

**Министерство образования и науки РФ
Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа программной инженерии**

КУРСОВАЯ РАБОТА

Алиса Fallout сервер

по дисциплине «Технологии разработки качественного
программного обеспечения»

Выполнил
студент гр. в3530904/70321

<подпись>

А.В. Ершова

Руководитель

<подпись>

Н.Г. Смирнов

«___»_____2020г.

Санкт-Петербург

2020

Содержание

| | |
|--|----|
| Обзор предметной области..... | 3 |
| Сравнение похожих по функционалу программ..... | 4 |
| Скриншоты работы | 5 |
| UML class диаграммы | 6 |
| Покрытие тестами | 8 |
| Модульное тестирование | 8 |
| Системное тестирование | 9 |
| Выводы | 11 |
| Список используемой литературы | 12 |
| Приложение 1. Исходный код сервера | 13 |
| Приложение 2. Исходный код клиента..... | 17 |
| Приложение 3. Junit тесты..... | 20 |

Обзор предметной области

В связи с активным развитием интернета и возможностей программного обеспечения в целом, у людей, нуждающихся в общении, появилась возможность общаться с виртуальными собеседниками (или чат-ботами).

Виртуальный собеседник представляет из себя программу, которая на определенные действия пользователя дает голосовой или текстовый ответ, либо выполняет необходимый скрипт. Такое приложение можно использовать как для восполнения недостатка общения, так и для решения более сложных задач, вроде установки будильника, планирования, совершений онлайн-заказов и покупок, поиска информации в интернете, – т.е. программа представляет из себя полноценного помощника.

Виртуальные ассистенты могут как иметь искусственный интеллект и быть способными к машинному обучению, так и быть просто закриптованными на заранее заданную последовательность действий. В любом случае, выбор виртуального помощника остается за пользователем, исходя из его потребностей.

Сравнение похожих по функционалу программ

Так как было необходимо реализовать сервер, который работает по протоколу «AlisaFalloutServer», т.е. выдает ответы в стиле ветки диалогов игры «Fallout 4», которая на старте не работала должным образом, нельзя не упомянуть голосовой помощник «Алиса».

«Алиса» имеет свою нейросеть, благодаря которой может вести осмысленный диалог с пользователем. Она может определять адрес, работать с системными приложениями (будильник, музыка и др.). У нее есть система навыков – т.е. ее можно обучать новым возможностям.

В основном она заточена на работу с сервисами Яндекс, но в нее можно интегрировать и другие.

Однако сходство «Алисы» с реализованной программой ограничивается имитацией общения с конечным пользователем и возможностью одновременного диалога с несколькими пользователями, т.к. ее функционал достаточно обширный.

Скриншоты работы

Приложение работает по принципу клиент-сервер и имитирует общение с пользователем, беря фрагмент его сообщения и задавая вопрос с этим фрагментом в ответ. Например, на фразу «Какой сегодня день недели?» сервер может ответить: «Какой?», «сегодня?», «день?», «недели?».

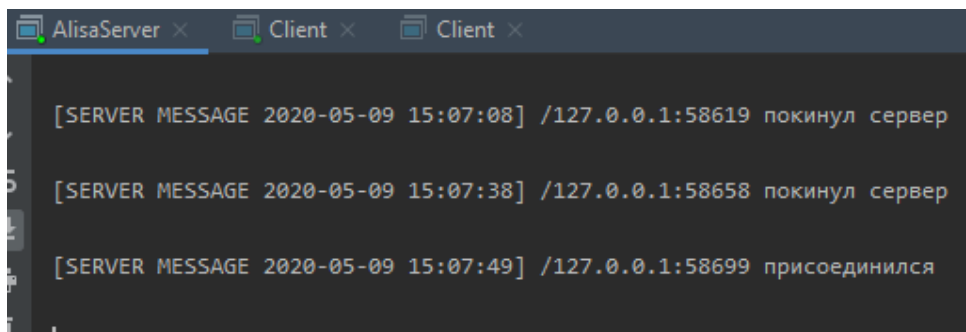


Рисунок 1. Уведомление сервера о подключениях.

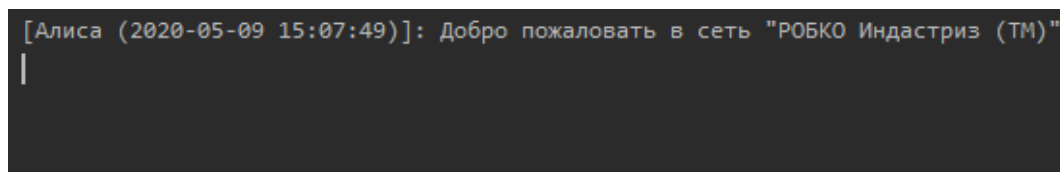


Рисунок 2. Приветствие пользователя.

