| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Socket.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/net/ServerSocket.html)   [**NEXT CLASS**](http://docs.google.com/java/net/SocketAddress.html) | [**FRAMES**](http://docs.google.com/index.html?java/net/Socket.html)    [**NO FRAMES**](http://docs.google.com/Socket.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#44sinio) |

## **java.net**

Class Socket

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 **java.net.Socket**

**Direct Known Subclasses:** [SSLSocket](http://docs.google.com/javax/net/ssl/SSLSocket.html)

public class **Socket**extends [Object](http://docs.google.com/java/lang/Object.html)

This class implements client sockets (also called just "sockets"). A socket is an endpoint for communication between two machines.

The actual work of the socket is performed by an instance of the SocketImpl class. An application, by changing the socket factory that creates the socket implementation, can configure itself to create sockets appropriate to the local firewall.

**Since:** JDK1.0 **See Also:**[setSocketImplFactory(java.net.SocketImplFactory)](http://docs.google.com/java/net/Socket.html#setSocketImplFactory(java.net.SocketImplFactory)), [SocketImpl](http://docs.google.com/java/net/SocketImpl.html), [SocketChannel](http://docs.google.com/java/nio/channels/SocketChannel.html)

| **Constructor Summary** | |
| --- | --- |
|  | [**Socket**](http://docs.google.com/java/net/Socket.html#Socket())()            Creates an unconnected socket, with the system-default type of SocketImpl. |
|  | [**Socket**](http://docs.google.com/java/net/Socket.html#Socket(java.net.InetAddress,%20int))([InetAddress](http://docs.google.com/java/net/InetAddress.html) address, int port)            Creates a stream socket and connects it to the specified port number at the specified IP address. |
|  | [**Socket**](http://docs.google.com/java/net/Socket.html#Socket(java.net.InetAddress,%20int,%20boolean))([InetAddress](http://docs.google.com/java/net/InetAddress.html) host, int port, boolean stream)  **Deprecated.** *Use DatagramSocket instead for UDP transport.* |
|  | [**Socket**](http://docs.google.com/java/net/Socket.html#Socket(java.net.InetAddress,%20int,%20java.net.InetAddress,%20int))([InetAddress](http://docs.google.com/java/net/InetAddress.html) address, int port, [InetAddress](http://docs.google.com/java/net/InetAddress.html) localAddr, int localPort)            Creates a socket and connects it to the specified remote address on the specified remote port. |
|  | [**Socket**](http://docs.google.com/java/net/Socket.html#Socket(java.net.Proxy))([Proxy](http://docs.google.com/java/net/Proxy.html) proxy)            Creates an unconnected socket, specifying the type of proxy, if any, that should be used regardless of any other settings. |
| protected | [**Socket**](http://docs.google.com/java/net/Socket.html#Socket(java.net.SocketImpl))([SocketImpl](http://docs.google.com/java/net/SocketImpl.html) impl)            Creates an unconnected Socket with a user-specified SocketImpl. |
|  | [**Socket**](http://docs.google.com/java/net/Socket.html#Socket(java.lang.String,%20int))([String](http://docs.google.com/java/lang/String.html) host, int port)            Creates a stream socket and connects it to the specified port number on the named host. |
|  | [**Socket**](http://docs.google.com/java/net/Socket.html#Socket(java.lang.String,%20int,%20boolean))([String](http://docs.google.com/java/lang/String.html) host, int port, boolean stream)  **Deprecated.** *Use DatagramSocket instead for UDP transport.* |
|  | [**Socket**](http://docs.google.com/java/net/Socket.html#Socket(java.lang.String,%20int,%20java.net.InetAddress,%20int))([String](http://docs.google.com/java/lang/String.html) host, int port, [InetAddress](http://docs.google.com/java/net/InetAddress.html) localAddr, int localPort)            Creates a socket and connects it to the specified remote host on the specified remote port. |

