ГУАП

КАФЕДРА № 51

ПРЕПОДАВАТЕЛЬ

доцент, кандидат техн. наук		Е.М.Линский
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ
ПО КУРСОВОЙ РАБОТЕ (ПРОЕКТУ)

ПОКЕР НА СОКЕТАХ

по курсу: ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №	5611		П.П.Недошивин
		подпись, дата	инициалы, фамилия

Санкт-Петербург 2018

СОДЕРЖАНИЕ

1 ПОСТАНОВКА ЗАДАЧИ

Реализовать клиент–серверное приложение покер, используя сокеты для соединения пользователей.

1.1 ПРАВИЛА ИГРЫ

В покере используется стандартная колода из 52 листов с равнозначными мастями. Играет несколько участников (2 до 5 за одним столом). Значения карт располагаются по нисходящей от туза и далее (король, дама, валет, 10. . . 2). Туз может рассматриваться и как младшая карта для образования последовательности (стрит) до 5 включительно, и как старшая (в комбинации туз — король — дама — валет — 10). Игра состоит из нескольких фаз, то есть раундов торговли. Каждая из них начинается с обязательных (вынужденных) ставок (большой блайнд и малый блайнд) и раздачи новых карт. После раздачи карт каждый игрок имеет возможность сделать ставку или выйти из игры. Победителем считается тот, чья комбинация из пяти карт окажется лучшей, или тот, кто сможет вытеснить из игры других игроков с помощью ставок или блеф-ставок и останется один до вскрытия карт [1].

1.1.1 КОМБИНАЦИИ КАРТ

Возможные комбинации карт в порядке убывания достоинства:

1. Роял-флэш: не является отдельной комбинацией, а является частным случаем стрит-флэша, старшим из всех возможных, и состоит из 5 старших (туз, король, дама, валет, десять) карт одной масти, например: Т♡К♡Д♡В♡10♡.

2. Стрит-флэш: любые пять карт одной масти по порядку, например: 9♠8♠7♠6♠5♠. Туз может как начинать порядок, так и заканчивать его.

3. Каре: четыре карты одного достоинства, например: 3♡3♢3♣3♠.

4. Фулл-хаус: одна тройка и одна пара, например: 10♡10♢10♠8♣8♡.

5. Флэш: пять карт одной масти, например: К♠В♠8♠4♠3♠.

6. Стрит: пять карт по порядку любых мастей, например: 5♢4♡3♠2♢T♢. Туз может как начинать порядок, так и заканчивать его. В данном примере T♢ начинает комбинацию и его достоинство оценивается в единицу, а 5♢ считается старшей картой.

7. Сет: три карты одного достоинства, например: 7♣7♡7♠.

8. Две пары: две пары карт, например: 8♣8♠4♡4♣.

9. Одна пара: две карты одного достоинства, например: 9♡ 9♠.

10. Старшая карта: ни одна из вышеописанных комбинаций, например (комбинация называется «старший туз»): T♢10♢9♠5♣4♣.

При совпадении комбинаций более сильной является комбинация со старшими картами, например 8♣8♠4♡4♣2♠ старше, чем 7♣7♠5♡5♣K♠, а комбинация 6♠5♢4♡3♠2♢ старше, чем 5♢4♡3♠2♢T♢.
При совпадении комбинаций и старшей карты (если он находится среди 5 открытых карт) выигрыш делится поровну между игроками с одинаковой комбинацией. В случае совпадения первой старшей карты сравнивается с второй или третий (в случае пары и трёх старших карт).

1.1.2 ТОРГОВЛЯ

Перед сдачей все игроки вносят одинаковую начальную ставку. В процессе торговли игрок может делать следующие ставки:

1. Поставить (bet) — сделать ставку;

2. Поставить все деньги (all in) — сделать ставку на все деньги;

3. Ответить (call) — поставить столько же, сколько соперник — уравнять;

4. Поднять (raise) — увеличить ставку — поставить больше, чем соперники;

5. Пасовать (fold) — отказаться от дальнейшего участия и сбросить карты;

6. Отметиться (check) — в ситуациях, когда ставки не были сделаны соперниками или ставка уже была сделана вслепую — не добавлять ставку, оставить как есть.

Круг торговли заканчивается, когда все соперники сделали равные ставки или сбросили карты. Сделанные ставки складываются в банк. Если в процессе последнего круга торговли осталось больше одного игрока, то карты открываются и комбинации игроков сравниваются между собой. Если только один игрок остался в игре, то он забирает банк. Если больше одного игрока победили в игре, банк делится поровну между всеми выигравшими.

