Московский Авиационный Институт
(Национальный исследовательский Университет)

Факультет: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»

Лабораторная работа №7
по курсу «ООП»

Тема:
Проектирование структуры классов.

Студент:	Вахрамян К.О.
Группа:	М80-206Б-18
Преподаватель:	Журавлев А.А.
Вариант:	3
Оценка:	
Дата:	

Москва
2019

1.**Код на C++:**
comand.h:

```
#include "comand.h"


void InsertComand::Exec() {
	doc->Add();
}

void InsertComand::Undo() {
	doc->popBack();
}

void RemoveComand::Exec() {
	try {
		figure = doc->Get(idx);
		pos = idx;
	} catch (std::exception& e) {
		std::cout << e.what()  << std::endl;
		return;
	}
	doc->Delete(idx);
}

void RemoveComand::Undo() {
	doc->Add(figure, pos);
}
```

command.cpp:

```
#include "comand.h"


void InsertComand::Exec() {
	doc->Add();
}

void InsertComand::Undo() {
	doc->popBack();
}

void RemoveComand::Exec() {
	try {
		figure = doc->Get(idx);
		pos = idx;
	} catch (std::exception& e) {
		std::cout << e.what()  << std::endl;
		return;
	}
	doc->Delete(idx);
```

```
}

void RemoveComand::Undo() {
	doc->Add(figure, pos);
}
```

figure.h:

```
#ifndef FIGURE_H
#define FIGURE_H

#include <iostream>
#include <cassert>
#include <stdexcept>
#include "point.h"
#include <cmath>
#include <exception>
#include <string>


enum class Figures {
	Rectangle,
	Rhombus,
	Trapezodid
};

class TFigure {
public:
	virtual void Print(std::ostream&) const = 0;
	virtual TPoint Center() const = 0;
	virtual double Square() const = 0;

};
#endif
```

rectangle.h

```
#ifndef RECTANGLE_H
#define RECTANGLE_H

#include "figure.h"

class TRectangle : public TFigure {
private:
	TPoint a, b, c, d;
public:
	void Print(std::ostream& os) const override;
	TPoint Center() const override;
	double Square() const override;
	TRectangle();
	TRectangle(TPoint p1, TPoint p2, TPoint p3, TPoint p4);
	TRectangle(std::istream& is);
};


#endif // RECTANGLE_H
```

```
#include "rectangle.h"

TRectangle::TRectangle (const TPoint p1, const TPoint p2, const TPoint p3, const TPoint p4) {
	a = p1;
	b = p2;
	c = p3;
	d = p4;
	TPoint ab, ad, cb, cd;
	ab.x = b.x - a.x;
	ab.y = b.y - a.y;
	ad.x = d.x - a.x;
	ad.y = d.y - a.y;
	cb.x = b.x - c.x;
	cb.y = b.y - c.y;
	cd.x = d.x - c.x;
	cd.y = d.y - c.y;
	if (acos((ab.x * ad.x + ab.y * ad.y) / (sqrt(ab.x * ab.x + ab.y * ab.y) * sqrt(ad.x * ad.x +
ad.y * ad.y))) / M_PI != 0.5 || acos((cb.x * cd.x + cb.y * cd.y) / (sqrt(cb.x * cb.x + cb.y * cb.y) *
sqrt(cd.x * cd.x + cd.y * cd.y))) / M_PI != 0.5) {
		throw std::logic_error("it's not rectangle\n");
	}
	//assert(acos((ab.x * ad.x + ab.y * ad.y) / (sqrt(ab.x * ab.x + ab.y * ab.y) * sqrt(ad.x *
ad.x + ad.y * ad.y))) / M_PI == 0.5 && acos((cb.x * cd.x + cb.y * cd.y) / (sqrt(cb.x * cb.x + cb.y
* cb.y) * sqrt(cd.x * cd.x + cd.y * cd.y))) / M_PI == 0.5);
}

TRectangle::TRectangle(std::istream& is) {
	is >> a >> b >> c >> d;
	TPoint ab, ad, cb, cd;
	ab.x = b.x - a.x;
	ab.y = b.y - a.y;
	ad.x = d.x - a.x;
	ad.y = d.y - a.y;
	cb.x = b.x - c.x;
	cb.y = b.y - c.y;
	cd.x = d.x - c.x;
	cd.y = d.y - c.y;
	if (acos((ab.x * ad.x + ab.y * ad.y) / (sqrt(ab.x * ab.x + ab.y * ab.y) * sqrt(ad.x * ad.x +
ad.y * ad.y))) / M_PI != 0.5 || acos((cb.x * cd.x + cb.y * cd.y) / (sqrt(cb.x * cb.x + cb.y * cb.y) *
sqrt(cd.x * cd.x + cd.y * cd.y))) / M_PI != 0.5) {
		throw std::logic_error("it's not rectangle\n");
	}

}

double TRectangle::Square () const {

	double ans = (b.x - a.x) * (c.y - a.y) - (c.x - a.x) * (b.y - a.y);
	return fabs(ans);
}

TPoint TRectangle::Center() const {
```

```
	TPoint p;
	double x = (a.x + b.x + c.x + d.x) / 4;
	double y = (a.y + b.y + c.y + d.y) / 4;
	p.x = x;
	p.y = y;
	return p;
}

void TRectangle::Print(std::ostream& os) const {
	os << "rectangle ";
	os <<  a << " " << b << " " << c << " " << d << "\n";
}
```

