**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

КУРСОВАЯ РАБОТА

**по дисциплине «Объектно-Ориентированное Программирование»**

**Тема: Создание объектно-ориентированной программы**

**Вариант 13**

|  |  |  |
| --- | --- | --- |
| Студенты группы № 2372 |  | Полуянов В.Н., Юрин А.А. |
| Преподаватель |  | Егоров С.С. |

Санкт-Петербург

2023

**Аннотация**

Курсовой проект заключается в разработке и реализации объектно-ориентированной программы на языке C++, реализующей приложение, описанное в постановке задачи. В ходе работы были изучены принципы Peer-to-peer коммуникации, принципы ООП, включая инкапсуляцию, наследование, полиморфизм и абстрагирование. В результате было написано полноценное приложение, реализующее игру с двумя игроками, правила которой описаны в постановке задачи.

**содержание**

|  |  |  |
| --- | --- | --- |
|  | Введение | 4 |
| 1. | Объектная модель | 5 |
| 1.1. | Диаграмма «сущность-связь» | 5 |
| 1.2. | Диаграмма классов | 6 |
| 1.3. | Перечень библиотечных конструкторов классов | 10 |
| 1.5. | Схема соединений «сигнал-слот» | 11 |
| 2. | Подсистема «Интерфейс» | 12 |
| 2.1. | Основное окно | 12 |
| 2.2. | Окно параметров | 12 |
| 2.3. | Окно управления событиями | 13 |
| 2.4. | Окно отображения состояния объектов | 13 |
| 2.5. | Заголовочные файлы интерфейсных классов | 14 |
| 3. | Подсистема «Модель» | 16 |
| 3.1 | Перечень событий, изменяющих состояние модели | 16 |
| 3.2 | Заголовочные файлы классов модели | 16 |
|  | Контрольный пример работы реализованного приложения | 18 |
|  | Заключение | 22 |

**введение**

**Постановка задачи**

Играют два игрока. Игра имеет следующие правила. Изначально перед каждым из них находится общий набор из N (15) закрытых ящиков с деньгами. Сумма денег в каждом ящике – случайная равномерно распределенная на интервале [S1, S2] (0-10) величина. Игроки поочередно открывают по одному выбранному ящику, узнавая сумму в нем хранящуюся. Повторно ящики не используются. При этом игрок до хода другого может взять деньги и закончить свои ходы или продолжить, передав ход сопернику, который видит открытый ящик, но не знает о взятой сумме. Всего может быть сделано по K (5) ходов каждым игроком. Выиграл тот, у которого денег оказалось больше.

Дополнительные требования:

- игра сетевая, т.е. приложение отвечает только за 1 игрока,

- для каждого игрока приложение одно и то же,

- первым ход делает игрок, приложение которого запущено первым,

- возможна работа на одном компьютере.

**Описание предметной области**

В ходе анализа условий поставленной задачи были выделены следующие объекты предметной области: Игра, Коробка, Игрок. Игра является контейнером, хранящим информацию как о себе (текущее состояние, является она хостом или клиентом и т.п.), так и о её составляющих (хранение экземпляров коробок и игроков).

Коробка является кнопкой, размер которой устанавливается постоянным, а положение в интерфейсе Игры определяется Игрой. Может быть открытой или закрытой, хранит информацию о количестве денег внутри.

**Обоснование выбора языка программирование и объектных библиотек конструкторских классов**

В курсовой работе использованы рекомендованные для применения язык программирования C++ и объектные библиотеки конструкторских классов Qt.