| **Method Summary** | |
| --- | --- |
| void | [**bind**](http://docs.google.com/java/net/Socket.html#bind(java.net.SocketAddress))([SocketAddress](http://docs.google.com/java/net/SocketAddress.html) bindpoint)            Binds the socket to a local address. |
| void | [**close**](http://docs.google.com/java/net/Socket.html#close())()            Closes this socket. |
| void | [**connect**](http://docs.google.com/java/net/Socket.html#connect(java.net.SocketAddress))([SocketAddress](http://docs.google.com/java/net/SocketAddress.html) endpoint)            Connects this socket to the server. |
| void | [**connect**](http://docs.google.com/java/net/Socket.html#connect(java.net.SocketAddress,%20int))([SocketAddress](http://docs.google.com/java/net/SocketAddress.html) endpoint, int timeout)            Connects this socket to the server with a specified timeout value. |
| [SocketChannel](http://docs.google.com/java/nio/channels/SocketChannel.html) | [**getChannel**](http://docs.google.com/java/net/Socket.html#getChannel())()            Returns the unique [SocketChannel](http://docs.google.com/java/nio/channels/SocketChannel.html) object associated with this socket, if any. |
| [InetAddress](http://docs.google.com/java/net/InetAddress.html) | [**getInetAddress**](http://docs.google.com/java/net/Socket.html#getInetAddress())()            Returns the address to which the socket is connected. |
| [InputStream](http://docs.google.com/java/io/InputStream.html) | [**getInputStream**](http://docs.google.com/java/net/Socket.html#getInputStream())()            Returns an input stream for this socket. |
| boolean | [**getKeepAlive**](http://docs.google.com/java/net/Socket.html#getKeepAlive())()            Tests if SO\_KEEPALIVE is enabled. |
| [InetAddress](http://docs.google.com/java/net/InetAddress.html) | [**getLocalAddress**](http://docs.google.com/java/net/Socket.html#getLocalAddress())()            Gets the local address to which the socket is bound. |
| int | [**getLocalPort**](http://docs.google.com/java/net/Socket.html#getLocalPort())()            Returns the local port to which this socket is bound. |
| [SocketAddress](http://docs.google.com/java/net/SocketAddress.html) | [**getLocalSocketAddress**](http://docs.google.com/java/net/Socket.html#getLocalSocketAddress())()            Returns the address of the endpoint this socket is bound to, or null if it is not bound yet. |
| boolean | [**getOOBInline**](http://docs.google.com/java/net/Socket.html#getOOBInline())()            Tests if OOBINLINE is enabled. |
| [OutputStream](http://docs.google.com/java/io/OutputStream.html) | [**getOutputStream**](http://docs.google.com/java/net/Socket.html#getOutputStream())()            Returns an output stream for this socket. |
| int | [**getPort**](http://docs.google.com/java/net/Socket.html#getPort())()            Returns the remote port to which this socket is connected. |
| int | [**getReceiveBufferSize**](http://docs.google.com/java/net/Socket.html#getReceiveBufferSize())()            Gets the value of the SO\_RCVBUF option for this Socket, that is the buffer size used by the platform for input on this Socket. |
| [SocketAddress](http://docs.google.com/java/net/SocketAddress.html) | [**getRemoteSocketAddress**](http://docs.google.com/java/net/Socket.html#getRemoteSocketAddress())()            Returns the address of the endpoint this socket is connected to, or null if it is unconnected. |
| boolean | [**getReuseAddress**](http://docs.google.com/java/net/Socket.html#getReuseAddress())()            Tests if SO\_REUSEADDR is enabled. |
| int | [**getSendBufferSize**](http://docs.google.com/java/net/Socket.html#getSendBufferSize())()            Get value of the SO\_SNDBUF option for this Socket, that is the buffer size used by the platform for output on this Socket. |
| int | [**getSoLinger**](http://docs.google.com/java/net/Socket.html#getSoLinger())()            Returns setting for SO\_LINGER. |
| int | [**getSoTimeout**](http://docs.google.com/java/net/Socket.html#getSoTimeout())()            Returns setting for SO\_TIMEOUT. |
| boolean | [**getTcpNoDelay**](http://docs.google.com/java/net/Socket.html#getTcpNoDelay())()            Tests if TCP\_NODELAY is enabled. |
| int | [**getTrafficClass**](http://docs.google.com/java/net/Socket.html#getTrafficClass())()            Gets traffic class or type-of-service in the IP header for packets sent from this Socket |
| boolean | [**isBound**](http://docs.google.com/java/net/Socket.html#isBound())()            Returns the binding state of the socket. |
| boolean | [**isClosed**](http://docs.google.com/java/net/Socket.html#isClosed())()            Returns the closed state of the socket. |
| boolean | [**isConnected**](http://docs.google.com/java/net/Socket.html#isConnected())()            Returns the connection state of the socket. |
| boolean | [**isInputShutdown**](http://docs.google.com/java/net/Socket.html#isInputShutdown())()            Returns whether the read-half of the socket connection is closed. |
| boolean | [**isOutputShutdown**](http://docs.google.com/java/net/Socket.html#isOutputShutdown())()            Returns whether the write-half of the socket connection is closed. |
| void | [**sendUrgentData**](http://docs.google.com/java/net/Socket.html#sendUrgentData(int))(int data)            Send one byte of urgent data on the socket. |
| void | [**setKeepAlive**](http://docs.google.com/java/net/Socket.html#setKeepAlive(boolean))(boolean on)            Enable/disable SO\_KEEPALIVE. |
| void | [**setOOBInline**](http://docs.google.com/java/net/Socket.html#setOOBInline(boolean))(boolean on)            Enable/disable OOBINLINE (receipt of TCP urgent data) By default, this option is disabled and TCP urgent data received on a socket is silently discarded. |
| void | [**setPerformancePreferences**](http://docs.google.com/java/net/Socket.html#setPerformancePreferences(int,%20int,%20int))(int connectionTime, int latency, int bandwidth)            Sets performance preferences for this socket. |
| void | [**setReceiveBufferSize**](http://docs.google.com/java/net/Socket.html#setReceiveBufferSize(int))(int size)            Sets the SO\_RCVBUF option to the specified value for this Socket. |
| void | [**setReuseAddress**](http://docs.google.com/java/net/Socket.html#setReuseAddress(boolean))(boolean on)            Enable/disable the SO\_REUSEADDR socket option. |
| void | [**setSendBufferSize**](http://docs.google.com/java/net/Socket.html#setSendBufferSize(int))(int size)            Sets the SO\_SNDBUF option to the specified value for this Socket. |
| static void | [**setSocketImplFactory**](http://docs.google.com/java/net/Socket.html#setSocketImplFactory(java.net.SocketImplFactory))([SocketImplFactory](http://docs.google.com/java/net/SocketImplFactory.html) fac)            Sets the client socket implementation factory for the application. |
| void | [**setSoLinger**](http://docs.google.com/java/net/Socket.html#setSoLinger(boolean,%20int))(boolean on, int linger)            Enable/disable SO\_LINGER with the specified linger time in seconds. |
| void | [**setSoTimeout**](http://docs.google.com/java/net/Socket.html#setSoTimeout(int))(int timeout)            Enable/disable SO\_TIMEOUT with the specified timeout, in milliseconds. |
| void | [**setTcpNoDelay**](http://docs.google.com/java/net/Socket.html#setTcpNoDelay(boolean))(boolean on)            Enable/disable TCP\_NODELAY (disable/enable Nagle's algorithm). |
| void | [**setTrafficClass**](http://docs.google.com/java/net/Socket.html#setTrafficClass(int))(int tc)            Sets traffic class or type-of-service octet in the IP header for packets sent from this Socket. |
| void | [**shutdownInput**](http://docs.google.com/java/net/Socket.html#shutdownInput())()            Places the input stream for this socket at "end of stream". |
| void | [**shutdownOutput**](http://docs.google.com/java/net/Socket.html#shutdownOutput())()            Disables the output stream for this socket. |
| [String](http://docs.google.com/java/lang/String.html) | [**toString**](http://docs.google.com/java/net/Socket.html#toString())()            Converts this socket to a String. |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [clone](http://docs.google.com/java/lang/Object.html#clone()), [equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [hashCode](http://docs.google.com/java/lang/Object.html#hashCode()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [wait](http://docs.google.com/java/lang/Object.html#wait()), [wait](http://docs.google.com/java/lang/Object.html#wait(long)), [wait](http://docs.google.com/java/lang/Object.html#wait(long,%20int)) |

| **Constructor Detail** |
| --- |

### Socket

public **Socket**()

Creates an unconnected socket, with the system-default type of SocketImpl.

