

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет «Информатика и вычислительная техника»

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

Зав. кафедрой \_\_ПОВТиАС\_\_

\_\_\_\_\_\_\_\_\_\_\_\_ \_\_ В.В. Долгов\_

(подпись) (И.О.Ф.)

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2021г.

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту по дисциплине Объектно-ориентированное программирование

на тему Объектно-ориентированная реализация приложения «Шахматы»

Автор проекта (работы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н.С. Ли

подпись И.О.Ф

Направление/специальность, профиль/специализация:

09.03.04 Программная инженерия \_

коднаправления наименование профиля (специализации)

Обозначение курсового проекта (работы) КР.430000.000 Группа ВПР32

Руководитель проекта: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Б.В Габрельян подпись (должность, И.О.Ф.)

Проект (работа) защищен (а)

дата оценка подпись

Ростов-на-Дону

2021



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет «Информатика и вычислительная техника»

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

Зав. кафедрой \_\_ПОВТиАС\_\_

\_\_\_\_\_\_\_\_\_\_\_\_ \_\_ В.В. Долгов\_

(подпись) (И.О.Ф.)

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2021г.

**ЗАДАНИЕ**

к курсовой работе по дисциплине Объектно-ориентированное программирование

Студент Н.С. Ли КР.430000.000 Группа ВПР32

Тема Объектно-ориентированная реализация приложения «Шахматы»

Срок представления проекта (работы) к защите «\_\_» \_\_\_\_\_\_\_\_\_ 2021г.

Исходные данные для курсового проекта (работы)

Гради Буч. Объектно-ориентированный анализ и проектирование с примерами приложений. — 3-е издание. — «Вильямс», 2010.

CLR via C#. Программирование на платформе Microsoft .NET Framework 2.0 на языке C#.

2е изд., исправ., СПб: Питер, 2008.

Содержание пояснительной записки

ВВЕДЕНИЕ:

В разделе «Введение» рассматриваются актуальность предметной области.

Наименование и содержание разделов:

1. В разделе «Аналитический обзор» происходит изучение предметной области для создания приложения «шахматы», разбор задания на курсовое проектирование, рассмотрение аналогичных уже готовых решений для игры в шахматы, выбор оптимального способа реализации.
2. В разделе «Алгоритмическое конструирование» дается общее описание работы программного средства, представлены блок-схемы разработанных алгоритмов для генерации текущего состояния игры и выполнения ходов.
3. В разделе «Программное конструирование» осуществляется рассмотрение проектных решений, представлена UML диаграмма классов, описано их взаимодействие, а также, даётся описание основных классов Chess и Board и их методов.
4. В разделе «Тестирование» представлены результаты тестов проведенных над готовым ПП и описание исправлений ошибок, если они были обнаружены.

ЗАКЛЮЧЕНИЕ:

В курсовой работе, согласно заданию, в рамках объектно-ориентированной парадигмы программирования были реализованы все задачи, создано рабочее приложение «Шахматы» с графическим и текстовым интерфейсом, реализована шахматная библиотека для интеграции с любым другим интерфейсом.

Руководитель проекта (работы) Б.В. Габрельян

подпись, дата И.О.Ф.

Задание принял к исполнению Н.С. Ли

подпись, дата И.О.Ф

Оглавление

[ВВЕДЕНИЕ 5](#_Toc88905451)

[1 Аналитический обзор предметной области 6](#_Toc88905452)

[1.1 Понятие игры в шахматы 6](#_Toc88905453)

[1.2 Обзор методов игры в шахматы 9](#_Toc88905454)

[1.3 Обзор шахматных приложений 9](#_Toc88905455)

[1.3.1 Chess.com 10](#_Toc88905456)

[1.3.2 Kasparov Chessmate 10](#_Toc88905457)

[1.4 Постановка задачи 11](#_Toc88905458)

[1.5 Вывод по главе 12](#_Toc88905459)

[2 Алгоритмическое конструирование 13](#_Toc88905460)

[2.1 Общий алгоритм работы программного средства 13](#_Toc88905461)

[2.2 Алгоритм генерации текущего состояния доски 14](#_Toc88905462)

[2.3 Алгоритм выполнения хода 16](#_Toc88905463)

[2.4 Вывод по главе 17](#_Toc88905464)

[3 Программное конструирование 18](#_Toc88905465)

[3.1 Обоснование выбора средств разработки 18](#_Toc88905466)

[3.2 Описание основных классов 19](#_Toc88905467)

[3.3 Вывод по главе 20](#_Toc88905468)

[4 Тестирование программного средства 21](#_Toc88905469)

[4.1 Описание процесса тестирования 21](#_Toc88905470)

[4.1.1 Command Line Interface 21](#_Toc88905471)

[4.1.2 Unity 23](#_Toc88905472)

[4.2 Вывод по главе 24](#_Toc88905475)

[ЗАКЛЮЧЕНИЕ 25](#_Toc88905476)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 26](#_Toc88905477)

[ПРИЛОЖЕНИЕ А 27](#_Toc88905478)

[UML-диаграмма классов 27](#_Toc88905479)

[ПРИЛОЖЕНИЕ B 28](#_Toc88905480)

[Исходный код программы 28](#_Toc88905481)

ВВЕДЕНИЕ

Шахматы — настольная логическая игра со специальными фигурами на 64-клеточной доске для двух соперников, сочетающая в себе элементы искусства (в части шахматной композиции), науки и спорта. Название берёт начало из персидского языка: шах мат, что значит буквально: «шах умер».

Сегодня шахматы, отвечая глубокой потребности людей в творческом самовыражении, рождены для испытания ума. Шахматы — абстракция, чистая мысль. Это демонстрация способности творить новое на основе известного. Они проверяют спортсменов резком противоборстве ума, воли, проницательности, воображения, дальновидности. В абстрактной форме они воплощают идею справедливости: перед шахматами все равны. Игра отвечает истинно человеческим потребностям разума, духовного нашего существа. В борьбе создаются, захватывающие дух своей красотой, произведения шахматного искусства, несущие на себе отпечаток драматизма, озарений и заблуждений.

С развитием электронной техники появилась возможность перенести всеми любимую игру в электронный вариант, и наслаждаться процессом в любом месте мира на любом устройстве.

Целью курсового проекта является разработка приложения, которое будет выполнять функции игры «Шахматы».

В ходе выполнения проекта будут задействованы знания и навыки в области объектно-ориентированного программирования. В качестве платформы для создания будет выступать язык программирования C# и Unity для графического интерфейса программы.

1 Аналитический обзор предметной области

В данном разделе даётся понятие игры в шахматы и рассматривается задача реализации шахматных алгоритмов, приводятся её особенности, определяется цель работы и задачи.

* 1. Понятие игры в шахматы

Ша́хматы (перс. شاه مات ‘шах мат’, буквальный перевод «шах умер») — настольная логическая игра с шахматными фигурами на 64-клеточной доске, сочетающая в себе элементы искусства (в том числе в части шахматной композиции), науки и спорта[1].

В шахматы обычно играют два игрока (именуемые шахматистами) друг против друга. Также возможна игра одной группы шахматистов против другой или против одного игрока, такие партии зачастую именуются консультационными. Кроме того, существует практика сеансов одновременной игры, когда против одного сильного игрока играет несколько противников, каждый на отдельной доске.

