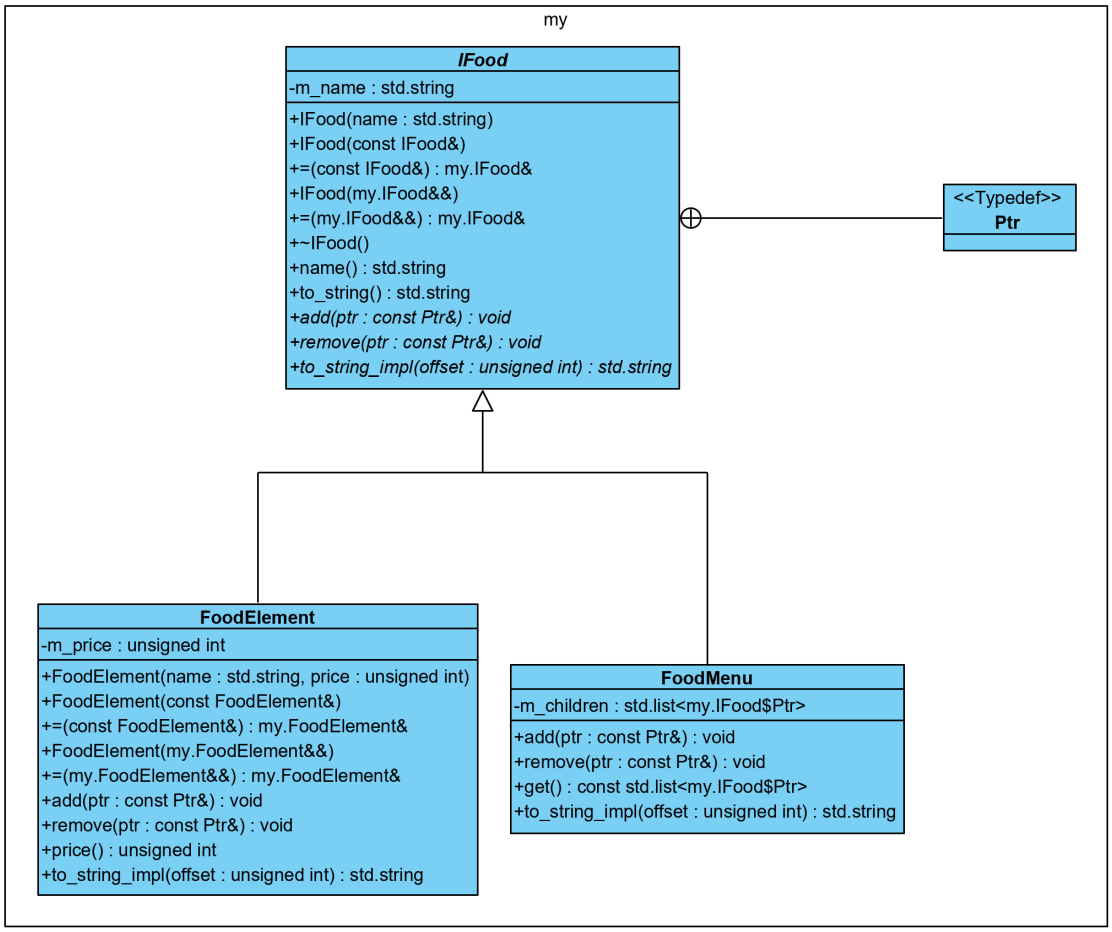
Лабораторная работа №5
«ПАТТЕРНЫ ПРОЕКТИРОВАНИЯ»
(часть 1)

по дисциплине "Методы программирования"

Выполнил Ковалев Даниил
СКБ171, вариант 2 (12)
МИЭМ НИУ ВШЭ

Условие. Используя паттерн Composite, реализовать иерархическую структуру с вложенными (многоуровневыми) подменю по категориям (кухня: холодные закуски, горячие закуски, салаты, супы, основные блюда; бар: безалкогольные напитки, алкогольные напитки, чай, кофе и т.д.).

UML-диаграмма классов:



Структура проекта:

- lab5:
  - CMakeLists.txt
  - composite.h — файл с описанием необходимых классов
  - composite.cpp — файл с реализацией необходимых классов
  - main.cpp — файл с примером использования паттерна Composite
- CMakeLists.txt — см. ЛР1

Результат работы программы:

```
Menu:

- Sandwitches:

  - Pork                $2.10
  - Beef                $3.50
  - Cheese              $1.80

- Desserts:

  - Ice cream           $3.00
  - Fruit cake          $4.05
```

Листинги с исходным кодом всех файлов расположены на следующих страницах отчета.

