МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева (Самарский университет)»

Институт информатики и кибернетики

Кафедра программных систем

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту по дисциплине «Объектная распределённая обработка» по теме «Распределённое клиент-серверное приложение «Игра «Реверси»

Обучающийся  В.Д. Гижевская

Руководитель О.А. Гордеева

Самара 2022

ЗАДАНИЕ

Написать распределённое клиент-серверное приложение, используя технологию сокетных соединений. Приложение реализует игру «Реверси» со следующими правилами:

– в игре присутствует игрок-пользователь и игрок-компьютер;

– игра имеет поле размером 8x8 и 64 двусторонних фишки, одна сторона которых чёрная, другая – белая;

– в начале игры в центре доски выставляются 4 фишки, две белых и две черных, по диагонали;

– пользователь играет чёрной стороной и начинает игру, компьютер играет белой стороной, ходы делаются по очереди;

– игрок ставит фишку своей стороной так, чтобы между ней и уже имеющейся фишкой того же цвета находился непрерывный ряд фишек соперника, при этом запертые фишки соперника переворачиваются;

– если хода нет, то игрок пропускает свой ход;

– игра продолжается до окончания фишек или ходов;

– выигравшим объявляется тот игрок, фишек чьего цвета на поле больше.

РЕФЕРАТ

Пояснительная записка 48 с, 14 рисунков, 1 таблица, 19 источников, 1 приложение.

РЕВЕРСИ, ИГРА, ФИШКА, ИГРОВОЕ ПОЛЕ, КЛЕТКА, КОМПЬЮТЕР, ПОЛЬЗОВАТЕЛЬ, КЛИЕНТ-СЕРВЕРНОЕ ПРИЛОЖЕНИЕ, СОКЕТ.

Во время курсового проектирования разработаны алгоритмы и соответствующая им программа игры «Реверси», с помощью которой пользователь сможет играть против компьютера. Визуализация происходит на клиентской части системы, а весь игровой процесс – на серверной.

Программа написана на языке IntelliJ IDEA Community Edition и функционирует под управлением операционной системы Windows 7 и выше.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Реверси (другое название – оте́лло) – настольная игра для двух человек на доске 8 на 8 клеток. [1].

В настольную игру “Реверси” люди играют уже более ста лет. Это далеко не беспечное развлечение, а серьёзный турнир, в котором требуется логическое мышление и глубокий просчёт последующих вариаций ходов [2].

В “Реверси” играли ещё во второй половине 19 века. Популяризации игры, придуманной Джоном Моллетом, во многом поспособствовала публикация в известном лондонском журнале другим английским джентльменом – мистером Вотерменом. Он несколько преобразил форму, оставив неизменными механику и стратегию. Игра вошла не только в светские лондонские салоны, но и почти достигла соревновательного уровня шахмат и шашек. В Европе стали проводиться целые турниры, но пик популярности сошёл на “нет” по неизвестной причине.

Понадобилось почти сто лет, чтобы “Реверси” вновь заявила о себе. Реинкарнация произошла в Японии – Страна восходящего солнца подарила миру “Отелло”. По сути это та же “Реверси” с другим названием. Она быстро завоевала любовь американцев и европейцев. С тех пор ежегодно проводятся турниры и международные чемпионаты, по накалу не менее захватывающие, чем шахматные соревнования. Русский вариант стал популярным в СССР в 80-е годы. За этой настолкой буквально охотились не только профессионалы, но и семьи, в которых по вечерам устраивались мини-турниры между поколениями [2].

Во время курсового проектирования написать распределённое клиент-серверное приложение игра «Реверси», с помощью которого игрок сможет составлять горизонтальные, вертикальные и диагональные линии своим цветом, чтобы замкнуть фишки компьютера.

Разработка системы будет производиться по технологии быстрой разработки приложений RAD (Rapid Application Development), которая поддерживается методологией структурного проектирования и включает

элементы объектно-ориентированного проектирования и анализа предметной области [3].

При проектировании системы будут использоваться методология ООАП (Object-Oriented Analysis/Design), в основу которой положена объектно-ориентированная методология представления предметной области в виде объектов, являющихся экземплярами соответствующих классов, и язык моделирования UML (Unified Modeling Language), который является стандартным инструментом для разработки «чертежей» программного обеспечения [4].

1 Описание и анализ предметной области

Под предметной областью (application domain) принято понимать ту часть реального мира, которая имеет существенное значение или непосредственное отношение к процессу функционирования программы. Другими словами, предметная область включает в себя только те объекты и взаимосвязи между ними, которые необходимы для описания требований и условий решения некоторой задачи [5].

1.1 Описание игры «Реверси»

Реверси – настольная игра, в которой необходимо захватывать фишки оппонента, окружая их своими и переворачивая его фишки получать фишки своего цвета. Игрок с наибольшим количеством фишек своего цвета на доске побеждает в игре. На рисунке 1 приведён внешний вид игры «Реверси» [6].

Хотя игру часто сравнивают с шахматами, версий возможной расстановки в игре гораздо меньше. В настольной игре перевёртыши (ещё одно из названий реверси) так же выделяют три этапа сражения: дебют (или начало), миттельшпиль (обозначение средней части игры) и эндшпиль (конец). Из-за ограниченного выбора дебюта, играть в набор проще, ведь даже новичку не понадобится время на запоминание этих раскладов. Самой сложной фазой игры считается миттельшпиль, ведь ограничений по ходам на нем гораздо меньше, чем в остальных частях сражения. Именно в этот момент участник закрепляет свои позиции или меняет ситуацию на доске в свою пользу. Даже если в середине игры участнику не удалось вырваться из отстающих, шанс ещё остаётся в конце игры. Чтобы стать победителем в конце, нужно делать каждый ход взвешенно и обдумывая возможные последствия. Любое ошибочное действие может лишить игрока преимущества. Вступление в эндшпиль зависит от окончания миттельшпиля. Именно необходимость просчитывать возможные варианты расстановки и тактики для победы делает компьютеры такими хорошими игроками: они могут найти все версии в разы быстрее, чем это делает человек.

Рисунок 1 – Внешний вид игры «Реверси»

Правила игры:

– Перед игрой в центр игрового поля ставятся четыре фишки (2 белые и 2 черные).

– Первый ход делает игрок черных фишек.

– Игрок размещает фишки с верхом своего цвета.

– Фишку необходимо положить на одну из полей своего цвета таким образом, чтобы между фишкой игрока и уже стоящей фишкой этого же цвета, находился непрерывный ряд фишек оппонента (вертикально, диагонально или горизонтально). То есть ряд фишек оппонента должен быть закрыт фишками игрока противоположного цвета. При этом все фишки оппонента, которые игрок закрыл, переворачиваются на другую сторону. То есть переходят к ходившему игроку.

– Если при ходе игрока закрываются больше одного ряда фишек оппонента, то все «закрытые» фишки в рядах переворачиваются и переходят к ходившему игроку.

– Любой из игроков может ходить так, как хочет.

– Отказываться от хода запрещается.

– Если нет возможности осуществить ход, то ход переходит оппоненту.

– Игра завершается, когда на игровом поле будут выставлены все фишки.

– Игра завершается, когда ни один игрок не может сделать ход.

– По завершению игры осуществляется подсчёт белых и черных фишек. У кого фишек на игровом поел больше, тот и победитель.

– Если на игровом поле черных и белых фишек одинаковое количество, то объявляется ничья [6].

1.2 Постановка задачи

Во время курсового проектирования написать распределённое клиент-серверное приложение игра «Реверси», с помощью которого игрок сможет составлять горизонтальные, вертикальные и диагональные линии своим цветом, чтобы замкнуть фишки компьютера.

В системе должно быть реализовано две роли игрока: пользователь и компьютер.

Основная функция игрока – окружая фишки оппонента своими, захватить их, перевернув фишки компьютера и получив фишки своего цвета. Игрок с наибольшим количеством фишек своего цвета на доске побеждает в игре.

Процесс игры подразумевает под собой выбор места на поле для выставления своей фишки. При постановке фишки в выбранное место, фишки оппонента, попавшие под горизонтальную, вертикальную или диагональную прямую должны перевернуться.

В системе также должна быть обеспечена возможность получения справочной информации как о самой системе, так и предоставляемых ею возможностях.

Таким образом, системы должна решать следующие задачи:

1) общесистемные функции:

- автоматическое составление игры;
- визуализация процесса игры;
- подсчёт количества фишек каждого игрока;
- выдача результата игры;
- выдача справочной информации о системе.

2) функции пользователя:

- выбор места выставления своей фишки;
- просмотр справочной информации о системе.

2 Проектирование системы

2.1 Выбор и обоснование архитектуры системы

Архитектура – совокупность важнейших решений об организации программной системы. Архитектура включает [7]:

– выбор структурных элементов и их интерфейсов, с помощью которых составлена система, а также их поведения в рамках сотрудничества структурных элементов;

– соединение выбранных элементов структуры и поведения во всё более крупные системы;

– архитектурный стиль, который направляет всю организацию – все элементы, их интерфейсы, их сотрудничество и их соединение [7].

Клиент-серверная архитектура–наиболее распространённая структура приложений в Интернете. В этой архитектуре клиенты (т.е. персональные компьютеры, устройства Интернета вещей и т. д.) сначала запрашивают ресурсы с сервера. Затем сервер отправляет обратно соответствующие ответы на запросы клиентов. Чтобы это произошло, должен быть какой-то механизм, реализованный как на стороне клиента, так и на стороне сервера, который поддерживает эту сетевую транзакцию. Этот механизм называется коммуникацией посредством сокетов [8].

Почти каждое приложение, которое полагается на сетевые операции, такие как извлечение данных с удалённых серверов и загрузка файлов на сервер, широко использует сокеты.

Сокет – это программная (логическая) конечная точка, устанавливающая двунаправленную коммуникацию между сервером и одной или несколькими клиентскими программами. Сокет – это нечто “программное”. Другими словами, сокет не существует на физическом уровне. Прикладное программное обеспечение определяет сокет так, чтобы он использовал порты на основном компьютере для его реализации. Это позволяет программистам комфортно работать с низкоуровневыми деталями

сетевых коммуникаций, такими как порты, маршрутизация и т. д., внутри прикладного кода. Существует два типа сокетов: TCP и UDP. В данной работе для соединения выбран TCP-сокет.

TCP-сокет устанавливает связь между клиентом и сервером в несколько этапов (см. рисунок 2):

– Socket() – на сервере создается конечная точка для коммуникации.

– Bind() – сокету присваивается уникальный номер и для него резервируется уникальная комбинации IP-адреса и порта.

– Listen() – после создания сокета сервер ожидает подключения клиента.

– Accept() – сервер получает запрос на подключение от клиентского сокета.

– Connect() – клиент и сервер соединены друг с другом.

– Send()/Recieve() – обмен данными между клиентом и сервером.

– Close() – после обмена данными сервер и клиент разрывают соединение [8].

