Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

**КУРСОВОЙ ПРОЕКТ**

Создание собственного GUI приложения

по дисциплине: «Технологии программирования (Java)»

Выполнил студент гр. 3530901/80202 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ К.А. Иванов

(подпись)

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.И. Глухих

(подпись)

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

.

Санкт-Петербург

2020

**Оглавление**

[Задание на выполнение курсового проекта 3](#_Toc54210419)

[Введение 4](#_Toc54210420)

[Работа приложения 5](#_Toc54210421)

[Заключение 8](#_Toc54210422)

[Список использованной литературы: 9](#_Toc54210423)

# Задание на выполнение курсового проекта

1. Тема проекта(работы): Создание собственного GUI-приложения
2. Срок сдачи законченной работы: 23 октября
3. Исходные данные к работе: Среда разработки IntelliJ IDEA, курс лекции по предмету за 2019 год, презентации и книги по курсу.
4. Содержание работы: GUI-приложение Poker

# Введение

Реализация проекта с графическим интерфейсом для игры в различные типы покера, а именно Texas Hold’em, Omaha, 7-Card Stud (на данный момент только Texas). Также с различным выбором типов посадки игроков (количество игроков одновременно) и качества этих игроков (сложность ботов) (на данный момент готовы элементарные вещи для игры в 2-ем, а также с 2 типами ботов).

Чтобы реализовать свою идею в рабочий проект я воспользовался следующими инструментами:

Язык программирование – **Kotlin**

Графические инструменты –**TornadoFX, JavaFX**

Так же я решил выбрать одну из распространенных моделей реализации, а именно **Model – View – Controller**, чуть более подробно можно объяснить так, что под

**Model** – скрыто все основные , без реализации которых просто невозможно играть в покер, например карты (Card.kt), колода(Deck.kt), какую позицию может занимать игрок в тот или иной промежуток времени за столом (сделал Fold, или же захотел идти в All-in) и т. д..

**View** – можно понять как логику для отображения и отрисовки данных для показа пользователю.

**Controller** – логика, правила игры, ход игры, связь между данными Model и отображением View.

Перейдем к разбору того, как же работает функционал игры изнутри.

# Работа приложения

Так как мой проект основан на модели MVC, пройдемся по каждому этапу, и я постараюсь Вас ввести в курс работы проекта:

**Model**

*Action*

Класс, указывающий на то, какое действие в данный момент производит игрок – будь это бот или пользователь. Так же держит два класса перечислений, один указывает на действие игрока в момент игры, а второй на то в какой позиции сейчас сама игра.

*Card*

Реализация карт, где так же хранятся два класса перечислений – масть и ранг карты (или иначе сила) – в данном классе так же имеем путь до того места, где лежат сами карты, в разные моменты игры карту можно показать или же держать в скрытом виде. Также имеет метод отрисовки, позволяет отрисовать указанную карту по ее пути в графическом интерфейсе.

*Combinations*

Класс, держащий информацию обо всех комбинациях, которые существуют и могут быть найдены во время игры (когда соответственно нужно решить исход партии, и узнать кто собрал наиболее лучшую комбинацию).

*Deck*

Колода, состоящая из карт, при обращении к классу сразу же получаем колоду с нужных размером карт (36, 52). Можно достать карту из колоды (во время раздачи игрокам и сдачи на стол), сжечь карту, и узнать размер колоды в данный момент.

*Play*

Интерфейс – держащий информацию об игроке, по которому можно создавать игроков (игрок, разные виды ботов).

*UInterface*

Интерфейс – который говорит главному окно отображения (Main View) какие методы нужно иметь чтобы была возможность следовать информации из логики и отправлять данные на отображение пользователю.

**Player**

*Human*

Реализует интерфейс Play – предоставляет (развертывает) информацию для игрока пользователя

*Low Bot*

Реализует интерфейс Play – является ботом - основная логика содержится в функции prepareForAction() – выполняет линейную логику, заключающуюся в том что на любую ставку отвечает соответственно похожим действием.

*Random Bot*

Реализует интерфейс Play – является ботом – основная логика содержится в функции *prepareForAction()* - делает свои действия исходя из полученного значения при помощи метода Random.

**Controller**

*Main Game Controller*

Хранит в себе всю логику следования игры (можно назвать, правила, по которым должна следовать игра), в который каждый раз обращается отображение (View) вызванное действием пользователя в графическом интерфейсе. При обращении к классу инициализирует начало игры. Содержит методы по началу новой раздачи, сдаче карт, реакцию в изменении модели на действие игрока, а также вызов работы бота и многие другие, см. проект.

**View**

*App –* точка запуска проекта, приложения, вызывающая Start Menu.

*Start Menu* – простейшее окно диалога с пользователем, в котором можно указать тип покера, количество игроков, и игроков которых пользователь хочет видеть на столе.

*Buttons* – Класс содержащий меню пользователя в котором он может продолжить играть раздачу (Check) поставить ставку (Bet, Raise) или же сбросить карты (Fold), так же содержит в себе окно ввода ставки и ползунок для указания свободного количества фишек.

*Hand UI –* маленький, вспомогательный класс по своему функционалу, позволяет добавлять карты для видимости в отображении, а также убирать их.

*Main View –* главное окно игры, реализация метода UInterface, взаимодействие пользователя с игрой, отображение хода игры в окне приложения, а также небольшая информация о ходе игры в чате.

# Заключение

Хочу отметить, что проект не окончен и если судить по названию (Engine – Poker), то я бы хотел реализовать что-то похожее на движок для карточных покерных игр, чтобы можно было так же в последствии добавлять и другие типы, ведь их намного больше, чем я раскрыл в данной работе.

Так же на данном этапе сдачи проекта, я свою работу над проектом еще не закончил, ведь реализована только малая часть того, что в действительности хотелось сделать, идея для меня отличная и я бы хотел видеть себя с финальной частью проекта в портфолио.

# Список использованной литературы:

1. Хабибуллин И. – Java 7. – СПб.: БХВ-Петербург, 2012
2. Joshua Bloch. Effective Java: Programming Language Guide, second edition. – ISBN 978-0-321-35667-0, 2008