**Объектная модель**

1. **Диаграмма «сущность-связь» (рис. 1)**

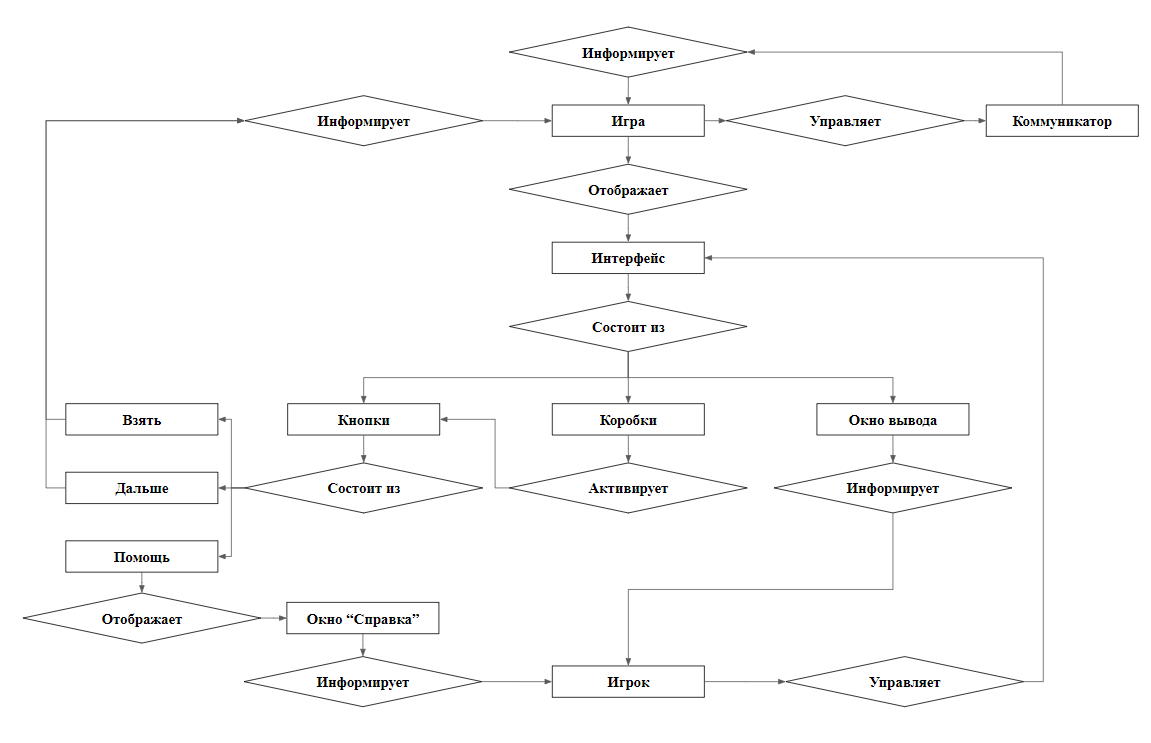


Рисунок 1. Модель "сущность-связь"

1. **Диаграмма классов (рис. 2)**

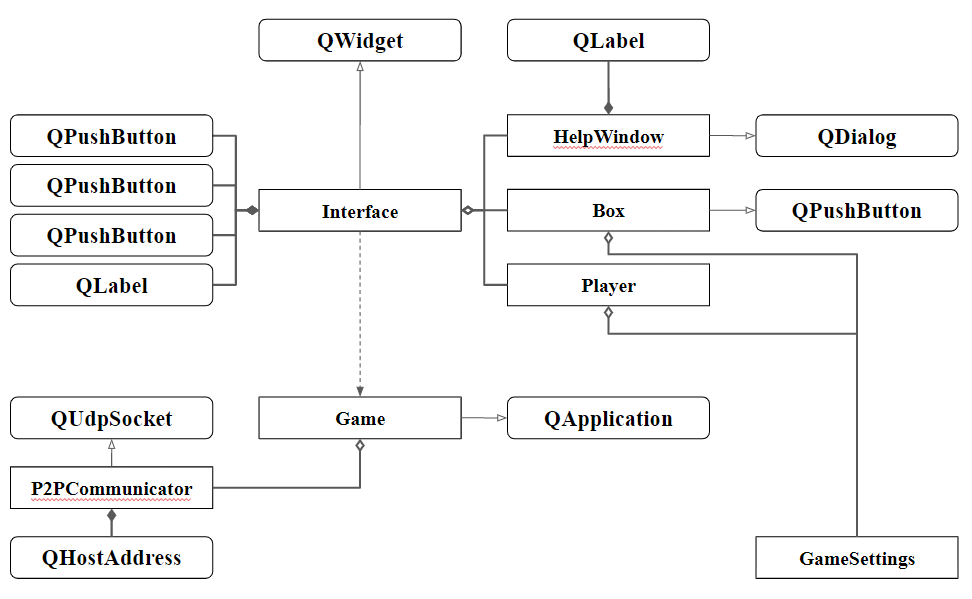


Рисунок 2. Диаграмма классов

Более подробные сведения о диаграмме классов представлены в таблицах 1-4 (табл. 1, 2, 3, 4):

