| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ServerSocket.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/SecureCacheResponse.html)   [**NEXT CLASS**](http://docs.google.com/java/net/Socket.html) | [**FRAMES**](http://docs.google.com/index.html?java/net/ServerSocket.html)    [**NO FRAMES**](http://docs.google.com/ServerSocket.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#3rdcrjn) |

## **java.net**

Class ServerSocket

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

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

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

This class implements server sockets. A server socket waits for requests to come in over the network. It performs some operation based on that request, and then possibly returns a result to the requester.

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

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

| **Constructor Summary** | |
| --- | --- |
| [**ServerSocket**](http://docs.google.com/java/net/ServerSocket.html#ServerSocket())()            Creates an unbound server socket. |
| [**ServerSocket**](http://docs.google.com/java/net/ServerSocket.html#ServerSocket(int))(int port)            Creates a server socket, bound to the specified port. |
| [**ServerSocket**](http://docs.google.com/java/net/ServerSocket.html#ServerSocket(int,%20int))(int port, int backlog)            Creates a server socket and binds it to the specified local port number, with the specified backlog. |
| [**ServerSocket**](http://docs.google.com/java/net/ServerSocket.html#ServerSocket(int,%20int,%20java.net.InetAddress))(int port, int backlog, [InetAddress](http://docs.google.com/java/net/InetAddress.html) bindAddr)            Create a server with the specified port, listen backlog, and local IP address to bind to. |

| **Method Summary** | |
| --- | --- |
| [Socket](http://docs.google.com/java/net/Socket.html) | [**accept**](http://docs.google.com/java/net/ServerSocket.html#accept())()            Listens for a connection to be made to this socket and accepts it. |
| void | [**bind**](http://docs.google.com/java/net/ServerSocket.html#bind(java.net.SocketAddress))([SocketAddress](http://docs.google.com/java/net/SocketAddress.html) endpoint)            Binds the ServerSocket to a specific address (IP address and port number). |
| void | [**bind**](http://docs.google.com/java/net/ServerSocket.html#bind(java.net.SocketAddress,%20int))([SocketAddress](http://docs.google.com/java/net/SocketAddress.html) endpoint, int backlog)            Binds the ServerSocket to a specific address (IP address and port number). |
| void | [**close**](http://docs.google.com/java/net/ServerSocket.html#close())()            Closes this socket. |
| [ServerSocketChannel](http://docs.google.com/java/nio/channels/ServerSocketChannel.html) | [**getChannel**](http://docs.google.com/java/net/ServerSocket.html#getChannel())()            Returns the unique [ServerSocketChannel](http://docs.google.com/java/nio/channels/ServerSocketChannel.html) object associated with this socket, if any. |
| [InetAddress](http://docs.google.com/java/net/InetAddress.html) | [**getInetAddress**](http://docs.google.com/java/net/ServerSocket.html#getInetAddress())()            Returns the local address of this server socket. |
| int | [**getLocalPort**](http://docs.google.com/java/net/ServerSocket.html#getLocalPort())()            Returns the port on which this socket is listening. |
| [SocketAddress](http://docs.google.com/java/net/SocketAddress.html) | [**getLocalSocketAddress**](http://docs.google.com/java/net/ServerSocket.html#getLocalSocketAddress())()            Returns the address of the endpoint this socket is bound to, or null if it is not bound yet. |
| int | [**getReceiveBufferSize**](http://docs.google.com/java/net/ServerSocket.html#getReceiveBufferSize())()            Gets the value of the SO\_RCVBUF option for this ServerSocket, that is the proposed buffer size that will be used for Sockets accepted from this ServerSocket. |
| boolean | [**getReuseAddress**](http://docs.google.com/java/net/ServerSocket.html#getReuseAddress())()            Tests if SO\_REUSEADDR is enabled. |
| int | [**getSoTimeout**](http://docs.google.com/java/net/ServerSocket.html#getSoTimeout())()            Retrieve setting for SO\_TIMEOUT. |
| protected  void | [**implAccept**](http://docs.google.com/java/net/ServerSocket.html#implAccept(java.net.Socket))([Socket](http://docs.google.com/java/net/Socket.html) s)            Subclasses of ServerSocket use this method to override accept() to return their own subclass of socket. |
| boolean | [**isBound**](http://docs.google.com/java/net/ServerSocket.html#isBound())()            Returns the binding state of the ServerSocket. |
| boolean | [**isClosed**](http://docs.google.com/java/net/ServerSocket.html#isClosed())()            Returns the closed state of the ServerSocket. |
| void | [**setPerformancePreferences**](http://docs.google.com/java/net/ServerSocket.html#setPerformancePreferences(int,%20int,%20int))(int connectionTime, int latency, int bandwidth)            Sets performance preferences for this ServerSocket. |
| void | [**setReceiveBufferSize**](http://docs.google.com/java/net/ServerSocket.html#setReceiveBufferSize(int))(int size)            Sets a default proposed value for the SO\_RCVBUF option for sockets accepted from this ServerSocket. |
| void | [**setReuseAddress**](http://docs.google.com/java/net/ServerSocket.html#setReuseAddress(boolean))(boolean on)            Enable/disable the SO\_REUSEADDR socket option. |
| static void | [**setSocketFactory**](http://docs.google.com/java/net/ServerSocket.html#setSocketFactory(java.net.SocketImplFactory))([SocketImplFactory](http://docs.google.com/java/net/SocketImplFactory.html) fac)            Sets the server socket implementation factory for the application. |
| void | [**setSoTimeout**](http://docs.google.com/java/net/ServerSocket.html#setSoTimeout(int))(int timeout)            Enable/disable SO\_TIMEOUT with the specified timeout, in milliseconds. |
| [String](http://docs.google.com/java/lang/String.html) | [**toString**](http://docs.google.com/java/net/ServerSocket.html#toString())()            Returns the implementation address and implementation port of this socket as 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** |
| --- |

