МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

**«ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**Лабораторная работа № 8:**

"Программа управляемая событиями."

Выполнила: студентка гр.РИС-23-3Б

Мокроусова Ангелина Андреевна

Проверила: доцент кафедры ИТАС

Полякова Ольга Андреевна

Пермь 2024

**Постановка задачи:**

Общая часть:

1. Определить иерархию пользовательских классов (см. лабораторную работу №5). Во главе иерархии должен стоять абстрактный класс с чисто виртуальными методами для ввода и вывода информации об атрибутах объектов.
2. Реализовать конструкторы, деструктор, операцию присваивания, селекторы и модификаторы.
3. Определить класс-группу на основе структуры, указанной в варианте.
4. Для группы реализовать конструкторы, деструктор, методы для добавления и удаления элементов в группу, метод для просмотра группы, перегрузить операцию для получения информации о размере группы.
5. Определить класс Диалог — наследника группы, в котором реализовать методы для обработки событий.
6. Добавить методы для обработки событий группой и объектами пользовательских классов.
7. Написать тестирующую программу.
8. Нарисовать диаграмму классов и диаграмму объектов.

Задание варианта:

Базовый класс ЧЕЛОВЕК (Регзоп):

* Имя – string
* Возраст – int

Производный класс АБИТУРИЕНТ(АВТОВТЕМТ):

* Количество баллов – int
* Специальность – string
* Группа – Дерево (Tree)

Команды:

* Создать группу (формат команды: m количество элементов группы).
* Добавить элемент в группу (формат команды: +)
* Удалить элемент из группы (формат команды -)
* Вывести информацию об элементах группы (формат команды: s)
* Вывести информацию о среднем возрасте (формат команды: z).
* Конец работы (формат команды: q)

**Анализ задачи:**

Классы:

* Person: Абстрактный базовый класс, представляющий общие свойства для всех лиц (имя и возраст).
* Applicant: Производный класс от Person, представляющий абитуриентов с дополнительными свойствами (балл и специальность).
* Group: Класс, представляющий группу абитуриентов. Хранит коллекцию объектов Applicant.
* Dialog: Класс, обеспечивающий интерактивный диалог с пользователем. Позволяет выполнять операции над группой (создание, добавление, удаление, вывод и т. д.).

Функциональность:

* Ввод и вывод данных абитуриентов: Методы Input() и Output() в классе Applicant позволяют вводить и выводить данные абитуриентов (имя, возраст, балл и специальность).
* Управление группой: Класс Group предоставляет методы для добавления, удаления и показа абитуриентов в группе.
* Средний возраст: Метод CalculateAvgAge() в классе Group вычисляет и возвращает средний возраст абитуриентов в группе.
* Интерактивный диалог: Класс Dialog предоставляет интерфейс командной строки, позволяющий пользователям создавать, управлять и получать информацию о группе абитуриентов.

**Код на С++ с комментариями:**

#include <iostream>

// Класс vector для хранения элементов в группе

#include <vector>

// Класс string для работы со строками

#include <string>

// Алгоритмы для работы с сортировкой

#include <algorithm>

using namespace std;

// Абстрактный базовый класс для работы с данными о персонах

class Person {

protected:

// Защищенные переменные для имени и возраста

string name;

int age;

public:

// Виртуальные методы для ввода и вывода данных

virtual void Input() = 0;

virtual void Output() = 0;

};

// Класс для работы с данными об абитуриентах (наследование от Person)

class Applicant : public Person {

private:

// Дополнительно храним балл и специальность

int score;

string specialty;

public:

void Input() override {

cout << "Введите имя: ";

cin >> name;

cout << "Введите возраст: ";

cin >> age;

cout << "Введите балл: ";

cin >> score;

cout << "Введите специальность: ";

cin >> specialty;

}

void Output() override {

cout << "Имя: " << name << endl;

cout << "Возраст: " << age << endl;

cout << "Балл: " << score << endl;

cout << "Специальность: " << specialty << endl;

}

int GetAge() {

return age;

}

};

