# Отчет по практикуму: Моделирование работы страховой компании

12 ноября 2025 г.

## Содержание

# 1 Формулировка задания

Задача заключалась в разработке модели управления страховой компанией, выполняющей страхование населения по трём направлениям:

- Страхование жилища
- Страхование автомобиля
- Страхование здоровья

Цель симуляции — дать пользователю (менеджеру компании) возможность управлять условиями страховых продуктов и наблюдать за финансовым состоянием компании в течение нескольких месяцев. Модель включает:

- Продажу полисов в зависимости от базового спроса и параметров продукта
- Выплаты по страховым случаям
- Уплату налога
- Возможность обновления условий полисов
- Риск банкротства компании

# 2 Диаграмма классов

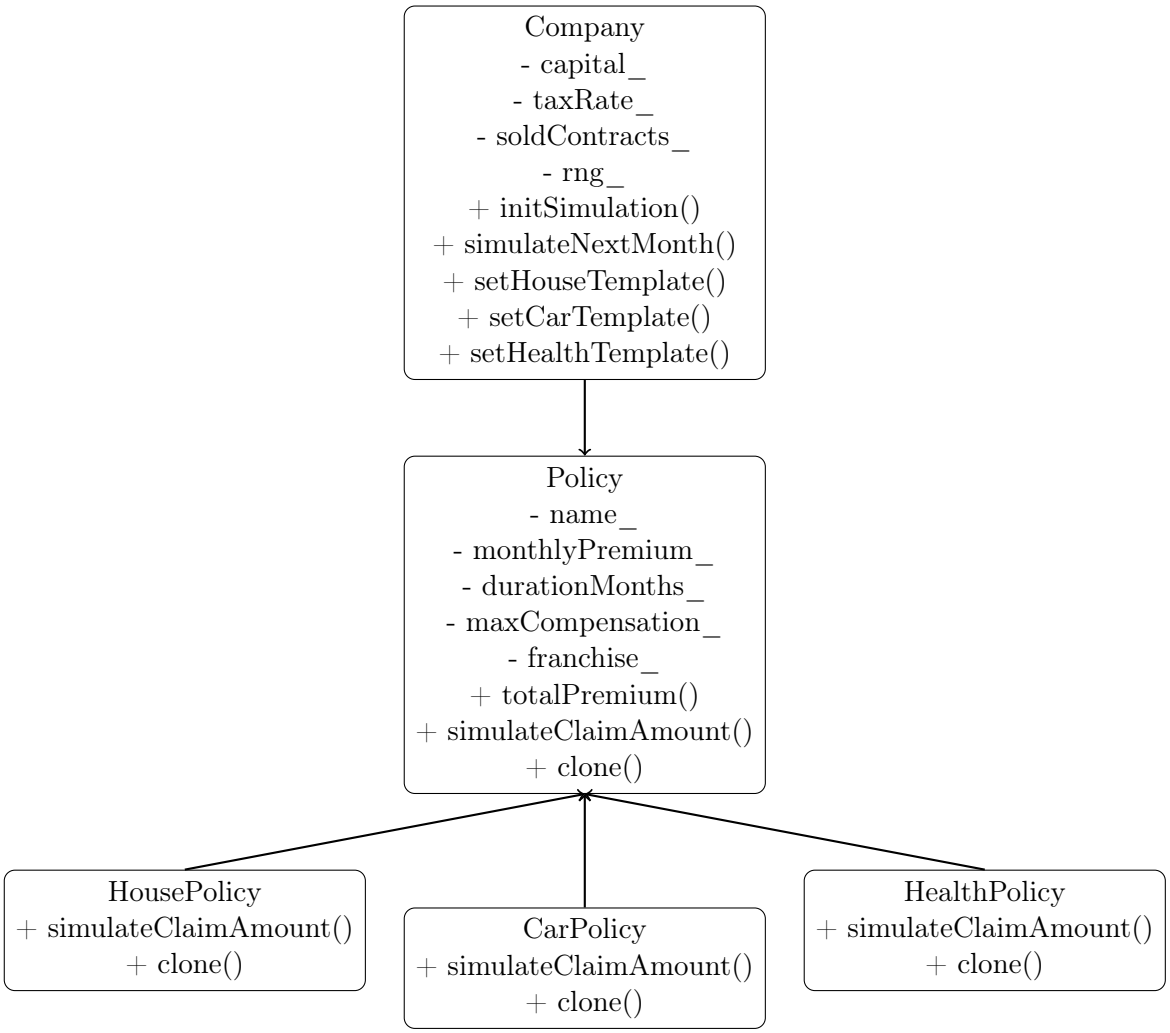


Рис. 1: Диаграмма классов программы

# 3 Спецификация основных классов

## 3.1 Policy

```
1  //
2  class Policy {
3  public:
4      Policy(const QString& name, double monthlyPremium, int
           durationMonths,
5              double maxCompensation, double franchise);
6      virtual ~Policy();
7      QString name() const;
8      double monthlyPremium() const;
9      int durationMonths() const;
10     double maxCompensation() const;
11     double franchise() const;
12     virtual double totalPremium() const;
13     virtual double simulateClaimAmount(std::mt19937 &rng) const = 0;
```

```
14      virtual std::unique_ptr<Policy> clone() const = 0;
15  };
```

### 3.2 HousePolicy / CarPolicy / HealthPolicy

Наследники класса Policy, реализуют метод `simulateClaimAmount()` для имитации страховых случаев.

### 3.3 Company

```
1   class Company {
2   public:
3       Company(double initialCapital = 30000.0, double taxRate = 0.09);
4       void setBaseDemandHouse(int d);
5       void setBaseDemandCar(int d);
6       void setBaseDemandHealth(int d);
7       void setHouseTemplate(std::unique_ptr<Policy> p);
8       void setCarTemplate(std::unique_ptr<Policy> p);
