КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра автоматизованих систем обробки інформації і управління

**КУРСОВА РОБОТА**

з дисципліни «Об’єктно-орієнтоване програмування»

на тему: «Система рейтингування студентів»

Студента 2 курсу групи ІП-з81

Спеціальності 121 «Інженерія програмного забезпечення»

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Кількість балів: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |
| --- |
| Національна оцінка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Члени комісії \_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (підпис) (посада, вчене звання, науковий ступінь, прізвище та ініціали)  \_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

(підпис) (посада, вчене звання, науковий ступінь, прізвище та ініціал

**ЗМІСТ**

[**ДОДАТОК А. ЛИСТ ЗАВДАННЯ**](#_heading=h.rwb7ukqfokld) **2**

[**ВСТУП**](#_heading=h.30j0zll) **5**

[**РОЗДІЛ 1. ОБ’ЄКТНО-ОРІЄНТОВАНИЙ АНАЛІЗ**](#_heading=h.1fob9te) **6**

[1.1 Постановка задачі](#_heading=h.3znysh7) 6

[1.2 Варіанти використання:](#_heading=h.2et92p0) 6

[1.4 Діаграма варіантів використання](#_heading=h.3dy6vkm) 6

[1.5 Опис варіантів використання](#_heading=h.1t3h5sf) 7

[1.6 Словник предметної області](#_heading=h.4d34og8) 8

[**РОЗДІЛ 2. ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОЕКТУВАННЯ**](#_heading=h.2s8eyo1) **10**

[2.1 Список класів](#_heading=h.17dp8vu) 10

[2.2 Клас Student](#_heading=h.3rdcrjn) 11

[2.3 Клас Group](#_heading=h.26in1rg) 12

[2.4 Проектування інтерфейсу](#_heading=h.lnxbz9) 12

[2.5 Інтерфейсні класи](#_heading=h.35nkun2) 14

[2.5 Діаграма класів](#_heading=h.1ksv4uv) 15

[**РОЗДІЛ 3. ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ**](#_heading=h.44sinio) **17**

[3.1 Алгоритмізація класу Student](#_heading=h.2jxsxqh) 17

[3.2 Алгоритмізація класу Group](#_heading=h.z337ya) 17

[3.3 Реалізація інтерфейсних класів](#_heading=h.3j2qqm3) 20

[**ВИСНОВКИ**](#_heading=h.qsh70q) **23**

[**СПИСОК ДЖЕРЕЛ**](#_heading=h.3as4poj) **24**

[**ДОДАТОК В. ЛІСТИНГИ ПРОГРАМИ**](#_heading=h.2p2csry) **27**

# **ДОДАТОК А. ЛИСТ ЗАВДАННЯ**

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра автоматизованих систем обробки інформації і управління

Дисципліна «Об’єктно-орієнтоване програмування»

Спеціальність "Інженерія програмного забезпечення"

Курс 2 Група ІП-з81 Семестр 2

**ЗАВДАННЯ**

**на курсову роботу студента**

Гордієнка Марка Парвез

1. Тема роботи: «Система рейтингування студентів»

2. Строк здачі студентом закінченої роботи: 01 червня 2019 року

3. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці): варіанти використання програми, класи, інтерфейсні класи, діаграма класів, гетери, сетери

5. Перелік графічного матеріалу ( з точним зазначенням обов’язкових креслень ): діаграма варіантів використання, прототип графічного інтерфейсу, діаграма класів, скріни роботи програми

6. Дата видачі завдання: 06 березня 2019 року

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Назва етапів курсової роботи | Термін виконання етапів роботи | Підписи керівника, студента |
| 1. | Отримання теми курсової роботи | 06.03.2019 |  |
| 2. | Підготовка ТЗ | 09.03.2019 |  |
| 3. | Аналіз предметної області | 19.03.2019 |  |
| 4. | Проектування архітектури програмної системи | 01.04.2019 |  |
| 5. | Розробка сценарію роботи програми | 05.04.2019 |  |
| 6. | Узгодження з керівником інтерфейсу користувача | 17.04.2019 |  |
| 7. | Розробка програмного забезпечення | 01.05.2019 |  |
| 8. | Узгодження з керівником плану тестування | 03.05.2019 |  |
| 9. | Тестування програми | 15.05.2019 |  |
| 10. | Підготовка пояснювальної записки | 17.05.2019 |  |
| 11. | Здача курсової роботи на перевірку | 01.06.2019 |  |
| 12. | Захист курсової роботи | 02.06.2019 |  |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Гордієнко Марк Парвез

(підпис)

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Носов Костянтин Сергійович

(підпис)

« 01 » травня 2019 р.

# **ВСТУП**

Метою курсової роботи є створення системи рейтингування студентів.

Для вирішення цього питання ми використаємо об’єктно-орієнтовану технологію програмування та проектування.

