Московский Авиационный Институт
(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа
по курсу «Объектно-ориентированное программирование»
III Семестр

Задание 7
Вариант 24
Проектирование структуры классов

Студент:	Синявский А.В
Группа:	М80-208Б-18
Преподаватель:	Журавлёв А.А
Оценка:	
Дата:	

Москва 2019

1. Код программы на языке C++

1.1 dot.h

```
#ifndef OOP_LAB7_DOT_H
#define OOP_LAB7_DOT_H

#include <iostream>

namespace figures {

    class Dot {
    public:
        double x;
        double y;
        Dot();
        Dot(double X, double Y);
        Dot& operator=(const Dot &A);
        Dot operator+(const Dot &A);
        Dot operator-(const Dot &A);
        Dot operator/(const double &A);
        friend std::ostream &operator<<(std::ostream &os, const Dot& A);
        friend std::istream &operator>>(std::istream &is, Dot& A);
        double Length(const Dot &A);
    };

    Dot operator""_dot(const char* str, size_t size);

}

#endif //OOP_LAB7_DOT_H
```

1.2 triangle.h

```
#ifndef OOP_LAB7_TRIANGLE_H
#define OOP_LAB7_TRIANGLE_H

#include "figure.h"

namespace figures {

    class Triangle : public Figure {
    private:
        Dot *coordinates;
        uint32_t Id_;
    public:
        Triangle();
        explicit Triangle(uint32_t id, std::istream& is);
        Dot Center() const override;
        void PrintOut(std::ostream& os) const override;
```

```
        friend std::ostream& operator<<(std::ostream& os, const Triangle& A);
        double Area() const override;
        void Save(std::ofstream& os) const override;
        void Load (std::ifstream& is) override;
        uint32_t Id() const override;
        ~Triangle() override;
    };

} //namespace figures

class Triangle_factory : public Factory {
public:
    std::shared_ptr<figures::Figure> Figure_create() const override;
    std::shared_ptr<figures::Figure> Figure_create(uint32_t id, std::istream& is) const override;
};

#endif //OOP_LAB7_TRIANGLE_H
```

1.3 square.h

```
#ifndef OOP_LAB7_SQUARE_H
#define OOP_LAB7_SQUARE_H

#include "figure.h"

namespace figures {

    class Square : public Figure {
    private:
        Dot *coordinates;
        uint32_t Id_;
    public:
        Square();
        explicit Square(uint32_t id, std::istream& is);
        Dot Center() const override;
        double Area() const override ;
        void PrintOut(std::ostream& os) const override;
        uint32_t Id() const override;
        void Save(std::ofstream& os) const override;
        void Load(std::ifstream& is) override;
        friend std::ostream& operator<<(std::ostream& os, const Square& A);
        ~Square() override;
    };

} //namespace figures

class Square_factory : public Factory {
public:
    std::shared_ptr<figures::Figure> Figure_create() const override;
    std::shared_ptr<figures::Figure> Figure_create(uint32_t id, std::istream& is) const override;
};
```

```
#endif //OOP_LAB7_SQUARE_H
```

1.4 octagon.h

```
#ifndef OOP_LAB7_OCTAGON_H
#define OOP_LAB7_OCTAGON_H

#include "figure.h"

namespace figures {

    class Octagon : public Figure {
    private:
        Dot* coordinates;
        uint32_t Id_;
    public:
        Octagon();
        explicit Octagon(uint32_t id, std::istream& is);
        Dot Center() const override;
        void PrintOut(std::ostream& os) const override;
        friend std::ostream& operator<<(std::ostream& os, const Octagon& A);
        uint32_t Id() const override;
        double Area() const override;
        void Save(std::ofstream& os) const override;
        void Load(std::ifstream& is) override;
        ~Octagon() override;
    };

} //namespace figures

class Octagon_factory : public Factory {
public:
    std::shared_ptr<figures::Figure> Figure_create() const override;
    std::shared_ptr<figures::Figure> Figure_create(uint32_t id, std::istream& is) const override;
};

#endif //OOP_LAB7_OCTAGON_H
```

