МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное

образовательное учреждение высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»**

Кафедра математической кибернетики и компьютерных наук

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4**

по дисциплине «Проектирование архитектуры информационных систем»

студента 3 курса 351 группы

направления 09.03.04 Программная инженерия

факультета компьютерных наук и информационных технологий

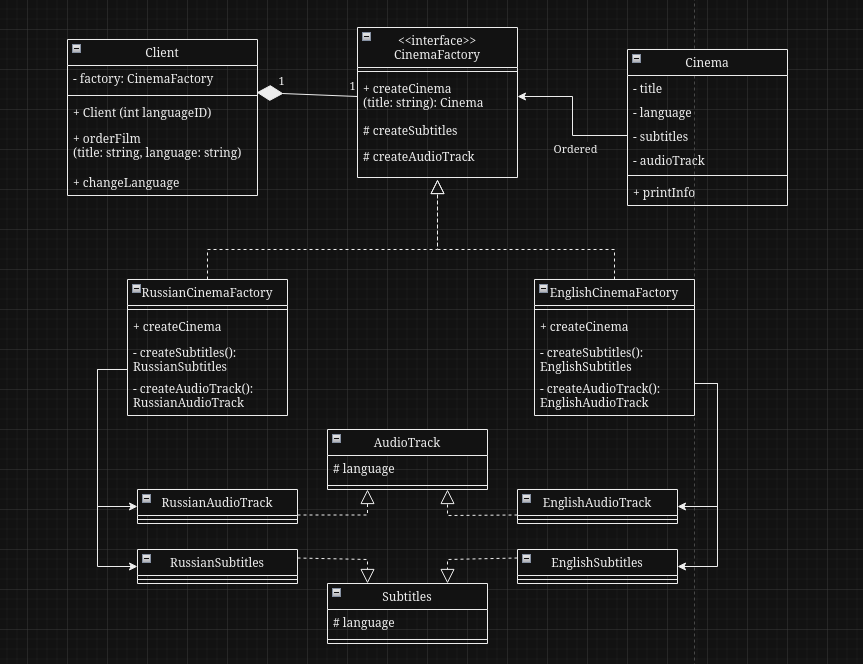
Устюшина Богдана Антоновича

Проверил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Папшев С. В.

**Тема: шаблон Abstract Factory (вариант 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.

**Созданная UML-диаграмма:**



**В данном классе:**

AudioTrack, Subtitles – классы, которые наследуются конкретными языковыми реализациями (английский, русский)

RussianAudioTrack, EnglishAudioTrack, RussianSubtitles, EnglishSubtitles – классы, наследующие общий тип объекта.

CinemaFactory – класс абстрактной фабрики, который наследуется двумя конкретными фабриками (в зависимости от языка)

RussianCinemaFactory, EnglishCinemaFactory – классы, наследующие абстрактную фабрику и по-своему реализующие создания субтитров и аудиодорожки.

Cinema – класс, объединяющий объект созданного кино.

Client – пользовательский интерфейс.

**Код программы:**

**#include <iostream>**

using namespace std;

enum Language

{

English,

Russian

};

class CinemaFactory;

class Cinema;

//Класс аудиодорожка

class AudioTrack

{

protected:

string language;

public:

AudioTrack()

{

cout << "[SYSTEM] AudioTrack created!" << endl;

}

};

//Класс субтитров

class Subtitles

{

protected:

string language;

public:

Subtitles()

{

cout << "[SYSTEM] Subtitles created!" << endl;

}

};

//Наследующий класс английской аудиодорожки

class EnglishAudioTrack : public AudioTrack

{

public:

EnglishAudioTrack()

{

this->language = "English";

cout << "[SYSTEM] English AudioTrack created!" << endl;

}

};

//Наследующий класс английских субтитров

class EnglishSubtitles : public Subtitles

{

public:

EnglishSubtitles()

{

this->language = "English";

cout << "[SYSTEM] English Subtitles created!" << endl;

}

};

//Наследующий класс русской аудиодорожки

class RussianAudioTrack : public AudioTrack

{

public:

RussianAudioTrack()

{

this->language = "Russian";

cout << "[SYSTEM] Russian AudioTrack created!" << endl;

}

};

//Наследующий класс русских субтитров

class RussianSubtitles : public Subtitles

{

public:

RussianSubtitles()

{

this->language = "Russian";

cout << "[SYSTEM] Russian Subtitles created!" << endl;

}

};

//Класс кино

class Cinema

{

public:

Cinema(string title, string language, Subtitles subtitles, AudioTrack audioTrack)

{

this->title = title;

this->language = language;

this->subtitles = subtitles;

this->audioTrack = audioTrack;

}

void printInfo()

{

cout << endl << "=FILM INFO=" << endl << title << endl << language << endl << endl;

}

private:

string title;

string language;

Subtitles subtitles;

AudioTrack audioTrack;

};

//Интерфейс (абстрактная фабрика)

class CinemaFactory

{

public:

virtual Cinema createCinema(string title) = 0;

virtual ~CinemaFactory() {}

protected:

virtual Subtitles createSubtitles() = 0;

virtual AudioTrack createAudioTrack() = 0;

};

//Наследующий класс английской абстрактной фабрики

class EnglishCinemaFactory : public CinemaFactory

{

public:

EnglishCinemaFactory()

{

cout << "[SYSTEM] English CinemaFactory created!" << endl;

}

Cinema createCinema(string title) override

{

return Cinema(

title,

"English",

createSubtitles(),

createAudioTrack()

);

}

private:

Subtitles createSubtitles() override

{

return EnglishSubtitles();

}

AudioTrack createAudioTrack() override

{

return EnglishAudioTrack();

}

};

//Наследующий класс русской абстрактной фабрики

class RussianCinemaFactory : public CinemaFactory

{

public:

RussianCinemaFactory()

{

cout << "[SYSTEM] Russian CinemaFactory created!" << endl;

}

Cinema createCinema(string title) override

{

return Cinema(

title,

"Russian",

createSubtitles(),

createAudioTrack()

);

}

Subtitles createSubtitles() override

{

return RussianSubtitles();

}

AudioTrack createAudioTrack() override

{

return RussianAudioTrack();

}

};

//Класс клиента

class Client

{

private:

CinemaFactory\* factory;

public:

Client(int languageID)

{

factory = nullptr;

changeLanguage(languageID);

}

Cinema orderFilm(string title)

{

return factory->createCinema(title);

}

void changeLanguage(int languageID)

{

if (factory != nullptr)

delete factory;

if (languageID == English)

factory = new EnglishCinemaFactory();

else if (languageID == Russian)

factory = new RussianCinemaFactory();

else

cout << "[SYSTEM] This language is not supported!" << endl;

}

};

//Примеры использования

int main()

{

Client cl = Client(English);

cl.orderFilm("Title1").printInfo();

cl.changeLanguage(Russian);

cl.orderFilm("Title1").printInfo();

cl.orderFilm("Title2").printInfo();

}