**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**КЛИЕНТСКАЯ ЧАСТЬ КЛИЕНТ-СЕРВЕРНОГО ПРИЛОЖЕНИЯ   
по дисциплине   
«Программные средства разработки интеллектуальных систем»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 1308, |  | Мельник Д. А. |
| Студент гр. 1308, |  | Томилов Д. Д. |
| Студент гр. 1308, |  | Лепов А. В. |

Санкт-Петербург

2022

**СОДЕРЖАНИЕ**

[1. ПОСТАНОВКА ЗАДАЧИ 3](#_Toc122640706)

[2. ДИЗАЙН ИНТЕРФЕЙСА КЛИЕНТСКОЙ ЧАСТИ И ОПИСАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ОПЫТА 3](#_Toc122640707)

[3. ВЫБРАННЫЕ ТЕХНОЛОГИИ 6](#_Toc122640708)

[4. РЕАЛИЗАЦИЯ КЛИЕНТСКОЙ ЧАСТИ И СВЯЗЬ ЕЁ С СЕРВЕРНОЙ ЧАСТЬЮ 6](#_Toc122640709)

[5. ТЕСТ-КЕЙСЫ 7](#_Toc122640710)

[6. ПРИМЕР РАБОТЫ 8](#_Toc122640711)

1. **ПОСТАНОВКА ЗАДАЧИ**

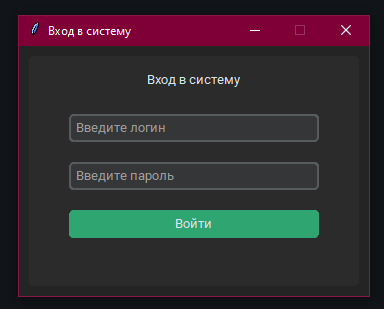
В качестве задачи на данном этапе необходимо реализовать клиентскую часть, которая даст пользователю (клиенту) рабочий, простой в использовании десктоп интерфейс для игры в шахматы.

1. **ДИЗАЙН ИНТЕРФЕЙСА КЛИЕНТСКОЙ ЧАСТИ И ОПИСАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ОПЫТА**

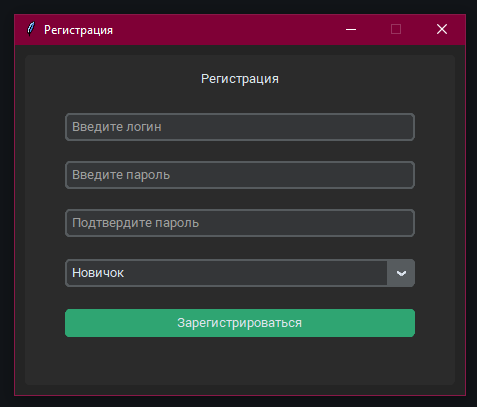
Описание пользовательского опыта:

* При запуске программы пользователь входит в систему в качестве гостя.
* Ему необходимо войти в систему, чтобы изменить роль на пользователя.

