**Технологии конструирования программного обеспечения**

**Отчет по лабораторной работе № 01**

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

## Задание на лабораторную работу:

Для заданного варианта задания разработать UML-диаграмму классов и диаграмму последовательности. Разработать программу решения задания в виде консольного приложения (C++, C#) с использованием принципа единственной обязанности (SRP) и принципа открытости/закрытости (OCP). Допускается вводить дополнительные понятия предметной области. В программе предусмотрите тестирование функциональности созданных объектов классов.

## Номер в списке - 15, вариант 3:

Система Больница. Пациенту назначается лечащий Врач. Врач может сделать назначение Пациенту (процедуры, лекарства, операции). Медсестра или другой Врач выполняют назначение. Пациент может быть выписан из Больницы по окончании лечения, при нарушении режима или при иных обстоятельствах.

## UML-диаграмма классов:

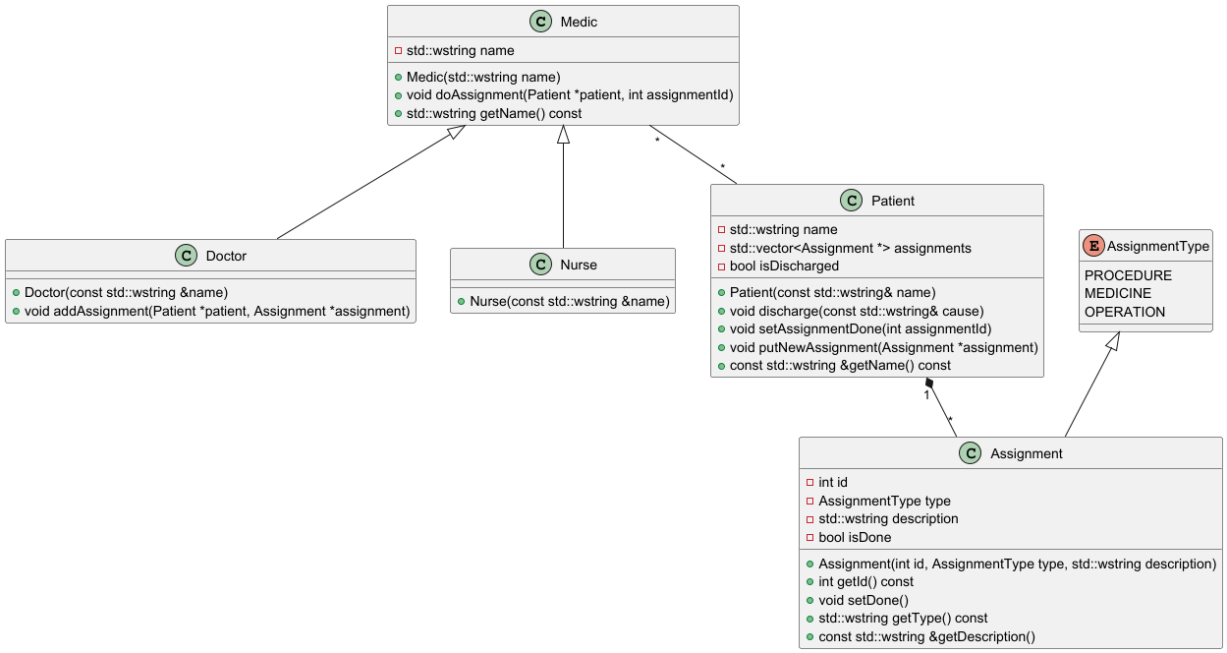


Figure 1: UML-диаграмма классов

## Диаграмма последовательности:

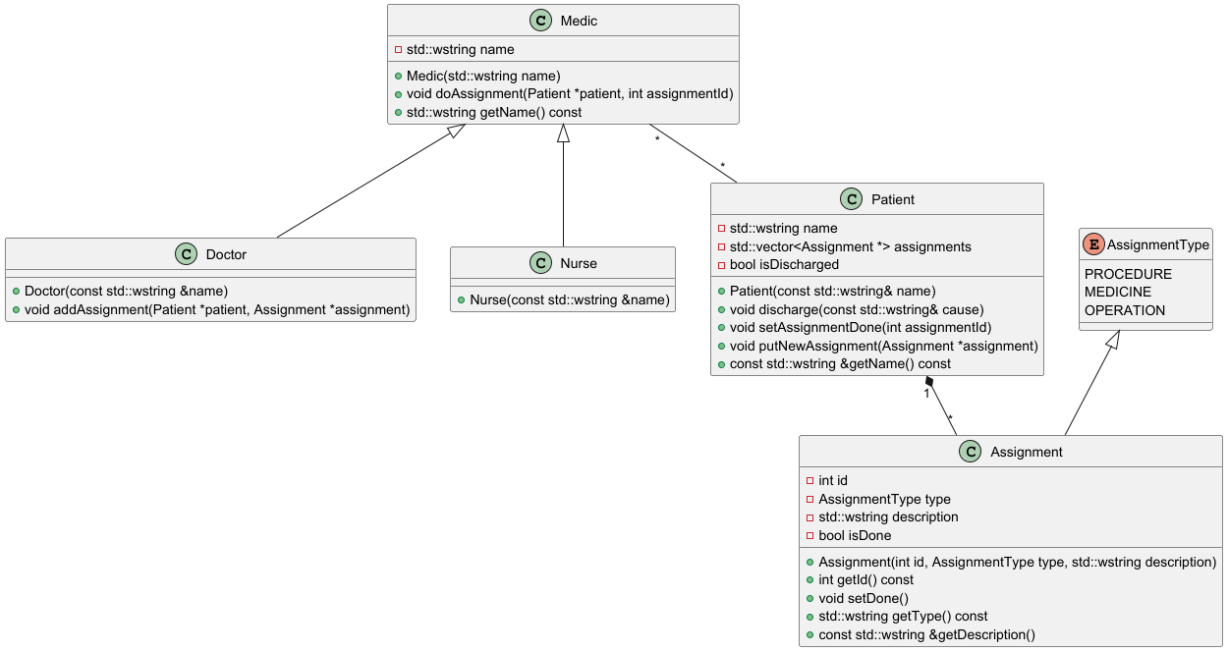


Figure 2: Диаграмма последовательности

## Исходный код программы:

### main.cpp

```cpp
#include <iostream>
#include <vector>

void log(const std::wstring &&message) {
    std::wcout << message << std::endl;
}

enum AssignmentType {
    PROCEDURE,
    MEDICINE,
    OPERATION
};

class Assignment {
    int id;
    AssignmentType type;
    std::wstring description;
    bool isDone;

public:
    explicit Assignment(int id, AssignmentType type, const std::wstring
        &description) :
            id(id), type(type), description(description), isDone(false) {}

    int getId() {
        return id;
    }

    void setDone() {
```

```
29          isDone = true;
30      }
31
32      std::wstring getType() {
33          switch (type) {
34              case AssignmentType::PROCEDURE:
35                  return L"'Процедура'";
36              case AssignmentType::MEDICINE:
37                  return L"'Лекарство'";
38              case AssignmentType::OPERATION:
39                  return L"'Операция'";
40          }
41      }
42
43      std::wstring getDescription() {
44          return description;
45      }
46  };
47
48  class Patient {
49      std::wstring name;
50      std::vector<Assignment *> assignments;
51      bool isDischarged;
52
53  public:
54      explicit Patient(const std::wstring &name) : name(name), assignments(),
            isDischarged(false) {
55          log(L"Добавлен пациент " + name);
56      }
57
58      void discharge(const std::wstring &cause) {
59          log(L"Пациент " + name + L" выписан по причине: " + cause);
60          isDischarged = true;
61      }
62
63      void setAssignmentDone(int assignmentId) {