1.2 ВОЗМОЖНОСТИ ПРОГРАММЫ

1. Многопользовательская игра через локальную сеть
2. Получение статистики баланса денег в игре
3. Выбор стола, к которому можно подключиться
4. Аутентификация пользователей

1.3 СЕТЕВОЕ ВЗАИМОДЕЙСТВИЕ

Для того, чтобы реализовать многопользовательскую игру, выбраны сокеты, работающие по протоколу TCP, так как при передаче данных между игроками недопустимо наличие ошибок. Следует заранее запустить сервер, который будет принимать клиентов, авторизовать их в системе, затем посылать информацию о столах клиенту и, наконец, обеспечивать, взаимодействие пользователей (рис.1).

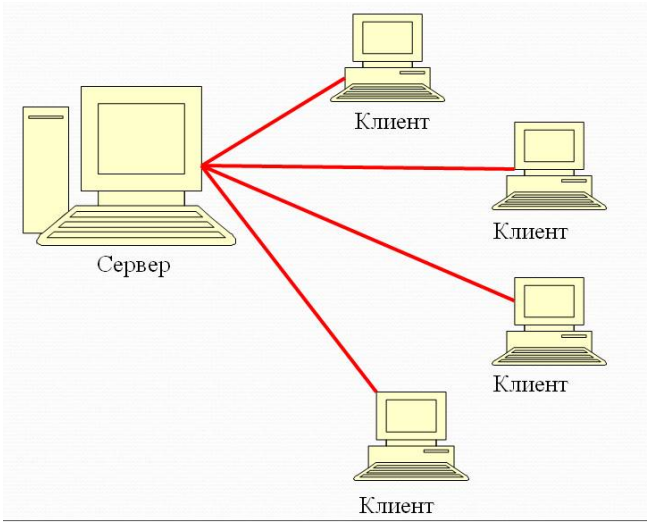

Рисунок 1 — Архитектура программы сервер-клиент.

1.4 ИНТЕРФЕЙС ПРОГРАММЫ

Для обеспечения удобного использования программы пользователями предлагается использовать библиотеку JavaFX для реализации графического интерфейса приложения [2].

1.5 ДОПОЛНИТЕЛЬНЫЕ ТРЕБОВАНИЯ

Проект включает junit5-тесты модели, собирается с помощью Maven и выложен на GitHub: https://github.com/PavelNedoshivin/TexasHoldem.

2 ИНСТРУКЦИЯ ПОЛЬЗОВАТЕЛЯ

Перед запуском приложения Texas Holdem должен быть запущен сервер! Для подключения к серверу нужно ввести ip и порт (рис. 2).

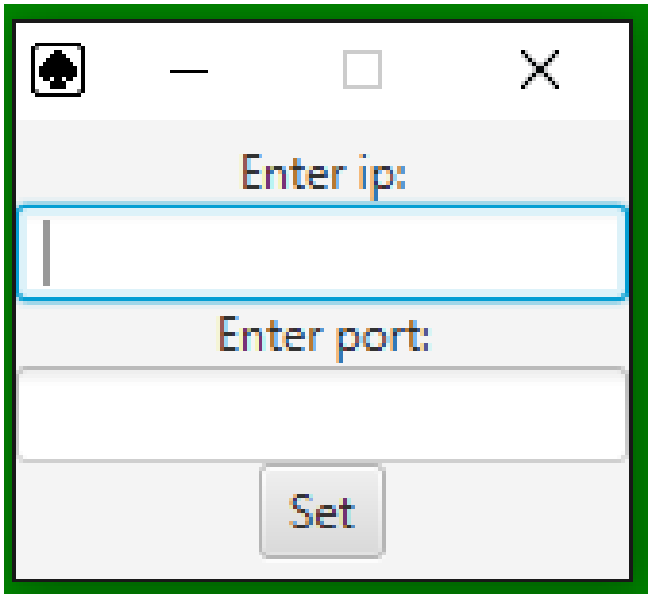

Рисунок 2 — Окно подключения.

При запуске программы выводится приглашение на авторизацию: регистрация или вход по логину и паролю (рис. 3).

Далее пользователь попадает на экран выбора стола (рис. 4). Пользователь может присоединиться к любому из 5 столов. На каждом столе не может быть больше 4 игроков (если за столом 4 игрока, то стол недоступен для подключения). Для запуска партии необходимо минимум 2 игрока за одним столом (просмотр статистики возможен только во время партии). Пользователь может в любой момент закрыть приложение, нажав крестик в верхем правом углу.

