**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 7**

Тема: Проектирование структуры классов

Студент: Киреев Александр Константинович

Группа: 80-206

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2020

1. **Постановка задачи**

Вариант 8. (8-угольник, треугольник, квадрат)

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 в std::cout);

• 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;

1. **Описание программы**

В программе реализован родительский класс Figure от которого наследуются классы Triangle, Square и Octagon. Родительский класс имеет 3 чисто виртуальных метода: getCenter(), print(), getArea(). В родительский класс встроен класс Dot, для удобства работы с точками. Также класс Figure имеет виртуальный деструктор, поле \_type и метод для получения типа фигуры для удобства работы с фигурами. Дочерние классы имеют перегруженные методы родительского класса, а также конструктор, который принимает поток ввода и конструирует объект из полученных из потока данных. Метод getArea() получает площадь фигуры по радиусу описанной окружности. Метод print() нуждается в координатах вершин, которые вычисляются в методе getCoords(). Зная координаты центра фигуры, координаты ее вершины, мы можем найти полярный угол и угол поворота, на который нужно отступить от заданной вершины, чтобы получить следующую вершину. В файле main.cpp мы имеем парсер команд для работы пользователя с программой, а также функции applyToVec(const std::vector<Figure\*>& vec), которая вычисляет всю информацию о каждой фигуре в векторе, и totalArea(const std::vector<Figure\*>& vec), которая считает общую площадь фигур в векторе. Интерактивный интерфейс работы с пользователем находится в файле main.cpp. В файле editor.cpp реализован сам редактор, который работает с родительским типом Figure, что позволяет обобщить работу с данным редактором, не заботясь о том, сколько типов фигур у нас есть. Также данный подход позволяет сделать программу гибкой для последующего расширения. В самом редакторе я добавил возможность создавать несколько файлов и переключаться между ними.

1. **Набор тестов**

Тест 1:

1

4 2

0 0 1 1

4 3

0 0 0 1

6

7

6

4 3

0 0 0 1

5

6

7

6

10

Тест 1 проверяет работоспособность программы в целом. В данном тесте мы добавляем пару фигур в документ, выводим их на экран, проверяем работу undo, также проверяем работу операции удаления фигуры.

Тест 2:

1

4 2

0 0 1 1

4 3

0 0 0 1

6

3 testfile.bin

1

6

2 testfile.bin

6

10

В тесте 2 мы проверяем работу сохранения документа в файл, загрузки сохранения документа из файла.

Тест 3:

7

5

sddvfb

1

5

8 10

10

В тесте 3 мы проверяем обработку невалидных данных, выход за границы при переключении документов, проверяем, что удаление корректно работает на пустом документе.

1. **Результаты выполнения тестов**

Тест 1:

MacBook-Air-K:Build AK$ ./lr7 < ../../test/test01.t

Command -- Description

1 -- Create new document

2 <filename> -- Load document from file

3 <filename> -- Save document from file

4 <type> -- Create new figure (1 -- Octagon, 2 -- Square, 3 -- Triangle)

5 -- Delete last figure from vector

6 -- Display document

7 -- Undo last delete/create

8 <num> -- Switch document to page <num>

9 -- Display help

10 -- End program

=========================DOCUMENT===========================

>>Next figure is Square:

Center coords is: (0.000, 0.000)

[ (1.000, 1.000), (-1.000, 1.000), (-1.000, -1.000), (1.000, -1.000) ]

>>Next figure is Triangle:

Center coords is: (0.000, 0.000)

[ (0.000, 1.000), (-0.866, -0.500), (0.866, -0.500) ]

======================END=OF=DOCUMENT=======================

=========================DOCUMENT===========================

>>Next figure is Square:

Center coords is: (0.000, 0.000)

[ (1.000, 1.000), (-1.000, 1.000), (-1.000, -1.000), (1.000, -1.000) ]

======================END=OF=DOCUMENT=======================

=========================DOCUMENT===========================

>>Next figure is Square:

Center coords is: (0.000, 0.000)

[ (1.000, 1.000), (-1.000, 1.000), (-1.000, -1.000), (1.000, -1.000) ]

======================END=OF=DOCUMENT=======================

=========================DOCUMENT===========================

>>Next figure is Square:

Center coords is: (0.000, 0.000)

[ (1.000, 1.000), (-1.000, 1.000), (-1.000, -1.000), (1.000, -1.000) ]

>>Next figure is Triangle:

Center coords is: (0.000, 0.000)

[ (0.000, 1.000), (-0.866, -0.500), (0.866, -0.500) ]

======================END=OF=DOCUMENT=======================

Тест 2:

Cmd: MacBook-Air-K:Build AK$ ./lr7 < ../../test/test02.t

Command -- Description

1 -- Create new document

2 <filename> -- Load document from file

3 <filename> -- Save document from file

4 <type> -- Create new figure (1 -- Octagon, 2 -- Square, 3 -- Triangle)

