Agenda

- Serwer
- Klient
- Pliki

Serwer

Gniazdo serwerowe

```
public class Server {
    private ServerSocket serverSocket;

    public Server() throws IOException {
        serverSocket = new ServerSocket(3000);
    }

    public void listen() throws IOException {
        System.out.println("Server started");
        serverSocket.accept();
        System.out.println("Client connected");
    }
}

public static void main(String[] args) throws IOException {
    Server server = new Server();
    server.listen();
}
```

Oczekiwanie na połączenie

```java
public void listen() throws IOException {
    System.out.println("Server started");
    serverSocket.accept();
    System.out.println("Client connected");
}
```

```
Server started
```

```
$ telnet localhost 3000
```

Nawiązanie połączenia

```java
public void listen() throws IOException {
    System.out.println("Server started");
    serverSocket.accept();
    System.out.println("Client connected");
}
```

```
Server started
Client connected
```

```
$ telnet localhost 3000
Trying ::1...
Connected to localhost.
Escape character is '^]'.
Connection closed by foreign host.
```

Odebranie wiadomości

```java
public void listen() throws IOException {
    System.out.println("Server started");
    Socket socket = serverSocket.accept();
    System.out.println("Client connected");

    InputStream input = socket.getInputStream();
    BufferedReader reader =
        new BufferedReader(
              new InputStreamReader(input)
        );

    String message;
    while(true) {
        message = reader.readLine();
        System.out.println(message);
    }
}
```

```
Server started
Client connected
```

```
$ telnet localhost 3000
message
```

Odebranie wiadomości

```java
public void listen() throws IOException {
    System.out.println("Server started");
    Socket socket = serverSocket.accept();
    System.out.println("Client connected");

    InputStream input = socket.getInputStream();
    BufferedReader reader =
        new BufferedReader(
              new InputStreamReader(input)
        );

    String message;
    while(true) {
        message = reader.readLine();
        System.out.println(message);
    }
}
```

```
Server started
Client connected
message
```

```
$ telnet localhost 3000
message ENTER
```

Odebranie wiadomości

```java
public void listen() throws IOException {
    System.out.println("Server started");
    Socket socket = serverSocket.accept();
    System.out.println("Client connected");

    InputStream input = socket.getInputStream();
    BufferedReader reader =
        new BufferedReader(
              new InputStreamReader(input)
        );

    String message;
    while(true) {
        message = reader.readLine();
        System.out.println(message);
    }
}
```

```
Server started
Client connected
message
message 2
```

```
$ telnet localhost 3000
message
message 2 ENTER
```

Zakończenie połączenia

```java
public void listen() throws IOException {
    /* ... */
    String message;
    while((message = reader.readLine()) != null)) {
        System.out.println(message);
    }

    socket.close();
    serverSocket.close();
    System.out.println("Server closed");
}
```

```
Server started
Client connected
message
Server closed
```

```
$ telnet localhost 3000
Trying ::1...
Connected to localhost.
Escape character is '^]'.
message
CTRL+]
telnet> CTRL+D
Connection closed by foreign host.
```

Wysłanie wiadomości

```java
public void listen() throws IOException {
    /* ... */

    OutputStream output =
                socket.getOutputStream();
    PrintWriter writer =
                new PrintWriter(output, true);

    String message;
    writer.println("Hello!");
    while ((message = reader.readLine()) != null)
        writer.println(message);


    /* ... */
}
```

```
Server started
Client connected
```

```
$ telnet localhost 3000
Hello!
message
```

Wysłanie wiadomości

```java
public void listen() throws IOException {
    /* ... */

    OutputStream output =
                 socket.getOutputStream();
    PrintWriter writer =
                 new PrintWriter(output, true);

    String message;
    writer.println("Hello!");
    while ((message = reader.readLine()) != null)
        writer.println(message);


    /* ... */
}
```

```
Server started
Client connected
```

```
$ telnet localhost 3000
Hello!
message [ENTER]
message
[CTRL+]] [CTRL+D]
Connection closed by foreign
host.
```