**Since:** JDK1.1

### Socket

public **Socket**([Proxy](http://docs.google.com/java/net/Proxy.html) proxy)

Creates an unconnected socket, specifying the type of proxy, if any, that should be used regardless of any other settings.

If there is a security manager, its checkConnect method is called with the proxy host address and port number as its arguments. This could result in a SecurityException.

Examples:

* Socket s = new Socket(Proxy.NO\_PROXY); will create a plain socket ignoring any other proxy configuration.
* Socket s = new Socket(new Proxy(Proxy.Type.SOCKS, new InetSocketAddress("socks.mydom.com", 1080))); will create a socket connecting through the specified SOCKS proxy server.

**Parameters:**proxy - a [Proxy](http://docs.google.com/java/net/Proxy.html) object specifying what kind of proxying should be used. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the proxy is of an invalid type or null. [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager is present and permission to connect to the proxy is denied.**Since:** 1.5 **See Also:**[ProxySelector](http://docs.google.com/java/net/ProxySelector.html), [Proxy](http://docs.google.com/java/net/Proxy.html)

### Socket

protected **Socket**([SocketImpl](http://docs.google.com/java/net/SocketImpl.html) impl)  
 throws [SocketException](http://docs.google.com/java/net/SocketException.html)

Creates an unconnected Socket with a user-specified SocketImpl.

**Parameters:**impl - an instance of a **SocketImpl** the subclass wishes to use on the Socket. **Throws:** [SocketException](http://docs.google.com/java/net/SocketException.html) - if there is an error in the underlying protocol, such as a TCP error.**Since:** JDK1.1

### Socket

public **Socket**([String](http://docs.google.com/java/lang/String.html) host,  
 int port)  
 throws [UnknownHostException](http://docs.google.com/java/net/UnknownHostException.html),  
 [IOException](http://docs.google.com/java/io/IOException.html)

Creates a stream socket and connects it to the specified port number on the named host.

If the specified host is null it is the equivalent of specifying the address as [InetAddress.getByName](http://docs.google.com/java/net/InetAddress.html#getByName(java.lang.String))(null). In other words, it is equivalent to specifying an address of the loopback interface.

If the application has specified a server socket factory, that factory's createSocketImpl method is called to create the actual socket implementation. Otherwise a "plain" socket is created.

If there is a security manager, its checkConnect method is called with the host address and port as its arguments. This could result in a SecurityException.

**Parameters:**host - the host name, or null for the loopback address.port - the port number. **Throws:** [UnknownHostException](http://docs.google.com/java/net/UnknownHostException.html) - if the IP address of the host could not be determined. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs when creating the socket. [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and its checkConnect method doesn't allow the operation.**See Also:**[setSocketImplFactory(java.net.SocketImplFactory)](http://docs.google.com/java/net/Socket.html#setSocketImplFactory(java.net.SocketImplFactory)), [SocketImpl](http://docs.google.com/java/net/SocketImpl.html), [SocketImplFactory.createSocketImpl()](http://docs.google.com/java/net/SocketImplFactory.html#createSocketImpl()), [SecurityManager.checkConnect(java.lang.String, int)](http://docs.google.com/java/lang/SecurityManager.html#checkConnect(java.lang.String,%20int))

### Socket

public **Socket**([InetAddress](http://docs.google.com/java/net/InetAddress.html) address,  
 int port)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Creates a stream socket and connects it to the specified port number at the specified IP address.

If the application has specified a socket factory, that factory's createSocketImpl method is called to create the actual socket implementation. Otherwise a "plain" socket is created.

If there is a security manager, its checkConnect method is called with the host address and port as its arguments. This could result in a SecurityException.

**Parameters:**address - the IP address.port - the port number. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs when creating the socket. [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and its checkConnect method doesn't allow the operation.**See Also:**[setSocketImplFactory(java.net.SocketImplFactory)](http://docs.google.com/java/net/Socket.html#setSocketImplFactory(java.net.SocketImplFactory)), [SocketImpl](http://docs.google.com/java/net/SocketImpl.html), [SocketImplFactory.createSocketImpl()](http://docs.google.com/java/net/SocketImplFactory.html#createSocketImpl()), [SecurityManager.checkConnect(java.lang.String, int)](http://docs.google.com/java/lang/SecurityManager.html#checkConnect(java.lang.String,%20int))

### Socket

public **Socket**([String](http://docs.google.com/java/lang/String.html) host,  
 int port,  
 [InetAddress](http://docs.google.com/java/net/InetAddress.html) localAddr,  
 int localPort)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Creates a socket and connects it to the specified remote host on the specified remote port. The Socket will also bind() to the local address and port supplied.

If the specified host is null it is the equivalent of specifying the address as [InetAddress.getByName](http://docs.google.com/java/net/InetAddress.html#getByName(java.lang.String))(null). In other words, it is equivalent to specifying an address of the loopback interface.

If there is a security manager, its checkConnect method is called with the host address and port as its arguments. This could result in a SecurityException.

**Parameters:**host - the name of the remote host, or null for the loopback address.port - the remote portlocalAddr - the local address the socket is bound tolocalPort - the local port the socket is bound to **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs when creating the socket. [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and its checkConnect method doesn't allow the operation.**Since:** JDK1.1 **See Also:**[SecurityManager.checkConnect(java.lang.String, int)](http://docs.google.com/java/lang/SecurityManager.html#checkConnect(java.lang.String,%20int))

### Socket

public **Socket**([InetAddress](http://docs.google.com/java/net/InetAddress.html) address,  
 int port,  
 [InetAddress](http://docs.google.com/java/net/InetAddress.html) localAddr,  
 int localPort)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Creates a socket and connects it to the specified remote address on the specified remote port. The Socket will also bind() to the local address and port supplied.

If there is a security manager, its checkConnect method is called with the host address and port as its arguments. This could result in a SecurityException.

**Parameters:**address - the remote addressport - the remote portlocalAddr - the local address the socket is bound tolocalPort - the local port the socket is bound to **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs when creating the socket. [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and its checkConnect method doesn't allow the operation.**Since:** JDK1.1 **See Also:**[SecurityManager.checkConnect(java.lang.String, int)](http://docs.google.com/java/lang/SecurityManager.html#checkConnect(java.lang.String,%20int))

### Socket

[@Deprecated](http://docs.google.com/java/lang/Deprecated.html)  
public **Socket**([String](http://docs.google.com/java/lang/String.html) host,  
 int port,  
 boolean stream)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

**Deprecated.** *Use DatagramSocket instead for UDP transport.*

Creates a stream socket and connects it to the specified port number on the named host.

