|  |
| --- |
|  |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования  **"МИРЭА - Российский технологический университет"**  **РТУ МИРЭА** |
| Институт информационных технологий (ИТ) |
| Кафедра инструментального и прикладного программного обеспечения |

|  |  |
| --- | --- |
| **ОТЧЕТ**  **ПО ПРАКТИЧЕСКОЙ РАБОТЕ №3** | |
| **по дисциплине** |  |
| **«Архитектура клиент-серверных приложений»**  **Тема: «Многопоточные приложения, как основараспределённых клиент-серверных вычислений»** | |
| Выполнил студент группы: ИКБО-20-21 | Хитров Н.С. |
| Принял преподаватель кафедры ИиППО | Волков М.Ю. |

|  |  |  |
| --- | --- | --- |
| Практические работы выполнены | « » 2023 г. |  |
| «Зачтено» | « » 2023 г. |  |

Москва

2023

ОГЛАВЛЕНИЕ

[Цель работы 3](#_Toc146909903)

[Теоретическое введение 3](#_Toc146909904)

[Постановка задачи 4](#_Toc146909905)

[Программный код 6](#_Toc146909906)

[Результаты работы 14](#_Toc146909907)

[Вывод 15](#_Toc146909908)

# Цель работы

Знакомство студентов с одной из базовых технологии реализации конечной точки для передачи и приема данных по сети - сокетом.

# Теоретическое введение

Клиент-серверная архитектура - это самая известная структура приложений в Интернете. В этой архитектуре клиенты (например, персональные компьютеры, устройства IoT и т. Д.) вначале запрашивают ресурсы с сервера, а затем сервер отправляет соответствующие ответы на запросы клиентов. Чтобы это произошло, как на клиентах, так и на серверах должен быть реализован какой-то механизм, поддерживающий эту сетевую транзакцию. Этот механизм называется коммуникацией через сокеты.

Стоит отметить, что существует два типа сокетов для TCP и UDP. Поскольку большинство сетевых приложений используют TCP, в тексте работы речь пойдёт только о TCP-сокетах и их реализации. Основное различие между ними заключается в том, что UDP не имеет соединения, то есть между клиентом и сервером нет сеанса, в то время как TCP ориентирован на соединение, то есть сначала должно быть установлено эксклюзивное соединение между клиентом и сервером для связи.

Сокет работает по типичной модели запрос / ответ, где в java-программе, называемой клиентом, вызывается другая программа, называемая сервером, работающая на другой JVM. Клиент должен отправить запрос, и сервер отвечает ответом. В этой модели обмен всегда инициируется клиентом; сервер не может отправлять какие-либо данные без предварительного запроса клиента. Стоит отметить, что данная модель хорошо работала во всемирной паутине, когда клиенты время от времени запрашивали документы, которые нечасто менялись, но ограничения этого подхода становятся все более актуальными, поскольку контент меняется быстро и пользователи ожидают более интерактивного взаимодействия в сети. Например, WebSocket устраняет эти ограничения, предоставляя полнодуплексный канал связи между клиентом и сервером. Протокол WebSocket (JSR 356) будет рассмотрен в следующей практической работе. Но на практике, сокеты активно используются при реализации клент-серверного взаимодействия.

Сокет - это одна из конечных точек двустороннего канала связи между двумя программами, работающими в сети. Сокет привязан к номеру порта, чтобы уровень TCP мог идентифицировать приложение, данные предназначенные для отправки.

Каждый сервер - это программа, которая работает в определенной системе и прослушивает определенный порт. Сокеты привязаны к номерам портов, и когда мы запускаем любой сервер, он просто слушает сокет и ждет клиентских запросов. Например, сервер tomcat, работающий на порту 8080, ожидает клиентских запросов и, получив любой клиентский запрос, отвечает на них.

Следовательно, для запуска любого сокета необходим запускаемый порт. Очевидно, что он размещен на машине, идентифицируемой именем хоста и уникальной адресованной IP.