Во время игры пользователю доступны кнопки (рис. 5):

1. Call/Check — сравнять ставку/пропустить ход
2. Bet/Raise — сделать ставку с помощью бегунка справа
3. Fold — сбросить карты
4. Think — случайный выбор хода

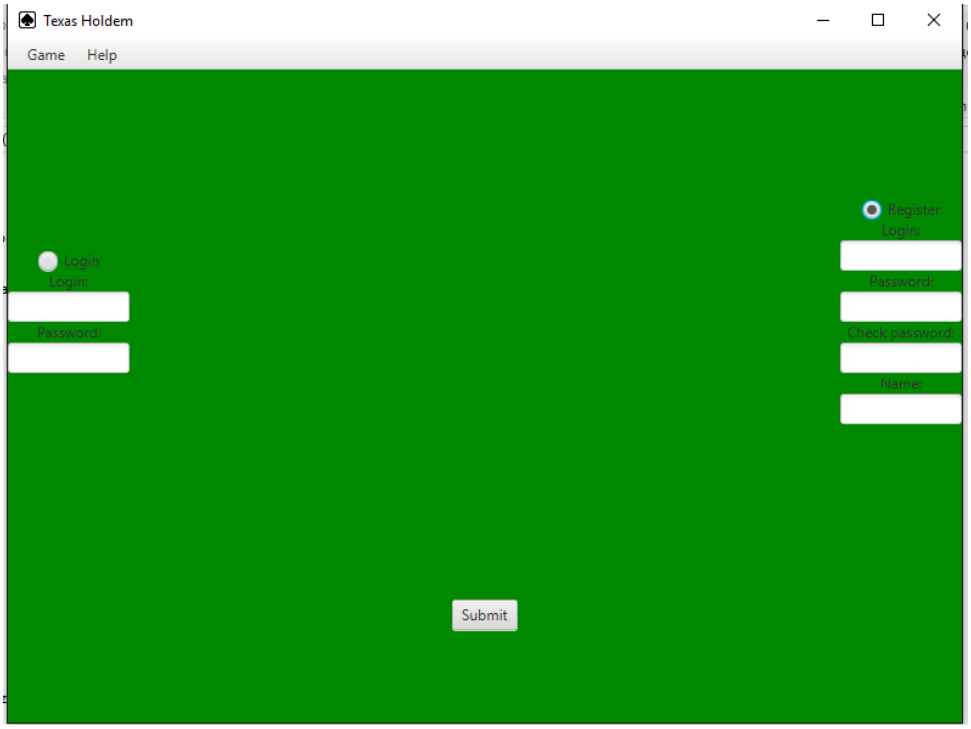

Рисунок 3 — Начальный экран приложения — аутентификация пользователя.

5. Deal — раздать карты в начале новой партии

6. Next — перейти к следующему раунду (рис. 6)

В конце партии определяется победитель (рис. 7).

На вкладке Game также доступно окно статистики баланса (рис. 8), а на вкладке Help окно помощи пользователю (рис. 9) и окно информации об авторе программы (рис. 10).