If the specified host is null it is the equivalent of specifying the address as [InetAddress.getByName](http://docs.google.com/java/net/InetAddress.html#getByName(java.lang.String))(null). In other words, it is equivalent to specifying an address of the loopback interface.

If the stream argument is true, this creates a stream socket. If the stream argument is false, it creates a datagram socket.

If the application has specified a server socket factory, that factory's createSocketImpl method is called to create the actual socket implementation. Otherwise a "plain" socket is created.

If there is a security manager, its checkConnect method is called with the host address and port as its arguments. This could result in a SecurityException.

If a UDP socket is used, TCP/IP related socket options will not apply.

**Parameters:**host - the host name, or null for the loopback address.port - the port number.stream - a boolean indicating whether this is a stream socket or a datagram socket. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs when creating the socket. [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and its checkConnect method doesn't allow the operation.**See Also:**[setSocketImplFactory(java.net.SocketImplFactory)](http://docs.google.com/java/net/Socket.html#setSocketImplFactory(java.net.SocketImplFactory)), [SocketImpl](http://docs.google.com/java/net/SocketImpl.html), [SocketImplFactory.createSocketImpl()](http://docs.google.com/java/net/SocketImplFactory.html#createSocketImpl()), [SecurityManager.checkConnect(java.lang.String, int)](http://docs.google.com/java/lang/SecurityManager.html#checkConnect(java.lang.String,%20int))

### Socket

[@Deprecated](http://docs.google.com/java/lang/Deprecated.html)  
public **Socket**([InetAddress](http://docs.google.com/java/net/InetAddress.html) host,  
 int port,  
 boolean stream)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

**Deprecated.** *Use DatagramSocket instead for UDP transport.*

Creates a socket and connects it to the specified port number at the specified IP address.

If the stream argument is true, this creates a stream socket. If the stream argument is false, it creates a datagram socket.

If the application has specified a server socket factory, that factory's createSocketImpl method is called to create the actual socket implementation. Otherwise a "plain" socket is created.

If there is a security manager, its checkConnect method is called with host.getHostAddress() and port as its arguments. This could result in a SecurityException.

If UDP socket is used, TCP/IP related socket options will not apply.

**Parameters:**host - the IP address.port - the port number.stream - if true, create a stream socket; otherwise, create a datagram socket. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs when creating the socket. [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and its checkConnect method doesn't allow the operation.**See Also:**[setSocketImplFactory(java.net.SocketImplFactory)](http://docs.google.com/java/net/Socket.html#setSocketImplFactory(java.net.SocketImplFactory)), [SocketImpl](http://docs.google.com/java/net/SocketImpl.html), [SocketImplFactory.createSocketImpl()](http://docs.google.com/java/net/SocketImplFactory.html#createSocketImpl()), [SecurityManager.checkConnect(java.lang.String, int)](http://docs.google.com/java/lang/SecurityManager.html#checkConnect(java.lang.String,%20int))

| **Method Detail** |
| --- |

### connect

public void **connect**([SocketAddress](http://docs.google.com/java/net/SocketAddress.html) endpoint)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Connects this socket to the server.

**Parameters:**endpoint - the SocketAddress **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs during the connection [IllegalBlockingModeException](http://docs.google.com/java/nio/channels/IllegalBlockingModeException.html) - if this socket has an associated channel, and the channel is in non-blocking mode [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if endpoint is null or is a SocketAddress subclass not supported by this socket**Since:** 1.4

### connect

public void **connect**([SocketAddress](http://docs.google.com/java/net/SocketAddress.html) endpoint,  
 int timeout)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Connects this socket to the server with a specified timeout value. A timeout of zero is interpreted as an infinite timeout. The connection will then block until established or an error occurs.

**Parameters:**endpoint - the SocketAddresstimeout - the timeout value to be used in milliseconds. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs during the connection [SocketTimeoutException](http://docs.google.com/java/net/SocketTimeoutException.html) - if timeout expires before connecting [IllegalBlockingModeException](http://docs.google.com/java/nio/channels/IllegalBlockingModeException.html) - if this socket has an associated channel, and the channel is in non-blocking mode [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if endpoint is null or is a SocketAddress subclass not supported by this socket**Since:** 1.4

### bind

public void **bind**([SocketAddress](http://docs.google.com/java/net/SocketAddress.html) bindpoint)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Binds the socket to a local address.

If the address is null, then the system will pick up an ephemeral port and a valid local address to bind the socket.

**Parameters:**bindpoint - the SocketAddress to bind to **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if the bind operation fails, or if the socket is already bound. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if bindpoint is a SocketAddress subclass not supported by this socket**Since:** 1.4 **See Also:**[isBound()](http://docs.google.com/java/net/Socket.html#isBound())

### getInetAddress

public [InetAddress](http://docs.google.com/java/net/InetAddress.html) **getInetAddress**()

Returns the address to which the socket is connected.

**Returns:**the remote IP address to which this socket is connected, or null if the socket is not connected.

### getLocalAddress

public [InetAddress](http://docs.google.com/java/net/InetAddress.html) **getLocalAddress**()

Gets the local address to which the socket is bound.

**Returns:**the local address to which the socket is bound or InetAddress.anyLocalAddress() if the socket is not bound yet.**Since:** JDK1.1

### getPort

public int **getPort**()

Returns the remote port to which this socket is connected.

**Returns:**the remote port number to which this socket is connected, or 0 if the socket is not connected yet.

### getLocalPort

public int **getLocalPort**()

Returns the local port to which this socket is bound.

**Returns:**the local port number to which this socket is bound or -1 if the socket is not bound yet.

### getRemoteSocketAddress

public [SocketAddress](http://docs.google.com/java/net/SocketAddress.html) **getRemoteSocketAddress**()

Returns the address of the endpoint this socket is connected to, or null if it is unconnected.