Wyodrębnienie obsługi klienta do metody

```
public void serveClient() throws IOException {
    Socket socket = serverSocket.accept();

    InputStream input = socket.getInputStream();
    BufferedReader reader = new BufferedReader(new InputStreamReader(input));
    OutputStream output = socket.getOutputStream();
    PrintWriter writer = new PrintWriter(output, true);

    String message;
    writer.println("Hello!");
    while((message = reader.readLine()) != null) {
        writer.println(message);
    }

    socket.close();
}
```

Sekwencyjna obsługa klientów

```java
public void listen() throws IOException {
        System.out.println("Server started");
        while(true) {
            System.out.println("Client connected");
            serveClient();
            System.out.println("Client disconnected");
        }
}
```

```
$ telnet localhost 3000
Hello!
message
message
```

```
Server started
Client connected
```

Próba połączenia drugiego klienta

```java
public void listen() throws IOException {
        System.out.println("Server started");
        while(true) {
            System.out.println("Client connected");
            serveClient();
            System.out.println("Client disconnected");
        }
}
```

```
$ telnet localhost 3000
Hello!
message
message
```

```
Server started
Client connected
```

```
$ telnet localhost 3000
```

Odblokowanie serwera

```java
public void listen() throws IOException {
        System.out.println("Server started");
        while(true) {
            System.out.println("Client connected");
            serveClient();
            System.out.println("Client disconnected");
        }
}
```

```
Server started
Client connected
Client disconnected
Client connected
```

```
$ telnet localhost 3000
Hello!
message
message
```

CTRL+] CTRL+D

```
Connection closed by foreign
host.
```

```
$ telnet localhost 3000
Hello!
```

Oczekiwanie na kolejne połączenie

```java
public void listen() throws IOException {
        System.out.println("Server started");
        while(true) {
            System.out.println("Client connected");
            serveClient();
            System.out.println("Client disconnected");
        }
}
```

```
Server started
Client connected
Client disconnected
Client connected
Client disconnected
Client connected
```

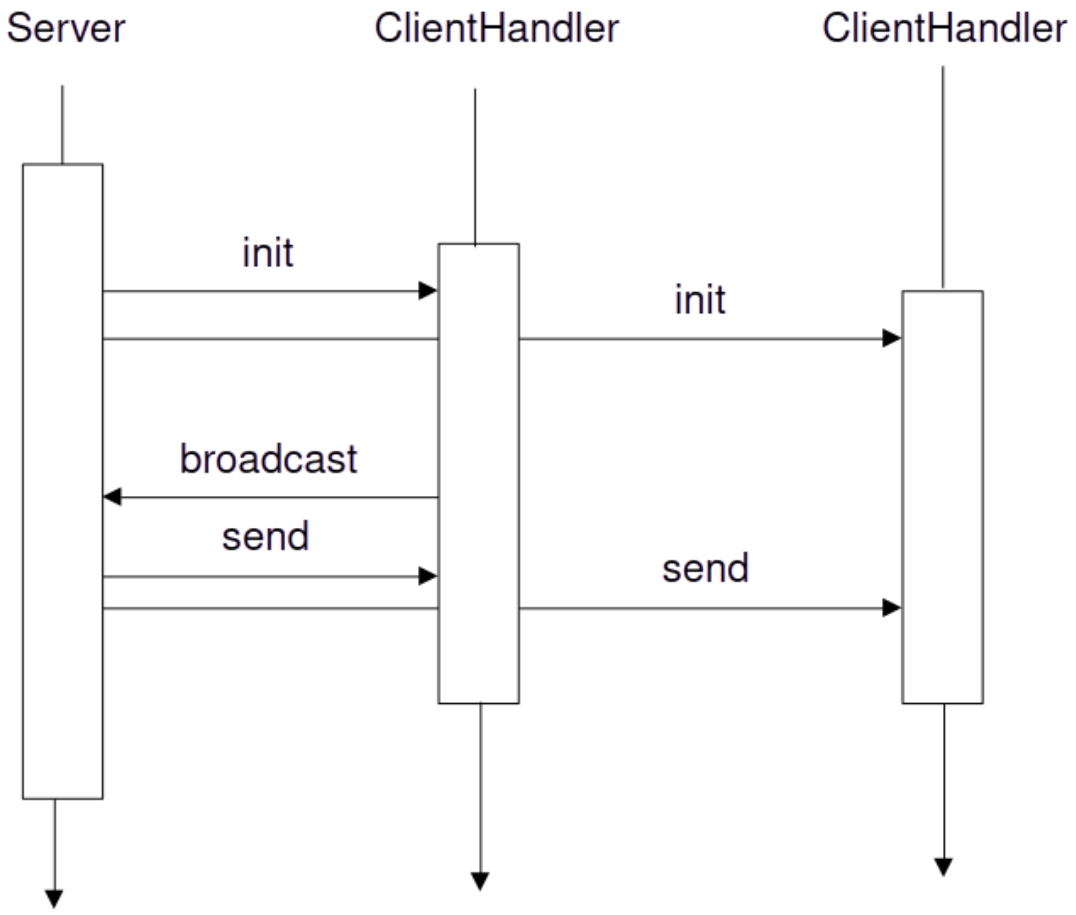
```
$ telnet localhost 3000
Hello!
message
message

Connection closed by foreign
host.
```

```
$ telnet localhost 3000
Hello!
```