// Класс для работы с группой абитуриентов

class Group {

private:

// Вектор для хранения абитуриентов

vector<Applicant> applicants;

public:

// Конструктор по умолчанию

Group() = default;

void AddApplicant(Applicant& applicant) {

// Добавляем абитуриента в группу

applicants.push\_back(applicant);

}

void RemoveApplicant() {

// Если группа не пустая, удаляем последнего абитуриента

if (!applicants.empty()) {

applicants.pop\_back();

} else {

cout << "Группа уже пуста." << endl;

}

}

void ShowGroup() {

// Вывод информации об абитуриентах в группе

for (int i = 0; i < applicants.size(); ++i) {

cout << "Заявитель " << i + 1 << ":" << endl;

applicants[i].Output();

cout << endl;

}

}

double CalculateAvgAge() {

// Подсчет среднего возраста в группе

int sum = 0;

for (auto& applicant : applicants) {

sum += applicant.GetAge();

}

return static\_cast<double>(sum) / applicants.size();

}

int GetSize() {

// Возвращаем размер группы

return applicants.size();

}

};

// Класс для работы с интерактивным диалогом для управления группой

class Dialog : public Group {

public:

void Execute() {

char choice;

bool running = true;

Group group;

while (running) {

cout << endl << "=================================================================================================================" << endl;

cout << "Введите команду: ";

cin >> choice;

switch (choice) {

case 'm': {

int n;

cout << "Введите количество абитуриентов: ";

cin >> n;

for (int i = 0; i < n; ++i) {

Applicant applicant;

applicant.Input();

group.AddApplicant(applicant);

}

break;

}

case '+': {

Applicant newApplicant;

newApplicant.Input();

group.AddApplicant(newApplicant);

break;

}

case '-': {

group.RemoveApplicant();

break;

}

case 's':

group.ShowGroup();

break;

case 'z':

cout << "Средний возраст: " << group.CalculateAvgAge() << endl;

break;

case 'q':

running = false;

break;

default:

cout << "Неверная команда. Пожалуйста, попробуйте еще раз." << endl;

}

cout << endl << "=================================================================================================================" << endl;

}

}

};

int main() {

setlocale(LC\_ALL, "ru");

cout << "m: Создать группу\n+: Добавить элемент в группу\n";

cout << "-: Удалить элемент из группы\ns: Вывести информацию об элементах группы\n";

cout << "z: Вывести информацию о среднем возрасте \nq: Конец работы\n";

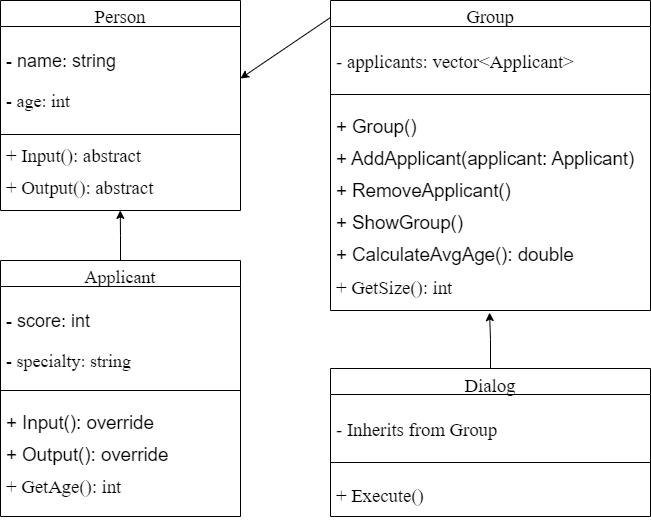
Dialog D;

