Технологии конструирования программного обеспечения

Отчет по лабораторной работе № 06

Группа: 221-329 **Студент:** Минчаков Аркадий Сергеевич

Задание на лабораторную работу:

На базе любой из ранее выполненных лабораторных работ разработать приложение (C++, C#) (применение Windows-форм позволит заработать дополнительные баллы) с применением паттерна MVC для разделения сложной модели (основной класс) и её представления.

В качестве базы была выбрана 4 лабораторная работа.

UML-диаграмма классов:

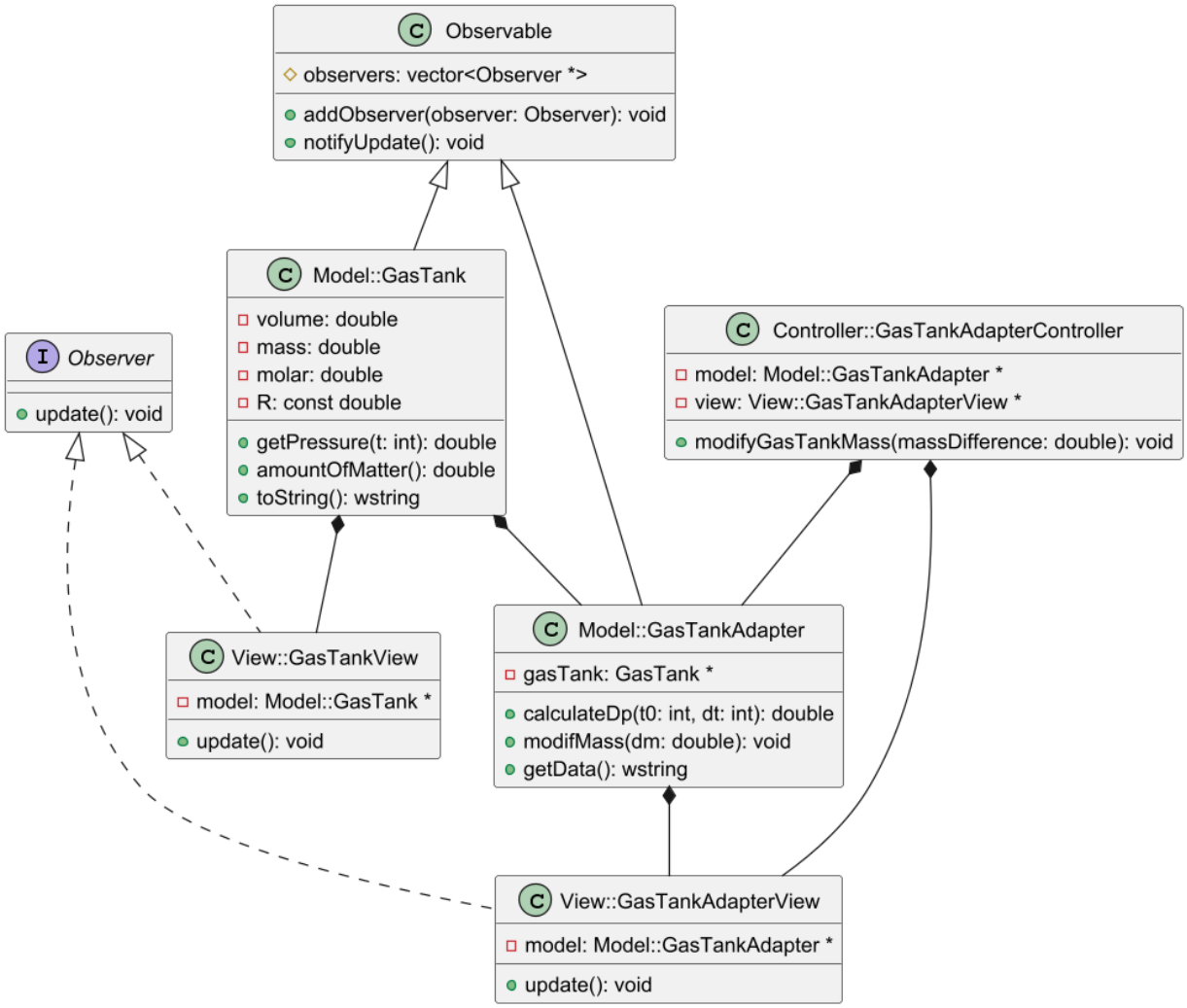

Figure 1: UML-диаграмма классов

Диаграмма последовательности:

