Кравченко Даниил Андреевич

студент

ФГБОУ ВО «Государственный университет морского

и речного флота имени адмирала С.О. Макарова»

г. Санкт-Петербург

РАЗРАБОТКА ПРИЛОЖЕНИЯ МНОГОПОЛЬЗОВАТЕЛЬСКОГО ЧАТА НА JAVA

Аннотация: *статья посвящена разработке многопользовательского чата на Java. Приведены и описаны использованные инструменты разработки, библиотеки и классы. Рассмотрена архитектура «клиент-сервер», показана и использована технология многопоточности.*

Ключевые слова*: чат, многопоточность, клиент, сервер, Java, разработка приложения.*

Введение. В данной статье рассмотрен принцип работы и разработки многопоточного клиент-серверного чата с графическим интерфейсом на языке программирования Java.

В настоящее время Интернет – это универсальная среда для общения, развлечений и обучения. Для многих людей общение через Сеть стало неотъемлемой частью жизни. На данный момент в мире существует огромное множество способов и средств общения, и большая часть из них использует в своей работе глобальные компьютерные сети.

Интернет, кроме полноценного источника разнообразной полезной информации, является основной формой виртуального общения. Связь с близкими и родными людьми, контакт с партнерами по работе, новые знакомства – все это является важным компонентом повседневной жизни человека, причем выбор наиболее удобных способов онлайн-общения у современного пользователя достаточно велик, например, онлайн-чат.

Инструменты разработки

Разработка приложения велась на объектно-ориентированном языке программирования Java. Приложение основано на архитектуре «клиент-сервер», характеризующейся наличием по крайней мере двух взаимодействующих, самостоятельных процесса – клиента и сервера (рис. 1). Помимо этого, использовалась IntelliJ IDEA – интегрированная среда разработки программного обеспечения.

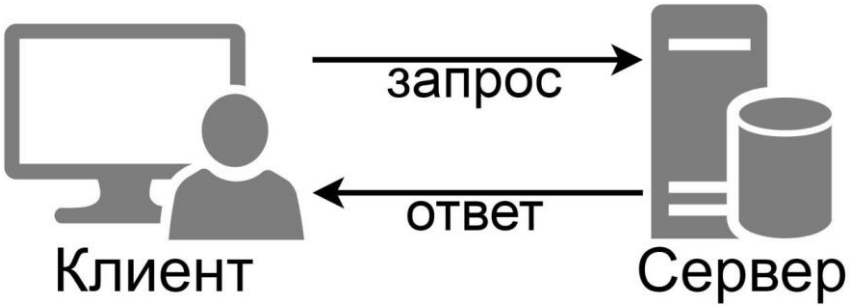

Рис. 1. Архитектура клиент-сервер

Использованные библиотеки и принципы их работы

Современным приложениям необходим графический интерфейс пользователя (GUI). Сложно представить пользователя готового работать с приложением через консоль, намного удобнее использовать GUI, при помощи визуальных компонентов, к которым относятся кнопки, текстовые поля, выпадающие списки и т. д. Для решения поставленной задачи используется библиотеки AWT и SWING, предоставляющие полный набор инструментов для создания и использования GUI.

Java для работы в сети имеет специальную библиотеку NET, содержащую классы по работе с сокетами – «гнездо» программного интерфейса для обеспечения обмена данными между процессами. Ключевыми классами являются:

– Java.net.ServerSocket – класс реализует серверный сокет, который ожидает запросы, приходящие от клиентов по сети, и может отправлять ответ;

– Java.net.Socket – класс реализует клиентский сокет.

В библиотеке IO содержатся классы для действий, базирующихся на потоках байт (такие классы как InputStream и OutputStream) и потоках символов

(Reader и Writer). Сюда входит работа с сетью, обмен данными между потоками исполнения (рис. 2), долговременное сохранение объектов.

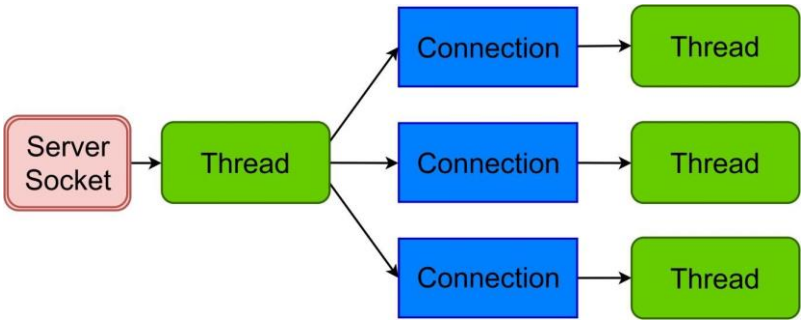

Рис. 2. Структура IO

Также использовалась библиотека UTIL с широким диапазоном возможностей и функций. В данном приложении применялись классы java.util.Observer и java.util.Observable, позволяющие отдельным компонентам реагировать на события, происходящие в других компонентах.

