**МИНОБРНАУКИ РОССИИ**
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**
**Кафедра Информационных систем**

# КУРСОВАЯ РАБОТА

**по дисциплине «Объектно-ориентированное программирование»**

**Тема: Разработка в объектно-ориентированной методологии распределенную программную систему (ПС), имитирующую жизненный цикл объектов описанной ПрО.**

**Вариант: 3**

Студент гр. 8374 _____ Пихтовников К.С.

Студент гр. 8374 _____ Подсекин Г.С.

Преподаватель _____ Егоров С.С.

Санкт-Петербург
2021

**ЗАДАНИЕ**
**НА КУРСОВУЮ РАБОТУ**

Студенты: Пихтовников К.С., Подсекин Г.С.

Группа 8374

Тема работы: Разработка в объектно-ориентированной методологии распределенной программной системы (ПС), имитирующую жизненный цикл объектов описанной ПрО.

Исходные данные:

В инструментальном отделении сборочного цеха работают N кладовщиков. Свободного кладовщика они выбирают равновероятно, а при занятости всех становятся в очередь. Очередь ограничены величинами M1, M2,..., MN соответственно. Обслуженные рабочие уходят.

Содержание пояснительной записки: «Содержание», «Введение», «Постановка задачи», «Приложение “Интерфейс”», «Приложение “Модель”», «Перечень типов и структуры сообщений», «Заключение», «Список использованных источников».

Предполагаемый объем пояснительной записки:

Не менее 20 страниц.

Дата выдачи задания: 10.02.2021

Дата сдачи работы: 01.06.2021

Дата защиты работы: 00.00.2021

Студент гр. 8374 _____ Пихтовников К.С.

Студент гр. 8374 _____ Подсекин Г.С.

Преподаватель _____ Егоров С.С.

## АННОТАЦИЯ

В данной курсовой работе реализуется программная система с интерфейсом, которая имитирует процессы сборочного цеха. Разработка ведется в объектно-ориентированной методологии с использованием языка программирования C++ и фреймворка Qt. Объектная модель описывается с помощью диаграмм в терминах UML.



## SUMMARY

In this course work, a software system with an interface that simulates the processes of the assembly shop is implemented. The development is carried out in an object-oriented methodology using the C++ programming language and the Qt framework. The object model is described using diagrams in UML terms.

**СОДЕРЖАНИЕ**

# ВВЕДЕНИЕ

Объектно-ориентированное программирование – это методика разработки программ, в основе которой лежит понятие объекта как некоторой структуры, описывающей объект реального мира, его поведение. Задача, решаемая с использованием методики объектно-ориентированного программирования, описывается в терминах объектов и операций над ними, а программа при таком подходе представляет собой набор объектов и связей между ними. Другими словами, можно сказать, что объектно-ориентированное программирование представляет собой метод программирования, которые весьма близко напоминает наше поведение.

Объектно-ориентированный язык программирования характеризуется тремя основными свойствами: инкапсуляция, наследование, полиморфизм

Для данной курсовой работы наше приложение было написано, опираясь на парадигмы объектно-ориентрованного программирования с использованием языка C++, конструкторских классов библиотеки C++ и фреймворка для разработки программного обеспечения - QT.

# 1. ПОСТАНОВКА ЗАДАЧИ

В инструментальном отделении сборочного цеха работают N кладовщиков. Свободного кладовщика они выбирают равновероятно, а при занятости всех становятся в очередь. Очередь ограничены величинами M1, M2,..., MN соответственно. Обслуженные рабочие уходят.

Требуется разработать в объектно-ориентированной методологии распределенную программную систему (ПС), имитирующую жизненный цикл объектов описанной ПрО.

ПС должна состоять из двух взаимодействующих приложений. Одного GUI приложения «Интерфейс» (клиент) и другого консольного «Модель» (сервер).

Приложение «Интерфейс» должно обеспечивать сетевое взаимодействие с приложением «Модель» для обеспечения своих интерфейсных функций. Реализация GUI интерфейса должна быть в виде множества интерфейсных окон, каждое из которых отвечает за определенный функциональный состав взаимодействия с пользователем:

- основное окно осуществляет управление окнами приложения «Интерфейс»,
- окно задания и отображения актуальных параметров модели ПрО,
- окно управления событиями должно состоять из интерфейсных элементов, необходимых для формирования команд управления моделью: «Начальное состояние», «Выбор и инициирование событий, изменяющих состояние ПрО», «Выбор файла сценария потока событий»
- отображение состояния объектов ПрО.

Подсистема «Модель» должна:

- обеспечить функции сервера сетевого взаимодействия с приложением клиента «Модель»,
- содержать реализацию объектной модели предметной области,

- обеспечивать изменение состояний объектов ПрО по командам управления,

- обеспечить передачу своего состояния в приложение «Интерфейс» для его отображения.

### 1.1. Описание предметной области

Предметной областью является пункт сборочного цеха, в который поступает поток рабочих с определённой интенсивностью, в результате чего формируется очередь, не превышающая фиксированной длины. Определенный рабочий выбирает свободного кладовщика с равной вероятностью. На обслуживание кладовщиком каждого рабочего тратится некоторое количество времени. Обслуженный рабочий уходит из пункта сборочного цеха.

### 1.2. Перечень библиотечных конструкторских классов, использованных в проекте

#### 1) QApplication

Класс QApplication управляет главным потоком и основными настройками приложения с GUI. Он содержит главный цикл обработки сообщений, где обрабатываются и пересылаются все сообщения посланные оконной системой и другими ресурсами. Также тут реализованы инициализация, завершение приложения и управление сессией. Также в данном классе реализованы возможности расширения системы и приложения.

#### 2) QWidget

Класс QWidget является базовым для всех объектов пользовательского интерфейса.

#### 3) QObject

Класс QObject — это базовый класс для всех объектов Qt.

Главная особенность в этой модели — это очень мощный механизм для связи объектов, называемый сигналами и слотами.

#### 4) QPushButton

QPushButton предоставляет командную кнопку.