### ServerSocket

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

Creates an unbound server socket.

**Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - IO error when opening the socket.

### ServerSocket

public **ServerSocket**(int port)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Creates a server socket, bound to the specified port. A port of 0 creates a socket on any free port.

The maximum queue length for incoming connection indications (a request to connect) is set to 50. If a connection indication arrives when the queue is full, the connection is refused.

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 checkListen method is called with the port argument as its argument to ensure the operation is allowed. This could result in a SecurityException.

**Parameters:**port - the port number, or 0 to use any free port. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs when opening the socket. [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and its checkListen method doesn't allow the operation.**See Also:**[SocketImpl](http://docs.google.com/java/net/SocketImpl.html), [SocketImplFactory.createSocketImpl()](http://docs.google.com/java/net/SocketImplFactory.html#createSocketImpl()), [setSocketFactory(java.net.SocketImplFactory)](http://docs.google.com/java/net/ServerSocket.html#setSocketFactory(java.net.SocketImplFactory)), [SecurityManager.checkListen(int)](http://docs.google.com/java/lang/SecurityManager.html#checkListen(int))

### ServerSocket

public **ServerSocket**(int port,  
 int backlog)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Creates a server socket and binds it to the specified local port number, with the specified backlog. A port number of 0 creates a socket on any free port.

The maximum queue length for incoming connection indications (a request to connect) is set to the backlog parameter. If a connection indication arrives when the queue is full, the connection is refused.

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 checkListen method is called with the port argument as its argument to ensure the operation is allowed. This could result in a SecurityException.

The backlog argument must be a positive value greater than 0. If the value passed if equal or less than 0, then the default value will be assumed.

**Parameters:**port - the specified port, or 0 to use any free port.backlog - the maximum length of the queue. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs when opening the socket. [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and its checkListen method doesn't allow the operation.**See Also:**[SocketImpl](http://docs.google.com/java/net/SocketImpl.html), [SocketImplFactory.createSocketImpl()](http://docs.google.com/java/net/SocketImplFactory.html#createSocketImpl()), [setSocketFactory(java.net.SocketImplFactory)](http://docs.google.com/java/net/ServerSocket.html#setSocketFactory(java.net.SocketImplFactory)), [SecurityManager.checkListen(int)](http://docs.google.com/java/lang/SecurityManager.html#checkListen(int))

### ServerSocket

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

Create a server with the specified port, listen backlog, and local IP address to bind to. The *bindAddr* argument can be used on a multi-homed host for a ServerSocket that will only accept connect requests to one of its addresses. If *bindAddr* is null, it will default accepting connections on any/all local addresses. The port must be between 0 and 65535, inclusive.

If there is a security manager, this method calls its checkListen method with the port argument as its argument to ensure the operation is allowed. This could result in a SecurityException.

The backlog argument must be a positive value greater than 0. If the value passed if equal or less than 0, then the default value will be assumed.

