**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 15 страниц. | | |
| Дата выдачи задания: 10.02.2021 | | |
| Дата сдачи работы: 00.06.2021 | | |
| Дата защиты работы: 00.06.2021 | | |
| 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 С++ 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 6](#_Toc73388126)

[1. Постановка задачи 7](#_Toc73388127)

[1.1. Описание предметной области 9](#_Toc73388128)

[1.2. Перечень библиотечных конструкторских классов, использованных в проекте 9](#_Toc73388129)

[2. Приложение «Интерфейс» 12](#_Toc73388130)

[2.1. Графическое представление интерфейсных окон 12](#_Toc73388131)

[2.1.1. Основное окно 12](#_Toc73388132)

[2.1.2. Окно параметров ПрО 12](#_Toc73388133)

[2.1.3. Окно управления событиями ПрО 12](#_Toc73388134)

[2.1.4. Окно отображения состояния объектов ПрО 13](#_Toc73388135)

[2.1.5. Заголовочные файлы интерфейсных классов 14](#_Toc73388136)

[2.2. Диаграмма классов 16](#_Toc73388137)

[2.3. Диаграмма объектов 17](#_Toc73388138)

[3. Приложение «Модель» 18](#_Toc73388139)

[3.1. Модель «сущность-связь» ПрО 18](#_Toc73388140)

[3.2. Перечень событий, изменяющих состояние ПрО 19](#_Toc73388141)

[3.3. Диаграмма классов 20](#_Toc73388142)

[3.3.1. Логическое описание полей классов 21](#_Toc73388143)

[3.3.2. Логическое описание методов классов 22](#_Toc73388144)

[3.3.3. Заголовочные файлы классов 24](#_Toc73388145)

[3.4. Диаграмма объектов 25](#_Toc73388146)

[3.5. Диаграммы последовательностей обработки каждого типа событий от приложения «Интерфейс» 26](#_Toc73388147)

[4. Перечень типов и структуры сообщений 27](#_Toc73388148)

[4.1. Перечень типов и структуры сообщений от клиента к серверу 27](#_Toc73388149)

[4.2. Перечень типов и структуры сообщений от сервера к клиенту 27](#_Toc73388150)

[5. ВЫВОД 29](#_Toc73388151)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 30](#_Toc73388152)

# 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.

# 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.

1. **QWidget**

Класс QWidget является базовым для всех объектов пользовательского интерфейса.

1. **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.

1. **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»

1. **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.

1. **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.

1. **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.

1. **QDialog**

Класс QDialog является базовым классом для диалоговых окон.

1. **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.

1. **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.

1. **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»

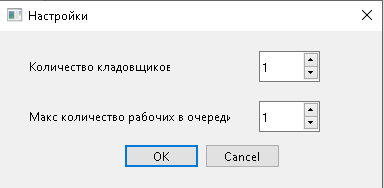
## 2.1. Графическое представление интерфейсных окон

### 2.1.1. Основное окно



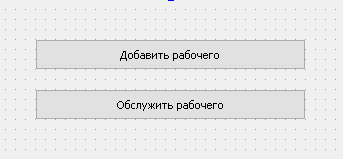
*Рис.1. Основное окно программы*

### 2.1.2. Окно параметров ПрО



*Рис.2. Окно параметров программы*

### 2.1.3. Окно управления событиями ПрО

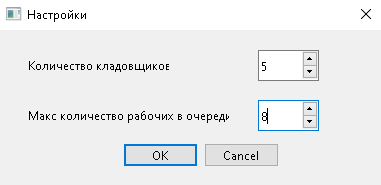


*Рис.3. Окно управлениями события*

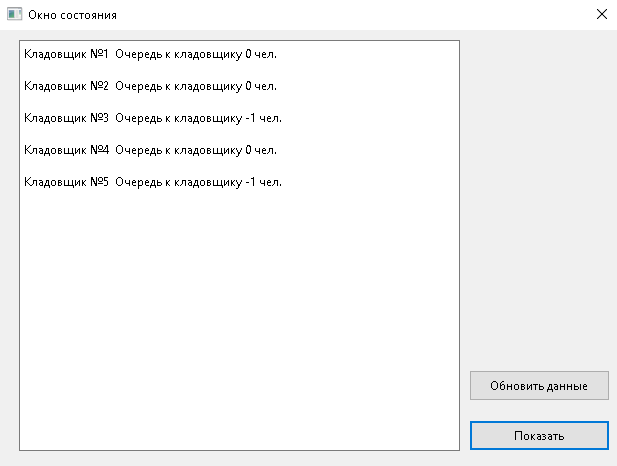
### 2.1.4. Окно отображения состояния объектов ПрО



