**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ**

**Московский технический университет связи и информатики**

Кафедра «Математическая кибернетика и информационные технологии»

**Курсовая работа**

по дисциплине «Java-программирование»

на тему «Разработка бота в Телеграмм для игры в камень-ножницы-бумага-ящерица-Спок»

Выполнил: студент группы БСУ1901

Панов Д. А.

Проверил: старший преподаватель кафедры МКиИТ

Мосева М.С.

Москва 2021

**Оглавление**

[Цель работы](#_Toc91019364)

[1. Техническое задание на разработку](#_Toc91019365)

[2. Описание программной реализации](#_Toc91019366)

[2.1. Описание классов, используемых в программе](#_Toc91019367)

[2.2. Описание библиотек, используемых в программе](#_Toc91019368)

[2.3. Код программы](#_Toc91019369)

[3. Описание процесса тестирования](#_Toc91019370)

[Заключение](#_Toc91019372)

# 

# Цель работы

Продемонстрировать свои навыки применения концепций программирования, представленных в курсе по принципам Java. По желанию можно выполнить дополнительные действия по размещению игры на хостинге – это позволит боту работать постоянно. В ином случае созданный Телеграм бот будет работать только при работе непосредственно написанной для него программы.

# 1. Техническое задание на разработку

Разработать и выполнить написание кода для бота в Телеграмм канал, который будет позволять производить игру между двумя пользователями, для демонстрации своих знаний об основах программирования на Java. Камень-ножницы-бумага-ящерица-спок это усложненная версия классической игры с тремя элементами. Отношение каждого из элементов к другому будет представлено на рисунке ниже:

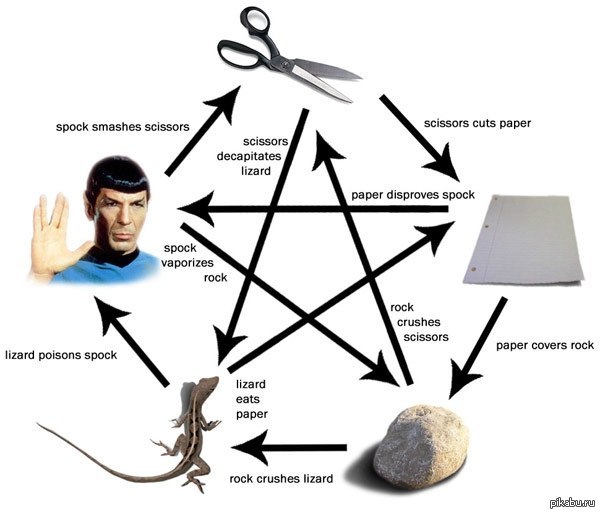


Рис. 1 – Комбинации игры.

Программа должна:

• уметь принимать запросы от двух пользователей и производить вычисления относительно их выбора

• включать хотя бы 2 класса, не считая тестовый класс;

• инкапсулировать эти и дополнительные классы;

• предоставлять хотя бы 2 поля и 2 метода в каждом классе, не включая методы получения или установки;

• для усложнения задачи выполним написание кода без базы данных

• быть способной принять ввод пользователя хотя бы один раз; всегда принимать ввод пользователя без сбоя.

# 2. Описание программной реализации

# 2.1. Описание классов, используемых в программе

* MainClass- Класс для запуска главного окна.
* AGamer- Абстрактный класс для игроков, включает в себя методы: shot()-сделал ход и win()-проверка на выигрыш.
* AI- Класс для ИИ, наследуется от AGamer, переопределяет методы shot() и win().
* MainForm- Главное окно программы.
* GameSettingsForm- Окно настроек.
* MainGameField- Игровое поле, наследуется от JPanеl.
* Player- Класс игрока, наследуется от AGamer, переопределяет методы shot() и win().

**2.2. Список библиотек, используемых в программе****:**

Основная библиотека для выполнения запросов совместно с Телеграм:

• telegram.api, при добавлении этой библиотеки она сразу подгружает дополнительные данные, которые могут потребоваться для работы с Телеграмом.

