Метою цієї роботи було удосконалення ієрархії класів в програмі, що реалізує просту базу даних Аптеки на мові програмування С++. До рефакторінгу коду, ієрархія класів була у вигляді звийної агрегації.

Взагалі кажучі, на етапі написання цієї програми, ООП властивостям багато уваги не приділялось, тому ієрархія класів булла складена необдумано (у вигляді агрегації).

class Medication {

public:

int medication\_id;

string name;

float price;

int quantity;

string dose;

string form;

string manufacturer;

string group;

};

class Customer {

private:

int customer\_id;

public:

string customer\_name;

string address;

string telephone\_number;

float money\_check = 0;

};

class Category {

public:

int category\_code;

string group;

vector<Medication> medication;

};

class Delivery {

public:

string address; // not name

int delivery\_id;

Customer customer;

string delivery\_time;

};

class Order {

private:

int order\_Id; //

public:

Customer customer;

vector<Medication> medication;

Delivery delivery;

string delivery\_address;

bool is\_active; //

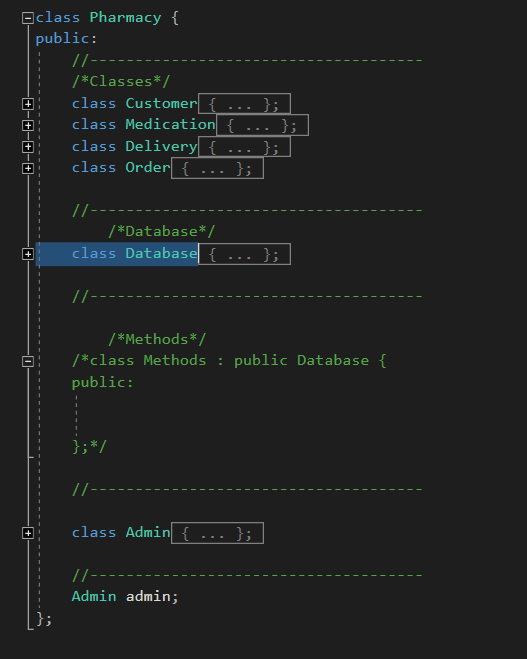
string order\_time;

};

Через те, що ООП реалізація не була чітко продумана, написання цієї програми видалось складним та незручним, тому в цьому семестрі в мене з’явилась можливість реконструювати цей код.

Перше, що мені довелось зробити, це, звичайно, продумати основну реалізацію нової ієрархії класів для того щоб покращити читабельність коду та полегшити його реалізацію.

Для цього, використовуючи знання основних концепцій ООП, я вирішила, що в цій програми дуже доречним буде використання композиції для основного класу Pharmacy



В класі Pharmacy містяться всі основні класи для роботи програми, а саме:

1. Customer, Medication, Delivery, Order – допоміжні класи
2. Database – класс із контейнерами
3. Admin – клас-розділ для реалізації основних методів

Важливо зазначити, що використання такої структури не привело до великих змін у основній логіці програми, та значно підвищило читабельність коду. Також, додавання будь яких нових методів, розділів, баз та іншого стає дуже простим завдяки чіткої ієрархії класів. Наприклад, якщо створити внутрішній класс New\_Rozdil{};, та унаслідувати його від DataBase, то він буде мати доступ до всіх йому потрібних інструментів, та звертання до нього також буде дуже зручним.