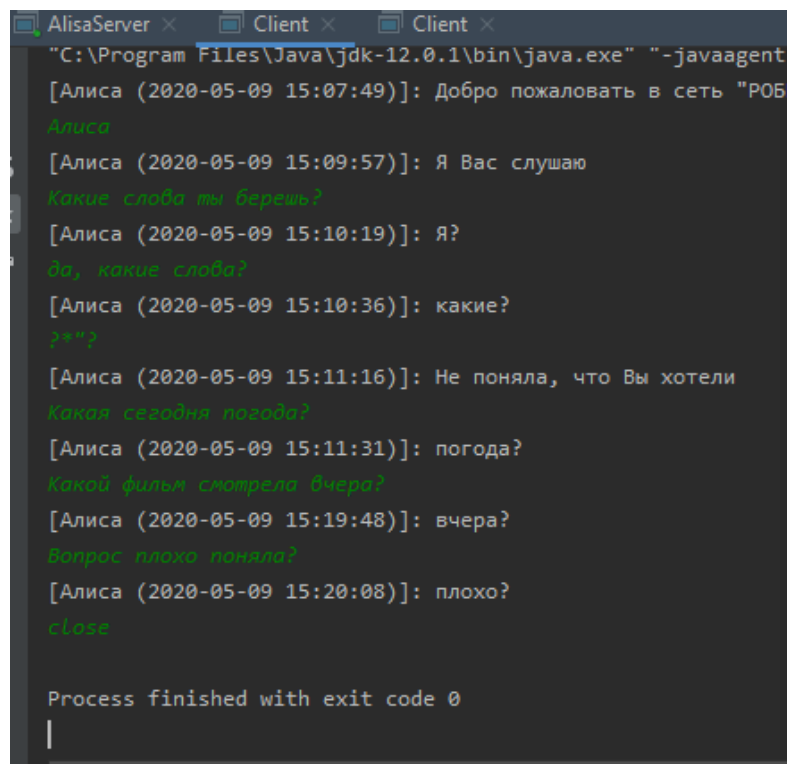


Рисунок 3. Общение с сервером и окончание диалога.

UML class диаграммы

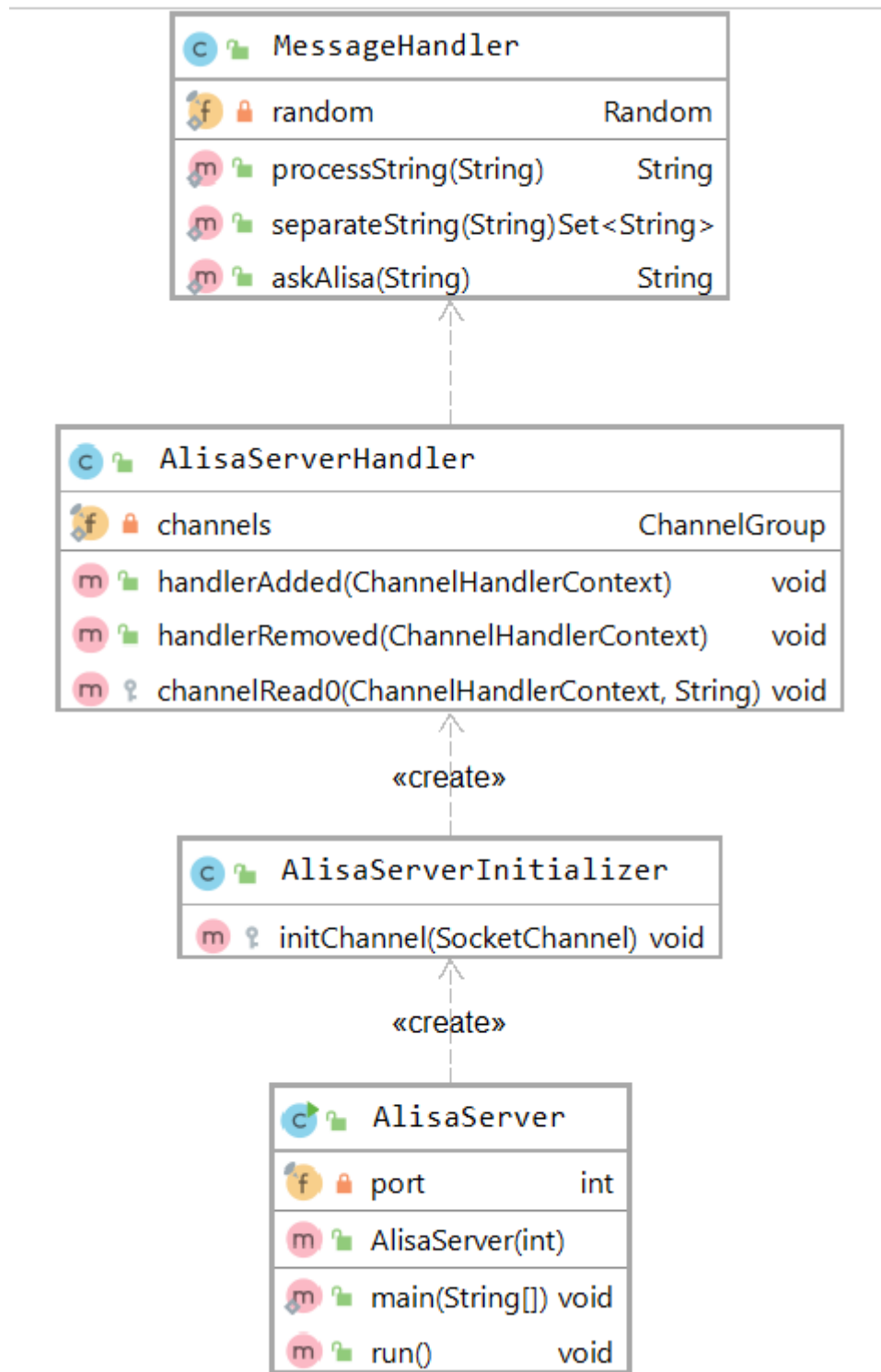


Рисунок 4. UML class диаграмм Сервера.