• java.util.ArrayList и java.util.List – две стандартные библиотеки для обработки массивов

**2.3. Код программы**

Код программы также можно посмотреть перейдя по ссылке <https://github.com/WouterLab/TG_botgame>

**3. Описание разработки и процесса тестирования**

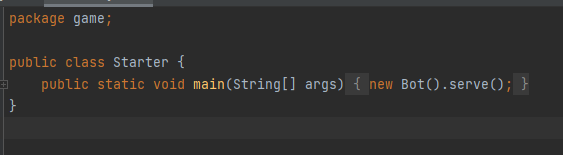


Рис. 2 – код файла Starter для запуска бота

Код файла Starter.java будет содержать команды только непосредственно для запуска основной программы. Весь основной код для работы бота будет описан в файле Bot.java

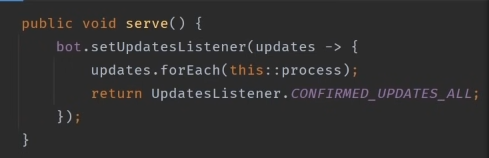


Рис 3 – Инициализация бота

На рисунке 3 демонстрируется код, в котором бот обеспечивает всю взаимосвязь с bot.api.

UpdateListener каждые 100мс проверяет, поступили ли какие-то запросы от пользователей, чтобы в дальнейшем их обработать (если конечно они поступили). Каждый процесс поступающий боту обрабатывается отдельно от всех, это позволяет сделать функция forEach.

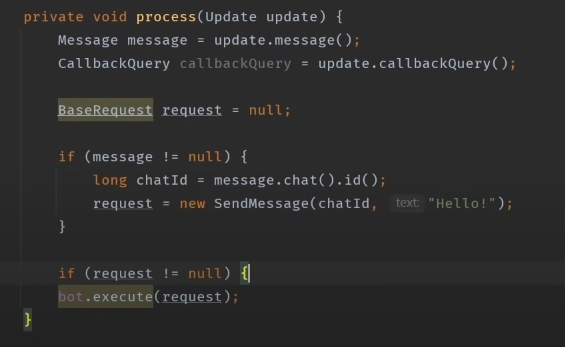


Рис. 4 – Метод process.

В этом методе бот ожидает получить какое-либо сообщение от пользователя, поэтому изначально мы проверяем, чтобы сообщение было не нулевое с помощью цикла (if message != null..).

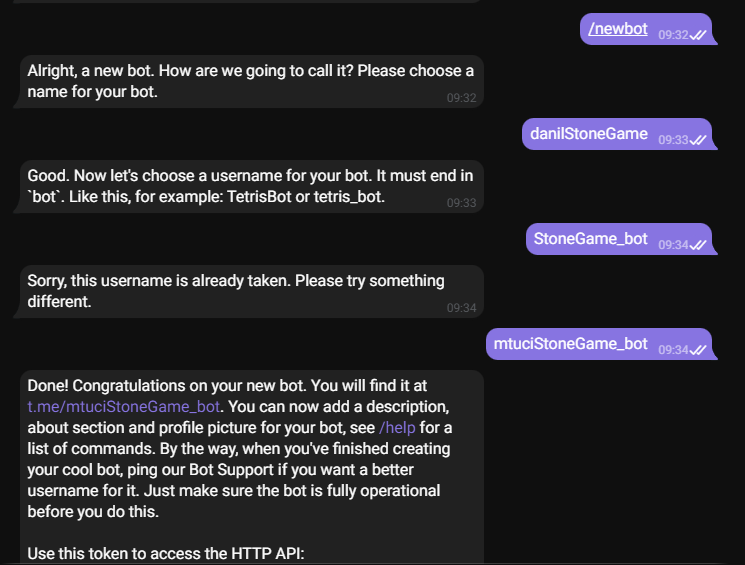


Рис. 5 – Создание бота.

Чтобы создать самого бота, который будет принимать запросы нам потребуется бот Телеграм под названием BotFather. Этот бот позволяет создать API токен (этот токен нужен для того, чтобы программа понимала для кого мы пишем код и кого нужно будет с помощью этого кода запустить) никнейм и название для нашего нового бота, чтобы пользователи могли найти и воспользоваться.

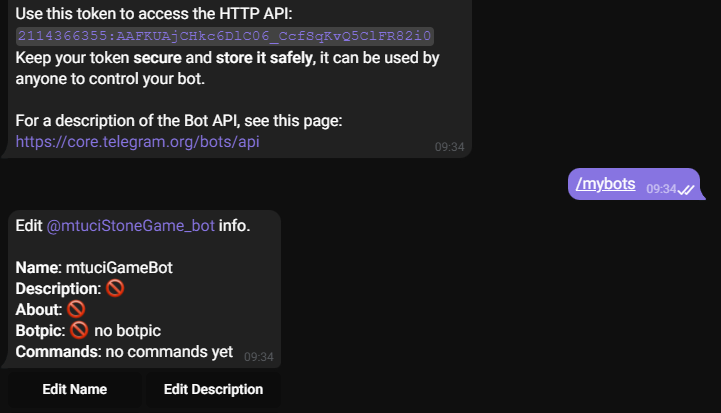


Рис. 6 – API токен и список моих ботов, созданных на текущий момент.