Нажимная кнопка или командная кнопка, возможно, является наиболее часто используемым виджетом в любом графическом пользовательском интерфейсе. Нажмите (щелкните) кнопку, чтобы дать компьютеру команду

выполнить какое-либо действие или ответить на вопрос. Типичные кнопки: «ОК», «Применить», «Отмена», «Закрыть», «Да», «Нет» и «Справка»

**5) QSpinBox**

Класс QSpinBox предоставляет виджет счетчика. QSpinBox позволяет пользователю указать значение щелкая мышью по кнопкам вверх/вниз или нажимая клавиши клавиатуры вверх/вниз для увеличения/уменьшения значения, отображенного в настоящий момент. Также пользователь может ввести значение вручную.

**6) QCloseEvent**

Класс QCloseEvent содержит параметры, описывающие событие закрытия.

События закрытия отправляются виджетам, которые пользователь хочет закрыть, обычно путем выбора «Close» в меню окна или путем нажатия кнопки X в строке заголовка. Они также отправляются, когда вы вызываете QWidget :: close() для программного закрытия виджета.

**7) QLabel**

QLabel обеспечивает отображение текста или изображения. Функциональность взаимодействия с пользователем не предусмотрена. Внешний вид метки можно настроить различными способами, и его можно использовать для указания мнемонической клавиши фокуса для другого виджета.

**8) QDialog**

Класс QDialog является базовым классом для диалоговых окон.

**9) QMainWindow**

Класс QMainWindow предоставляет главное окно приложения.

Главное окно предоставляет структуру для создания пользовательского интерфейса приложения. Qt имеет класс QMainWindow и связанные с ним классы для управления главным окном.

**10) QHostAddress**

Этот класс содержит IPv4 или IPv6 - адрес независимо от платформы и протокола.

QHostAddress обычно используется с QTcpSocket, QTcpServer и QUdpSocket для подключения к хосту или настройки сервера.

**11) QUdpSocket**

QUdpSocket — это дейтаграммный сокет, для осуществления обмена пакетами данных. С помощью этого сокета данные отправляются без проверки дошли ли данные или нет.