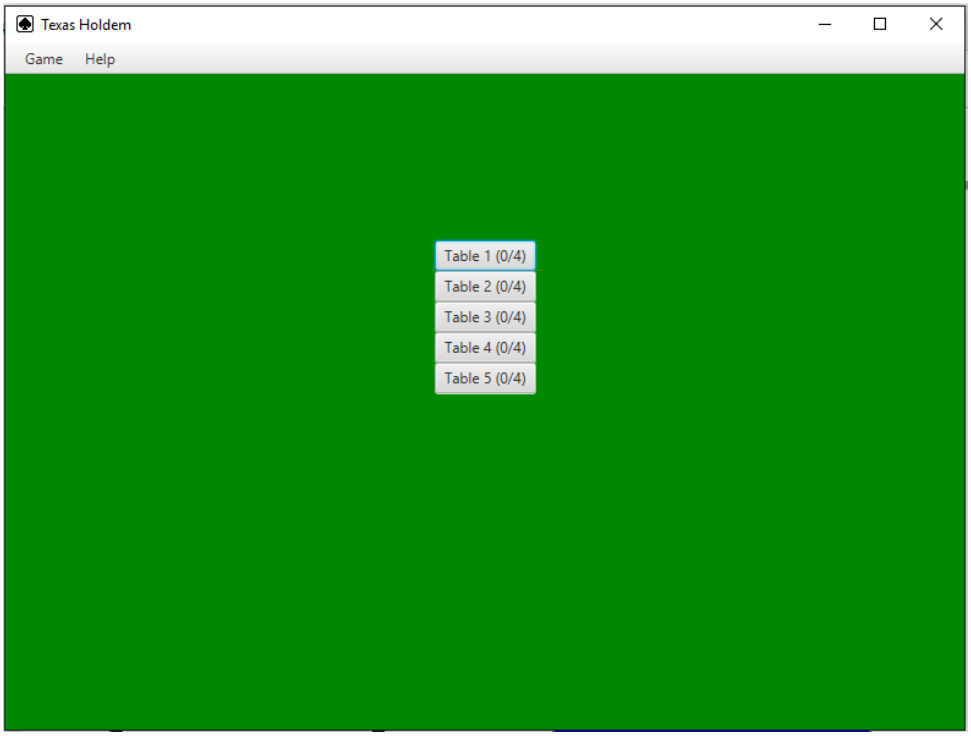

Рисунок 4 — Экран выбора стола.

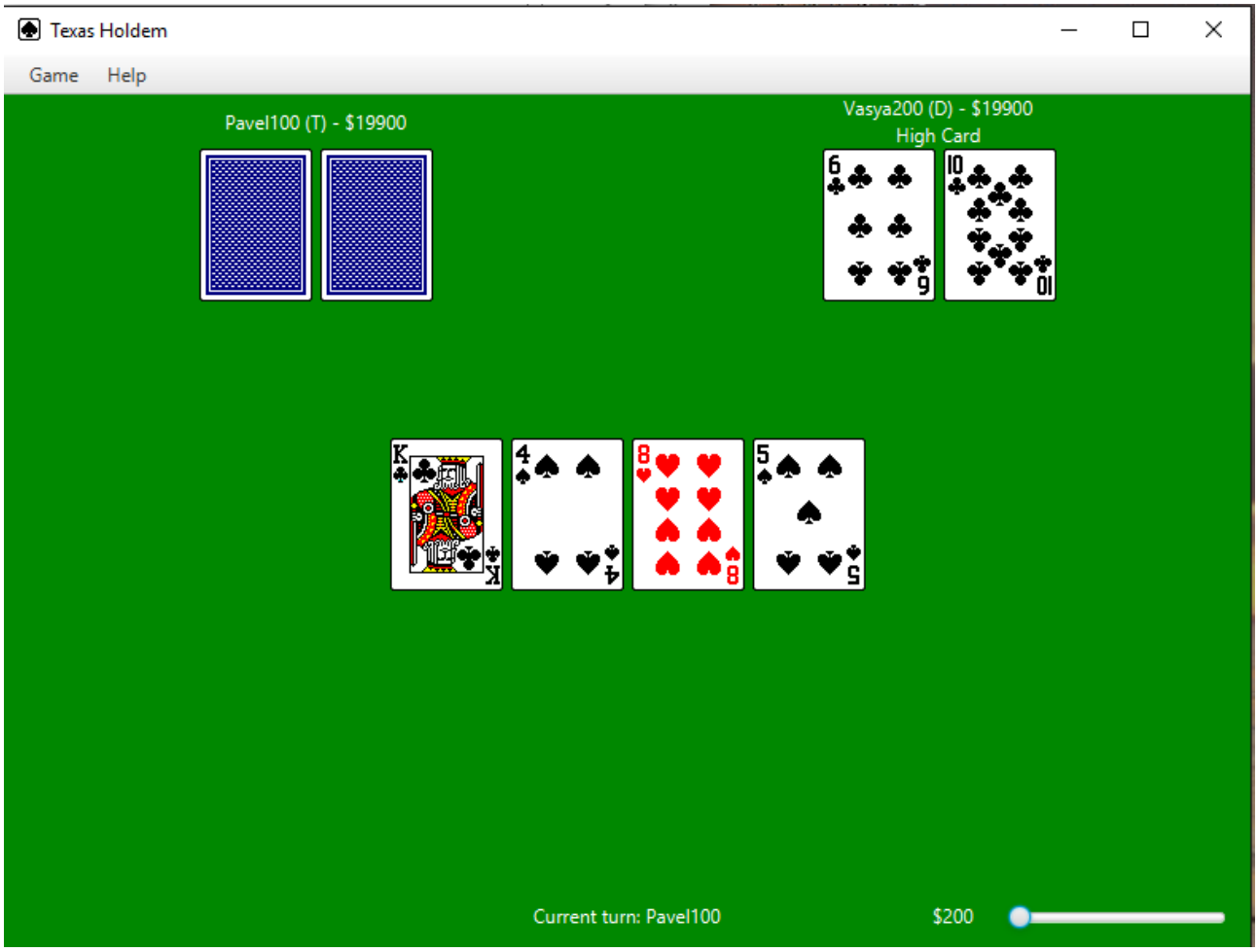

Рисунок 5 — Экран во время партии.