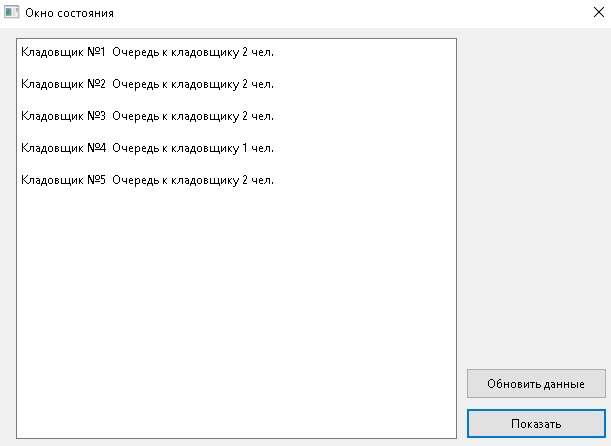
*Рис.4. Основное отображения состояния*



*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);

1. **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);

1. **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();

1. **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();

1. **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();

1. **settings.h**

**// Поля**

TInterface \*interface;

Ui::Settings \*ui;

**// Прототипы методов**

void initValuesIfTheyAre();

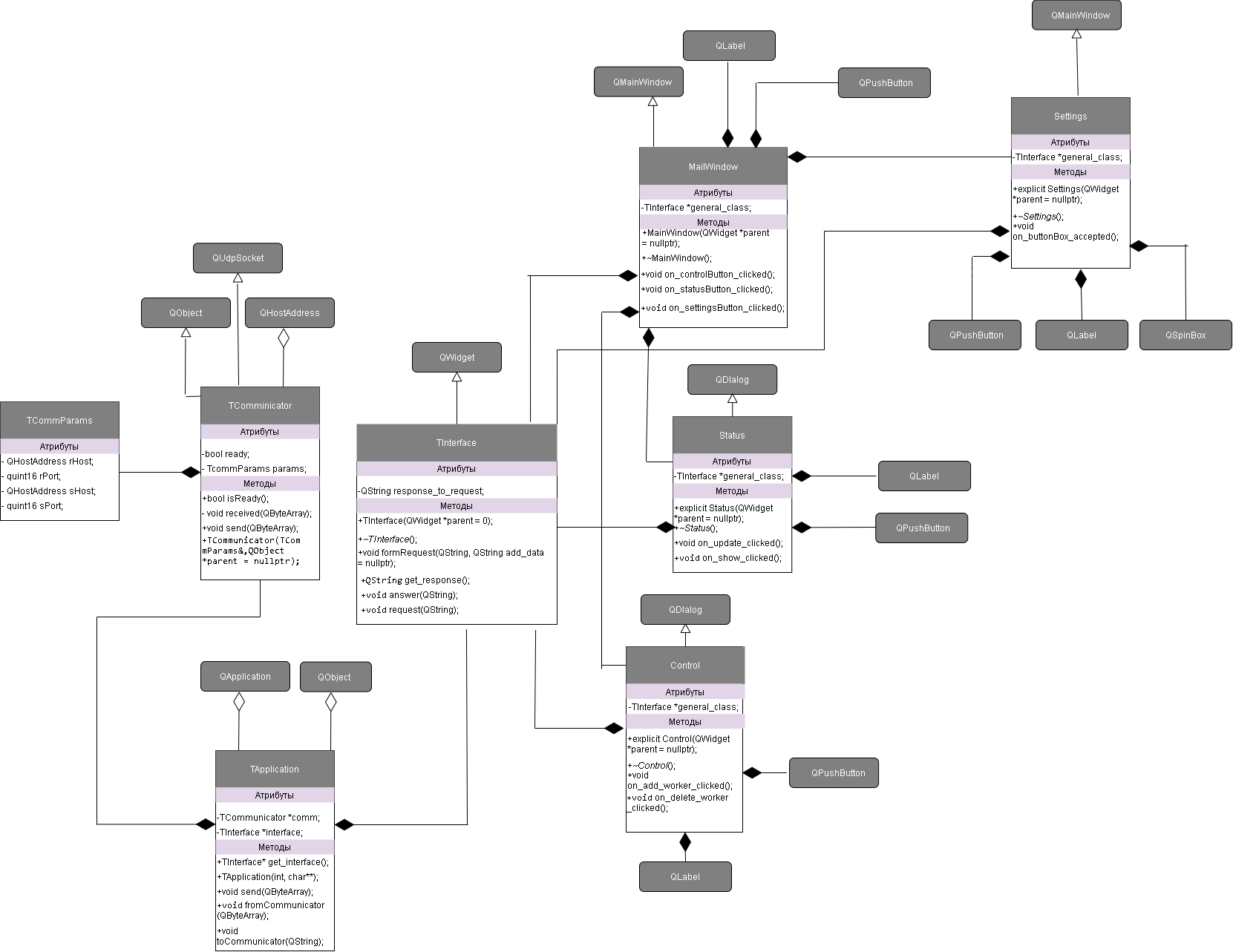
explicit Settings(QWidget \*parent = nullptr);