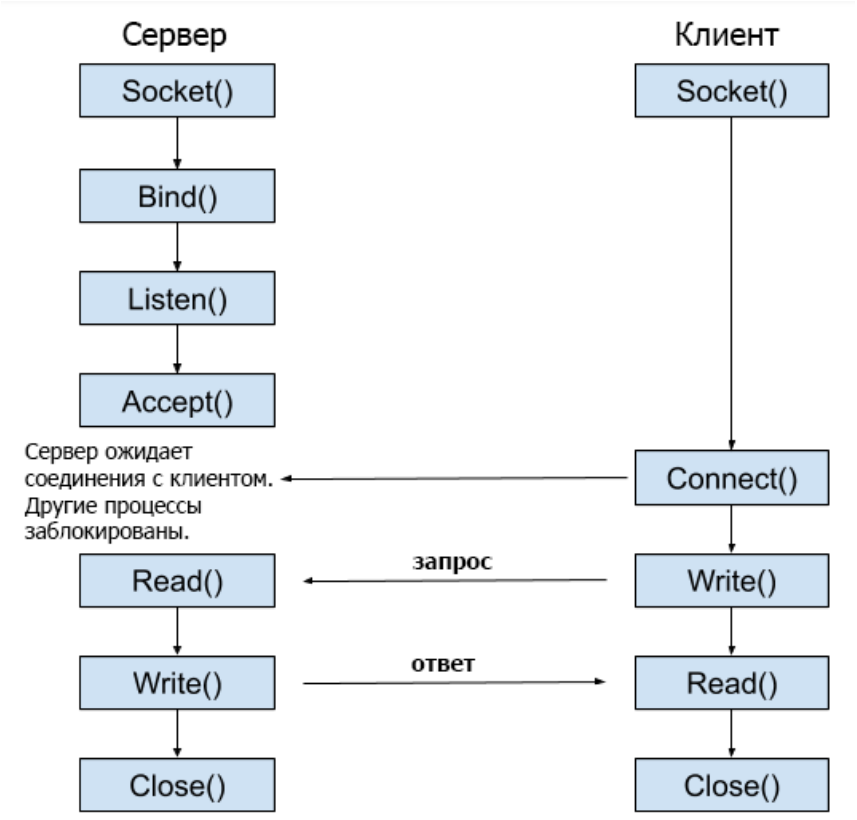

Рисунок 2 – Блок-схема коммуникации TCP-сокета

2.2 Структурная схема системы

Система (греч. «составленное из частей», «соединение» от «соединяю») – множество элементов, находящихся в отношениях и связях друг с другом, которое образует определённую целостность, единство [9].

Как следует из определения, отличительным (главным свойством) системы является её целостность: комплекс объектов, рассматриваемых в качестве системы, должен обладать общими свойствами и поведением. Очевидно, необходимо рассматривать и связи системы с внешней средой. В самом общем случае понятие «система» характеризуется [10]:

- наличием множества элементов;
- наличием связей между ними;
- целостным характером данного устройства или процесса.

Структурная схема – это схема, определяющая основные функциональные части изделия, их назначение и взаимосвязи [11].

Сущность структурного подхода к разработке системы заключается в её декомпозиции (разбиении) на автоматизируемые функции: система разбивается на функциональные подсистемы, которые в свою очередь делятся на подфункции, подразделяемые на задачи и так далее. Процесс разбиения продолжается вплоть до конкретных процедур. При этом автоматизируемая система сохраняет целостное представление, в котором все составляющие компоненты взаимосвязаны [12].

На рисунке 3 приведена структурная схема разрабатываемой системы, разделяется клиентскую и серверную часть. Взаимодействие между ними осуществляется по TCP протоколу.

В состав клиентской части входит:

– Подсистема взаимодействия с сервером, которая осуществляет установку соединения с сервером, формирование и отправку запросов.

– Подсистема визуализации, которая отображает игровое поле и результаты действий пользователя на экран.

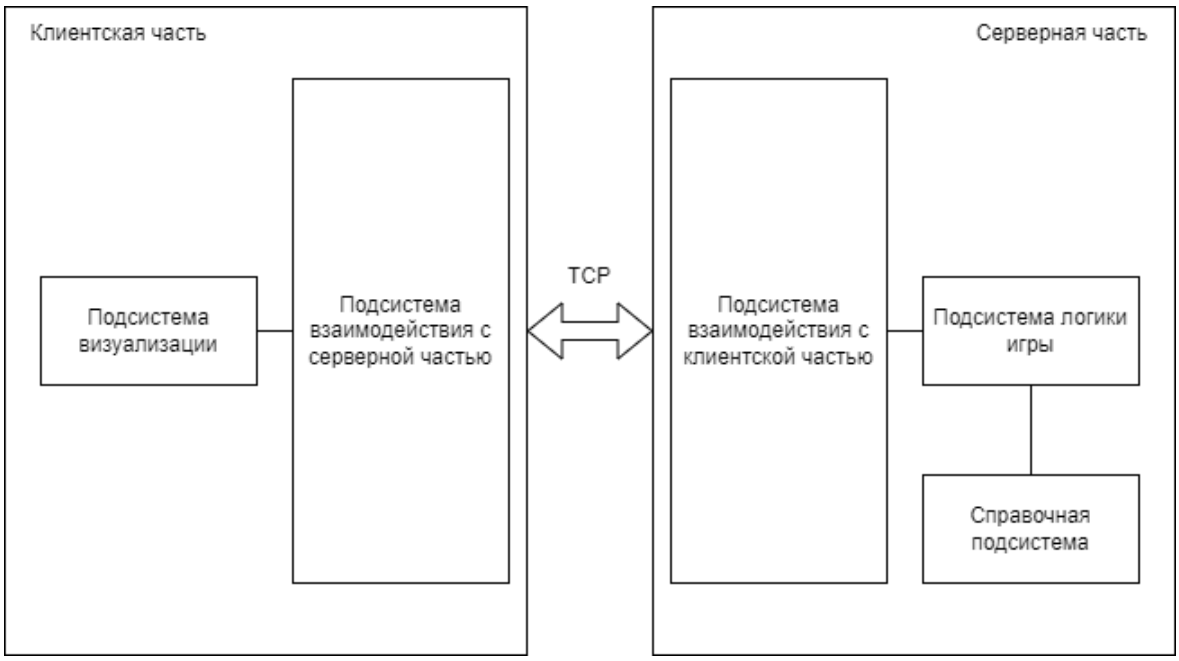

Рисунок 3 – Структурная схема системы

В состав серверной части входит:

– Подсистема взаимодействия с клиентом, которая осуществляет приём данных с клиента и передачу их на обработку.

– Подсистема логики игры, которая отвечает за игровой процесс.

– Справочная подсистема, которая содержит сведения о системе (правила игры) и об её разработчиках.

2.3 Разработка прототипа интерфейса пользователя системы

Интерфейс пользователя является одним из важнейших элементов программы, это та часть программы, которая находится у всех на виду. Недочёты в пользовательском интерфейсе могут серьёзно испортить впечатление даже о самых многофункциональных программах. Именно поэтому разработке и проектированию пользовательского интерфейса нужно уделять особое внимание [13].

Процесс создания интерфейса начинается с определения целей проекта, а также внутренних и внешние обстоятельств, которые необходимо принять во внимание.

Для того чтобы правильно расставить приоритеты, необходимо учитывать [13]:

− опыт работы пользователей с компьютером, типовые ситуации использования;

− какая информация необходима и когда, какие результаты должны быть получены;

− технологию разработки и платформа, на которой будут работать пользователи.

На рисунке 4 приведён прототип главной формы приложения. На данной форме будет проходить весь игровой процесс. Слева будет находится игровое поле 8x8, где игрок сможет выставлять свои фишки в ячейки (возможные ходы будут подсвечиваться). Справа будет отображаться результаты игры (количество фишек каждого игрока на поле). При нажатии на кнопку «Новая игра», текущая игра сбросится, и игрок сможет начать всё заново. Нажав на кнопку «Правила», пользователь сможет получить информацию об процессе игры и её правилах (см. рисунок 5).