Таблица 1. Описание классов

|  |  |  |
| --- | --- | --- |
| Наименование | Идентификатор | Назначение |
| Приложение Игра | Game | Представляет тип «Игра». Отвечает за взаимодействие игроков |
|  | Наследуется от: | QApplication |
| Наследники: | - |
| P2P Коммуникатор | P2PCommunicator | Предоставляет peer-to-peer соединение. Позволяет создавать связь типа «хост-клиент». Отвечает за установку соединения и обмен сообщениями |
|  | Наследуется от: | QUdpSocket |
| Наследники: | - |
| Интерфейс | Interface | Представляет собой виджет Qt. Отвечает за отображение информации об игре, отправляет и отвечает на запросы Игры |
|  | Наследуется от: | QWidget |
|  | Наследники: | - |
| Окно «Справка» | HelpWindow | Представляет собой диалог Qt. Отвечает за отображение текста-справки по игре и правилам |
|  | Наследуется от: | QDialog |
|  | Наследники: | - |
| Коробка с деньгами | Box | Представляет собой кнопку, хранит информацию о количестве денег внутри ящика и текущее состояние (открыт или нет) |
|  | Наследуется от: | QPushButton |
|  | Наследники: | - |
| Игрок | Player | Представляет собой модель Игрок. Хранит информацию о количестве потенциально полученных денег и о количестве оставшихся ходов. |
|  | Наследуется от: | - |
|  | Наследники: | - |

Таблица 2. Описание атрибутов классов

|  |  |  |
| --- | --- | --- |
| Game | | |
| Наименование | Идентификатор | Назначение |
| -Коммуникатор | Communicator | Реализует коммуникацию между игроками |
| -Интерфейс | Interface | Отвечает за GUI |
| P2PCommunicator | | |
| -Размер хост-запроса | hostMsgSize | Размер сообщения при приветствии |
| -Запрос о хосте | hostRequest | Запрос на получение сервера |
| -Ответ о хосте | hostResponse | Ответ от сервера |
| -Параметры соединения | params | Адреса и хосты сервера и клиента |
| -Статус соединения | handshakeStatus | Отражает статус «приветствия» |
| -Статус сокета | isReady | Отражает состояние сокета |
| Interface |  |  |
| -Количество коробок | N | Получает из файла-конфигуратора количество коробок |
| -Массив коробок | boxes | Содержит указатели на каждую из коробок |
| -Игрок 1 | player1 | Хранит экземпляр текущего игрока |
| -Игрок 2 | player2 | Хранит экземпляр второго игрока |
| -Кнопка «Взять» | btnTake | Отвечает за действие «взять деньги и закончить игру» |
| -Кнопка «Пропустить» | btnPass | Отвечает за действие «не брать деньги и пропустить ход» |
| -Кнопка «Справка» | btnHelp | Вызывает окно «Справка» |
| -Окно вывода | output | Прямоугольное окно с информативным текстом |
| HelpWindow |  |  |
| -Окно вывода | output | Текстовое поле |
| Box |  |  |
| -Открыта ли коробка? | opened | Истина если коробка была открыта |
| -Количество денег | value | Хранит количество денег внутри коробки |
| Player |  |  |
| -Количество денег | money | Количество денег из последней коробки |
| -Ходов осталось | movesLeft | Количество оставшихся ходов |