# 2. ПРИЛОЖЕНИЕ «ИНТЕРФЕЙС»

## 2.1. Графическое представление интерфейсных окон

### 2.1.1. Основное окно

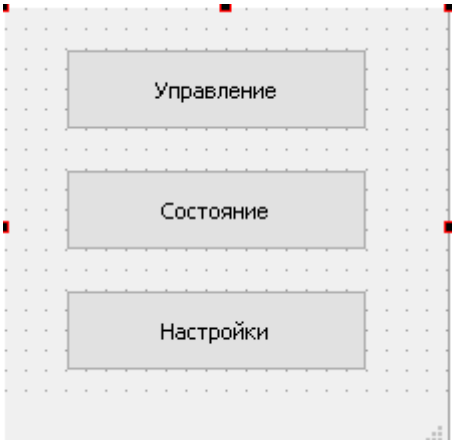


*Рис.1. Основное окно программы*

### 2.1.2. Окно параметров ПрО

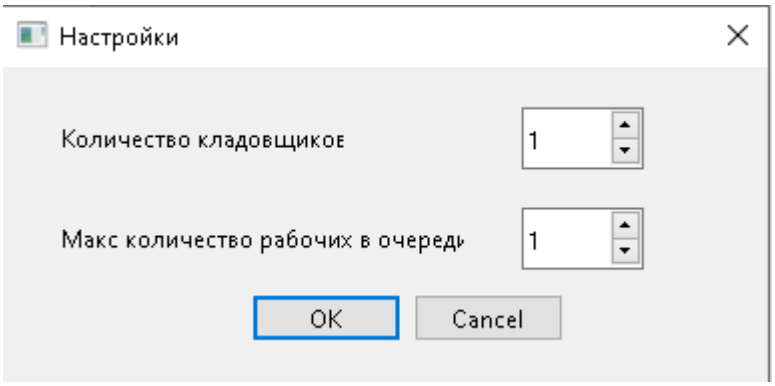


*Рис.2. Окно параметров программы*

### 2.1.3. Окно управления событиями ПрО

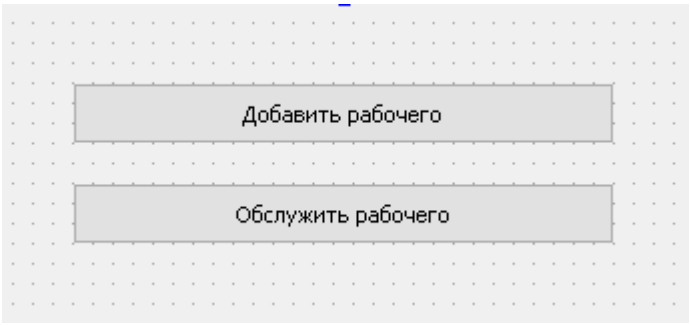


*Рис.3. Окно управлениями события*

### 2.1.4. Окно отображения состояния объектов ПрО


*Рис.4. Основное отображения состояния*

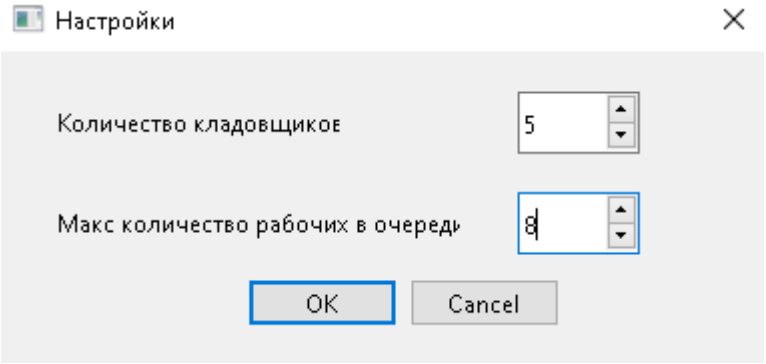


*Рис.5. Пример выбора параметров в окне параметров*

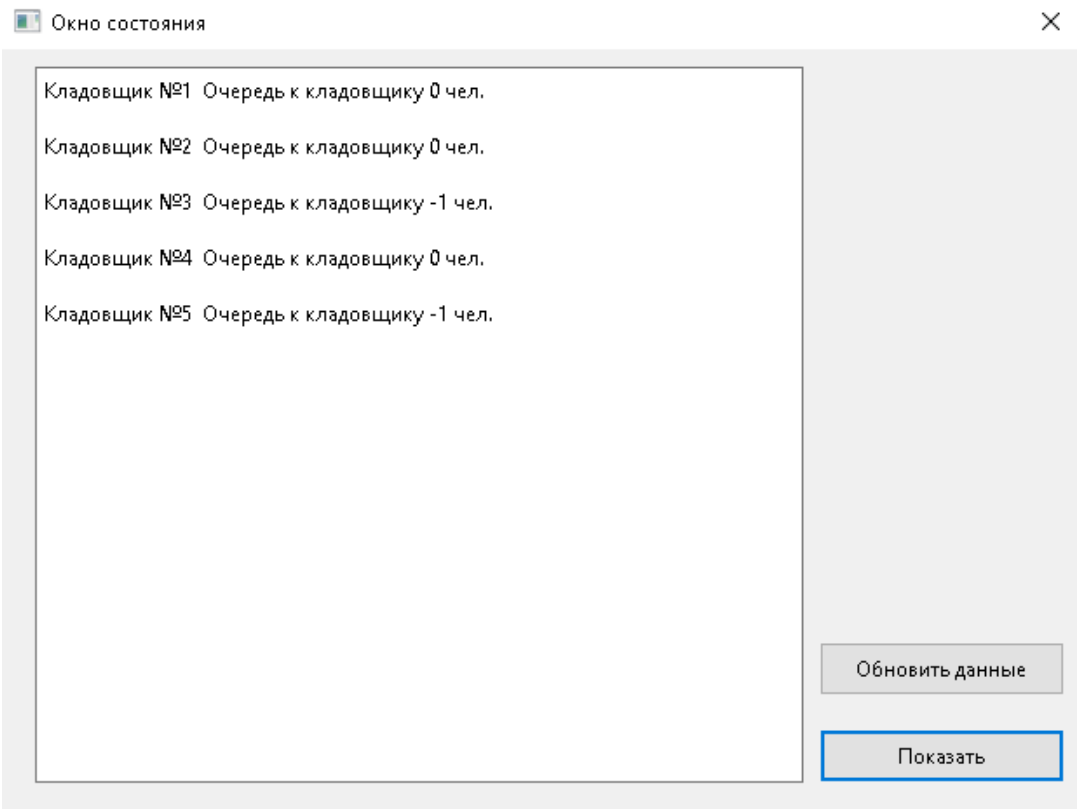


*Рис.6. Пример отображения состояний при добавлении нескольких рабочих при свободных кладовщиках*

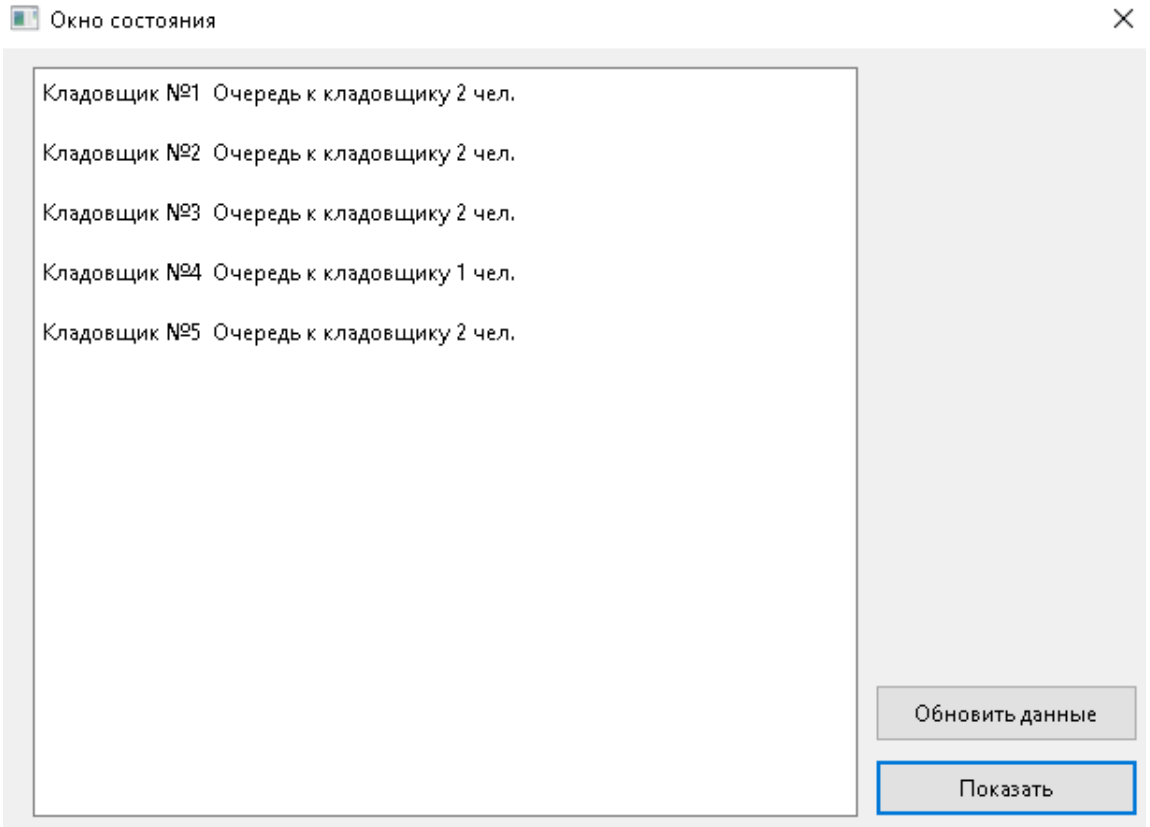


*Рис.7. Пример отображения состояний при добавлении в очередь нескольких рабочих при занятых кладовщиках*

### 2.1.5. Заголовочные файлы интерфейсных классов

**1) application.h**

**// Поля**

TCommunicator *comm;

TInterface *interface;

**// Прототипы методов**

TApplication(int, char**);