CTRL+] CTRL+D

```
Connection closed by foreign
host.
```

Równoległa obsługa klientów



Wątek obsługujący klienta

```java
public class ClientHandler implements Runnable {
    private final Socket socket;
    private final BufferedReader reader;
    private final PrintWriter writer;

    public ClientHandler(Socket socket) throws IOException {
        this.socket = socket;
        InputStream input = socket.getInputStream();
        OutputStream output = socket.getOutputStream();
        reader = new BufferedReader(new InputStreamReader(input));
        writer = new PrintWriter(output, true);
    }

    @Override
    public void run() {
        /* ... */
    }
}
```

Metoda run()

```java
public class ClientHandler implements Runnable {
    /* ... */

    @Override
    public void run() {
        System.out.println("Client connected");
        String message;
        try {
            while ((message = reader.readLine()) != null)
                writer.println(message);
            socket.close();
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
        System.out.println("Client disconnected");
    }
}
```

Uruchomienie wątku

```
public class Server {
    private ServerSocket serverSocket;
    private ArrayList<ClientHandler> handlers = new ArrayList<>();

    /* ... */

    public void listen() throws IOException {
        System.out.println("Server started");
        while(true) {
            Socket socket = serverSocket.accept();
            ClientHandler handler = new ClientHandler(socket);
            Thread thread = new Thread(handler);
            thread.start();
            handlers.add(handler);
        }
    }
}
```

Równoległa obsługa klientów

```
Server started

Client connected

Client connected

Client disconnected
Client disconnected
```

```
$ telnet localhost 3000

Hello!
message
message
```

CTRL+] CTRL+D

```
$ telnet localhost 3000

Hello!
message
message
```

CTRL+] CTRL+D

Komunikacja między klientami

```java
public class ClientHandler implements Runnable {
    private final Server server;
    /* ... */
    public ClientHandler(Socket socket, Server server) throws IOException {/* ... */ }

    public void send(String message) {
        writer.println(message);
    }

    @Override
    public void run() {
        String message;
        try {
            while ((message = reader.readLine()) != null)
                server.broadcast(message);
            socket.close();
        } catch (IOException e) { throw new RuntimeException(e); }
    }
}
```

Rozsyłanie wiadomości wszystkim klientom

```
public class Server {
     /* ... */
    public void broadcast(String message) {
        handlers.forEach(handler -> handler.send(message));
    }

    public void listen() throws IOException {
        System.out.println("Server started");
        while(true) {
            Socket socket = serverSocket.accept();
            ClientHandler handler = new ClientHandler(socket, this);
            Thread thread = new Thread(handler);
            thread.start();
            handlers.add(handler);
        }
    }
}
```

Komunikacja między klientami

```
Server started

Client connected

Client connected
```

```
$ telnet localhost 3000

Hello!


message 1
message 1


message 2
```

```
$ telnet localhost 3000

Hello!

message 1

message 2
message 2
```