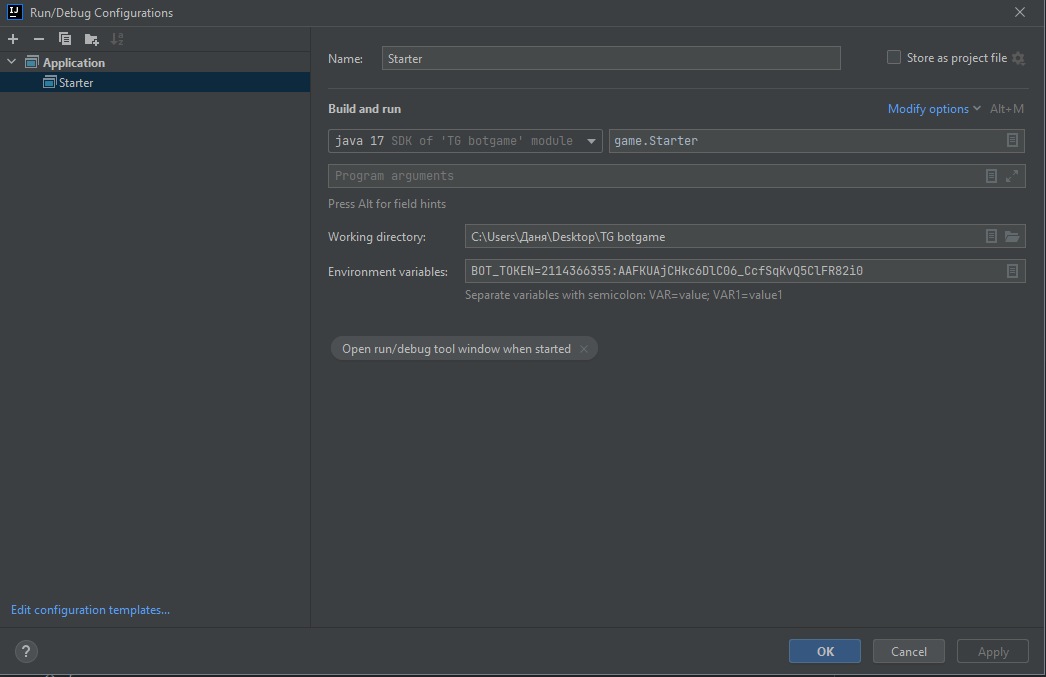


Рис. 7 – Настройки конфигурации и переменная Environment.

Переменную Environment бот получает в строке 21 файла Bot.java, эта переменная содержит в себе созданный нами ранее токен.

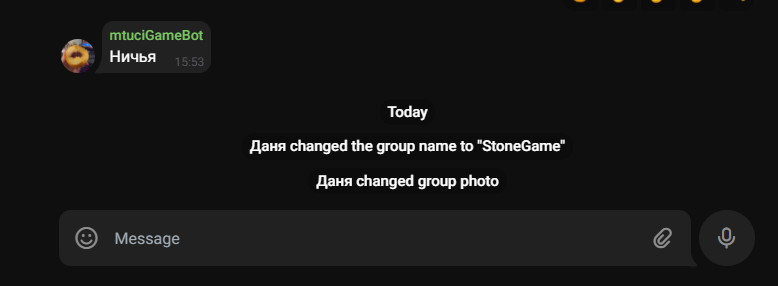


Рис. 8 – Создание канала.

На этом этапе мы создаем канал, в который добавляем нашего бота и людей, с которыми планируем проводить игры. Для теста можно добавить себя (но играть с самим собой будет не так наглядно) или своих друзей.

На рисунке 4 можно увидеть, что бот отправляет приветственное сообщение “Hello!” когда попадает в чат. Также на данном этапе бот реагирует на каждое сообщение в чате, но, чтобы ему видеть чужие сообщения нужно выдать боту права администратора.