64          for (auto &assignment: assignments) {
65              if (assignment->getId() == assignmentId) {
66                  assignment->setDone();
67                  return;
68              }
69          }
70      }
71
72      void putNewAssignment(Assignment *assignment) {
73          assignments.push_back(assignment);
74      }
75
76      std::wstring getName() {
77          return name;
78      }
79  };
80
81  class Medic {
82  protected:
83      std::wstring name;
84
85  public:
86      explicit Medic(const std::wstring &name) : name(name) {}
87
88      void doAssignment(Patient *patient, int assignmentId) {
89          log(L"Медицинский работник " + name + L" выполняет назначение с id=" +
                std::to_wstring(assignmentId));
90          patient->setAssignmentDone(assignmentId);
91      }
92
```

```
93      std::wstring getName() {
94          return name;
95      }
96  };
97
98  class Doctor : public Medic {
99  public:
100     explicit Doctor(const std::wstring &name) : Medic(name) {
101         log(L"Добавлен доктор " + name);
102     }
103
104     void addAssignment(Patient *patient, Assignment *assignment) {
105         log(L"Доктор " + name + L" назначает пациенту " + patient->getName() + L"
                процедуру с типом "
106              + assignment->getType() + L": " + assignment->getDescription());
107         patient->putNewAssignment(assignment);
108     }
109 };
110
111 class Nurse : public Medic {
112 public:
113     explicit Nurse(const std::wstring &name) : Medic(name) {
114         log(L"Добавлена медсестра " + name);
115     }
116 };
117
118
119 int main() {
120     setlocale(LC_ALL, "Russian");
121
122     Doctor doctor1(L"Доктор №1");
123     Doctor doctor2(L"Доктор №2");
124
125     Nurse nurse(L"Медсестра №1");
126
127     Patient patient1(L"Пациент №1");
128     Patient patient2(L"Пациент №2");
129     Patient patient3(L"Пациент №3");
130
131     doctor1.addAssignment(&patient1, new Assignment(1,
            AssignmentType::PROCEDURE, L"Прогулка на свежем воздухе"));
132     doctor2.addAssignment(&patient2, new Assignment(2, AssignmentType::MEDICINE,
            L"Парацетамол, 2 таблетки"));
133     doctor1.addAssignment(&patient3, new Assignment(3,
            AssignmentType::OPERATION, L"Пересадка почки"));
134     doctor2.addAssignment(&patient3, new Assignment(4,
            AssignmentType::PROCEDURE, L"Подтягивания, 15 раз"));
135
136     nurse.doAssignment(&patient1, 1);
137     doctor1.doAssignment(&patient2, 2);
138     nurse.doAssignment(&patient3, 4);
139     doctor2.doAssignment(&patient3, 3);
140
141     patient1.discharge(L"Выздоровел");
142     patient2.discharge(L"Нарушение режима");
143
144     return 0;
145 }
```