Московский Авиационный Институт

(Национальный Исследовательский Университет)

Кафедра 806 «Вычислительная информатика и программирование»

Факультет: «Информационные технологии и прикладная математика»

Лабораторная работа

Дисциплина: «Объектно-ориентированное программирование»

III семестр

Задание 7: «Проектирование структуры классов»

|  |  |
| --- | --- |
| Группа: | М8О-208Б-18, №2 |
| Студент: | Алексеева Мария |
| Преподаватель: | Журавлёв Андрей Андреевич |
| Оценка: |  |
| Дата: | 30.12.2019 |

Москва, 2019

1. **Тема**: Проктирование структуры классов
2. **Цель работы**: Получение практических навыков в хороших практиках проектирования структуры классов приложения
3. **Задание** (*вариант № 2* ):

Фигуры — прямоугольник, трапеция, квадрат.

1. **Адрес репозитория на GitHub** [https://github.com/PowerMasha/oop\_exercise\_0](https://github.com/wAlienUFOx/oop_exercise_01)7
2. **Код программы на С++**

main.cpp

#include <iostream>

#include "factory.h"

#include "editor.h"

void menu() {

std::cout << "menu\n"

"create\n"

"load\n"

"save\n"

"add\n"

"remove\n"

"print\n"

"undo\n"

"exit\n";

}

void create(editor& edit) {

std::string tmp;

std::cout << "Enter name of new document\n";

std::cin >> tmp;

edit.CreateDocument(tmp);

std::cout << "Document create\n";

}

void load(editor& edit) {

std::string tmp;

std::cout << "Enter path to the file\n";

std::cin >> tmp;

try {

edit.LoadDocument(tmp);

std::cout << "Document loaded\n";

} catch (std::runtime\_error& e) {

std::cout << e.what();

}

}

void save(editor& edit) {

std::string tmp;

try {

edit.SaveDocument();

std::cout << "save document\n";

} catch (std::runtime\_error& e) {

std::cout << e.what();

}

}

void add(editor& edit) {

factory fac;

try {

std::shared\_ptr<figure> newElem = fac.FigureCreate(std::cin);

edit.InsertInDocument(newElem);

} catch (std::logic\_error& e) {

std::cout << e.what() << '\n';

}

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

}

void remove(editor& edit) {

uint32\_t index;

std::cout << "Enter index\n";

std::cin >> index;

try {

edit.DeleteInDocument(index);

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

} catch (std::logic\_error& err) {

std::cout << err.what() << "\n";

}

}

int main() {

editor edit;

std::string command;

while (true) {

std::cin >> command;

if (command == "menu") {

menu();

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

create(edit);

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

load(edit);

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

save(edit);

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

break;

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

add(edit);

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

remove(edit);

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

edit.PrintDocument();

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

try {

edit.Undo();

} catch (std::logic\_error& e) {

std::cout << e.what();

}

} else {

std::cout << "Unknown command\n";

}

}

return 0;

}

point.h

#ifndef OOP\_POINT\_H

#define OOP\_POINT\_H

#include <iostream>

struct point {

double x, y;

point (double a,double b) { x = a, y = b;};

point() = default;

};

//std::istream& operator >> (std::istream& is,point& p );

//std::ostream& operator << (std::ostream& os,const point& p);

std::istream& operator >> (std::istream& is,point& p ) {

return is >> p.x >> p.y;

}

std::ostream& operator << (std::ostream& os,const point& p) {

return os << p.x <<' '<< p.y;

}

#endif

command.h

#ifndef OOP\_COMMAND\_H

#define OOP\_COMMAND\_H

#include "document.h"

struct Acommand {

virtual ~Acommand() = default;

virtual void UnExecute() = 0;

protected:

std::shared\_ptr<document> doc\_;

};

struct InsertCommand : public Acommand {

public:

void UnExecute() override;

InsertCommand(std::shared\_ptr<document>& doc);

};

struct DeleteCommand : public Acommand {

public:

DeleteCommand(std::shared\_ptr<figure>& newFigure, uint32\_t newIndex,std::shared\_ptr<document>& doc);

void UnExecute() override;

private:

std::shared\_ptr<figure> figure\_;

uint32\_t index\_;

};

//====================================realize=============================

void InsertCommand::UnExecute() {

doc\_->RemoveLast();

}

InsertCommand::InsertCommand(std::shared\_ptr<document> &doc) {

doc\_ = doc;

}

DeleteCommand::DeleteCommand(std::shared\_ptr<figure> &newFigure, uint32\_t newIndex, std::shared\_ptr<document> &doc) {

doc\_ = doc;

figure\_ = newFigure;

index\_ = newIndex;