Об'є́ктно-орієнто́ване програмува́ння — одна з парадигм програмування, яка розглядає програму як множину «об'єктів», що взаємодіють між собою. Основу ООП складають чотири основні концепції: інкапсуляція, успадкування, поліморфізм та абстракція. Однією з переваг ООП є краща модульність програмного забезпечення (тисячу функцій процедурної мови, в ООП можна замінити кількома десятками класів із своїми методами)

На відміну від традиційних поглядів, коли програму розглядали як набір підпрограм, або як перелік інструкцій комп'ютеру, ООП програми можна вважати сукупністю об'єктів. Відповідно до парадигми об'єктно-орієнтованого програмування, кожен об'єкт здатний отримувати повідомлення, обробляти дані, та надсилати повідомлення іншим об'єктам. Кожен об'єкт — своєрідний незалежний автомат з окремим призначенням та відповідальністю

Ціль курсової роботи – оволодіти об’єктно-орієнтованою технологією проектування та програмування, дізнатися про етапи створення програм, аналіз предметної області, створення класів та архітектури майбутньої програми, підключення користувацького інтерфейсу та оволодіти об’єктно-орієнтованим програмуванням.

# **РОЗДІЛ 1. ОБ’ЄКТНО-ОРІЄНТОВАНИЙ АНАЛІЗ**

## 1.1 Постановка задачі

Завдання: Реалізувати складання списку студентів, що отримують стипендію. На вхід подається список студентів. Рядки мають такий формат: Прізвище, предмет №1,№2,№3,№4,№5, контрактник, наприклад:

Ivanov,78,61,95,87,90,+

Petrov,85,66,70,99,100,+

Необхідно розібрати список, підрахувати для кожного студента-бюджетника середній бал та вивести вищі 40% сумарного рейтингу студентів + мінімальний бал для отримання стипендії.

## 1.2 Варіанти використання:

* Запустити програму
* Ввести початкові дані: ім’я студента, його оцінки, бюджетник чи ні
* Перевірити отриманий список
* Вивести список студентів із середнім балом
* Вивести відсортований за середнім балом список студентів
* Вивести список студентів, які отримають стипендію
* Вивести мінімальний середній бал, який потрібен для отримання стипендії

## 1.4 Діаграма варіантів використання

Узагальнимо варіанти використання в діаграмі варіантів використання

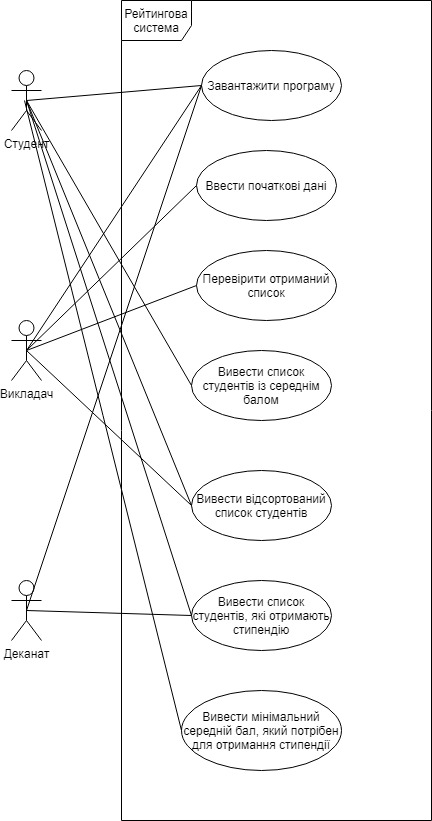


Рисунок 1.1 Діаграма варіантів використання

## 1.5 Опис варіантів використання

1.5.1 Розпочати програму

Коли запускається програма, на екран має виводитися меню, з якого користувач може вибрати потрібну дію.

1.5.2 Внести початкові дані

Програма передбачає користувацький інтерфейс, за допомогою якого можна додавати студентів до загального списку. Користувач повинен ввести дані студентів: їх ПІБ, оцінки та бюджетники вони чи ні

1.5.3. Перевірити список

Програма повинна надати можливість переглянути отриманий список, аби користувач пересвідчився у правильності введених даних

1.5.3 Вивести список студентів із середнім балом

Програма повинна порахувати середній бал для кожного студента окремо і потім вивести список студентів з їх балами

1.5.4 Вивести відсортований за середнім балом список студентів

Далі програма сортує масив студентів, створює рейтинговий список і виводить його.

1.5.5 Вивести список студентів, які отримають стипендію

Задається процент студентів, які можуть претендувати на стипендію. Після цього програма вираховує, скільки перших студентів отримують стипендію, знаходить бюджетників і виводить список.

