Министерство образования Республики Беларусь

Учреждение образования

БелорусскиЙ государственный университет

информатики и радиоэлектроники

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

**Игровое Программное средство «MEOW-MEOW»**

|  |  |  |
| --- | --- | --- |
| Студент |  | М.Г. Гладкий |
| Руководитель |  | А.В. Варфоломеев |

Минск 2020

Содержание

[Введение 5](#_Toc42242688)

[1. Анализ предметной области 7](#_Toc42242689)

[1.1. Обзор аналогов 7](#_Toc42242690)

[1.2. Формирование требований 9](#_Toc42242691)

[2. Разработка программного средства 10](#_Toc42242692)

[2.1. Структура программы 10](#_Toc42242693)

[2.2. Серверная часть 10](#_Toc42242694)

[2.3. Клиентская часть 11](#_Toc42242695)

[2.4. Удалённый доступ 12](#_Toc42242696)

[2.5. Передаваемые сообщения 14](#_Toc42242697)

[2.6. Графическое взаимодействие 16](#_Toc42242698)

[2.7. Системные сообщения 18](#_Toc42242699)

[3. Руководство пользователя 20](#_Toc42242700)

[3.1. Правила игры 20](#_Toc42242701)

[3.2. Работа с программой 20](#_Toc42242702)

[Заключение 23](#_Toc42242703)

[Список использованных источников 24](#_Toc42242704)

[Приложение А 25](#_Toc42242705)

Введение

На сегодняшний день можно с уверенностью сказать, что компьютерные сети стали неотъемлемой частью нашей жизни, а область их применения охватывает буквально все сферы человеческой деятельности. Это вызвано несколькими причинами:

- объединение компьютеров в сеть позволяет значительно экономить денежные средства за счет уменьшения затрат на содержание компьютеров (достаточно иметь определенное дисковое пространство на файл-сервере (главном компьютере сети) с установленными на нем программными продуктами, используемыми несколькими рабочими станциями);

- локальные сети позволяют использовать почтовый ящик для передачи сообщений на другие компьютеры, что позволяет в наиболее короткий срок передавать документы с одного компьютера на другой;

- локальные сети, при наличии специального программного обеспечения (ПО), служат для организации совместного использования файлов.

Локальная сеть – это группа компьютеров, которые могут связываться друг с другом, совместно использовать периферийное оборудование (например, жесткие диски, принтеры и т.д.) и обращаться к удаленным центральным ЭВМ или другим локальным сетям. Локальная сеть может состоять из одного или более файл-серверов, рабочих станций и периферийных устройств. Пользователи сети могут совместно использовать одни и те же файлы (как файлы данных, так и файлы программ), посылать сообщения непосредственно между рабочими станциями и защищать файлы с помощью мощной системы защиты. Основными видами локальных вычислительных сетей являются Ethernet и ARCNET.

Одним из итогов развития компьютерных сетей являются сетевые игры. На сегодняшний день огромное количество разных по интересам людей проводят время в компьютерных играх. Всех объединяет одно – желание испытать в виртуальных мирах что-то новое, неизвестное и получить наслаждение как от игрового процесса, так и от достигнутых в игре результатов. Компьютерные игры стали настоящим культурным феноменом – возникнув как незамысловатый плод творческой мысли программистов, они с каждым годом приобретали все большую и большую популярность – и развились до того, что стали отдельной специфической спортивной дисциплиной – киберспортом.

Автором идеи видеоигр считают американского изобретателя Ральфа Бэра, родившегося в 1922-м году в Германии. Седьмого мая 1967-го года Бэр со своим подчиненным Харрисоном продемонстрировал совету директоров Sanders первую видеоигру, однако считается что первым игровым продуктом стала «ОХО» или, как ее по другому называют, крестики-нолики, в одиночку сделанная А.С. Дугласом в далёком 1952-м году. После свет увидела «Tennis for two», придуманная и реализованная Уильямом Хигинботэмом в 1958-м году в Нью-Йорке. Посетители его лаборатории могли поиграть в теннис на цифровом корте, управляя своими "ракетками" с помощью джойстиков Первой многопользовательской игрой считается Tennis for Two (1958 год). Игра была сделана для аналоговой ЭВМ и выводила игровое поле на осциллографе. Игра Maze War стала и первой реализацией deathmatch'а, и первой сетевой игрой. В ней игроки перемещаются по лабиринту; доступна возможность перемещения вперёд, назад, поворачиваться направо и налево (каждый раз на 90°), а также заглядывать в дверные проёмы. В игре используется простая тайловая графика – таким образом, игрок перемещается по невидимым квадратам. Другие участники игры представлены на экране в виде глазных яблок. При появлении соперника на экране игрок может стрелять в него. За каждое убийство начисляются очки, а за каждую смерть — снимаются.