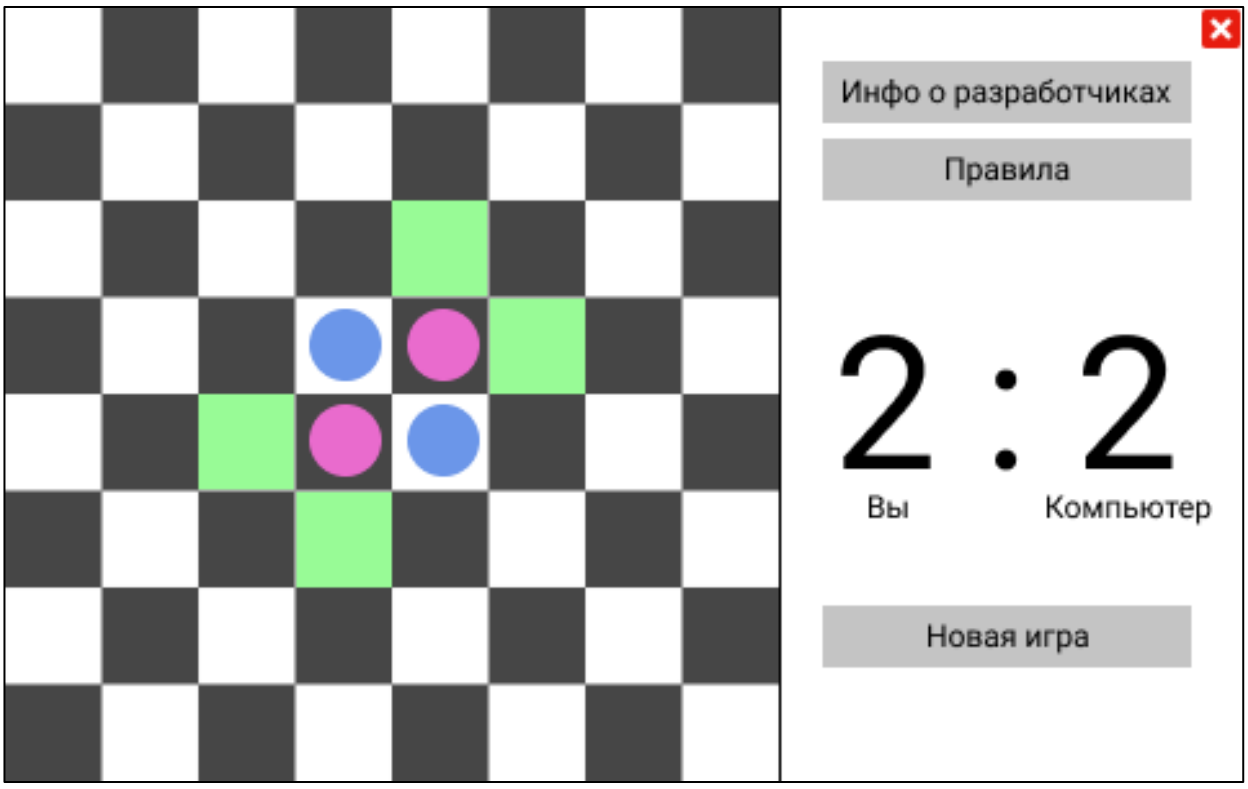

Рисунок 4 – Прототип главной формы приложения

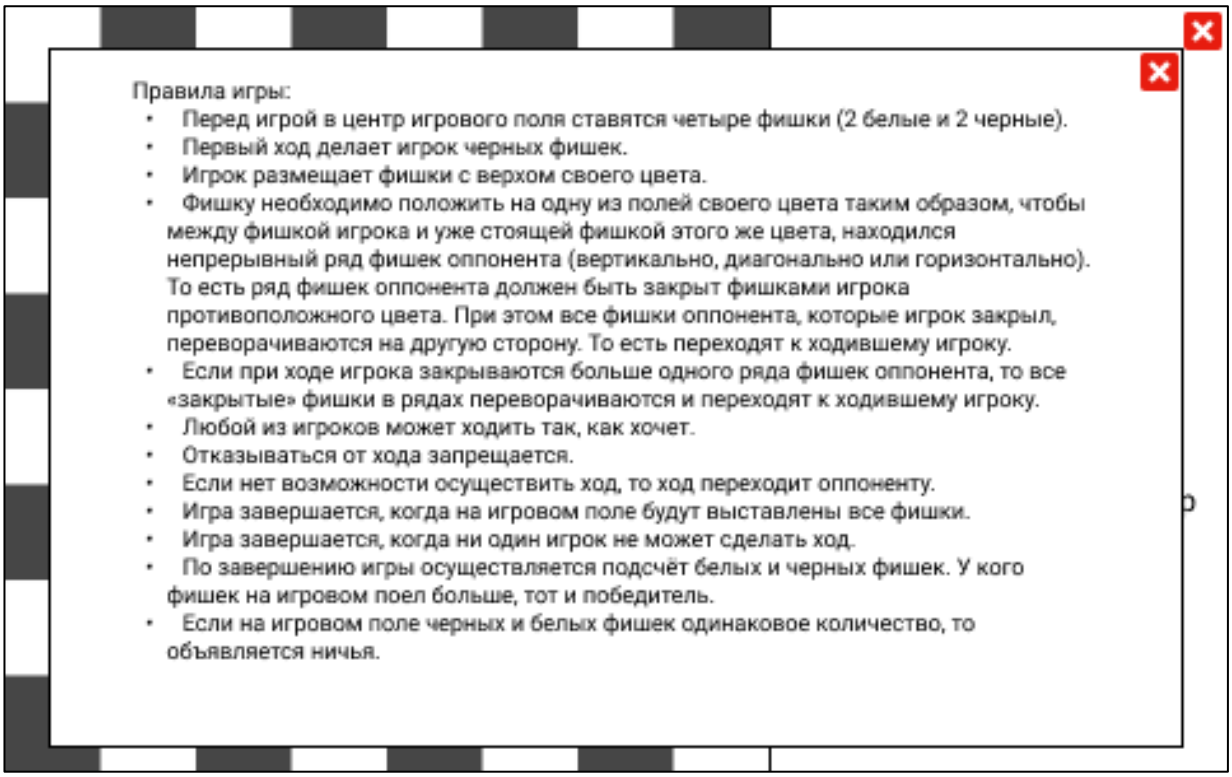

Рисунок 5 – Прототип экранной формы «Правила игры»

Кнопка «Информация о разработчиках» позволит пользователю узнать о создателях данной системы (см. рисунок 6).

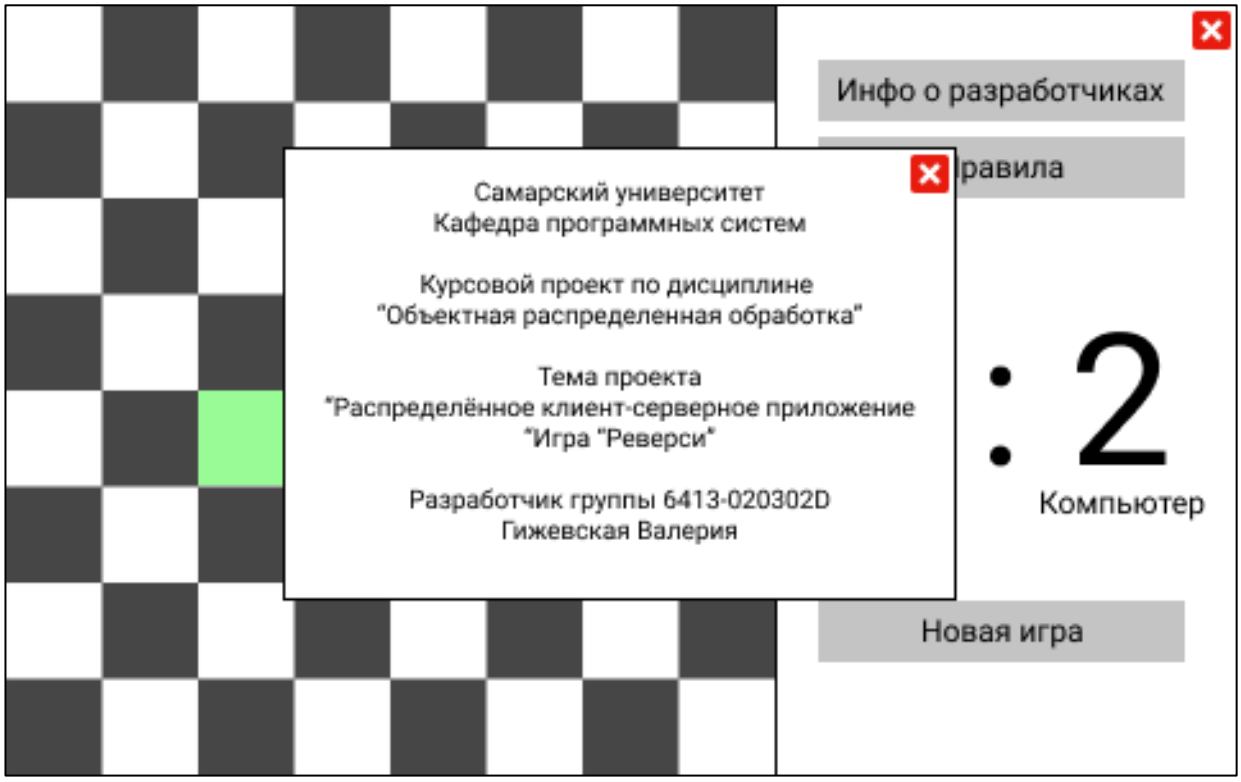

Рисунок 6 – Прототип экранной формы «Информация о разработчиках»

Для того, чтобы пользователю выйти из системы, ему будет необходимо закрыть окно.

2.4 Разработка информационно-логического проекта системы

Цель инфологического проектирования является обеспечение наиболее естественных для человека способов сбора и представления той информации, которую предполагается хранить в создаваемой базе данных. Поэтому инфологическая модель данных построена по аналогии с естественным языком. Основными конструктивными элементами инфологических моделей являются сущности, связи между ними и их атрибуты [14].

2.4.1 Язык UML

Для специфицирования (построения точных, недвусмысленных и полных моделей) системы и её документирования используется унифицированный язык моделирования UML.

UML (англ. Unified Modeling Language – унифицированный язык моделирования) – язык графического описания для объектного моделирования в области разработки программного обеспечения. UML является языком широкого профиля, это – открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой UML-моделью. UML был создан для определения, визуализации, проектирования и документирования, в основном, программных систем. UML не является языком программирования, но на основании UML-моделей возможна генерация кода [15].

Использование UML не ограничивается моделированием программного обеспечения. Его также используют для моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

UML позволяет разработчикам программного обеспечения достигнуть соглашения в графических обозначениях для представления общих понятий,

таких как класс, компонент, обобщение, агрегация и поведение и больше сконцентрироваться на проектировании и архитектуре.

2.4.2 Диаграмма вариантов использования

Диаграмма вариантов использования представляет собой наиболее общую концептуальную модель сложной системы, которая является исходной для построения всех остальных диаграмм. На ней изображаются отношения между актёрами и вариантами использования.

Актёр (actor) – согласованное множество ролей, которые играют внешние сущности по отношению к вариантам использования при взаимодействии с ними.

Вариант использования – внешняя спецификация последовательности действий, которые система или другая сущность могут выполнять в процессе взаимодействия с актёрами.

Цель спецификации варианта использования заключается в том, чтобы зафиксировать некоторый аспект или фрагмент поведения проектируемой системы без указания особенностей реализации данной функциональности. В этом смысле каждый вариант использования соответствует отдельному сервису, который предоставляет моделируемая система по запросу актёра. Сервис, который инициализируется по запросу актёра, должен представлять собой законченную последовательность действий. Это означает, что после того как система закончит обработку запроса актёра, она должна возвратиться в исходное состояние, в котором снова готова к выполнению следующих запросов.

На рисунке 7 приведена общая диаграмма вариантов использования. Пользователь может посмотреть информацию о разработчиках и правила игры. Также пользователь может управлять игровым процессом, а именно поставить фишку в возможное место или начать новую игру.

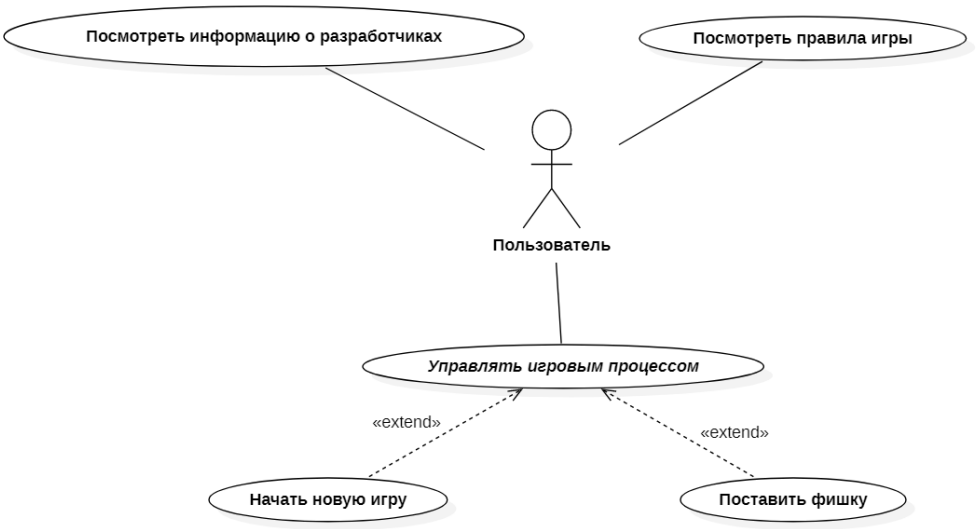

Рисунок 7 – Диаграмма вариантов использования системы

2.4.3 Диаграмма деятельности

Для моделирования процесса выполнения операций в языке UML используются диаграммы деятельности. Применяемая в них графическая нотация во многом похожа на нотацию диаграммы состояний, поскольку на диаграммах деятельности также присутствуют обозначения состояний и переходов. Отличие заключается в семантике состояний, которые используются для представления деятельности и действий, а также в отсутствии на переходах сигнатуры событий. Каждое состояние на диаграмме деятельности соответствует выполнению некой операции, а переход в следующее состояние происходит только после завершения выполнения этой операции. Диаграмма деятельности представляется в форме графа деятельности, вершинами которого являются состояния действия или деятельности, а дугами – переходы от одного состояния действия к другому.

Диаграммы деятельности – частный случай диаграмм состояний. Они позволяют реализовать в языке UML особенности процедурного и синхронного управления, обусловленного завершением внутренних действий и деятельности. В контексте языка UML деятельность представляет собой совокупность отдельных вычислений, выполняемых автоматом. При этом отдельные элементарные вычисления могут приводить к результату или

действию. На диаграмме деятельности отображается логика или последовательность перехода от одной деятельности к другой, при этом внимание фиксируется на результате деятельности. Сам же результат может привести к изменению состояния системы или возвращению некоторого значения [16].