**Returns:**a SocketAddress reprensenting the remote endpoint of this socket, or null if it is not connected yet.**Since:** 1.4 **See Also:**[getInetAddress()](http://docs.google.com/java/net/Socket.html#getInetAddress()), [getPort()](http://docs.google.com/java/net/Socket.html#getPort()), [connect(SocketAddress, int)](http://docs.google.com/java/net/Socket.html#connect(java.net.SocketAddress,%20int)), [connect(SocketAddress)](http://docs.google.com/java/net/Socket.html#connect(java.net.SocketAddress))

### getLocalSocketAddress

public [SocketAddress](http://docs.google.com/java/net/SocketAddress.html) **getLocalSocketAddress**()

Returns the address of the endpoint this socket is bound to, or null if it is not bound yet.

**Returns:**a SocketAddress representing the local endpoint of this socket, or null if it is not bound yet.**Since:** 1.4 **See Also:**[getLocalAddress()](http://docs.google.com/java/net/Socket.html#getLocalAddress()), [getLocalPort()](http://docs.google.com/java/net/Socket.html#getLocalPort()), [bind(SocketAddress)](http://docs.google.com/java/net/Socket.html#bind(java.net.SocketAddress))

### getChannel

public [SocketChannel](http://docs.google.com/java/nio/channels/SocketChannel.html) **getChannel**()

Returns the unique [SocketChannel](http://docs.google.com/java/nio/channels/SocketChannel.html) object associated with this socket, if any.

A socket will have a channel if, and only if, the channel itself was created via the [SocketChannel.open](http://docs.google.com/java/nio/channels/SocketChannel.html#open()) or [ServerSocketChannel.accept](http://docs.google.com/java/nio/channels/ServerSocketChannel.html#accept()) methods.

**Returns:**the socket channel associated with this socket, or null if this socket was not created for a channel**Since:** 1.4

### getInputStream

public [InputStream](http://docs.google.com/java/io/InputStream.html) **getInputStream**()  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Returns an input stream for this socket.

If this socket has an associated channel then the resulting input stream delegates all of its operations to the channel. If the channel is in non-blocking mode then the input stream's read operations will throw an [IllegalBlockingModeException](http://docs.google.com/java/nio/channels/IllegalBlockingModeException.html).

Under abnormal conditions the underlying connection may be broken by the remote host or the network software (for example a connection reset in the case of TCP connections). When a broken connection is detected by the network software the following applies to the returned input stream :-

* The network software may discard bytes that are buffered by the socket. Bytes that aren't discarded by the network software can be read using [read](http://docs.google.com/java/io/InputStream.html#read()).
* If there are no bytes buffered on the socket, or all buffered bytes have been consumed by [read](http://docs.google.com/java/io/InputStream.html#read()), then all subsequent calls to [read](http://docs.google.com/java/io/InputStream.html#read()) will throw an [IOException](http://docs.google.com/java/io/IOException.html).
* If there are no bytes buffered on the socket, and the socket has not been closed using [close](http://docs.google.com/java/net/Socket.html#close()), then [available](http://docs.google.com/java/io/InputStream.html#available()) will return 0.

Closing the returned [InputStream](http://docs.google.com/java/io/InputStream.html) will close the associated socket.

**Returns:**an input stream for reading bytes from this socket. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs when creating the input stream, the socket is closed, the socket is not connected, or the socket input has been shutdown using [shutdownInput()](http://docs.google.com/java/net/Socket.html#shutdownInput())

### getOutputStream

public [OutputStream](http://docs.google.com/java/io/OutputStream.html) **getOutputStream**()  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Returns an output stream for this socket.

If this socket has an associated channel then the resulting output stream delegates all of its operations to the channel. If the channel is in non-blocking mode then the output stream's write operations will throw an [IllegalBlockingModeException](http://docs.google.com/java/nio/channels/IllegalBlockingModeException.html).

Closing the returned [OutputStream](http://docs.google.com/java/io/OutputStream.html) will close the associated socket.

**Returns:**an output stream for writing bytes to this socket. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs when creating the output stream or if the socket is not connected.

### setTcpNoDelay

public void **setTcpNoDelay**(boolean on)  
 throws [SocketException](http://docs.google.com/java/net/SocketException.html)

Enable/disable TCP\_NODELAY (disable/enable Nagle's algorithm).

**Parameters:**on - true to enable TCP\_NODELAY, false to disable. **Throws:** [SocketException](http://docs.google.com/java/net/SocketException.html) - if there is an error in the underlying protocol, such as a TCP error.**Since:** JDK1.1 **See Also:**[getTcpNoDelay()](http://docs.google.com/java/net/Socket.html#getTcpNoDelay())

### getTcpNoDelay

public boolean **getTcpNoDelay**()  
 throws [SocketException](http://docs.google.com/java/net/SocketException.html)

Tests if TCP\_NODELAY is enabled.

**Returns:**a boolean indicating whether or not TCP\_NODELAY is enabled. **Throws:** [SocketException](http://docs.google.com/java/net/SocketException.html) - if there is an error in the underlying protocol, such as a TCP error.**Since:** JDK1.1 **See Also:**[setTcpNoDelay(boolean)](http://docs.google.com/java/net/Socket.html#setTcpNoDelay(boolean))

### setSoLinger

public void **setSoLinger**(boolean on,  
 int linger)  
 throws [SocketException](http://docs.google.com/java/net/SocketException.html)

Enable/disable SO\_LINGER with the specified linger time in seconds. The maximum timeout value is platform specific. The setting only affects socket close.

**Parameters:**on - whether or not to linger on.linger - how long to linger for, if on is true. **Throws:** [SocketException](http://docs.google.com/java/net/SocketException.html) - if there is an error in the underlying protocol, such as a TCP error. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the linger value is negative.**Since:** JDK1.1 **See Also:**[getSoLinger()](http://docs.google.com/java/net/Socket.html#getSoLinger())

### getSoLinger

public int **getSoLinger**()  
 throws [SocketException](http://docs.google.com/java/net/SocketException.html)

Returns setting for SO\_LINGER. -1 returns implies that the option is disabled. The setting only affects socket close.

**Returns:**the setting for SO\_LINGER. **Throws:** [SocketException](http://docs.google.com/java/net/SocketException.html) - if there is an error in the underlying protocol, such as a TCP error.**Since:** JDK1.1 **See Also:**[setSoLinger(boolean, int)](http://docs.google.com/java/net/Socket.html#setSoLinger(boolean,%20int))

### sendUrgentData

public void **sendUrgentData**(int data)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Send one byte of urgent data on the socket. The byte to be sent is the lowest eight bits of the data parameter. The urgent byte is sent after any preceding writes to the socket OutputStream and before any future writes to the OutputStream.