Таблица 3. Описание методов классов

|  |  |
| --- | --- |
| Game | |
| Идентификатор | Назначение |
| +Game | Конструктор, инициализирует P2P-соединение, отображает интерфейс и соединяет сигналы со слотами |
| +handshakeDone | Слот завершения приветствия. Передает сигнал о завершении в интерфейс |
| +from | Сообщает интерфейсу о необходимости ответа |
| +to | Запрашивает отправку сообщения у коммуникатора |
| +gameStarted | Сигнал для игрока-сервера. Вызывает первый ход |
| +startWaiting | Сигнал для игрока-клиента. Вызывает ожидание хода |
| P2PCommunicator |  |
| +P2PCommunicator | Конструктор. Инициализирует соединение. Проверяет готовность |
| +initHandshake | Инициализирует приветствие. Отправляет сообщение-запрос на получение сервера |
| -swapPorts | Обменивает локальный и удалённый порты |
| -verifyOrder | Утверждает порядок сервер-клиент |
| +send | Отправляет сообщение |
| -receive | Принимает сообщение |
| +received | Сигнал о получении сообщения |
| +handshakeDone | Сигнал об окончании приветствия |
| Interface |  |
| +Interface | Конструктор. Генерирует интерфейс, соединяет подокна с нужными действиями |
| +~Interface | Деструктор. Освобождает память от составляющих |
| -enableBoxes | Переводит кнопки-коробки в состояние «Включено» |
| -disableBoxes | Переводит кнопки-коробки в состояние «Выключено» |
| -getMoneyMsg | Возвращает сообщение у вас «N монет» в правильном склонении |
| +startGame | Инициализирует игру для первого игрока |
| +waitForTurn | Инициализирует игру для клиента – ожидает хода сервера |
| +answer | Обрабатывает полученное сообщение от противника |
| -onHelpPressed | Вызывает окно «Справка» |
| -onBoxClicked | Обрабатывает открытие коробки. Передаёт управление кнопкам «взять» и «пропустить» (или заканчивает игру) |
| -onTakeOrPassPressed | Обрабатывает взятие и пропуск хода. Передаёт сообщение противнику |
| +request | Сигнал о появлении запроса |
| HelpWindow |  |
| +HelpWindow | Конструктор. Создаёт окно-справку |
| Box |  |
| +Box | Конструктор. Устанавливает атрибуты коробки |
| +setRandomValue | Установить случайное количество денег в коробке |
| +setValue | Установить определенное количество денег в коробке |
| +setOpen | Переключить состояние коробки |
| +getValue | Получить числовой эквивалент денег в коробке |
| +isOpen | Получить состояние коробки |
| +draw | Отобразить коробку на координатах x, y |
| Player |  |
| +Player | Конструктор. Устанавливает атрибуты игрока |
| +setMoney | Установить определенное количество денег у игрока |
| +setMovesLeft | Установить количество оставшихся ходов |
| +move | Сделать ход. Понизить количество оставшихся ходов на 1 |
| +getMoney | Получить количество денег у игрока |
| +getMovesLeft | Получить количество оставшихся ходов игрока |

Таблица 4. Описание отношений между классами

|  |  |  |
| --- | --- | --- |
| Класс с исходящей стрелкой | Класс со входящей стрелкой | Вид отношения |
| TMainWindow | QWidget | Обобщение |
|  |  |  |

1. **Перечень библиотечных конструкторов классов**

В ходе работы были использованы следующие библиотечные конструкторы:

* QApplication – родительский класс приложения, отвечает за его инициализацию;
* QWidget – для создания составных окон;
* QDialog – для создания всплывающего окна;
* QObject – для создания объектов Qt;
* QLabel – для создания надписей;
* QPushButton – для создания кнопок;
* QUdpSocket – для работы с UDP протоколом;
* QHostInfo – для получения информации о сервере в сети;
* QHostAdress – для представления адресов в сети.

1. **Схема соединений сигнал-слот** **(рис. 3)**

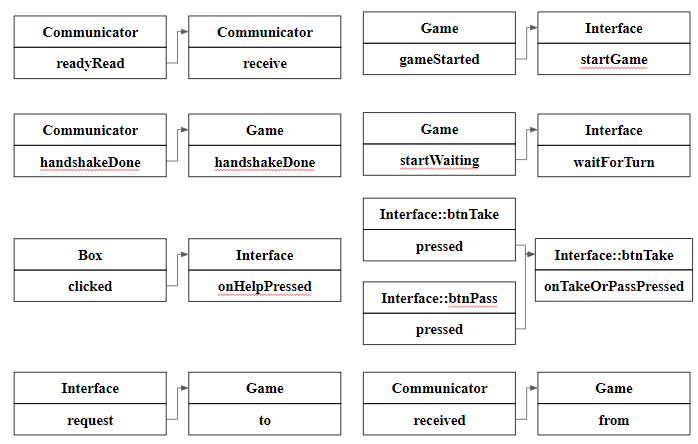


Рисунок 3. Схема соединений сигнал-слот

**Подсистема «Интерфейс»**

1. **Основное окно**

Основное окно (рис. 4) хранит в себе все подокна, которые могут быть скрыты/показаны (кнопки) или включены/выключены (коробки).

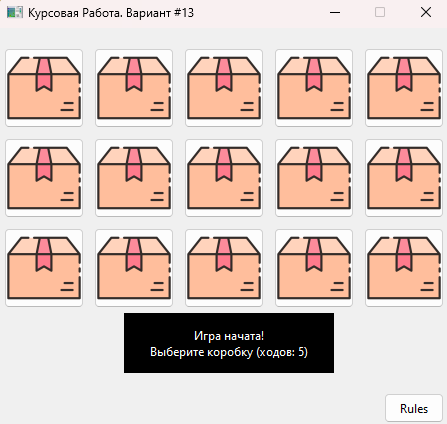


Рисунок 4. Основное окно

1. **Окно «Справка»**

Диалоговое окно «Справка» хранит в себе текст-справку о правилах игры (рис. 5).