На рисунке 8 приведена диаграмма деятельности системы. Система открывает главное окно, устанавливает фишки на начальное положение и подсвечивает возможные для хода ячейки, после чего пользователь может нажать на ячейку, система поставит фишки игрока и компьютера, и опять подсветит возможные варианты. Также пользователю доступен просмотр правил и информации о разработчиках. Система открывает эти окна по нажатию на соответствующие кнопки и закрывает по нажатию на кнопку «X». При нажатии на кнопку «Новая игра», система перейдёт в начально состояние. Чтобы выйти из системы, пользователю необходимо нажать кнопку «X» на главном окне.

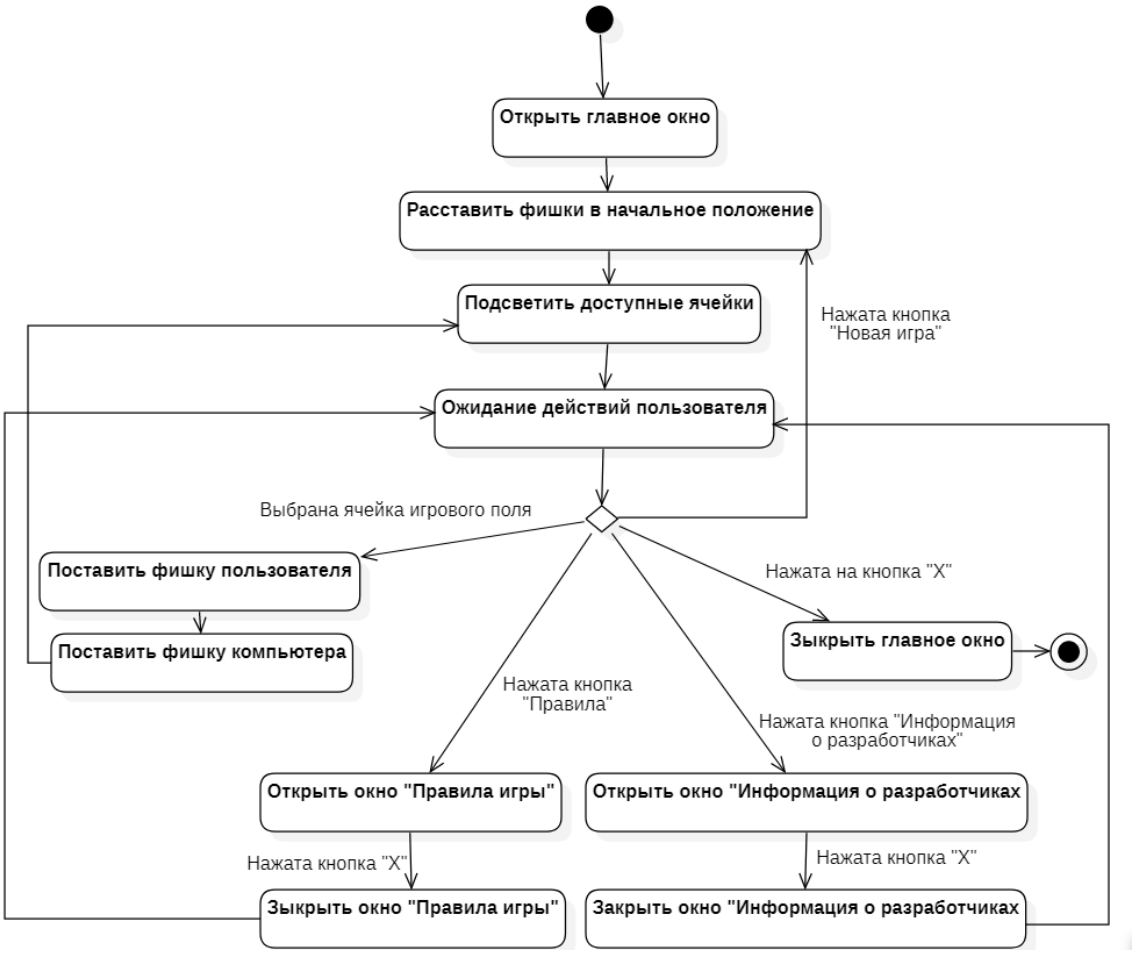

Рисунок 8 – Диаграмма деятельности системы

2.4.4 Диаграмма последовательности

Диаграмма последовательности UML (sequence diagram) – такая диаграмма, на которой показаны взаимодействия объектов, упорядоченные по времени их проявления. Основные элементы диаграммы последовательности это: обозначения объектов (прямоугольники), вертикальные линии, отображающие течение времени при деятельности объекта, и стрелки, показывающие выполнение действий объектами [17].

На рисунке 9 приведена диаграмма последовательности системы.

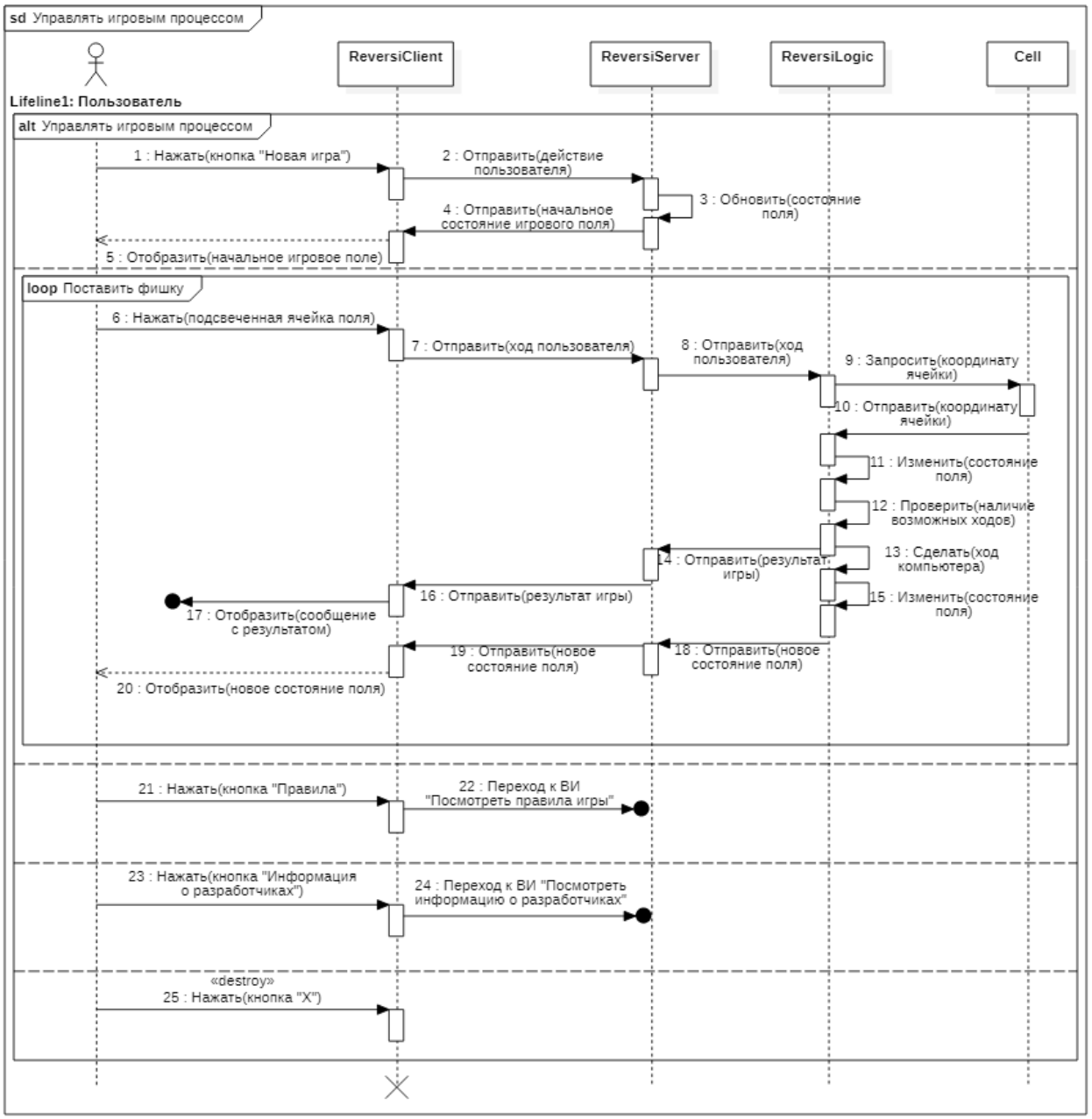

Рисунок 9 – Диаграмма последовательности системы

3 Реализация системы

3.1 Разработка и описание интерфейса пользователя

На рисунке 10 приведена главная форма приложения. На данной форме проходит весь игровой процесс. Слева находится игровое поле 8x8, где игрок может выставлять свои фишки в ячейки (возможные ходы подсвечиваются зелёным). Справа отображаются текущие результаты игры (количество фишек каждого игрока на поле). При нажатии на кнопку «Новая игра», текущая игра сбросится, и игрок сможет начать всё заново.

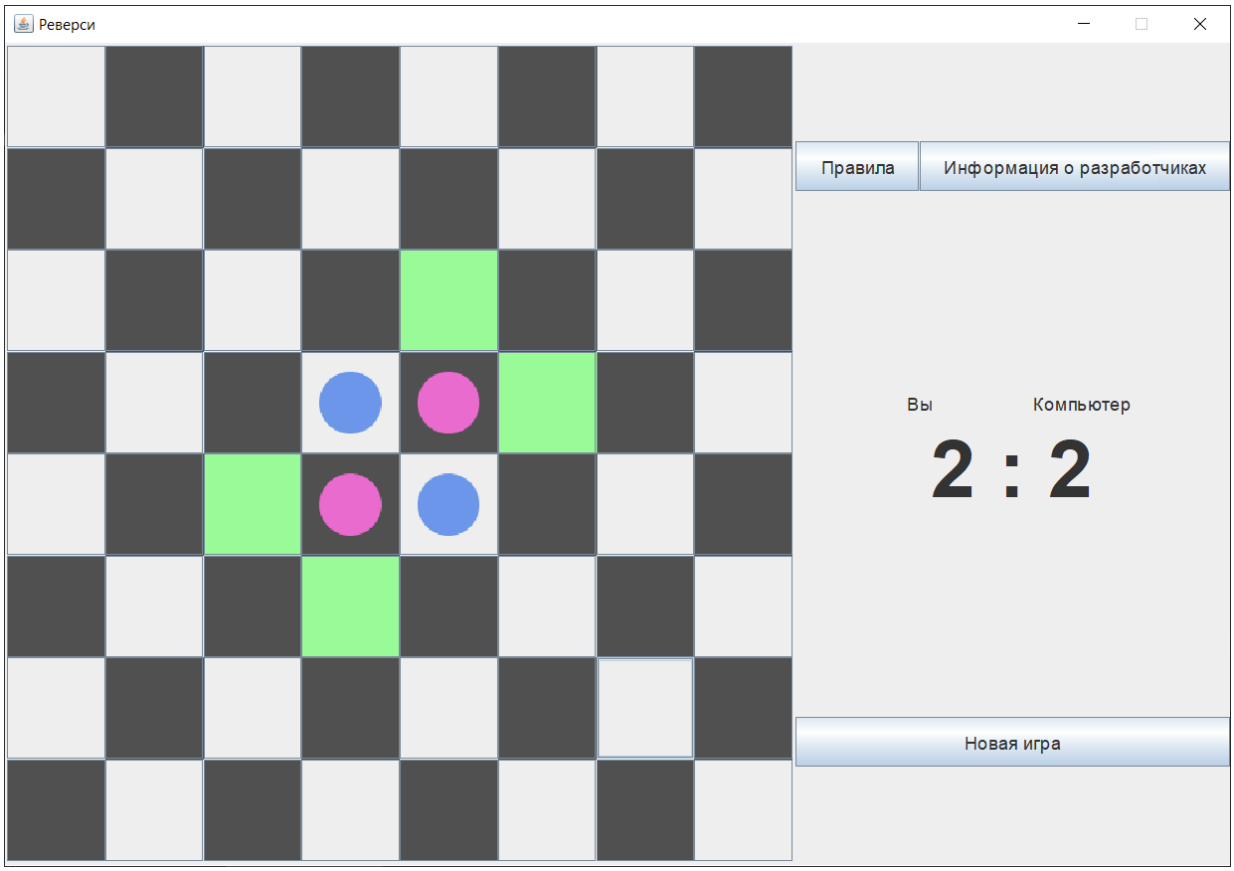

Рисунок 10 – Главное окно приложения

Нажав на кнопку «Правила», пользователь сможет получить информацию об процессе игры и её правилах (см. рисунок 11).