1.5.6 Вивести мінімальний середній бал, який потрібен для отримання стипендії

Аналізуючи середній бал останнього студента, який потрапив у рейтинг, програма виводить мінімальний бал, який студент повинен отримати, аби потрапити у рейтинг.

## 1.6 Словник предметної області

Після того як ми описали можливі варіанти використання та сценарії роботи, потрібно зіставити словник предметної області. Для цього перечитаємо варіанти використання, знайдемо іменники, які потенційно можуть відповідати класам, та дієслова, які відповідатимуть методам.

Отриманий список іменників:

* Екран користувацького інтерфейсу
* Меню
* Користувач
* Кнопка
* Поле
* Функція
* Повідомлення про помилку
* Студент
* Список студентів
* Група
* Стипендія
* Бал
* Рейтинг

Отриманий список дієслів:

* Запустити програму
* Вивести список студенів із середніми балами
* Вивести відсортований список
* Вивести список студентів, які отримають стипендію
* Вивести мінімальний середній бал, який потрібен для отримання стипендії

# **РОЗДІЛ 2. ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОЕКТУВАННЯ**

## 2.1 Список класів

Тепер нам потрібно проаналізувати список іменників і виділити с них ті поняття, які стануть майбутніми класами програми.

Екран користувацького інтерфейсу може виступати як клас. Так як для програмування віконного інтерфейсу ми будемо використовувати програму Qt Creator, то будемо використовувати вбудовані класи програми. Серед них:

* QMainWindow, який виступає як екран або вікно, на якому будуть розміщуватися усі інші компоненти
* QPushButton, який відповідатиме за слово «кнопка» в нашому словнику
* QLineEdit, поле для вводу даних
* QLabel, який відповідає за текст
* QTabWidget, відповідає за меню з вкладеннями

Слово «функція» не можна віднести ні до класу, ні до атрибутів класу. Це слово виступає як загальне поняття.

Слово «Студент» візьмемо як клас. Його атрибутами будуть ім’я та прізвище студента, 5 оцінок, та булева змінна, яка відповідає за те бюджетник студент чи ні. Наступним класом візьмемо слово “Список”, а точніше ми назвемо цей клас “Група”. Оскільки Студент являється частиною групи, але може працювати окремо від Групи, то він вступає у відносини агрегації із класом Група.

Отже маємо остаточний список класів:

* QMainWindow
* QPushButton
* QLineEdit
* QLabel
* QTabWidget
* Student
* Group

## 2.2 Клас Student

В цьому підрозділі ми розробимо архітектуру класу Student. Аналізуючи список дієслів та попередній підрозділ, отримаємо такий список атрибутів:

* string name: ПІБ студента
* string mark1: 5 оцінок. Оцінки мають тип string, а не int, оскільки в реалізації методів класу буде використовуватися строкова функція getline(). Зі строкою працювати зручніше, ніж з числом
* string mark2
* string mark3
* string mark4
* string mark5
* string ifbudget: бюджетник чи ні
* double avgmark: середня оцінка

Дотримуючись принципу інкапсуляції, дані атрибути помістимо в секцію private, аби користувач не міг мати до них прямого доступу. Замість цього створимо гетери, за допомогою яких користувач та інші класи зможуть отримати потрібні дані, та сетери, які дозволяють ініціалізувати атрибути.

Гетери повинні повертати значення атрибутів, тому мають той самий тип даних, який має відповідний атрибут. Список гетерів:

* double GetAv();
* string GetN();
* string GetIfBud();
* string GetM1();
* string GetM2();
* string GetM3();
* string GetM4();
* string GetM5();

Сетери не повертають значення, тому мають тип void, але приймають введені дані користувача як параметри. Список сетерів:

* void SetAv(double a);
* void SetN(string a);
* void SetM1(string a);
* void SetM2(string a);
* void SetM3(string a);
* void SetM4(string a);
* void SetM5(string a);
* void SetIB(string a);

## 2.3 Клас Group

Для класу Group маємо такі атрибути:

* Student \*list: масив об’єктів класу Student
* int studnum: загальна кількість студентів
* double coef: рейтинговий коефіцієнт

Саме тут буде реалізовуватись основна функціональність програми. Методи класу:

* Group(int): конструктор, який ініціалізує масив студентів
* ~Group(): видаляє масив та звільняє пам’ять.
* void SetAtributes(int, string, string, string, string, string, string, string, double, int): заповнює пустий масив студентами і ініціалізує кожному доданому студенту його атрибути
* string calcavg(): рахує середній бал
* string sort(): сортує список
* string grants(): список студентів, які отримують стипендію. Також виводить мінімальний середній бал

## 2.4 Проектування інтерфейсу

При проектуванні інтерфейсу були виявлені два можливі його варіанти. Перший варіант інтерфейсу полягає в створенні декількох діалогових вікон, які послідовно будуть з’являтися після дій користувача. Наприклад, в першому вікні програма просить користувача написати ім’я студента. Після того як він це зробить, з’являється наступне вікно із можливістю записати оцінки студента.