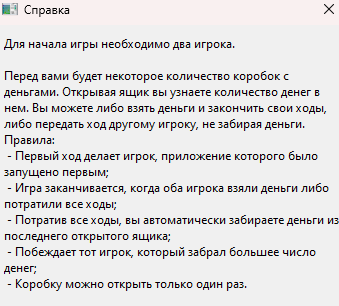


Рисунок 5. Окно "Справка"

1. **Окно «Игра»**

Окно игра содержит в себе подокна: коробки, «взять или пропустить», «справка» и окно вывода (рис. 6). Коробки представляют собой кнопки, координаты которых задаются основным окном. Кнопки «взять» и «пропустить» генерируются основным окном и отображаются при необходимости действия. Кнопка «справка» генерируется основным окном и отображается вне зависимости от состояния игры. Окно вывода генерируется основным окном и обновляется в соответствии с состоянием игры.

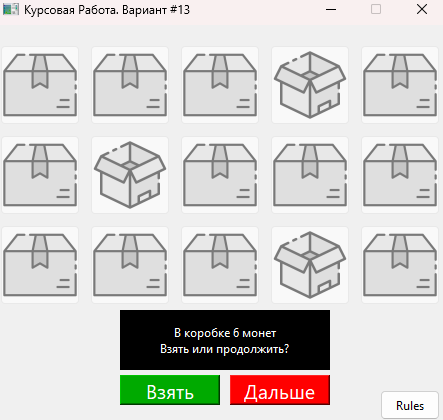


Рисунок 6. Окно "Игра"

1. **Заголовочные файлы интерфейсных классов**

* Класс Interface:

*/\*\*  
 \* @class Interface  
 \* @brief Represents the player's main GUI  
 \*  
 \* Manages the game interface, including boxes, players, and buttons.  
 \* Handles P2P signals of the game  
 \*/*class Interface : public QWidget {  
private:  
 **Q\_OBJECT** *// Boxes* static const int N = GameSettings::boxAmount;  
 Box \*boxes[N];  
  
 *// Players* Player player1;  
 Player player2;  
  
 *// Buttons* QPushButton \*btnTake;  
 QPushButton \*btnPass;  
 QPushButton \*btnHelp;  
  
 *// Labels* QLabel \*output;  
  
public:  
 *// Constructor and destructor* explicit Interface(QWidget\* = nullptr);  
 ~Interface() override;  
  
private:  
 void enableBoxes(); *// Set the boxes clickable* void disableBoxes(); *// Disable the boxes* QString getMoneyMsg(int); *// Get the correct declination of the number of coins*public **slots**:  
 void startGame(); *// P1 makes the first move* void waitForTurn(); *// P2 waits for the P1's first move* void answer(QString); *// Make next move*private **slots**:  
 void onHelpPressed(); *// Show the help window* void onBoxClicked(Box\*); *// Handle box click, go to Take/Pass part* void onTakeOrPassPressed(); *// Handle Take/Pass click***signals**:  
 void request(QString); *// Send the message to the P2*};

* Класс Box:

*/\*\*  
 \* @class Box  
 \* @brief Represents a box in the game  
 \*  
 \* A descendant of QPushButton, but instead of  
 \* a normal button "box.png" is displayed  
 \*/*class Box : public QPushButton {  
private:  
 **Q\_OBJECT** bool opened; *// Is the box opened* int value; *// Amount of the money*public:  
 explicit Box(QWidget\* = nullptr);  
  
 void setRandomValue(); *// Random value setter* void setValue(int); *// Value setter* void setOpen(bool); *// Opened setter* int getValue(); *// Value getter* bool isOpen(); *// Opened getter* void draw(int, int);  
};

* Класс HelpWindow:

*/\*\*  
 \* @class HelpWindow  
 \* @brief Represents a help dialog window in the game  
 \*  
 \* Simply creates a window with game rules text  
 \*/*class HelpWindow : public QDialog {  
private:  
 **Q\_OBJECT** QLabel \*output;  
  
public:  
 HelpWindow(QDialog\* = nullptr);  
};

**Подсистема «Модель»**

1. **Перечень событий, изменяющих состояние модели**

К событиям, изменяющим состояние модели относятся:

* Установление соединения сервер-хост;
* Нажатие кнопки «справка»;
* Закрытие окна «справка»;
* Нажатие кнопки «взять»;
* Нажатие кнопки «пропустить»;
* Нажатие кнопки Коробка;
* Получение сообщения о новом ходе.