Правила игры в основном сложились к XV веку; в современных официальных турнирах применяются правила Международной шахматной федерации[2], которыми регламентируются не только передвижение фигур, но и права судьи, правила поведения игроков и контроль времени. Игра, осуществляемая дистанционно, например, по переписке, по телефону или через Интернет, имеет особые правила. Существует множество вариантов шахмат, отличающихся от классических: с нестандартными правилами, фигурами, размерами доски и т. п. Соответствующий раздел шахматной композиции — сказочные шахматы. Некоторые аспекты шахматной игры изучаются в математике (например, классические «Задача о ходе коня» и «Задача о восьми ферзях»), в том числе при помощи компьютерного моделирования.

Шахматная партия играется между двумя партнерами, которые поочередно перемещают фигуры на квадратной доске, названной "шахматной". Играющий белыми начинает партию. Игрок получает право хода, когда его партнер сделал ход.

Цель каждого игрока - атаковать короля партнера таким образом, чтобы партнер не имел никаких возможных ходов, которые позволяют избежать "взятия" короля на следующем ходу. Об игроке, который достиг этой цели, говорят, что он поставил мат королю партнера и выиграл партию. Партнер, королю которого был поставлен мат, проиграл партию.

Если позиция такова, что никто из партнеров не может поставить мат, партия заканчивается вничью. Шахматная доска состоит из 64 равных квадратов (8х8), поочередно светлых ("белые" поля) и темных ("черные" поля). В начале партии один игрок имеет 16 светлых фигур ("белые"); другой - 16 темных фигур ("черные"): Белый король, Белый ферзь, Две белые ладьи, Два белых слона, Два белых коня, Восемь белых пешек, Черный король, Черный ферзь, Две черные ладьи, Два черных слона, Два черных коня, Восемь черных пешек. Начальная позиция фигур на шахматной доске представлена на рисунке 1.1.

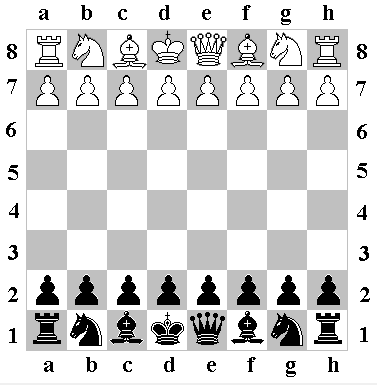


Рисунок 1.1 – Начальное расположение фигур на шахматной доске.

Ни одна из фигур не может быть перемещена на поле, занятое фигурой того же цвета. Если фигура переходит на поле, занимаемое фигурой партнера, последняя считается взятой и убирается с шахматной доски как часть того же самого хода.

Слон может ходить на любое поле по диагоналям, на которых он стоит.

Ладья может ходить на любое поле по вертикали или горизонтали, на которых она стоит.

Ферзь ходит на любое поле по вертикали, горизонтали или диагонали, на которых он стоит.

Когда делаются эти ходы, ферзь, ладья или слон не могут перемещаться через поле, занятое другой фигурой.

Конь может ходить на одно из ближайших полей от того, на котором он стоит, но не на той же самой вертикали, горизонтали или диагонали.

Пешка может ходить вперед на свободное поле, расположенное непосредственно перед ней на той же самой вертикали, или с исходной позиции пешка может продвинуться на два поля по той же самой вертикали, если оба эти поля не заняты, или пешка ходит на поле, занимаемое фигурой партнера, которая расположена по диагонали на смежной вертикали, одновременно забирая эту фигуру. Когда пешка достигает самой дальней горизонтали от своей исходной позиции, она должна быть заменена на ферзя, ладью, слона или коня "своего" цвета, что является частью того же хода. Выбор игрока не ограничивается фигурами, которые были уже сняты с доски. Эта замена пешки на другую фигуру называется "превращением", и действие новой фигуры начинается сразу.

Король может перемещаться двумя различными путями:

* ходить на любое примыкающее поле, которое не атаковано одной или более фигурами партнера. Считается, что фигуры партнера атакуют поле, даже в том случае, когда они не могут ходить.
* "рокируясь". Это перемещение короля и одной из ладей того же цвета по крайней горизонтали считается одним ходом короля и выполняется следующим образом: король перемещается с его исходного поля на два поля по направлению к ладье, затем ладья переставляется через короля на последнее поле, которое только что король пересек.

Партия выигрывается игроком, который поставил мат королю партнера.

Партия выигрывается игроком, партнер которого заявляет, что он сдается.

Партия считается закончившейся вничью, если тот игрок, за которым очередь хода не имеет никаких возможных ходов, и его король не под шахом.

Партия считается закончившейся вничью, если возникла позиция, когда ни один из партнеров не может заматовать короля любыми возможными ходами.

Партия считается закончившейся вничью по соглашению между двумя партнерами в ходе игры.

Для записи игры удобно использовать нотацию Форсайта–Эдвардса (англ. Forsyth–Edwards Notation, FEN) — стандартную нотацию записи шахматных диаграмм.

* 1. Обзор методов игры в шахматы

Существуют различные методы игры в шахматы. Наиболее распространенными являются такие методы, как:

* «Живые» шахматы, классическая игра на настоящей доске.
* Шахматы по переписке.
* Программные средства для ПК.
* Online сервисы.

Одним из наиболее популярных, на сегодняшний день, методов является специальные программы для ПК, именно его я и буду рассматривать далее.

* 1. Обзор шахматных программ

На сегодняшний день существует достаточно много уже готовых приложений, помогающих людям играть в шахматы. Приведу примеры некоторых из этих программных средств.

* + 1. Chess.com

Chess.com — сайт для игры в шахматы онлайн, форум, шахматная социальная сеть, а также одноимённая компания, владеющая данным сайтом. По данным Alexa.com, является самым популярным шахматным сайтом в мире. Работа данного ПС изображена на рисунке 1.1.



Рисунок 1.2 – Интерфейс программного средства «Chess.com»

* + 1. Kasparov Chessmate

Kasparov Chessmate - Или Шахматы с Гарри Каспаровым. Создана при непосредственном участии 13-го чемпиона мира. Содержит в себе ряд исторических партий Каспарова, а также упражнения и задачи, им составленные. В программе два одиночных режима: в первом игрок имеет возможность брать подсказки, изменять время на ход и уровень сложности; второй режим – турнир с возрастающим от каждого тура уровнем соперников, в заключительном раунде игроку предстоит сыграть с самим Каспаровым.

Интерфейс последней версии программы изображён на рисунке 1.2.



Рисунок 1.3 – Интерфейс программного средства «Kasparov Chessmate»

1.4 Постановка задачи

Проведенный выше обзор показал, оптимальным решением для игры в шахматы является приложение для ПК.

В результате аналитического обзора была поставлена цель: разработать программное средство, позволяющее пользователю запускать партию в шахматы в любой момент времени.

Для достижения поставленной цели необходимо решить следующие задачи:

* разработать алгоритмы контроля правильности ходов;
* разработать алгоритмы композиции и декомпозиции FEN-нотации
* выбрать среду и язык реализации;
* разработать модуль интуитивно-понятного интерфейса программного средства (ПС) и выполнить его интеграцию с написанными алгоритмами;
* провести тестирование ПС на нескольких контрольных примерах;
* в соответствии с отчетом по тестированию выполнить корректировку ПС;

В программном средстве будут реализованы следующие функции: декомпозиция FEN-нотации, расстановка шахматных фигур на доске согласно этой нотации, возможность сделать ход, проверка правильности сделанного хода для каждой из фигур, специфические шахматные правила.

