

Технологии конструирования программного обеспечения

Отчет по заданию №1

Группа: 221-329

Студент: Едисеев Олег Владимирович

Задание на лабораторную работу

Необходимо написать программы в соответствии с полученными заданиями. При написании программ допускается использовать языки программирования C++, C#, Java. Программы должны быть снабжены комментариями, характеризующими: автора, назначение классов, методов и ключевых объектов. Рекомендуется включить комментарий, описывающий идею представленной реализации задания.

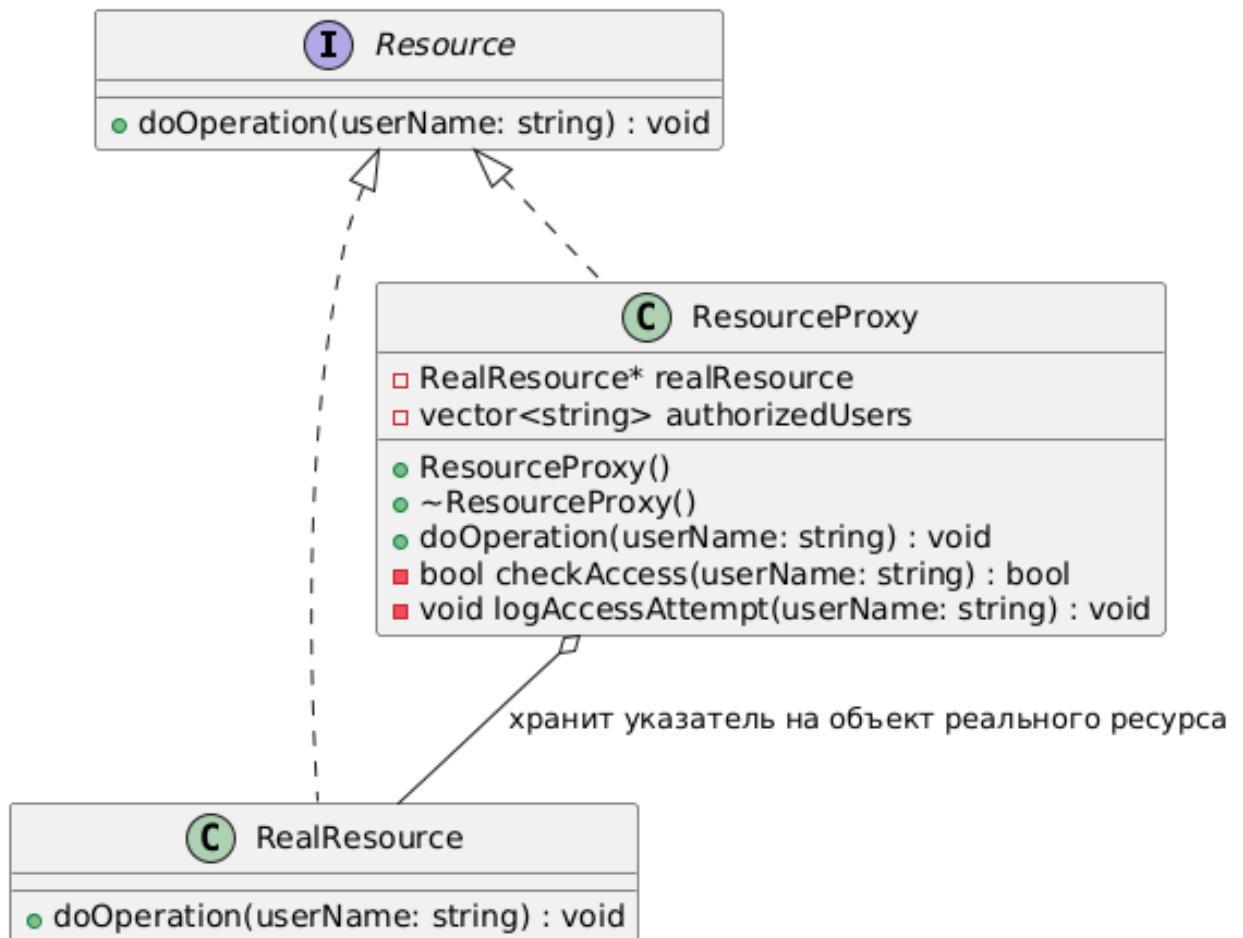
Для каждой программы построить диаграмму классов, описать назначение компонентов диаграммы, охарактеризовать связи между ними.

