

# Gniazda TCP / UDP - sprawozdanie z zajęć

Kody źródłowe dostępne w załączonym archiwum.

Zadania wykonałem samodzielnie.

## Zadanie 1 - dwukierunkowa komunikacja UDP <-> UDP

Kod Serwera:

```
1 public class Server {
2     public static void main(String[] args){
3         DatagramSocket socket = null;
4         try{
5             socket = new DatagramSocket(9876);
6             byte[] receiver = new byte[1024];
7
8             InetAddress clientAddress = null;
9             int port = 0;
10
11             while(true){
12                 DatagramPacket receivePacket = new DatagramPacket(receiver, receiver.length);
13                 socket.receive(receivePacket);
14
15                 clientAddress = receivePacket.getAddress();
16                 port = receivePacket.getPort();
17
18                 String msg = new String(receivePacket.getData());
19                 System.out.println("Server received: " + msg);
20                 System.out.println("Server received from: " + clientAddress);
21
22                 break;
23             }
24
25             byte[] sender = "Answer".getBytes();
26             DatagramPacket sendPacket =
27                 new DatagramPacket(sender, sender.length, clientAddress, port);
28             socket.send(sendPacket);
29
30         } catch (Exception e) {
31             e.printStackTrace();
32         }
33     }
34 }
```

Zmiany wprowadzone do kodu serwera:

- Dodałem pola na adres i port nadawcy (linijki 8 i 9)
- Dodałem pobieranie adresu i numeru portu nadawcy (linijki 15 i 16)
- Dodałem możliwość odesłania wiadomości do klienta (linijki 25 - 28)

Kod Klienta:

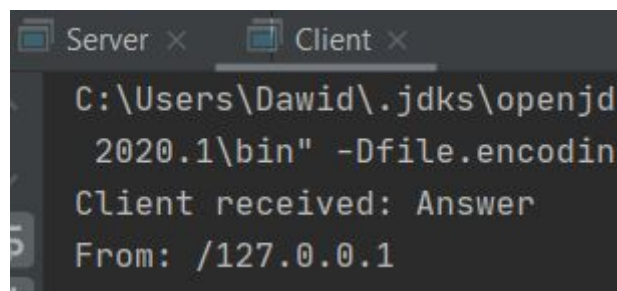
```
1 public class Client {
2     public static void main(String[] args){
3         DatagramSocket socket = null;
4         try {
5             socket = new DatagramSocket();
6             InetAddress address = InetAddress.getByName("localhost");
7             byte[] sendBuffer = "Message".getBytes();
8
9             DatagramPacket packet =
10                 new DatagramPacket(sendBuffer, sendBuffer.length, address, 9876);
11             socket.send(packet);
12
13             byte[] receiveBuffer = new byte[1024];
14
15             while(true) {
16                 DatagramPacket receiver = new DatagramPacket(receiveBuffer, receiveBuffer.length);
17                 socket.receive(receiver);
18
19                 String msg = new String(receiver.getData());
20                 System.out.println("Client received: " + msg);
21                 System.out.println("From: " + receiver.getAddress());
22                 break;
23             }
24         } catch (Exception e) {
25             e.printStackTrace();
26         } finally {
27             if(socket != null){
28                 socket.close();
29             }
30         }
31     }
32 }
33 }
```

Zmiany wprowadzone w kodzie:

- Dodałem kod do otrzymywania wiadomości (linijki 15 - 22)

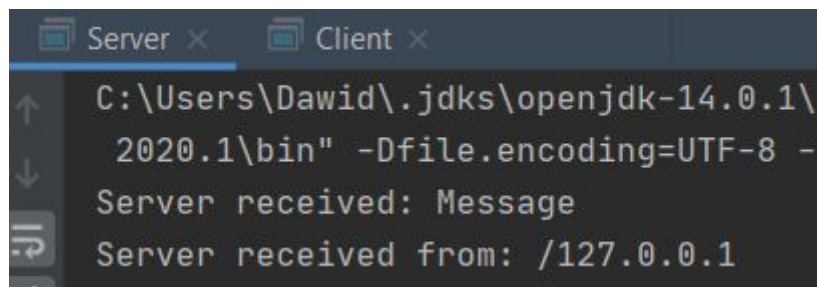
Wyniki działania programów:

Klient:



```
Server x Client x
C:\Users\Dawid\.jdk\openjdk-2020.1\bin" -Dfile.encoding=UTF-8
Client received: Answer
From: /127.0.0.1
```

Server:

A screenshot of a terminal window with two tabs: 'Server' and 'Client'. The 'Server' tab is active. The terminal shows the following text:

```
C:\Users\Dawid\.jdk\openjdk-14.0.1\
2020.1\bin" -Dfile.encoding=UTF-8 -
Server received: Message
Server received from: /127.0.0.1
```

## Zadanie 2 - Komunikacja między Javą, a Pythonem

Kod serwera:



```
1 public class JavaServer {
2     public static void main(String[] args){
3
4         DatagramSocket socket = null;
5
6         try{
7             socket = new DatagramSocket(9876);
8             byte[] receiverB = new byte[1024];
9             while(true){
10                 DatagramPacket packet = new DatagramPacket(receiverB, receiverB.length);
11                 socket.receive(packet);
12
13                 String mes = new String(packet.getData());
14                 System.out.println("Received: " + mes);
15             }
16         } catch (Exception e){
17             e.printStackTrace();
18         } finally{
19             if(socket != null){
20                 socket.close();
21             }
22         }
23     }
24 }
```

Modyfikacje kodu:

- brak modyfikacji

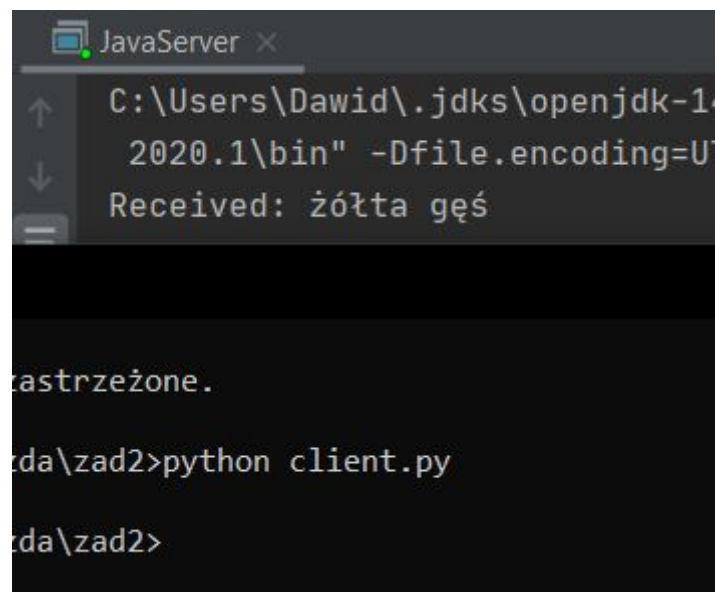
Kod Klienta:

```
1 import socket
2
3 serverIP = "127.0.0.1"
4 serverPort = 9876
5 msg = "żółta gęś"
6
7 client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
8 client.sendto(bytes(msg, 'utf-8'), (serverIP, serverPort))
```

Modyfikacje kodu:

- Zakodowanie wysłanej wiadomości do formatu utf-8

Wyniki działania programów:



The screenshot shows two windows. The top window, titled 'JavaServer', displays the path 'C:\Users\Dawid\.jdk\openjdk-14' and the command '2020.1\bin" -Dfile.encoding=UTF-8'. Below this, it shows 'Received: żółta gęś'. The bottom window is a terminal with the command 'da\zad2>python client.py' and the prompt 'da\zad2>'.

```
JavaServer x
C:\Users\Dawid\.jdk\openjdk-14
2020.1\bin" -Dfile.encoding=UTF-8
Received: żółta gęś

da\zad2>python client.py
da\zad2>
```

## Zadanie 3 - Przesył liczby między Pythonem a Javą

Kod serwera:

```
1 public class Server {
2     public static void main(String[] args){
3         DatagramSocket socket = null;
4         try{
5             socket = new DatagramSocket(9876);
6             InetAddress clientAddress = null;
7             int port = 0;
8             int num;
9             while(true){
10                 DatagramPacket receivePacket = new DatagramPacket(new byte[256], 256);
11                 socket.receive(receivePacket);
12
13                 clientAddress = receivePacket.getAddress();
14                 port = receivePacket.getPort();
15
16                 num =
17                 ByteBuffer.wrap(receivePacket.getData()).order(ByteOrder.LITTLE_ENDIAN).getInt();
18                 System.out.println("Server received: " + num);
19                 System.out.println("Server received from: " + clientAddress);
20
21                 break;
22             }
23             num++;
24             byte[] sender =
25             ByteBuffer.allocate(4).order(ByteOrder.LITTLE_ENDIAN).putInt(num).array();
26             DatagramPacket sendPacket =
27             new DatagramPacket(sender, sender.length, clientAddress, port);
28             socket.send(sendPacket);
29         } catch (Exception e) {
30             e.printStackTrace();
31         }
32     }
```

Modyfikacje kodu:

- adres i port klienta (analogicznie jak w zadaniu 1)
- dodanie wywołania order (w liniijkach 16 i 23)

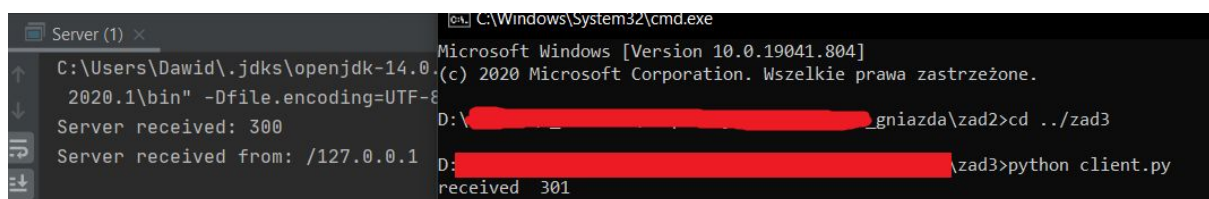
Kod klienta:

```
1 import socket
2
3 serverIP = "127.0.0.1"
4 serverPort = 9876
5 msg_bytes = (300).to_bytes(4, byteorder='little')
6
7 client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
8 client.sendto(msg_bytes, (serverIP, serverPort))
9
10 while True:
11     buff, address = client.recvfrom(9876)
12     print("received ", int.from_bytes(buff, byteorder="little"))
13     break
```

Modyfikacje kodu:

- inna wiadomość do wysłania (linijka 5)
- dodanie czekania na wiadomość (linijki 10 - 13)
- zmiana byteorder na "little" (linijki 5 oraz 12)

Wyniki działania:



```
Server (1) x
C:\Users\Dawid\.jdk\openjdk-14.0.2\bin -Dfile.encoding=UTF-8
Server received: 300
Server received from: /127.0.0.1

C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19041.804]
(c) 2020 Microsoft Corporation. Wszelkie prawa zastrzeżone.
D:\gniazda\zad2>cd ../zad3
D:\gniazda\zad3>python client.py
received 301
```

Jak widać, serwer otrzymał liczbę 300, natomiast klient otrzymał liczbę 301.