Результатом разработки будет являться шахматный движок в виде .dll библиотеки и графический интерфейс для представления работы.

1.5 Вывод по главе

В данной главе были рассмотрены понятия шахмат и правил игры в них, приведён обзор уже существующих программных средств выполняющих функции шахматных приложений. Также были выявлены основные задачи и методы по их реализации. На основе проведенного обзора была поставлена задача и описаны функции, которые будет выполнять разрабатываемое программное средство.

2 Алгоритмическое конструирование

В данном разделе рассматриваются основные алгоритмы работы программного средства: общий алгоритм, алгоритм генерации текущего состояния доски и алгоритм выполнения хода, а также приведено их описание и схема работы.

2.1 Общий алгоритм работы программного средства

Словесное описание общего алгоритма работы ПС:

1. Запуск программы. Инициализация главного окна. Инициализация всех управляющих элементов.
2. Сборка доски. Выставление фигур согласно FEN.
3. Если не конец игры отобразить доску и ждать выполнения хода.
4. Проверить корректность выполненного хода и записать его в FEN. Переход к п.2
5. Если игра завершилась и пользователь хочет сыграть снова, перейти к п.2 с начальным FEN, в противном случае – закрыть окно.

Блок-схема общего алгоритма работы представлена на рисунке 2.1.



Рисунок 2.1 – Общий алгоритм работы программы

2.2 Алгоритм генерации текущего состояния доски

Для выполнения хода необходимо генерировать состояние доски и транслировать FEN-нотацию в состояния флагов и массивов.

Алгоритм добавления показан на рисунке 2.2.

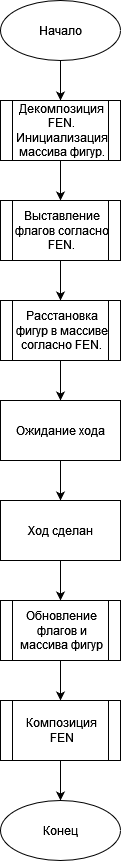


Рисунок 2.2 – Алгоритм генерации текущего состояния доски

2.3 Алгоритм выполнения хода

При выполнении хода создаётся экземпляр текущего состояния доски с уже сделанным ходом, если он возможен и обновляются флаги.

Данный алгоритм поиска операции в таблице приведён на рисунке 2.3.

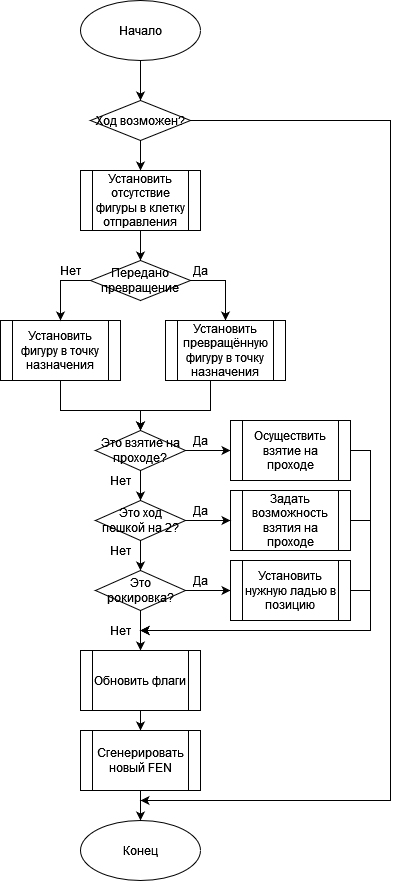


Рисунок 2.3 – Алгоритм выполнения хода

2.4 Вывод по главе

В данной главе был рассмотрен основной алгоритм работы программного средства. Также были описаны алгоритмы генерации текущего состояния доски и выполнения хода, а также приведены схемы их работы.

Все приведенные алгоритмы используются для дальнейшего проектирования приложения.

3 Программное конструирование

В данном разделе будут обоснованы выбор языка программирования, используемый для реализации программы, а также представлены основания выбора среды программирования. Будут определены и описаны основные подпрограммы разрабатываемого приложения.

3.1 Обоснование выбора средств разработки

Для реализации разработанных алгоритмов был выбран язык программирования С#.

C# (произносится си шарп) — объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота[3] как язык разработки приложений для платформы Microsoft .NET Framework. Впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270.

Язык программирования С# имеет ряд преимуществ: объектно-ориентированность, поддержка подавляющего большинства продуктов Microsoft, бесплатность ряда инструментов — Visual Studio, облако Azure и др., типы данных имеют фиксированный размер (32-битный int и 64-битный long), что повышает «мобильность» языка и упрощает программирование, так как мы всегда знаем точно, с чем мы имеем дело, автоматическая «сборка мусора», большое количество «синтаксического «сахара» — специальных конструкций, разработанных для понимания и написания кода.

В качестве среды разработки выбран продукт Microsoft – visual studio. Visual Studio признана лучшей IDE для C#. Дело в том, что оба продукта принадлежат корпорации Microsoft. Поэтому они идеально подходят для работы друг с другом.

В число преимуществ Visual Studio входит следующее:

* Среда содержит множество инструментов, которые очень хорошо работают на C#.
* Наличие бесплатной версии — Community Edition.
* Community содержит все что нужно для независимого разработчика.
* Самое эффективное ПО для разработки на любой платформе, включая .Net и C#.
* Возможность хранения данных в облаке.

3.2 Описание основных классов и их взаимодействия

Программное средство состоит из множества классов и которые обеспечивают его работоспособность. Структура отдельных классов и формы их взаимодействий, которых представлены в виде диаграммы UML, в приложении А.

Публичным классом для взаимодействия с любым графическим интерфейсом является класс Chess. Он содержит в себе всю логику игры, которую передаёт во «вне». Саму по себе игру в шахматы можно декомпозировать на примитивные сущности для оптимального дальнейшего проектирования. Самыми примитивными в шахматах являются сущности фиугр и клеток, они представлены в виде классов Figure и Square соответственно. Эти примитивы объединяются в классе FigureOnSquare, если на конкретной клетке есть данная фигура – то это объект этого класса. Класс FigureMoving необходим для осуществления логики передвижения фигуры с клетки на клетку, содержит в себе все 3 вышеописанных класса. Класс Moves отвечает за логику передвижения фигуры, проверку правильности хода, содержит в себе FigureMoving и класс Board который в свою очередь представляет собой саму доску. Таким образом, разбив задачу на компоненты была описана игра в шахматы и собран «по деталям» класс Chess.

Более детально структура основных классов программы описана ниже.

3.2.1 Класс Chess

Основной общедоступный класс Chess должен обеспечивать базовую логику игры и связки всех компонентов. Реализует публичный интерфейс взаимодействия. В таблице 3.1 представлены методы класса Chess.