Карточные игры появились гораздо раньше, чем их аналоги на компьютеры и мобильные устройства. Они относятся к жанру аркад и очень популярны.

Мау-Мау – это карточная игра до шести игроков, появившаяся впервые в Германии. Она использует карты как для игры в «Дурака», но правила напоминают «UNO». Цель состоит в том, чтобы первым избавиться от всех карт на руке. Когда у игрока закончились карты, он должен сказать: «Мау-Мау», иначе возьмёт несколько карт и продолжит игру.

1. Анализ предметной области

1.1. Обзор аналогов

На сегодняшний день существует множество аналогов игры «May-May» и большинство из них реализовано для мобильных устройств на платформах Android и iOS, однако можно встретить и браузерные версии. Одной из наиболее популярных является «Дурак онлайн» в App Store от компании R-Soft. В данном приложении реализован выбор нескольких режимов игры, размера колоды карт. Пользователи начинают игру после подключения к комнате, созданной кем-либо. Пример комнаты на рисунке 1.1. Цель игры состоит в том, чтобы обыграть своих соперников и получить виртуальную валюту, за которую можно купить различного типа бонусы.

Достоинства:

- минималистичный и понятный интерфейс;

- большое количество игроков и комнат;

- малый вес приложения.

Недостатки:

- нестабильность соединения во время игры;

- нет разделения между новичками и умелыми игроками.



Рисунок 1.1 – Игровая комната в «Дурак онлайн».

На просторах интернета можно также сыграть в игру под названием «Heartstone» от компании «Blizzard Entertainment». Опробовать данное приложение можно на платформах PC, iOS, Android. Игра является довольно популярной, по ней проводятся киберспортивные мероприятия, однако новичку освоиться в ней сложно, ввиду множества различных механик и способностей персонажей и карт. Разработчики постоянно привносят в игру что-то новое, давая пользователям возможность пробовать нечто новое. Внешний вид игрового процесса представлен на рисунке 1.2.

Достоинства:

- большое разнообразие карт и способностей;

- стабильное соединение при игре;

- современный и продуманный интерфейс.

Недостатки:

- большой вес приложения;

- новичку сложно освоиться.



Рисунок 1.2 – Игровой процесс «Heartstone».

1.2. Формирование требований

Используя выявленные достоинства и недостатки аналогов программы, можно определить основные требования к данному курсовому проекту, которые покрывали бы недостатки и имели достоинства рассмотренных приложений.

Из достоинств «Дурак онлайн» будет использован малый вес приложения, а также понятный интерфейс. Исходя из игры «Heartstone» будет реализована возможность установления стабильного соединения во время игрового процесса.

Таким образом, в ходе разработки данного программного средства планируется реализовать следующие функции:

- создание сервера и клиента;

- подключение клиента к удалённому серверу в локальной сети;

- передача объектов между сервером и клиентами;

- реализация взаимодействия интерфейса между игроками;

- авторизация клиентов через базу данных.

Для разработки программного средства используется платформа Java. Она предоставляет программный интерфейс к ресурсам операционной системы, таким как файловая система, сетевые сокеты и тому подобное, для языка Java.

В качестве языка разработки будет использован Java. В нем достаточно просто создать многопоточное приложение, имеются классы для создания сокетов, а также удобная работа с графикой.

В качестве среды разработки использовалась IntelliJ IDEA. Она анализирует код в поисках связей между символами во всех файлах и на всех языках, используемых в проекте. На основе этого анализа IntelliJ IDEA предоставляет помощь при написании кода, удобную навигацию, проверку ошибок в коде и, конечно, рефакторинги [1].

Для управления версиями используется система контроля версий Git, которая на данный момент является самой популярной реализацией в этой области. Система контроля версий позволяет разделять разработку на ветви и вести ее продуктивнее. Также есть возможность откатывать исходный код к рабочему состоянию в случае, когда вносились изменения, нарушающие нормальную работу проекта.