TInterface* get_interface();

void fromCommunicator(QByteArray);

void toCommunicator(QString);

**2) interface.h**

**// Поля**

QString response_to_request;

**// Прототипы методов**

TInterface(QWidget *parent = 0);

~TInterface();

void formRequest(int, QString informationFromSettings = nullptr);

QString get_response();

void answer(QString);

void request(QString);

**3) control.h**

**// Поля**

TInterface *interface;

Ui::Control *ui;

**// Прототипы методов**

explicit Control(QWidget *parent = nullptr);

explicit Control(TInterface*, QWidget *parent = nullptr);

~Control();

void on_clearButton_clicked();

void on_add_worker_clicked();

void on_delete_worker_clicked();

**4) mainwindow.h**

**// Поля**

TInterface *interface;

Ui::MainWindow *ui;

// **Прототипы методов**

MainWindow(QWidget *parent = nullptr);

MainWindow(TInterface*, QWidget *parent = nullptr);

~MainWindow();

void on_controlButton_clicked();

void on_statusButton_clicked();

void on_settingsButton_clicked();

**5) status.h**

// **Поля**

TInterface *interface;

Ui::Settings *ui;

// **Прототипы методов**

explicit Status(QWidget *parent = nullptr);

explicit Status(TInterface*, QWidget *parent = nullptr);

~Status();

void on_update_clicked();

void on_show_clicked();

**6) settings.h**

// **Поля**

TInterface *interface;

Ui::Settings *ui;

// **Прототипы методов**

void initValuesIfTheyAre();

explicit Settings(QWidget *parent = nullptr);

explicit Settings(TInterface*, QWidget *parent = nullptr);

~Settings();

void on_buttonBox_accepted();