Нажав на кнопку «Информация о разработчиках», пользователь сможет узнать о создателях данной системы (см. рисунок 12).

После завершение игры, система выведет выпадающее окно с результатом игры. Пример этого окна можно приведён на рисунке 13.

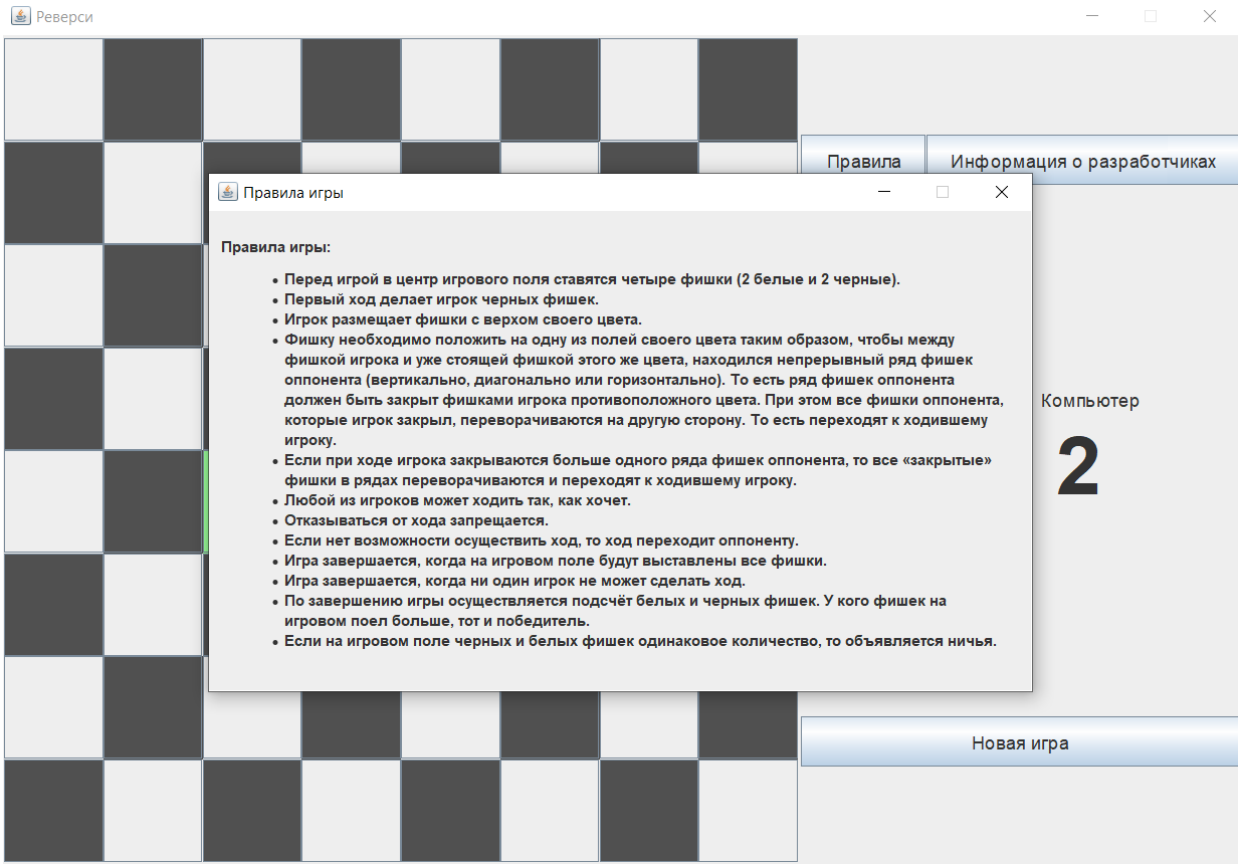

Рисунок 11 – Окно «Правила игры»

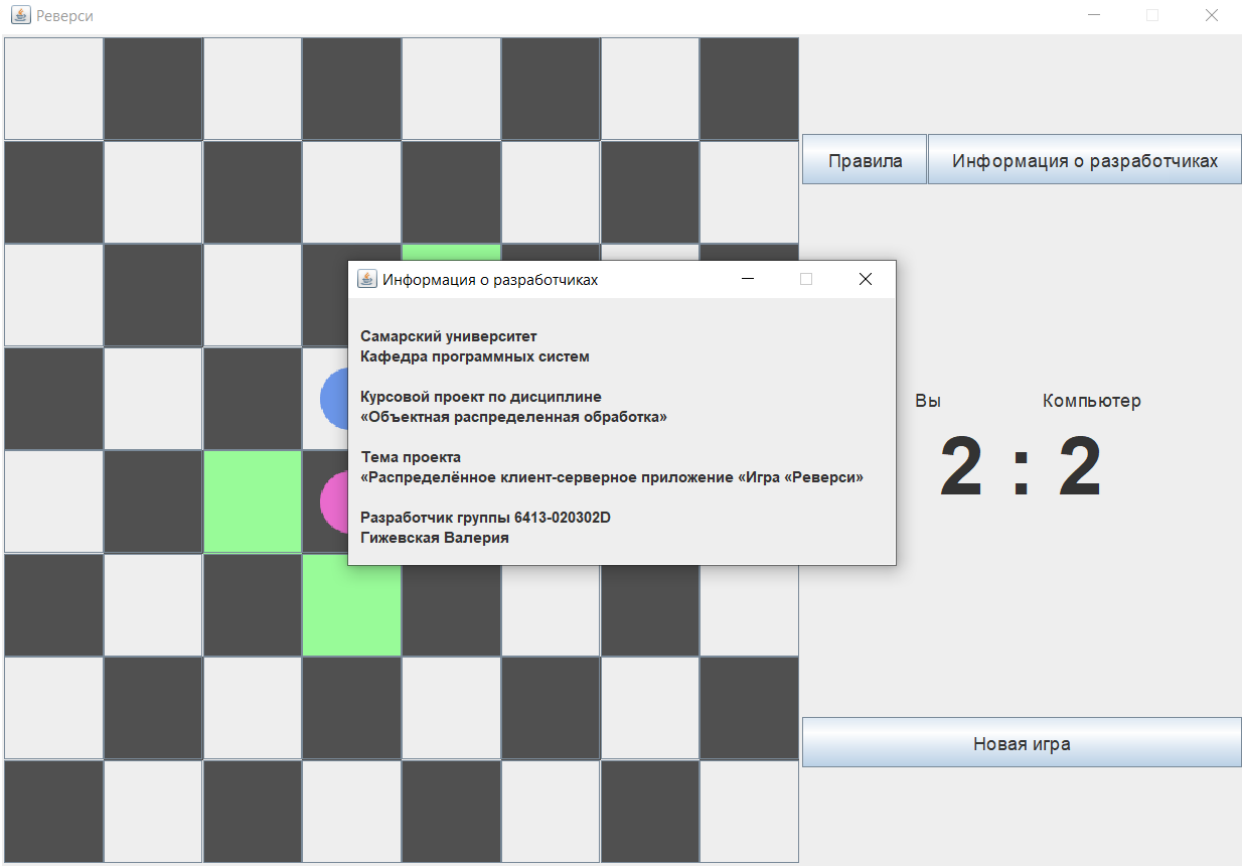

Рисунок 12 – Окно «Информация о разработчиках»

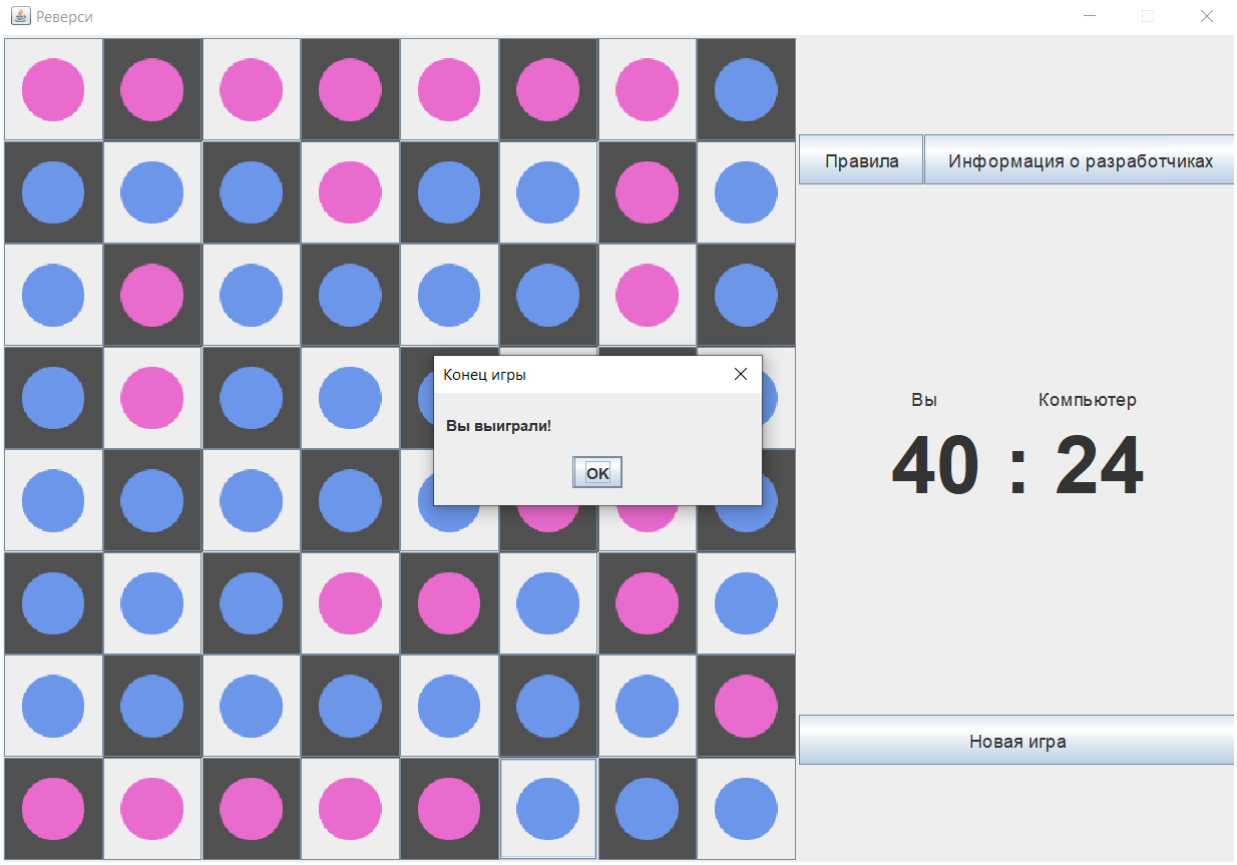

Рисунок 13 – Выпадающее окно с результатом игры

Для того, чтобы пользователю выйти из системы, ему необходимо закрыть главное окно, нажав на кнопку «X».