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

~Settings();

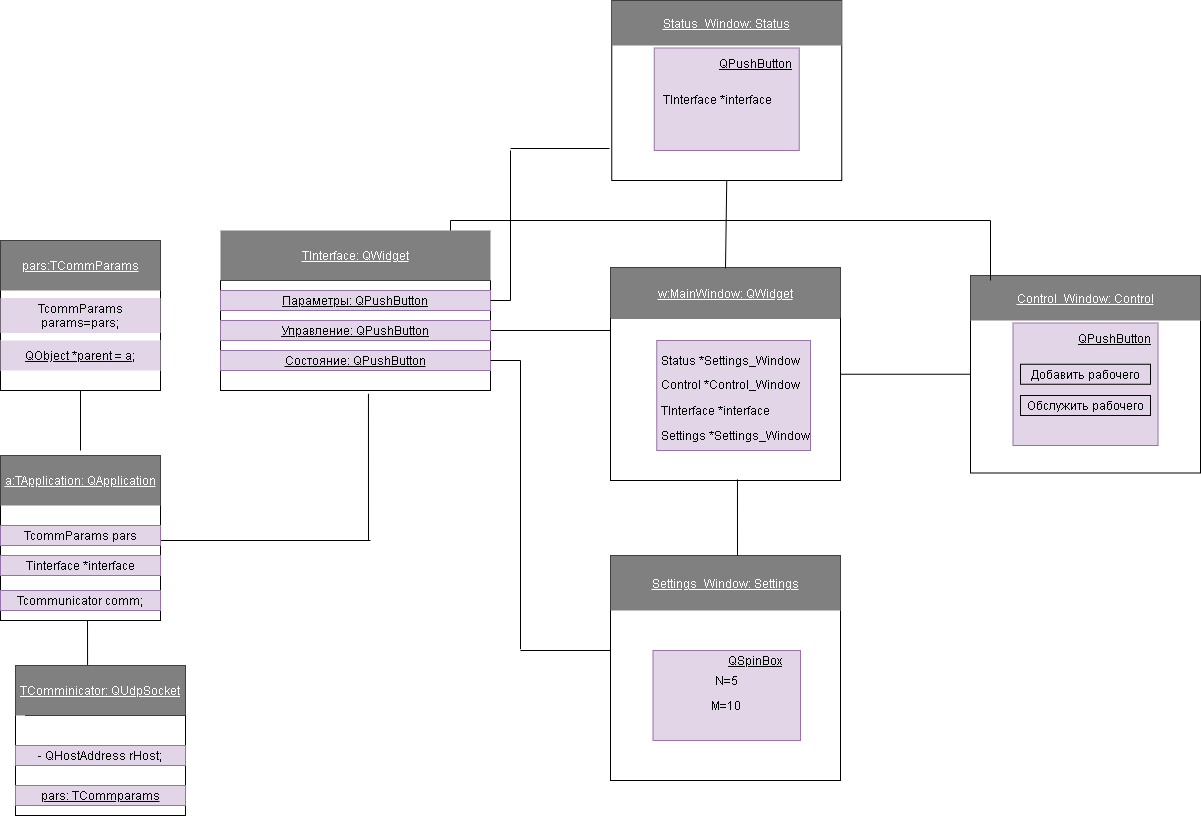
void on\_buttonBox\_accepted();

