ЛАБОРАТОРНАЯ РАБОТА

по дисциплине «Programming Techniques»

направления 10.05.01 «Компьютерная безопасность»

Практическое задание по теме «Паттерны проектирования 1»

Вариант 8

Выполнил студент группы СКБ172

Федоров Сергей Андреевич

«25» Мая 2020 г.

Москва, 2020

Задача**:**

**Вариант 8.**

*Вариант 8. Используя паттерн Builder, реализовать процесс создания объекта*

*типа «Пицца» разных видов.*

**Цель лабораторной работы**

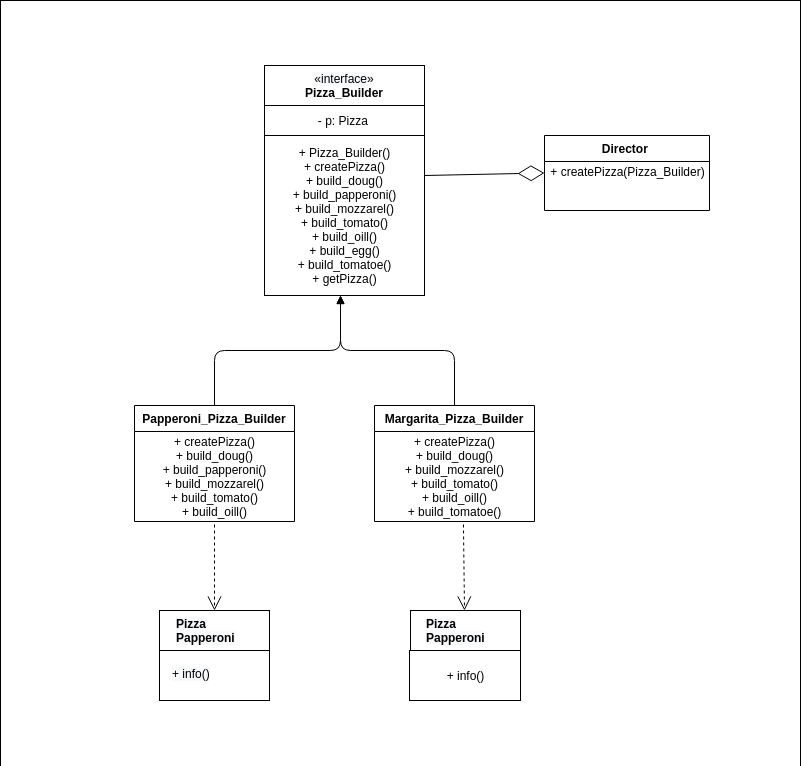
Цель лабораторной работы – познакомиться с различными типами паттернов программирования, научиться использовать их в прикладных задачах, сделать выводы о возможном использовании паттернов.