1.5 figure.h

```
#ifndef OOP_LAB7_FIGURE_H
#define OOP_LAB7_FIGURE_H

#include <memory>
#include <fstream>
#include "dot.h"
```

```
enum figure_t {
   TRIANGLE,
   SQUARE,
   OCTAGON
};

namespace figures {

   class Figure {
   public:
      virtual Dot Center() const = 0;
      virtual void PrintOut(std::ostream& os) const = 0;
      virtual double Area() const = 0;
      virtual void Save(std::ofstream& os) const = 0;
      virtual void Load(std::ifstream& is) = 0;
      virtual uint32_t Id() const = 0;
      virtual ~Figure() = default;
   };

} //namespace figures

class Factory {
public:
   virtual std::shared_ptr<figures::Figure> Figure_create() const = 0;
   virtual std::shared_ptr<figures::Figure> Figure_create(uint32_t id, std::istream& is) const = 0;
};

#endif //OOP_LAB7_FIGURE_H
```

1.6 document.h

```
#ifndef OOP_LAB7_DOCUMENT_H
#define OOP_LAB7_DOCUMENT_H

#include <fstream>
#include <memory>
#include <list>
#include "../figures/figure.h"
#include "../figures/octagon.h"
#include "../figures/square.h"
#include "../figures/triangle.h"

const uint32_t FORMAT_CODE = 06032001;

namespace doc_class {

   class Document {
   public:
      Document();
```

```
        explicit Document(std::string name);

        ~Document() = default;

        void Rename(const std::string &new_name);

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

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

        void Print() const;

        void Remove_figure(uint32_t id);

        void Remove_last_figure();

        void Add_figure(figure_t type, std::istream &is);

        uint32_t Get_position(uint32_t id);

        std::shared_ptr<figures::Figure> Get_figure(uint32_t id);

        void Insert_figure(uint32_t pos, std::shared_ptr<figures::Figure>& figure);

    private:
        uint32_t Id_;
        std::string Name_;
        std::list<std::shared_ptr<figures::Figure>> Buf_;
        Triangle_factory trg_fact;
        Square_factory sqr_fact;
        Octagon_factory oct_fact;

        void Save_impl(const std::string& filename) const;
        void Load_impl(const std::string& filename);
        };
}

#endif //OOP_LAB7_DOCUMENT_H
```

1.7 document.cpp

```
#include <algorithm>
#include <cstdint>
#include "document.h"

doc_class::Document::Document(): Id_(1), Name_(""), Buf_(0), trg_fact(), sqr_fact(), oct_fact() {}

doc_class::Document::Document(std::string name): Id_(1), Name_(std::move(name)), Buf_(0),
trg_fact(), sqr_fact(), oct_fact() {}

void doc_class::Document::Rename(const std::string &new_name) {
```

```
    Name_ = new_name;
}

void doc_class::Document::Save(const std::string &filename) const {
    Save_impl(filename);
}

void doc_class::Document::Load(const std::string &filename) {
    Load_impl(filename);
}

void doc_class::Document::Print() const {
    std::for_each(Buf_.begin(), Buf_.end(), [&](const std::shared_ptr<figures::Figure>& shape) {
        shape->PrintOut(std::cout);
        //std::cout << *shape << '\n';
    });
}

void doc_class::Document::Remove_figure(uint32_t id) {
    auto it = std::find_if(Buf_.begin(), Buf_.end(), [id](const std::shared_ptr<figures::Figure>&
shape) -> bool {
        return id == shape->Id();
    });

    if (it == Buf_.end())
        throw std::logic_error("Figure with this id doesn't exist");

    Buf_.erase(it);
}

void doc_class::Document::Remove_last_figure() {
    if (Buf_.empty()) {
        throw std::logic_error("Doc is empty");
    }
    Buf_.pop_back();
}

void doc_class::Document::Add_figure(figure_t type, std::istream& is) {
    switch (type) {
        case TRIANGLE:
            Buf_.push_back(trg_fact.Figure_create(Id_++, std::cin));
            break;
        case SQUARE:
            Buf_.push_back(sqr_fact.Figure_create(Id_++, std::cin));
            break;
        case OCTAGON:
            Buf_.push_back(oct_fact.Figure_create(Id_++, std::cin));
            break;
    }
}

uint32_t doc_class::Document::Get_position(uint32_t id) {
```

```
    auto it = std::find_if(Buf_.begin(), Buf_.end(), [id](std::shared_ptr<figures::Figure> shape) ->
bool {
        return id == shape->Id();
    });
    return std::distance(Buf_.begin(), it);
}

std::shared_ptr<figures::Figure> doc_class::Document::Get_figure(uint32_t id)  { //ПЕРЕДАЧА
ПО ССЫЛКЕ
    auto it = std::find_if(Buf_.begin(), Buf_.end(), [id](std::shared_ptr<figures::Figure>& shape) ->
bool {
        return id == shape->Id();
    });
    return *it;
}

void doc_class::Document::Insert_figure(uint32_t pos, std::shared_ptr<figures::Figure>& figure) {
    auto it = Buf_.begin();
    std::advance(it, pos);
    Buf_.insert(it, figure);
}

void doc_class::Document::Save_impl(const std::string &filename) const {
    std::ofstream os;
    os.open(filename, std::ios_base::binary | std::ios_base::out);
    if (!os.is_open()) {
        throw std::runtime_error("File is not opened");
    }
    uint32_t format = FORMAT_CODE;
    uint32_t nameLen = Name_.size();
    os.write((char*)&format, sizeof(format));
    os.write((char*)&nameLen, sizeof(nameLen));
    os.write((char*)(Name_.c_str()), nameLen);
    std::for_each(Buf_.begin(), Buf_.end(), [&](const std::shared_ptr<figures::Figure>& shape) {
        shape->Save(os);
    });
}

void doc_class::Document::Load_impl(const std::string &filename) {
    std::ifstream is;
    is.open(filename, std::ios_base::binary | std::ios_base::in);
    if (!is.is_open()) {
        throw std::runtime_error("File is not opened");
    }
    uint32_t format;
    uint32_t nameLen;
    is.read((char*)&format, sizeof(format));
    if (format != FORMAT_CODE)
        throw std::runtime_error("Bad file");
    is.read((char*)&nameLen, sizeof(nameLen));
    char* name = new char[nameLen + 1];
    name[nameLen] = 0;
```

```
    is.read(name, nameLen);
    Name_ = std::string(name);
    delete[] name;
    figure_t type;
    while(true) {
        is.read((char*)&type, sizeof(type));
        if (is.eof())
            break;
        switch (type) {
            case TRIANGLE:
                Buf_.push_back(trg_fact.Figure_create());
                break;
            case SQUARE:
                Buf_.push_back(sqr_fact.Figure_create());
                break;
            case OCTAGON:
                Buf_.push_back(oct_fact.Figure_create());
                break;
        }
        Buf_.back()->Load(is);
    }
    Id_ = Buf_.size();
```