## 2.2. Диаграмма классов

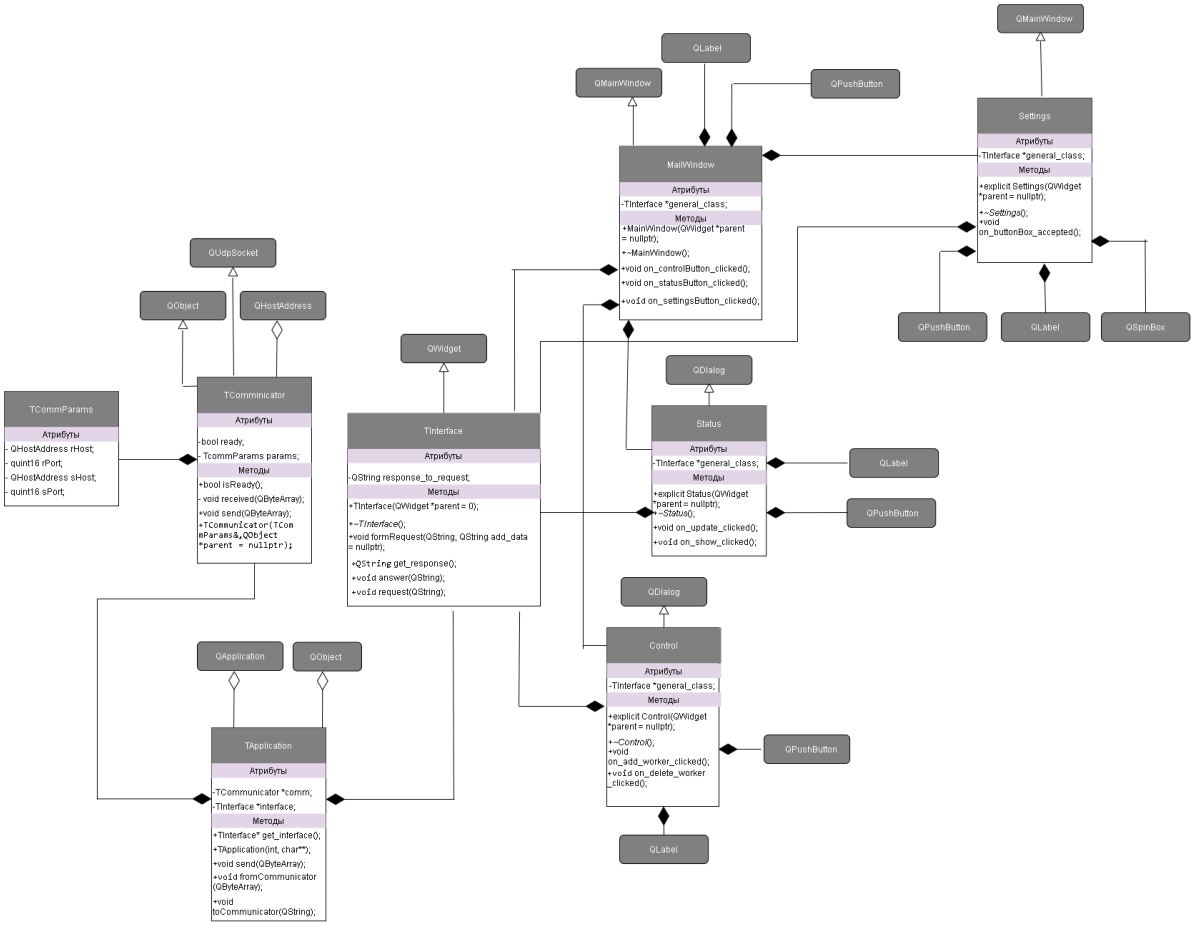


*Рис.8. Диаграмма классов приложения “Клиент”*

## 2.3. Диаграмма объектов

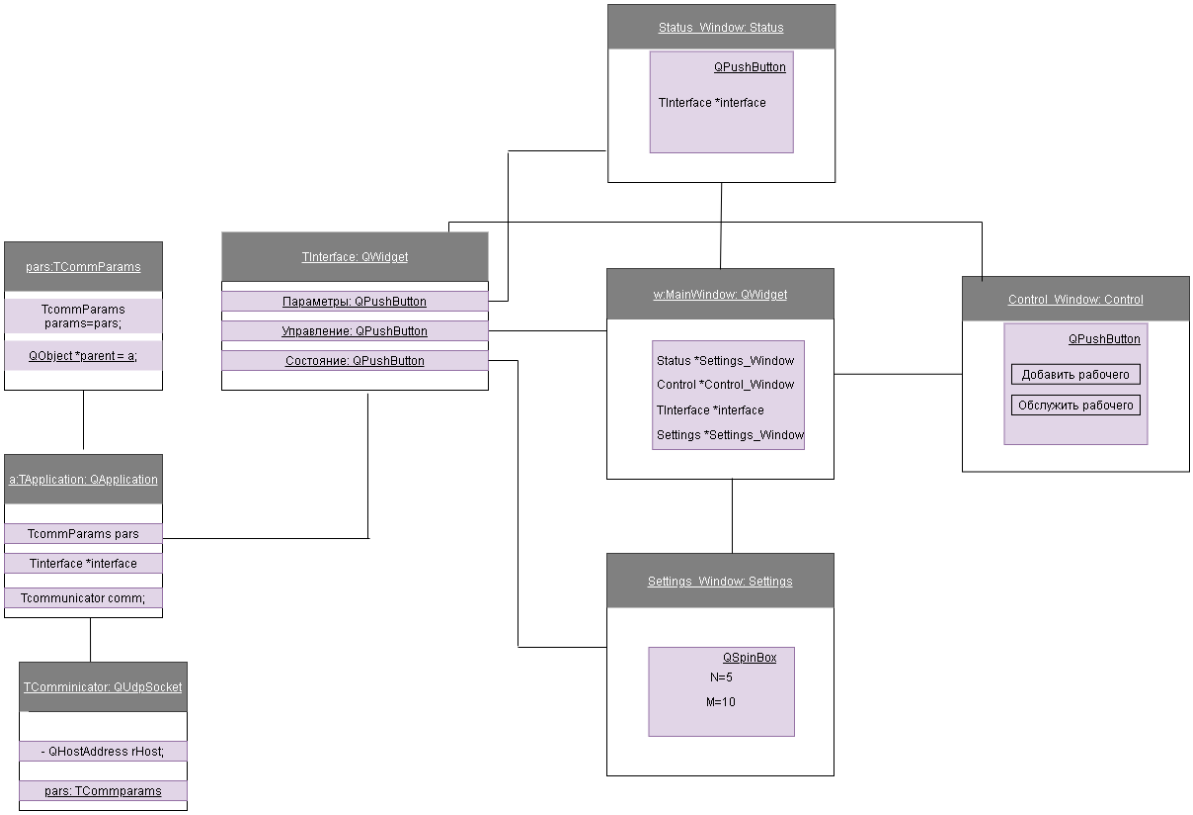


*Рис.9. Диаграмма объектов приложения "Клиент"*

# 3. ПРИЛОЖЕНИЕ «МОДЕЛЬ»

## 3.1. Модель «сущность-связь» ПрО

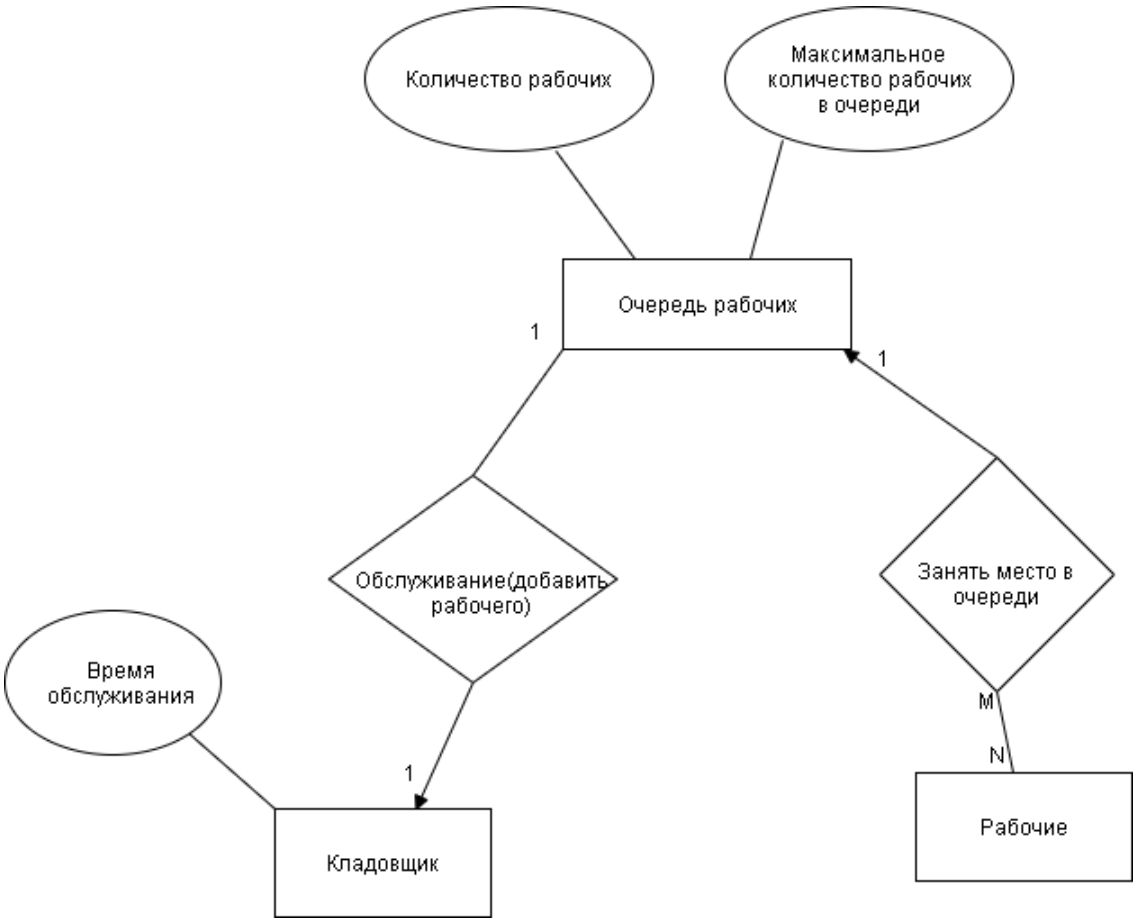


*Рис.10. Диаграмма сущность-связь Про*

### 3.2. Перечень событий, изменяющих состояние ПрО

Представленная программная система имеет следующие события, изменяющие ПрО:

**1)** Рабочий добавлен на обслуживание к кладовщику

**2)** Рабочий добавлен в очередь

**3)** Рабочий покинул обслуживание

**4)** Все кладовщики прекратили обслуживание рабочих

**5)** Изменение количества кладовщиков и количества рабочих в очереди

## 3.3. Диаграмма классов

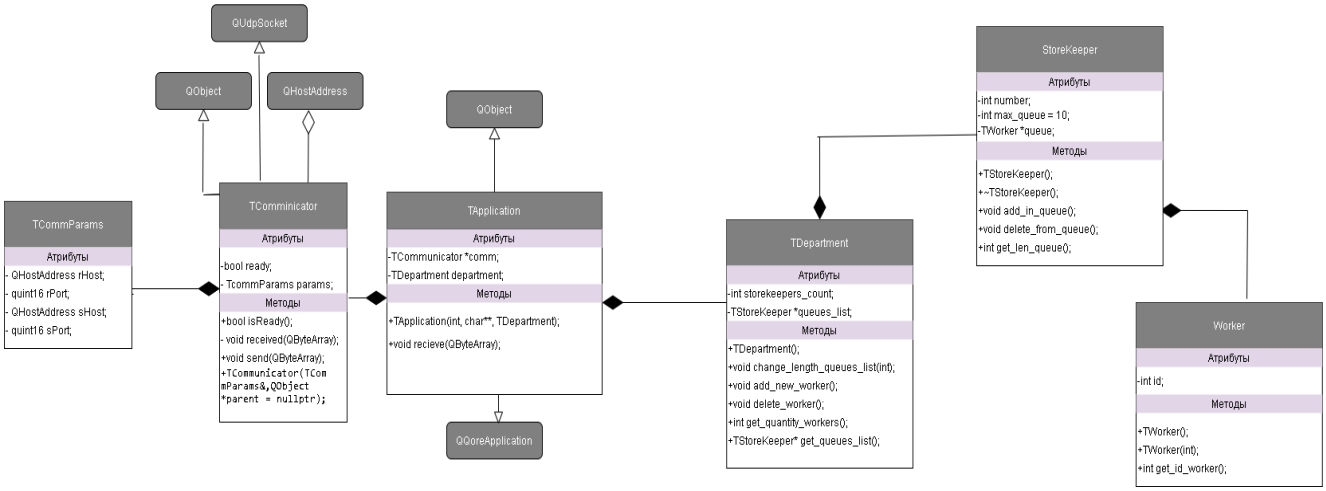


*Рис.11. Диаграмма классов приложения "Сервер"*

### 3.3.1. Логическое описание полей классов

Класс TApplication

| Поле | Описание |
|---|---|
| TCommunicator *comm; | область видимости - private.<br>Объект класса Tcommunicator, область видимости - private |
| TDepartment department; | объект класса TDepartment, область видимости private |

*Таблица 1. Класс TApplication*

Класс TDepartment

| Поле | Описание |
|---|---|
| int storekeepers_count; | область видимости - private.<br>Переменная, которая хранит количество кладовщиков |
| TStoreKeeper *queues_list; | объект класса TStoreKeeper, область видимости private |

*Таблица 2. Класс TDepartment*

Класс TStoreKeeper

| Поле | Описание |
|---|---|
| int max_queue = 10; | область видимости - private.<br>Переменная, которая максимальное количество рабочих в очереди |
| TWorker *queue; | область видимости - private.<br>Объект класса TWorker |

*Таблица 3. Класс TStoreKeeper*

Класс TWorker

| Поле | Описание |
|---|---|
| Int id; | область видимости - private.<br>Переменная, которая отвечает за номер рабочего в очереди |

*Таблица 4. Класс TWorker*

### 3.3.2. Логическое описание методов классов

Класс TApplication

| Поле | Описание |
|---|---|
| TApplication(int, char**, TDepartment); | область видимости - public.<br>Конструктор класса |
| void recieve(QByteArray); | Тип формального параметра - QByteArray, область видимости public. Метод обрабатывает сообщение с запросом от клиентской части распределенного приложения. |

*Таблица 5. Класс TApplication*

Класс TDepartment

| Поле | Описание |
|---|---|
| TDepartment(); | область видимости - public.<br>Конструктор класса |
| void change_length_queues_list(int); | Тип формального параметра - int, область видимости public. Метод изменяет количество кладовщиков |
| void add_new_worker(); | Формальных параметров нет, область видимости public. Метод добавляет рабочего |
| void delete_worker(); | Формальных параметров нет, область видимости public. Метод удаляет рабочего |
| TStoreKeeper* get_queues_list(); | Формальных параметров нет, область видимости public. Вычисляет количество рабочих в очереди |
| int get_quantity_workers(); | Формальных параметров нет, тип возвращаемого значения - int, область видимости public. Вычисляет количество рабочих |

*Таблица 6. Класс TDepartment*

Класс TStoreKeeper

| Поле | Описание |
|---|---|
| TStoreKeeper(); | область видимости - public.<br>Конструктор класса |
| ~TStoreKeeper(); | область видимости - public.<br>Деструктор класса |
| void add_in_queue(); | Формальных параметров нет, область видимости public. Метод добавляет рабочего в очередь |
| void delete_from_queue(); | Формальных параметров нет, область видимости public. Метод удаляет рабочего из очереди |
| int get_len_queue(); | Формальных параметров нет, тип возвращаемого значения - int, область видимости public. Вычисляет длину очереди |

*Таблица 7. Класс TStoreKeeper*

## Класс TWorker

| Поле | Описание |
|---|---|
| TWorker(int); | область видимости - public.<br>Конструктор класса |
| int get_id_worker(); | Формальных параметров нет, тип возвращаемого значения - int, область видимости public. Вычисляет номер рабочего в очереди |