}

void DeleteCommand::UnExecute() {

doc\_->InsertIndex(figure\_,index\_);

}

#endif //OOP\_COMMAND\_H

document.h

#ifndef OOP\_DOCUMENT\_H

#define OOP\_DOCUMENT\_H

#include <fstream>

#include <cstdint>

#include <memory>

#include <string>

#include <algorithm>

#include "figure.h"

#include <vector>

#include "factory.h"

struct document {

public:

void Print() const ;

explicit document(std::string& newName): name\_(newName), factory\_(), buffer\_(0) {};

void Insert(std::shared\_ptr<figure>& ptr);

void Save (const std::string& filename) const;

void Load(const std::string& filename);

std::shared\_ptr<figure> GetFigure(uint32\_t index);

void Erase(uint32\_t index);

std::string GetName();

size\_t Size();

private:

friend class InsertCommand;

friend class DeleteCommand;

factory factory\_;

std::string name\_;

std::vector<std::shared\_ptr<figure>> buffer\_;

void RemoveLast();

void InsertIndex(std::shared\_ptr<figure>& newFigure, uint32\_t index);

};

void document::Print() const {

{

if (buffer\_.empty()) {

std::cout << "Buffer is empty\n";

}

for (auto elem : buffer\_) {

elem->print(std::cout);

}

}

}

void document::Insert(std::shared\_ptr<figure> &ptr) {

buffer\_.push\_back(ptr);

}

void document::Save(const std::string &filename) const {

std::ofstream fout;

fout.open(filename);

if (!fout.is\_open()) {

throw std::runtime\_error("File is not opened\n");

}

fout << buffer\_.size() << '\n';

for (auto elem : buffer\_) {

elem->printFile(fout);

}

}

void document::Load(const std::string &filename) {

std::ifstream fin;

fin.open(filename);

if (!fin.is\_open()) {

throw std::runtime\_error("File is not opened\n");

}

size\_t size;

fin >> size;

buffer\_.clear();

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

buffer\_.push\_back(factory\_.FigureCreateFile(fin));

}

name\_ = filename;

}

std::shared\_ptr<figure> document::GetFigure(uint32\_t index) {

return buffer\_[index];

}

void document::Erase(uint32\_t index) {

if ( index >= buffer\_.size()) {

throw std::logic\_error("Out of bounds\n");

}

buffer\_[index] = nullptr;

for (; index < buffer\_.size() - 1; ++index) {

buffer\_[index] = buffer\_[index + 1];

}

buffer\_.pop\_back();

}

std::string document::GetName() {

return this->name\_;

}

size\_t document::Size() {

return buffer\_.size();

}

void document::RemoveLast() {

if (buffer\_.empty()) {

throw std::logic\_error("Document is empty");

}

buffer\_.pop\_back();

}

void document::InsertIndex(std::shared\_ptr<figure> &newFigure, uint32\_t index) {

buffer\_.insert(buffer\_.begin() + index, newFigure);

}

#endif

editor.h

#ifndef OOP7\_EDITOR\_H

#define OOP7\_EDITOR\_H

#include "figure.h"

#include "document.h"

#include <stack>

#include "command.h"

struct editor {

private:

std::shared\_ptr<document> doc\_;

std::stack<std::shared\_ptr<Acommand>> history\_;

public:

~editor() = default;

void PrintDocument();

void CreateDocument(std::string& newName);

bool DocumentExist();

editor() : doc\_(nullptr), history\_()

{

}

void InsertInDocument(std::shared\_ptr<figure>& newFigure);

void DeleteInDocument(uint32\_t index);

void SaveDocument();

void LoadDocument(std::string& name);

void Undo();

};

//=================================realize=================================

void editor::PrintDocument() {

if (doc\_ == nullptr) {

std::cout << "No document!\n";

return;

}

doc\_->Print();

}

void editor::CreateDocument(std::string &newName) {

doc\_ = std::make\_shared<document>(newName);

}

bool editor::DocumentExist() {

return doc\_ != nullptr;

}

void editor::InsertInDocument(std::shared\_ptr<figure> &newFigure) {

if (doc\_ == nullptr) {

std::cout << "No document!\n";

return;

}

std::shared\_ptr<Acommand> command = std::shared\_ptr<Acommand>(new InsertCommand(doc\_));

doc\_->Insert(newFigure);

history\_.push(command);

}

void editor::DeleteInDocument(uint32\_t index) {

if (doc\_ == nullptr) {

std::cout << "No document!\n";

return;

}

if (index >= doc\_->Size()) {

std::cout << "Out of bounds\n";

return;