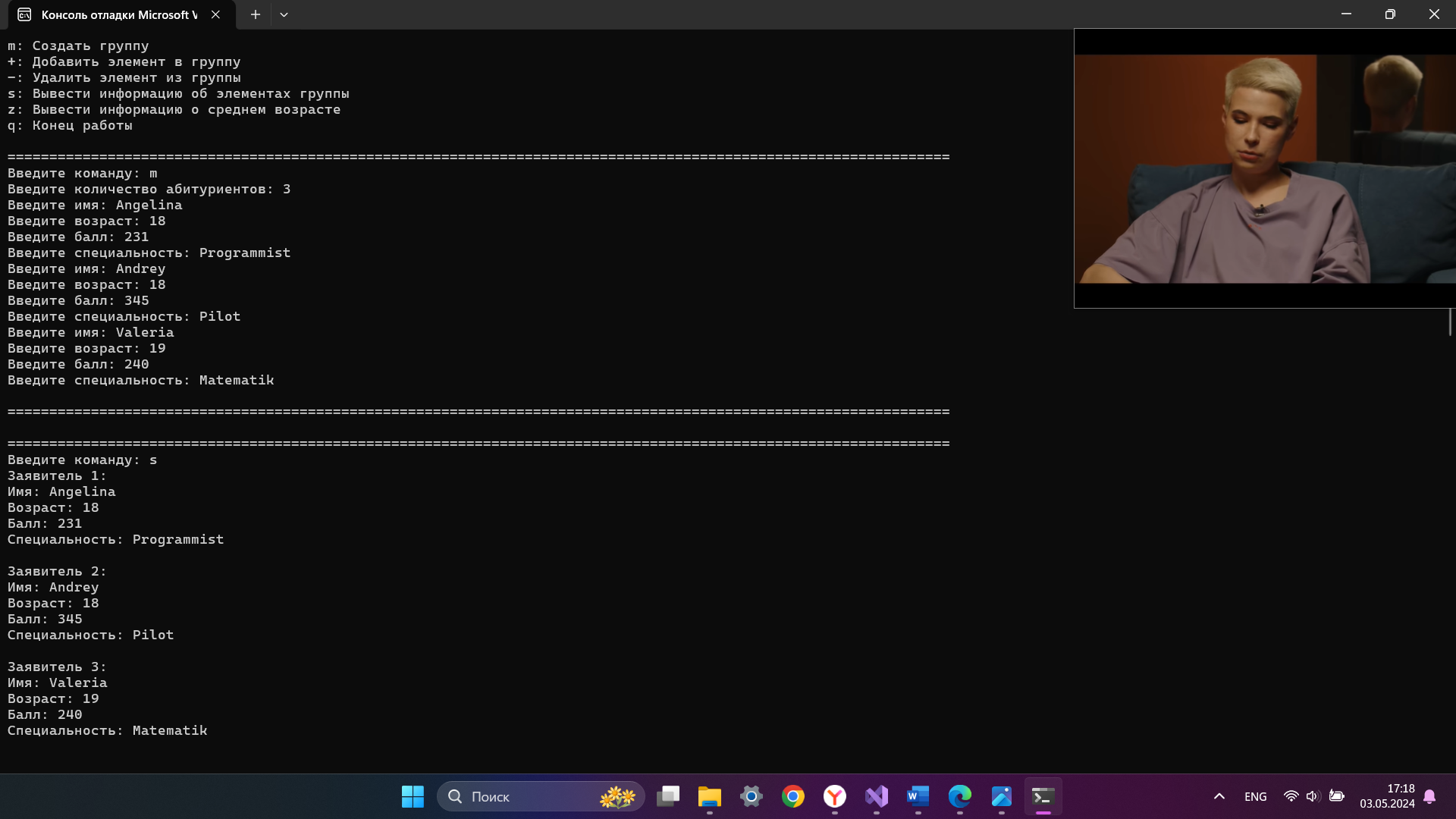
D.Execute();