*Таблица 8. Класс TWorker*

### 3.3.3. Заголовочные файлы классов

**1) application.h**

**// Поля**

```
TCommunicator *comm;

TDepartment department;
```

**// Прототипы методов**

```
TApplication(int, char**, TDepartment);

void recieve(QByteArray);
```

**2) department.h**

**// Поля**

```
 int storekeepers_count;
 TStoreKeeper *queues_list;
```

**// Прототипы методов**

```
  TDepartment();
  void change_length_queues_list(int);
  void add_new_worker();
  void delete_worker();
  int get_quantity_workers();
 TStoreKeeper* get_queues_list();
```

**3) storekeeper.h**

**// Поля**

```
int number;
   int max_queue = 10;
   TWorker *queue;
```

**// Прототипы методов**

```
TStoreKeeper();
~TStoreKeeper();
void add_in_queue();
  void delete_from_queue();
  int get_len_queue();
```

**4) worker.h**

**// Поля**

```
int id;
```

**// Прототипы методов**

```
TWorker();
TWorker(int);
int get_id_worker();
```

## 3.4. Диаграмма объектов

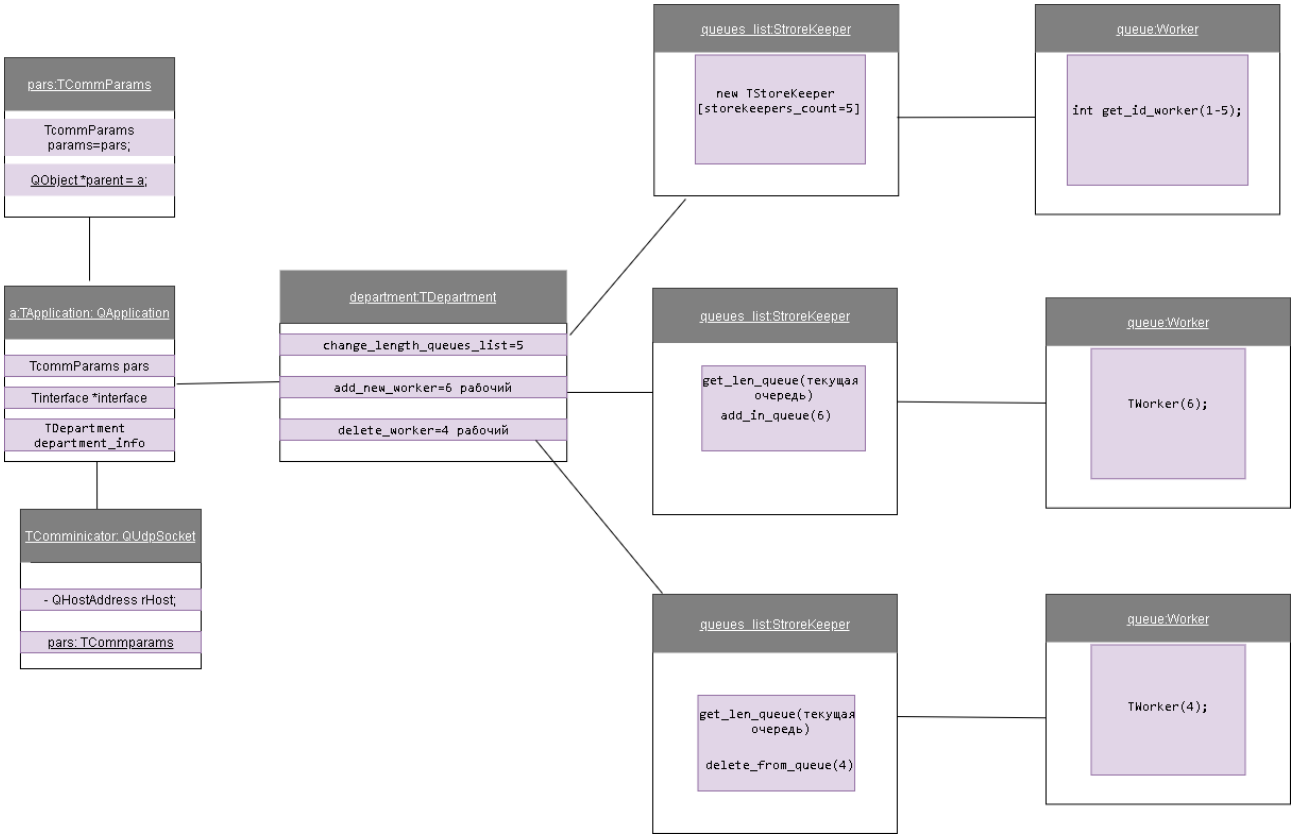


*Рис.12. Диаграмма объектов приложения "Сервер"*

### 3.5. Диаграммы последовательностей обработки каждого типа событий от приложения «Интерфейс»

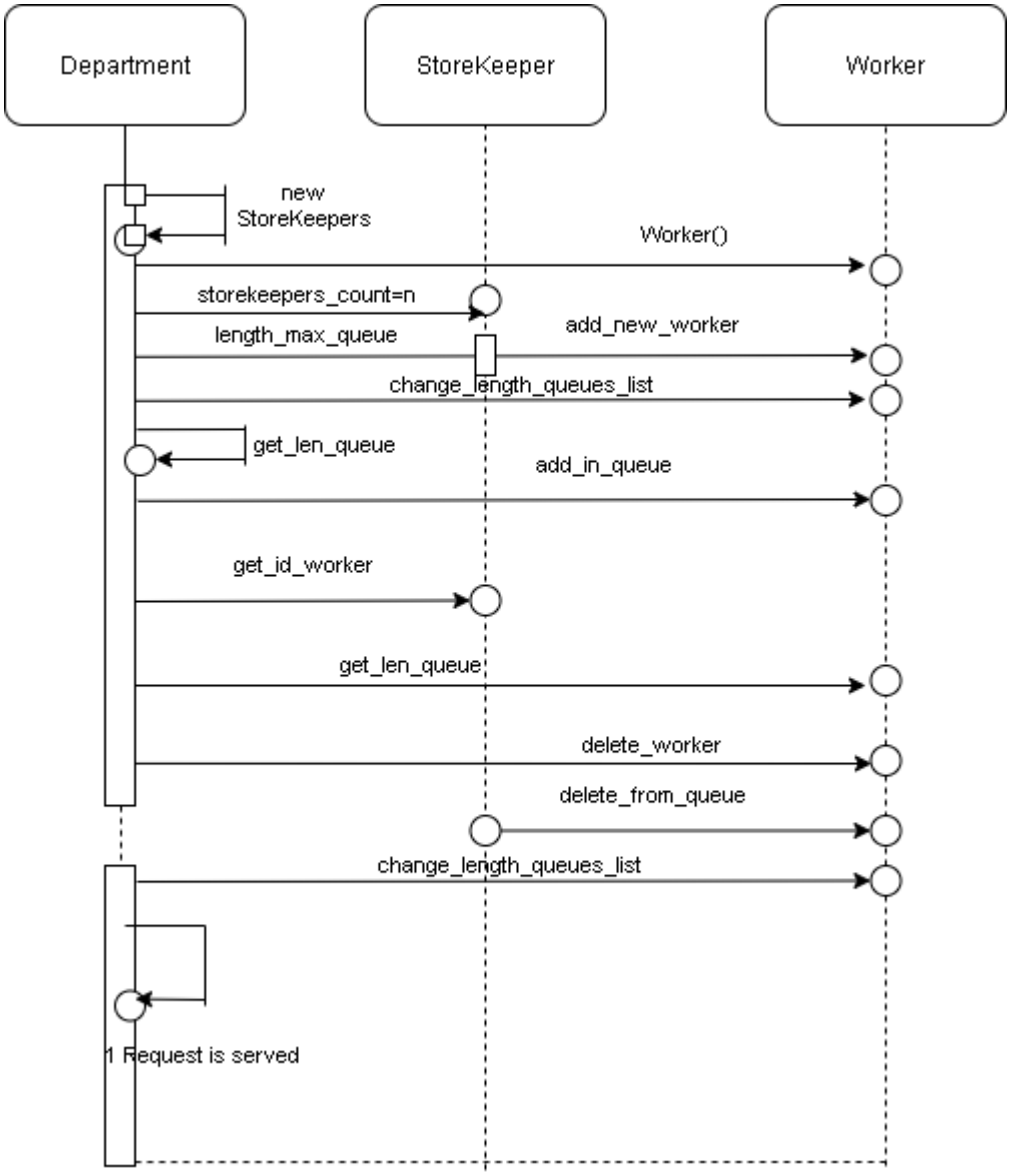


*Рис.13. Диаграмма последовательностей*

# 4. ПЕРЕЧЕНЬ ТИПОВ И СТРУКТУРЫ СООБЩЕНИЙ

В следующем перечислении содержится перечень типов возможных сообщений от сервера и клиента:

```
enum messages
{
   CHANGE_SETTINGS_REQUEST = 1,
   CHANGE_SETTINGS_ANSWER,
   ADD_WORKER_REQUEST,
   ADD_WORKER_ANSWER,
   DELETE_WORKER_REQUEST,
   DELETE_WORKER_ANSWER,
   GET_DATA_REQUEST,
   GET_DATA_ANSWER,
};
```

## 4.1. Перечень типов и структуры сообщений от клиента к серверу

При нажатии пользователем кнопки “Добавить рабочего” или “Удалить рабочего” в подсистему “Сервер” передаются сообщения: ADD_WORKER_REQUEST и DELETE_WORKER_REQUEST.

Из окна настроек при нажатии на кнопку “Ок” на сервер отправляются данные о количестве кладовщиков и максимальном количестве рабочих в очережи с помощью сообщений: CHANGE_SETTINGS_REQUEST = 1, при ходят при помощи CHANGE_SETTINGS_ANSWER.

При нажатии на кнопку “Обновить данные” в окне состояния информация передается с помощью сообщения: GET_DATA_REQUEST.

