Ćwicz. 1.

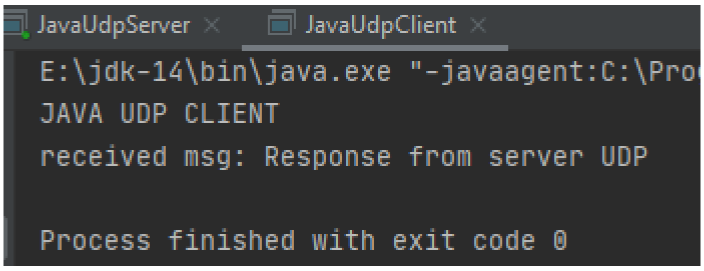
Dodany kod w „JavaUdpServer”:

```
// Send response to client: (added code)
InetAddress senderAddress = receivePacket.getAddress();
int senderPort = receivePacket.getPort();
byte[] sendBuffer = "Response from server UDP".getBytes();
DatagramPacket sendResponse = new DatagramPacket(sendBuffer,
sendBuffer.length, senderAddress, senderPort);
socket.send(sendResponse);
```

Dodany kod w “JavaUdpClient”:

```
// Receive response from server:
byte[] receiveBuffer = new byte[1024];
DatagramPacket receivePacket = new
DatagramPacket(receiveBuffer, receiveBuffer.length);
socket.receive(receivePacket);
String msg = new String(receivePacket.getData());
System.out.println("received msg: " + msg);
```

Rezultat:



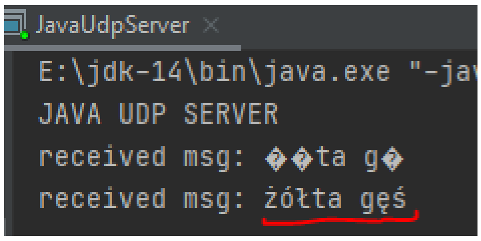
```
E:\jdk-14\bin\java.exe "-javaagent:C:\Pro
JAVA UDP CLIENT
received msg: Response from server UDP

Process finished with exit code 0
```

Ćwicz. 2.

Zmienione kodowanie w “PythonUdpClient” na UTF-8:

```
client.sendto(bytes(msg, 'UTF-8'), (serverIP, serverPort))
```

Rezultat:



```
E:\jdk-14\bin\java.exe "-ja
JAVA UDP SERVER
received msg: ��ta g�
received msg: żółta gęś
```

Ćwicz. 3.

Przed wysłaniem i po odebraniu danych konwertujemy kolejność bajtów na little endian.

Kod w “PythonUdpClient”:

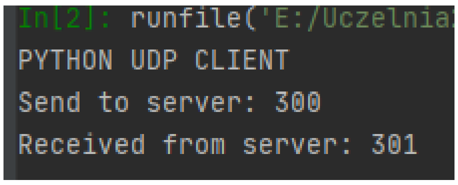
```
msg = 300
msg_bytes = msg.to_bytes(4, byteorder='little')
```

```
client.sendto(bytes(msg_bytes), (serverIP, serverPort))

print("Send to server: " + str(msg))
buff, address = client.recvfrom(1024)
print("Received from server: " + str(int.from_bytes(buff,
byteorder='little')))
```

Kod w „JavaUdpServer”:

```
int nb =
ByteBuffer.wrap(receiveBuffer).order(ByteOrder.LITTLE_ENDIAN).
getInt();
System.out.println("received msg: " + nb);

// Send response to client:
InetAddress senderAddress = receivePacket.getAddress();
int senderPort = receivePacket.getPort();
// byte[] sendBuffer = "Response from server UDP".getBytes();
byte[] sendBuffer =
ByteBuffer.allocate(4).order(ByteOrder.LITTLE_ENDIAN).putInt(+
+nb).array();
DatagramPacket sendResponse = new DatagramPacket(sendBuffer,
sendBuffer.length, senderAddress, senderPort);
socket.send(sendResponse);
```

Rezultat:



```
In[2]: runfile('E:/Uczelnia
PYTHON UDP CLIENT
Send to server: 300
Received from server: 301
```

Ćwicz. 4.

Przesyłam dane w postaci obiektu(zamienionego na JSONa), obiekt ma pole msg oraz clientType.

Kod w “PythonUdpClient”:

```
data = {
    "msg": "Ping Python Udp",
    "clientType": "PYTHON"
}

client.sendto(bytes(json.dumps(data), 'UTF-8'), (serverIP, serverPort))

buff, address = client.recvfrom(1024)
# print("Received from server: " + str(int.from_bytes(buff,
byteorder='little')))
print("Received from server: " + str(buff, 'utf-8'))
```

Kod w „JavaUdpServer”:

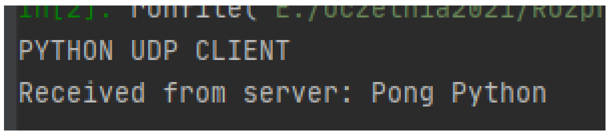
```
// Receive object:
Gson gson = new Gson();
DataModel data = gson.fromJson(new String(receivePacket.getData()).trim(),
DataModel.class);
```

```java
System.out.println("received msg: " + data.msg);

// Send response to client:
InetAddress senderAddress = receivePacket.getAddress();
int senderPort = receivePacket.getPort();
// byte[] sendBuffer = "Response from server UDP: ".getBytes();
// byte[] sendBuffer =
ByteBuffer.allocate(4).order(ByteOrder.LITTLE_ENDIAN).putInt(++nb).array();
byte[] sendBuffer = ("Pong " + data.clientType).getBytes();
DatagramPacket sendResponse = new DatagramPacket(sendBuffer,
sendBuffer.length, senderAddress, senderPort);
socket.send(sendResponse);
```

Użyty model danych:

```java
import java.io.Serializable;

public class DataModel implements Serializable {
    public String msg;
    public ClientType clientType;

    public DataModel(String msg, ClientType clientType) {
        this.msg = msg;
        this.clientType = clientType;
    }

    public enum ClientType {
        JAVA("Java"),
        PYTHON("Python");

        private final String text;

        ClientType(String text) {
            this.text = text;
        }

        @Override
        public String toString() {
            return this.text;
        }
    }

}
```

Rezultat:



```
In[2]: runfile('E:/Uczelnia2021/Rozpr
PYTHON UDP CLIENT
Received from server: Pong Python
```