2. Разработка программного средства

2.1. Структура программы

Проект данного программного средства будет иметь структуру, описанную ниже:

Сборка Server:

- Card – класс, представляющий информацию об конкретной карте, включая её стоимость и масть;

- ClientHandler – класс, реализующий отправку и обработку сообщений между клиентом и сервером;

- CourseS – класс, выполняющий запуск сервера;

- GField – класс, описывающий информацию об игровом поле, такую как: игроки, карты на руке и в отбое;

- Message – класс, определяющий сообщение, отправляемое между сервером и пользователем;

- Player – класс, включающий в себя информацию об игроке;

- Server – класс, отвечающий за подключение клиентов и отправку сообщений всем пользователям;

- UDPMessage – класс, хранящий логику работы UDP сервера.

Сборка Client:

- Client – класс, реализующий подключение клиента к серверу, вывод игрового меню и запуск игрового процесса;

- CourseCl – класс, осуществляющий запуск и инициализацию приложения на стороне клиента;

- GController – класс, контролирующий процесс игры, обновляющий информацию на столе, а также включающий в себя игровую логику.

2.2. Серверная часть

Для установления соединения между сервером и клиентом использовались протоколы TCP/IP и UDP. Протокол TCP/IP используется для обмена данными между сервером и клиентами, UDP, в свою очередь, необходим для получения ip-адресов сервера со стороны клиента и клиентов, со стороны сервера.

TCP – один из основных протоколов передачи данных интернета, предназначенный для управления передачей данных. В стеке протоколов TCP/IP выполняет функции транспортного уровня модели OSI. Механизм TCP предоставляет поток данных с предварительной установкой соединения, осуществляет повторный запрос данных в случае потери данных и устраняет дублирование при получении двух копий одного пакета, гарантируя тем самым, в отличие от UDP, целостность передаваемых данных и уведомление отправителя о результатах передачи.

Когда осуществляется передача от компьютера к компьютеру через Интернет, TCP работает на верхнем уровне между двумя конечными системами, например, браузером и веб-сервером. TCP осуществляет надёжную передачу потока байтов от одного процесса к другому. TCP реализует управление потоком, управление перегрузкой, рукопожатие, надёжную передачу.

В языке Java существуют классы, такие как java.net.Socket и java.net.ServerSocket, позволяющие создать сокеты для сервера и клиента, таким образом, что образуется соединение по TCP протоколу [2]. Это позволяет быстро и просто подключиться к серверу со стороны клиента.

После установки соединения с клиентом, создается новый поток, в котором обрабатываются сообщения, полученные от клиента. В языке Java за создание отдельных потоков отвечает класс Thread. При этом, если в потоке необходимо создать объект какого-либо класса, этот класс должен реализовывать интерфейс Runnable. Также при подключении клиента увеличивается счётчик общего количества пользователей, который используется для отображения информации о текущем количестве игроков, подключенных к серверу.

Для отправки и получения сообщений используются классы ObjectInputStream и ObjectOutputStream. Это необходимо для отправки не простых сток, а объектов, которые должны реализовывать интерфейс Serializable. Так же на приём сообщений устанавливается специальный буфер, который проверяет, пришло сообщение или нет.

После получения сообщения, сервер проверяет поле типа String в имеющемся объекте и, в зависимости от него, формирует ответ и отправляет его клиенту. Среди возможных сообщений – «close» (отключение клиента от сервера), «login» (введённое имя пользователя), «game\_upd» (присылается клиентом, во время нахождения в игре).

2.3. Клиентская часть

Также как и сервер, клиент использует сокеты, для установки соединения с сервером. Аналогично использует объекты классов ObjectInputStream и ObjectOutputStream для отправки сообщений, и специальный буфер для проверки существования нового сообщения. Для того, чтобы осуществить работу с графическим интерфейсом, необходимо в отдельном потоке обрабатывать полученные от сервера сообщения и отправлять новые обратно. Это позволяет создать интерфейс, который динамично изменяется в процессе игры. Инициализация клиента представлена на рисунке 2.1.

В том случае, если клиент решил покинуть приложение, необходимо отправить серверу сообщение о выходе, а после чего оборвать соединение. Это позволит избежать непредвиденных ошибок при выходе пользователя из приложения. Если клиент успешно оборвал соединение, остальные клиенты увидят сообщение об этом в консоли.