## 4.2. Перечень типов и структуры сообщений от сервера к клиенту

При нажатии пользователем кнопки “Обновить данные” окне состояния в подсистему “Сервер” передаётся сигнал об отправке результата работы клиента, после чего из подсистемы “Сервер” в подсистему “Клиент ” передаётся сообщение GET_DATA_ANSWER и последовательность символов типа string, содержащих соответственно количество обслуженных и необслуженных

рабочих. Далее при нажатии на кнопку “Показать” данные полученного сообщения выводятся в окно отображения состояния системы.

Также при нажатии на кнопки “Добавить рабочего” или “Удалить рабочего” с сервера клиенту передаются сообщения: ADD_WORKER_ANSWER и DELETE_WORKER_ANSWER, а при изменении параметров передается сообщение - CHANGE_SETTINGS_ANSWER.

# 5. ВЫВОД

В ходе данной курсовой работы было создано клиент-серверное приложение, выполняющее функцию инструментального отделения сборочного цеха. Реализована возможность добавлять рабочих с равной вероятностью, обслуживать или ставить их в очередь, когда нет свободных кладовщиков. Максимальное возможное количество рабочих в очереди ограничено и равно 10.

Программа была реализована в объектно-ориентированной парадигме с использованием языка C++ и QT. Объектная модель описана набором диаграмм в терминах UML. Программная система взаимодействующих приложений обменивается информацией по UDP протоколу. Разработанная программная система отвечает заданным требованиям.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Егоров С.С. Методические указания к выполнению курсовой работы по курсу «Объектно-ориентированное программирование». 24 с.

2. Егоров С.С. Лекции по курсу «Объектно-ориентированное программирование».

3. Программирование на языке C++ в среде Qt Creator: / Е. Р. Алексеев, Г. Г. Злобин, Д. А. Костюк,О. В. Чеснокова, А. С. Чмыхало — М. : ALT Linux, 2015. — 448 с. : ил. — (Библиотека ALT Linux).

4. М. Шлее «Qt 4.8. Профессиональное программирование на C++» БХВ-Петербург, 2012 год, 912 стр.

5. Фаулер М. UML. Основы, 3-е издание. – Пер. с англ. – СПб: Символ-Плюс, 2004. – 192 с.

6. Проектирование на UML. Сборник задач по проектированию программных систем. 2-е. изд. – Екатеринбург.: Издательские решения, 2017. – 240 с.