1.8 command.h

```
#ifndef OOP_LAB7_COMMAND_H
#define OOP_LAB7_COMMAND_H

#include <stack>
#include <utility>
#include "doc_functions/document.h"

class Command {
public:
    virtual ~Command() = default;
    virtual void Execute() = 0;
    virtual void Abort() = 0;
    void SetDocument(std::shared_ptr<doc_class::Document> doc) {
        Doc_ = std::move(doc); //МУВ ДЛЯ ИЗБЕГАНИЯ ЛИШНЕГО КОПИРОВАНИЯ
    }
protected:
    std::shared_ptr<doc_class::Document> Doc_;
};

class Insert_cmd: public Command {
public:
    Insert_cmd(figure_t type, std::istream& is): Type_(type), Input_stream_(is) {}

    void Execute() override {
        Doc_->Add_figure(Type_, Input_stream_);
    }
```

```
    void Abort() override {
        Doc_->Remove_last_figure();
    }

private:
    figure_t Type_;
    std::istream& Input_stream_;
};

class Remove_cmd : public Command {
public:
    explicit Remove_cmd(uint32_t id): Id_(id), Pos_(0), Figure_(nullptr) {}

    void Execute() override {
        Figure_ = Doc_->Get_figure(Id_);
        Pos_ = Doc_->Get_position(Id_);
        Doc_->Remove_figure(Id_);
    }

    void Abort() override {
        Doc_->Insert_figure(Pos_, Figure_);
    }

private:
    uint32_t Id_;
    uint32_t Pos_;
    std::shared_ptr<figures::Figure> Figure_;

};

#endif //OOP_LAB7_COMMAND_H
```

