

Студент: Минчаков Аркадий Сергеевич

Задание на лабораторную работу:

Для заданного варианта задания (предметной области) разработать UML-диаграмму классов и диаграмму последовательности. Разработать программу решения задания в виде консольного приложения (C++, C#) с использованием принципа подстановки Лисков (LSP), принципа разделения интерфейсов (ISP) и принципа инверсии (DIP). Допускается вводить дополнительные понятия предметной области. Наделите классы атрибутами и функциональностью по вашему усмотрению. В программе предусмотрите тестирование функциональности созданных объектов классов.

Номер в списке - 15, вариант 2:

Организация, страховая компания, нефтегазовая компания, завод.

UML-диаграмма классов:

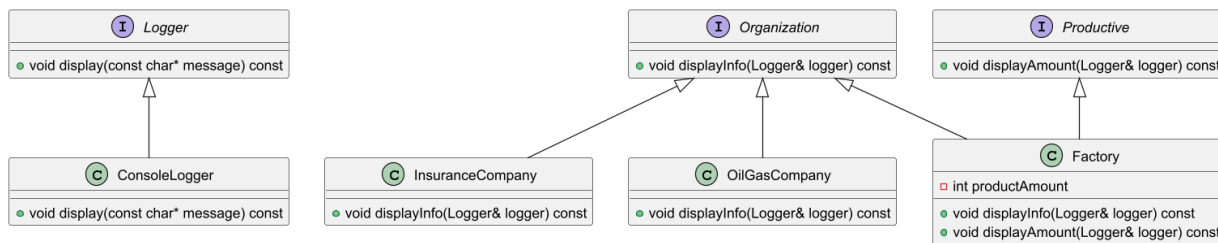


Figure 1: UML-диаграмма классов

Диаграмма последовательности:

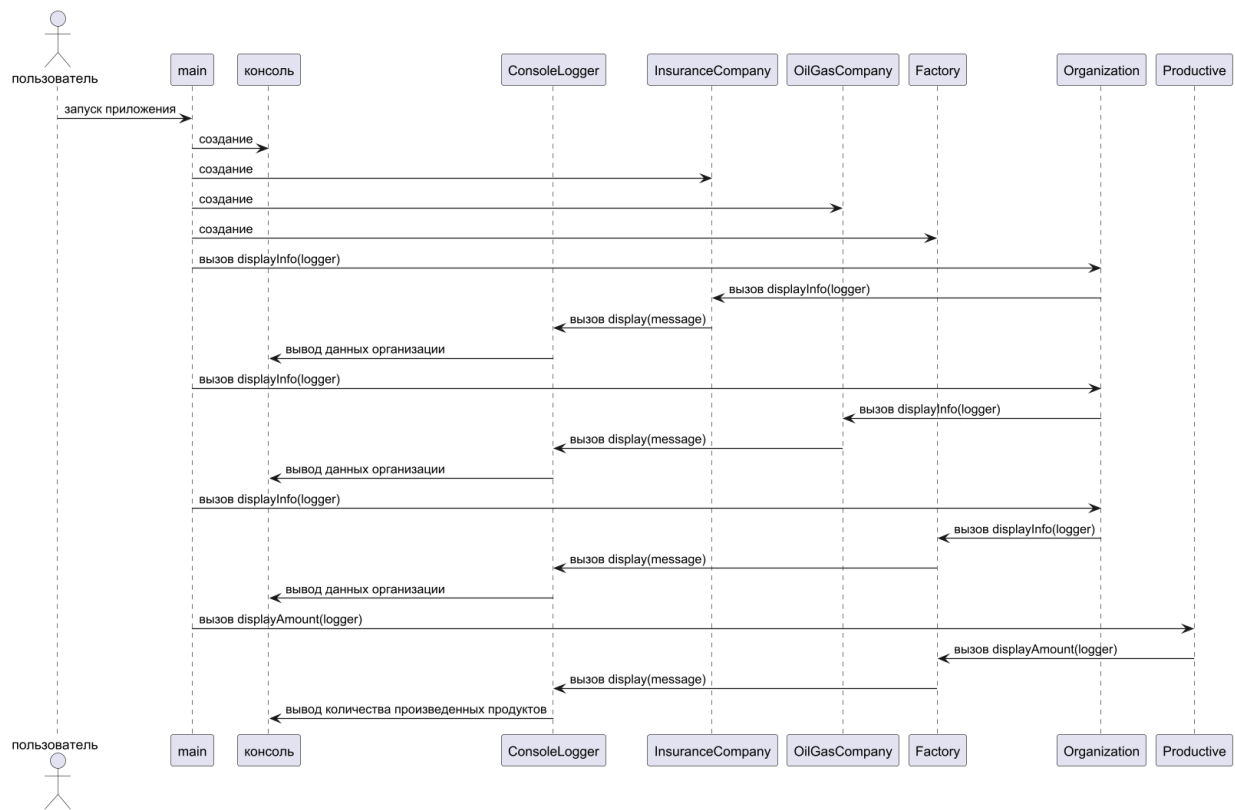


Figure 2: Диаграмма последовательности

Исходный код программы:

main.cpp

```

1  #include <iostream>
2  #include <vector>
3  #include <memory>
4
5  using std::cout, std::endl, std::make_unique, std::unique_ptr, std::vector;
6
7  class Logger {
8  public:
9      virtual ~Logger() = default;
10
11     virtual void display(const char *message) const = 0;
12 };
13
14 class ConsoleLogger : public Logger {
15 public:
16     void display(const char *message) const override {
17         std::cout << message << std::endl;
18     }
19 };
20
21 class Organization {
22 public:
23     virtual ~Organization() = default;
24
25     virtual void displayInfo(Logger &logger) const = 0;
26 };
27

```

```

28 // принцип разделения интерфейсов
29 class Productive {
30 public:
31     virtual ~Productive() = default;
32
33     virtual void displayAmount(Logger &logger) const = 0;
34 };
35
36 class InsuranceCompany : public Organization {
37 public:
38     void displayInfo(Logger &logger) const override {
39         // принцип инверсии зависимостей
40         logger.display("This is an insurance company.");
41     }
42 };
43
44 class OilGasCompany : public Organization {
45 public:
46     void displayInfo(Logger &logger) const override {
47         logger.display("This is an oil and gas company.");
48     }
49 };
50
51 class Factory : public Organization, public Productive {
52 private:
53     int productAmount;
54 public:
55     explicit Factory(int amount) : productAmount(amount) {}
56
57     void displayInfo(Logger &logger) const override {
58         logger.display("This is a factory.");
59     }
60
61     void displayAmount(Logger &logger) const override {
62         logger.display(("Product amount: " + std::to_string(productAmount) +
63             ".").c_str());
64     }
65 };
66
67 int main() {
68     ConsoleLogger logger;
69
70     // принцип подстановки Лисков
71     vector<unique_ptr<Organization>> organizations;
72
73     organizations.push_back(make_unique<InsuranceCompany>());
74     organizations.push_back(make_unique<OilGasCompany>());
75     organizations.push_back(make_unique<Factory>(100));
76
77     for (const auto &organization: organizations) {
78         organization->displayInfo(logger);
79     }
80
81     unique_ptr<Productive> productive = make_unique<Factory>(200);
82     productive->displayAmount(logger);
83
84     return 0;
85 }

```