Работа приложения и реализация многопоточности

Разработанное приложение работает везде, где можно установить Java Runtime Environment 8. Сервер и клиенты могут работать на разных компьютерах в одной сети, как пример в локальной сети (LAN).

Благодаря многопоточности к серверу возможно подключение нескольких клиентов, где они могут общаться друг с другом и каждый может видеть сообщения других пользователей. После подключения к серверу пользователь должен написать свое имя для начала общения. Каждый пользователь уведомляется, когда кто-то присоединяется или отключается. Каждое сообщение имеет префикс с именем пользователя, чтобы было видно, кто оставил сообщение.

Сервер работает в так называемом «вечном» цикле. Как только подключается новый клиент, он создает для работы с ним новый поток. Таким образом все соединения обслуживаются параллельно и не мешают друг другу (рис. 3). Реализовано это при помощи специального класса Thread.

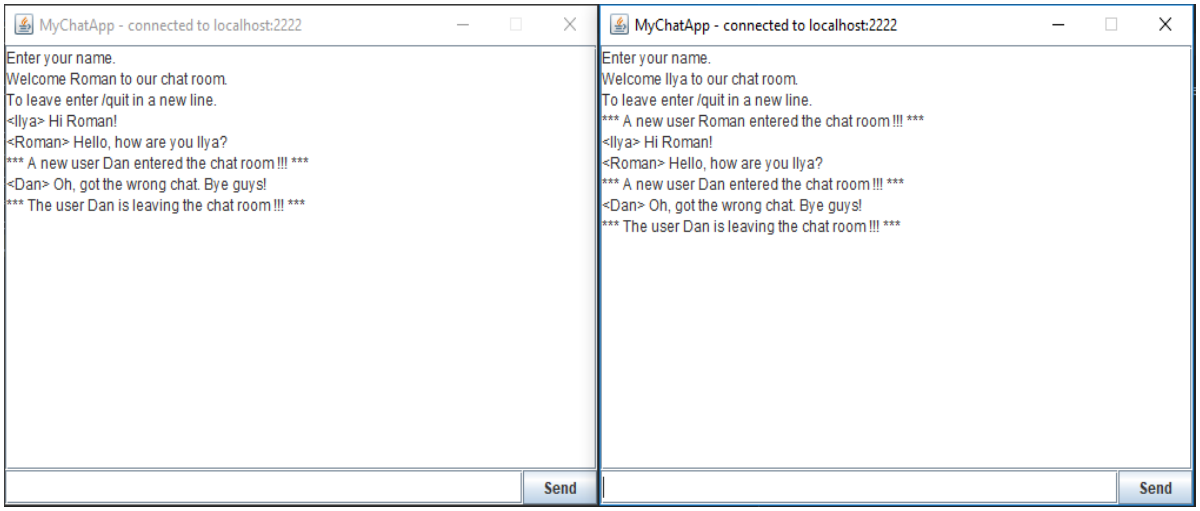

Рис. 3. Работа приложения, демонстрация многопоточности

Заключение. В этой статье была разобрана клиент-серверная архитектура, на ее основе разработано приложение многопользовательского чата с графическим интерфейсом при применении полезных функций различных библиотек и классов. Подведя итог можно сделать вывод, что многопоточность полезна и необходима для серверов любой сложности.

Исходный код приложения размещен в свободном доступе на сайте github.com (https://github.com/Beranill/Multi-User_Chat_on_Java).

Список литературы

1. Брюс Э. Философия Java. – Питер, 2009. – С. 17–47.

2. Документация Java Standard Edition [Электронный ресурс]. – Режим доступа: https://docs.oracle.com/javase/8/docs/

3. Нимейер П. Программирование на Java. Исчерпывающее руководство для профессионалов / П. Нимейер, Д. Леук. – 2014. – С. 640–704.

4. Анатольев А.Г. Компоненты сетевого приложения. Клиент-серверное взаимодействие и роли серверов [Электронный ресурс]. – Режим доступа: http://www.4stud.info/networking/lecture5.html

5. Программирование сокетов на Java [Электронный ресурс]. – Режим доступа: http://www.quizful.net/post/java-socket-programming