Таблица 3.1 – Методы класса «Chess»

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Входные параметры | Описание метода | Возвращаемое значение |
| Chess | String fen | Конструктор класса инициализирующий поля по заданному FEN | Нет |
| Chess | Board board | Приватный конструктор класса инициализирующий поля по переданному состоянию доски | Нет |
| Move | string move | Обновляет состояние поля board, возвращает доску с новым сделанным ходом | Chess |
| GetFigure | int x, int y  или  string square | По заданным координатам или по строке характеризующей шахматную клетке возвращает фигуру, стоящую на этой клетке | char |
| GetAllMoves | Нет | Возвращает список всех возможных ходов в данной позиции для всех фигур | List<string> |
| IsCheck | Нет | Публичный метод для проверки на наличие шаха | bool |
| IsValidMove | string move | Публичный метод для проверки корректности хода | bool |

3.2.1 Класс Board

Приватный класс Board представляет собой шахматную доску. Он берет на себя всю работу с FEN, инициализацию доски и фигур на ней. Этот класс имеет скрытые поля: fen – хранящий сведения о текущем состоянии доски в формате строки; figures – представляющий из себя двумерный массив фигур на доске; moveColor – для хранения цвета той стороны, которая совершает ход; moveNum – номер текущего хода. Так же имеются публичные свойства: CanCastle – для хранения информации о рокировке; Enpassant – для хранения информации о взятии на проходе. Методы класса Board с их описанием представлены в таблице 3.2.

Таблица 3.2 – Методы класса «Board»

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Входные параметры | Описание метода | Возвращаемое значение |
| Board | String fen | Конструктор класса инициализирующий поля по заданному FEN | Нет |
| Move | FigureMoving fm | Метод для совершения хода. Возвращает новое состояние доски | Board |
| GetFigure | Square square | По заданной клетке  возвращает фигуру, стоящую на этой клетке | Figure |
| SetFigure | Square square, Figure figure | Позволяет задать необходимую фигуру на нужную клетку в поле figures | нет |
| IsCheck | Нет | Публичный метод для проверки на наличие шаха | bool |
| IsCheckAfterMove | FigureMoving fm | Публичный метод для проверки на наличие шаха после заданного хода. | bool |
| Init | Нет | Метод для инициализации полей в классе Board.  Декомпозирует FEN, нужные части добавляет к соответствующим полям. | Нет |
| GenerateFen | Нет | Метод обратный методу Init. Собирает информацию из полей и создаёт на их основании строку FEN обновляя соответствующее поле | Нет |

3.3 Вывод по главе

В данной главе был обоснован выбор языка программирования, а также

среды программирования для создаваемого программного средства. Кроме того, были описаны основные классы и входящие в них методы данного программного средства.

4 Тестирование программного средства

В данном разделе продемонстрированы основные этапы тестирования программного средства на основе различных входных данных.

4.1 Описание процесса тестирования

Были протестированы все возможные функции приложения шахматы, такие как: запуск игры, совершение ходов.

4.1.1 Command Line Interface

Главное окно программного средства с консольным интерфейсом изображено на рисунке 4.1.

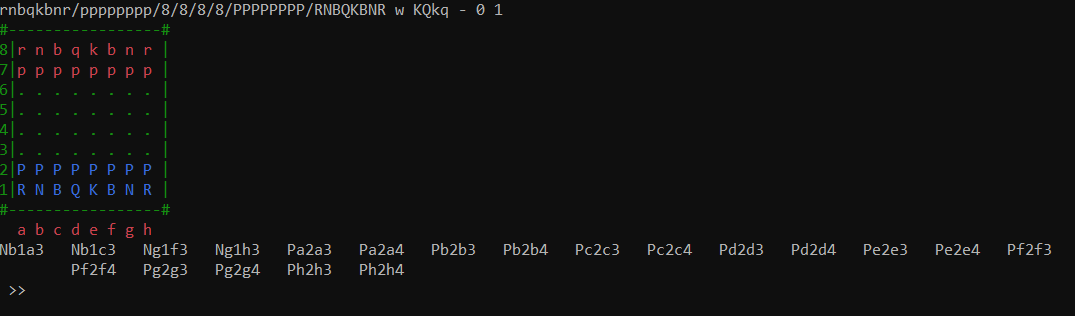


Рисунок 4.1 – Главное окно программы с консольным интерфейсом.

Перед нами предстаёт окно программы на котором располагаются необходимые элементы: в первой строке находится FEN, текущей позиции на доске, для возможности импорта. Далее изображена доска с фигурами с помощью ASCII графики. И под конец список из всех возможных ходов.

Предоставляется возможность хода фигурами по правилам. Демонстрация возможности хода показана на рисунке 4.2.

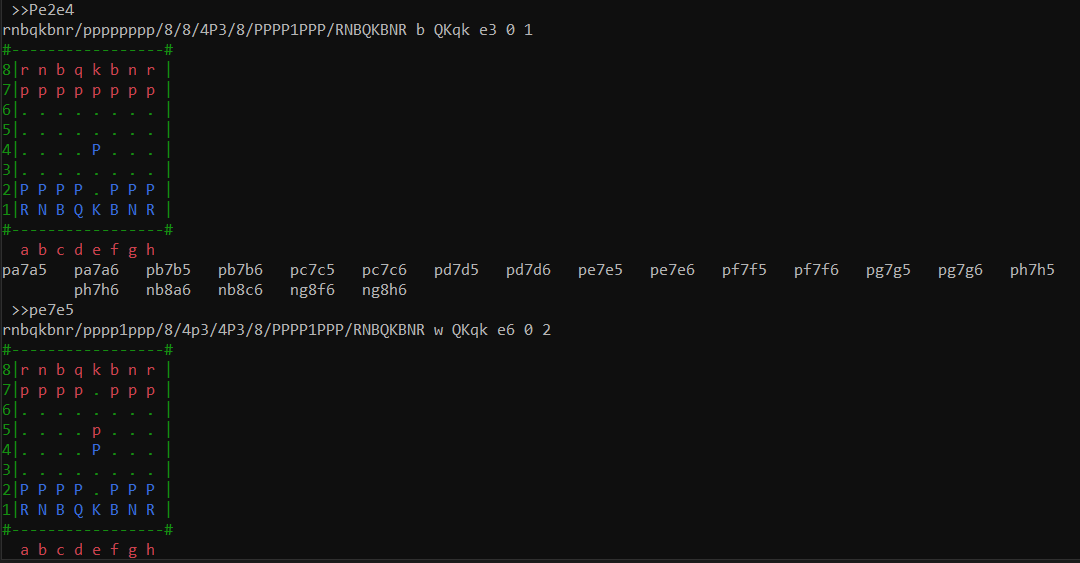


Рисунок 4.2 – Сделано 2 хода.

Разыграв классическую итальянскую партию, попробуем смухлевать и съесть ладьёй белых ферзя чёрных. Результат на рисунке 4.3.

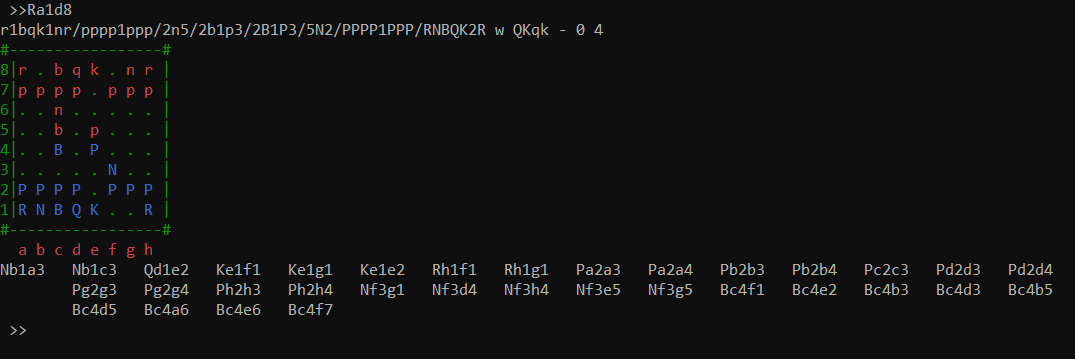


Рисунок 4.3 – Невозможный ход.