1. **Заголовочные файлы классов модели**

* Класс Game:

*/\*\*  
 \* @class Game  
 \* @brief Represents the game application  
 \*  
 \* Manages the main actions of the game.  
 \* Includes Peer2Peer communication between two games  
 \*/*class Game : public QApplication {  
private:  
 **Q\_OBJECT** P2PCommunicator \*communicator; *// Handle players communication* Interface \*interface; *// Player interface*public:  
 Game(int, char \*\*); *// Constructor*public **slots**:  
 void handshakeDone(int); *// Start the game* void from(const QByteArray&); *// Receive a message* void to(const QString&); *// Send a message***signals**:  
 void gameStarted(); *// Player 1 starts the game* void startWaiting(); *// Player 2 starts waiting*};

* Класс Player:

*/\*\*  
 \* @class Player  
 \* @brief Represents a player of the game  
 \*  
 \* Holds the current state of a player,  
 \* including their money and how many moves they have left.  
 \*/*class Player {  
private:  
 int money; *// How much money player could take* int movesLeft; *// How many moves left*public:  
 Player();  
  
 void setMoney(int); *// Set custom amount of money* void setMovesLeft(int); *// Set custom amount of moves* void move(); *// Decrease number of moves* int getMoney(); *// Return amount of money* int getMovesLeft(); *// Return amount of moves left*};

* Класс Box:

*/\*\*  
 \* @class Box  
 \* @brief Represents a box in the game  
 \*  
 \* A descendant of QPushButton, but instead of  
 \* a normal button "box.png" is displayed  
 \*/*class Box : public QPushButton {  
private:  
 **Q\_OBJECT** bool opened; *// Is the box opened* int value; *// Amount of the money*public:  
 explicit Box(QWidget\* = nullptr);  
  
 void setRandomValue(); *// Random value setter* void setValue(int); *// Value setter* void setOpen(bool); *// Opened setter* int getValue(); *// Value getter* bool isOpen(); *// Opened getter* void draw(int, int);  
};

**Контрольный пример реализованного приложения**

Контрольный пример описывает следующий порядок действий двух игроков. В рамках контрольного примера количество ходов было уменьшено с пяти до трех с целью полноценного изучения поведения программы (для изучения большего числа ходов не нужно):

1. Запуск первого экземпляра приложения (рис. 7);
2. Запуск второго экземпляра приложения (рис. 8);
3. Выбор коробки первым игроком (рис. 9);
4. Выбор действия «пропустить» первым игроком (рис. 10);
5. Выбор коробки и действия «взять» вторым игроком (рис. 11);
6. Выбор коробки первым игроком (рис. 12);
7. Выбор действия «пропустить» первым игроком (рис. 13);
8. Выбор коробки первым игроком (рис. 14);
9. Получение результатов игры (рис. 15).