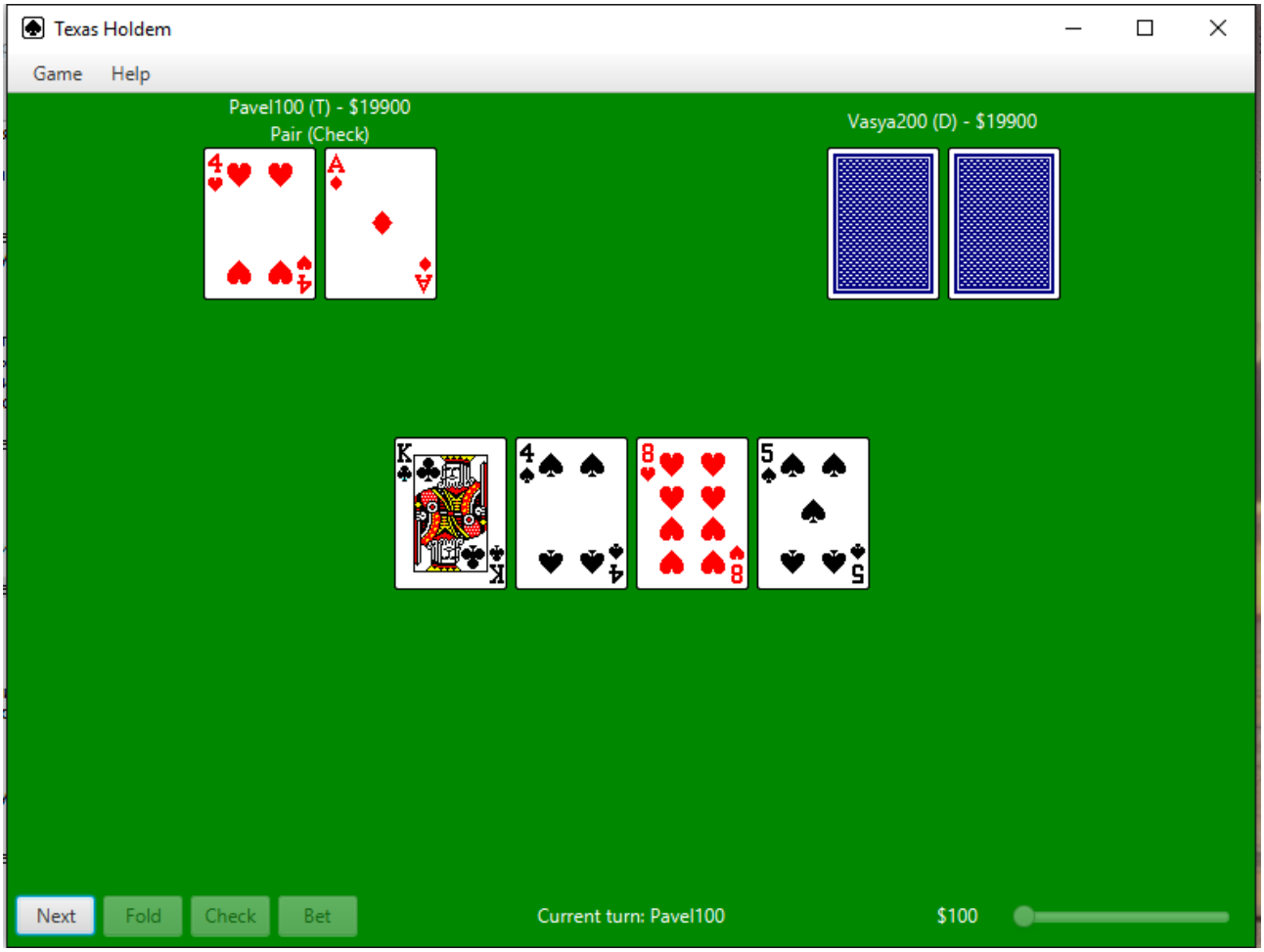

Рисунок 6 — Начало нового раунда торговли.