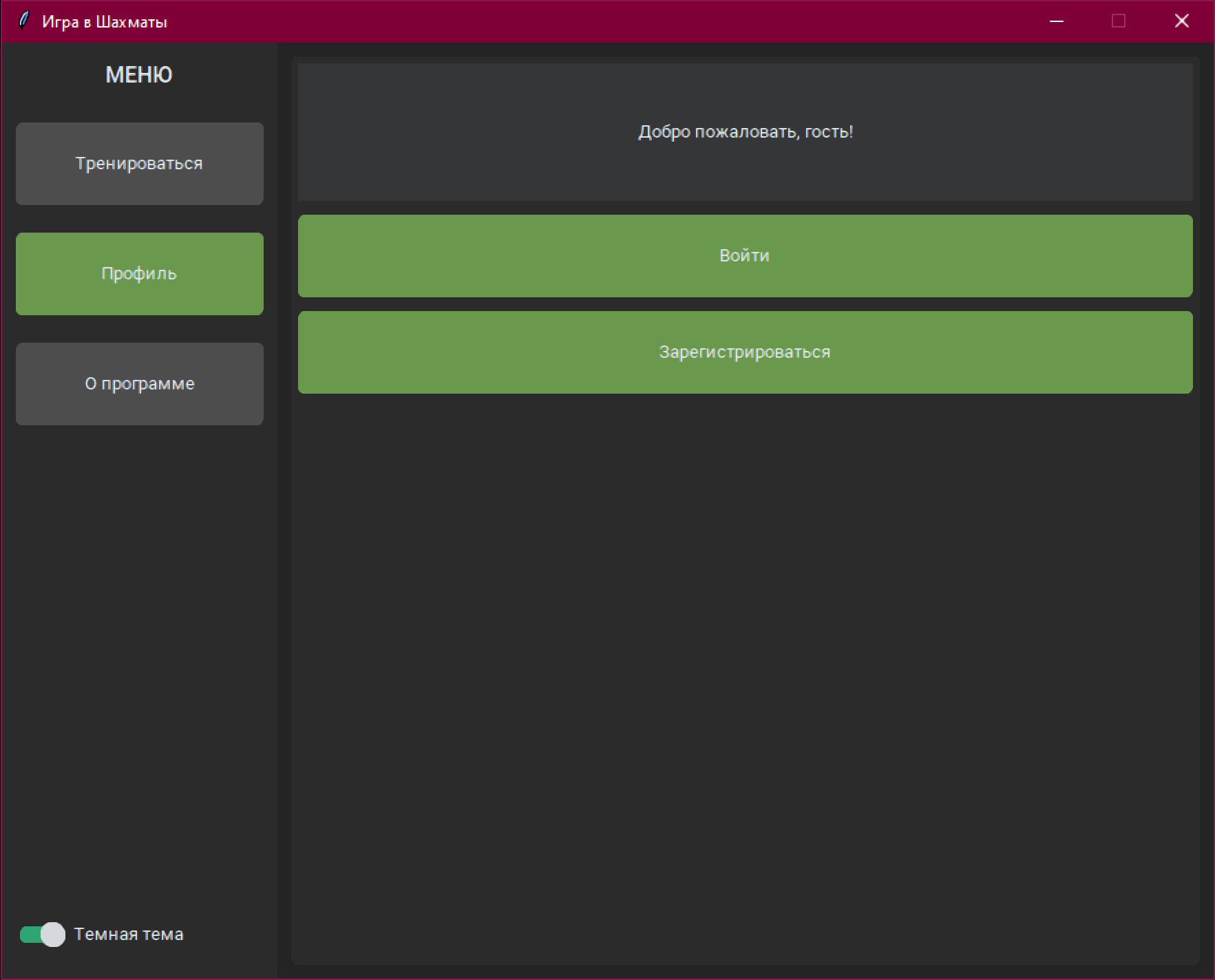
На рисунках 1-6 представлен дизайн интерфейса клиентской части.