Программа отказалась выполнять невозможный ход, что означает, что она работает корректно.

4.1.2 Unity

Главное окно программного средства с графическим интерфейсом изображено на рисунке 4.4.



Рисунок 4.4 – Главное окно программы с графическим интерфейсом

Здесь мы наблюдаем меню с 3мя кнопками на выбор. Для старта игры нажимаем “play”. И перед нами появляется основное игровое окно с возможностью играть непосредственно в шахматы, выйти в меню, выйти из игры и кнопкой подтверждения хода.

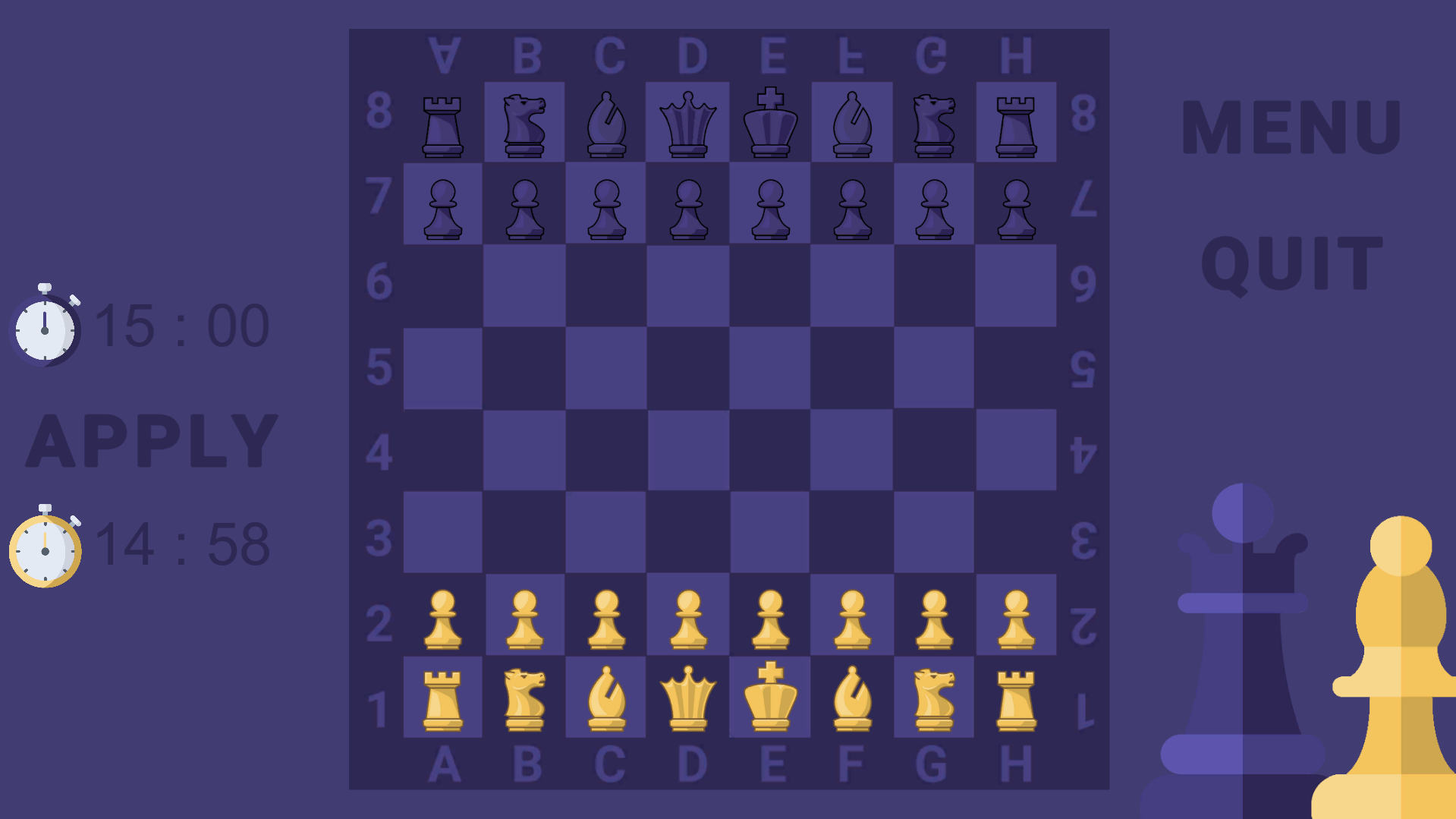


Рисунок 4.5 – Окно игры

На рисунке 4.6 показана возможность сделать ход пешкой, и, подтверждения хода, право выбора передаётся чёрным.