**Parameters:**data - The byte of data to send **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if there is an error sending the data.**Since:** 1.4

### setOOBInline

public void **setOOBInline**(boolean on)  
 throws [SocketException](http://docs.google.com/java/net/SocketException.html)

Enable/disable OOBINLINE (receipt of TCP urgent data) By default, this option is disabled and TCP urgent data received on a socket is silently discarded. If the user wishes to receive urgent data, then this option must be enabled. When enabled, urgent data is received inline with normal data.

Note, only limited support is provided for handling incoming urgent data. In particular, no notification of incoming urgent data is provided and there is no capability to distinguish between normal data and urgent data unless provided by a higher level protocol.

**Parameters:**on - true to enable OOBINLINE, false to disable. **Throws:** [SocketException](http://docs.google.com/java/net/SocketException.html) - if there is an error in the underlying protocol, such as a TCP error.**Since:** 1.4 **See Also:**[getOOBInline()](http://docs.google.com/java/net/Socket.html#getOOBInline())

### getOOBInline

public boolean **getOOBInline**()  
 throws [SocketException](http://docs.google.com/java/net/SocketException.html)

Tests if OOBINLINE is enabled.

**Returns:**a boolean indicating whether or not OOBINLINE is enabled. **Throws:** [SocketException](http://docs.google.com/java/net/SocketException.html) - if there is an error in the underlying protocol, such as a TCP error.**Since:** 1.4 **See Also:**[setOOBInline(boolean)](http://docs.google.com/java/net/Socket.html#setOOBInline(boolean))

### setSoTimeout

public void **setSoTimeout**(int timeout)  
 throws [SocketException](http://docs.google.com/java/net/SocketException.html)

Enable/disable SO\_TIMEOUT with the specified timeout, in milliseconds. With this option set to a non-zero timeout, a read() call on the InputStream associated with this Socket will block for only this amount of time. If the timeout expires, a **java.net.SocketTimeoutException** is raised, though the Socket is still valid. The option **must** be enabled prior to entering the blocking operation to have effect. The timeout must be > 0. A timeout of zero is interpreted as an infinite timeout.

**Parameters:**timeout - the specified timeout, in milliseconds. **Throws:** [SocketException](http://docs.google.com/java/net/SocketException.html) - if there is an error in the underlying protocol, such as a TCP error.**Since:** JDK 1.1 **See Also:**[getSoTimeout()](http://docs.google.com/java/net/Socket.html#getSoTimeout())

### getSoTimeout

public int **getSoTimeout**()  
 throws [SocketException](http://docs.google.com/java/net/SocketException.html)

Returns setting for SO\_TIMEOUT. 0 returns implies that the option is disabled (i.e., timeout of infinity).

**Returns:**the setting for SO\_TIMEOUT **Throws:** [SocketException](http://docs.google.com/java/net/SocketException.html) - if there is an error in the underlying protocol, such as a TCP error.**Since:** JDK1.1 **See Also:**[setSoTimeout(int)](http://docs.google.com/java/net/Socket.html#setSoTimeout(int))

### setSendBufferSize

public void **setSendBufferSize**(int size)  
 throws [SocketException](http://docs.google.com/java/net/SocketException.html)

Sets the SO\_SNDBUF option to the specified value for this Socket. The SO\_SNDBUF option is used by the platform's networking code as a hint for the size to set the underlying network I/O buffers.

Because SO\_SNDBUF is a hint, applications that want to verify what size the buffers were set to should call [getSendBufferSize()](http://docs.google.com/java/net/Socket.html#getSendBufferSize()).

**Parameters:**size - the size to which to set the send buffer size. This value must be greater than 0. **Throws:** [SocketException](http://docs.google.com/java/net/SocketException.html) - if there is an error in the underlying protocol, such as a TCP error. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the value is 0 or is negative.**Since:** 1.2 **See Also:**[getSendBufferSize()](http://docs.google.com/java/net/Socket.html#getSendBufferSize())

### getSendBufferSize

public int **getSendBufferSize**()  
 throws [SocketException](http://docs.google.com/java/net/SocketException.html)

Get value of the SO\_SNDBUF option for this Socket, that is the buffer size used by the platform for output on this Socket.

**Returns:**the value of the SO\_SNDBUF option for this Socket. **Throws:** [SocketException](http://docs.google.com/java/net/SocketException.html) - if there is an error in the underlying protocol, such as a TCP error.**Since:** 1.2 **See Also:**[setSendBufferSize(int)](http://docs.google.com/java/net/Socket.html#setSendBufferSize(int))

### setReceiveBufferSize

public void **setReceiveBufferSize**(int size)  
 throws [SocketException](http://docs.google.com/java/net/SocketException.html)

Sets the SO\_RCVBUF option to the specified value for this Socket. The SO\_RCVBUF option is used by the platform's networking code as a hint for the size to set the underlying network I/O buffers.

Increasing the receive buffer size can increase the performance of network I/O for high-volume connection, while decreasing it can help reduce the backlog of incoming data.

Because SO\_RCVBUF is a hint, applications that want to verify what size the buffers were set to should call [getReceiveBufferSize()](http://docs.google.com/java/net/Socket.html#getReceiveBufferSize()).

The value of SO\_RCVBUF is also used to set the TCP receive window that is advertized to the remote peer. Generally, the window size can be modified at any time when a socket is connected. However, if a receive window larger than 64K is required then this must be requested **before** the socket is connected to the remote peer. There are two cases to be aware of:

1. For sockets accepted from a ServerSocket, this must be done by calling [ServerSocket.setReceiveBufferSize(int)](http://docs.google.com/java/net/ServerSocket.html#setReceiveBufferSize(int)) before the ServerSocket is bound to a local address.
2. For client sockets, setReceiveBufferSize() must be called before connecting the socket to its remote peer.