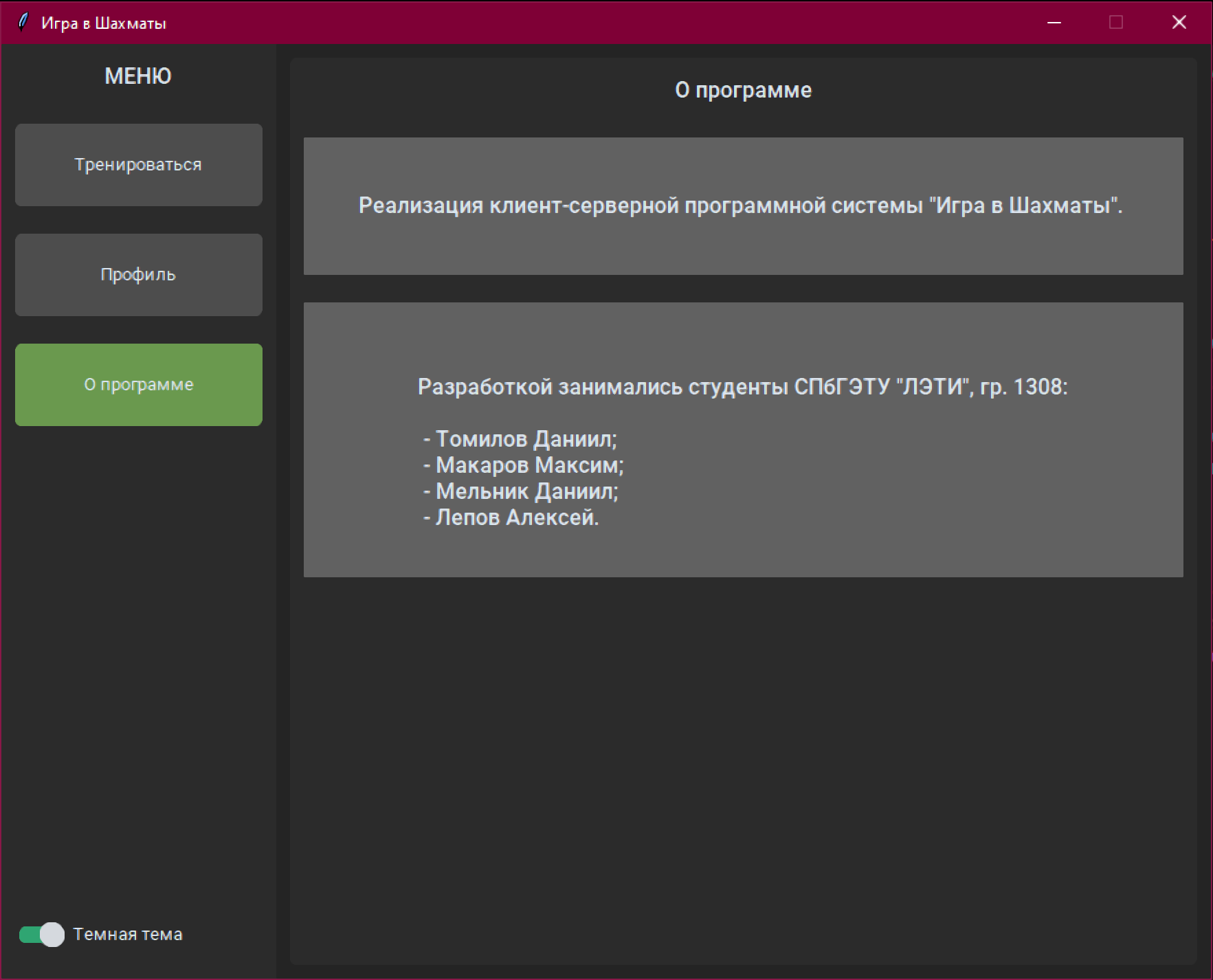
1. *Интерфейс входа в систему*



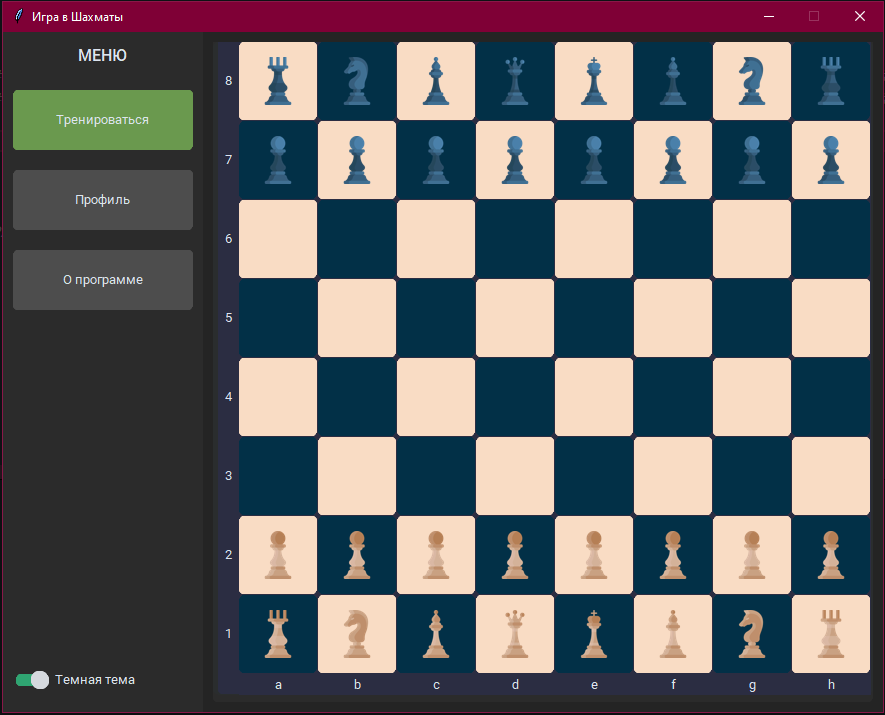
1. *Интерфейс входа в систему*



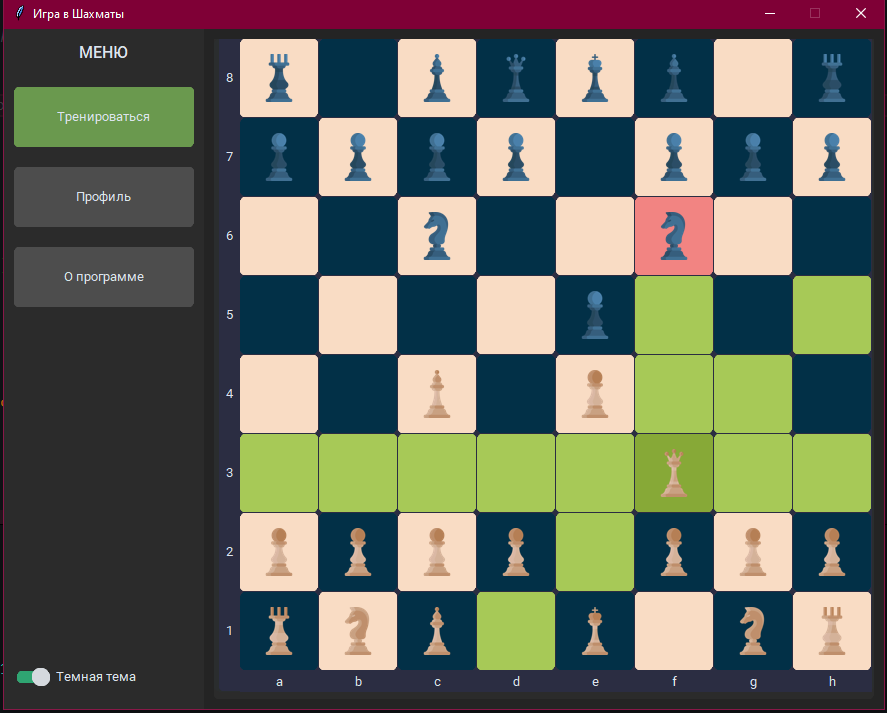
1. *Интерфейс входа в систему*



1. *Интерфейс входа в систему*



1. *Интерфейс входа в систему*



1. *Интерфейс входа в систему*
2. **ВЫБРАННЫЕ ТЕХНОЛОГИИ**

Выбранные технологии:

* Python 3;
* TKinter;
* CustomTkinter;
* IDE: VS Code;
* IDE: PyCharm.

1. **РЕАЛИЗАЦИЯ КЛИЕНТСКОЙ ЧАСТИ И СВЯЗЬ ЕЁ С СЕРВЕРНОЙ ЧАСТЬЮ**

Технологии серверной части приложения:

При создании серверной части приложения была использованы средства библиотеки socket.