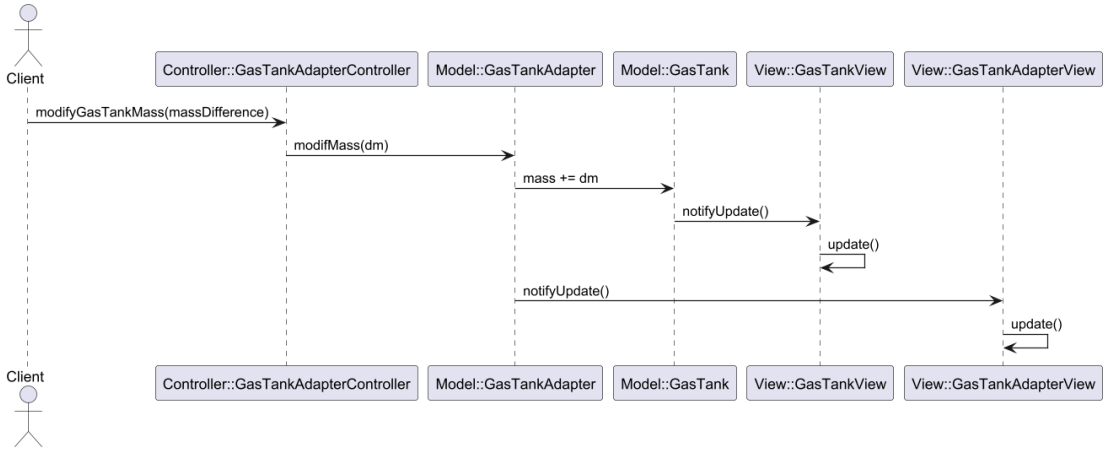

Figure 2: Диаграмма последовательности

Исходный код программы:

main.cpp

```
#include <string>
#include <iostream>
#include <vector>

class Observer {
public:
    virtual void update() = 0;
};

class Observable {
protected:
    std::vector<Observer *> observers;
public:
    void addObserver(Observer *observer) {
        observers.push_back(observer);
    }

    void notifyUpdate() {
        int size = observers.size();
        for (int i = 0; i < size; i++) {
            observers[i]->update();
        }
    }
};

namespace Model {
    class GasTank : public Observable {
    private:
        const double R = 8.31;

    public:
        double volume;

        double mass;

        double molar;
```

```
37
38          GasTank(double volume, double mass, double molar) : volume(volume),
                mass(mass), molar(molar) {}
39
40          double getPressure(int t) {
41              // p V = v R T => p = v R T / V
42              return amountOfMatter() * R * t / volume;
43          }
44
45          double amountOfMatter() {
46              // p V = v R T = m R T / M => v = m / M
47              return mass / molar;
48          }
49
50          std::wstring toString() {
51              return L"Баллон с газом {\n"
52                       "  Объем сосуда: " + std::to_wstring(volume) + L" м^3\n"
53                       "  Масса газа: " + std::to_wstring(mass) + L" кг\n"
54                       "  Молярная масса газа: " + std::to_wstring(molar) + L"
                             кг/моль\n}";
55          }
56      };
57
58      class GasTankAdapter : public Observable {
59      private:
60          GasTank *gasTank;
61
62      public:
63          GasTankAdapter(GasTank *gasTank) : gasTank(gasTank) {}
64
65          double calculateDp(int t0, int dt) {
66              // dp(t0, dt) = p - p0 = p(t0 + dt) - p(t0)
67              return gasTank->getPressure(t0 + dt) - gasTank->getPressure(t0);
68          }
69
70          void modifMass(double dm) {
71              gasTank->mass += dm;
72              gasTank->notifyUpdate();
73              notifyUpdate();
74          }
75
76          std::wstring getData() {
77              return L"Adapter {\n" + gasTank->toString() + L"\n}";
78          }
79      };
80  }
81
82  namespace View {
83      class GasTankView : public Observer {
84      private:
85          Model::GasTank *model;
86
87      public:
88          explicit GasTankView(Model::GasTank *model) : model(model) {
89              model->addObserver(this);
90              update();
91          }
92
93          void update() override {
94              std::wcout << L"--------(GasTankView update begin)--------" <<
                    std::endl;
95              std::wcout << model->toString() << std::endl;
96              std::wcout << L"* Количество вещества в этом баллоне: " <<
                    model->amountOfMatter() << L" моль" << std::endl;
97              std::wcout << L"--------(GasTankView update end)--------" <<
                    std::endl;
```

```
        }
    };

    class GasTankAdapterView : public Observer {
    private:
        Model::GasTankAdapter *model;

    public:
        explicit GasTankAdapterView(Model::GasTankAdapter *model) : model(model)
            {
            model->addObserver(this);
            update();
        }

        void update() override {
            std::wcout << L"--------(GasTankAdapterView update begin)--------"
                << std::endl;
            std::wcout << model->getData() << std::endl;
            std::wcout << L"--------(GasTankAdapterView update end)--------" <<
                std::endl;
        }
    };
}

namespace Controller {
    class GasTankAdapterController {
    private:
        Model::GasTankAdapter *model;
        View::GasTankAdapterView *view;
    public:
        GasTankAdapterController(Model::GasTankAdapter *model,
            View::GasTankAdapterView *view) :
            model(model), view(view) {}

        void modifyGasTankMass(double massDifference) {
            model->modifMass(massDifference);
        }
    };
}

int main() {
    setlocale(LC_ALL, "Russian");

    std::wcout << L"Модели отображения созданы:" << std::endl << std::endl;

    Model::GasTank gasTank(3.12, 50, 0.002016);

    View::GasTankView gasTankView(&gasTank);

    Model::GasTankAdapter gasTankAdapter(&gasTank);

    View::GasTankAdapterView gasTankAdapterView(&gasTankAdapter);

    std::wcout << std::endl << std::endl << L"Создаем контроллер и меняем через него
        массу газа в баллоне" << std::endl << std::endl;

    Controller::GasTankAdapterController
        gasTankAdapterController(&gasTankAdapter, &gasTankAdapterView);

    gasTankAdapterController.modifyGasTankMass(40);

    return 0;
}
```