## 2.2. Диаграмма классов



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

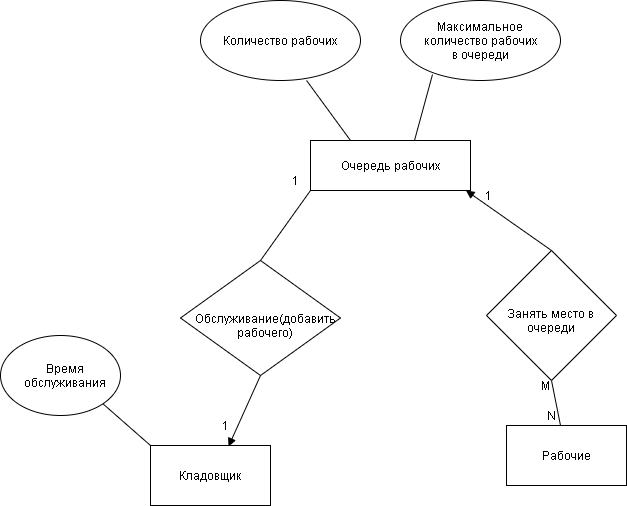
## 2.3. Диаграмма объектов



*Рис.9. Диаграмма объектов приложения “Клиент”*

# 3. Приложение «Модель»

## 3.1. Модель «сущность-связь» ПрО



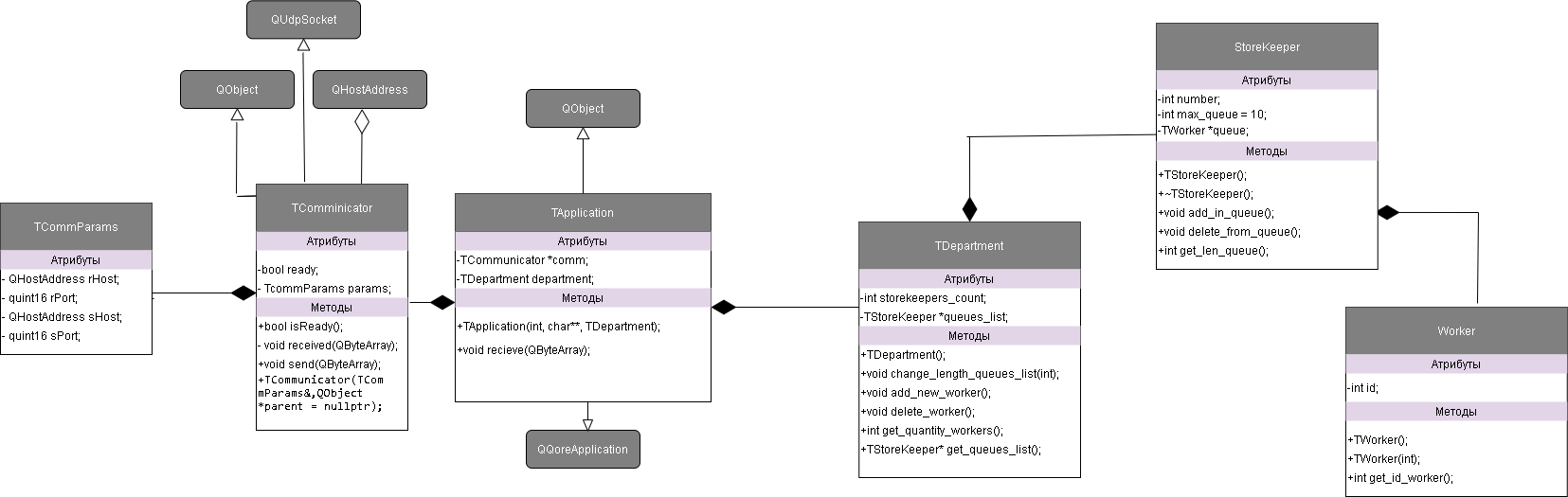
*Рис.10. Диаграмма сущность-связь Про*

## 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

## 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.  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.  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.  Переменная, которая максимальное количество рабочих в очереди |
| TWorker \*queue; | область видимости - private.  Объект класса TWorker |