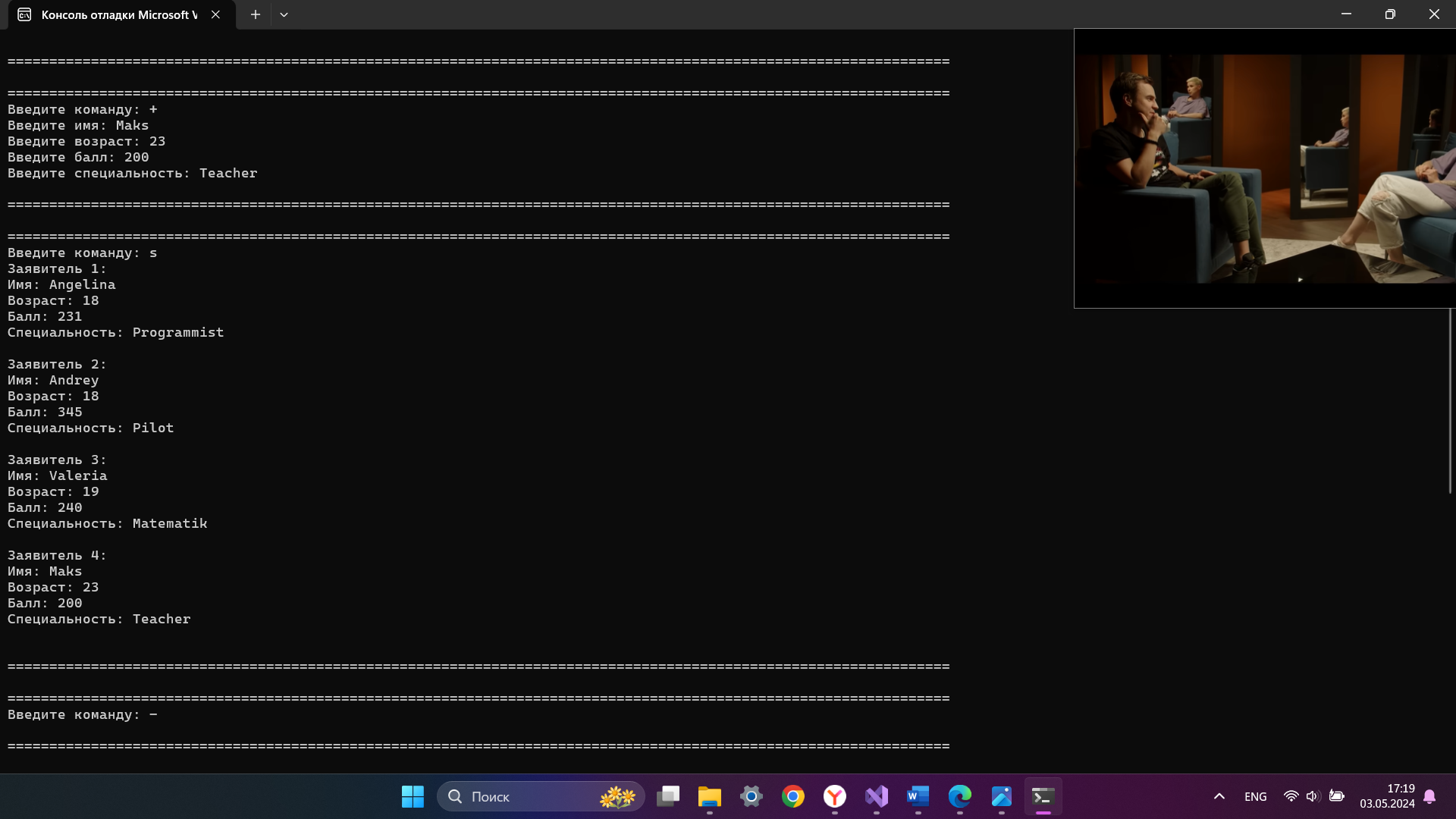
return 0;