Недоліком даного варіанту є те, що користувач не може вільно змінювати введені дані, а повинен пройти всі вікна, повернутися на початок програми і заново ввести всі дані. Крім того, така архітектура інтерфейсу ускладнює тестування програми, при якому потрібна більша свобода для введення даних.

Тому був розроблений другий варіант інтерфейсу, який полягає у створені єдиного екрану, який розділений на вкладення. В першому вкладенні розташовуються поля для ведення ім’я та оцінок студента. Наступні вкладення демонструються результати: середній бал, відсортований список, рейтинговий список, мінімальний середній бал

Прототип інтерфейсу наведений нижче:

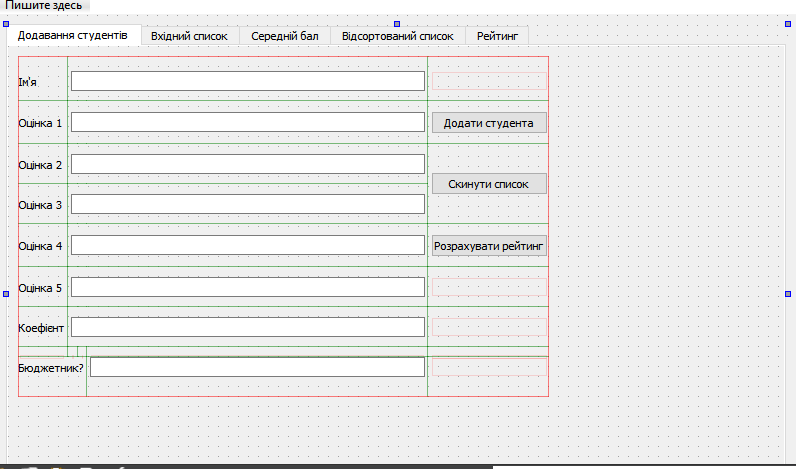


Рисунок 2.1 Прототип користувацького інтерфейсу

## 2.5 Інтерфейсні класи

Програма Qt Creator має власні вбудовані класи для проектування та програмування користувацького інтерфейсу. Ці класи мають особливості у відносинах між класам. Клас QMainWindow є головним класом програми, який агрегує в собі абстрактний клас QWidget. Від нього в свою чергу наслідуються клас конкретних віджетів: кнопок, полів и т.д. В нашому випадку ми маємо файл main.cpp, де створюється об’єкт класу QMainWindow, який виступатиме екраном, на якому будуть розміщуватися інші компоненти. Крім того, класи QPushButton, QLabel та QLineEdit перебувають у відносинах композиції зі створеним нами класом MainWindow і розміщуються в файлі mainwindow.cpp, оскільки вони є частиною форми і не можуть існувати окремо без неї.

Опишемо архітектуру Інтерфейсу. Його атрибутами є об’єкти класів QButton, QLabel та QLineEdit:

* TabsWidget: головний віджет вікна з вкладеннями
* IniDataTab: вкладення для вводу даних
* InputTab: вкладення із отриманим списком
* AvgTab: вкладення із середніми балами
* SortTab: вкладення для відсортованого списку
* RatingTab: вкладення для рейтингового списку
* AddButton: кнопка для додавання студента
* BudgetLabel: текст “Бюджетник?”
* CoefEdit: поле для вводу коефіцієнта
* CoefLabel: текст “Коефіцієнт”
* M1Edit: поле для вводу першої оцінки
* M1label: текст “Оцінка” 1
* M2Edit
* M2label
* M3Edit
* M3label
* M4Edit
* M4label
* M5Edit
* M5label
* NameEdit: поле для вводу ПІБ
* NameLabel: текст “Ім’я”
* RateButton: кнопка для розрахунку рейтингу
* ResetButton: кнопка для скидання списку студентів

До методів класу належить так званий обробник подій. Основними подіями є натискання на кнопки та зміна тексту в полях:

* void on\_AddButton\_clicked(): натискання на кнопку “Додати студента”
* void on\_ResetButton\_clicked(): натискання на кнопку “Скинути список”
* void on\_RateButton\_clicked(): натискання на кнопку “Розрахувати рейтинг”

## 2.5 Діаграма класів

Тепер узагальнимо розроблені класи та їх відносини в діаграмі класів:

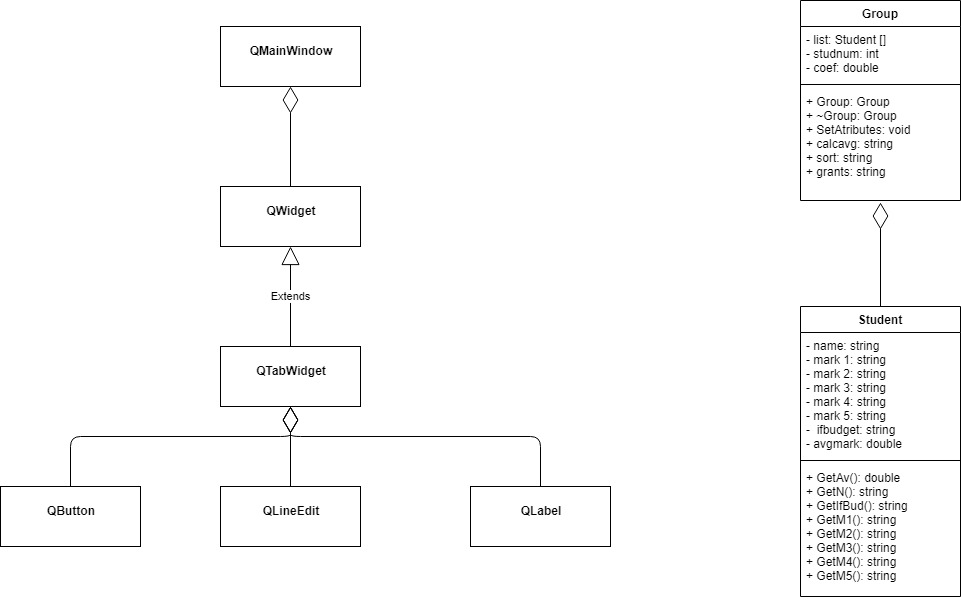


Рисунок 2.2 Діаграма класів

# **РОЗДІЛ 3. ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ**

## 3.1 Алгоритмізація класу Student

Реалізація цього класу є дуже простою, оскільки він має лише гетери та сетери, але деякі особливості. Гетери реалізуються через методи, які повертають значення атрибуту. Приклад гетеру:

double Student::GetAv()

{

return avgmark;

}

Методи-сетери приймають дані, які вводить користувач як параметри і через оператор this присвоює ці дані відповідному атрибуту. Реалізація сетеру

void Student::SetN(string name)

{

this->name = name;

}

Реалізація сетерів для атрибуту n (кількість відрізків розбиття), d (інтервал розбиття) та b(кінець відрізка) відрізняється тим, що мають додаткову умову, оскільки відрізків розбиття повинно бути більше або дорівнює трьом, інтервал розбиття повинен бути меншим за задану точність, а кінець відрізка – більшим початку:

## 3.2 Алгоритмізація класу Group

Сетери реалізуються так само як і в класі Student: користувач передає дані у вигляді параметрів, і атрибутам присвоюються ці дані. Клас Group має лише один сетер:

void Group::SetAtributes(int studindex,

string name,

string m1,

string m2,

string m3,

string m4,

string m5,

string ifbudget,

double coef,

int studnum)

{

list[studindex].SetN(name);

list[studindex].SetM1(m1);

list[studindex].SetM2(m2);

list[studindex].SetM3(m3);

list[studindex].SetM4(m4);

list[studindex].SetM5(m5);

list[studindex].SetIB(ifbudget);

this->coef = coef;

this->studnum = studnum;

}

Реалізація методу calcavg:

string Group::calcavg()

{

for (int i = 0; i < studnum; i++)

{

list[i].SetAv((stod(list[i].GetM1()) + stod(list[i].GetM2()) + stod(list[i].GetM3()) + stod(list[i].GetM4()) + stod(list[i].GetM5())) / 5);

}

string temp = "";

string s;

for (int i = 0; i < studnum; i++)

{

s = list[i].GetN() + "," + to\_string(list[i].GetAv()) + "\n" + temp;

temp = s;

}

return s;

}

Реалізація методу sort:

string Group::sort()

{

for (int i = 0; i < studnum - 1; i++)

{

int max = i;

for (int j = i+1; j < studnum; j++)

{

if (list[j].GetAv() > list[max].GetAv())

{

max = j; }

}

if (max != i)

{

double temp = list[i].GetAv();

list[i].SetAv(list[max].GetAv());

list[max].SetAv(temp);

}

}

string temp = "";

string s;

for (int i = studnum-1; i > 0; i--)

{

s = list[i].GetN() + "," + to\_string(list[i].GetAv()) + "," + list[i].GetIfBud() + "\n" + temp;

temp = s;

}

return s;

}

Реалізація методу grants:

string Group::grants()

{

int grantnum = studnum \* coef;

string temp = "";

string s;

for (int i = grantnum - 1; i > 0; i--)

{

if (list[i].GetIfBud() == "ні")

{

i--;

grantnum++;

}

else {

s = list[i].GetN() + "," + to\_string(list[i].GetAv()) + "\n" + temp;

temp = s;

}

}

return s;