*Таблица 3. Класс TStoreKeeper*

Класс TWorker

|  |  |
| --- | --- |
| Поле | Описание |
| Int id; | область видимости - private.  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.  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.  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.  Конструктор класса |
| ~TStoreKeeper(); | область видимости - public.  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.  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);

1. **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();

1. **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();

1. **worker.h**

**// Поля**

int id;

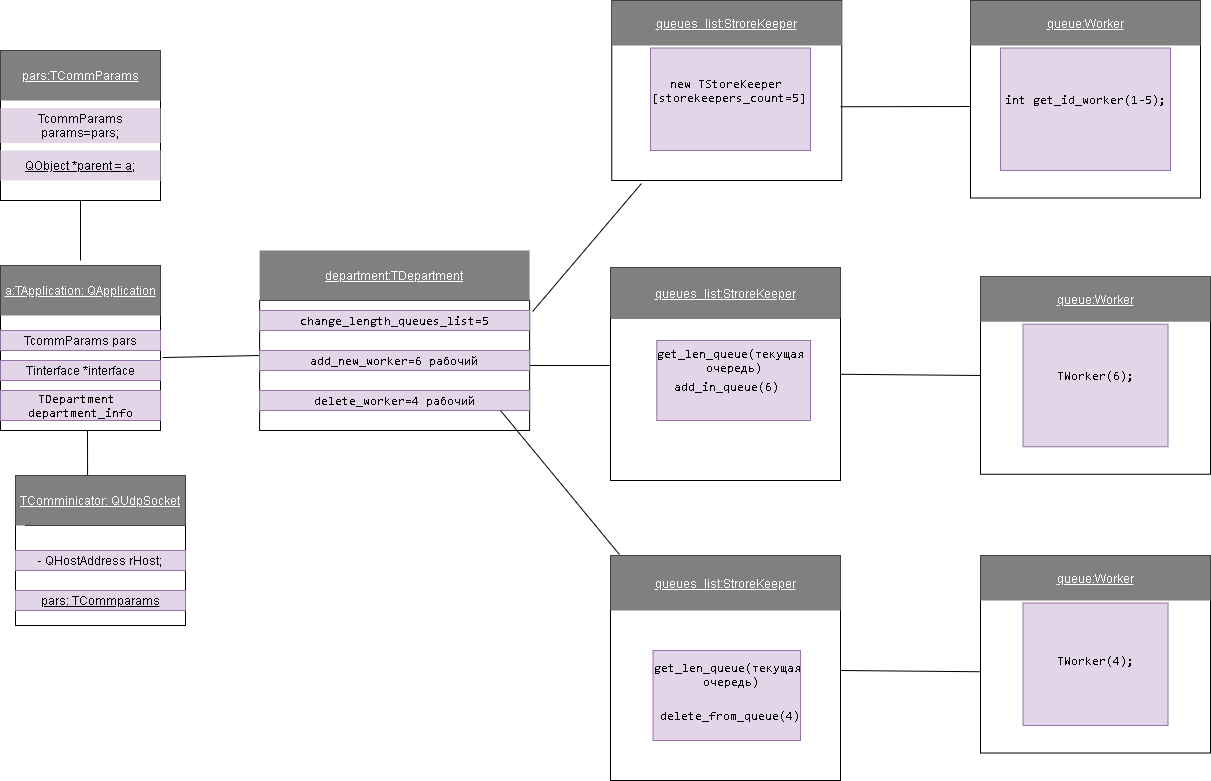
**// Прототипы методов**

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»

# 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 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 С++ 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 M. 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.