**Технологии конструирования программного обеспечения**

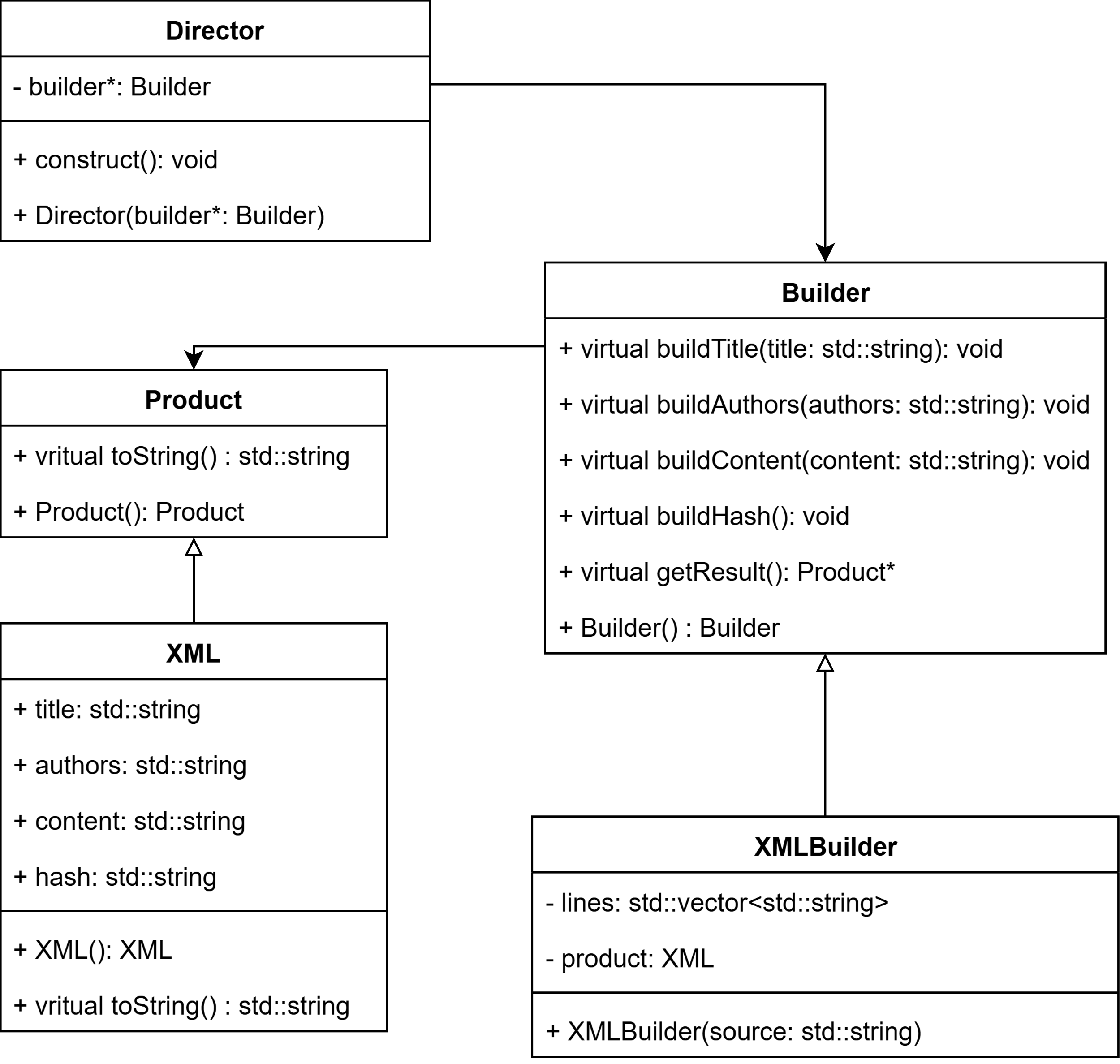
**Отчет по лабораторной работе № 3**

**Группа:** 221-3210 **Студент:** Обухов Алексей Сергеевич

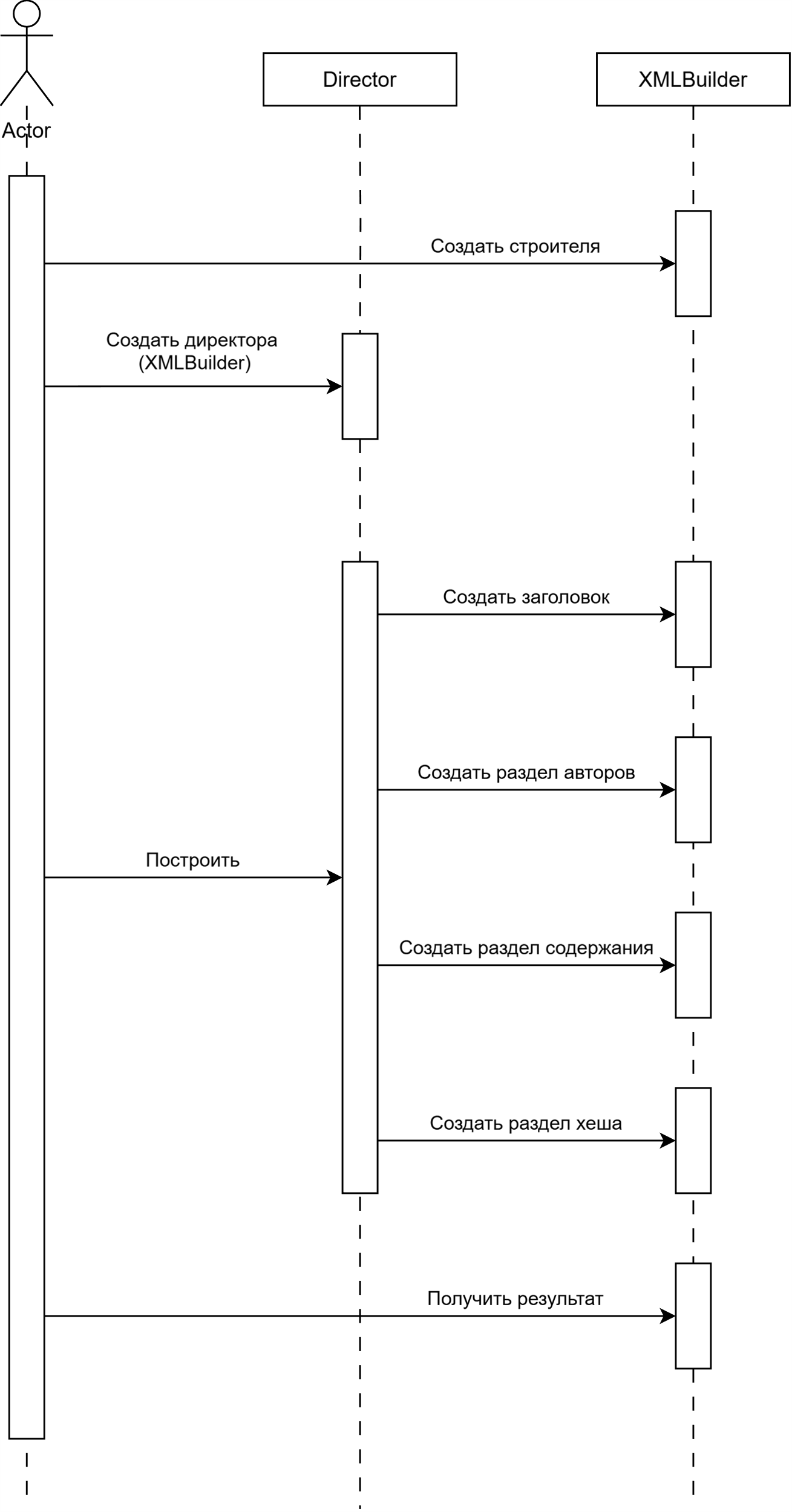
**Задание на лабораторную работу**

1. Паттерн Builder. Имеется текст статьи в формате TXT. Статья состоит из заголовка (первая строка), фамилий авторов (вторая строка), самого текста статьи и хеш-кода текста статьи (последняя строка). Написать приложение, позволяющее конвертировать документ в формате ТХТ в документ формата XML. Необходимо также проверять корректность хеш-кода статьи.