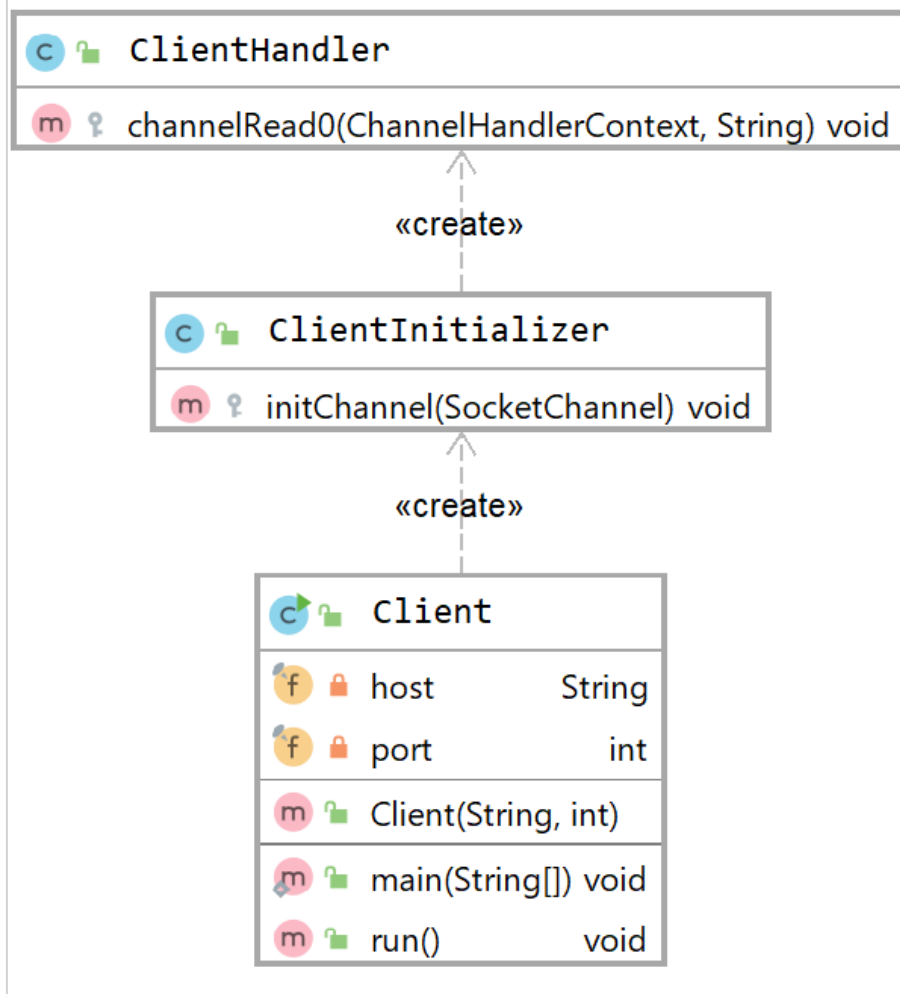


Рисунок 5. UML class диаграмма Клиента.

Покрытие тестами

Целью тестирования является проверка работоспособности отдельных модулей и программы в целом, поэтому было проведено модульное и системное тестирование. Нефункциональное тестирование не проводилось.

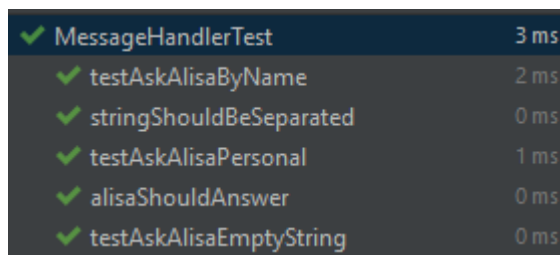
Таблица 1. Параметры тестового окружения.

| | |
|-------------------|--|
| Сервер | Характеристики |
| Сервер приложений | Intel(R) Core(TM) i7-7700K CPU @ 4.20GHz, RAM16GB 3000mhz, Win10 |

Модульное тестирование

Таблица 2. Тестирование обработчика сообщений.

| Тест-кейс | Пример строки | Результат |
|--|-----------------------|-----------|
| Строка должна разделиться на отдельные слова без символов. | Как ?:*? дела? | passed |
| В качестве ответа должно быть выбрано одно из слов входящего сообщения + «?». | какая сегодня погода? | passed |
| На обращение по имени должна возвращаться строка «Я Вас слушаю». | Алиса | passed |
| Если после обработки строки нет слов, должна возвращаться строка «Не поняла, что Вы хотели». | ? :;? | passed |
| После местоимения 2 лица должна возвращаться строка «Я?» | Вы | passed |



| | |
|---------------------------|------|
| ✓ MessageHandlerTest | 3 ms |
| ✓ testAskAlisaByName | 2 ms |
| ✓ stringShouldBeSeparated | 0 ms |
| ✓ testAskAlisaPersonal | 1 ms |
| ✓ alisaShouldAnswer | 0 ms |
| ✓ testAskAlisaEmptyString | 0 ms |

Рисунок 6. Тесты обработчика сообщений прошли.