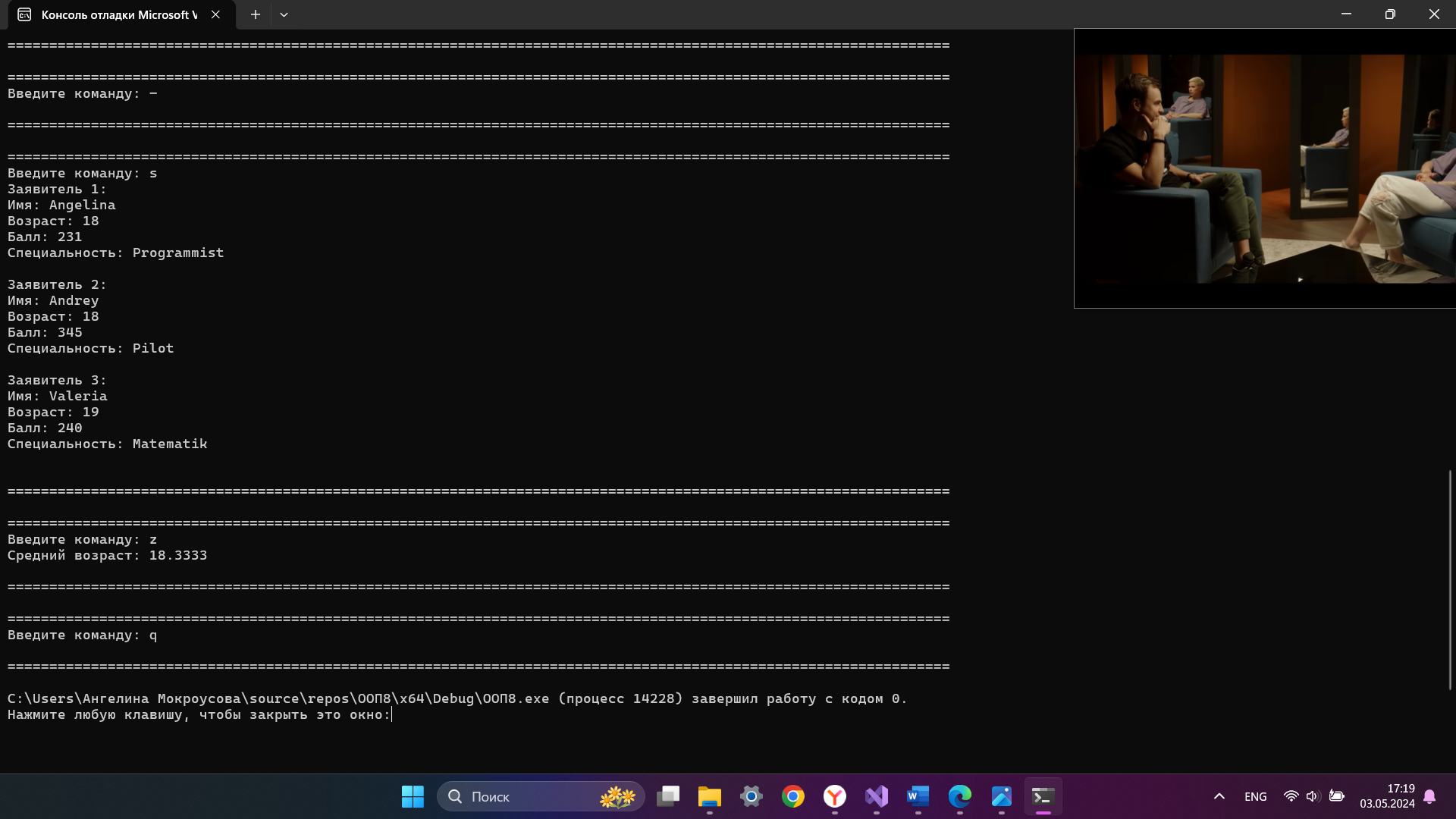
}

**UML диаграмма:**

**Скриншот работы программы:**







**Вывод:**

Мною была аписана программа, которая успешно выполняет поставленную перед ней задачу. В процессе написания был повторен и закреплен на практике пройденный ранее материал.

**Ответы на вопросы:**

1. Что такое класс-группа? Привести примеры таких классов.

Ответ: Класс-группа - это класс, который содержит коллекцию объектов другого типа. Примеры:

vector<int> (коллекция целых чисел)

list<string> (коллекция строк)

map<string, int> (коллекция пар ключ-значение, где ключ - строка, а значение - целое число)

1. Привести пример описания класса-группы Список (List).

