Написание сетевого чата. Часть II

Разработка интерфейса для клиентской части. Авторизация. Механизмы взаимодействия клиента и сервера

# 

[Серверная часть](#_3znysh7)

[Клиентская часть](#_2et92p0)

[Домашнее задание](#_tyjcwt)

[Дополнительные материалы](#_1t3h5sf)

# 

# 

# Серверная часть

*\* При рассмотрении серверной и клиентской части не будут рассматриваться моменты, описанные в методичках 6-7.*

В класс MyServer добавилось два метода sendMsgToClient() и broadcastClientsList(). Метод sendMsgToClient() отсылает сообщение от клиента *from* клиенту с указанным ником(*nickTo*), если пользователя-получателя нет в списке клиентов, отправителю сообщается об этом. Метод broadcastClientsList() формирует список участников чата в виде строки «/clients nick1 nick2 nick3...» и рассылает его всем клиентам. При вызове методов subscribe() и unsubscribe(), то есть изменении списка клиентов на сервере производится авторассылка нового списка клиентов всем пользователям.

|  |
| --- |
| public class MyServer {  ...  public synchronized void sendMsgToClient(ClientHandler from, String nickTo, String msg) {  for (ClientHandler o : clients) {  if (o.getName().equals(nickTo)) {  o.sendMsg("от " + from.getName() + ": " + msg);  from.sendMsg("клиенту " + nickTo + ": " + msg);  return;  }  }  from.sendMsg("Участника с ником " + nickTo + " нет в чат-комнате");  }  public synchronized void broadcastClientList() {  StringBuilder sb = new StringBuilder("/clients ");  for (ClientHandler o : clients) {  sb.append(o.getName() + " ");  }  String msg = sb.toString();  broadcastMsg(msg);  }  public synchronized void unsubscribe(ClientHandler o) {  clients.remove(o);  broadcastClientsList();  }  public synchronized void subscribe(ClientHandler o) {  clients.add(o);  broadcastClientsList();  } } |

Цикл обработки сообщений в классе ClientHandler также претерпел некоторые изменения. Как только от клиента приходит сообщение, производится проверка на наличие служебных команд, начинающихся с символа /, если такой символ стоит на первом месте, обрабатываем пришедшую команду, если нет - делаем рассылку сообщения всем участникам чата.

В качестве служебной команды добавлена возможность отсылки личных сообщений через шаблон «/w имя\_получателя сообщение», если сервер получает такое сообщение, извлекает имя\_получателя и текстовое сообщение, после чего через метод сервера sendMsgToClient() отсылает его.

|  |
| --- |
| while (true) {  String str = in.readUTF();  if (str.startsWith("/")) {  if (str.equals("/end")) break;  if (str.startsWith("/w ")) {  String[] tokens = str.split("\\s");  String nick = tokens[1];  String msg = str.substring(4 + nick.length());  myServer.sendMsgToClient(this, nick, msg);  }  } else {  myServer.broadcastMsg(name + ": " + str);  } } |

# Клиентская часть

Клиентская часть изменилась незначительно. Обработка сообщений перенесена в метод start(), то есть подключение к серверу, создание объектов типа DataInputStream и DataOutputStream, запуск потока чтения сообщений с сервера. При попытке нажать на кнопку авторизации производится проверка на подключение к серверу, если его ещё не было или произошел обрыв соединения, метод onAuthClick() автоматически запустит метод start() для подключения к серверу.

После авторизации от сервера приходит сообщение формата «/authok nick», т.е. клиент узнает, под каким ником авторизовался. Если по какой-то причине соединение с сервером обрывается, клиентская часть «обнуляет» авторизацию, сбрасывает ник и закрывает сокет.

|  |
| --- |
| public void start() {  try {  setAuthorized(false);  socket = new Socket("localhost", 8189);  in = new DataInputStream(socket.getInputStream());  out = new DataOutputStream(socket.getOutputStream());  Thread t = new Thread(() -> {  try {  while (true) {  String str = in.readUTF();  if (str.startsWith("/authok")) {  setAuthorized(true);  myNick = str.split("\\s")[1];  break;  }  textArea.appendText(str + "\n");  }  ...  } catch (IOException e) {  e.printStackTrace();  } finally {  try {  setAuthorized(false);  socket.close();  myNick = "";  } catch (IOException e) {  e.printStackTrace();  }  }  });  t.start();  } catch (IOException e) {  showAlert("Не удалось подключиться к серверу");  e.printStackTrace();  } }  public void onAuthClick() {  if (socket == null || socket.isClosed())  start();  try {  out.writeUTF("/auth " + loginField.getText() + " " + passField.getText());  loginField.setText("");  passField.setText("");  } catch (Exception e) {  e.printStackTrace();  } } |

# Домашнее задание

1. Разобраться с кодом.
2. Добавить отключение неавторизованных пользователей по таймауту (120 сек. ждём после подключения клиента, и если он не авторизовался за это время, закрываем соединение).

# Дополнительные материалы

1. Кей С. Хорстманн, Гари Корнелл Java. Библиотека профессионала. Том 1. Основы // Пер. с англ. - М.: Вильямс, 2014. - 864 с.
2. Брюс Эккель Философия Java // 4-е изд.: Пер. с англ. – СПб.: Питер, 2016. – 1168 с.
3. Г. Шилдт Java 8. Полное руководство // 9-е изд.: Пер. с англ. - М.: Вильямс, 2015. - 1376 с.
4. Г. Шилдт Java 8: Руководство для начинающих. // 6-е изд.: Пер. с англ. - М.: Вильямс, 2015. - 720 с.