2.4. Удалённый доступ

Для предоставления возможности удалённого доступа к серверу, был использован UDP протокол. UDP использует простую модель передачи, без неявных «рукопожатий» для обеспечения надёжности, упорядочивания или целостности данных. Таким образом, UDP предоставляет ненадёжный сервис, и дейтаграммы могут прийти не по порядку, дублироваться или вовсе исчезнуть без следа. UDP подразумевает, что проверка ошибок и исправление либо не нужны, либо должны исполняться в приложении. Чувствительные ко времени приложения часто используют UDP, так как предпочтительнее сбросить пакеты, чем ждать задержавшиеся пакеты, что может оказаться невозможным в системах реального времени. При необходимости исправления ошибок на сетевом уровне интерфейса приложение может задействовать TCP или SCTP, разработанные для этой цели.

Природа UDP как протокола без сохранения состояния также полезна для серверов, отвечающих на небольшие запросы от огромного числа клиентов, например DNS и потоковые мультимедийные приложения вроде IPTV, Voice over IP, протоколы туннелирования IP и многие онлайн-игры.

UDP-приложения используют дейтаграммные сокеты для установки соединения между хостами. Приложение связывает сокет с его конечной точкой передачи данных, которая является комбинацией IP-адреса и порта службы. Порт – это программная структура, определяемая номером порта — 16-битным целочисленным значением (то есть от 0 до 65535). Порт 0 зарезервирован, хотя и является допустимым значением порта источника в случае, если процесс-отправитель не ожидает ответных сообщений.

Со стороны сервера происходит создание объекта класса DatagramSocket, который ожидает получения дейтаграммы от клиента. После этого, из полученного сообщения он узнает ip-адрес и порт для обратной отправки дейтаграммы, при этом также в неё заносится порт, для установки соединения с TCP сервером. Для того, чтобы отправить порт сервера через дейтаграмму, его необходимо разбить на байты и записать в соответствующие байты дейтаграммы. Код этого процесса представлен ниже:

//Занесение номера порта TCP сервера   
buf[0] = (byte)((serverPort >> 24) & 0xff);  
buf[1] = (byte)((serverPort >> 16) & 0xff);  
buf[2] = (byte)((serverPort >> 8) & 0xff);  
buf[3] = (byte)(serverPort & 0xff);

Со стороны клиента создаётся дейтаграмма, в которой указаны номер порта UDP сервера, а также ip-адрес для широковещательного сигнала. В данном случае используется Directed Broadcast(192.168.100.255). Пакет, отправленный на адрес 255.255.255.255 (Limited Broadcast) ограничен лишь той сетью, где он зародился – МАС-адрес выставляется в FFFF-FFFF-FFFF. Если пакет отправляется на 192.168.100.255, то сначала согласно всем правилам маршрутизации пакет достигает сети назначения 192.168.100.0, а уже потом рассылается всем хостам в этой сети [3]. Это необходимо для того, чтобы найти работающий сервер в локальной сети. Также создаётся объект класса DatagramSocket, и после чего отправляется широковещательный сигнал. Как уже было сказано, сервер, получив его, отправит клиенту дейтаграмму обратно. Как только клиент её получит, ему станет известен ip-адрес TCP сервера и его порт, что позволит успешно установить соединение с сервером.



Рисунок 2.1 – Блок-схема инициализации клиента.

2.5. Передаваемые сообщения

В качестве передаваемого сообщения между клиентом и сервером используются объекты класса Message. Он состоит из двух полей: текстовое сообщение и список доступных для клиентов игр. Для того, чтобы сервер и клиент понимали друг друга, существует разные виды текстовых сообщений, такие как:

- «close» – сообщение, получаемое сервером. Означает выход клиента из приложения. При получении такого сообщения сервер закрывает соединение с вышедшим клиентом, а также останавливает поток, который обрабатывал его сообщения. После чего отправляет сообщение всем оставшимся клиентам, о том, что кто-то вышел.

- «login» – сообщение, получаемое сервером. Означает ввод пользователем своего имени и разрешение на авторизацию. После получения данного сообщения сервер отсылает сообщение-подтверждение клиенту.

- «game\_upd» – сообщение, получаемое сервером. Означает отправку клиентом изменённых данных списка доступных игр. После получения такого сообщения, сервер отправляет полученные данные всем клиентам, при этом сервер никак не изменяет эти данные.