В Java классы для поддержки программирования сокетов упакованы в пакет java.net. В соответствии с Java, java.net пакет поставляется с двумя классами Socket и ServerSocket для функций клиента и сервера соответственно.

# Постановка задачи

Используя материалы данной практической работы Необходимо создать клиент-серверное приложение на языке JAVA с использованием socket, для широковещательного общения пользователей. Приложение может быть как консольным, так и оснащённым полноценным GUI. Клиентское приложение считывает данные из стандартного ввода и отсылает сообщение серверу (с помощью TCP/IP). Сервер, в свою очередь, накапливает сообщения и раз в 5 секунд осуществляет массовую рассылку всем клиентам. Если сообщений за указанный период не поступило, то рассылка не производится. Клиент, получивший сообщение, отображает на экране текст данного сообщения. Структуру и поведение данного клиент-серверного приложения, в том числе, например, в части регистрации конкретного клиента и формата широковещательного сообщения, студент определяет самостоятельно.

# Программный код

На данном рисунке (рис. 1) показан конструктор серверной части приложения. Также показан переопредленный метод run(), в котором мы принимаем имя пользователя, и считываем сообщения, пока не будет введено ключевое слово «stop»

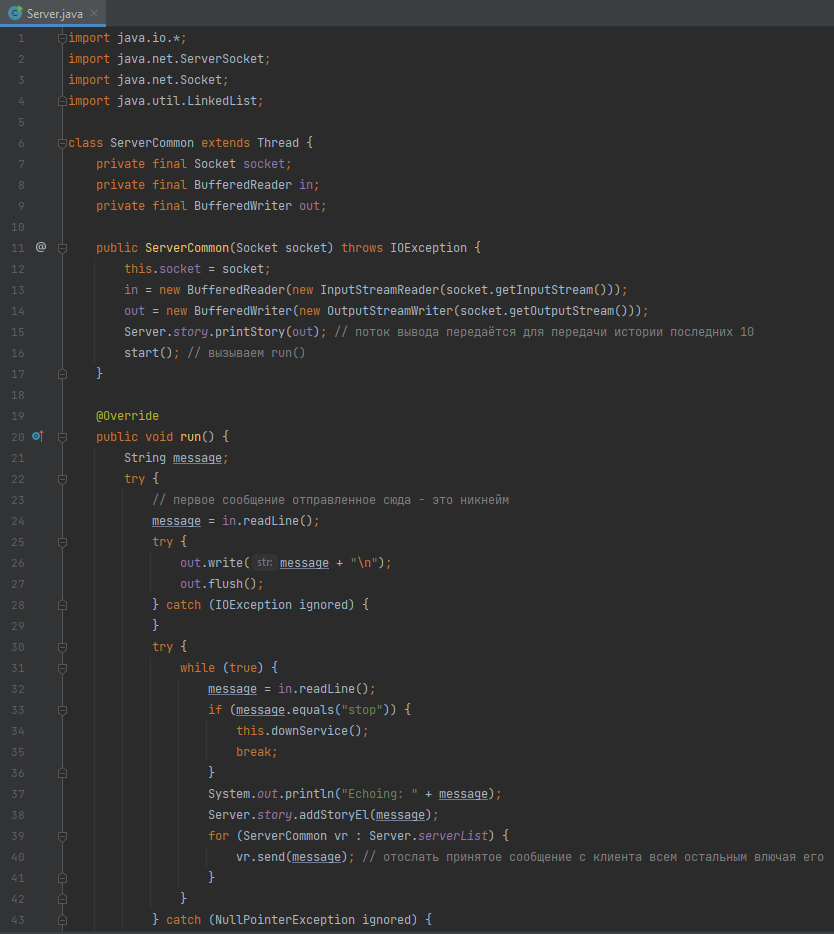


Рисунок 1 – Код сервеной части (часть 1)

На данном рисунке (рис. 2) представлены сервисные методы, которые отвечают за отправку сообщений, а тажке за остановку таких вспомогательных элементов, как сокет, входной и выходной потоки.