5 -- Delete last figure from vector

6 -- Display document

7 -- Undo last delete/create

8 <num> -- Switch document to page <num>

9 -- Display help

10 -- End program

=========================DOCUMENT===========================

>>Next figure is Square:

Center coords is: (0.000, 0.000)

[ (1.000, 1.000), (-1.000, 1.000), (-1.000, -1.000), (1.000, -1.000) ]

>>Next figure is Triangle:

Center coords is: (0.000, 0.000)

[ (0.000, 1.000), (-0.866, -0.500), (0.866, -0.500) ]

======================END=OF=DOCUMENT=======================

=========================DOCUMENT===========================

======================END=OF=DOCUMENT=======================

=========================DOCUMENT===========================

>>Next figure is Square:

Center coords is: (0.000, 0.000)

[ (1.000, 1.000), (-1.000, 1.000), (-1.000, -1.000), (1.000, -1.000) ]

>>Next figure is Triangle:

Center coords is: (0.000, 0.000)

[ (0.000, 1.000), (-0.866, -0.500), (0.866, -0.500) ]

======================END=OF=DOCUMENT=======================

Тест 3:

Cmd: MacBook-Air-K:Build AK$ ./lr7 < ../../test/test03.t

Command -- Description

1 -- Create new document

2 <filename> -- Load document from file

3 <filename> -- Save document from file

4 <type> -- Create new figure (1 -- Octagon, 2 -- Square, 3 -- Triangle)

5 -- Delete last figure from vector

6 -- Display document

7 -- Undo last delete/create

8 <num> -- Switch document to page <num>

9 -- Display help

10 -- End program

ERROR: There are no any documents in editor

ERROR: There are no any documents in editor

Invalid command.

ERROR: There are no any figures in document

ERROR: Page number is invalid

1. **Листинг программы**

**main.cpp**

// Киреев Александр Константинович 206

#include <iostream>

#include <vector>

#include <sstream>

#include <string>

#include "editor.hpp"

// мануал

void help() {

std::cout << "Command -- Description\n" <<

"1 -- Create new document\n" <<

"2 <filename> -- Load document from file\n" <<

"3 <filename> -- Save document from file\n" <<

"4 <type> -- Create new figure (1 -- Octagon, 2 -- Square, 3 -- Triangle)\n" <<

"5 -- Delete last figure from vector\n" <<

"6 -- Display document\n" <<

"7 -- Undo last delete/create\n" <<

"8 <num> -- Switch document to page <num>\n" <<

"9 -- Display help\n" <<

"10 -- End program" << std::endl;

}

void mainLoop() {

int command = 0, idx = 0, type = 0;

Editor ed;

std::string s;

std::string filename;

std::ifstream ifs;

std::ofstream ofs;

while (std::cout << "Cmd: " && std::cin >> s) {

if (s.length() > 2) {

std::cout << "Invalid command." << std::endl;

continue;

}

std::stringstream ss(s);

ss >> command;

if (command == 10)

break;

switch (command) {

case 1:

ed.createNewDoc();

break;

case 2:

std::cin >> filename;

ifs.open(filename, std::ios::binary);

if (ifs.fail()) {

std::cerr << "ERROR: Unable to open file " << filename << " in read mode\n";

break;

}

try {

ed.loadDoc(ifs);

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

std::cerr << "ERROR: " << e.what() << "\n";

continue;

}

ifs.close();

break;

case 3:

std::cin >> filename;

ofs.open(filename, std::ios::binary);

if (ofs.fail()) {

std::cerr << "ERROR: Unable to open file " << filename << " in write mode\n";

break;

}

try {

ed.saveDoc(ofs);

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

std::cerr << "ERROR: " << e.what() << "\n";

continue;

}

ofs.close();

break;

case 4:

std::cin >> type;

switch (type) {

case 1:

try {

ed.createPrimitive(std::cin, 'O');

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

std::cerr << "ERROR: " << e.what() << "\n";

continue;

}

break;

case 2:

try {

ed.createPrimitive(std::cin, 'S');

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

std::cerr << "ERROR: " << e.what() << "\n";

continue;

}

break;

case 3:

try {

ed.createPrimitive(std::cin, 'T');

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

std::cerr << "ERROR: " << e.what() << "\n";

continue;

}

break;

default:

std::cout << "Invalid type." << std::endl;

break;

}

break;

case 5:

try {

ed.deletePrimitive();

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

std::cerr << "ERROR: " << e.what() << "\n";

continue;

}

break;

case 6:

try {

ed.print();

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

std::cerr << "ERROR: " << e.what() << "\n";

continue;

}

break;

case 7:

try {

if (!ed.undo()) {

std::cout << "Unable to do undo.\n";

}

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

std::cerr << "ERROR: " << e.what() << "\n";

continue;

}

break;

case 8:

std::cin >> idx;

try {

ed.switchDoc(idx);

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

std::cerr << "ERROR: " << e.what() << "\n";

continue;

}

break;

case 9:

help();

break;

default:

std::cout << "Invalid command!" << std::endl;

break;

}

}

}

int main() {

help();

mainLoop();

return 0;

}

**editor.cpp**

// М8О-206Б-19 Киреев А.К.