Таблица 3. Тестирование сервера Алиса.

| Тест-кейс | Результат |
|----------------------------|-----------|
| Сервер должен запуститься. | passed |

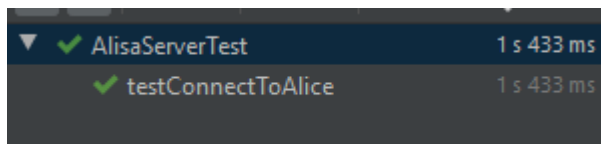


Рисунок 7. Тест сервера прошел.

Таблица 4. Тестирование клиента.

| Тест-кейс | Результат |
|---------------------------------------|-----------|
| Клиент должен подключиться к серверу. | passed |

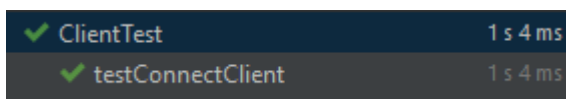


Рисунок 8. Тест клиента прошел.

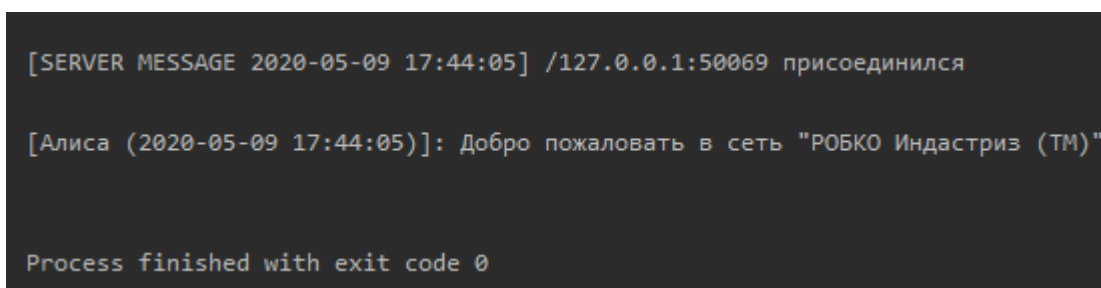
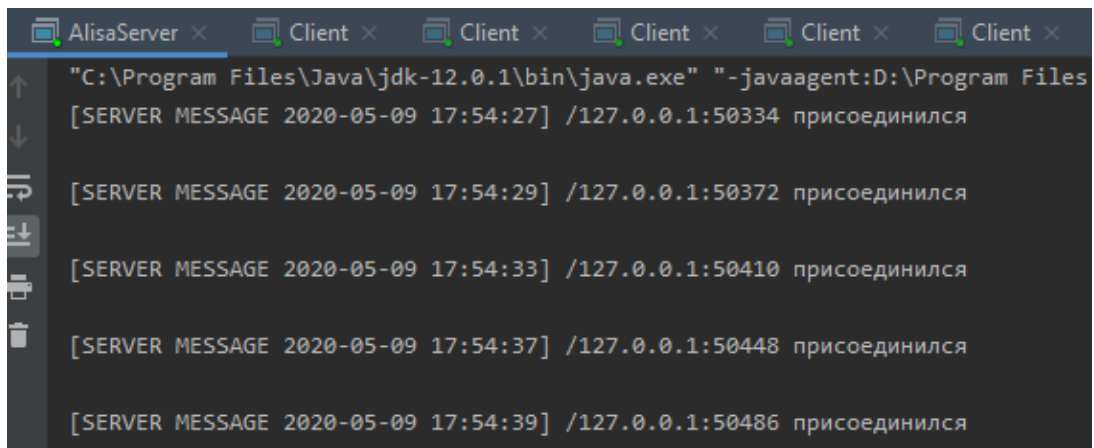


Рисунок 9. Вывод в консоль при тестировании.

