Московский Авиационный Институт

(Национальный исследовательский Университет)

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

**Лабораторной работе № 07**

**по курсу «Объектно-ориентированное программирование»**

**Тема:**

**« Проектирование структуры классов»**

|  |  |
| --- | --- |
| Студент: | Пшеницын А. А. |
| Группа: | М80-208Б-18 |
| Преподаватель: | Журавлев А. А. |
| Вариант: | 17 |
| Оценка: |  |
| Дата: |  |

Москва

2019

**Цель:**

Цель:

• Получение практических навыков в хороших практиках проектирования структуры классов приложения;

**Задание**

Спроектировать простейший графический векторный редактор.

Требование к функционалу редактора:

• создание нового документа

• импорт документа из файла

• экспорт документа в файл

• создание графического примитива (согласно варианту задания)

• удаление графического примитива

• отображение документа на экране (печать перечня графических объектов и их характеристик)

• реализовать операцию undo, отменяющую последнее сделанное действие. Должно действовать для операций добавления/удаления фигур.

Требования к реализации:

• Создание графических примитивов необходимо вынести в отдельный класс – Factory.

• Сделать упор на использовании полиморфизма при работе с фигурами;

• Взаимодействие с пользователем (ввод команд) реализовать в функции main;

**Код**

**main.cpp**

#include <iostream>

#include <string>

#include <vector>

#include <memory>

#include "figure.h"

#include "factory.h"

#include "command.h"

int main() {

factory fact;

std::unique\_ptr<doc\_oper> doc = std::make\_unique<doc\_oper>();

while(1){

std::string command;

std::cin >> command;

if(command == "save"){

std::string path;

std::cin >> path;

std::ofstream os(path);

doc -> save(os);

os.close();

}else if(command=="load"){

std::string path;

std::cin >> path;

std::ifstream is(path);

if(is) {

doc -> load(is);

}else {

std::cout << "No such file\n";

}

is.close();

}else if(command=="add"){

size\_t id;

std::cin >> id;

doc -> add(std::cin, id);

}else if(command=="erase"){

size\_t id;

std::cin >> id;

doc -> erase(id);

}else if(command=="print"){

doc -> print(std::cout);

}else if(command == "undo"){

doc -> undo();

} else if(command == "exit"){

break;

} else{

std::cout << "EROOR";

}

}

return 0;

}

**document.h**

#ifndef D\_DOCUMENT\_H\_

#define D\_DOCUMENT\_H\_

#include <fstream>

#include <cstdint>

#include <memory>

#include <string>

#include <algorithm>

#include "figure.h"

#include <vector>

#include "factory.h"

struct document{

public:

document() = default;

void save\_fig(std::ostream& os) const {

for (size\_t i = 0; i < figs.size(); ++i) {

figs[i] -> print\_dop(os);

}

}

void load\_fig(std::ifstream& is){

while(!is.eof()){

std::shared\_ptr<fig> ptr;

ptr = fact.fig\_create\_file(is);

if(ptr == nullptr){

break;

}

figs.push\_back(ptr);

}

}

void print\_fig(std::ostream& os) const {

if(figs.size() == 0) {

os << "Empty\n";

return;

}

for (size\_t i = 0; i < figs.size(); ++i) {

os << "figure " << i << ":" << "\n";

figs[i] -> print(os);

os << "center :" << "[" << figs[i] -> center().x << ", " << figs[i] -> center().y << "]" << "\n";

os << "square :" << figs[i] -> square() << "\n";

}

}

void add\_fig(std::istream& is, size\_t id){

std::shared\_ptr<fig> ptr = fact.fig\_create(is);

figs.insert(figs.begin() + id, ptr);

}

void erase\_fig(size\_t id){

figs.erase(figs.begin() + id);

}

std::shared\_ptr<fig> get\_fig(size\_t id) {

if (id >= figs.size()) {

return nullptr;

}

return figs[id];

}

void add\_fig\_dop(std::shared\_ptr<fig>& ptr, size\_t id){

figs.insert(figs.begin() + id, ptr);

}

private:

factory fact;

std::vector<std::shared\_ptr<fig>> figs;

};

#endif

**command.h**

#ifndef OOP7\_COMMAND\_H

#define OOP7\_COMMAND\_H

#include "document.h"

#include "document.h"

struct doc\_oper : private document{

private:

struct command{

size\_t id;

std::shared\_ptr<fig> ptr;

virtual void undo(document &doc) = 0;

