Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Алтайский государственный технический университет им. И. И. Ползунова»

Факультет информационных технологий Кафедра прикладной математики

Отчет защищен с оценкой

Преподаватель (подпись)

« » 2023 г.

Отчет

по лабораторной работе № 8

**«** Структурные паттерны 2 - проектирование (Bridge, Flyweight)

по дисциплине «Объектно-ориентированное программирование»

Студент гр. ПИ-02 Чередов Р.А.

Ассистент кафедры ПМ,

Рахманин Д. С.

Барнаул 2023

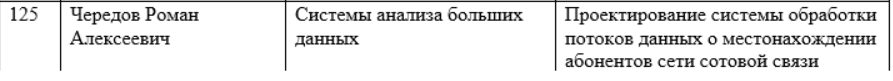
**Тема:** Структурные паттерны

(Bridge, Flyweight)

# Отчет:

Диаграмма классов

Краткое описание назначения классов в системе Постановка задачи конфигурирования системы



# Bridge

#include <iostream>

#include <cctype>

using namespace std;

class Screen { // Интерфейс реализации

public:

virtual void WriteText(const string& str) = 0;

virtual ~Screen() {};

};

class Abstraction { // Абстракция

protected:

Screen\* impl;

public:

virtual void WriteAnswer(const string& str) = 0;

Abstraction(Screen\* inImpl) { impl = inImpl; }

~Abstraction() {}

};

class Text : public Abstraction {//Вывод в формате текста

public:

virtual void WriteAnswer(const string& str) {

impl->WriteText(str);

}

Text(Screen\* inImpl) : Abstraction(inImpl) {}

~Text() {}

};

class Table : public Abstraction {//Вывод в формате таблицы

public:

virtual void WriteAnswer(const string& str) {

cout << " ";

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

cout << "\_";

}

cout << endl;

impl->WriteText("|" + str + "|");

cout << endl;

cout << "|";

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

cout << "\_";

}

cout << "|";

}

Table(Screen\* inImpl) : Abstraction(inImpl) {}

~Table() {}

};

class ForPC : public Screen { // конкретная реализация для ПК

public:

virtual void WriteText(const string& str) {

cout << str;

}

~ForPC() {}

};

class ForPhone : public Screen { // конкретная реализация для телефона

public:

virtual void WriteText(const string& str) {

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

char ch = toupper(str[i]);

cout << ch;

}

}

~ForPhone() {}

};

int main() {

//Текстовая рекомендация на ПК

Text\* txtPC = new Text(new ForPC());

cout << "Text ( ForPC )" << endl;

txtPC->WriteAnswer("Answer");

//Текстовая рекомендация на смартфон

Text\* txtPH = new Text(new ForPhone());

cout << endl << endl << "Text ( ForPhone )" << endl;

txtPH->WriteAnswer("Answer");

//Табличная рекомендация на ПК

Table\* tblPC = new Table(new ForPC());

cout << endl << endl << "Table( ForPC )" << endl;

tblPC->WriteAnswer("Answer");

//Табличная рекомендация на смартфон

Table\* tblPH = new Table(new ForPhone());

cout << endl << endl << " Table( ForPhone )" << endl;

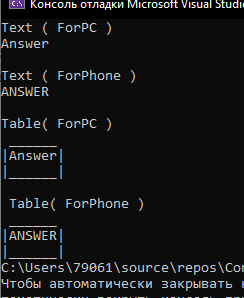
tblPH->WriteAnswer("Answer");

delete tblPC, tblPH;

delete txtPC, txtPH;

return 0;

}



# Flyweight

#include <iostream>

#include <map>

#include <random>

#include <time.h>

using namespace std;

//Сигнал

class Signal {

protected:

int Code; //Кода сигнала

public:

Signal(int Code) { this->Code = Code; }

virtual void display() = 0;

};

class AlarmSystem : public Signal {

public:

AlarmSystem(int Code) : Signal(Code) {}

virtual void display() { cout << "AlarmSystem" << endl; }

};

class LineSystem : public Signal {

public:

LineSystem(int Code) : Signal(Code) {}

virtual void display() { cout << "LineSystem" << endl; }

};

class ServerSystem : public Signal {

public:

ServerSystem(int Code) : Signal(Code) {}

virtual void display() { cout << "ServerSystem" << endl; }

};

//Компьютер - фабрика сигналов

class Computer {

private:

typedef map<int, Signal\*> ArraySignal;

ArraySignal mArraySignal;

public:

Signal\* getSignal(int inCode) {

cout << inCode % 10 << " grade - ";

ArraySignal::iterator it = mArraySignal.find(inCode);

if (mArraySignal.end() == it) {//Если не найден, то создаем новый

cout << "(new) - ";

Signal\* f;

if (inCode / 10 == 1)

f = new AlarmSystem(inCode);

else {

if (inCode / 10 == 2)

f = new LineSystem(inCode);

else

f = new ServerSystem(inCode);

}

mArraySignal[inCode] = f;

return f;

}

else {//Если найден, то возвращаем

cout << "(return) - ";

return it->second;

}

}

~Computer() { //При удалении удаляем все созданные сигналы

ArraySignal::iterator it = mArraySignal.begin();

for (int i = 0; it != mArraySignal.end(); it++, i++) {

delete mArraySignal[i];

}

}

};

//Получить случайное число

int randomRange(int min, int max)

{

return int(double(rand()) / RAND\_MAX \* (max - min)) + min;

}

int main() {

srand(time(NULL));

Computer\* fact = new Computer();

const int CodeRange = 39; //Диапазон кода

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

fact->getSignal(randomRange(10, CodeRange))->display();

}

return 0;

delete fact;

# }

# 