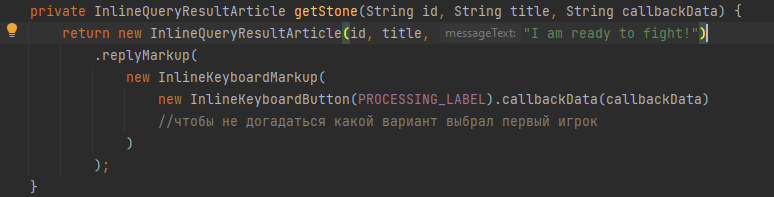


Рис. 9 – Код метода InlineQueryResultArticle.

Такое длинное название этот метод имеет из-за того, что в данной библиотеке можно использовать разные варианты. Это может быть геолокация, аудио или видео файл, изображение, или что-то приближенное к нашему – article. Это очень близкое к простому текстовому сообщению.  
Этот метод получает входной сигнал о выборе элемента от первого пользователя и его готовности к “сражению” и выводит сообщение “I am ready to fight!”.

В этом коде так же происходит одна хитрая махинация, которую я опишу ниже.  
Так как в нашей программе мы не используем базы данных, но информацию о выборе первого игрока нам нужно где-то хранить - мы записываем информацию прямо в кнопку. Но этот выбор в чате не будет видно, так как кнопка будет перекрыта надписью (например “Processing…”), а позже эта кнопка будет быстро удаляться, так что игроки не успеют увидеть, какой выбор сделал противник.



Рис. 10 – Программный код добавления элементов для выбора.

На рисунке 10 мы добавляем элементы, которые впоследствии будут использованы игроками, и на рисунке 11 можно увидеть результат проделанной работы.

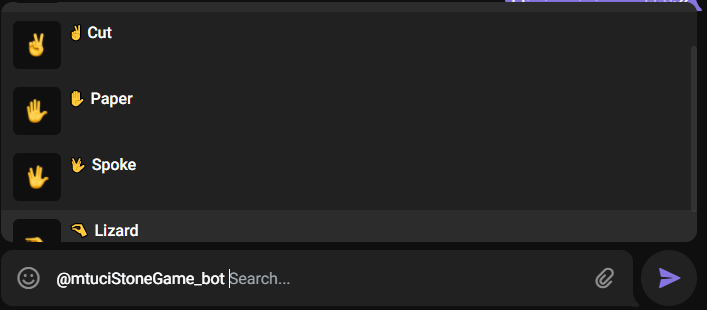


Рис. 11 – Варианты элементов для выбора игроками (первым игроком).



Рис. 12 – Добавление дополнительных условий для обработки входящих данных.

Для того, чтобы бот не здоровался с нами после каждого сообщения, и отправлял варианты выбора второму игроку только тогда, когда первый игрок выбрал элемент игры – мы добавляем несколько условий:  
- Сообщение не равно нулю  
- Сообщение отправлено через бота  
- Сообщение отправлено с помощью бота mtuciStoneGame\_bot.

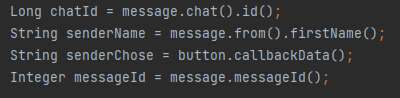


Рис. 13 – Программный код заданных направлений отправки данных.

На рисунке 13 прописан код, который позволяет боту определить куда отправлять ответ, и чтобы вывести кто выиграл или проиграл.