9       void setHealthTemplate(std::unique_ptr<Policy> p);
10      void setTrustFactor(double factor);
11      void initSimulation(int monthsToSimulate, std::uint32_t rngSeed =
            std::random_device{}());
12      QString simulateNextMonth();
13      double capital() const;
14      bool isBankrupt() const;
15      int currentMonth() const;
16  private:
17      double capital_;
18      double taxRate_;
19      std::vector<SoldContract> soldContracts_;
20      std::mt19937 rng_;
21  };
```

## 4 Диаграмма объектов

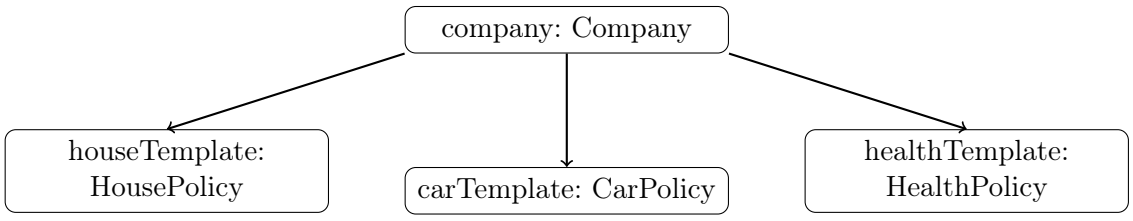


Рис. 2: Диаграмма объектов программы

## 5 Использованные инструментальные средства

- Язык C++17
- Qt6 Widgets (QMainWindow, QPushButton, QLineEdit, QTextEdit, QLabel, QTableWidget)

- Стандартная библиотека C++ (std::vector, std::unique_ptr, std::mt19937, std::uniform_int_distribution, std::uniform_real_distribution)
- CMake 3.16+

# 6 Модульная структура программы

- **policy.h / policy.cpp** — классы Policy и наследники, фабричные функции создания полисов
- **company.h / company.cpp** — класс Company, управление капиталом, продажами и выплатами
- **mainwindow.h / mainwindow.cpp** — графический интерфейс, управление симуляцией
- **main.cpp** — точка входа приложения
- **CMakeLists.txt** — сборка проекта

# 7 Описание пользовательского интерфейса

Программа содержит главное окно с тремя основными блоками:

1. **Параметры компании:** ввод начального капитала, налоговой ставки, количества месяцев моделирования.
2. **Условия полисов:** ввод ежемесячного взноса, длительности, максимальной суммы возмещения и франшизы для каждого типа полиса.
3. **Базовый спрос и доверие:** настройка базового спроса по каждому типу полиса и коэффициента доверия населения.

Ниже расположены кнопки управления:

- Инициализация симуляции
- Следующий месяц
- Обновить условия полисов

В нижней части окна отображается журнал симуляции и текущий капитал компании.