Системное тестирование

Было проведено ручное системное тестирование. При критических разрывах соединений клиентов сервер продолжает работать. Сервер отвечает всем подключенным клиентам, если их больше одного.



```
AlisaServer x Client x Client x Client x Client x Client x
"C:\Program Files\Java\jdk-12.0.1\bin\java.exe" "-javaagent:D:\Program Files
[SERVER MESSAGE 2020-05-09 17:54:27] /127.0.0.1:50334 присоединился

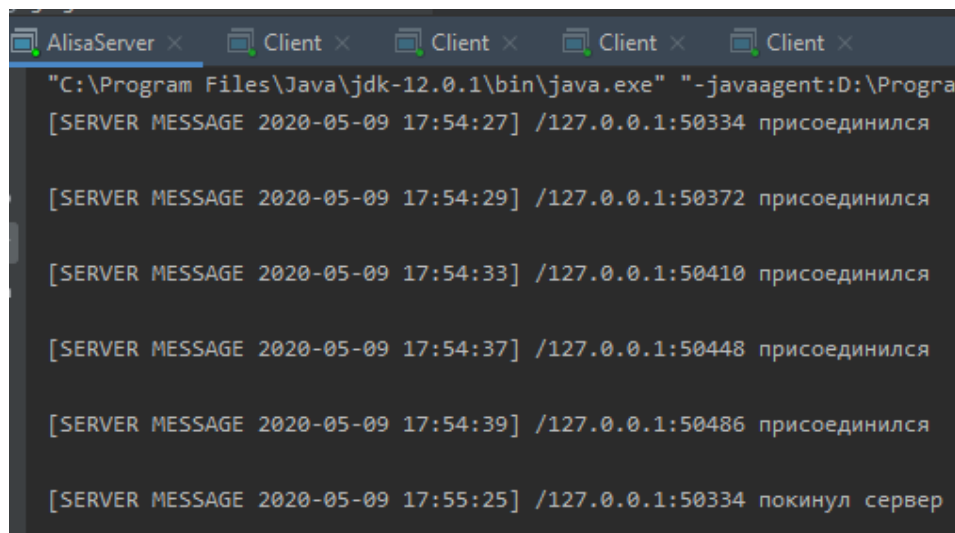
[SERVER MESSAGE 2020-05-09 17:54:29] /127.0.0.1:50372 присоединился

[SERVER MESSAGE 2020-05-09 17:54:33] /127.0.0.1:50410 присоединился

[SERVER MESSAGE 2020-05-09 17:54:37] /127.0.0.1:50448 присоединился

[SERVER MESSAGE 2020-05-09 17:54:39] /127.0.0.1:50486 присоединился
```

Рисунок 10. 5 клиентов подключилось.



```
AlisaServer x Client x Client x Client x Client x
"C:\Program Files\Java\jdk-12.0.1\bin\java.exe" "-javaagent:D:\Program
[SERVER MESSAGE 2020-05-09 17:54:27] /127.0.0.1:50334 присоединился

[SERVER MESSAGE 2020-05-09 17:54:29] /127.0.0.1:50372 присоединился

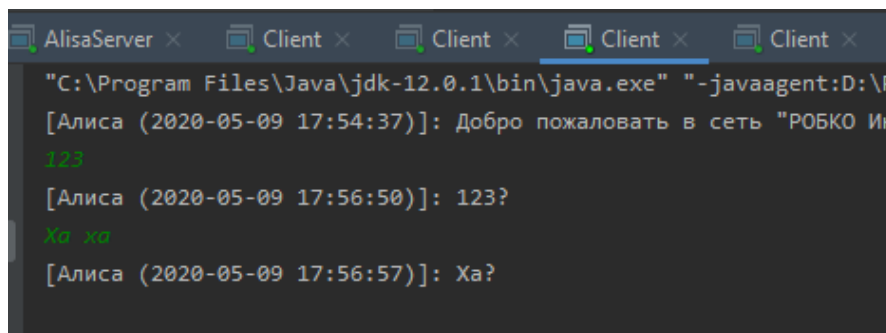
[SERVER MESSAGE 2020-05-09 17:54:33] /127.0.0.1:50410 присоединился

[SERVER MESSAGE 2020-05-09 17:54:37] /127.0.0.1:50448 присоединился

[SERVER MESSAGE 2020-05-09 17:54:39] /127.0.0.1:50486 присоединился

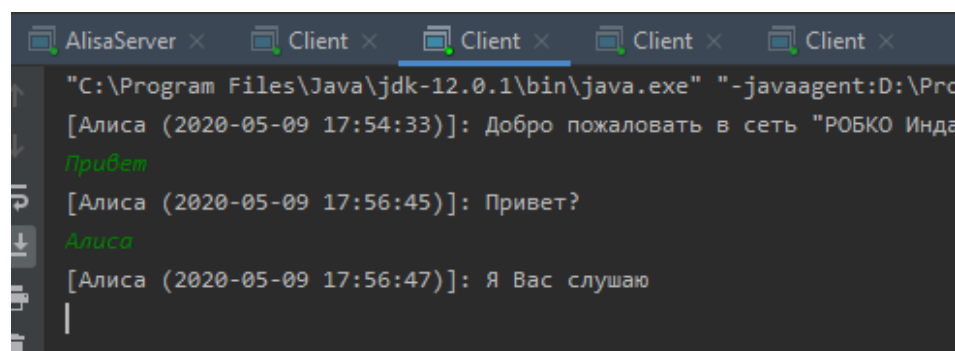
[SERVER MESSAGE 2020-05-09 17:55:25] /127.0.0.1:50334 покинул сервер
```