Ответ на каждое задание оформить в виде отдельного документа pdf, в котором указать ФИО и группу автора, привести полную формулировку задания, диаграмму классов с описанием, исходный код модулей программы. Ответ загрузить в СДО.

Задание 1.

Паттерн «Заместитель». Приложение «Журнал». Реализовать процедуры контроля доступа и логирования (запись времени обращения) обращений клиентов к разделяемому ресурсу. Ресурс любой. Количество клиентов не менее 3. Результаты работы Заместителя выводить в консоль или записывать в файл.

Диаграмма классов



- Resource – интерфейс.
- RealResource – конкретная реализация интерфейса Resource.
- ResourceProxy – класс-заместитель, также реализует интерфейс Resource, но внутри содержит указатель на RealResource и добавляет логику контроля доступа и логирования.
- На диаграмме показаны:
 - Наследование (реализация интерфейса) от RealResource к Resource и от ResourceProxy к Resource.
 - Отношение агрегации, отражающее, что ResourceProxy хранит экземпляр RealResource и управляет его временем жизни.

Исходный код программы

```
// Едисеев Олег Владимирович 221-329

// Интерфейс Resource определяет метод doOperation, который будет вызываться
// клиентами.
// Класс RealResource содержит логику доступа к ресурсу (вывод сообщения).
// Класс ResourceProxy:
//     Содержит список авторизованных пользователей (для контроля доступа)
//     Логирует каждую попытку доступа (выводит текущее время и имя
//     пользователя).
//     Если пользователь входит в список разрешённых, перенаправляет вызов к
//     реальному ресурсу, иначе сообщает об отказе в доступе.

#include <iostream>
#include <string>
#include <vector>
#include <ctime>
#include <chrono>
#include <iomanip>
#include <sstream>

using namespace std;

// Интерфейс ресурса
class Resource {
public:
    virtual ~Resource() = default;
    // Какая-либо операция
    virtual void doOperation(const string& userName) = 0;
};

// Класс, представляющий реальный ресурс
class RealResource : Resource {
public:
    void doOperation(const string& userName) override {
        // Выполнение какой-либо операции
        cout << "[Resource] Выполнена операция от имени пользователя: "
        << userName << endl;
    }
};

// Класс-заместитель (Proxy)
class ResourceProxy : Resource {
public:
    ResourceProxy() : realResource(new RealResource()) {
        // Список авторизованных пользователей
        authorizedUsers = { "Олег", "Алексей" };
    }
    ~ResourceProxy() {
        delete realResource;
    }
    void doOperation(const string& userName) override {
        // Логируем попытку доступа с указанием времени
        logAccessAttempt(userName);

        // Проверяем, есть ли пользователь в списке разрешённых
        if (checkAccess(userName)) {
```

```

        // Если доступ разрешён, перенаправляем вызов к реальному
ресурсу
        cout << "[Proxy] Доступ разрешён пользователю: " << userName
<< endl;
        realResource->doOperation(userName);
    } else {
        // Если доступ запрещён
        cout << "[Proxy] Доступ отказан пользователю: " << userName
<< endl;
    }
    cout << endl;
}

private:
    RealResource* realResource;
    vector<string> authorizedUsers; // Список авторизованных
пользователей

    // Метод для проверки доступа
    bool checkAccess(const string& userName) {
        for (const auto& user : authorizedUsers) {
            if (user == userName) {
                return true;
            }
        }
        return false;
    }

    // Метод для логирования попыток доступа с указанием времени
    void logAccessAttempt(const string& userName) {
        auto now = chrono::system_clock::now();
        time_t timeNow = chrono::system_clock::to_time_t(now);
        tm* localTime = localtime(&timeNow);

        ostringstream oss;
        oss << put_time(localTime, "%Y-%m-%d %H:%M:%S");

        cout << "[Proxy] " << oss.str() << " | Попытка доступа.
Пользователь: " << userName << endl;
    }
};

int main() {
    ResourceProxy proxy;

    // Имена пользователей (3 клиента - один из них не имеет доступа)
    vector<string> users = { "Олег", "Женя", "Алексей" };

    // Все пытаются выполнить одну и ту же операцию над ресурсом
    for (const auto& userName : users) {
        proxy.doOperation(userName);
    }
    return 0;
}

```