1.9 editor.h

```
#ifndef OOP_LAB7_EDITOR_H
#define OOP_LAB7_EDITOR_H

#include <stack>
#include "doc_functions/document.h"
#include "command.h"

class Editor {

public:
    Editor(): Doc_(nullptr), History_() {}

    void Create_document(const std::string& name) {
        Doc_ = std::make_shared<doc_class::Document>(name);
    }

    void Insert_figure(figure_t type, std::istream& is) {
```

```
        std::shared_ptr<Command> command = std::shared_ptr<Command>(new Insert_cmd(type,
is));
        command->SetDocument(Doc_);
        command->Execute();
        History_.push(command);
    }

    void Remove_figure(uint32_t id) {
        std::shared_ptr<Command> command = std::shared_ptr<Command>(new Remove_cmd(id));
        command->SetDocument(Doc_);
        command->Execute();
        History_.push(command);
    }

    void Save_document(const std::string& filename) {
        Doc_->Save(filename);
    }

    void Load_document(const std::string& filename) {
        Doc_ = std::make_shared<doc_class::Document>("NoName");
        Doc_->Load(filename);
    }

    void Undo() {
        if (History_.empty()) {
            throw std::logic_error("History is empty");
        }
        std::shared_ptr<Command> last_cmd = History_.top();
        last_cmd->Abort();
        History_.pop();
    }

    void Print_document() {
        Doc_->Print();
    }

    bool Document_exist() {
        return Doc_ != nullptr;
    }

    ~Editor() = default;

private:
    std::shared_ptr<doc_class::Document> Doc_;
    std::stack<std::shared_ptr<Command>> History_;
};

#endif //OOP_LAB7_EDITOR_H
```

1.10 main.cpp

```
#include <iostream>
#include "figures/octagon.h"
#include "figures/square.h"
#include "figures/triangle.h"
#include "doc_functions/document.h"
#include "command.h"
#include "editor.h"

bool quit (Editor& editor) {
    char c;
    std::cout << "Do you want to save before exit? [Y/N]: ";
    std::cin >> c;
    if (c == 'N' || c == 'n') {
        return true;
    }
    else if (c == 'Y' || c == 'y') {
        std::string name;
        std::cout << "Enter name for savefile: ";
        std::cin >> name;
        try {
            editor.Save_document(name);
            std::cout << "Successfully saved in " << name << '\n';
        } catch  (std::runtime_error& err) {
            std::cout << err.what() << "\n";
            return false;
        }
        return true;
    } else {
        std::cout << "so yes or no?\n";
        return false;
    }
}

void man () {
    std::cout << "create - create new document\n"
    << "save - save current document to file\n"
    << "load - load document from file\n"
    << "add - add new figure\n"
    << "remove - remove figure by it`s ID\n"
    << "undo - abort previous operation\n"
    << "print - print the document contents\n"
    << "quit - close program and exit\n";
}

bool create(Editor& editor) {
    char c;
    if (editor.Document_exist()) {
        std::cout << "Save current document? [Y/N]\n";
        std::cin >> c;
        if (c == 'N' || c == 'n') {
        }
        else if (c == 'Y' || c == 'y') {
```

```
            std::string name;
            std::cout << "Enter name for savefile: ";
            std::cin >> name;
            try {
                editor.Save_document(name);
                std::cout << "Successfully saved in " << name << '\n';
            } catch  (std::runtime_error& err) {
                std::cout << err.what() << "\n";
                return false;
            }
        } else {
            std::cout << "so yes or no?\n";
            return false;
        }
    }
    std::string doc_name;
    std::cout << "Enter name of new project\n";
    std::cin >> doc_name;
    editor.Create_document(doc_name);
    std::cout << "Document " << doc_name << " is created\n";
    return true;
}

bool load(Editor& editor) {
    char c;
    if (editor.Document_exist()) {
        std::cout << "Save current document? [Y/N]\n";
        std::cin >> c;
        if (c == 'N' || c == 'n') {
        }
        else if (c == 'Y' || c == 'y') {
            std::string name;
            std::cout << "Enter name for savefile: ";
            std::cin >> name;
            try {
                editor.Save_document(name);
                std::cout << "Successfully saved in " << name << '\n';
            } catch  (std::runtime_error& err) {
                std::cout << err.what() << "\n";
                return false;
            }
        } else {
            std::cout << "so yes or no?\n";
            return false;
        }
    }
    std::string file_name;
    std::cout << "Enter name of load file\n";
    std::cin >> file_name;
    try {
        editor.Load_document(file_name);
        std::cout << "Successfully loaded from " << file_name << "\n";
```

```
    } catch (std::runtime_error& err) {
        std::cout << err.what() << "\n";
        return false;
    }
    return true;
}

bool save(Editor& editor) {
    std::string file_name;
    std::cout << "Enter name for savefile: ";
    std::cin >> file_name;

    try {
        editor.Save_document(file_name);
        std::cout << "Successfully saved in " << file_name << '\n';
    } catch (std::runtime_error& err) {
        std::cout << err.what() << "\n";
        return false;
    }
    return true;
}

void add (Editor& editor) {
    int type;
    std::cout << "Enter type of figure (1 for triangle, 2 for square, 3 - for octagon): ";
    std::cin >> type;
    switch(type) {
        case 1:
            std::cout << "Enter triangle (X, then Y for each dot): ";
            editor.Insert_figure(TRIANGLE, std::cin);
            break;
        case 2:
            std::cout << "Enter square (coordinates of 2 opposite angles): ";
            editor.Insert_figure(SQUARE, std::cin);
            break;
        case 3:
            std::cout << "Enter octagon (enter dots consequently): ";
            editor.Insert_figure(OCTAGON, std::cin);
            break;
        default:
            std::cout << "Please, enter 1, 2 or 3 to choose figure\n";
            return;
    }
    std::cout << "Figure is added\n";
}

bool remove(Editor& editor) {
    uint32_t id;
    std::cout << "enter ID of figure you fant to remove (you can see it in print): ";
    std::cin >> id;

    try {
```

```
        editor.Remove_figure(id);
        std::cout << "Figure with ID " << id << " is removed\n";
    } catch (std::logic_error& err) {
        std::cout << err.what() << "\n";
        return false;
    }
    return true;
}



int main() {
    Editor editor;
    std::string cmd;

    while (cmd != "quit") {
        std::cin >> cmd;
        if (cmd == "quit") {
           if (quit(editor)) return 0;
        } else if (cmd == "man") {
            man();
        } else if (cmd == "create") {
            create(editor);
        } else if (cmd == "save ") {
            save(editor);
        } else if (cmd == "load") {
            load(editor);
        } else if (cmd == "add") {
            add(editor);
        } else if (cmd == "remove") {
            remove(editor);
        } else if (cmd == "undo") {
            editor.Undo();
            std::cout << "Done (hopefully)\n";
        } else if (cmd == "print") {
            editor.Print_document();
        }
    }
    return 0;
}
```