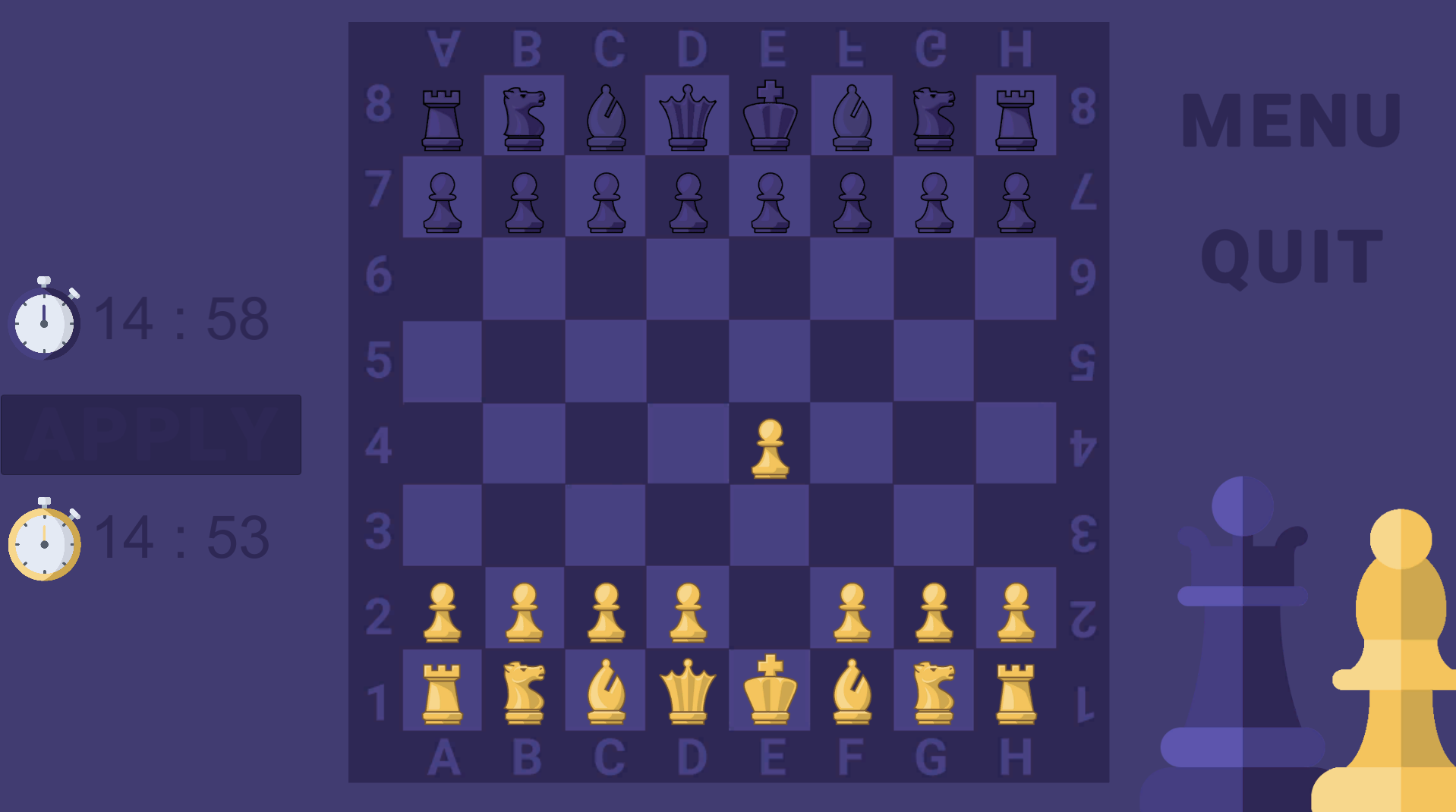


Рисунок 4.6 – Выполнен ход белой пешкой E2-E4.

После ответа чёрных D2-D4, белые попытались нарушить правила, сходив ферзём D1-D5 и подтвердили ход ситуация представлена на рисунке 4.7.

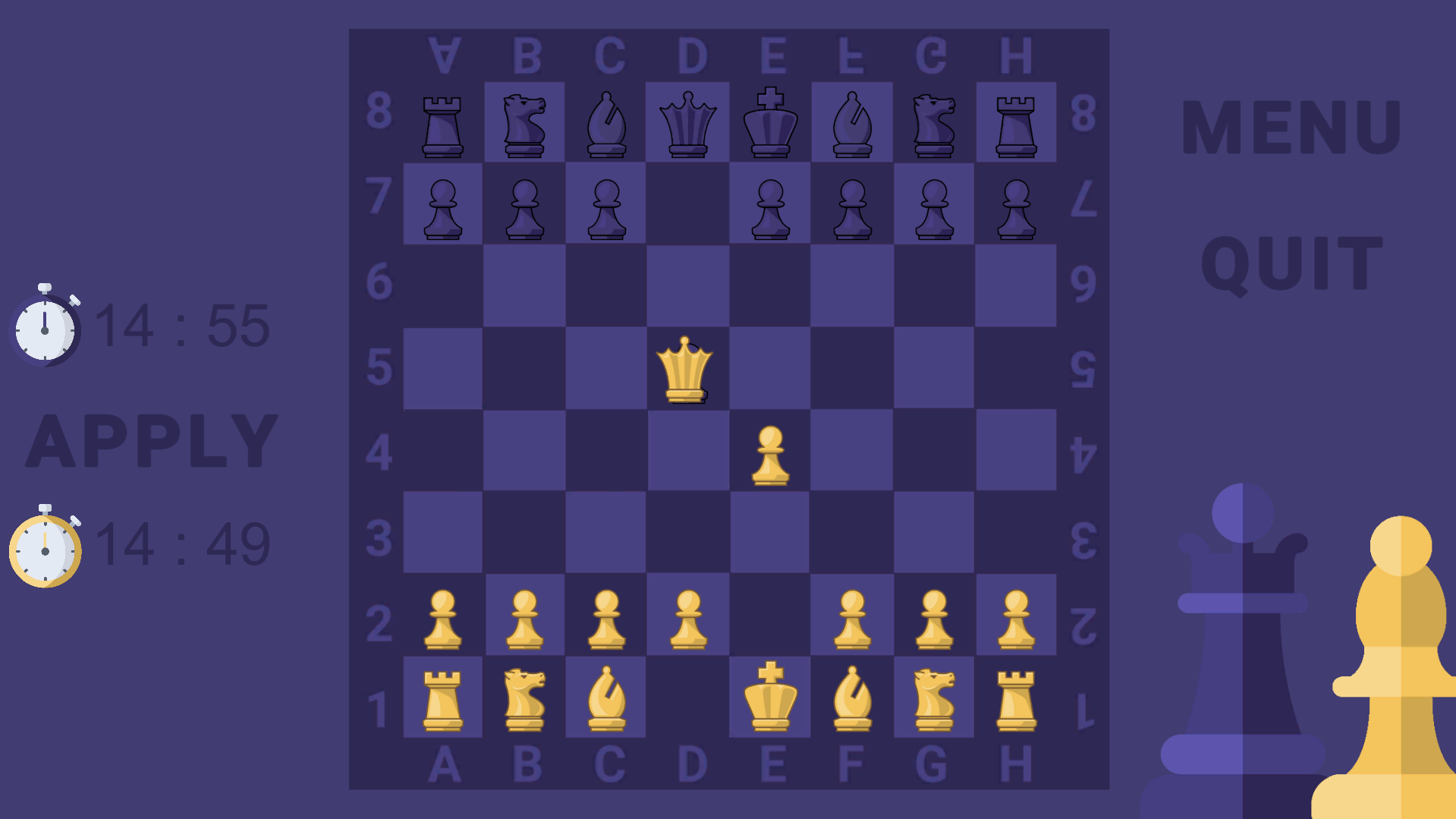


Рисунок 4.7 – Попытка нарушить правила.

Но у них ничего не вышло, и все фигуры остались стоять на своих местах, правила работают как надо.