rhombus.h:

```
#ifndef RHOMBUS_H
#define RHOMBUS_H

#include "figure.h"
#include <iostream>

class TRhombus : public TFigure{
private:
	TPoint a, b, c, d;

public:
	double Square() const override;
	TPoint Center() const override;
	void Print(std::ostream&) const override;
	TRhombus();
	TRhombus(const TPoint p1, const TPoint p2, const TPoint p3, const TPoint p4);
	TRhombus(std::istream& is);
};


#endif
```

rhombus.cpp:

```
#include "rhombus.h"

TRhombus::TRhombus (const TPoint p1, const TPoint p2, const TPoint p3, const TPoint p4) {
	a = p1;
	b = p2;
	c = p3;
	d = p4;
	TPoint ab, bc, cd, da;
```

```
	ab.x = b.x - a.x;
	ab.y = b.y - a.y;
	bc.x = c.x - b.x;
	bc.y = c.y - b.y;
	cd.x = d.x - c.x;
	cd.y = d.y - c.y;
	da.x = a.x - d.x;
	da.y = a.y - d.y;
	if (sqrt(ab.x * ab.x + ab.y * ab.y) !=  sqrt(bc.x * bc.x + bc.y * bc.y) || sqrt(bc.x * bc.x + bc.y * bc.y) != sqrt(cd.x * cd.x + cd.y * cd.y) || sqrt(cd.x * cd.x + cd.y * cd.y) != sqrt(da.x * da.x + da.y * da.y)) {
		throw std::logic_error("it's not rhombus\n");
	}
	//assert(sqrt(ab.x * ab.x + ab.y * ab.y) ==  sqrt(bc.x * bc.x + bc.y * bc.y) && sqrt(bc.x * bc.x + bc.y * bc.y) == sqrt(cd.x * cd.x + cd.y * cd.y) && sqrt(cd.x * cd.x + cd.y * cd.y) == sqrt(da.x * da.x + da.y * da.y));

}

TRhombus::TRhombus(std::istream& is) {
	is >> a >> b >> c >> d;
	TPoint ab, bc, cd, da;
	ab.x = b.x - a.x;
	ab.y = b.y - a.y;
	bc.x = c.x - b.x;
	bc.y = c.y - b.y;
	cd.x = d.x - c.x;
	cd.y = d.y - c.y;
	da.x = a.x - d.x;
	da.y = a.y - d.y;
	if (sqrt(ab.x * ab.x + ab.y * ab.y) !=  sqrt(bc.x * bc.x + bc.y * bc.y) || sqrt(bc.x * bc.x + bc.y * bc.y) != sqrt(cd.x * cd.x + cd.y * cd.y) || sqrt(cd.x * cd.x + cd.y * cd.y) != sqrt(da.x * da.x + da.y * da.y)) {
		throw std::logic_error("it's not rhombus\n");
	}

}

double TRhombus::Square() const {
	double ans = 0.5 * sqrt(pow(a.x - c.x, 2) + pow(a.y - c.y, 2)) * sqrt(pow(b.x - d.x, 2) + pow(b.y - d.y, 2));
	return fabs(ans);
}

TPoint TRhombus::Center() const {
	TPoint p;
	double x = (a.x + b.x + c.x + d.x) / 4;
	double y = (a.y + b.y + c.y + d.y) / 4;
	p.x = x;
	p.y = y;
	return p;
}
```

```
void TRhombus::Print(std::ostream& os) const {
	os << "rhombus ";
	os << a << " " << b << " " << c << " " << d << "\n";
}
```