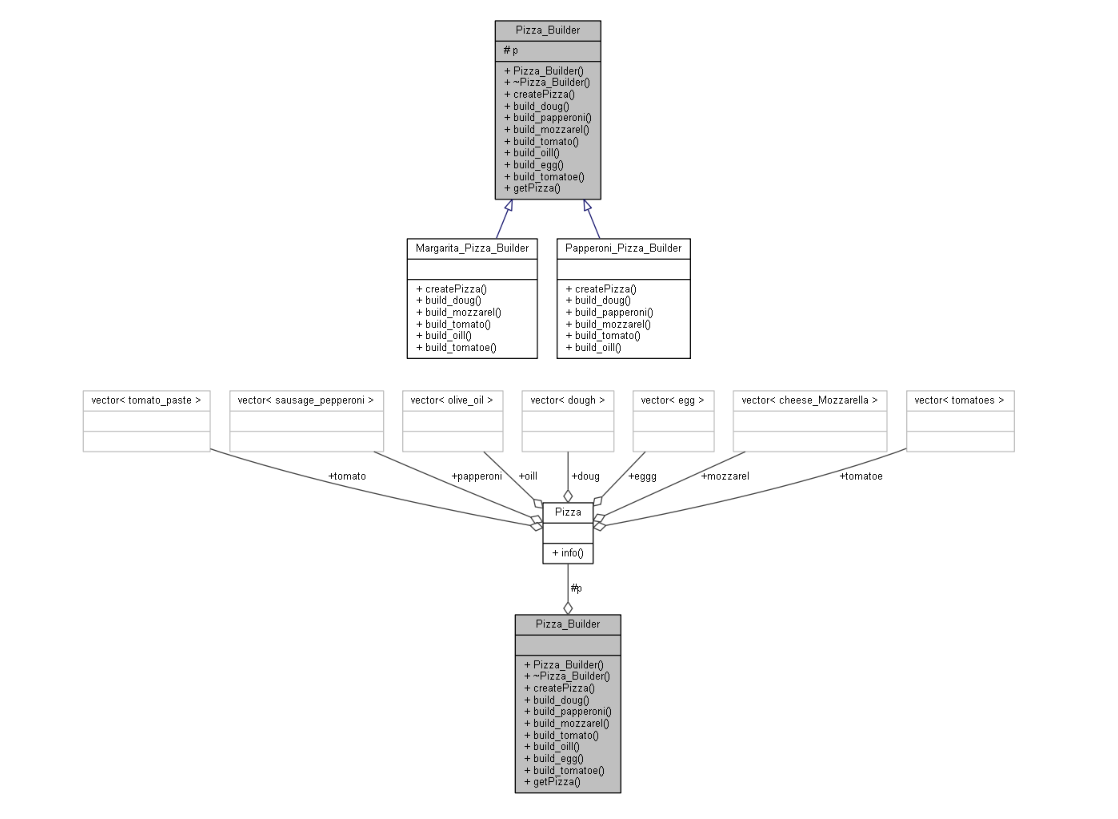
**Результаты работы**

1. Используя паттерн *Builder*, был реализован процесс создания пиццы типа пепперони и маргарита.

****

**«UML-диаграмма»**

****



Построено с помощью доксиген.

**Вывод:**

Данный паттерн удобен когда ваш код должен создавать разные представления какого-то объекта. Например, пиццу разного вида.

**Код программы:**

*/\*! \mainpage <CENTER> Лабораторная работа N5 Выполнил Федоров Сергей Скб172 </CENTER>*

*\* <CENTER> Вариант 8 </CENTER>*

*\* Вариант 8. Используя паттерн Builder, реализовать процесс создания объекта типа «Пицца» разных видов.*

*\*/*

#include <iostream>

#include <vector>

using namespace std;

// Сначала объявим продукты (классы всех возможных продуктов) которые используются для приготовления пиццы

*/\*!*

*\brief Тесто*

*Ингредиент*

*\*/*

class dough

{

public:

void info() {

cout << "Dough" << endl;

}

};

*/\*!*

*\brief колбаса пепперони*

*Ингредиент*

*\*/*

class sausage\_pepperoni

{

public:

void info() {

cout << "Sausage pepperoni" << endl;

}

};

*/\*!*

*\brief Сыр моцарелла*

*Ингредиент*

*\*/*

class cheese\_Mozzarella

{

public:

void info() {

cout << "Mozzarella" << endl;

}

};

*/\*!*

*\brief Томатная паста*

*Ингредиент*

*\*/*

class tomato\_paste

{

public:

void info() {

cout << "Tomato paste" << endl;

}

};

*/\*!*

*\brief Оливковое масло*

*Ингредиент*

*\*/*

class olive\_oil

{

public:

void info() {

cout << "Olive oil" << endl;

}

};

*/\*!*

*\brief Яйца*

*Ингредиент*

*\*/*

class egg

{

public:

void info() {

cout << "Egg" << endl;

}

};

*/\*!*

*\brief Помидор*

*ингредиент*

*\*/*

class tomatoes

{

public:

void info() {

cout << "Tomatoes" << endl;

}

};