Листинг 1: lab5/CMakeLists.txt

```
set(PROJECT_NAME 5_patterns)

project(${PROJECT_NAME} LANGUAGES CXX)

set(SOURCES main.cpp composite.cpp)
set(HEADERS composite.h)

add_executable(${PROJECT_NAME} ${SOURCES} ${HEADERS})
```

Листинг 2: lab5/main.cpp

```
#include "composite.h"

#include <iostream>
#include <stdexcept>
#include <memory>

int main() try
{
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);

    my::IFood::Ptr menu        = std::make_shared<my::FoodMenu>("Menu");

    my::IFood::Ptr sandwitches = std::make_shared<my::FoodMenu>("Sandwitches");
    my::IFood::Ptr desserts    = std::make_shared<my::FoodMenu>("Desserts");

    my::IFood::Ptr pork        = std::make_shared<my::FoodElement>("Pork", 210);
    my::IFood::Ptr beef        = std::make_shared<my::FoodElement>("Beef", 350);
    my::IFood::Ptr cheese      = std::make_shared<my::FoodElement>("Cheese", 180);

    my::IFood::Ptr ice_cream   = std::make_shared<my::FoodElement>("Ice cream", 300);
    my::IFood::Ptr fruit_cake  = std::make_shared<my::FoodElement>("Fruit cake", 405);

    sandwitches->add(std::move(pork));
    sandwitches->add(std::move(beef));
    sandwitches->add(std::move(cheese));

    desserts->add(std::move(ice_cream));
    desserts->add(std::move(fruit_cake));

    menu->add(std::move(sandwitches));
    menu->add(std::move(desserts));

    std::cout << menu->to_string();
}
catch (const std::exception& e)
{
    std::cerr << e.what() << '\n';
    return 1;
}
```

Листинг 3: lab5/composite.h

```
#ifndef COMPOSITE_H
#define COMPOSITE_H

#include <list>
#include <memory>
#include <string>

namespace my
{

class IFood
```

```
12 {
13 public:
14     using Ptr = std::shared_ptr<IFood>;
15
16     IFood(std::string name);
17
18     IFood(const IFood&) = default;
19     IFood& operator=(const IFood&) = default;
20     IFood(IFood&&) = default;
21     IFood& operator=(IFood&&) = default;
22
23     virtual ~IFood() = default;
24
25     std::string name() const;
26     std::string to_string() const;
27
28     virtual void add(const Ptr& ptr) = 0;
29     virtual void remove(const Ptr& ptr) = 0;
30
31     virtual std::string to_string_impl(unsigned int offset) const = 0;
32
33 private:
34     std::string m_name;
35 };
36
37 class FoodMenu : public IFood
38 {
39 public:
40     using IFood::IFood;
41
42     void add(const Ptr& ptr) override;
43     void remove(const Ptr& ptr) override;
44
45     const std::list<Ptr> get() const;
46
47     std::string to_string_impl(unsigned int offset) const override;
48
49 private:
50     std::list<Ptr> m_children;
51 };
52
53 class FoodElement : public IFood
54 {
55 public:
56     FoodElement(std::string name, unsigned int price);
57
58     FoodElement(const FoodElement&) = default;
59     FoodElement& operator=(const FoodElement&) = default;
60     FoodElement(FoodElement&&) = default;
61     FoodElement& operator=(FoodElement&&) = default;
62
63     void add(const Ptr& ptr) override;
64     void remove(const Ptr& ptr) override;
65
66     unsigned int price() const;
67
68     std::string to_string_impl(unsigned int offset) const override;
69
70 private:
71     unsigned int m_price;
72 };
73
74 } // namespace my
75
76 #endif // COMPOSITE_H
```

Листинг 4: lab5/composite.cpp

```cpp
#include "composite.h"

#include <sstream>
#include <utility>

namespace my
{

IFood::IFood(std::string name)
    : m_name(std::move(name))
{
}

std::string IFood::name() const
{
    return m_name;
}

std::string IFood::to_string() const
{
    return to_string_impl(0);
}

void FoodMenu::add(const FoodMenu::Ptr& ptr)
{
    m_children.emplace_back(ptr);
}

void FoodMenu::remove(const FoodMenu::Ptr& ptr)
{
    m_children.remove(ptr);
}

const std::list<IFood::Ptr> FoodMenu::get() const
{
    return m_children;
}

std::string FoodMenu::to_string_impl(unsigned int offset) const
{
    std::string offset_str(offset, ' ');
    std::ostringstream stream;
    stream << name() << ":\n\n";
    for (const Ptr& elem : m_children)
        stream << offset_str << "- " << elem->to_string_impl(offset + 2) << '\n';
    return stream.str();
}

FoodElement::FoodElement(std::string name, unsigned int price)
    : IFood(std::move(name))
    , m_price(price)
{
}

void FoodElement::add(const Ptr& /* ptr */)
{
    throw std::runtime_error("Can't add to FoodElement");
}

void FoodElement::remove(const Ptr& /* ptr */)
{
    throw std::runtime_error("Can't remove from FoodElement");
}

unsigned int FoodElement::price() const
{
    return m_price;
}
```

```
69
70 std::string FoodElement::to_string_impl(unsigned int /* offset */) const
71 {
72     unsigned int dollars = m_price / 100, cents = m_price - dollars * 100;
73     std::string ret = name() + "\t\t$" + std::to_string(dollars) + '.';
74     if (cents == 0)
75         ret += "00";
76     else if (cents < 10)
77         ret += "0" + std::to_string(cents);
78     else
79         ret += std::to_string(cents);
80     return ret;
81 }
82
83 } // namespace my
```