}

## 3.3 Реалізація інтерфейсних класів

Перед основними подіями створюються глобальні допоміжні змінні та об’єкт класу Group:

int studnum = 0;

int studindex = 0;

QString temp = "";

Group g (30);

В програмі курсової роботи реалізовано три події:

* on\_AddButton\_clicked()
* on\_ResetButton\_clicked()
* on\_RateButton\_clicked()

void MainWindow::on\_AddButton\_clicked()

{

studnum++;

string name = (ui->NameEdit->text()).toStdString();

string m1 = (ui->M1Edit->text()).toStdString();

string m2 = (ui->M2Edit->text()).toStdString();

string m3 = (ui->M3Edit->text()).toStdString();

string m4 = (ui->M4Edit->text()).toStdString();

string m5 = (ui->M5Edit->text()).toStdString();

string budget = (ui->BudgetEdit->text()).toStdString();

double coef = (ui->CoefEdit->text()).toDouble();

if (budget == "так" || budget == "ні") {

g.SetAtributes(studindex, name, m1, m2, m3, m4, m5, budget, coef, studnum);

studindex++;

QString inputlist = ui->NameEdit->text() + "," +

ui->M1Edit->text() + "," +

ui->M2Edit->text() + "," +

ui->M3Edit->text() + "," +

ui->M4Edit->text() + "," +

ui->M5Edit->text() + "," +

ui->BudgetEdit->text();

ui->InputList->setText(inputlist + "\n" + temp);

temp = inputlist + "\n" + temp;

}

else {

QMessageBox::critical(this, "Помилка", "Введіть так або ні");

}

}

void MainWindow::on\_ResetButton\_clicked()

{

ui->InputList->setText("");

temp = "";

studnum = 0;

studindex = 0;

}

void MainWindow::on\_RateButton\_clicked()

{

string s1 = g.calcavg();

QString qs1 = QString::fromStdString(s1);

ui->AvgList->setText(qs1);

string s2 = g.sort();

QString qs2 = QString::fromStdString(s2);

ui->SortList->setText(qs2);

string s3 = g.grants();

QString qs3 = QString::fromStdString(s3);

ui->RateList->setText(qs3);

}

# **ВИСНОВКИ**

Виконуючи цю лабораторну роботу ми провели бізнес-аналіз предметної області задачі. В процесі його проведення виявили:

* чому присвячена предметна область, які в ній є терміни і поняття, суб'єкти та об'єкти, способи взаємодії суб'єктів, способи використання об'єктів, закономірності.
* що входить в словник предметної області, окремо виділивши список іменників і список дієслів, які можуть бути пов'язані з іменниками.
* які функціональні вимоги до розроблюваної системи.

Вся інформація узагальнена у вигляді діаграми прецедентів.

На етапі проектування враховувались не тільки особливості предметної області, а й такі принципи об'єктно-орієнтованого підходу як абстракція, інкапсуляція, поліморфізм. Результати проектування були оформлені у вигляді діаграми класів.

На етапі програмування відбувається програмна реалізація класів. Також розробляється тестовий додаток, що запускає процес моделювання та перевіряє правильність створених інтерфейсів класів. Даний додаток повинен має віконний інтерфейс, що включає меню і інтерактивні інтерфейсні засоби, за допомогою яких можна реалізувати визначену функціональність додатку.

На етапі тестування ми провели тестування розробленого програмного забезпечення. Методика тестування розробленого програмного забезпечення передбачає виконання спрощеного варіанта модульного тестування, що зводиться до тестування всіх методів розроблених класів

Тестуючи програму, ми виявили, що вона правильно обчислює екстремуми, чи досягли мети курсової роботи.

# **СПИСОК ДЖЕРЕЛ**

1. Лафоре Р. Объектно-ориентированное программирование в C++. С.-Петербург: Питер, 2006.

2. Мацяшек Лешек А. Анализ и проектирование информационных систем с помощью UML 2.0

3. Екстремум функції. Wikipedia

4. Об'єктно-орієнтоване програмування. Wikipedia

5. Qt Creator, документація. Веб-ресурс: <http://doc.crossplatform.ru/qt/4.8.x/html-qt/>

6. Курс “Об’єктно-орієнтоване програмування”. Веб-ресурс: https://www.intuit.ru/studies/courses/71/71/info

7. Курс “Об’єктно-орієнтоване проектування”. Веб-ресурс: https://www.intuit.ru/studies/courses/72/72/lecture/2150

**ДОДАТОК Б. ТЕХНІЧНЕ ЗАВДАННЯ**

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра

автоматизованих систем обробки інформації та управління

Затвердив

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_201\_ р.