**Диаграмма классов**

****

**Диаграмма последовательности**

****

**Исходный код программы**

Файл «Builder.h»

#pragma once

#include <string>

#include <vector>

#include <sstream>

#include <cstdlib>

// Базовый класс продукта

class Product {

public:

Product() {};

// Методы преобразования продукта в строку

virtual std::string toString() = 0;

virtual bool checkHash() = 0;

};

// Класс XML

class XML : public Product {

public:

std::string title; // Заголовок

std::string authors; // Авторы

std::string content; // Содержание

std::string hash; // Хеш-код статьи

unsigned long originalHash = 0; // Хеш, полученный из файла

unsigned long calculatedHash = 0; // Хеш, полученный из строителя

XML() {};

// Преобразование XML в строку

std::string toString() override {

std::string result;

result += "<?xml version=\"1.0\"?>\n";

result += "<ARTICLE>\n";

result += title + "\n";

result += authors + "\n";

result += content + "\n";

result += hash + "\n";

result += "</ARTICLE>\n";

return result;

}

bool checkHash() override {

return originalHash == calculatedHash;

}

};

// Базовый класс строителя

class Builder {

public:

Builder() {};

virtual void buildTitle() = 0;

virtual void buildAuthors() = 0;

virtual void buildContent() = 0;

virtual void buildHash() = 0;

virtual Product\* getResult() = 0;

};

// Класс строителя XML

class XMLBuilder : public Builder {

// Исходный текст, из которого формируется XML

protected:

std::vector<std::string> lines;

XML\* product;

public:

void buildTitle() override {

product->title = "<TITLE>" + lines[0] + "</TITLE>";

};

void buildAuthors() override {

product->authors = "<AUTHORS>" + lines[1] + "</AUTHORS>";

};

void buildContent() override {

product->content = "<CONTENT>\n";

for (int i = 2; i < lines.size() - 1; i++)

product->content += lines[i] + "\n";

product->content += "</CONTENT>";

};

void buildHash() override {

std::hash<std::string> hasher;

std::string fileContent;

for (int i = 0; i < lines.size() - 1; i++)

fileContent += lines[i];

unsigned int calculatedHash = hasher(fileContent);

std::cout << calculatedHash << "\n\n";

product->hash = "<HASH>" + lines[lines.size() - 1] + "</HASH>";

product->originalHash = atol(lines[lines.size() - 1].c\_str());

product->calculatedHash = calculatedHash;

};

Product\* getResult() override {

return product;

}

XMLBuilder(std::string source) {

product = new XML();

std::stringstream ss(source);

std::string line;

while (std::getline(ss, line)) {

if (!line.empty())

lines.push\_back(line);

}

}

};

// Класс директора

class Director {

protected:

Builder\* builder;

public:

Director(Builder\* builder) {

this->builder = builder;

}

void construct() {

builder->buildTitle();

builder->buildAuthors();

builder->buildContent();

builder->buildHash();

}

};

Файл «main.cpp»

#include <iostream>

#include <string>

#include <fstream>

#include "Builder.h"

std::string readFile(std::string path) {

std::string content;

std::ifstream file(path);

for (std::string line; std::getline(file, line);)

content += line + "\n";

file.close();

return content;

}

int main() {

setlocale(LC\_ALL, "Russian");

std::string fileContent = readFile("test.txt");

std::cout << "Оригинал файла:\n";

std::cout << fileContent << "\n";

XMLBuilder builder(fileContent);

Director director(&builder);

director.construct();

Product\* xml = builder.getResult();

std::cout << xml->toString() << "\n";

std::cout << "XML хеш верный?: " << (xml->checkHash() ? "Да" : "Нет") << "\n";

return 0;

}

2. Паттерн Abstract Factory. Разработать систему Кинопрокат. Пользователь может выбрать определённую киноленту, при заказе киноленты указывается язык звуковой дорожки, который совпадает с языком файла субтитров. Система должна поставлять фильм с требуемыми характеристиками, причём при смене языка звуковой дорожки должен меняться и язык файла субтитров и наоборот.

**Диаграмма классов**

**Диаграмма последовательности**

**Исходный код программы**