**Parameters:**size - the size to which to set the receive buffer size. This value must be greater than 0. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the value is 0 or is negative. [SocketException](http://docs.google.com/java/net/SocketException.html) - if there is an error in the underlying protocol, such as a TCP error.**Since:** 1.2 **See Also:**[getReceiveBufferSize()](http://docs.google.com/java/net/Socket.html#getReceiveBufferSize()), [ServerSocket.setReceiveBufferSize(int)](http://docs.google.com/java/net/ServerSocket.html#setReceiveBufferSize(int))

### getReceiveBufferSize

public int **getReceiveBufferSize**()  
 throws [SocketException](http://docs.google.com/java/net/SocketException.html)

Gets the value of the SO\_RCVBUF option for this Socket, that is the buffer size used by the platform for input on this Socket.

**Returns:**the value of the SO\_RCVBUF option for this Socket. **Throws:** [SocketException](http://docs.google.com/java/net/SocketException.html) - if there is an error in the underlying protocol, such as a TCP error.**Since:** 1.2 **See Also:**[setReceiveBufferSize(int)](http://docs.google.com/java/net/Socket.html#setReceiveBufferSize(int))

### setKeepAlive

public void **setKeepAlive**(boolean on)  
 throws [SocketException](http://docs.google.com/java/net/SocketException.html)

Enable/disable SO\_KEEPALIVE.

**Parameters:**on - whether or not to have socket keep alive turned on. **Throws:** [SocketException](http://docs.google.com/java/net/SocketException.html) - if there is an error in the underlying protocol, such as a TCP error.**Since:** 1.3 **See Also:**[getKeepAlive()](http://docs.google.com/java/net/Socket.html#getKeepAlive())

### getKeepAlive

public boolean **getKeepAlive**()  
 throws [SocketException](http://docs.google.com/java/net/SocketException.html)

Tests if SO\_KEEPALIVE is enabled.

**Returns:**a boolean indicating whether or not SO\_KEEPALIVE is enabled. **Throws:** [SocketException](http://docs.google.com/java/net/SocketException.html) - if there is an error in the underlying protocol, such as a TCP error.**Since:** 1.3 **See Also:**[setKeepAlive(boolean)](http://docs.google.com/java/net/Socket.html#setKeepAlive(boolean))

### setTrafficClass

public void **setTrafficClass**(int tc)  
 throws [SocketException](http://docs.google.com/java/net/SocketException.html)

Sets traffic class or type-of-service octet in the IP header for packets sent from this Socket. As the underlying network implementation may ignore this value applications should consider it a hint.

The tc **must** be in the range 0 <= tc <= 255 or an IllegalArgumentException will be thrown.

Notes:

For Internet Protocol v4 the value consists of an octet with precedence and TOS fields as detailed in RFC 1349. The TOS field is bitset created by bitwise-or'ing values such the following :-

* IPTOS\_LOWCOST (0x02)
* IPTOS\_RELIABILITY (0x04)
* IPTOS\_THROUGHPUT (0x08)
* IPTOS\_LOWDELAY (0x10)

The last low order bit is always ignored as this corresponds to the MBZ (must be zero) bit.

Setting bits in the precedence field may result in a SocketException indicating that the operation is not permitted.

As RFC 1122 section 4.2.4.2 indicates, a compliant TCP implementation should, but is not required to, let application change the TOS field during the lifetime of a connection. So whether the type-of-service field can be changed after the TCP connection has been established depends on the implementation in the underlying platform. Applications should not assume that they can change the TOS field after the connection.

For Internet Protocol v6 tc is the value that would be placed into the sin6\_flowinfo field of the IP header.

**Parameters:**tc - an int value for the bitset. **Throws:** [SocketException](http://docs.google.com/java/net/SocketException.html) - if there is an error setting the traffic class or type-of-service**Since:** 1.4 **See Also:**[getTrafficClass()](http://docs.google.com/java/net/Socket.html#getTrafficClass())

### getTrafficClass

public int **getTrafficClass**()  
 throws [SocketException](http://docs.google.com/java/net/SocketException.html)

Gets traffic class or type-of-service in the IP header for packets sent from this Socket

As the underlying network implementation may ignore the traffic class or type-of-service set using [setTrafficClass(int)](http://docs.google.com/java/net/Socket.html#setTrafficClass(int)) this method may return a different value than was previously set using the [setTrafficClass(int)](http://docs.google.com/java/net/Socket.html#setTrafficClass(int)) method on this Socket.

**Returns:**the traffic class or type-of-service already set **Throws:** [SocketException](http://docs.google.com/java/net/SocketException.html) - if there is an error obtaining the traffic class or type-of-service value.**Since:** 1.4 **See Also:**[setTrafficClass(int)](http://docs.google.com/java/net/Socket.html#setTrafficClass(int))

### setReuseAddress

public void **setReuseAddress**(boolean on)  
 throws [SocketException](http://docs.google.com/java/net/SocketException.html)

Enable/disable the SO\_REUSEADDR socket option.

When a TCP connection is closed the connection may remain in a timeout state for a period of time after the connection is closed (typically known as the TIME\_WAIT state or 2MSL wait state). For applications using a well known socket address or port it may not be possible to bind a socket to the required SocketAddress if there is a connection in the timeout state involving the socket address or port.

Enabling SO\_REUSEADDR prior to binding the socket using [bind(SocketAddress)](http://docs.google.com/java/net/Socket.html#bind(java.net.SocketAddress)) allows the socket to be bound even though a previous connection is in a timeout state.

When a Socket is created the initial setting of SO\_REUSEADDR is disabled.

The behaviour when SO\_REUSEADDR is enabled or disabled after a socket is bound (See [isBound()](http://docs.google.com/java/net/Socket.html#isBound())) is not defined.

**Parameters:**on - whether to enable or disable the socket option **Throws:** [SocketException](http://docs.google.com/java/net/SocketException.html) - if an error occurs enabling or disabling the SO\_RESUEADDR socket option, or the socket is closed.**Since:** 1.4 **See Also:**[getReuseAddress()](http://docs.google.com/java/net/Socket.html#getReuseAddress()), [bind(SocketAddress)](http://docs.google.com/java/net/Socket.html#bind(java.net.SocketAddress)), [isClosed()](http://docs.google.com/java/net/Socket.html#isClosed()), [isBound()](http://docs.google.com/java/net/Socket.html#isBound())

### getReuseAddress

public boolean **getReuseAddress**()  
 throws [SocketException](http://docs.google.com/java/net/SocketException.html)

Tests if SO\_REUSEADDR is enabled.