trapezoid.h:

```
#ifndef TRAPEZOID_H
#define TRAPEZOID_H

#include "figure.h"
#include <cmath>

class TTrapezoid : public TFigure{
private:
	TPoint a, b, c, d;
public:
	double Square() const override;
	TPoint Center() const override;
	void Print(std::ostream&) const override;
	TTrapezoid();
	TTrapezoid(const TPoint p1, const TPoint p2, const TPoint p3, const TPoint p4);
	TTrapezoid(std::istream& is);
};
#endif
```

trapezoid.cpp:

```
#include "trapezoid.h"

TTrapezoid::TTrapezoid (const TPoint p1, const TPoint p2, const TPoint p3, const TPoint p4) {
	a = p1;
	b = p2;
	c = p3;
	d = p4;
	TPoint ab, ad, bc, dc;
	ab.x = b.x - a.x;
	ab.y = b.y - a.y;
	ad.x = d.x - a.x;
	ad.y = d.y - a.y;
	bc.x = c.x - b.x;
	bc.y = c.y - b.y;
	dc.x = c.x - d.x;
	dc.y = c.y - d.y;
	if (acos((ab.x * dc.x + ab.y * dc.y) / (sqrt(ab.x * ab.x + ab.y * ab.y) * sqrt(dc.x * dc.x +
dc.y * dc.y))) != 0 && acos((ad.x * bc.x + ad.y * bc.y) / (sqrt(ad.x * ad.x + ad.y * ad.y) *
sqrt(bc.x * bc.x + bc.y * bc.y))) != 0) {
		throw std::logic_error("it's not trapezoid\n");
	}
```

```
	//assert(acos((ab.x * dc.x + ab.y * dc.y) / (sqrt(ab.x * ab.x + ab.y * ab.y) * sqrt(dc.x * dc.x + dc.y * dc.y))) == 0 || acos((ad.x * bc.x + ad.y * bc.y) / (sqrt(ad.x * ad.x + ad.y * ad.y) * sqrt(bc.x * bc.x + bc.y * bc.y))) == 0);

}

TTrapezoid::TTrapezoid(std::istream& is) {
	is >> a >> b >> c >> d;
	TPoint ab, ad, bc, dc;
	ab.x = b.x - a.x;
	ab.y = b.y - a.y;
	ad.x = d.x - a.x;
	ad.y = d.y - a.y;
	bc.x = c.x - b.x;
	bc.y = c.y - b.y;
	dc.x = c.x - d.x;
	dc.y = c.y - d.y;
	if (acos((ab.x * dc.x + ab.y * dc.y) / (sqrt(ab.x * ab.x + ab.y * ab.y) * sqrt(dc.x * dc.x + dc.y * dc.y))) != 0 && acos((ad.x * bc.x + ad.y * bc.y) / (sqrt(ad.x * ad.x + ad.y * ad.y) * sqrt(bc.x * bc.x + bc.y * bc.y))) != 0) {
		throw std::logic_error("it's not trapezoid\n");
	}

}

TPoint TTrapezoid::Center() const {
	TPoint p;
	double x = (a.x + b.x + c.x + d.x) /4;
	double y = (a.y + b.y + c.y + d.y) /4;
	p.x = x;
	p.y = y;

	return p;

}
double TTrapezoid::Square() const {
	TPoint p = this->Center();
	double t1 = 0.5 * fabs((b.x - a.x) * (p.y - a.y) - (p.x - a.x) * (b.y - a.y));
	double t2 = 0.5 * fabs((c.x - b.x) * (p.y - b.y) - (p.x - b.x) * (c.y - b.y));
	double t3 = 0.5 * fabs((d.x - c.x) * (p.y - c.y) - (p.x - c.x) * (d.y - c.y));
	double t4 = 0.5 * fabs((a.x - d.x) * (p.y - d.y) - (p.x - d.x) * (a.y - d.y));
	return t1 + t2 + t3 + t4;
}

void TTrapezoid::Print(std::ostream& os) const {
	os << "trapezoid ";
	os << a << " " << b << " " << c << " " << d << "\n";
}
```