Usunięcie zakończonego wątku

```
public class ClientHandler implements Runnable {
    /* ... */
    private void close() throws IOException {
        socket.close(); server.removeHandler(this);
    }

    @Override
    public void run() {
        System.out.println("Client connected");
        String message;
        try {
            while ((message = reader.readLine()) != null)
                writer.println(message);
            close();
        } catch (IOException e) { throw new RuntimeException(e); }
        System.out.println("Client disconnected");
    }
}
```

Usunięcie zakończonego wątku

```
public class Server {
    private ServerSocket serverSocket;
    private ArrayList<ClientHandler> handlers = new ArrayList<>();

    /* ... */

    public void removeHandler(ClientHandler handler) {
        handlers.remove(handler);
    }

    public void listen() throws IOException {
        System.out.println("Server started");
        while(true) {
            Socket socket = serverSocket.accept();
            ClientHandler handler = new ClientHandler(socket, this);
            /* ... */
        }
    }
}
```

Zerwanie połączenia z serwerem

```
Server started

Client connected

Client connected
```

CTRL+C

```
$ telnet localhost 3000

Hello!

Connection closed by
foreign host.
```

```
$ telnet localhost 3000

Hello!

Connection closed by
foreign host.
```

Zamknięcie połączeń

```
public class Server {
    /* ... */
    public void disconnectHandlers() {
        handlers.forEach(handler-> handler.send("Bye!"));
        handlers.clear();
    }
}

public static void main(String[] args) throws IOException {
    Server server = new Server();

    Runtime.getRuntime().addShutdownHook(new Thread(() -> {
        server.disconnectHandlers();
    }));

    server.listen();
}
```

Zerwanie połączenia z serwerem

```
Server started

Client connected

Client connected
```

CTRL+C

```
$ telnet localhost 3000

Hello!



Bye!
Connection closed by
foreign host.
```

```
$ telnet localhost 3000

Hello!

Bye!
Connection closed by
foreign host.
```

Klient

Minimalny klient

```java
public class Main {
    public static void main(String[] args) throws IOException {
        Socket socket = new Socket("localhost", 3000);

        InputStream input = socket.getInputStream();
        OutputStream output = socket.getOutputStream();
        BufferedReader reader = new BufferedReader(new InputStreamReader(input));
        PrintWriter writer = new PrintWriter(output, true);

        writer.println("message");
        String result = reader.readLine();

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

```
Server started
Client connected

Client disconnected
```

```
message
```

Klient jako wątek

```java
public class Client implements Runnable  {
    private final Socket socket;
    private final BufferedReader reader;
    private final PrintWriter writer;

    public Client(String address, int port) throws IOException {
        socket = new Socket(address, port);
        InputStream input = socket.getInputStream();
        OutputStream output = socket.getOutputStream();
        reader = new BufferedReader(new InputStreamReader(input));
        writer = new PrintWriter(output, true);
    }

    @Override
    public void run() { /* ... */ }

    /* ... */
}
```

Metoda run()

```
public class Client implements Runnable {
    /* ... */

    @Override
    public void run() {
        try {
            String message;
            while ((message = reader.readLine()) != null)
                System.out.println(message);
        } catch (IOException e) { e.printStackTrace(); }
    }

    public void send(String message) {
        writer.println(message);
    }
}
```

Uruchomienie klienta

```java
public class Main {
    public static void main(String[] args) throws IOException {
        Client client = new Client("localhost", 3000);
        new Thread(client).start();
        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

        while(true) {
            String message = reader.readLine();
            client.send(message);
        }
    }
}
```

```
Server started
Client connected


Client disconnected
```

```
message
message
```

Wywołanie konsumenta

```
public class Client implements Runnable {

    private Consumer<String> display;

    public void setDisplay(Consumer<String> display) {
        this.display = display;
    }

    @Override
    public void run() {
        try {
            String message;
            while ((message = reader.readLine()) != null)
                display.accept(message);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