Виконавець:

Студент \_\_\_\_<*ПІБ>\_\_\_\_\_\_\_\_\_\_*

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_201\_ р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання курсової роботи

на тему: «Система рейтингування студентів»

з дисципліни:

«Об’єктно-орієнтоване програмування»

Київ 2020

1. *Мета*: Метою курсової роботи є розробка програми для рейтингування студентів

**2.** *Дата початку роботи*: « 06 » березня 2019 р.

**3.** *Дата закінчення роботи*: « 01 » червня 2019 р.

4. *Вимоги до програмного забезпечення*.

1) Функціональні вимоги:

* введення вихідних даних за допомогою розробленої самостійно екранної форми (введення частини даних організувати засобами меню);
* введення даних студентів
* збереження доданих студентів у вигляді списку
* одержання середніх оцінок
* одержання відсортованого списку
* одержання рейтингового списку

2) Нефункціональні вимоги:

* Все програмне забезпечення та супроводжуюча технічна документація повинні задовольняти наступним ДЕСТам:

ГОСТ 29.401 - 78 - Текст програми. Вимоги до змісту та оформлення.

ГОСТ 19.106 - 78 - Вимоги до програмної документації.

ГОСТ 7.1 - 84 та ДСТУ 3008 - 95 - Розробка технічної документації.

5. *Стадії та етапи розробки*:

1) Об'єктно-орієнтований аналіз предметної області задачі (до 19.03.2019 р.)

2) Об'єктно-орієнтоване проектування архітектури програмної системи (до 01.04.2019 р.)

3) Розробка програмного забезпечення (до 01.05.2019 р.)

4) Тестування розробленої програми (до 15.05.2019 р.)

5) Розробка пояснювальної записки (до 17.05.2019 р.).

6) Захист курсової роботи (до 02.05.2019 р.).

6. *Порядок контролю та приймання*. Поточні результати роботи над КР регулярно демонструються викладачу. Своєчасність виконання основних етапів графіку підготовки роботи впливає на оцінку за КР відповідно до критеріїв оцінювання.

# **ДОДАТОК В. ЛІСТИНГИ ПРОГРАМИ**

**В.1 Файл main.cpp**

#include "mainwindow.h"

#include <QApplication>

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

MainWindow w;

w.show();

return a.exec();

}

**В.2 Файл mainwindow.h**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

QT\_BEGIN\_NAMESPACE

namespace Ui { class MainWindow; }

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

private slots:

void on\_AddButton\_clicked();

void on\_ResetButton\_clicked();

void on\_RateButton\_clicked();

private:

Ui::MainWindow \*ui;

};

#endif // MAINWINDOW\_H

**В.3 Файл mainwindow.cpp**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "Group.h"

#include "Student.h"

#include <istream>

#include <sstream>

#include <QMessageBox>

using namespace std;

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

}

MainWindow::~MainWindow()

{

delete ui;

}

int studnum = 0;

int studindex = 0;

QString temp = "";

Group g (30);

void MainWindow::on\_AddButton\_clicked()

{

studnum++;

string name = (ui->NameEdit->text()).toStdString();

string m1 = (ui->M1Edit->text()).toStdString();

string m2 = (ui->M2Edit->text()).toStdString();

string m3 = (ui->M3Edit->text()).toStdString();

string m4 = (ui->M4Edit->text()).toStdString();

string m5 = (ui->M5Edit->text()).toStdString();

string budget = (ui->BudgetEdit->text()).toStdString();

double coef = (ui->CoefEdit->text()).toDouble();

if (budget == "так" || budget == "ні") {

g.SetAtributes(studindex, name, m1, m2, m3, m4, m5, budget, coef, studnum);

studindex++;

QString inputlist = ui->NameEdit->text() + "," +

ui->M1Edit->text() + "," +

ui->M2Edit->text() + "," +

ui->M3Edit->text() + "," +

ui->M4Edit->text() + "," +

ui->M5Edit->text() + "," +

ui->BudgetEdit->text();

ui->InputList->setText(inputlist + "\n" + temp);

temp = inputlist + "\n" + temp;

}

else {

QMessageBox::critical(this, "Помилка", "Введіть так або ні");

}

}

void MainWindow::on\_ResetButton\_clicked()

{

ui->InputList->setText("");

temp = "";

studnum = 0;

studindex = 0;

}

void MainWindow::on\_RateButton\_clicked()

{

string s1 = g.calcavg();

QString qs1 = QString::fromStdString(s1);

ui->AvgList->setText(qs1);

string s2 = g.sort();

QString qs2 = QString::fromStdString(s2);

ui->SortList->setText(qs2);

string s3 = g.grants();

QString qs3 = QString::fromStdString(s3);

ui->RateList->setText(qs3);