- «start» – сообщение, получаемое клиентом. Означает подтверждение на вход пользователя в игру. После получения, у клиента открывается окно с доступными играми. Оно приходит клиенту после отправки сообщения «login».

- «lobbies» – сообщение, получаемое клиентом. Означает получение данных о списке доступных игр. Оно приходит всем пользователям, после получения сервером сообщения «game\_upd». При получении этого сообщения происходит обновление таблицы доступных игр для ожидающих пользователей, а также смена графического интерфейса на этапе игры.

Вторым полем класса Message, как было сказано ранее, является список доступных игр. В качестве элемента данного списка используется объект класса GField, который включает в себя объекты классов Player и Card. Каждый объект класса GField хранит в себе информацию о: списке пользователей, вошедших в комнату; колоде карт, которая «перетасовывается» в конструкторе этого класса; а также отбое карт. Как говорилось ранее, для передачи списка объектов они должны реализовывать интерфейс Serializable. В Java имеются такие компоненты, как: ObservableArrayList и ArrayList, однако первый список невозможно сериализовать, поэтому был использован ArrayList.

Алгоритм обработки сообщений от клиента сервером прилагается на рисунке 2.2.



Рисунок 2.2 – блок-схема алгоритма обработки сообщений сервером

2.6. Графическое взаимодействие

При открытии окна с таблицей игр пользователю приходит сообщение об уже существующих играх, данные о которых хранятся в объекте Message. Так, например, как только один из пользователей создаст, выйдет, удалит или войдёт в игру, у других пользователей произойдёт соответствующее изменение. На рисунке 2.3 представлено данное изменение.

Как только второй клиент войдёт в игру, соответствующая метка изменится на имя этого пользователя, это позволяет определить, сколько человек находится в комнате.

Когда все три игрока зайдут в одно и то же лобби, отобразятся карты, и появится информация о порядке производимых ходов. Это показано на рисунке 2.4.

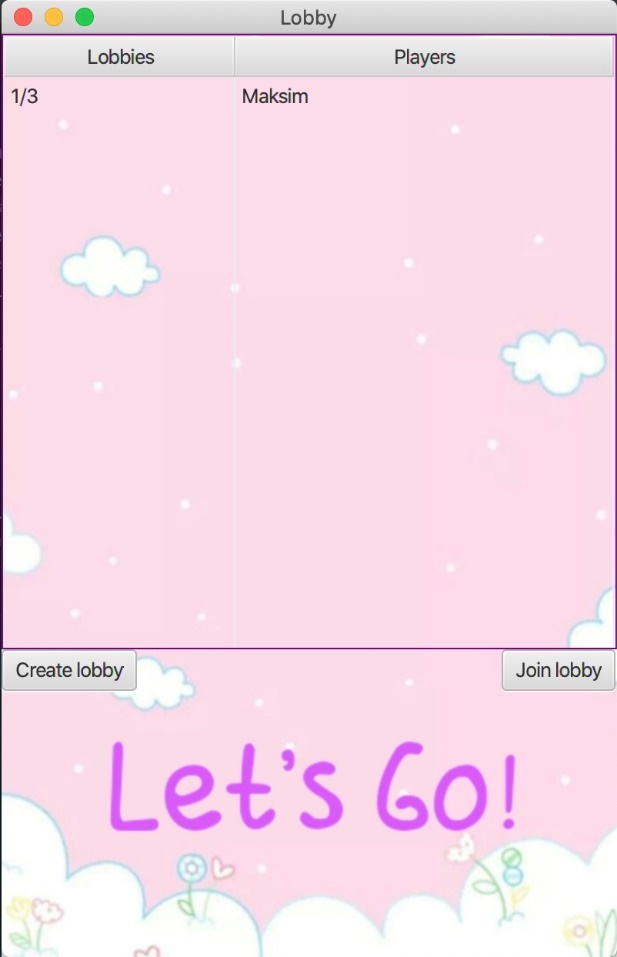


Рисунок 2.3 – Изменение таблицы игр.



Рисунок 2.4 – Начало игры.

Во время игры у каждой карты есть определённое действие, при этом, каждый раз, когда она вступает в игру, появляется сообщение об этом действии и происходит графическое изменение. Данное графическое взаимодействие представлено на рисунке 2.5.



Рисунок 2.5 – Отображение действия карты.