Ответ:

class List {

private:

vector<int> elements;

public:

// ... (конструкторы, методы добавления, удаления и поиска элементов)

};

1. Привести пример конструктора (с параметром, без параметров, копирования) для класса-группы Список.

Ответ:

// Конструктор по умолчанию

List() {}

// Конструктор с параметром

List(int size) {

elements.resize(size);

}

// Конструктор копирования

List(const List& other) {

elements = other.elements;

}

1. Привести пример деструктора для класса-группы Список.

Ответ:

~List() {

elements.clear();

}

1. Привести пример метода для просмотра элементов для класса-группы Список.

Ответ:

void PrintElements() {

for (int element : elements) {

cout << element << " ";

}

cout << endl;

}

1. Какой вид иерархии дает группа?

Ответ: Линейную иерархию (группа содержит однородные элементы)

1. Почему во главе иерархии классов, содержащихся в группе объектов должен находиться абстрактный класс?

Ответ: Чтобы обеспечить единообразный интерфейс для всех классов в группе и предотвратить дублирование кода.

1. Что такое событие? Для чего используются события?

Ответ: Событие - это происшествие, которое происходит в системе. События используются для уведомления объектов о том, что произошло определенное событие, чтобы они могли соответствующим образом отреагировать.

1. Какие характеристики должно иметь событие-сообщение?

Ответ: Событие-сообщение должно иметь следующие характеристики:

Тип (что произошло)

Источник (кто сгенерировал сообщение)

Данные (дополнительная информация о событии)

1. Привести пример структуры, описывающей событие.

Ответ:

struct Event {

enum Type {

MOUSE\_CLICK, KEY\_PRESS, MESSAGE\_RECEIVED

};

Type type;

int source;

string data;

};

1. Какие значения, и в каких случаях присваиваются полю command? Ответ: Полю command присваивается код команды:

В случае события типа MESSAGE\_RECEIVED (сообщение получено)

В случае события типа MOUSE\_CLICK, если нажата левая кнопка мыши

В случае события типа KEY\_PRESS, если нажата клавиша пробела

1. Какие значения, и в каких случаях присваиваются полю command? Ответ: Полю command присваивается код команды:

В случае события типа MESSAGE\_RECEIVED (сообщение получено)

В случае события типа KEY\_PRESS, если нажата клавиша пробела

1. Для чего используются поля a и message?

Ответ: Поле a используется как вспомогательное поле для хранения дополнительной информации, связанной с командой. Поле message используется для хранения сообщения, связанного с событием.

1. Какие методы необходимы для организации обработки сообщений? Ответ:

GetEvent(): Получение следующего события из очереди событий

HandleEvent(): Обработка полученного события

ClearEvent(): Очистка очереди событий

1. Какой вид имеет главный цикл обработки событий-сообщений?

Ответ:

while (true) {

Event event = GetEvent();

HandleEvent(event);

}

1. Какую функцию выполняет метод ClearEvant()? Каким образом?

Ответ: Метод ClearEvent() очищает очередь событий, удаляя из нее все события. Он делает это путем вызова метода pop() для стека событий.

1. Какую функцию выполняет метод HandleEvent()?Каким образом?

Ответ: Метод HandleEvent() обрабатывает полученное событие. Он переключается в зависимости от типа события и выполняет соответствующий код обработки.

1. Какую функцию выполняет метод GetEvent()?

Ответ: Метод GetEvent() извлекает следующее событие из очереди событий и возвращает его вызывающей функции.

1. Для чего используется поле Endstate? Какой класс (объект) содержит это поле?

Ответ: Поле Endstate используется для указания, должен ли главный цикл обработки событий завершить свою работу. Обычно оно содержится в глобальном объекте приложения.

1. Для чего используется функция Valid()?

Ответ: Функция Valid() проверяет, является ли данное событие допустимым. Недопустимые события обычно удаляются из очереди событий.

**Скриншот с GitHab:**