**Returns:**a boolean indicating whether or not SO\_REUSEADDR is enabled. **Throws:** [SocketException](http://docs.google.com/java/net/SocketException.html) - if there is an error in the underlying protocol, such as a TCP error.**Since:** 1.4 **See Also:**[setReuseAddress(boolean)](http://docs.google.com/java/net/Socket.html#setReuseAddress(boolean))

### close

public void **close**()  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Closes this socket.

Any thread currently blocked in an I/O operation upon this socket will throw a [SocketException](http://docs.google.com/java/net/SocketException.html).

Once a socket has been closed, it is not available for further networking use (i.e. can't be reconnected or rebound). A new socket needs to be created.

Closing this socket will also close the socket's [InputStream](http://docs.google.com/java/io/InputStream.html) and [OutputStream](http://docs.google.com/java/io/OutputStream.html).

If this socket has an associated channel then the channel is closed as well.

**Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs when closing this socket.**See Also:**[isClosed()](http://docs.google.com/java/net/Socket.html#isClosed())

### shutdownInput

public void **shutdownInput**()  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Places the input stream for this socket at "end of stream". Any data sent to the input stream side of the socket is acknowledged and then silently discarded.

If you read from a socket input stream after invoking shutdownInput() on the socket, the stream will return EOF.

**Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs when shutting down this socket.**Since:** 1.3 **See Also:**[shutdownOutput()](http://docs.google.com/java/net/Socket.html#shutdownOutput()), [close()](http://docs.google.com/java/net/Socket.html#close()), [setSoLinger(boolean, int)](http://docs.google.com/java/net/Socket.html#setSoLinger(boolean,%20int)), [isInputShutdown()](http://docs.google.com/java/net/Socket.html#isInputShutdown())

### shutdownOutput

public void **shutdownOutput**()  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Disables the output stream for this socket. For a TCP socket, any previously written data will be sent followed by TCP's normal connection termination sequence. If you write to a socket output stream after invoking shutdownOutput() on the socket, the stream will throw an IOException.

**Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs when shutting down this socket.**Since:** 1.3 **See Also:**[shutdownInput()](http://docs.google.com/java/net/Socket.html#shutdownInput()), [close()](http://docs.google.com/java/net/Socket.html#close()), [setSoLinger(boolean, int)](http://docs.google.com/java/net/Socket.html#setSoLinger(boolean,%20int)), [isOutputShutdown()](http://docs.google.com/java/net/Socket.html#isOutputShutdown())

### toString

public [String](http://docs.google.com/java/lang/String.html) **toString**()

Converts this socket to a String.

**Overrides:**[toString](http://docs.google.com/java/lang/Object.html#toString()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**a string representation of this socket.

### isConnected

public boolean **isConnected**()

Returns the connection state of the socket.

**Returns:**true if the socket successfuly connected to a server**Since:** 1.4

### isBound

public boolean **isBound**()

Returns the binding state of the socket.

**Returns:**true if the socket successfuly bound to an address**Since:** 1.4 **See Also:**[bind(java.net.SocketAddress)](http://docs.google.com/java/net/Socket.html#bind(java.net.SocketAddress))

### isClosed

public boolean **isClosed**()

Returns the closed state of the socket.

**Returns:**true if the socket has been closed**Since:** 1.4 **See Also:**[close()](http://docs.google.com/java/net/Socket.html#close())

### isInputShutdown

public boolean **isInputShutdown**()

Returns whether the read-half of the socket connection is closed.

**Returns:**true if the input of the socket has been shutdown**Since:** 1.4 **See Also:**[shutdownInput()](http://docs.google.com/java/net/Socket.html#shutdownInput())

### isOutputShutdown

public boolean **isOutputShutdown**()

Returns whether the write-half of the socket connection is closed.

**Returns:**true if the output of the socket has been shutdown**Since:** 1.4 **See Also:**[shutdownOutput()](http://docs.google.com/java/net/Socket.html#shutdownOutput())

### setSocketImplFactory

public static void **setSocketImplFactory**([SocketImplFactory](http://docs.google.com/java/net/SocketImplFactory.html) fac)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Sets the client socket implementation factory for the application. The factory can be specified only once.

When an application creates a new client socket, the socket implementation factory's createSocketImpl method is called to create the actual socket implementation.

Passing null to the method is a no-op unless the factory was already set.

If there is a security manager, this method first calls the security manager's checkSetFactory method to ensure the operation is allowed. This could result in a SecurityException.

**Parameters:**fac - the desired factory. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs when setting the socket factory. [SocketException](http://docs.google.com/java/net/SocketException.html) - if the factory is already defined. [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and its checkSetFactory method doesn't allow the operation.**See Also:**[SocketImplFactory.createSocketImpl()](http://docs.google.com/java/net/SocketImplFactory.html#createSocketImpl()), [SecurityManager.checkSetFactory()](http://docs.google.com/java/lang/SecurityManager.html#checkSetFactory())

### setPerformancePreferences

public void **setPerformancePreferences**(int connectionTime,  
 int latency,  
 int bandwidth)

Sets performance preferences for this socket.

Sockets use the TCP/IP protocol by default. Some implementations may offer alternative protocols which have different performance characteristics than TCP/IP. This method allows the application to express its own preferences as to how these tradeoffs should be made when the implementation chooses from the available protocols.

Performance preferences are described by three integers whose values indicate the relative importance of short connection time, low latency, and high bandwidth. The absolute values of the integers are irrelevant; in order to choose a protocol the values are simply compared, with larger values indicating stronger preferences. Negative values represent a lower priority than positive values. If the application prefers short connection time over both low latency and high bandwidth, for example, then it could invoke this method with the values (1, 0, 0). If the application prefers high bandwidth above low latency, and low latency above short connection time, then it could invoke this method with the values (0, 1, 2).

Invoking this method after this socket has been connected will have no effect.

**Parameters:**connectionTime - An int expressing the relative importance of a short connection timelatency - An int expressing the relative importance of low latencybandwidth - An int expressing the relative importance of high bandwidth**Since:** 1.5

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Socket.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/net/ServerSocket.html)   [**NEXT CLASS**](http://docs.google.com/java/net/SocketAddress.html) | [**FRAMES**](http://docs.google.com/index.html?java/net/Socket.html)    [**NO FRAMES**](http://docs.google.com/Socket.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#44sinio) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).