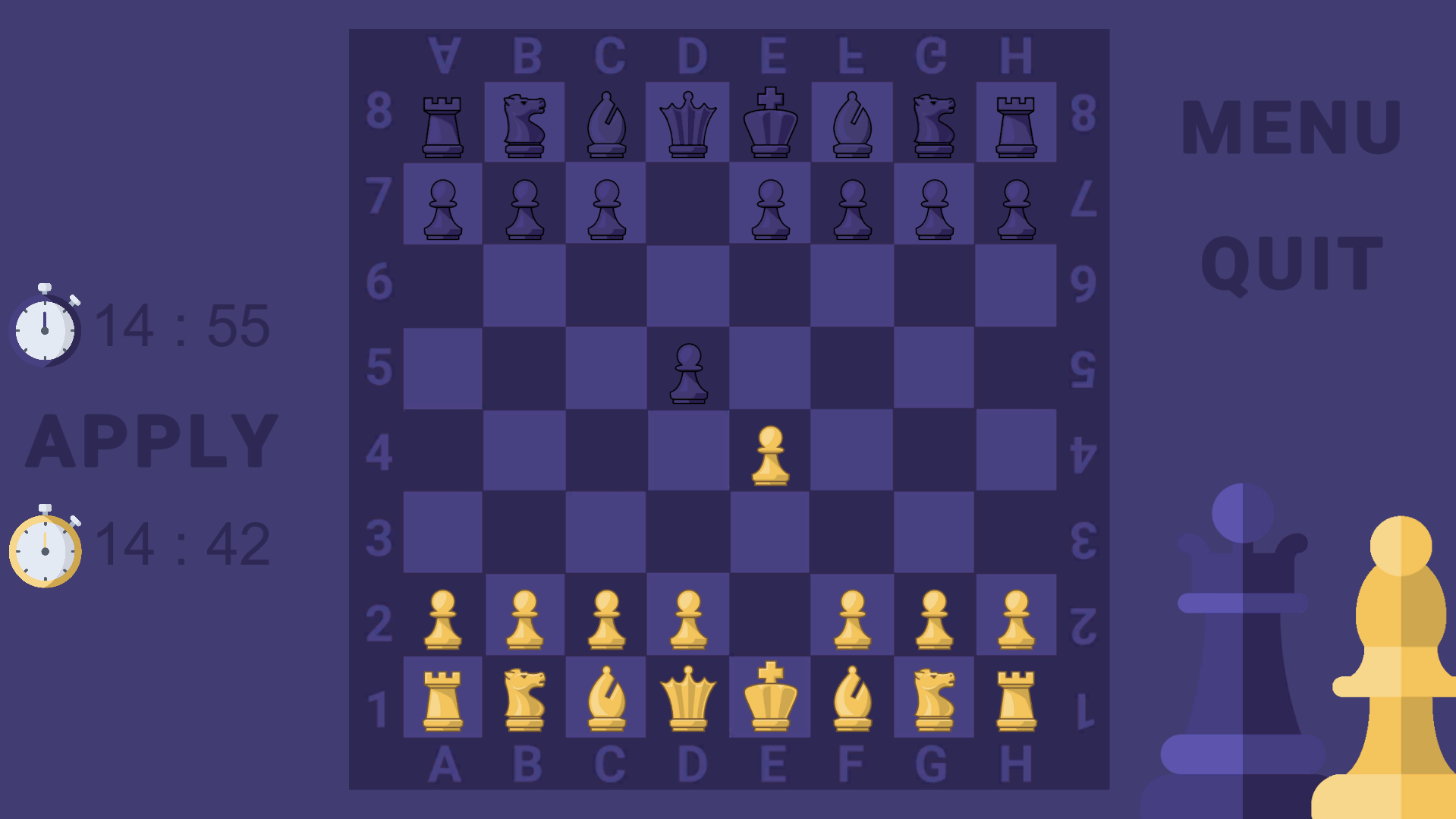


Рисунок 4.8 – Никто не пострадал.

4.2 Вывод по главе

В данной главе была показана работа программного средства. Программное средство реализует все функции, которые были заданы в техническом задании. Также были решены верно все контрольные примеры. Программное средство работает стабильно.

ЗАКЛЮЧЕНИЕ

В рамках данной работы было разработано программное средство, выполняющее функции приложения «Шахматы».

Данное программное средство полностью реализует задачи, сформулированные в техническом задании.

Реализованы алгоритмы:

* Генерации доски.
* Выполнения ходов.
* Графического представления шахмат.

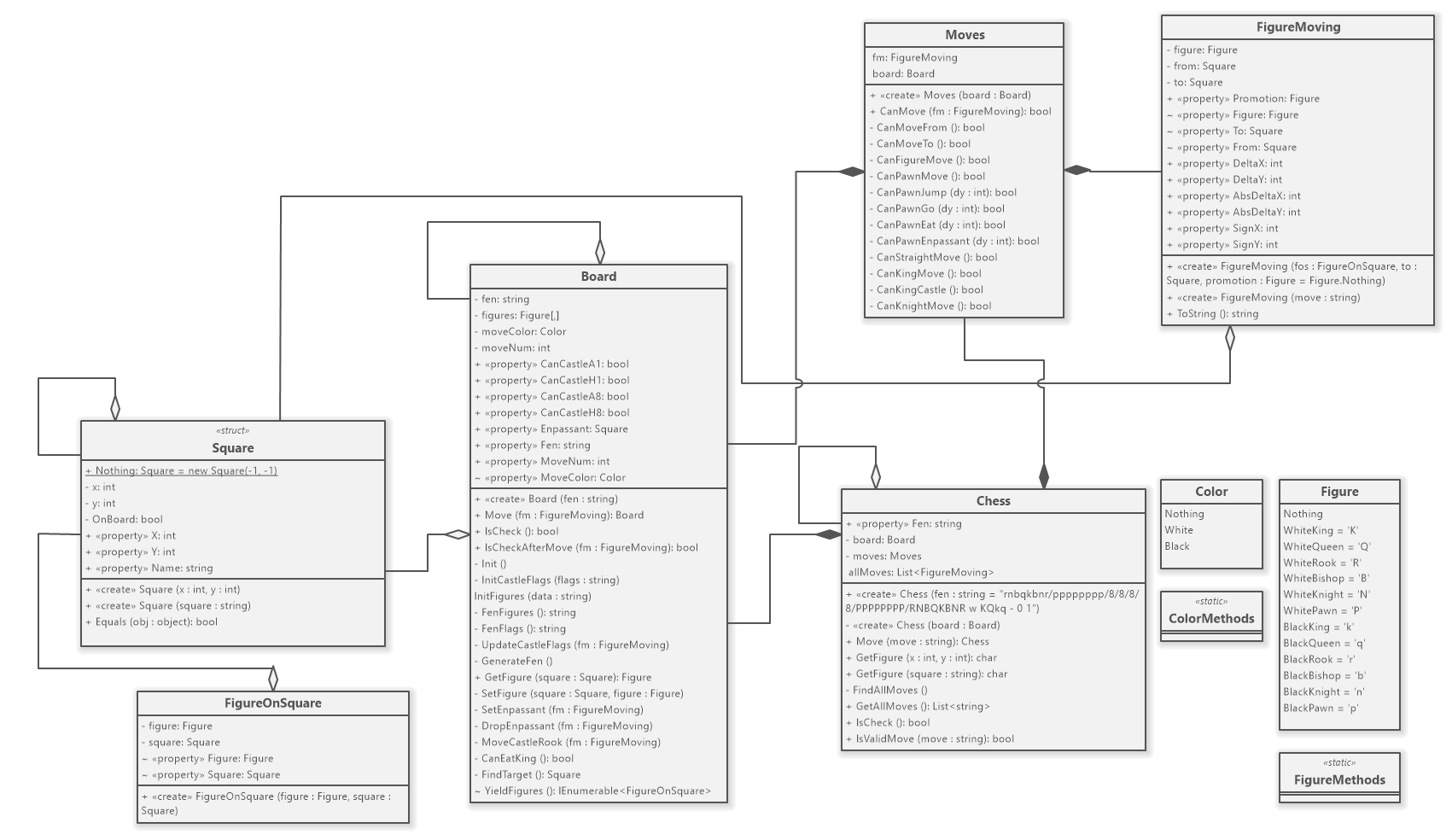
Таким образом была решена поставленная задача и выполнена проверка корректной работоспособности программного средства.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