3.2 Диаграммы реализации

Диаграммы реализации предназначены для отображения состава компилируемых и выполняемых модулей системы, а также связей между ними. Диаграммы реализации разделяются на два конкретных вида: диаграммы компонентов (component diagrams) и диаграммы развёртывания (deployment diagrams) [18].

3.2.1 Диаграмма компонентов

Диаграмма компонентов описывает особенности физической реализации приложения, определяет архитектуру приложения и устанавливает зависимость между компонентами, в роли которых выступает исполняемый код. Диаграмма компонентов отображает общие зависимости

между компонентами. Основными графическими элементами диаграммы являются компоненты, интерфейсы и зависимости между ними [19].

Диаграмма компонентов отображена на рисунке 14. Компоненты, входящие в диаграмму представлены в таблице 1.

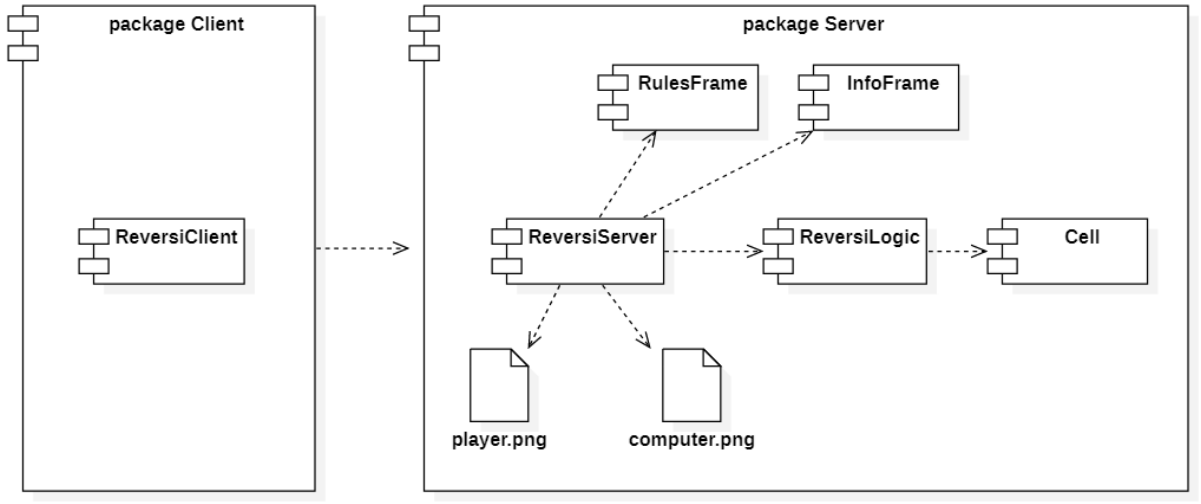

Рисунок 14 – Диаграмма компонентов системы

Таблица 1 – Основные компоненты системы

Название	Назначение
ReversiClient	Класс, предназначенный для получения и передачи сообщений серверу
ReversiServer	Класс, предназначенный для получения и передачи сообщений клиенту
ReversiLogic	Класс, определяющий логику работы игры
Cell	Класс, предназначенный для доступа к ячейке игрового поля
RulesFrame	Класс, предназначенный для вывода правил игры
InfoFrame	Класс, предназначенный для вывода информации о разработчиках

ЗАКЛЮЧЕНИЕ

В процессе выполнения курсового проекта было разработано распределённое клиент-серверное приложение «Игра «Реверси» с использованием технологий сокетного соединения.

В первом разделе приведены основные понятия предметной области, рассмотрена история и правила игры «Реверси». На основе проведённого анализа была сформулирована постановка задачи.

Во втором разделе была описана структурная схема системы, разработаны прототипы экранных форм системы. Также был разработан информационно-логический проект по методологии UML, в который вошли диаграммы вариантов использования, деятельности и последовательности.

В третьем разделе приведено описание интерфейса пользователя и диаграмма компонентов.

Разработанная система может использоваться пользователями разных возрастов для разнообразия своего досуга.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Реверси [Электронный ресурс] // Википедия: [сайт]. URL: https://ru.wikipedia.org/wiki/Реверси (дата обращения: 02.03.2022).

2 Как играть и выигрывать в игру “Реверси”? [Электронный ресурс] // NasIgra.ru: [сайт]. URL: https://nasigra.ru/igrovoe-pole/reversi (дата обращения: 05.03.2022).

3 Буч Г., Рамбо Д., Якобсон А. Язык UML. Руководство пользователя. Изд. 2-е. М.: ДМК Пресс, 2006. 546 с.

4 Буч Г., Рамбо Д., Якобсон А. Язык UML … С. 31.

5 Леоненков, А. В. Нотация и семантика языка UML [Электронный ресурс]/ А.В. Леоненков. – Интернет-университет информационных технологий. URL: http://www.intuit.ru/department/pl/umlbasics (дата обращения: 06.03.2022).

6 Настольная игра Реверси (Отелло) [Электронный ресурс] // Настолкофф: [сайт]. URL: https://nastolkoff.ru/taktika-i-strategiya/nastolnaya-igra-reversi-otello (дата обращения: 06.03.2022).

7 Анализ предметной области [Электронный ресурс] // Интуит: [сайт]. URL: http://www.intuit.ru/studies/courses/574/430/lecture/9749 (дата обращения: 08.03.2022).

8 Основы программирования TCP-сокетов на Java [Электронный ресурс] // NP: [сайт]. URL: https://nuancesprog.ru/p/8583/ (дата обращения: 08.03.2022).

9 Буч Г., Рамбо Д., Якобсон А. Язык UML … С. 496.

10 Большой Российский энциклопедический словарь. М.: БРЭ, 2003.

11 Автоматизация управления типовыми объектами производства [Текст]. - М.: КНИТУ, 2020 (дата обращения: 13.10.2021).

12 Структурный подход к проектированию ИС. [Электронный ресурс] // CITforum: [сайт]. URL: http://citforum.ru/database/case/glava2_1.shtml (дата обращения: 13.10.2021).

13 Разработка прототипа интерфейса пользователя системы [Электронный ресурс] // StudFiles: [сайт]. URL: https://studfile.net/preview/2114024/page:7/ (дата обращения: 01.11.2021).

14 Информационно-логическое проектирование. [Электронный ресурс] // Studwood.ru: [сайт]. URL: https://studwood.ru/1884856/informatika/informatsionno_logicheskoe_proektirovanie (дата обращения: 15.11.2021).

15 UML [Электронный ресурс] // Аналитик: [сайт]. URL: https://www.sites.google.com/site/infoprobusinessanalysis/project-definition/uml (дата обращения: 15.11.2021).

16 Диаграмма деятельности (activity diagram) [Электронный ресурс] // masters.donntu.org: [сайт]. URL: https://masters.donntu.org/2005/kita/shapovalova/library/uml2.html (дата обращения: 24.11.2021).

17 Диаграмма последовательности [Электронный ресурс] // StudFiles: [сайт]. URL: https://studfile.net/preview/5187971/page:4/ (дата обращения: 24.11.2021).

18 Диаграммы реализации [Электронный ресурс] // maksakov-sa.ru: [сайт]. URL: http://www.maksakov-sa.ru/ModelUML/DiagrReal/index.html (дата обращения: 23.12.2021).

19 Диаграмма компонентов (component diagram) [Электронный ресурс] // Студопедия: [сайт]. URL: https://studopedia.ru/6_33179_diagramma-klassov-class-diagram.html (дата обращения: 23.12.2021).

ПРИЛОЖЕНИЕ А

Листинг программы

ReversiClient

```
import javax.swing.*;
import java.awt.*;
import java.net.Socket;
public class ReversiClient extends JFrame{
    private static JPanel panel;
    public ReversiClient() {
        super("Реверси");
        setLayout(new BorderLayout());
        Socket socket = null;
        try {
            socket = new Socket("127.0.0.1", 1025);
        } catch (Exception e) {
            return;
        }
        panel = new ReversiServer(socket);
        add(panel, BorderLayout.CENTER);
        setSize(1000, 700);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setResizable(false);
        setVisible(true);
    }
    public static void main(String[] args) {
        new ReversiClient();
    }
}
```