Также было принято решение улучшить интерфейс путём добавления стилей в файлах css и использованием анимаций. При наведении мыши на карту в своей руке, она станет непрозрачной и увеличится в размерах, а при убирании курсора с изображения, карта вновь станет затемнённой и стандартных размеров. Это было реализовано с использованием класса FadeTransition, который позволяет реализовать работу с анимациями появления и исчезновения, и свойства scale, благодаря которым можно изменить размер карты, в файле game.css [4]. Пример анимации приводится на рисунке 2.6.



Рисунок 2.6 – Анимация при наведении на карту.

В каждом окне программы на задний фон было помещено изображение с использованием компонента ImageView и класса Image. Это позволило сделать интерфейс более живым, насыщенным и разнообразным [5].

С помощью css были изменены стандартные размеры шрифтов у объектов класса Label, что сделало их более читаемыми и заметными, было изменено оформление таблицы доступных лобби в главном меню, что сделало возможным установление заднего фона для самой таблицы, а не только для целого окна.

2.7. Системные сообщения

При работе клиента с сервером могут возникать обстоятельства, при которых сервер попытается отправить сообщение на закрытый сокет, или же клиент не сможет вовсе установить соединение. Для таких случаев, как у сервера, так и у клиента существует вывод специальных сообщений, описывающих определённую ошибку соединения.

Если клиент потеряет связь с сервером, например из-за выключения компьютера или потери соединения с локальной сетью, со стороны сервера произойдёт ошибка записи в сокет. Соответствующее сообщение придёт на сервер, и он отключит данного клиента.

Также, если клиент при подключении к серверу не будет подключён к локальной сети ему выведется соответствующее сообщение, и программа выключится. Данное извещение демонстрируется на рисунке 2.7.

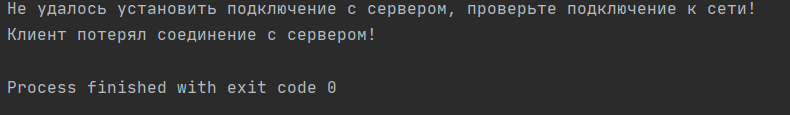


Рисунок 2.7 – Ошибка установки соединения.

При отправлении дейтаграммы клиенту, сервер так же выводит в консоль системное сообщение, которое содержит информацию об ip-адресе и порте клиента, а также порт работы TCP сервера. Сообщение показано на рисунке 2.8.

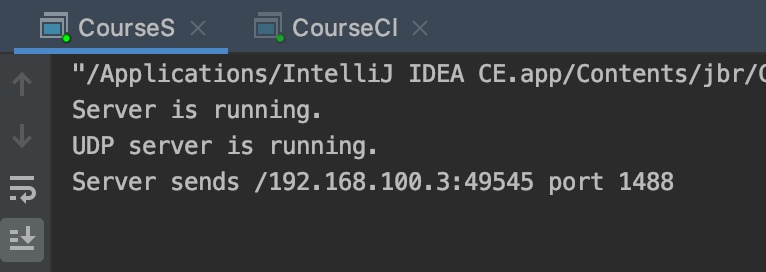


Рисунок 2.8 – Системное сообщение от UDP сервера

3. Руководство пользователя

3.1. Правила игры

В начале игры каждому игроку выдаётся по 5 карт. Также одна карта кладётся на стол, причём она обязательно должна быть шестёркой, девяткой либо дамой. Первый игрок кладёт на неё карту соответствующей масти или номинала. В игре у всех карт, кроме шести, девяти и дамы имеются специальные действия. Действия у карт:

- «7» – Следующий игрок после положившего берёт две карты из колоды.

- «8» – Игрок, положивший данную карту, выбирает карту, которую должны положить.

- «10» – Следующий игрок после положившего вытягивает карту у положившего игрока.

- «J» – Является козырной картой, на которую не действуют ограничения масти, номинала и правила восьмёрки. Игрок, положивший данную карту, выбирает масть, которую должны положить.

- «К» – Игроки меняются картами по кругу, при этом, если король последняя карта в руке, побеждает следующий игрок.

- «А» – Следующий игрок, после положившего пропускает ход.

Если игроку нечем ходить, он обязан взять карту, если даже после этого вариантов для хода нет, он пропускает его.

Игра заканчивается, когда у одного из игроков не остаётся карт на руках. Он считается победителем.