Для привязки сокета к адресу использован метод bind, принимающий присваемый ip и порт в нём.

Для получения сообщения от определённого (по адресу) клиента используется метод recvfrom(), принимающий на вход максимальный объём информации для получения, возвращающий данные и адрес, с которого они были получены

Для отправки сообщений использован метов sendto(), принимающий данные и адрес принимающей стороны.

Технологии клиентской части приложения:

При создании клиентской части приложения была использованы средства библиотек socket и threading.

Для отправки сообщений использован метов sendto(), принимающий данные и адрес принимающей стороны.

Для приёма сообщения был использован метод recv(), принимающий на вход максимальный объём информации, возможный к приёму.

Для корректности попеременной работы клиентов с сервером был реализован поток через объект Thread библиотеки Threading.

1. **ТЕСТ-КЕЙСЫ**

import unittest

import main

class TestStringMethods(unittest.TestCase):

    def test\_castling\_1(self):

        board = main.chessEngine.Board.from\_FEN("r3k2r/8/8/8/8/8/8/R3K2R w KQkq - 0 1")

        board\_arr = board.get\_piece\_arr()

        self.assertTrue(board.move\_piece(board\_arr[0][4], main.chessEngine.Position(0,6)))

    def test\_false\_castling\_2(self):

        board = main.chessEngine.Board.from\_FEN("rrrrkrrr/8/8/8/8/8/8/RRRRKRRR w KQkq - 0 1")

        board\_arr = board.get\_piece\_arr()

        self.assertFalse(board.move\_piece(board\_arr[0][4], main.chessEngine.Position(0,2)))

    def test\_castling\_3(self):

        board = main.chessEngine.Board.from\_FEN("r3k2r/8/8/8/8/8/8/R3K2R b KQkq - 0 1")

        board\_arr = board.get\_piece\_arr()

        self.assertTrue(board.move\_piece(board\_arr[7][4], main.chessEngine.Position(7,6)))

    def test\_false\_castling\_4(self):

        board = main.chessEngine.Board.from\_FEN("rrrrkrrr/8/8/8/8/8/8/RRRRKRRR b KQkq - 0 1")

        board\_arr = board.get\_piece\_arr()

        self.assertFalse(board.move\_piece(board\_arr[7][4], main.chessEngine.Position(7,2)))

    def test\_piece\_eat(self):

        board = main.chessEngine.Board.from\_FEN("rnbqkbnr/ppp1pppp/8/3p4/4P3/8/PPPP1PPP/RNBQKBNR w KQkq - 0 1")

        board\_arr = board.get\_piece\_arr()

        self.assertTrue(board.move\_piece(board\_arr[3][4], main.chessEngine.Position(4,3)))

    def test\_false\_piece\_eat(self):

        board = main.chessEngine.Board.from\_FEN("rnbqkbnr/ppp1pppp/8/3p4/4P3/8/PPPP1PPP/RNBQKBNR w KQkq - 0 1")

        board\_arr = board.get\_piece\_arr()

        self.assertFalse(board.move\_piece(board\_arr[4][3], main.chessEngine.Position(3,4)))

    def test\_queenValue(self):

        self.assertEqual(main.chessEngine.Piece.get\_value("queen"), 9)

    def test\_getStartWhitePownPos(self):

        self.assertEqual(main.chessEngine.Piece.get\_start\_position("pawn", main.chessEngine.Color.WHITE), [main.chessEngine.Position(1, i) for i in range(8)])

    def test\_isupper(self):

        self.assertFalse((main.chessEngine.Piece.\_\_str\_\_(main.chessEngine.Piece(main.chessEngine.Color.BLACK,"pawn",1,None,True))=="White"))

if \_\_name\_\_ == '\_\_main\_\_':

    unittest.main()

1. **ПРИМЕР РАБОТЫ**

С примерами работы можно ознакомиться, перейдя на онлайн-репозиторий проекта: <https://github.com/AlexeyLepov/ClientServerChessApp>