головного меню: Функція Main\_Menu() виводить головне меню програми, де користувач може вибрати опцію для перегляду інформації про викладачів, студентів або виходу з програми.
* Обробка вибору користувача: За допомогою циклу while або do-while оброблюється вибір користувача. Програма очікує введення номера опції від користувача
* Повторення циклу: Після обробки вибору користувача, програма переходить до початку циклу, виводить головне меню і очікує нового вибору користувача. Цей процес повторюється, поки користувач не обере опцію "Exit".

Таким чином, функція main() виконує циклічне взаємодію з користувачем, надає йому можливість керувати даними про викладачів та студентів, і повторює цей процес до завершення програми або вибору опції "Exit".

Розглянемо діаграму цих класів(побудована за допомоги інструменту StarUML):



# Висновок

Під час виконання цієї робити, ми познайомился з ООП на базі мови программування С++. А саме чітко ознайомилися з терміном ієрархія класів, та те як вони взаємодіють між собою.

Деякі ключові аспекти, які можна виділити:

* **Об'єктно-орієнтоване програмування (ООП):** Код використовує підхід ООП, де дані та функції пов'язані у класах. Класи Teacher, Student та Grade використовуються для представлення відповідних об'єктів з їх властивостями та методами.
* **Поліморфізм:** Код використовує поліморфізм шляхом використання віртуальних методів у класах Teacher та Student. Це дає змогу замінювати об'єкти підкласів (наприклад, викладачів та студентів) на об'єкти базового класу (наприклад, людей) для більш універсальної обробки.
* **Ієрархія класів:** У коді існує ієрархія класів, де клас Human є базовим класом для класів Teacher та Student. Це дозволяє спільно використовувати загальні властивості та методи, такі як повне ім'я та вік, між різними типами об'єктів.
* **Успадкування:** Класи Teacher та Student успадковують властивості та методи від базового класу Human. Наприклад, вони успадковують властивості full\_name та age. Це дозволяє зменшити дублювання коду та забезпечити більш структурований підхід до організації даних.
* **Поліморфізм на рівні списків**: Код використовує списки викладачів та студентів як колекції об'єктів, що дозволяє працювати з ними як зі списком загальних об'єктів типу Human. Це демонструє можливість застосування поліморфізму на рівні колекцій.

Загалом, цей проект відображає розуміння основ об'єктно-орієнтованого програмування, поліморфізму та ієрархії класів. Я використовувала класи, успадкування та поліморфізм для організації даних та забезпечення гнучкості в програмі.

# Список використаної літератури

* https://en.cppreference.com/w/cpp/language/classes
* https://uk.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%96%D0%BC%D0%BE%D1%80%D1%84%D1%96%D0%B7%D0%BC\_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)