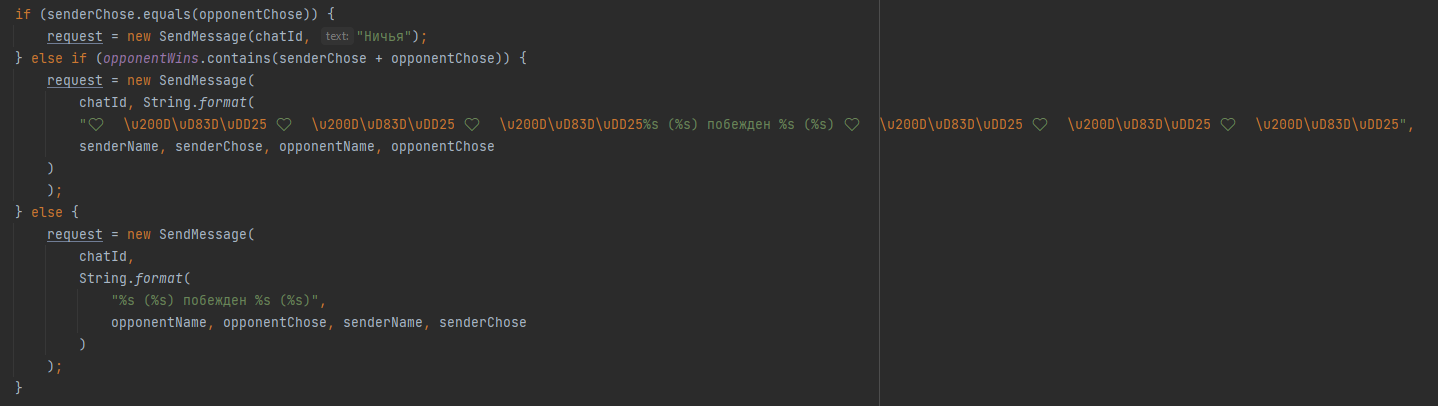


Рис. 14 – Сравнение выборов обоих игроков и вывод результата.

На рисунке 14 представлен программный код условий проверки результатов выбора двух игроков. В первом случае, если принимаемые значения совпадают программа выводит результат “Ничья”.

Если первый игрок победил, то соответствующий результат, если второй, то соответствующий этому условию.

Ниже я приведу примеры нескольких игр для наглядности:

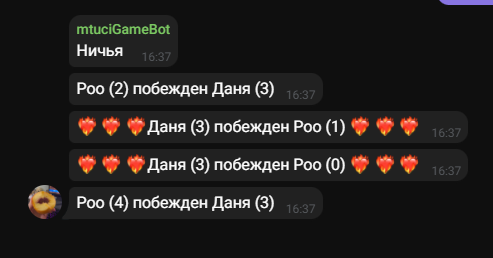


Рис. 15 – Примеры вывода сообщений при разных вариантах выбора игровых элементов.

Основной принцип работы бота заключается в том, что бот проверяет все возможные комбинации при которых выигрывает второй игрок (поэтому opponentWins), если полученные условия не попадают ни под один из критериев, значит выиграл первый игрок или ничья. Проверку “ничьи” я описал выше.

На рисунке 16 представлены все комбинации, при которых выигрывает первый игрок (по ним бот выполняет проверку).

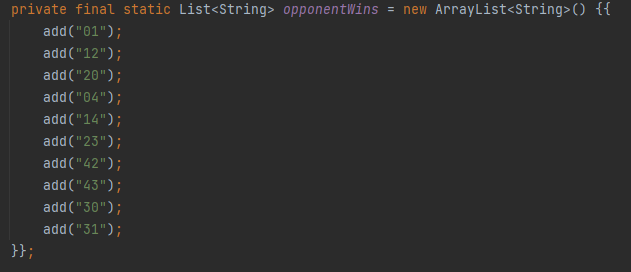


Рис. 16 – Выигрышные комбинации для второго игрока.

Примечание: первая цифра означает id элемента, который выбрал второй игрок, вторая цифра – элемент, который выбрал первый игрок.  
Все id элементов можно увидеть на рисунке 10.