Рисунок 7 — Определение победителя.

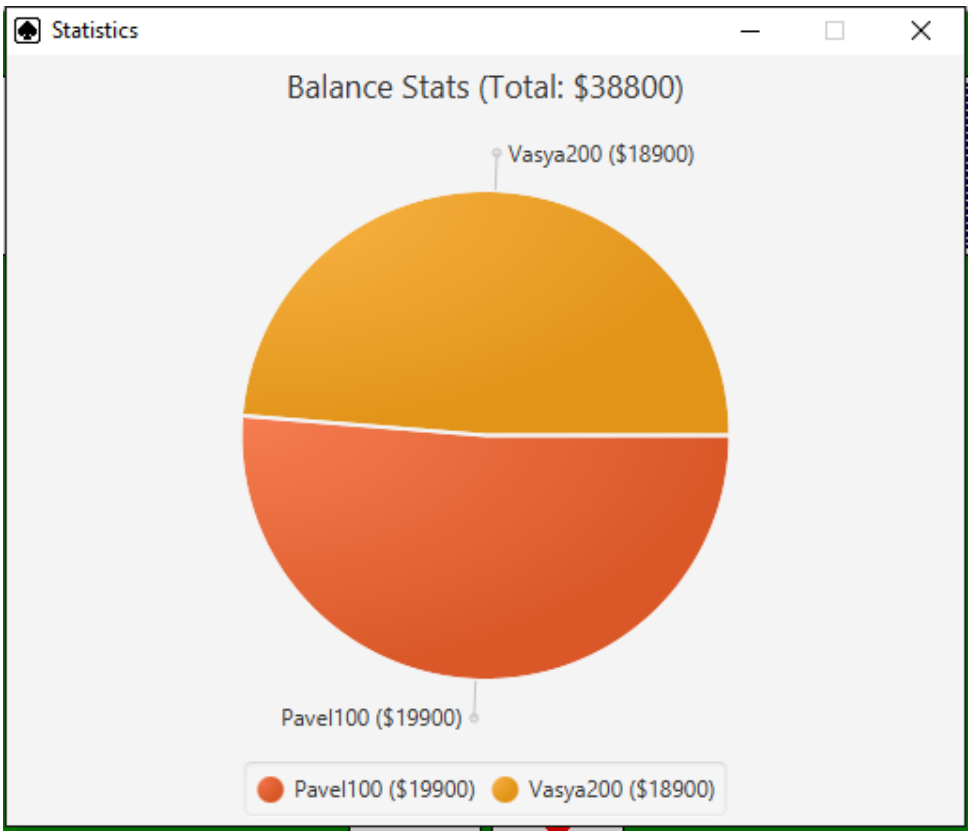

Рисунок 8 — Статистика баланса денег в игре.

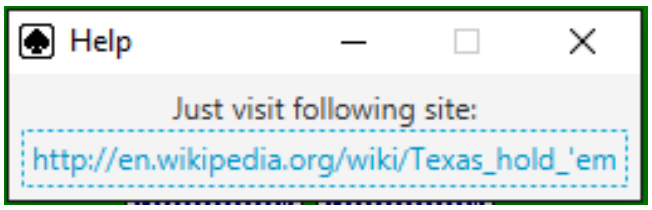

Рисунок 9 — Помощь начинающему пользователю.

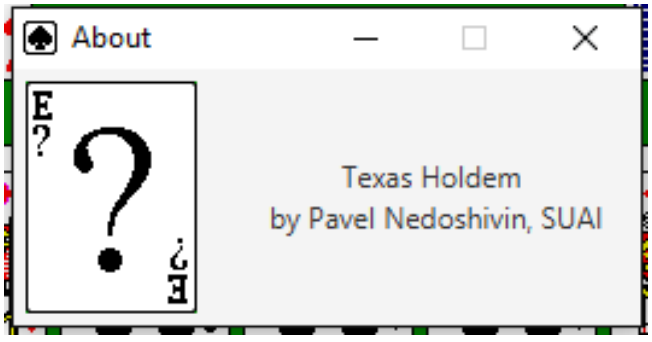

Рисунок 10 — Сведения о разработчике.

3 ОБЩАЯ СТРУКТУРА ПРОГРАММЫ

Программа выполнена по схеме разделения Model–View–Controller. Краткое описание модулей программы:

1. Model — содержит внутреннее представление игровых компонентов: карты игроков, очередность ходов игроков, выполнение ходов и так далее. Также хранит модели самих игроков, колоды карт и самого стола для игры.