document.h:

```
#pragma once

#include "figure.h"
#include "factory.h"
#include <memory>
#include <vector>
#include <string>



class TDocument {
        friend class TComand;
public:
        void New();
        void Save(std::ostream& os);
        void Load(std::istream& is);
        void Print();
        void Add();
        void Add(std::shared_ptr<TFigure> f, int idx);
        void Delete(int idx);
        std::shared_ptr<TFigure> Get(int idx);
        void popBack();

private:
        std::vector<std::shared_ptr<TFigure>> figures;
        Factory factory;
        void Serialize(const std::string& file);
        void Deserialize(const std::string& file);
};
```

document.cpp:

```
#include "document.h"
#include <string>

void TDocument::New() {
        figures.clear();
}

void TDocument::Save(std::ostream& os) {
        int i = 0;
        for (auto &tmp : figures) {
                os << i << " ";
                i++;
                tmp->Print(os);
        }
}

void TDocument::Add(std::shared_ptr<TFigure> f, int idx) {
        figures.insert(figures.begin() + idx, std::move(f));
}

void TDocument::Delete(int idx) {
        figures.erase(figures.begin() + idx);
}
```

```
void TDocument::Print() {
	for (auto &tmp : figures) {

		tmp->Print(std::cout);
		std::cout << "Square: " << tmp->Square() << std::endl;
		std::cout << "Center: " << tmp->Center() << std::endl;
	}
}

void TDocument::Load(std::istream& is) {
	this->New();
	int num;
			std::string n;
	while (is >> num) {

		std::shared_ptr<TFigure> figure = this->factory.FigureCreate(is);
		if (figure) {
			figures.push_back(figure);
		}
	}
}

void TDocument::popBack() {
	if (!figures.size()) {
		throw std::logic_error("Doc is empty\n");
	}
	figures.pop_back();
}

std::shared_ptr<TFigure> TDocument::Get(int idx) {
	int i = 0;
	for (const auto& figure : figures) {
		if (i == idx)  {
			return figure;
		}
		i++;
	}
	throw std::invalid_argument("No figure with such Id\n");
}

void TDocument::Add() {
	std::shared_ptr<TFigure> figure = this->factory.FigureCreate(std::cin);
	if (figure) {
		figures.push_back(figure);

	}
}
```

editor.h:

```
#pragma once

#include <stack>
#include "comand.h"
#include "document.h"
#include "figure.h"


class TEditor {
public:
	TEditor() : doc(nullptr) {};
	void CreateDoc();
	void Add();
	void Add(int idx);
	void Remove(int idx);
	void SaveDoc(std::ostream& os);
	void LoadDoc(std::istream& is);
	void Undo();
	void Print();
private:
	std::shared_ptr<TDocument> doc;
	std::stack<std::shared_ptr<TComand>> cmds;
};
```