Рисунок 11. У одного клиента случился disconnect.



```
AlisaServer x Client x Client x Client x Client x
"C:\Program Files\Java\jdk-12.0.1\bin\java.exe" "-javaagent:D:\P
[Алиса (2020-05-09 17:54:37)]: Добро пожаловать в сеть "РОБКО Ин
123
[Алиса (2020-05-09 17:56:50)]: 123?
Ха ха
[Алиса (2020-05-09 17:56:57)]: Ха?
```

Рисунок 12. Алиса отвечает клиенту 3.



```
AlisaServer x Client x Client x Client x Client x
"C:\Program Files\Java\jdk-12.0.1\bin\java.exe" "-javaagent:D:\Pro
[Алиса (2020-05-09 17:54:33)]: Добро пожаловать в сеть "РОБКО Инда
Привет
[Алиса (2020-05-09 17:56:45)]: Привет?
Алиса
[Алиса (2020-05-09 17:56:47)]: Я Вас слушаю
|
```

Рисунок 13. Алиса отвечает клиенту 4.

Выводы

В ходе работы была реализована клиентская и серверная части приложения, в котором сервер работает по протоколу «AlisaFalloutServer», а клиент может посылать текстовые сообщения в консоль серверу и получает на них зашифрованный ответ, созданный из посланных им слов.

Приложение создано с использованием библиотеки Netty.io v 4.1.49. Тестирование проведено с использованием библиотеки Junit v 4.13.

Можно было реализовать поиск по ассоциативному словарю, загрузив предварительно словарь в HashMap парой «Key(слово) : Value (перечень его ассоциаций)» и при совпадении одного из слов пользователя, находящихся в HashSet, с ключом HashMap, разбивать Value на слова, и уже из этих слов выбирать ответ (возможно, опираясь на частоту встречаемости в ассоциациях, указанную после слова). Но я не смогла в одиночку придумать красивую и оптимальную по памяти и ООП реализацию. А лучше в финальный продукт добавлять разработки, в которых уверен (хотя бы более-менее).

Ввиду неопытности плохо реализован unit тест самого подключения клиента и сервера, поэтому есть неточность в тестировании.

В целом, приложение выполняет поставленную задачу.

Список используемой литературы

- 1) Документация Netty.io <https://netty.io/4.1/api/index.html>
- 2) Яндекс Справка <https://yandex.ru/support/alice/>

Приложение 1. Исходный код сервера

```
package ru.ershova.lab34.server;

import io.netty.bootstrap.ServerBootstrap;
import io.netty.channel.EventLoopGroup;
import io.netty.channel.nio.NioEventLoopGroup;
import io.netty.channel.socket.nio.NioServerSocketChannel;

/**
 * @author Anna
 *
 * прослушивает входящие соединения и передает их для обработки
 */
public class AlisaServer {

    public static void main(String[] args) {
        new AlisaServer(8000).run();
    }

    private final int port;

    public AlisaServer(int port) {
        this.port = port;
    }

    public void run() {
        // nIO группа boss принимает входящие соединения, когда они прибывают, и
        // передает рабочей группе на обработку
        EventLoopGroup bossGroup = new NioEventLoopGroup();
        EventLoopGroup workerGroup = new NioEventLoopGroup();

        // аналогично клиенту, обработку входящих соединений определяет
        ServerBootstrap
            try {
                ServerBootstrap bootstrap = new ServerBootstrap()
                    .group(bossGroup, workerGroup)
                    .channel(NioServerSocketChannel.class) // IO сокет для связи
                    .childHandler(new AlisaServerInitializer()); // инициализация
                // обработки входящих сообщений

                // слушать порт входящих соединений
                bootstrap.bind(port).sync().channel().closeFuture().sync();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            finally { // очистить ELG
                bossGroup.shutdownGracefully();
                workerGroup.shutdownGracefully();
            }
    }
}
```

```
package ru.ershova.lab34.server;

import io.netty.channel.ChannelInitializer;
import io.netty.channel.ChannelPipeline;
import io.netty.channel.socket.SocketChannel;
```

```

import io.netty.handler.codec.DelimiterBasedFrameDecoder;
import io.netty.handler.codec.Delimiters;
import io.netty.handler.codec.string.StringDecoder;
import io.netty.handler.codec.string.StringEncoder;

/**
 * @author Anna
 *
 * инициализация канала сокета сервера
 */
public class AlisaServerInitializer extends ChannelInitializer<SocketChannel> {

    @Override
    protected void initChannel(SocketChannel socketChannel) throws Exception {
        ChannelPipeline pipeline = socketChannel.pipeline();

        // ожидать кадры не более 8192, каждый из которых ограничен окончанием строки
        pipeline.addLast("framer", new DelimiterBasedFrameDecoder(8192,
Delimiters.lineDelimiter()));
        // байты в строки
        pipeline.addLast("decoder", new StringDecoder());
        // строки в байты
        pipeline.addLast("encoder", new StringEncoder());

        // определить класс, который обрабатывает все декодированные входящие строки
Клиента
        pipeline.addLast("handler", new AlisaServerHandler());
    }
}

