Java Sockets

Network Programming with Java

DI. Dr. Peter René Dietmüller

Institut für Informationsverarbeitung und Mikroprzessortechnik

Johannes Kepler Universität Linz

Überblick

- TCP-Kommunikation
- UDP-Kommunikation
- Klasse Socket
- Klasse ServerSocket
- Klasse DatagramPacket
- Klasse DatagramSocket

TCP-Kommunikation

	Client	Server
Klasse	Socket	ServerSocket
Socket öffnen	Konstruktor	Konstruktor
Socket benennen	--	--
Socket verbinden	--	accept()
Daten senden, empfangen	getInputStream() / getOutputStream()	getInputStream() / getOutputStream()
Socket schließen	close()	close()

UDP-Kommunikation

	Client	Server
Klasse	DatagramSocket, DatagramPacket	DatagramSocket, DatagramPacket
Socket öffnen	Konstruktor	Konstruktor
Socket benennen	--	--
Socket verbinden	--	--
Daten senden, empf.	send() / receive()	send() / receive()
Socket schließen	close()	close()

Socket (1)

```
public Socket(InetAddress address, int port)
                throws IOException
```

Parameter:

- address - the IP address.
- port - the port number.

Throws:

- IOException - if an I/O error occurs when creating the socket.
- SecurityException - if a security manager exists and its check-Connect method doesn't allow the operation.

Socket (2)

```
public Socket(String host, int port)
        throws UnknownHostException,
                IOException
```

Parameters:

- host - the host name.
- port - the port number.

Throws:

- IOException - if an I/O error occurs when creating the socket.
- SecurityException - if a security manager exists and its checkConnect method doesn't allow the operation.

Example: Socket s = new Socket(„www.fim.uni-linz.ac.at“, 80)

Socket (3)

```
public Socket(String host, int port,
              InetAddress lAddr, int lPort)
       throws IOException
```

Parameters:

- host - the name of the remote host
- port - the remote port
- lAddr - the local address the socket is bound to
- lPort - the local port the socket is bound to

Throws:

- SecurityException - if a security manager exists and its checkConnect method doesn't allow the operation.

Method getInputStream

```
public InputStream getInputStream()
                    throws IOException
```

Returns:

- an input stream for reading bytes from this socket.

Throws:

- IOException - if an I/O error occurs when creating the input stream.

Example:

```
BufferedReader sin = new BufferedReader(
 new InputStreamReader(s.getInputStream()));
zeile = sin.readLine();
```

Method getOutputStream

```
public OutputStream getOutputStream()
                    throws IOException
```

Returns:

- an output stream for writing bytes to this socket.

Throws:

- IOException - if an I/O error occurs when creating the output stream.

Example:

```
PrintWriter sout = new
     PrintWriter(s.getOutputStream(), true);
sout.println(zeile);
```

Method close

```
public void close()
            throws IOException
```

Throws:

- IOException - if an I/O error occurs when closing this socket.

ServerSocket (1)

```
public ServerSocket(int port)
                throws IOException
```

Parameters:

- port - the port number, or 0 to use any free port.

Throws:

- IOException - if an I/O error occurs when opening the socket.
- SecurityException - if a security manager exists and its checkListen method doesn't allow the operation.

Example:

```
ServerSocket s = new ServerSocket(80);
```

ServerSocket (2)

```
public ServerSocket(int port,
                    int backlog)
             throws IOException
```

Parameters:

- port - the specified port, or 0 to use any free port.
- backlog - the maximum length of the queue.

Throws:

- IOException - if an I/O error occurs when opening the socket.
- SecurityException - if a security manager exists and its checkListen method doesn't allow the operation.

Method accept

```
public Socket accept()
                throws IOException
```

Throws:

- IOException - if an I/O error occurs when waiting for a connection.
- SecurityException - if a security manager exists and its checkListen method doesn't allow the operation.

Example:

```
ServerSocket ss = new ServerSocket(80);
Socket s = ss.accept(); /* blocks until a
                           connection is made */
```

DatagramPacket (1)

```
public DatagramPacket(byte[] buf,
                      int length)
```

Parameters:

- buf - buffer for holding the incoming datagram.
- length - the number of bytes to read.

Example:

```
/* -- Packet for receiving data -- */
p = new DatagramPacket(d, d.length);
```

DatagramPacket (2)

```
public DatagramPacket(byte[] buf,
                      int length,
                      InetAddress address,
                      int port)
```

Parameters:

- buf - the packet data.
- length - the packet length.
- addr - the destination address.
- port - the destination port number.

Example:

```
p = new DatagramPacket(d, d.length, adr, port);
```

DatagramSocket (1)

```
public DatagramSocket()
                    throws SocketException
```

Throws:

- SocketException - if the socket could not be opened, or the socket could not bind to the specified local port.
- SecurityException - if a security manager exists and its checkListen method doesn't allow the operation.

Example:

```
s = new DatagramSocket();
```

DatagramSocket (2)

```
public DatagramSocket(int port)
                    throws SocketException
```

Parameters:

- port - port to use.

Throws:

- SocketException - if the socket could not be opened, or the socket could not bind to the specified local port.
- SecurityException - if a security manager exists and its checkListen method doesn't allow the operation.

Example:

```
s = new DatagramSocket(1920);
```

Method send

```
public void send(DatagramPacket p)
          throws IOException
```

Parameters:

- p - the DatagramPacket to be sent.

Throws:

- IOException - if an I/O error occurs.
- SecurityException - if a security manager exists and its checkMulticast or checkConnect method doesn't allow the send.

Example: (assuming that d is a byte array)

```
s.send(new DatagramPacket(d, d.length, adr, port));
```

Method receive

```
public void receive(DatagramPacket p)
                throws IOException
```

Parameters:

- p - the DatagramPacket into which to place the incoming data.

Throws:

- IOException - if an I/O error occurs.

Example: (assuming that d is a byte array)

```
s = new DatagramSocket(PORT);
p = new DatagramPacket(d, d.length);
s.receive(p); s.close();
```