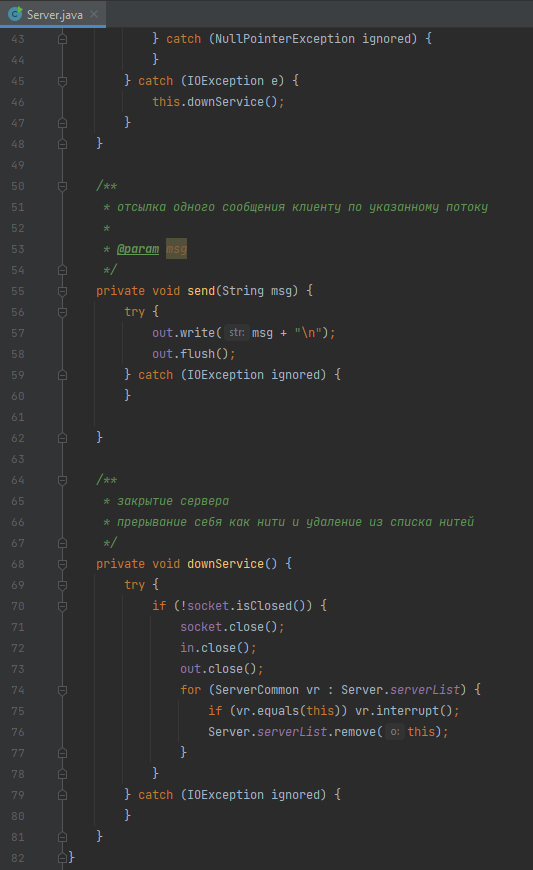


Рисунок 2 – Код сервеной части (часть 2)

На данном рисунке (рис. 3) показан класс Story, который отвечает за сохранение и отправку истории последних сообщений (количество определяется самостоятельно)



Рисунок 3 – Код сервеной части (часть 3)

На данном рисунке (рис. 4) показан класс Server, в котором определяются такие параметры как порт, список клиентов и история сообщений. Также тут представлен главный метод серверного класса программы.



Рисунок 4 – Код сервеной части (часть 4)

На данном рисунке (рис. 5) представлен код клиентского класса. В данном фрагменте кода показан конструктор общей части клиентского класса.