editor.cpp:

```
#include "editor.h"
void TEditor::CreateDoc() {
	doc = std::make_shared<TDocument>();
	while(!cmds.empty()) {
		cmds.pop();
	}
}

void TEditor::Add() {
	std::shared_ptr<TComand> c = std::shared_ptr<TComand>(new InsertComand(doc));
	c->Exec();
	cmds.push(c);
}

void TEditor::Remove(int idx) {
	try {
		std::shared_ptr<TComand> c = std::shared_ptr<TComand>(new
RemoveComand(doc, idx));
		c->Exec();
		cmds.push(c);
	} catch (std::exception& e) {
		std::cout << e.what() << std::endl;
	}
}
```

```
void TEditor::SaveDoc(std::ostream& os) {
	doc->Save(os);
}

void TEditor::LoadDoc(std::istream& is) {
	CreateDoc();
	doc->Load(is);
}

void TEditor::Undo() {
	if (cmds.empty()) {
		throw std::logic_error("Stack comands is empty\n");
	}
	std::shared_ptr<TComand> c = cmds.top();
	c->Undo();
	cmds.pop();
}

void TEditor::Print() {
	doc->Print();
}
```

2. **Ссылка на репозиторий в GitHub:**
https://github.com/vebcreatex7/oop_exercise_07

3.**Результаты выполнения программы:**

```
new - to create new document
insert- to insert figure to document
delete idx- to delete figure on position idx from document
undo - to cancel last comand
save - to save document in file
load - to load document from file
print - to print all figure
help - to show this page
exit - to finish execution of program
new
insert rectangle 0 0 0 1 1 1 1 0
insert trapezoid 0 0 0 2 2 2 2 0
insert rhombus 0 0 0 3 3 3 3 0
print
rectangle 0 0 0 1 1 1 1 0
Square: 1
Center: 0.5 0.5
trapezoid 0 0 0 2 2 2 2 0
Square: 4
Center: 1 1
rhombus 0 0 0 3 3 3 3 0
```

Square: 9
Center: 1.5 1.5
save file.txt
delete 0
print
trapezoid 0 0 0 2 2 2 2 0
Square: 4
Center: 1 1
rhombus 0 0 0 3 3 3 3 0
Square: 9
Center: 1.5 1.5
delete 2
No figure with such Id

delete 1
delete 0
print
load file.txt
print
rectangle 0 0 0 1 1 1 1 0
Square: 1
Center: 0.5 0.5
trapezoid 0 0 0 2 2 2 2 0
Square: 4
Center: 1 1
rhombus 0 0 0 3 3 3 3 0
Square: 9
Center: 1.5 1.5
undo
Stack comands is empty

delete 1
print
rectangle 0 0 0 1 1 1 1 0
Square: 1
Center: 0.5 0.5
rhombus 0 0 0 3 3 3 3 0
Square: 9
Center: 1.5 1.5
undo
print
rectangle 0 0 0 1 1 1 1 0
Square: 1
Center: 0.5 0.5
trapezoid 0 0 0 2 2 2 2 0
Square: 4
Center: 1 1
rhombus 0 0 0 3 3 3 3 0
Square: 9
Center: 1.5 1.5
delete 0
delete 0

delete 0
print
undo
unod
Wrong comand
undo
undo
undo
Stack comands is empty

print
rectangle 0 0 0 1 1 1 1 0
Square: 1
Center: 0.5 0.5
trapezoid 0 0 0 2 2 2 2 0
Square: 4
Center: 1 1
rhombus 0 0 0 3 3 3 3 0
Square: 9
Center: 1.5 1.5
exit

4. Объяснение результатов работы программы:
Пользователь вводит команды с терминала. В программе реализованы функции создания нового документа, чтение и запись в файл. Команда undo отменяет последнее добавление или удаление фигуры.

5.**Вывод:**
В данной лабораторной работе реализован примитивный текстовый редактор. Проведена работа с чтением и записью в файл. Также осознано и реализовано то, как работает отмена последнего действия при помощи команды undo.