3.2. Работа с программой

При запуске программы появляется окно авторизации для ввода имени пользователя. Необходимо ввести имя пользователя в игре и нажать кнопку войти, как демонстрирует рисунок 3.1.

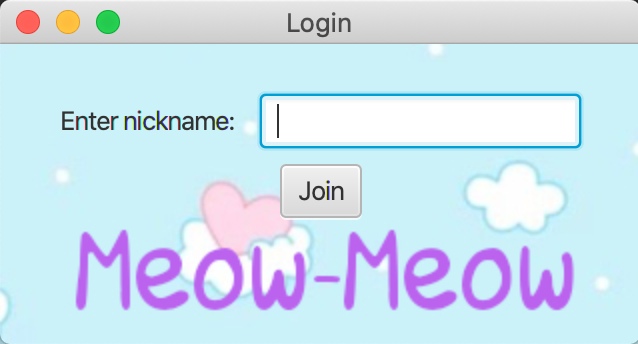


Рисунок 3.1 – Окно авторизации.

После чего открывается игровое меню, с таблицей доступных игр. На нем имеются кнопки «Create lobby» и «Join lobby». При нажатии на первую, в таблицу игр добавляется новая комната и у создавшего комнату клиента появляется игровое окно, что иллюстрирует рисунок 3.2.

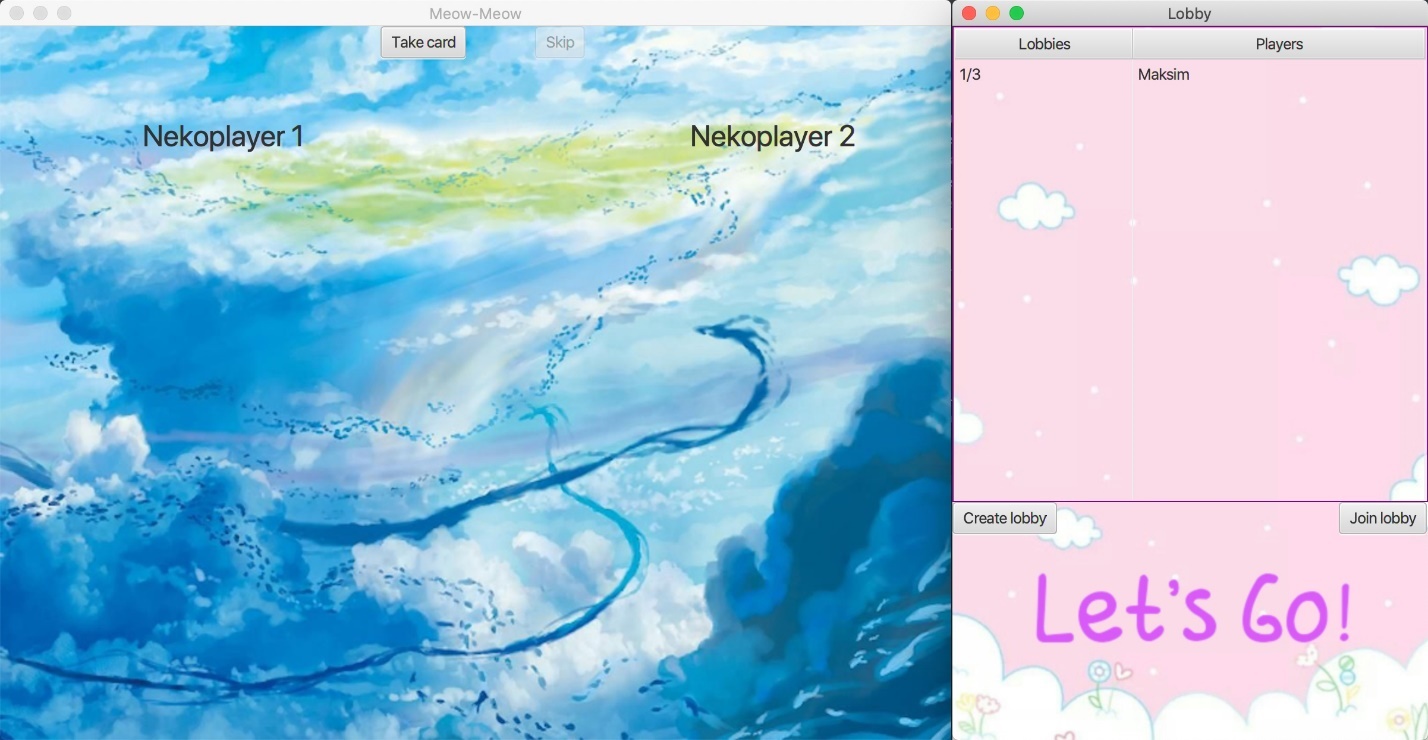


Рисунок 3.2 – Игровое меню и окно.