2. Ссылка на репозиторий на GitHub

https://github.com/Siegmeyer1/oop_exercise_07

3. Набор тестов

1)
man
create document
add
1
0 2 2 0 0 0
add
1
0 2 2 0 2 2
add
2
0 0 3 3
add
2
0 2 2 2
print
remove
3
print
create
Y
save.txt
new_document
load
N
save.txt
print
quit

2)
create
undo_test
add
2
0 0 1 1
add
2
0 0 2 2
print
undo
print
add 1 0 0 3 0 0 3
print
remove

3
print
undo
print
remove
1
undo
print
quit
n

4. Результаты тестов

1)

man
create - create new document
save - save current document to file
load - load document from file
add - add new figure
remove - remove figure by it`s ID
undo - abort previous operation
print - print the document contents
quit - close program and exit
create document
Enter name of new project
Document document is created
add
Enter type of figure (1 for triangle, 2 for square, 3 - for octagon): 1
Enter triangle (X, then Y for each dot): 0 2 2 0 0 0
Figure is added
add
Enter type of figure (1 for triangle, 2 for square, 3 - for octagon): 1
Enter triangle (X, then Y for each dot): 0 2 2 0 2 2
Figure is added
add
Enter type of figure (1 for triangle, 2 for square, 3 - for octagon): 2
Enter square (coordinates of 2 opposite angles): 0 0 3 3
Figure is added
add
Enter type of figure (1 for triangle, 2 for square, 3 - for octagon): 2
Enter square (coordinates of 2 opposite angles): 0 2 2 2
Figure is added
print
ID: 1 Type: triangle Area: 2 Center: (0.666667; 0.666667) Dots: (0; 2), (2; 0), (0; 0)
ID: 2 Type: triangle Area: 2 Center: (1.33333; 1.33333) Dots: (0; 2), (2; 0), (2; 2)