```

```

package ru.ershova.lab34.server;

import io.netty.channel.Channel;
import io.netty.channel.ChannelHandlerContext;
import io.netty.channel.SimpleChannelInboundHandler;
import io.netty.channel.group.ChannelGroup;
import io.netty.channel.group.DefaultChannelGroup;
import io.netty.util.concurrent.GlobalEventExecutor;

import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;

import static ru.ershova.lab34.server.MessageHandler.processString;

/**
 * @author Anna
 *
 * обрабатывает входящие подключения к серверу
 * возвращает клиенту ответ в стиле Fallout 4
 */
public class AlisaServerHandler extends SimpleChannelInboundHandler<String> {
    // список клиентов
    private static final ChannelGroup channels = new
DefaultChannelGroup(GlobalEventExecutor.INSTANCE);

    // при новом подключении к серверу
    @Override
    public void handlerAdded(ChannelHandlerContext ctx) throws Exception {
        Channel incoming = ctx.channel();
    }
}

```

```

        System.out.println("[SERVER MESSAGE " +
LocalDateTime.now().format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"))
        + "]" + incoming.remoteAddress() + " присоединился\n");

        incoming.writeAndFlush("[Алиса (" +
LocalDateTime.now().format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"))
        + ")] : Добро пожаловать в сеть \"РОБКО Индастриз (ТМ)\" \n\n");

        // добавить в список клиентов
        channels.add(ctx.channel());
    }

    // при отключении клиента
    @Override
    public void handlerRemoved(ChannelHandlerContext ctx) throws Exception {
        Channel incoming = ctx.channel();

        System.out.println("[SERVER MESSAGE " +
LocalDateTime.now().format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"))
        + "]" + incoming.remoteAddress() + " покинул сервер\n");

        // убрать из списка клиентов
        channels.remove(ctx.channel());
    }

    @Override
    protected void channelRead0(ChannelHandlerContext channelHandlerContext, String
message) throws Exception {
        // отправитель сообщения
        Channel incoming = channelHandlerContext.channel();

        // обработка сообщения
        String answer = processString(message);

        incoming.writeAndFlush("[Алиса (" +
LocalDateTime.now().format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"))
        + ")] : " + answer + "\n\n");
    }
}

```

```

package ru.ershova.lab34.server;

import java.util.*;

/**
 * @author Anna
 *
 * обработка сообщений - разделение, выбор ответа
 */
public class MessageHandler {

    private final static Random random = new Random();

    public static String processString(String message) {
        String answer = "";
        Set<String> stringSet = separateString(message);

        if (stringSet.size() > 0) {
            int count = 0;
            int index = random.nextInt(stringSet.size());

```

```

        for (String string: stringSet) {
            if (count == index){
                answer = string;
                break;
            }
            count++;
        }
    }

    answer = askAlisa(answer);

    return answer;
}

// разделить строку на слова
public static Set<String> separateString(String string) {
    Set<String> stringSet = new HashSet<>();

    // разделить по любым символам, кроме букв
    string = string.replaceAll("[^\\w\\u0430-\\u044f\\u0410-\\u042f]", " ");
    String[] parts = string.split(" ");

    // убрать повторы
    Collections.addAll(stringSet, parts);

    return stringSet;
}

public static String askAlisa(String answer) {

    if (answer.equalsIgnoreCase("Алиса")) {
        answer = "Я Вас слушаю";
    } else if (answer.equals("") || answer.equals(" ")) {
        answer = "Не поняла, что Вы хотели";
    } else if (answer.equalsIgnoreCase("Ты") || answer.equalsIgnoreCase("Вы")) {
        answer = "Я?";
    } else answer += "?";

    return answer;
}
}

```


Приложение 2. Исходный код клиента

```
package ru.ershova.lab34.client;

import io.netty.bootstrap.Bootstrap;
import io.netty.channel.Channel;
import io.netty.channel.EventLoopGroup;
import io.netty.channel.nio.NioEventLoopGroup;
import io.netty.channel.socket.nio.NioSocketChannel;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

/**
 * @author Anna
 *
 * подключается к серверу
 */
public class Client {

    public static void main(String[] args) {
        new Client("localhost", 8000).run();
    }

    private final String host;
    private final int port;

    public Client(String host, int port) {
        this.host = host;
        this.port = port;
    }

    public void run() {
        EventLoopGroup group = new NioEventLoopGroup();

        try {
            Bootstrap bootstrap = new Bootstrap()
                .group(group)
                .channel(NioSocketChannel.class) // nIO сокет ввода вывода
                .handler(new ClientInitializer()); // инициализация обработки
сообщений

            Channel channel = bootstrap.connect(host, port).sync().channel(); //
создать соединение с сервером
            BufferedReader in = new BufferedReader(new
InputStreamReader((System.in))); // захват консоли ввода

            // разрыв соединения
            boolean breakTheConnection = false;
            String message;

            while (!breakTheConnection) { // принятие ввода и запись на сервер
                message = in.readLine();
                channel.writeAndFlush(message + "\r\n");
                if (message.equals("close")) {
                    breakTheConnection = true;
                }
            }
        }
    }
}
```

```

    } catch (IOException | InterruptedException e) {
        e.printStackTrace();
    } finally { // после выхода из цикла отключить ELG
        group.shutdownGracefully();
    }
}
}