Для того, чтобы войти в уже существующее лобби необходимо выбрать его, и нажать на кнопку «Join lobby». Откроется то же игровое окно.

Как только все игроки присоединились к комнате, начинается игра. Согласно правилам игры, на столе находится либо шесть, либо девять, либо дама. На экране написано, чей ход на данный момент. Также присутствуют кнопки «Take card» и «Skip». В один и тот же момент времени одна из них заблокирована, а другая нет. При нажатии на одну из них происходит смена блокировки. Начало игры представлено на рисунке 3.3.

Во время игры всегда известно, кто ходит на данный момент и какие действия выполняет положенная карта. Данная информация находится посередине, в верхней части окна игрового поля.

Как только появится победитель, у каждого игрока появится извещение о нём. При его закрытии или нажатии на кнопку «Ок» игровое окно закрывается и появляется игровое меню. На рисунке 3.4 показано данное сообщение.



Рисунок 3.3 – Начало игры.



Рисунок 3.4 – Сообщение о победителе.

Заключение

В рамках данного курсового проекта было разработано игровое программное средство «Meow-Meow». Согласно поставленным задачам, в данном приложении были реализованы следующие функции:

- создание сервера и клиента;

- подключение клиента к удалённому серверу в локальной сети;

- передача объектов между сервером и клиентами;

- реализация взаимодействия интерфейса между игроками;

Для успешного создания данного игрового программного средства было необходимо изучить возможности среды программирования Java, изучить теорию работы сервера с клиентом, прочитать методические указания по устройству TCP и UDP протоколов.

Существует множество способов дальнейшего улучшения приложения. Например, усовершенствовать игровые алгоритмы, добавить чат в игру для удобной коммуникации между игроками.

Данное приложение позволит не только приятно провести время, но и также развить логическое мышление и способность просчитывать наилучшие варианты ходов.

Список использованных источников

[1] Руководство по IntelliJ IDEA. – Режим доступа <https://www.jetbrains.com/ru-ru/idea/documentation/>

[2] Сетевое программирование с сокетами и каналами. – Режим доступа <http://javatutor.net/books/tiej/socket>

[3] Посылка широковещательных сообщений. – Режим доступа <http://www.frolov-lib.ru/programming/javasamples/vol8/vol8_9/index.html>

[4] Руководство по языку программирования Java. – Режим доступа <https://metanit.com/java/tutorial/>

[5] Шилдт, Герберт Java 8. Руководство для начинающих: / Шилдт, Герберт. – СПб.: ООО "Альфа-книга', 2018. – 1488 с.

Приложение А

**Исходный код программы**

package main;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import java.io.IOException;

import java.net.ServerSocket;

import java.net.Socket;

public class Server {

static final int PORT = 1488;

private ObservableList<ClientHandler> clients = FXCollections.observableArrayList();

public ObservableList<GField> gamesArray;

private static UDPMessage connectionListener;

public Server() {

Socket clientSocket = null;

ServerSocket serverSocket = null;

try {

serverSocket = new ServerSocket(PORT);

connectionListener = new

UDPMessage(serverSocket.getLocalPort());

System.out.println("Server is running.");

while (true) {

clientSocket = serverSocket.accept();

ClientHandler client = new

ClientHandler(clientSocket, this);

clients.add(client);

new Thread(client).start();

}

}

catch (IOException ex) {

ex.printStackTrace();

}

finally {

try {

if (clientSocket != null) {

clientSocket.close();

}

System.out.println("Server stopped.");

if (serverSocket != null) {

serverSocket.close();

}

}

catch (IOException ex) {

ex.printStackTrace();

}

}

}

public void sendMessageToAllClients(String msg, ObservableList<GField> gamesList) {

for (ClientHandler client : clients) {

client.sendMsg(msg, gamesList);

}

}

public void removeClient(ClientHandler client) {

clients.remove(client);

}

}