ID: 3 Type: square Area: 9 Center: (1.5; 1.5) Dots: (3; 0), (0; 0), (0; 3), (3; 3)
ID: 4 Type: square Area: 2 Center: (1; 2) Dots: (1; 1), (0; 2), (1; 3), (2; 2)
remove
enter ID of figure you fant to remove (you can see it in print): 3
Figure with ID 3 is removed
print
ID: 1 Type: triangle Area: 2 Center: (0.666667; 0.666667) Dots: (0; 2), (2; 0), (0; 0)
ID: 2 Type: triangle Area: 2 Center: (1.33333; 1.33333) Dots: (0; 2), (2; 0), (2; 2)
ID: 4 Type: square Area: 2 Center: (1; 2) Dots: (1; 1), (0; 2), (1; 3), (2; 2)
create
Save current document? [Y/N]
Y
Enter name for savefile: save.txt
Successfully saved in save.txt
Enter name of new project
new_document
Document new_document is created
load
Save current document? [Y/N]
N
Enter name of load file
save.txt
Successfully loaded from save.txt
print
ID: 1 Type: triangle Area: 2 Center: (0.666667; 0.666667) Dots: (0; 2), (2; 0), (0; 0)
ID: 2 Type: triangle Area: 2 Center: (1.33333; 1.33333) Dots: (0; 2), (2; 0), (2; 2)
ID: 4 Type: square Area: 2 Center: (1; 2) Dots: (1; 1), (0; 2), (1; 3), (2; 2)
quit
Do you want to save before exit? [Y/N]: n
anri@andrew-HP-250-G6:~/Documents/Github_repositories/OOP_lab7/build$

2)

create
Enter name of new project
undo_test
Document undo_test is created
add
Enter type of figure (1 for triangle, 2 for square, 3 - for octagon): 2
Enter square (coordinates of 2 opposite angles): 0 0 1 1
Figure is added
add
Enter type of figure (1 for triangle, 2 for square, 3 - for octagon): 2
Enter square (coordinates of 2 opposite angles): 0 0 2 2
Figure is added
print
ID: 1 Type: square Area: 1 Center: (0.5; 0.5) Dots: (1; 0), (0; 0), (0; 1), (1; 1)
ID: 2 Type: square Area: 4 Center: (1; 1) Dots: (2; 0), (0; 0), (0; 2), (2; 2)
undo

```
Done (hopefully)
print
ID: 1   Type: square     Area: 1             Center: (0.5; 0.5)          Dots: (1; 0), (0; 0), (0; 1), (1; 1)
add 1 0 0 3 0 0 3
Enter type of figure (1 for triangle, 2 for square, 3 - for octagon): Enter triangle (X, then Y for each dot): Figure is added
print
ID: 1   Type: square     Area: 1             Center: (0.5; 0.5)          Dots: (1; 0), (0; 0), (0; 1), (1; 1)
ID: 3   Type: triangle   Area: 4.5           Center: (1; 1)        Dots: (0; 0), (3; 0), (0; 3)
remove
enter ID of figure you fant to remove (you can see it in print): 3
Figure with ID 3 is removed
print
ID: 1   Type: square     Area: 1             Center: (0.5; 0.5)          Dots: (1; 0), (0; 0), (0; 1), (1; 1)
undo
Done (hopefully)
print
ID: 1   Type: square     Area: 1             Center: (0.5; 0.5)          Dots: (1; 0), (0; 0), (0; 1), (1; 1)
ID: 3   Type: triangle   Area: 4.5           Center: (1; 1)        Dots: (0; 0), (3; 0), (0; 3)
remove
enter ID of figure you fant to remove (you can see it in print): 1
Figure with ID 1 is removed
undo
Done (hopefully)
print
ID: 1   Type: square     Area: 1             Center: (0.5; 0.5)          Dots: (1; 0), (0; 0), (0; 1), (1; 1)
ID: 3   Type: triangle   Area: 4.5           Center: (1; 1)        Dots: (0; 0), (3; 0), (0; 3)
quit
Do you want to save before exit? [Y/N]: n
anri@andrew-HP-250-G6:~/Documents/Github_repositories/OOP_lab7/build$
```

5. Объяснение работы программы

Итак. У нас есть 3 класса фигур, которые наследуются от абстрактного класса figure. Все фигуры вынесеныв отдельный нейпспейс figures. Конструкторы всех фигур вызываются отдельным классом factory, который в свою очередь тоже наследуется от абстрактного. Еслть класс Editor, который является “обёрткой” над классом документа, вызывает его методы через КОМАНДЫ, и записывает их в стек команд для возможности их отмены. Класс команд – наследуется от абстрактного класса для удобного вызова из стека команд. Каждый экземпляр команды имеет методы “выполнить” и “отменить”, а также в случае, если это команда удаления – ссылку на удалённую фигуру, её позицию и ID. Про ID: он присваивается каждой фигуре при добавлении, сохраняется при удалении и загрузке. Возможно является слабым местом программы, но мне её сломать не удалось.

Вывод

Проделав работу, я создал достаточно сложную систему классов, лучше разобрался в принципах наследования, и организации подобных структур классов. Как мне кажется, начал понимать, в чём суть объектно-ориентированного программирования. Успешно распилил один файл с классами фигур на 3.