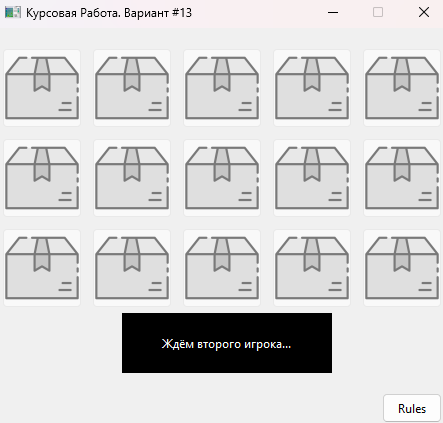


Рисунок 7. Запуск первого экземпляра

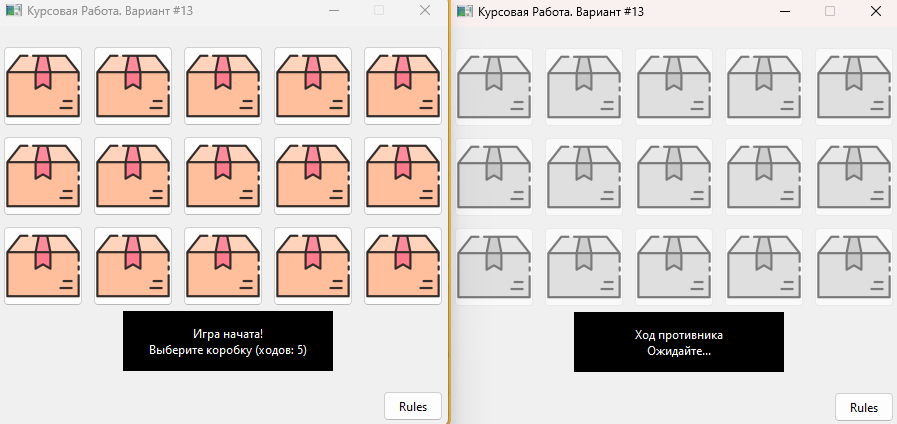


Рисунок 8. Запуск второго экземпляра

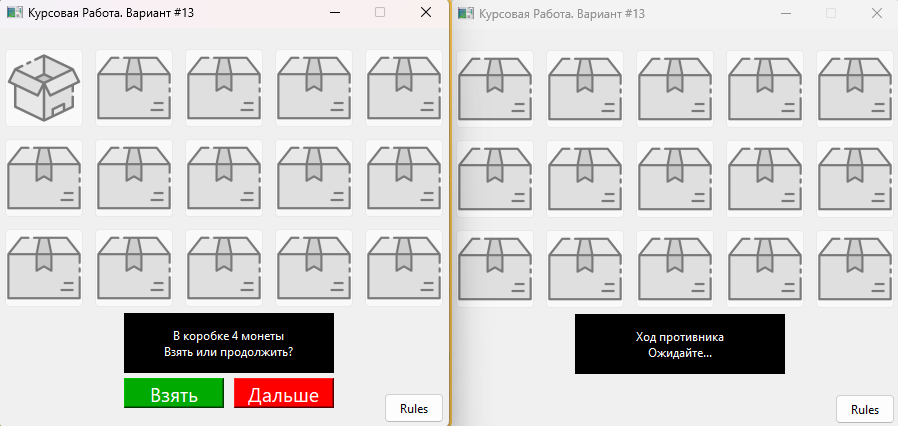


Рисунок 9. Первый выбор коробки игроком 1



Рисунок 10. Нажатие кнопки "дальше". Переход хода

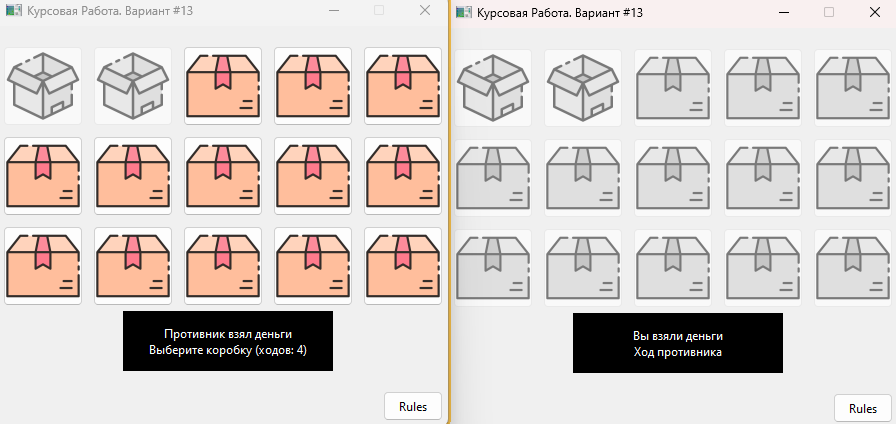


Рисунок 11. Выбор и взятие денег вторым игроком

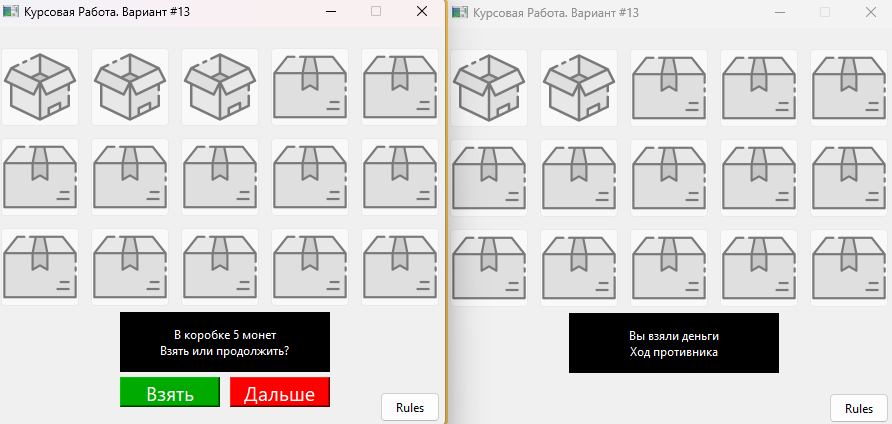


Рисунок 12. Второй выбор коробки первым игроком

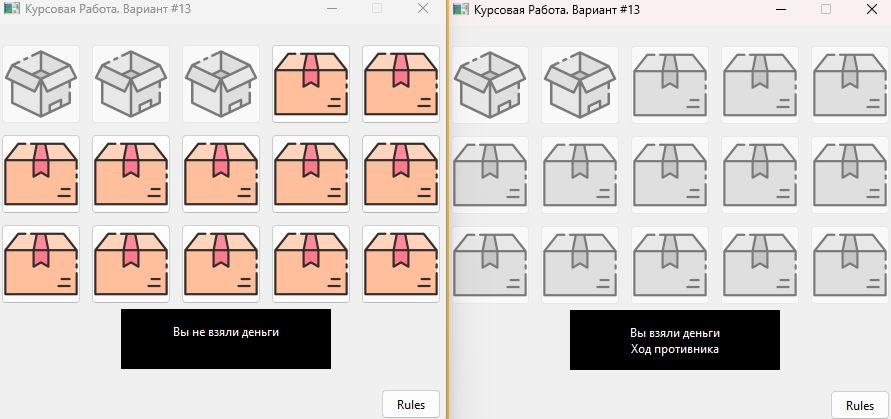