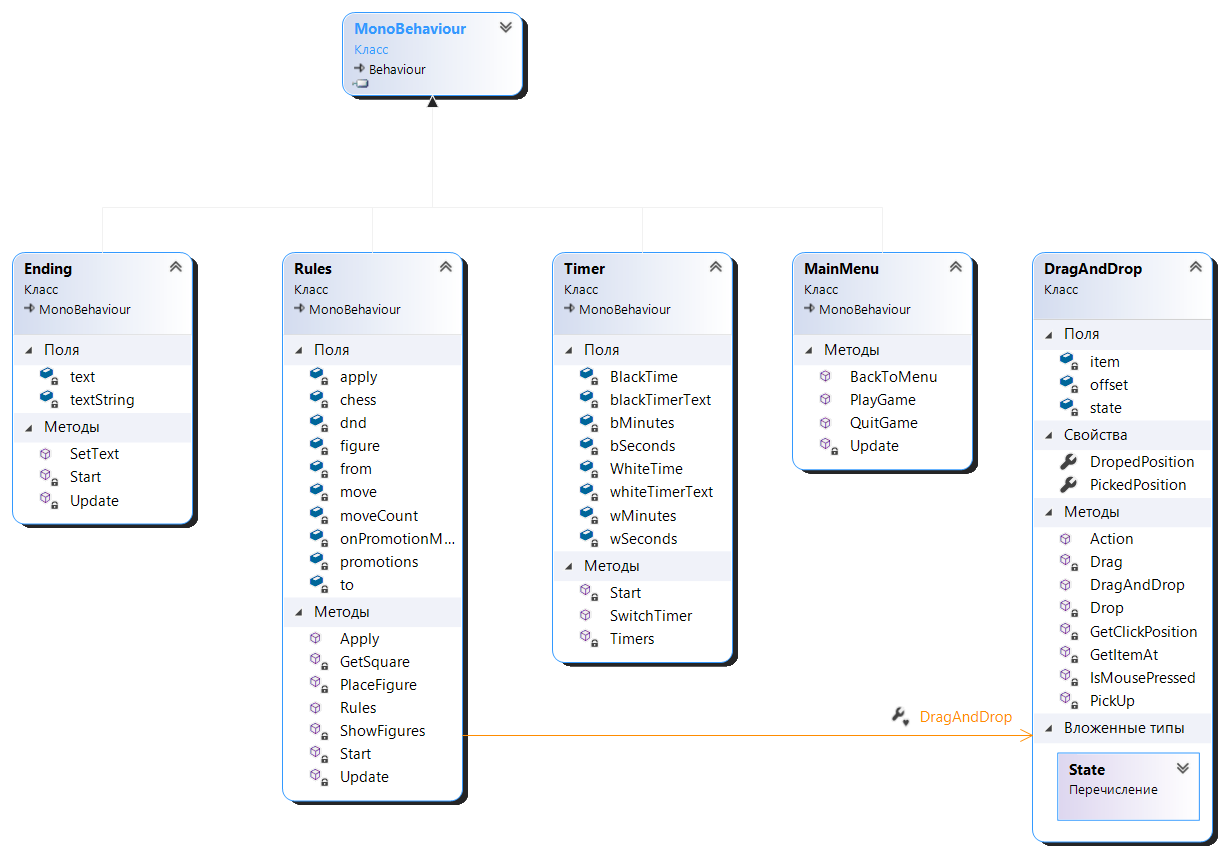
1. Шахматы // Большая советская энциклопедия : [в 30 т.] / гл. ред. А. М. Прохоров. — 3-е изд. — М. : Советская энциклопедия, 1969—1978.
2. Электронный ресурс «Правила игры в шахматы ФИДЕ» <https://handbook.fide.com/files/handbook/LawsOfChess2018Russian.pdf>
3. Либерти Д. Язык программирования C# // Программирование на C#. — Санкт-Петербург. — 2003: Символ-Плюс, 2003.
4. Джон Скит. C# для профессионалов: тонкости программирования, 3-е издание, новый перевод = C# in Depth, 3rd ed.. — М.: «Вильямс», 2014. — 608 с.

ПРИЛОЖЕНИЕ А

UML-диаграмма классов библиотеки ChessLib



UML-диаграмма классов приложения ClassicChessGame



ПРИЛОЖЕНИЕ B

Исходный код программы

using System.Collections.Generic;

/// <summary>

/// The main public chess class needed to provide the basic logic of the game and the bundles of all components.

/// Implements a public interaction interface.

///

/// Public methods:

/// Chess - Creates a board according to the given FEN

/// Move - On a given move generates a new board with the completed move.

/// GetFigure - Returns a figure standing on this square by a given move or coordinates

/// IsCheck - Check for Check

///

/// </summary>

namespace ChessLib

{

public class Chess

{

public string Fen { get; private set; }

private Board board;

private Moves moves;

List<FigureMoving> allMoves;

/// <summary>

/// Constructor generating the initial board

/// </summary>

/// <param name="fen">String in international format FEN</param>

public Chess(string fen = "rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1")

{

Fen = fen;

board = new Board(fen);

moves = new Moves(board);

}

private Chess(Board board)

{

this.board = board;

Fen = board.Fen;

moves = new Moves(board);

}

/// <summary>

/// Method for moving a figure

/// </summary>

/// <param name="move">String motion specification in (\*figure\*\*SquareFrom\*\*SquareTo\*) format.

/// For example: Pe2e4 - WhitePawn from e2 to e4

/// </param>

/// <returns>An instance of the Chess class with the move made.</returns>

public Chess Move(string move)

{

if (!IsValidMove(move)) return this;

Board nextBoard = board.Move(new FigureMoving(move));

return new Chess(nextBoard);

}

/// <summary>

/// Method for getting a figure on a square

/// </summary>

/// <param name="x">int coord of x</param>

/// <param name="y">int coord of y</param>

/// <returns>Char name of figure</returns>

public char GetFigure(int x, int y)

{

Square square = new Square(x, y);

Figure fig = board.GetFigure(square);

return fig == Figure.Nothing ? '.' : (char)fig;

}

/// <summary>

/// Method for getting a figure on a square

/// </summary>

/// <param name="square">coord of a square in ('a'-'h'+1-8) format. </param>

/// <returns>Char name of figure</returns>

public char GetFigure(string square)

{

Square sq = new Square(square);

Figure fig = board.GetFigure(sq);

return fig == Figure.Nothing ? '.' : (char)fig;

}

/// <summary>

/// Private method to find all possible moves for all pieces.

/// </summary>

private void FindAllMoves()

{

allMoves = new List<FigureMoving>();

foreach (FigureOnSquare fs in board.YieldFigures())

foreach (Square to in Square.YieldSquares())

foreach (Figure promotion in fs.Figure.YieldPromotion(to))

{

FigureMoving fm = new FigureMoving(fs, to, promotion);

if (moves.CanMove(fm) && !board.IsCheckAfterMove(fm)) allMoves.Add(fm);

}

}

/// <summary>

/// Public method for getting all possible moves.

/// </summary>

/// <returns>Return List of string with all moves</returns>

public List<string> GetAllMoves()

{

FindAllMoves();

List<string> list = new List<string>();

foreach (FigureMoving fm in allMoves)

list.Add(fm.ToString());

return list;

}

public bool IsCheck()

{

return board.IsCheck();

}

public bool IsValidMove(string move)

{

FigureMoving fm = new FigureMoving(move);

if (!moves.CanMove(fm) || board.IsCheckAfterMove(fm)) return false;

return true;

}

}

}