2. View — содержит реализацию пользовательского интерфейса на базе библиотеки JavaFX.

3. Controller — содержит сетевую часть программы, позволяет отправлять данные, полученные от пользователя, на сервер и получать от сервера измененную модель стола.

Сервер, к которому подключаются клиенты и на котором находится модель игры, расположен в локальной сети. Локальный сервер может запускаться на ПК. Игроки подключаются к локальному серверу. Сервер является отдельной составляющей приложения без графического интерфейса, на нем хранятся данные пользователей, статистика, модель игры и тому подобное.

Общая структура программы представлена на рис. 11.

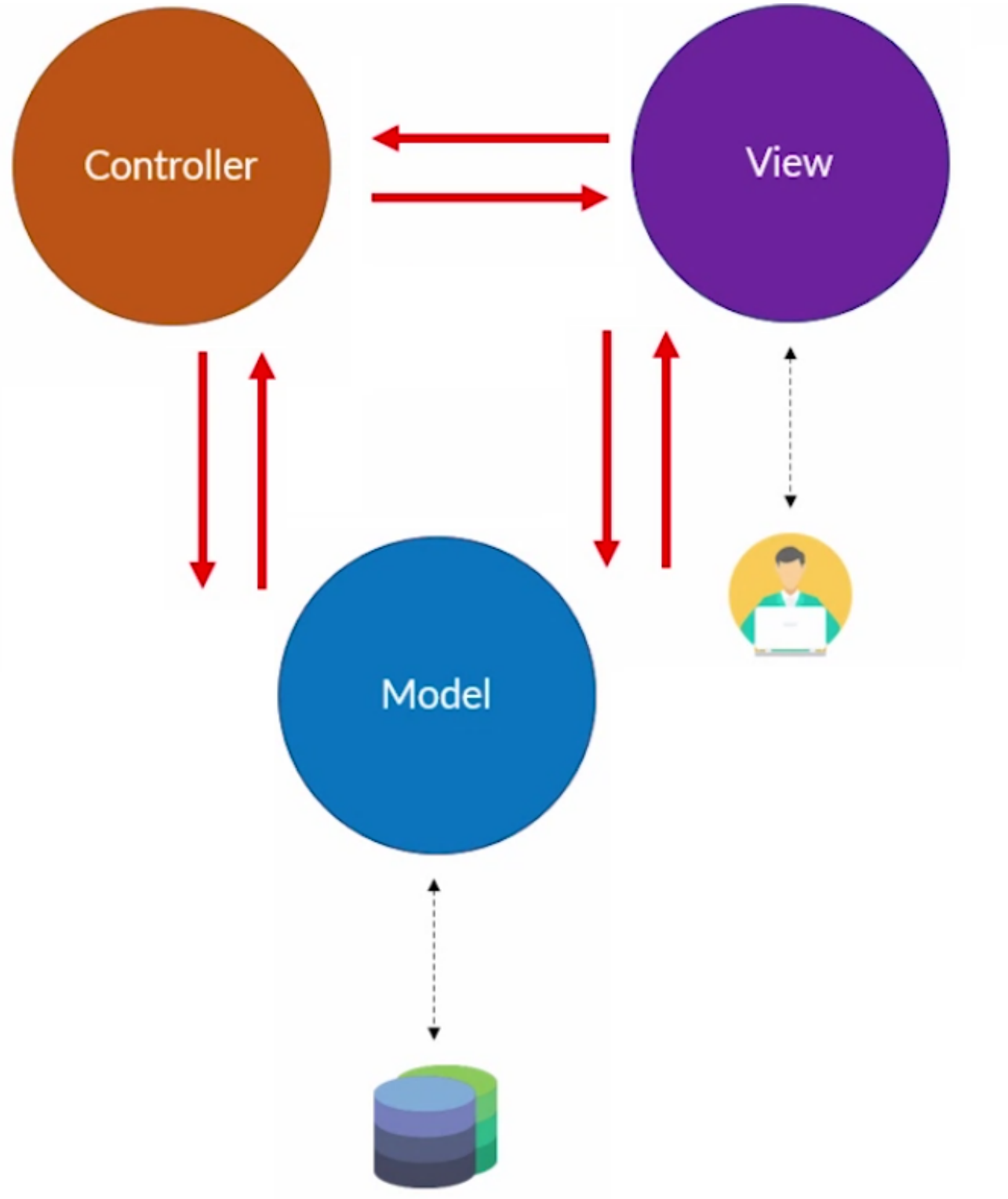

Рисунок 11 — Структура программы.