ReversiServer

```
import java.awt.Color;
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.ArrayList;
import javax.imageio.ImageIO;
```

```
import javax.swing.ImageIcon;
public class ReversiServer extends JPanel {
    JPanel boardPanel;
    static JLabel score;
    static JButton[] cell;
    static ReversiLogic board;
    static ArrayList<ReversiLogic> arrReversi = new ArrayList<>();
    static JButton newGame;
    static JButton rules;
    static JButton info;
    static public int userScore = 2;
    static public int pcScore = 2;
    private static ReversiLogic start;
    private static int rows = 8;
    private static int cols = 8;
    public Image imgL, imgD;
    {
        try {
            imgL = ImageIO.read(getClass().getResource("images/computer.png"));
            imgD = ImageIO.read(getClass().getResource("images/player.png"));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    public ImageIcon imgLight = new ImageIcon(imgL);
    public ImageIcon imgDark = new ImageIcon(imgD);
    public Color darkGray = new Color(80, 80, 80);
    public Color lightGray = new Color(238, 238, 238);
    public Color green = new Color(152, 251, 152);
    Font font55 = new Font("Courier", Font.BOLD, 65);
    Font font15 = new Font("Courier", Font.TRUETYPE_FONT, 15);
    public Color ColorCheck(int col, int row) {
        Color color = darkGray;
        if (((col % 2 == 0) && (row % 2 == 0)) || ((col % 2 == 1) && (row % 2 == 1))) {
            color = lightGray;
        }
        return color;
    }
    public ReversiServer(Socket socket) {
        super(new BorderLayout());
        setPreferredSize(new Dimension(1000, 700));
        setLocation(0, 0);
```

```
                board = new ReversiLogic();
                start = board;
                arrReversi.add(new ReversiLogic(board));
                newGame = new JButton("Новая игра");
                newGame.setFont(font15);
                newGame.setPreferredSize(new Dimension(350, 40));
                newGame.addActionListener(new Action());
                rules = new JButton("Правила");
                rules.setFont(font15);
                rules.setPreferredSize(new Dimension(100, 40));
                rules.addActionListener(new Action());
                info = new JButton("Информация о разработчиках");
                info.setFont(font15);
                info.setPreferredSize(new Dimension(250, 40));
                info.addActionListener(new Action());
                boardPanel = new JPanel(new GridLayout(8, 8));
                boardPanel.setBounds(0, 300, 750, 750);
                cell = new JButton[64];
                int k = 0;
                for (int row = 0; row < rows; row++) {
                   for (int col = 0; col < cols; col++) {
                      cell[k] = new JButton();
                      cell[k].setSize(50, 50);
                      cell[k].setBackground(ColorCheck(col, row));
                      if (board.gameCells[row][col].getCh() == 'X') {
                         cell[k].setIcon(imgDark);
                      } else if (board.gameCells[row][col].getCh() == 'O') {
                         cell[k].setIcon(imgLight);
                      } else if (row == 2 && col == 4 || row == 3 && col == 5 || row == 4 && col == 2 || row == 5 &&
col == 3) {
                         cell[k].setBackground(green);
                      }
                      cell[k].addActionListener(new Action());
                      boardPanel.add(cell[k]);
                      k++;
                   }
                }
                add(boardPanel);
                JLabel dark = new JLabel();
                dark.setIcon(imgDark);
                JLabel light = new JLabel();
                light.setIcon(imgLight);
```

```
    JPanel gamePanel = new JPanel(new GridBagLayout());
    GridBagConstraints c = new GridBagConstraints();
    JPanel infoPanel = new JPanel(new GridBagLayout());
    GridBagConstraints c1 = new GridBagConstraints();
    JPanel scorePanel = new JPanel(new GridBagLayout());
    GridBagConstraints c2 = new GridBagConstraints();
    c.weightx = 300;
    c.weighty = 800;
    c1.gridx = 0;
    c1.gridy = 0;
    infoPanel.add(rules, c1);
    c1.gridx = 1;
    c1.gridy = 0;
    infoPanel.add(info, c1);
    c.gridx = 0;
    c.gridy = 1;
    gamePanel.add(infoPanel, c);
    score = new JLabel();
    score.setText(userScore + " : " + pcScore);
    score.setFont(font55);
    JLabel players = new JLabel();
    players.setText("   Вы                Компьютер");
    players.setFont(font15);
    c2.gridx = 0;
    c2.gridy = 0;
    scorePanel.add(players, c2);
    c2.gridx = 0;
    c2.gridy = 1;
    scorePanel.add(score, c2);
    c.gridx = 0;
    c.gridy = 2;
    gamePanel.add(scorePanel, c);
    c.gridx = 0;
    c.gridy = 3;
    gamePanel.add(newGame, c);
    add(gamePanel, BorderLayout.EAST);
}
public static void main(String[] args) throws IOException {
    ServerSocket serverSocket = new ServerSocket(1025);
    System.out.println("Игра \"Реверси\" запущена...");
    PrintWriter out = null;
    BufferedReader in = null;
```

```
        Socket socket = null;
        while (true) {
            try {
                socket = serverSocket.accept();
                out = new PrintWriter(new OutputStreamWriter(socket.getOutputStream()), true);
                in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            } catch (Exception e) {
                System.err.println(e.getMessage());
            }
            new ReversiServer(socket);
            try {
                out.println();
                System.out.println("Клиент успешно подключён...\n");
            } catch (Exception e) {
                System.err.println("Ошибка");
            }
            out.close();
            in.close();
            socket.close();
        }
    }
    class Action implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            if (e.getSource() == newGame) {
                arrReversi.clear();
                board = new ReversiLogic();
                arrReversi.add(new ReversiLogic(board));
                int k = 0;
                for (int row = 0; row < rows; row++) {
                    for (int colum = 0; colum < cols; colum++) {
                        cell[k].setIcon(null);
                        cell[k].setBackground(ColorCheck(colum, row));
                        if (board.gameCells[row][colum].getCh() == 'X') {
                            cell[k].setIcon(imgDark);
                        } else if (board.gameCells[row][colum].getCh() == 'O') {
                            cell[k].setIcon(imgLight);
                        }
                        if (row == 2 && colum == 4 || row == 3 && colum == 5 || row == 4 && colum == 2 || row
== 5 && colum == 3) {
                            cell[k].setBackground(green);
                        }
```

```
            ++k;
        }
    }
    userScore = 2;
    pcScore = 2;
    score.setText(userScore + " : " + pcScore);
} else if (e.getSource() == rules) {
    new RulesFrame();
} else if (e.getSource() == info) {
    new InfoFrame();
} else {
    for (int i = 0; i < 64; i++) {
        if (e.getSource() == cell[i]) {
            int xCor, yCor, ct, point;
            int arr[] = new int[3];
            if (i == 0) {
                xCor = 0;
                yCor = 0;
            } else {
                yCor = i % 8;
                xCor = i / 8;
            }
            ct = board.gamePlay(xCor, yCor);
            System.out.printf("Ход игрока: (%d;%d)\n", xCor, yCor);
            if (ct == 0) {
                arrReversi.add(new ReversiLogic(board));
                int k = 0;
                for (int row = 0; row < rows; row++) {
                    for (int colum = 0; colum < cols; colum++) {
                        if (board.gameCells[row][colum].getCh() == 'X') {
                            cell[k].setIcon(imgDark);
                            cell[k].setBackground(ColorCheck(colum, row));
                        } else if (board.gameCells[row][colum].getCh() == 'O') {
                            cell[k].setIcon(imgLight);
                            cell[k].setBackground(ColorCheck(colum, row));
                        }
                        k++;
                    }
                }
                board.fillingFieldWithControls(arr);
                userScore = arr[0];
                pcScore = arr[1];
```

```
                        point = arr[2];
                        score.setText(userScore + " : " + pcScore);
                    }
                    if (ct == 0 || ct == -1) {
                        board.gamePlay();
                        arrReversi.add(new ReversiLogic(board));
                        ArrayList<Integer> arrList = new ArrayList<Integer>();
                        int k = 0;
                        for (int row = 0; row < rows; row++) {
                            for (int colum = 0; colum < cols; colum++) {
                                if (board.gameCells[row][colum].getCh() == 'X') {
                                    cell[k].setIcon(imgDark);
                                } else if (board.gameCells[row][colum].getCh() == 'O') {
                                    cell[k].setIcon(imgLight);
                                } else if (board.gameCells[row][colum].getCh() == '.') {
                                    cell[k].setIcon(null);
                                }
                                cell[k].setBackground(ColorCheck(colum, row));
                                k++;
                            }
                        }
                        board.findLegalMove(arrList);
                        for (int j = 0; j < arrList.size(); j += 2) {
                            cell[arrList.get(j) * rows + arrList.get(j + 1)].setBackground(green);
                        }
                        board.fillingFieldWithControls(arr);
                        userScore = arr[0];
                        pcScore = arr[1];
                        point = arr[2];
                        score.setText(userScore + " : " + pcScore);
                    }
                }
            }
            int st = board.endOfGame();
            if (st == 0) {
                if (userScore > pcScore)
                    JOptionPane.showMessageDialog(null, "Вы выиграли!\nВозможных ходов нет.", "Конец
игры", JOptionPane.PLAIN_MESSAGE);
                else
                    JOptionPane.showMessageDialog(null, "Вы проиграли!\nВозможных ходов нет.",
"Конец игры", JOptionPane.PLAIN_MESSAGE);
            } else if (st == 1 || st == 3) {
```

```
                    JOptionPane.showMessageDialog(null, "Вы проиграли!", "Конец игры",
JOptionPane.PLAIN_MESSAGE);
                } else if (st == 2 || st == 4) {
                    JOptionPane.showMessageDialog(null, "Вы выиграли!", "Конец игры",
JOptionPane.PLAIN_MESSAGE);
                } else if (st == 5) {
                    JOptionPane.showMessageDialog(null, "Ничья!", "Конец игры",
JOptionPane.PLAIN_MESSAGE);
                }
            }
        }
    }
}
```

