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-206Б-18, №14 |
| Студент: | Орозбакиев Э.Д. |
| Преподаватель: | Журавлёв Андрей Андреевич |
| Оценка: |  |
| Дата: |  |

Москва, 2019

1. **Задание**

Разработать программу на языке C++ согласно варианту задания. Программа на C++ должна собираться с помощью системы сборки CMake. Программа должна получать данные из стандартного ввода и выводить данные в стандартный вывод.

Необходимо настроить сборку лабораторной работы с помощью CMake. Собранная программа должна называться oop\_exercise\_07 (в случае использования Windows oop\_exercise\_07.exe)

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;

1. **Адрес репозитория на GitHub**

https://github.com/p0kemo4ik/oop\_exercise\_07

1. **Код программы на С++**

#include **<iostream>**#include **"factory.hpp"**#include **"editor.hpp"****void** help() { std::cout << **"help - выводит это меню\n"** **"create <path> - создает новый файл\n"** **"save - сохраняет данные в файл\n"** **"load <path> - выгружает данные из файла\n"** **"add <square, rectangle or trapezoid> <vertices> - добавить фигуру\n"** **"remove <index> - удаляет фигуру по индексу\n"** **"print - выводит все фигуру\n"** **"undo - отменяет последнее действие\n"** **"exit\n"**;}**void** create(editor& edit) { std::string tmp; std::cin >> tmp; edit.CreateDocument(tmp); std::cout << **"OK\n"**;}**void** load(editor& edit) { std::string tmp; std::cin >> tmp; **try** { edit.LoadDocument(tmp); std::cout << **"OK\n"**; } **catch** (std::runtime\_error& e) { std::cout << e.what(); }}**void** save(editor& edit) { std::string tmp; **try** { edit.SaveDocument(); std::cout << **"OK\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::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 == **"help"**) { help(); } **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;}

*command.hpp*

#ifndef **\_COMMAND\_H\_**#define **\_COMMAND\_H\_**#include **"document.hpp"****struct** Acommand {**public**: **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\_;};#endif *// \_COMMAND\_H\_*

*editor.cpp*

#include **<iostream>**#include **"editor.hpp"****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();}

*factory.cpp*

#include **"factory.hpp"**std::shared\_ptr<figure> factory::FigureCreate(std::istream &is) { std::string name; is >> name; **if** (name == **"pentagon"** ) { **return** std::shared\_ptr<figure> ( **new** pentagon(is)); } **else if** (name == **"hexagon"**) { **return** std::shared\_ptr<figure> ( **new** hexagon(is)); } **else if** (name == **"octagon"**) { **return** std::shared\_ptr<figure> ( **new** octagon(is)); } **else** { **throw** std::logic\_error(**"There is no such figure\n"**); }}std::shared\_ptr<figure> factory::FigureCreateFile(std::ifstream &is) { std::string name; is >> name; **if** (name == **"pentagon"** ) { **return** std::shared\_ptr<figure> (**new** pentagon(is)); } **else if** (name == **"hexagon"**) { **return** std::shared\_ptr<figure> (**new** hexagon(is)); } **else if** (name == **"octagon"**) { **return** std::shared\_ptr<figure> (**new** octagon(is)); } **else** { **throw** std::logic\_error(**"There is no such figure\n"**); }}

1. **Результаты выполнения тестов**

test\_02.txt

create figures1.txt  
print  
add pentagon 0 0 0 0 0 0 0 0 0 0  
undo  
print  
exit

*p0kemo4ik@PcPokemo4ik:~/Рабочий стол/oop/lab7/cmake-build-debug$ ./lab7 < test\_02.txt*

*OK*

*Buffer is empty*

*Ok*

*Buffer is empty*

1. **Объяснение результатов работы программы**

В программе реализованы функции сохранения фигур (пятиугольник, шестиугольник, восьмиугольник) в файл, загрузки из файла и отмены последнего добавления / удаления фигуры в файл.

1. **Вывод**

Благодаря данной лабораторной работе студент может улучшить свои навыки в проектировании более сложных программ. Умение проектировать структуру классов позволяет сделать дальнейшую разработку более гибкой и простой, повысить читаемость кода.