}

**В.4 Файл Student.h**

#pragma once

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

class Student

{

string name; //ФИО

string mark1; //переменные, которые отвечают

string mark2; //за оценки

string mark3; //специально обьявлены в типе string

string mark4; //чтобы можно было использовать функцию

string mark5; //getline()

string ifbudget; //бюджетники или нет

double avgmark; //средняя оценка

public:

double GetAv();

string GetN();

string GetIfBud();

string GetM1();

string GetM2();

string GetM3();

string GetM4();

string GetM5();

void SetAv(double a);

void SetN(string a);

void SetM1(string a);

void SetM2(string a);

void SetM3(string a);

void SetM4(string a);

void SetM5(string a);

void SetIB(string a);

};

**В.5 Файл Student.cpp**

#include "Student.h"

double Student::GetAv()

{

return avgmark;

}

string Student::GetN()

{

return name;

}

string Student::GetIfBud()

{

return ifbudget;

}

string Student::GetM1()

{

return mark1;

}

string Student::GetM2()

{

return mark2;

}

string Student::GetM3()

{

return mark3;

}

string Student::GetM4()

{

return mark4;

}

string Student::GetM5()

{

return mark5;

}

void Student::SetN(string name)

{

this->name = name;

}

void Student::SetM1(string mark1)

{

this->mark1 = mark1;

}

void Student::SetM2(string mark2)

{

this->mark2 = mark2;

}

void Student::SetM3(string mark3)

{

this->mark3 = mark3;

}

void Student::SetM4(string mark4)

{

this->mark4 = mark4;

}

void Student::SetM5(string mark5)

{

this->mark5 = mark5;

}

void Student::SetIB(string ifbudget)

{

this->ifbudget = ifbudget;

}

void Student::SetAv(double avg)

{

this->avgmark = avg;

}

**В.6 Файл Group.h**

#pragma once

#include "Student.h"

#include <iomanip>

class Group

{

Student \*list;

int studnum;

double coef;

public:

Group(int);

~Group();

void SetAtributes(int, string, string, string, string, string, string, string, double, int);

string calcavg();

string sort(); //сортируем список по убыванию средних оценок

string grants(); //расчитываем списко студентов, которые получают стипендиб и сохраняем в файл

};

**В.7 Файл Group.cpp**

#include "Group.h"

#include <istream>

#include <sstream>

using namespace std;

Group::Group(int studnum)

{

this->studnum = studnum;

list = new Student[studnum];

}

Group::~Group()

{

delete[] list;

}

void Group::SetAtributes(int studindex,

string name,

string m1,

string m2,

string m3,

string m4,

string m5,

string ifbudget,

double coef,

int studnum)

{

list[studindex].SetN(name);

list[studindex].SetM1(m1);

list[studindex].SetM2(m2);

list[studindex].SetM3(m3);

list[studindex].SetM4(m4);

list[studindex].SetM5(m5);

list[studindex].SetIB(ifbudget);

this->coef = coef;

this->studnum = studnum;

}

string Group::calcavg()

{

for (int i = 0; i < studnum; i++)

{

list[i].SetAv((stod(list[i].GetM1()) + stod(list[i].GetM2()) + stod(list[i].GetM3()) + stod(list[i].GetM4()) + stod(list[i].GetM5())) / 5); //преобразовываем string в int и расчитываем среднее арифметическое

}

string temp = "";

string s;

for (int i = 0; i < studnum; i++)

{

s = list[i].GetN() + "," + to\_string(list[i].GetAv()) + "\n" + temp;

temp = s;

}

return s;

}

string Group::sort()

{

for (int i = 0; i < studnum - 1; i++) //отвечат за отсортированную часть

{

int max = i; //переменная сохраняет индекс максимального элемента

for (int j = i+1; j < studnum; j++) //внутренний цикл отвечает за неотсортированную часть

{

if (list[j].GetAv() > list[max].GetAv()) //если находится элемент больше текущего

{

max = j; //то меняем индекс

}

}

if (max != i) //если последний индекс отсортированной части не совпадает с индексом максимального элемента

{

double temp = list[i].GetAv(); //меняем их местами

list[i].SetAv(list[max].GetAv());

list[max].SetAv(temp);

}

}

string temp = "";

string s;

for (int i = studnum-1; i > 0; i--)

{

s = list[i].GetN() + "," + to\_string(list[i].GetAv()) + "," + list[i].GetIfBud() + "\n" + temp;

temp = s;

}

return s;

}

string Group::grants()

{

int grantnum = studnum \* coef; //кол-во студентов, которые получают стипендию

string temp = "";

string s;

for (int i = grantnum - 1; i > 0; i--)

{

if (list[i].GetIfBud() == "ні") //если студент не бюджетник

{

i--; //то пропускаем его и переходим к следующему

grantnum++; //и увеличиваем переменную, чтобы набрать нужно кол-во студентов

}

else {

s = list[i].GetN() + "," + to\_string(list[i].GetAv()) + "\n" + temp;

temp = s;

}

}

return s;

}