ReversiLogic

```
import java.util.ArrayList;
public class ReversiLogic {
    private int rows = 8;
    private int cols = 8;
    private int computerCount = 0;
    private int playerCount = 0;
    public Cell gameCells[][];
    public ReversiLogic() {
        int mid;
        mid = rows / 2;
        gameCells = new Cell[8][8];
        for (int i = 0; i < rows; ++i)
            gameCells[i] = new Cell[8];
        for (int i = 0; i < rows; ++i) {
            for (int j = 0; j < cols; ++j) {
                gameCells[i][j] = new Cell();
                char c = (char) (97 + j);
                if ((i == mid - 1) && (j == mid - 1)) {
                    gameCells[i][j].setPosition(c, 'X', i + 1);
                } else if ((i == mid - 1) && (j == mid)) {
                    gameCells[i][j].setPosition(c, 'O', i + 1);
                } else if ((i == mid) && (j == mid - 1)) {
                    gameCells[i][j].setPosition(c, 'O', i + 1);
                } else if ((i == mid) && (j == mid)) {
                    gameCells[i][j].setPosition(c, 'X', i + 1);
                } else {
                    gameCells[i][j].setPosition(c, '.', i + 1);
```

```
            }
        }
    }
}
public ReversiLogic(ReversiLogic reversi) {
    int y;
    char c, x;
    gameCells = new Cell[8][8];
    for (int i = 0; i < rows; ++i)
        gameCells[i] = new Cell[8];
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
            gameCells[i][j] = new Cell();
            c = reversi.gameCells[i][j].getCh();
            y = reversi.gameCells[i][j].getCorY();
            x = reversi.gameCells[i][j].getCorX();
            gameCells[i][j].setPosition(x, c, y);
        }
    }
}
public void findLegalMove(ArrayList<Integer> legalMoves) {
    int change;
    change = 0;
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            if (gameCells[i][j].getCh() == '.') {
                int numberOfMoves[] = new int[1];
                move(i, j, change, 'X', 'O', numberOfMoves);
                if (numberOfMoves[0] != 0) {
                    legalMoves.add(i);
                    legalMoves.add(j);
                }
            }
        }
    }
}
public int gamePlay() {
    int change, max = 0, mX = 0, mY = 0;
    change = 0;
    int numberOfMoves[] = new int[1];
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
```

```
            if (gameCells[i][j].getCh() == '.') {
                move(i, j, change, 'O', 'X', numberOfMoves);
                if (max < numberOfMoves[0]) {
                    max = numberOfMoves[0];
                    mX = i;
                    mY = j;
                }
            }
        }
    }
    computerCount = max;
    if (computerCount == 0) {
        computerCount = -1;
        return -1;
    }
    if (computerCount != 0) {
        change = 1;
        move(mX, mY, change, 'O', 'X', numberOfMoves);
    }
    return 0;
}
public int gamePlay(int xCor, int yCor) {
    int status;
    int change = 0, max = 0;
    int numberOfMoves[] = new int[1];
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
            if (gameCells[i][j].getCh() == '.') {
                move(i, j, change, 'X', 'O', numberOfMoves);
                if (max < numberOfMoves[0])
                    max = numberOfMoves[0];
            }
        }
    }
    playerCount = max;
    if (playerCount == 0) {
        playerCount = -1;
        return -1;
    }
    if (playerCount != 0) {
        change = 1;
        if (!(gameCells[xCor][yCor].getCh() == '.'))
```

```
            return 1;
        status = move(xCor, yCor, change, 'X', 'O', numberOfMoves);
        if (status == -1)
            return 1;
    }
    return 0;
}
public int endOfGame() {
    int[] arr = new int[3];
    int cross, circular, point;
    fillingFieldWithControls(arr);
    cross = arr[0];
    circular = arr[1];
    point = arr[2];
    if ((playerCount == -1 && computerCount == -1) || point == 0) {
        if (playerCount == -1 && computerCount == -1)
            return 0;
        if (circular > cross)
            return 1;
        else if (cross > circular)
            return 2;
        else if (cross == 0)
            return 3;
        else if (circular == 0)
            return 4;
        else
            return 5;
    }
    return -1;
}
public void fillingFieldWithControls(int[] controls) {
    int cross = 0, point = 0, circular = 0;
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
            if (gameCells[i][j].getCh() == 'X')
                cross++;
            else if (gameCells[i][j].getCh() == 'O')
                circular++;
            else if (gameCells[i][j].getCh() == '.')
                point++;
        }
    }
```

```
    controls[0] = cross;
    controls[1] = circular;
    controls[2] = point;
}
int move(int xCor, int yCor, int change, char char1, char char2, int[] numberOfMoves) {
    int cont, st2 = 0, st = 0;
    int status = -1, corX, corY, temp;
    char str;
    int ix, y, x;
    x = xCor;
    y = yCor;
    numberOfMoves[0] = 0;
    if ((x + 1 < rows) && (gameCells[x + 1][y].getCh() == char2)) {
        cont = x;
        while ((cont < rows) && (st2 != -1) && (st != 2)) {
            cont++;
            if (cont < rows) {
                if (gameCells[cont][y].getCh() == char2)
                    st = 1;
                else if (gameCells[cont][y].getCh() == char1)
                    st = 2;
                else
                    st2 = -1;
            }
        }
        if (st == 2) {
            temp = cont - x - 1;
            numberOfMoves[0] += temp;
        }
        if (st == 2 && change == 1) {
            for (int i = x; i < cont; ++i) {
                str = gameCells[i][y].getCorX();
                ix = gameCells[i][y].getCorY();
                gameCells[i][y].setPosition(str, char1, ix);
            }
            status = 0;
        }
        st = 0;
        st2 = 0;
    }
    if ((x - 1 >= 0) && (gameCells[x - 1][y].getCh() == char2)) {
        cont = x;
```

```
    while ((cont >= 0) && (st2 != -1) && (st != 2)) {
        cont--;
        if (cont >= 0) {
            if (gameCells[cont][y].getCh() == char2)
                st = 1;
            else if (gameCells[cont][y].getCh() == char1)
                st = 2;
            else
                st2 = -1;
        }
    }
    if (st == 2) {
        temp = x - cont - 1;
        numberOfMoves[0] += temp;
    }
    if (st == 2 && change == 1) {
        for (int i = cont; i <= x; ++i) {
            str = gameCells[i][y].getCorX();
            ix = gameCells[i][y].getCorY();
            gameCells[i][y].setPosition(str, char1, ix);
        }
        status = 0;
    }
    st = 0;
    st2 = 0;
}
if ((y + 1 < cols) && (gameCells[x][y + 1].getCh() == char2)) {
    cont = y;
    while ((cont < cols) && (st2 != -1) && (st != 2)) {
        cont++;
        if (cont < cols) {
            if (gameCells[x][cont].getCh() == char2)
                st = 1;
            else if (gameCells[x][cont].getCh() == char1)
                st = 2;
            else
                st2 = -1;
        }
    }
    if (st == 2) {
        temp = cont - y - 1;
        numberOfMoves[0] += temp;
```

```
    }
    if (st == 2 && change == 1) {
        for (int i = y; i < cont; ++i) {
            str = gameCells[x][i].getCorX();
            ix = gameCells[x][i].getCorY();
            gameCells[x][i].setPosition(str, char1, ix);
        }
        status = 0;
    }
    st = 0;
    st2 = 0;
}
if ((y - 1 >= 0) && (gameCells[x][y - 1].getCh() == char2)) {
    cont = y;
    while ((cont >= 0) && (st2 != -1) && (st != 2)) {
        cont--;
        if (cont >= 0) {
            if (gameCells[x][cont].getCh() == char2)
                st = 1;
            else if (gameCells[x][cont].getCh() == char1)
                st = 2;
            else
                st2 = -1;
        }
    }
    if (st == 2) {
        temp = y - cont - 1;
        numberOfMoves[0] += temp;
    }
    if (st == 2 && change == 1) {
        for (int i = cont; i <= y; ++i) {
            str = gameCells[x][i].getCorX();
            ix = gameCells[x][i].getCorY();
            gameCells[x][i].setPosition(str, char1, ix);
        }
        status = 0;
    }
    st = 0;
    st2 = 0;
}
if ((x - 1 >= 0) && (y + 1 < cols) && (gameCells[x - 1][y + 1].getCh() == char2)) {
    corY = y;
```

```
    corX = x;
    while ((corX >= 0) && (corY < cols) && (st2 != -1) && (st != 2)) {
        corX--;
        corY++;
        if ((corX >= 0) && (corY < cols)) {
            if (gameCells[corX][corY].getCh() == char2)
                st = 1;
            else if (gameCells[corX][corY].getCh() == char1)
                st = 2;
            else
                st2 = -1;
        }
    }
    if (st == 2) {
        temp = x - corX - 1;
        numberOfMoves[0] += temp;
    }
    if (st == 2 && change == 1) {
        while ((corX <= x) && (y < corY)) {
            corX++;
            corY--;
            if ((corX <= x) && (y <= corY)) {
                str = gameCells[corX][corY].getCorX();
                ix = gameCells[corX][corY].getCorY();
                gameCells[corX][corY].setPosition(str, char1, ix);
            }
        }
        status = 0;
    }
    st = 0;
    st2 = 0;
}
if ((x - 1 >= 0) && (y - 1 >= 0) && (gameCells[x - 1][y - 1].getCh() == char2)) {
    corY = y;
    corX = x;
    while ((corX >= 0) && (corY >= 0) && (st2 != -1) && (st != 2)) {
        corX--;
        corY--;
        if ((corX >= 0) && (corY >= 0)) {
            if (gameCells[corX][corY].getCh() == char2)
                st = 1;
            else if (gameCells[corX][corY].getCh() == char1)
```

```
                st = 2;
            else
                st2 = -1;
        }
    }
    if (st == 2) {
        temp = x - corX - 1;
        numberOfMoves[0] += temp;
    }
    if (st == 2 && change == 1) {
        while ((corX <= x) && (corY <= y)) {
            corX++;
            corY++;
            if ((corX <= x) && (corY <= y)) {
                str = gameCells[corX][corY].getCorX();
                ix = gameCells[corX][corY].getCorY();
                gameCells[corX][corY].setPosition(str, char1, ix);
            }
        }
        status = 0;
    }
    st = 0;
    st2 = 0;
}
if ((x + 1 < rows) && (y + 1 < cols) && (gameCells[x + 1][y + 1].getCh() == char2)) {
    corY = y;
    corX = x;
    while ((corX < rows) && (corY < cols) && (st2 != -1) && (st != 2)) {
        corX++;
        corY++;
        if ((corX < rows) && (corY < cols)) {
            if (gameCells[corX][corY].getCh() == char2)
                st = 1;
            else if (gameCells[corX][corY].getCh() == char1)
                st = 2;
            else
                st2 = -1;
        }
    }
    if (st == 2) {
        temp = corX - x - 1;
        numberOfMoves[0] += temp;
```

```
    }
    if (st == 2 && change == 1) {
        while ((corX >= x) && (corY >= y)) {
            corX--;
            corY--;
            if ((corX >= x) && (corY >= y)) {
                str = gameCells[corX][corY].getCorX();
                ix = gameCells[corX][corY].getCorY();
                gameCells[corX][corY].setPosition(str, char1, ix);
            }
        }
        status = 0;
    }
    st = 0;
    st2 = 0;
}
if ((x + 1 < rows) && (y - 1 >= 0) && (gameCells[x + 1][y - 1].getCh() == char2)) {
    corY = y;
    corX = x;
    while ((corX < rows) && (corY >= 0) && (st2 != -1) && (st != 2)) {
        corX++;
        corY--;
        if ((corX < rows) && (corY >= 0)) {
            if (gameCells[corX][corY].getCh() == char2)
                st = 1;
            else if (gameCells[corX][corY].getCh() == char1)
                st = 2;
            else
                st2 = -1;
        }
    }
    if (st == 2) {
        temp = corX - x - 1;
        numberOfMoves[0] += temp;
    }
    if (st == 2 && change == 1) {
        while ((corX >= x) && (corY <= y)) {
            corX--;
            corY++;
            if ((corX >= x) && (corY <= y)) {
                str = gameCells[corX][corY].getCorX();
                ix = gameCells[corX][corY].getCorY();
```

```
                    gameCells[corX][corY].setPosition(str, char1, ix);
                }
            }
            status = 0;
        }
        st = 0;
        st2 = 0;
    }
    if (status == 0)
        return 0;
    else
        return -1;
  }
}
```

Cell

```
public class Cell {
  private int corY;
  private char corX;
  private char ch;
  public Cell(char x, int y, char c) {
    corY = y;
    corX = x;
    ch = c;
  }
  Cell() {
  }
  char getCorX() {
    return corX;
  }
  int getCorY() {
    return corY;
  }
  char getCh() {
    return ch;
  }
  void setPosition(char x, char c, int y) {
    corX = x;
    corY = y;
    ch = c;
  }
}
```

RulesFrame

```
import javax.swing.*;
public class RulesFrame extends JFrame {
    private JLabel label;
    public RulesFrame() {
        super("Правила игры");
        setBounds(165, 140, 670, 420);
        setResizable(false);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setLayout(null);
        label = new JLabel("<html>Правила игры:" +
                "<ul>" +
                "<li>Перед игрой в центр игрового поля ставятся четыре фишки (2 белые и 2 черные).</li>" +
                "<li>Первый ход делает игрок черных фишек.</li>" +
                "<li>Игрок размещает фишки с верхом своего цвета.</li>" +
                "<li>Фишку необходимо положить на одну из полей своего цвета таким образом, чтобы между фишкой игрока и уже стоящей фишкой этого же цвета, находился непрерывный ряд фишек оппонента (вертикально, диагонально или горизонтально). То есть ряд фишек оппонента должен быть закрыт фишками игрока противоположного цвета. При этом все фишки оппонента, которые игрок закрыл, переворачиваются на другую сторону. То есть переходят к ходившему игроку.</li>" +
                "<li>Если при ходе игрока закрываются больше одного ряда фишек оппонента, то все «закрытые» фишки в рядах переворачиваются и переходят к ходившему игроку.</li>" +
                "<li>Любой из игроков может ходить так, как хочет.</li>" +
                "<li>Отказываться от хода запрещается.</li>" +
                "<li>Если нет возможности осуществить ход, то ход переходит оппоненту.</li>" +
                "<li>Игра завершается, когда на игровом поле будут выставлены все фишки.</li>" +
                "<li>Игра завершается, когда ни один игрок не может сделать ход.</li>" +
                "<li>По завершению игры осуществляется подсчёт белых и черных фишек. У кого фишек на игровом поел больше, тот и победитель.</li>" +
                "<li>Если на игровом поле черных и белых фишек одинаковое количество, то объявляется ничья.</li>" +
                "</ul></html>");
        label.setBounds(10, 0, 630, 380);
        add(label);
        setVisible(true);
    }
}
```

InfoFrame

```
import javax.swing.*;
```

```
public class InfoFrame extends JFrame {
    private JLabel label;
    public InfoFrame() {
        super("Информация о разработчиках");
        setBounds(275, 210, 450, 250);
        setResizable(false);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setLayout(null);
        label = new JLabel("<html>Самарский университет" +
                "<br>Кафедра программных систем" +
                "<br><br>Курсовой проект по дисциплине" +
                "<br>«Объектная распределенная обработка»" +
                "<br><br>Тема проекта" +
                "<br>«Распределённое клиент-серверное приложение «Игра «Реверси»" +
                "<br><br>Разработчик группы 6413-020302D" +
                "<br>Гижевская Валерия" +
                "</html>");
        label.setBounds(10, 0, 400, 220);
        add(label);
        setVisible(true);
    }
}
```