*/\*!*

*\brief Класс пицца*

*Содержит все возможные ингредиенты*

*\*/*

class Pizza

{

public:

vector <dough> doug;

vector <sausage\_pepperoni> papperoni;

vector <cheese\_Mozzarella> mozzarel ;

vector <tomato\_paste> tomato;

vector <olive\_oil> oill;

vector <egg> eggg;

vector <tomatoes> tomatoe;

void info() {

int i;

int counter=1;

for(i=0; i<doug.size(); ++i) {cout<<counter <<") "; doug[i].info(); counter++;}

for(i=0; i<papperoni.size(); ++i) {cout<<counter <<") "; papperoni[i].info(); counter++;}

for(i=0; i<mozzarel.size(); ++i) {cout<<counter <<") "; mozzarel[i].info(); counter++;}

for(i=0; i<tomato.size(); ++i) {cout<<counter <<") "; tomato[i].info(); counter++;}

for(i=0; i<oill.size(); ++i) {cout<<counter <<") "; oill[i].info(); counter++;}

for(i=0; i<eggg.size(); ++i) {cout<<counter <<") "; eggg[i].info(); counter++;}

for(i=0; i<tomatoe.size(); ++i) {cout<<counter <<") "; tomatoe[i].info(); counter++;}

}

};

*/\*!*

*\brief Базовый класс*

*Базовый класс который объявляет интерфейс для поэтапного построения пиццы и предусматривает его реализацию по умолчанию*

*\*/*

class Pizza\_Builder

{

protected:

Pizza\* p;

public:

Pizza\_Builder(): p(0) {}

virtual ~Pizza\_Builder() {}

virtual void createPizza() {}

virtual void build\_doug() {}

virtual void build\_papperoni() {}

virtual void build\_mozzarel() {}

virtual void build\_tomato() {}

virtual void build\_oill() {}

virtual void build\_egg() {}

virtual void build\_tomatoe() {}

virtual Pizza\* getPizza() { return p; }

};

*/\*!*

*\brief Пепперони*

*Состоит из: 1) Тесто 2) Колбаса пепперони 3) Моцарелла 4) Томатная паста 5) Оливковое масло*

*\*/*

class Papperoni\_Pizza\_Builder: public Pizza\_Builder

{

public:

void createPizza() { p = new Pizza; }

void build\_doug() { p->doug.push\_back( dough()); }

void build\_papperoni() { p->papperoni.push\_back( sausage\_pepperoni()); }

void build\_mozzarel() { p->mozzarel.push\_back( cheese\_Mozzarella()); }

void build\_tomato() { p->tomato.push\_back( tomato\_paste()); }

void build\_oill() { p->oill.push\_back( olive\_oil()); }

};

*/\*!*

*\brief Маргарита*

*Состоит из: 1) Моцарелла 2) Оливковое масло 3) Томатная паста 4) Помидоры 5) Тесто*

*\*/*

class Margarita\_Pizza\_Builder: public Pizza\_Builder

{

public:

void createPizza() { p = new Pizza; }

void build\_doug() { p->doug.push\_back( dough()); }

void build\_mozzarel() { p->mozzarel.push\_back( cheese\_Mozzarella()); }

void build\_tomato() { p->tomato.push\_back( tomato\_paste()); }

void build\_oill() { p->oill.push\_back( olive\_oil()); }

void build\_tomatoe() { p->tomatoe.push\_back( tomatoes()); }

};

*/\*!*

*\brief Класс-распорядитель*

*Поэтапно создает пиццу определенного вида.*

*Тут мы можем найти алгорим построения пиццы.*

*\*/*

class Director

{

public:

Pizza\* createPizza( Pizza\_Builder & builder )

{

builder.createPizza();

builder.build\_doug();

builder.build\_papperoni();

builder.build\_mozzarel();

builder.build\_tomato();

builder.build\_oill();

builder.build\_tomatoe();

builder.build\_egg();

return( builder.getPizza());

}

};

*/\*!*

*\brief главная функция*

*И тут происходит client часть*

*\*/*

int main()

{

Director dir;

Papperoni\_Pizza\_Builder PP\_builder;

Margarita\_Pizza\_Builder MP\_builder;

Pizza \* Papperonii = dir.createPizza( PP\_builder);

Pizza \* Margaritaa = dir.createPizza( MP\_builder);

cout << "Papperoni:" << endl;

Papperonii->info();

cout << "**\n**Morgarita:" << endl;

Margaritaa->info();

return 0;

}