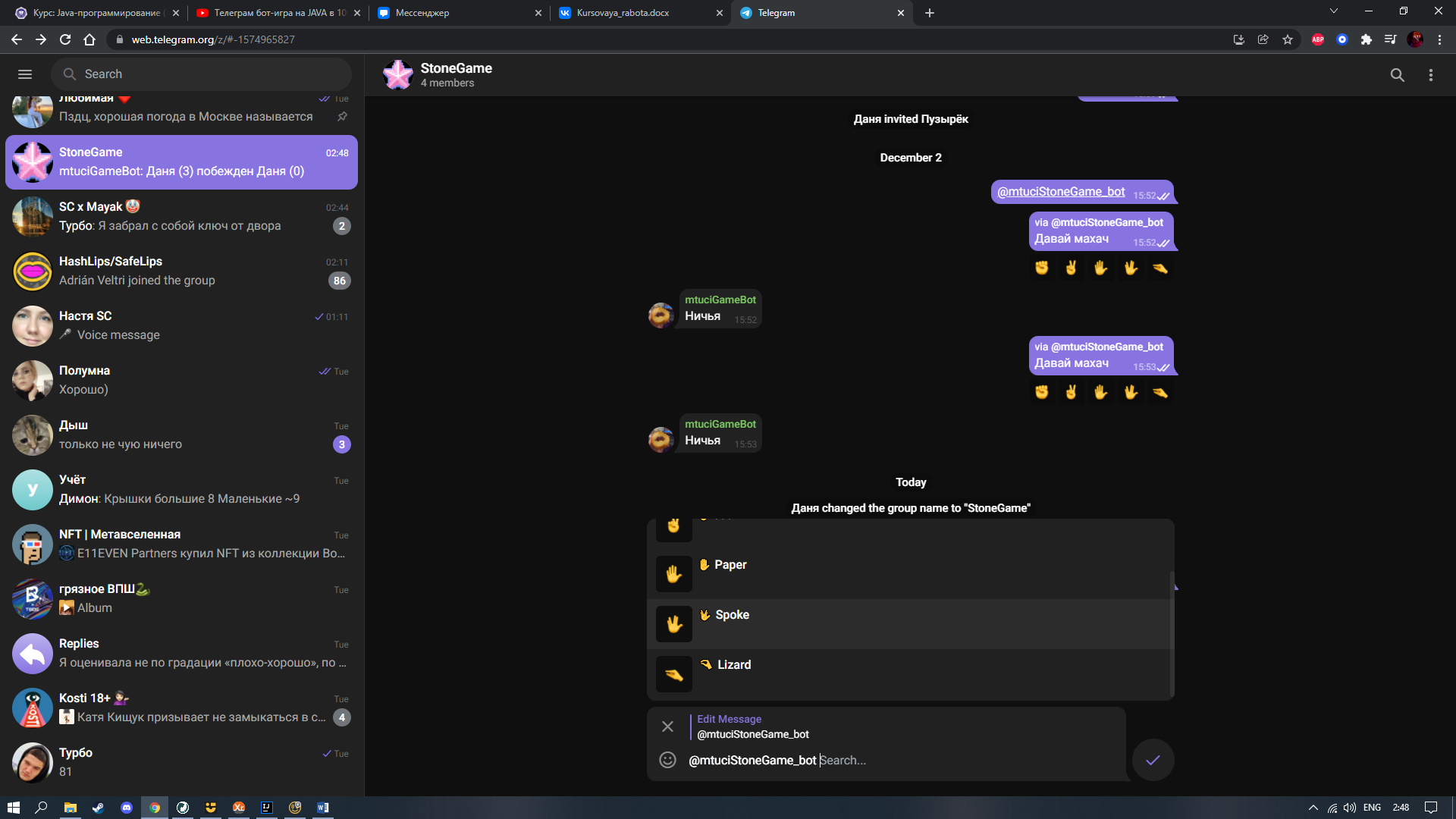


Рис. 17 – Выбор элемента первым игроком.

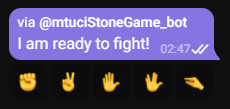


Рис. 18 – Выбор элемента вторым игроком.

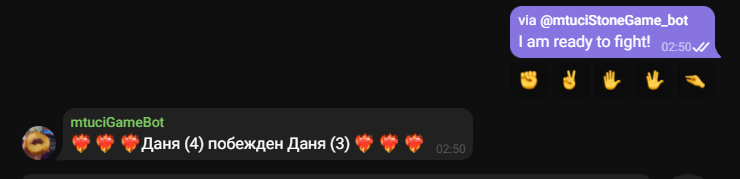


Рис. 19 – Вывод результата.

На в данном случае первым игроком я выбрал Ящерицу – это подтверждает цифра 4 рядом с первым игроком. А вторым игроком я выбрал Спока ( id = 3). Как мы видим из рисунка 1 – ящерица побеждает Спока, значит программа работает правильно. Приведу ещё один пример, в котором выберу два одинаковых результата в качестве первого и второго игрока, это будет камень.

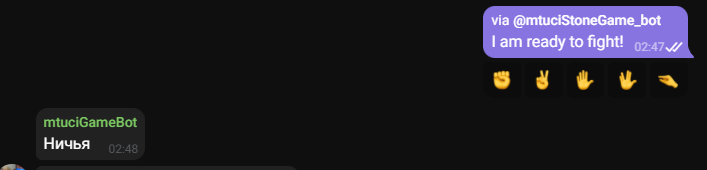


Рис. 20 – Вывод программы при ничьей.

**Заключение**

В результате выполнения данной курсовой работы я разработал Телеграм бота, который позволяет играть в усовершенствованную версию игры Камень-Ножницы-Бумага, используя классы и методы, а также библиотеки telegram.api на языке программирования Java, и на основе работы продемонстрировал свои знания об основах программирования на Java.