#include "editor.hpp"

Editor::~Editor() {

for (auto s: saved) {

if (s.type == 'd') {

delete s.f;

}

}

for (auto d : docs) {

for (auto f: d) {

delete f;

}

}

}

void Editor::createNewDoc() {

docs.push\_back(std::vector<Figure\*>());

saved.push\_back({});

page++;

}

void Editor::loadDoc(std::ifstream& fs) {

Dot center;

Dot node;

char type;

std::vector<Figure\*> newDoc;

while (true) {

fs.read((char\*)&type, sizeof(char));

if (type == 'E') {

break;

}

if (fs.bad()) {

throw std::logic\_error("Unable to read from file");

return;

}

fs.read((char\*)&center, sizeof(Dot));

if (fs.bad()) {

throw std::logic\_error("Unable to read from file");

return;

}

fs.read((char\*)&node, sizeof(Dot));

if (fs.bad()) {

throw std::logic\_error("Unable to read from file");

return;

}

Figure\* f;

if (type == 'S') {

f = new Square(center, node);

} else if (type == 'O') {

f = new Octagon(center, node);

} else {

f = new Triangle(center, node);

}

newDoc.push\_back(f);

}

createNewDoc();

docs[page] = newDoc;

}

void Editor::saveDoc(std::ofstream& fs) {

for (auto f: docs[page]) {

std::vector<Dot> dots;

f->getCoords(dots);

Dot center = f->getCenter();

std::string type = f->getType();

char charType;

if (type == "Triangle") {

charType = 'T';

} else if (type == "Square") {

charType = 'S';

} else {

charType = 'O';

}

fs.write((char\*)&charType, sizeof(char));

if (fs.bad()) {

throw std::logic\_error("Unable to write in file");

return;

}

fs.write((char\*)&center, sizeof(Dot));

if (fs.bad()) {

throw std::logic\_error("Unable to write in file");

return;

}

fs.write((char\*)&dots[0], sizeof(Dot));

if (fs.bad()) {

throw std::logic\_error("Unable to write in file");

return;

}

}

char endOfFile = 'E';

fs.write((char\*)&endOfFile, sizeof(char));

if (fs.bad()) {

throw std::logic\_error("Unable to write in file");

return;

}

}

void Editor::createPrimitive(std::istream& is, char type) {

if (docs.empty()) {

throw std::logic\_error("There are no any documents in editor");

return;

}

Figure\* f;

if (type == 'T') {

f = new Triangle(is);

} else if (type == 'O') {

f = new Octagon(is);

} else {

f = new Square(is);

}

docs[page].push\_back(f);

if (saved[page].type == 'd') {

delete saved[page].f;

}

saved[page].type = 'c';

saved[page].f = f;

}

void Editor::deletePrimitive() {

if (docs.empty()) {

throw std::logic\_error("There are no any documents in editor");

return;

}

int lastIdx = docs[page].size() - 1;

if (lastIdx < 0) {

throw std::logic\_error("There are no any figures in document");

return;

}

Figure\* f = docs[page][lastIdx];

docs[page].pop\_back();

if (saved[page].type == 'd') {

delete saved[page].f;

}

saved[page].type = 'd';

saved[page].f = f;

}

void Editor::print() {

if (docs.empty()) {

throw std::logic\_error("There are no any documents in editor");

return;

}

std::cout << "=========================DOCUMENT===========================\n";

for (auto f: docs[page]) {

std::cout << ">>Next figure is " << f->getType() << ": " << std::endl;

std::cout << "Center coords is: " << f->getCenter() << "\n";

f->print(std::cout);

std::cout << "\n";

}

std::cout << "======================END=OF=DOCUMENT=======================\n";

}

bool Editor::undo() {

if (docs.empty()) {

throw std::logic\_error("There are no any documents in editor");

return false;

}

if (saved[page].type == 'u') {

return false;

}

if (saved[page].type == 'c') {

docs[page].pop\_back();

delete saved[page].f;

saved[page].f = nullptr;

saved[page].type = 'u';

} if (saved[page].type == 'd') {

docs[page].push\_back(saved[page].f);

saved[page].f = nullptr;

saved[page].type = 'u';

}

return true;

}

void Editor::switchDoc(int newPage) {

if (newPage < 0 || newPage >= docs.size()) {

throw std::logic\_error("Page number is invalid");

return;

}

page = newPage;

}

1. **Вывод**

Фабричный метод — это порождающий паттерн проектирования, который решает проблему создания различных продуктов, без указания конкретных классов продуктов. Данный паттерн позволяет сделать код более гибким, модульным. Также данный подход позволяет абстрагироваться от конкретных реализаций и особенностей объектов.

**Список литературы**

1. “Язык программирования C++. Краткий курс”, Бьярне Страуструп, 2-е издание, (перевод Игоря Красикова), бумага офсетная-белая, твердый переплет, 320 стр., ISBN 978-5-907144-12-5.