};

std::vector<std::shared\_ptr<command>> opers;

struct add\_com : public command{

void undo(document &doc) override {

doc.erase\_fig(id);

}

};

struct remove\_com : public command{

void undo(document &doc) override {

doc.add\_fig\_dop(ptr, id);

}

};

public:

void add(std::istream& is, size\_t id){

add\_fig(is, id);

std::shared\_ptr<fig> fig = get\_fig(id);

std::shared\_ptr<add\_com> op = std::make\_shared<add\_com>();

op -> id = id;

op -> ptr = fig;

opers.push\_back(op);

}

void erase(size\_t id){

std::shared\_ptr<fig> fig = get\_fig(id);

erase\_fig(id);

std::shared\_ptr<remove\_com> op = std::make\_shared<remove\_com>();

op -> id = id;

op -> ptr = fig;

opers.push\_back(op);

}

void undo(){

if(opers.size() == 0){

std::cout << "EMPTY";

return;

}

opers[opers.size() - 1] -> undo(\*this);

opers.pop\_back();

}

void save(std::ostream& os){

save\_fig(os);

}

void load(std::ifstream& is){

load\_fig(is);

}

void print(std::ostream& os){

print\_fig(os);

}

};

#endif //OOP7\_COMMAND\_H

**factory.h**

#ifndef D\_FACTORY\_H

#define D\_FACTORY\_H

#include <memory>

#include <iostream>

#include <fstream>

#include "trigon.h"

#include "rectangle.h"

#include "quadrate.h"

#include <string>

struct factory {

std::shared\_ptr<fig> fig\_create(std::istream& is);

std::shared\_ptr<fig> fig\_create\_file(std::ifstream& is);

};

std::shared\_ptr<fig> factory::fig\_create(std::istream &is) {

try {

std::string comm;

is >> comm;

if (comm == "trigon") {

return std::shared\_ptr<fig>(new trigon(is));

} else if (comm == "quadrate") {

return std::shared\_ptr<fig>(new quadrate(is));

} else if (comm == "rectangle") {

return std::shared\_ptr<fig>(new rectangle(is));

} else {

throw "This is not a figure";

}

} catch (const char\* f){

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

return nullptr;

}

}

std::shared\_ptr<fig> factory::fig\_create\_file(std::ifstream &is) {

try {

std::string comm;

is >> comm;

if (comm == "trigon") {

return std::shared\_ptr<fig>(new trigon(is));

} else if (comm == "quadrate") {

return std::shared\_ptr<fig>(new quadrate(is));

} else if (comm == "rectangle") {

return std::shared\_ptr<fig>(new rectangle(is));

} else {

throw " ";

}

} catch (const char\* f){

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

return nullptr;

}

}

#endif

**Ссылка на репозиторий на GitHub**

<https://github.com/AlexPshen/oop_exercise_06.git>

**Тесты**

**test\_01.txt**

add 0 trigon 0 0 0 1 1 0

print

add 1 quadrate 0 0 1 1 0 1 1 0

print

undo

print

save file

exit

**res\_01.txt**

add 0 trigon 0 0 0 1 1 0

print

figure 0:

trigon: (0, 0) (0, 1) (1, 0)

center :[0.333333, 0.333333]

square :0.5

add 1 quadrate 0 0 1 1 0 1 1 0

print

figure 0:

trigon: (0, 0) (0, 1) (1, 0)

center :[0.333333, 0.333333]

square :0.5

figure 1:

quadrate: (0, 0) (1, 1) (0, 1) (1, 0)

center :[0.5, 0.5]

square :1

undo

print

figure 0:

trigon: (0, 0) (0, 1) (1, 0)

center :[0.333333, 0.333333]

square :0.5

save file

exit

**test\_02.txt**

load file

print

add 0 rectangle 0 0 0 1 1 0 1 1

print

erase

0

print

indo

print

save

**Объяснение результатов работы программы**

Данная программа предназначена для обработки фигур, таких как треугольник, квадрат, прямоугольник. Опишем команды-обработчики:

* save – сохранение фигур в файл
* load – загрузка фигур из файла
* add – добавление фигуры
* erase – удаление фигуры
* print – печать характеристик всех фигур
* undo – отмена последней команды (может отменять только add и erase)
* exit – конец программы

**Вывод**

В данной лабораторной работе мы попробовали спроектировать нормальную структуру классов. Данная работа направлена на приучение к написанию хороших классов, которые должны иметь наименьшее возможное количество связей между собой. Это делается для того, чтобы было проще читать код, а также для более легкого редактирования данного кода впоследствии.