}

std::shared\_ptr<figure> tmp = doc\_->GetFigure(index);

std::shared\_ptr<Acommand> command = std::shared\_ptr<Acommand>(new DeleteCommand(tmp,index,doc\_));

doc\_->Erase(index);

history\_.push(command);

}

void editor::SaveDocument() {

if (doc\_ == nullptr) {

std::cout << "No document!\nNot ";

return;

}

std::string saveName = doc\_->GetName();

doc\_ ->Save(saveName);

}

void editor::LoadDocument(std::string &name) {

try {

doc\_ = std::make\_shared<document>(name);

doc\_->Load(name);

while (!history\_.empty()){

history\_.pop();

}

} catch(std::logic\_error& e) {

std::cout << e.what();

}

}

void editor::Undo() {

if (history\_.empty()) {

throw std::logic\_error("History is empty\n");

}

std::shared\_ptr<Acommand> lastCommand = history\_.top();

lastCommand->UnExecute();

history\_.pop();

}

#endif //OOP7\_EDITOR\_H

factory.h

#ifndef OOP\_FACTORY\_H

#define OOP\_FACTORY\_H

#include <memory>

#include <iostream>

#include <fstream>

#include "square.h"

#include "rectangle.h"

#include "trapez.h"

#include <string>

struct factory {

std::shared\_ptr<figure> FigureCreate(std::istream &is) {

std::string name;

is >> name;

if ( name == "rectangle" ) {

return std::shared\_ptr<figure> ( new Rectangle(is));

} else if ( name == "trapez") {

return std::shared\_ptr<figure> ( new Trapez(is));

} else if ( name == "square") {

return std::shared\_ptr<figure> ( new Square(is));

} else {

throw std::logic\_error("There is no such figure\n");

}

}

std::shared\_ptr<figure> FigureCreateFile(std::ifstream &is) {

std::string name;

is >> name;

if ( name == "rectangle" ) {

return std::shared\_ptr<figure> ( new Rectangle(is));

} else if ( name == "trapez") {

return std::shared\_ptr<figure> ( new Trapez(is));

} else if ( name == "square") {

return std::shared\_ptr<figure> ( new Square(is));

} else {

throw std::logic\_error("There is no such figure\n");

}

}

};

#endif //OOP\_FACTORY\_H

figure.h

#ifndef OOP\_FIGURE\_H

#define OOP\_FIGURE\_H

#include <iostream>

#include "point.h"

#include <fstream>

struct figure {

virtual point center() const = 0;

virtual void print(std::ostream&) const = 0 ;

virtual void printFile(std::ofstream&) const = 0 ;

virtual double area() const = 0;

virtual ~figure() = default;

};

#endif //OOP\_FIGURE\_H

square.h

#ifndef OOP\_SQUARE\_H

#define OOP\_SQUARE\_H

#include <cmath>

#include "point.h"

#include "figure.h"

struct Square : figure {

public:

point a1, a2, a3, a4;

point center() const {

double x, y;

x = (a1.x + a2.x + a3.x + a4.x) / 4;

y = (a1.y + a2.y + a3.y + a4.y) / 4;

point p(x, y);

return p;

}

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

os << "square "<< a1 <<" "<< a2 <<" "<< a3 <<" "<< a4 <<"\n";

}

void printFile(std::ofstream &of) const {

of << "square "<< a1 <<" "<< a2 <<" "<< a3 <<" "<< a4 <<"\n";

}

double area() const {

return (-0.5) \* ((a1.x \* a2.y + a2.x \* a3.y + a3.x \* a4.y + a4.x \* a4.y) -

(a1.y \* a2.x + a2.y \* a3.x + a3.y \* a4.x + a4.y \* a1.x));

}

Square(std::istream &is) {

is >> a1 >> a2 >> a3 >> a4;

}

Square(std::ifstream &is) {

is >> a1 >> a2 >> a3 >> a4;

}

};

#endif //OOP\_SQUARE\_H

rectangle.h

#ifndef OOP\_RECTANGLE\_H

#define OOP\_RECTANGLE\_H

#include <cmath>

#include "point.h"

#include "figure.h"

struct Rectangle : figure {

point a1, a2, a3, a4;

point center() const {

double x, y;

x = (a1.x + a2.x + a3.x + a4.x) / 4;

y = (a1.y + a2.y + a3.y + a4.y) / 4;

point p(x, y);

return p;

}

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

os << "rectangle "<< a1 <<" "<< a2 <<" "<< a3 <<" "<< a4 <<"\n";

}

void printFile(std::ofstream &of) const {

of << "rectangle "<< a1 <<" "<< a2 <<" "<<a3<<" "<< a4 <<"\n";