Рисунок 13. Нажатие "дальше" первым игроком

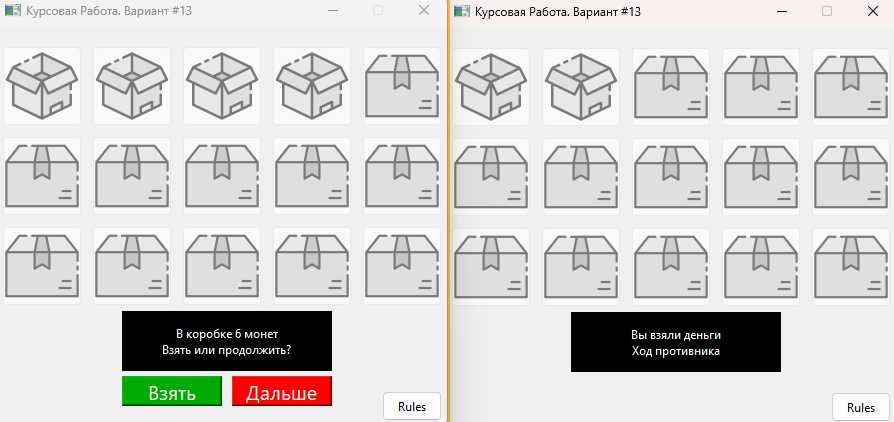


Рисунок 14. Третий выбор коробки первым игроком

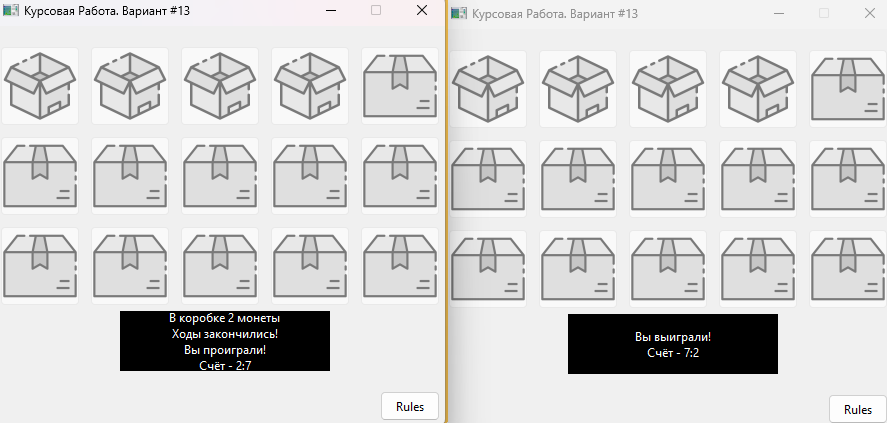


Рисунок 15. Нажатие "Дальше". Вывод результатов игры

**ЗАКЛЮЧЕНИЕ**

В ходе работы были изучены основные принципы ООП, серверного peer-to-peer соединения по протоколу UDP и основы разработки приложений с помощью Qt. Разработано, реализовано и отлажено объектно-ориентированное приложение, решающее поставленную задачу.