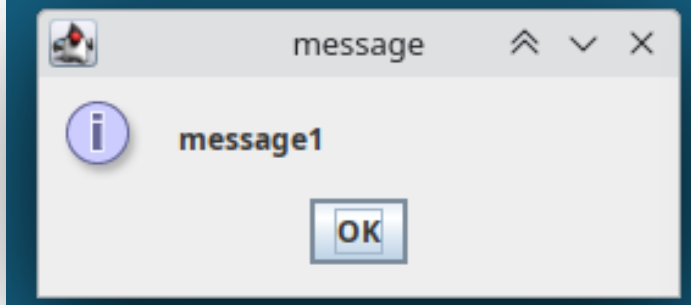
Definicja konsumenta

```
public class Main {
    public static void main(String[] args) throws IOException {
        Client client = new Client("localhost", 3000);
        new Thread(client).start();
        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

        client.setDisplay(message -> { JOptionPane
              .showMessageDialog(null, message, "message", JOptionPane.INFORMATION_MESSAGE);
        });

        while(true) {
            String message = reader.readLine();
            client.send(message);
        }
    }
}
```

Działanie klienta z konsumentem

```
Server started

Client connected
Client connected
```

```
message1
```



Pliki

Droga przesyłania

Wysyłanie pliku z klienta

```
public void sendFile(String path) throws IOException {

    File file = new File(filePath);

    FileInputStream fileIn = new FileInputStream(file);
    DataOutputStream fileOut = new DataOutputStream(socket.getOutputStream());

    byte[] buffer = new byte[64];
    int count;
    while ((count = fileIn.read(buffer)) > 0)
        fileOut.write(buffer,0,count);

    fileIn.close();

}
```

Przesyłanie pliku przez serwer

```
public void transferFile(ClientHandler sender, ClientHandler recipient) throws IOException {

    DataInputStream fileIn = new DataInputStream(sender.getSocket().getInputStream());
    DataOutputStream fileOut = new DataOutputStream(recipient.getSocket().getOutputStream());

    byte[] buffer = new byte[64];
    int count;

    while((count = fileIn.read(buffer)) > 0)
        fileOut.write(buffer, 0, count);

}
```

Odbiór pliku przez klienta

```
public void receiveFile() throws IOException {

    File file = new File(String.valueOf(
                    Path.of(System.getProperty("java.io.tmpdir")).resolve("result.bin")
                ));
    DataInputStream fileIn = new DataInputStream(socket.getInputStream());
    FileOutputStream fileOut = new FileOutputStream(file);

    byte[] buffer = new byte[64];
    int count;
    while ((count = fileIn.read(buffer)) > 0) {
        System.out.print(count);
        fileOut.write(buffer, 0, count);
    }
    fileOut.close();
}
```

```
64
64
64
58
```

Informacja o rozmiarze (nadawca)

```
public void sendFile(String path) throws IOException {

    File file = new File(filePath);
    long fileSize = file.length();
    writer.println(fileSize);

    FileInputStream fileIn = new FileInputStream(file);
    DataOutputStream fileOut = new DataOutputStream(socket.getOutputStream());

    byte[] buffer = new byte[64];
    int count;
    while ((count = fileIn.read(buffer)) > 0)
        fileOut.write(buffer,0,count);

    fileIn.close();

}
```

Informacja o rozmiarze (serwer)

```
public void transferFile(String fileSize, ClientHandler sender, ClientHandler recipient)
throws IOException {

    DataInputStream fileIn = new DataInputStream(sender.getSocket().getInputStream());
    DataOutputStream fileOut = new DataOutputStream(recipient.getSocket().getOutputStream());

    byte[] buffer = new byte[64];
    int count;

    recipient.send(fileSize);

    while((count = fileIn.read(buffer)) > 0)
        fileOut.write(buffer, 0, count);

}
```

Informacja o rozmiarze (odbiorca)

```
public void receiveFile(String size) throws IOException {
  long fileSize = Long.parseLong(size);

    File file = new File(String.valueOf(
                    Path.of(System.getProperty("java.io.tmpdir")).resolve("result.bin")
                ));
    DataInputStream fileIn = new DataInputStream(socket.getInputStream());
    FileOutputStream fileOut = new FileOutputStream(file);

    byte[] buffer = new byte[64];
    int count, receivedSize = 0;
    while ((count = fileIn.read(buffer)) > 0) {
        System.out.print(
     "\r" + (receivedSize * 100 / fileSize) + "%"
   );
        fileOut.write(buffer, 0, count);
    }
    fileOut.close();
}
```

```
25%
51%
76%
100%
```