}

double area() const {

return (-0.5) \* ((a1.x \* a2.y + a2.x \* a3.y + a3.x \* a4.y + a4.x \* a1.y) -

(a1.y \* a2.x + a2.y \* a3.x + a3.y \* a4.x + a4.y \* a1.x));

}

Rectangle(std::istream &is) {

is >> a1 >> a2 >> a3 >> a4;

}

Rectangle(std::ifstream &is) {

is >> a1 >> a2 >> a3 >> a4;

}

};

#endif //OOP\_RECTANGLE\_H

trepez.h

#ifndef OOP\_TRAPEZ\_H

#define OOP\_TRAPEZ\_H

#include <cmath>

#include <iostream>

#include "point.h"

#include "figure.h"

struct Trapez : figure{

point a1,a2,a3,a4;

point center() const {

double x,y;

x = (a1.x + a2.x + a3.x + a4.x ) / 4;

y = (a1.y + a2.y + a3.y + a4.y ) / 4;

point p(x,y);

return p;

}

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

os << "trapez "<< a1 <<" "<< a2 <<" "<< a3 <<" "<< a4 <<"\n";

}

void printFile(std::ofstream &of) const {

of << "trapez "<< a1 <<" "<< a2 <<" "<< a3 <<" "<< a4 <<"\n";

}

double area() const {

return (-0.5) \* ((a1.x\*a2.y + a2.x\*a3.y + a3.x\*a4.y + a4.x\*a1.y) - ( a1.y\*a2.x + a2.y\*a3.x + a3.y\*a4.x + a4.y\*a1.x ));

}

Trapez(std::istream& is) {

is >> a1 >> a2 >> a3 >> a4 ;

}

Trapez(std::ifstream& is) {

is >> a1 >> a2 >> a3 >> a4 ;

}

};

#endif //OOP\_TRAPEZ\_H

CmakeLists.txt

cmake\_minimum\_required(VERSION 3.10.2)

project(oop\_exercise\_07)

set(CMAKE\_CXX\_STANDARD 17)

add\_executable(oop\_exercise\_07

main.cpp

point.h

trapez.h

figure.h

rectangle.h

square.h

document.h

factory.h

command.h

editor.h)

1. **Набор testcases**

test\_01.txt

create ss.txt

add square 1 1 1 1 1 1 1 1 1 1

remove 0

undo

save

print

exit

test\_02.txt

create Masha.txt

add trapez 0 0 1 2 3 2 4 0

add square 0 0 0 0 0 0 0 0

add rectangle 1 1 1 1 1 1 1 1

print

save

remove 1

remove 0

undo

undo

add square 0 0 0 1 1 1 1 0

undo

undo

print

undo

print

undo

print

undo

exit

test\_03.txt

create Power.txt

add square 0 0 0 1 1 1 1 0

print

save

remove 0

load Power.txt

print

undo

exit

masha@masha-VirtualBox:~/2kurs/oop\_exercise\_07/tmp$ ./oop\_exercise\_07 < ~/2kurs/oop\_exercise\_07/test\_01.txt

Enter name of new document

Document create

Ok

Unknown command

Unknown command

Enter index

Ok

save document

square 1 1 1 1 1 1 1 1

masha@masha-VirtualBox:~/2kurs/oop\_exercise\_07/tmp$ ./oop\_exercise\_07 < ~/2kurs/oop\_exercise\_07/test\_02.txt

Enter name of new document

Document create

Ok

Ok

Ok

trapez 0 0 1 2 3 2 4 0

square 0 0 0 0 0 0 0 0

rectangle 1 1 1 1 1 1 1 1

save document

Enter index

Ok

Enter index

Ok

Ok

trapez 0 0 1 2 3 2 4 0

square 0 0 0 0 0 0 0 0

trapez 0 0 1 2 3 2 4 0

Buffer is empty

History is empty

masha@masha-VirtualBox:~/2kurs/oop\_exercise\_07/tmp$ ./oop\_exercise\_07 < ~/2kurs/oop\_exercise\_07/test\_03.txt

Enter name of new document

Document create

Ok

square 0 0 0 1 1 1 1 0

save document

Enter index

Ok

Enter path to the file

Document loaded

square 0 0 0 1 1 1 1 0

History is empty

1. **Объяснение результатов работы программы - вывод**

В main.cpp посредством editor.h, выступающим в роли редактора осуществляются действия с документом, его создание, удаление, сохранение и тд. В command.h реализованы вставка, удаление и обратное выполнение команды, необходимые для реализации undo; в document.h — действия с документом, в factory.h реализовано создание фигур квадрат, прямоугольник и трапеция.

В ходе лабораторной работы были усовершенствованны навыки объектно-ориентированного программирования, укреплены знания о наследовании, полиморфизме, класcах.