```

```

package ru.ershova.lab34.client;

import io.netty.channel.ChannelInitializer;
import io.netty.channel.ChannelPipeline;
import io.netty.channel.socket.SocketChannel;
import io.netty.handler.codec.DelimiterBasedFrameDecoder;
import io.netty.handler.codec.Delimiters;
import io.netty.handler.codec.string.StringDecoder;
import io.netty.handler.codec.string.StringEncoder;

/**
 * @author Anna
 *
 * инициализация канала сокетa клиента
 */
public class ClientInitializer extends ChannelInitializer<SocketChannel> {

    @Override
    protected void initChannel(SocketChannel socketChannel) throws Exception {
        ChannelPipeline pipeline = socketChannel.pipeline();

        // ожидать кадры не более 8192, каждый из которых ограничен окончанием строки
        pipeline.addLast("framer", new DelimiterBasedFrameDecoder(8192,
Delimiters.LineDelimiter()));
        // байты в строки
        pipeline.addLast("decoder", new StringDecoder());
        // строки в байты
        pipeline.addLast("encoder", new StringEncoder());

        // определить класс, который обрабатывает все декодированные входящие строки
Сервера
        pipeline.addLast("handler", new ClientHandler());
    }
}

```

```

package ru.ershova.lab34.client;

import io.netty.channel.ChannelHandlerContext;
import io.netty.channel.SimpleChannelInboundHandler;

/**
 * @author Anna
 *
 * обрабатывает входящие объекты - строки
 */
public class ClientHandler extends SimpleChannelInboundHandler<String> {

    @Override
    protected void channelRead0(ChannelHandlerContext channelHandlerContext, String
s) {
        // печать полученного с сервера сообщения
    }
}

```

```
        System.out.println(s);  
    }  
}
```

Приложение 3. Junit тесты

```
import org.junit.Assert;
import org.junit.Test;
import ru.ershova.lab34.server.MessageHandler;

import java.util.HashSet;
import java.util.Set;

/**
 * @author Anna
 */
public class MessageHandlerTest {

    @Test
    public void stringShouldBeSeparated() {
        Set<String> expectedSet = new HashSet<>();
        expectedSet.add("Как");
        expectedSet.add("дела");
        Assert.assertEquals(expectedSet, MessageHandler.separateString("Как дела?"));
    }

    @Test
    public void alisaShouldAnswer() {
        String message = MessageHandler.processString("какая сегодня погода?");
        switch (message) {
            case "какая?": Assert.assertEquals("какая?", message); break;
            case "сегодня?": Assert.assertEquals("сегодня?", message); break;
            case "погода?": Assert.assertEquals("погода?", message); break;
        }
    }

    @Test
    public void testAskAlisaByName() {
        Assert.assertEquals("Я Вас слушаю", MessageHandler.askAlisa("Алиса"));
    }

    @Test
    public void testAskAlisaEmptyString() {
        Assert.assertEquals("Не поняла, что Вы хотели", MessageHandler.askAlisa(""));
    }

    @Test
    public void testAskAlisaPersonal() {
        Assert.assertEquals("Я?", MessageHandler.askAlisa("ты"));
    }
}
```

```
import org.junit.BeforeClass;
import org.junit.Test;
import ru.ershova.lab34.client.Client;
import ru.ershova.lab34.server.AlisaServer;

/**
 * @author Anna
 */
public class ClientTest {
```

```

private static String host = "localhost";
private static int port = 8000;

@BeforeClass
public static void initServer() {
    Thread thread = new Thread(() -> {
        new AlisaServer(port).run();
    });
    thread.start();
}

@Test
public void testConnectClient() throws InterruptedException {
    Thread thread2 = new Thread(() -> {
        new Client(host, port).run();
    });
    thread2.start();

    Thread.sleep(1000);
}
}

```

```

import io.netty.bootstrap.Bootstrap;
import io.netty.channel.EventLoopGroup;
import io.netty.channel.nio.NioEventLoopGroup;
import io.netty.channel.socket.nio.NioSocketChannel;
import org.junit.BeforeClass;
import org.junit.Test;
import ru.ershova.lab34.client.ClientInitializer;
import ru.ershova.lab34.server.AlisaServer;

/**
 * @author Anna
 */
public class AlisaServerTest {

    private static String host = "localhost";
    private static int port = 8000;

    @BeforeClass
    public static void initServer() {
        Thread thread = new Thread(() -> {
            new AlisaServer(port).run();
        });
        thread.start();
    }

    @Test
    public void testConnectToAlice() throws InterruptedException {
        EventLoopGroup group = new NioEventLoopGroup();

        try {
            Bootstrap bootstrap = new Bootstrap()
                .group(group)
                .channel(NioSocketChannel.class)
                .handler(new ClientInitializer());
        }
    }
}

```

```
        bootstrap.connect(host, port).sync().channel();
    } catch (InterruptedException e) {
        e.printStackTrace();
    } finally {
        group.shutdownGracefully();
    }
}
```