4 ОПИСАНИЕ РЕАЛИЗАЦИИ

Здесь приводится описание наиболее важных классов программы.

Основные компоненты модели:

public class CardComparator — класс, предназначенный для сравнения значений карт.

public enum Card — представление каждой карты из стандартной колоды в 52 карты.

public enum HandCategory — представление комбинации карт, имеющейся в руках игрока.

public enum PlayerStatus — представление статуса игрока (какой ход сделал, может ли совершить ход).

public class Deck — класс, содержащий список карт в колоде.

public List<Card> generate() — метод, генерирующий стандартную колоду в 52 карты. Возвращает колоду.

public void shuffle() — метод, перемешивающий карты в колоде.

public Card draw() — метод, вытаскивающий карту из колоды. Возвращает вынутую карту.

public class Pot — класс, представляющий банк (некоторое количество денег, которое является призом для победителя партии).

public void addPlayer(Player player) — метод, добавляющий к кассе нового игрока. Принимает аргумент player – игрок, отправляющий деньги в кассу.

public void cleanUpWinnerList() — метод, очищающий список победителей в партии.

public void addWinner(Player player) — метод, добавляющий игрока в список победителей. Принимает аргумент player – игрок, победивший в партии.

public class Hand — класс, предназначенный для анализа комбинации у игрока.

public class Player — класс, содержащий модель игрока (имя, баланс, статус).

public class Table — класс, представляющий стол, где происходит партия между игроками.

Основные компонеты графического интерфейса:

public class Main — основной класс, запускающий приложение.
public void start(Stage stage) — метод, запускающий графический интерфейс. Принимает аргумент stage – начальная панель, полученная из fxml файла.
public class MainController — класс, управляющий основным окном приложения.
public void initialize(URL arg0, ResourceBundle arg1) — метод, задающий начальные параметры элементов окна. Принимает аргументы arg0 – адрес fxml файла, arg1 – адрес директории с другими файлами.
public void loadImages() — метод, заранее загружающий изображения карт.
public void statistics() — метод, показывающий окно статистики.
public void help() — метод, показывающий окно помощи.
public void about() — метод, показывающий окно сведений об авторе.
public void updateTopPane() — метод, обновляющий верхнюю панель с картами игроков.
public void updateCenterPane() — метод, обновляющий центральную панель с общими картами.
public void updateBottomPane() — метод, обновляющий нижнюю панель с картами конкретного игрока.
public void updatePlayerDetails() — метод, обновляющий нижнюю метку с именем игрока и бегунок ставок.
public void updateButtonState() — метод, определяющий состояние кнопок в конкретный момент времени.
public void toggleButtons(Boolean action0, Boolean action1, Boolean action2, Boolean action3, Boolean action4) — метод, определяющий, нужно ли выключить кнопки. Принимает аргументы action0 – кнопка Deal/Next/Think, action1 – кнопка Fold, action2 – кнопка Check/Call, action3 – кнопка Bet/Raise/All in, action4 – бегунок размера ставки.
public void updateAction0() — метод, обновляющий надпись на первой кнопке.
private class UpdateThread — класс-поток, отвечающий за отправку сигнала клиенту об изменении модели.
private class ReceiverThread — класс-поток, отвечающий за отрисовку стола, полученного с сервера.
public class StatisticsController — класс, управляющий окном статистики.
public class ImageScroller — класс, представляющий элемент ImageView,

который пробегает по списку изображений карт.
public void scroll(ImageView imageView, Image image) — метод, устанавливающий изображение. Принимает аргументы imageView – место, где будет находиться изображение, image – новое изображение.

Основные компоненты контроллера:
public class Server — класс, представляющий собой постоянно запущенный сервер, который работает с клиентами.
private class ServerInputThread — класс-поток, отвечающий за авторизацию новых пользователей и прием столов.
private class ServerOutputThread — класс-поток, отвечающий за отправку стола пользователям.
public class Client — класс, являющийся сетевой компонентой приложения у пользователя.
private class ClientInputThread — класс-поток, отвечающий за отправку аутентификационных данных серверу и прием столов.
private class ClientOutputThread — класс-поток, отвечающий за отправку стола на сервер.

СПИСОК ЛИТЕРАТУРЫ

[1] Покер — Википедия
https://ru.wikipedia.org/wiki/Покер (2018)

[2] Руководства по JavaFX
https://o7planning.org/ru/11009/javafx