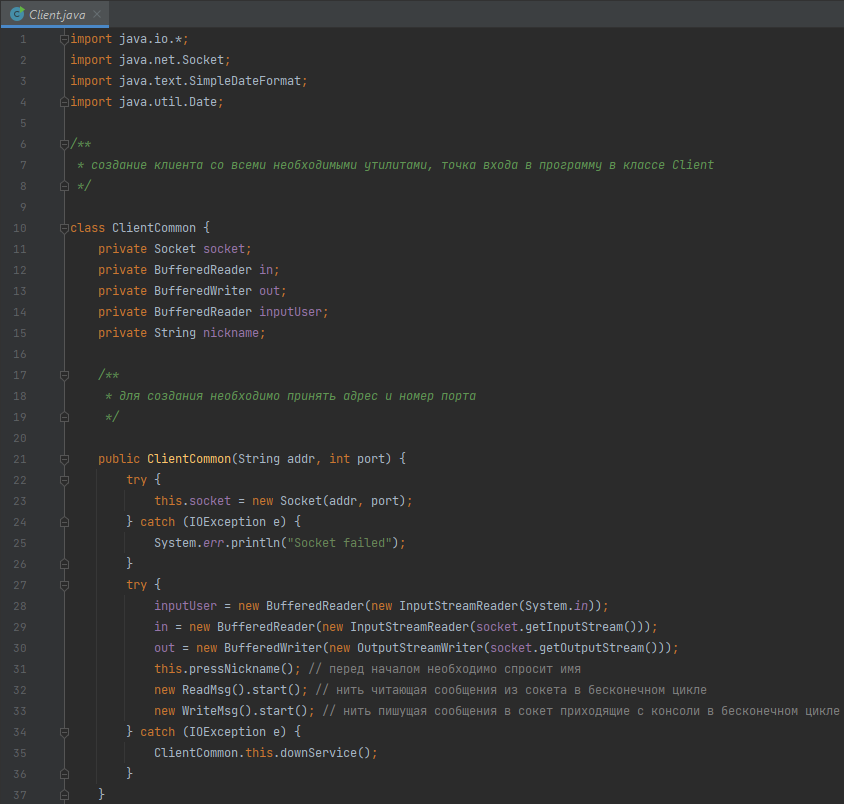


Рисунок 5 – Код клиентской части (часть 1)

На данном рисунке (рис. 6) представлен код сервисных методов для ввода имени пользователя и остановки таких вспомогательных элементов, как сокет, входной и выходной потоки.



Рисунок 6 – Код клиентской части (часть 2)

На данном рисунке (рис. 7) отображены классы для постоянного считывания сообщений в поток ввода и их отправки в поток вывода.



Рисунок 7 – Код клиентской части (часть 3)

На данном рисунке (рис. 8) представлен код клиентского класса, который содержит реализацию главного метода, создающего нового клиента.

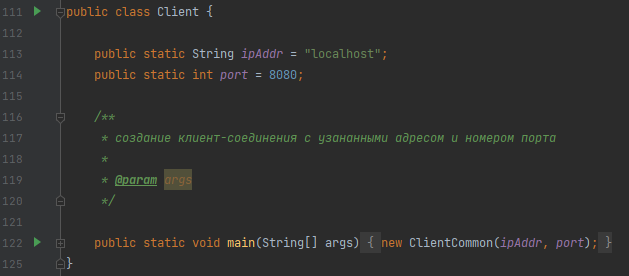


Рисунок 8 – Код клиентской части (часть 4)

# Результаты работы

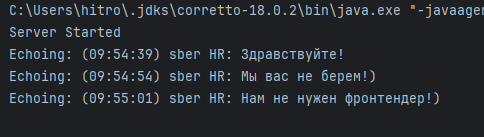
****

Рисунок 9 – Работа сервера

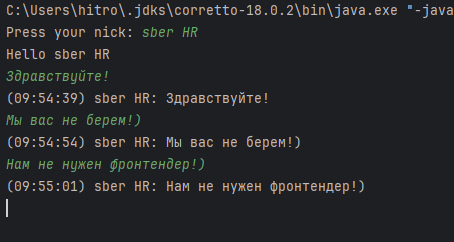


Рисунок 10 – Работа клиента 1

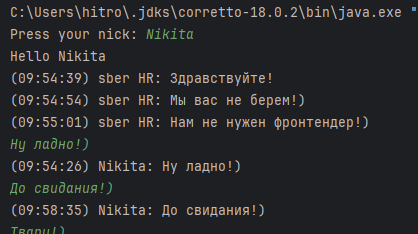


Рисунок 11 – Работа клиента 2

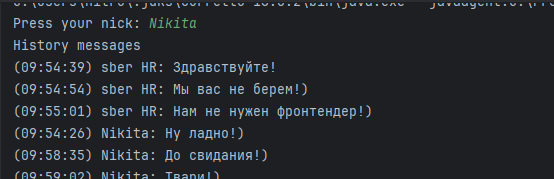


Рисунок 12 – Вывод истории сообщений

# Вывод

В рамках практической работы были получены навыки работы с одной из базовых технологий реализации конечной точки для передачи и приема данных по сети в языке Java – сокетом.

**Список использованных источников**

1. Справочная информация JavaRush – URL: https://javarush.com/groups/posts/654-klassih-socket-i-serversocket-ili-allo-server-tih-menja-slihshishjh (дата обращения: 20.09.2023) – текст: электронный.
2. Статья Baeldung – URL: https://www.baeldung.com/a-guide-to-java-sockets (дата обращения: 20.09.2023) – текст: электронный.