**Parameters:**port - the local TCP portbacklog - the listen backlogbindAddr - the local InetAddress the server will bind to **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and its checkListen method doesn't allow the operation. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs when opening the socket.**Since:** JDK1.1 **See Also:**[SocketOptions](http://docs.google.com/java/net/SocketOptions.html), [SocketImpl](http://docs.google.com/java/net/SocketImpl.html), [SecurityManager.checkListen(int)](http://docs.google.com/java/lang/SecurityManager.html#checkListen(int))

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

### bind

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

Binds the ServerSocket to a specific address (IP address and port number).

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

**Parameters:**endpoint - The IP address & port number 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. [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a SecurityManager is present and its checkListen method doesn't allow the operation. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if endpoint 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) endpoint,  
 int backlog)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Binds the ServerSocket to a specific address (IP address and port number).

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

The backlog argument must be a positive value greater than 0. If the value passed if equal or less than 0, then the default value will be assumed.

**Parameters:**endpoint - The IP address & port number to bind to.backlog - The listen backlog length. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if the bind operation fails, or if the socket is already bound. [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a SecurityManager is present and its checkListen method doesn't allow the operation. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if endpoint is a SocketAddress subclass not supported by this socket**Since:** 1.4

### getInetAddress

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

Returns the local address of this server socket.

**Returns:**the address to which this socket is bound, or null if the socket is unbound.

### getLocalPort

public int **getLocalPort**()

Returns the port on which this socket is listening.

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

### 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:**[getInetAddress()](http://docs.google.com/java/net/ServerSocket.html#getInetAddress()), [getLocalPort()](http://docs.google.com/java/net/ServerSocket.html#getLocalPort()), [bind(SocketAddress)](http://docs.google.com/java/net/ServerSocket.html#bind(java.net.SocketAddress))

### accept

public [Socket](http://docs.google.com/java/net/Socket.html) **accept**()  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Listens for a connection to be made to this socket and accepts it. The method blocks until a connection is made.

A new Socket s is created and, if there is a security manager, the security manager's checkAccept method is called with s.getInetAddress().getHostAddress() and s.getPort() as its arguments to ensure the operation is allowed. This could result in a SecurityException.

**Returns:**the new Socket **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs when waiting for a connection. [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and its checkAccept method doesn't allow the operation. [SocketTimeoutException](http://docs.google.com/java/net/SocketTimeoutException.html) - if a timeout was previously set with setSoTimeout and the timeout has been reached. [IllegalBlockingModeException](http://docs.google.com/java/nio/channels/IllegalBlockingModeException.html) - if this socket has an associated channel, the channel is in non-blocking mode, and there is no connection ready to be accepted**See Also:**[SecurityManager.checkAccept(java.lang.String, int)](http://docs.google.com/java/lang/SecurityManager.html#checkAccept(java.lang.String,%20int))

### implAccept

protected final void **implAccept**([Socket](http://docs.google.com/java/net/Socket.html) s)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Subclasses of ServerSocket use this method to override accept() to return their own subclass of socket. So a FooServerSocket will typically hand this method an *empty* FooSocket. On return from implAccept the FooSocket will be connected to a client.

**Parameters:**s - the Socket **Throws:** [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 [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs when waiting for a connection.**Since:** JDK1.1

### close

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

Closes this socket. Any thread currently blocked in [accept()](http://docs.google.com/java/net/ServerSocket.html#accept()) will throw a [SocketException](http://docs.google.com/java/net/SocketException.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 the socket.

### getChannel

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

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

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

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

### isBound

public boolean **isBound**()

Returns the binding state of the ServerSocket.

**Returns:**true if the ServerSocket succesfuly bound to an address**Since:** 1.4

### isClosed

public boolean **isClosed**()

Returns the closed state of the ServerSocket.

**Returns:**true if the socket has been closed**Since:** 1.4

### 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 call to accept() for this ServerSocket will block for only this amount of time. If the timeout expires, a **java.net.SocketTimeoutException** is raised, though the ServerSocket 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:** JDK1.1 **See Also:**[getSoTimeout()](http://docs.google.com/java/net/ServerSocket.html#getSoTimeout())

### getSoTimeout

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

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

**Returns:**the SO\_TIMEOUT value **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs**Since:** JDK1.1 **See Also:**[setSoTimeout(int)](http://docs.google.com/java/net/ServerSocket.html#setSoTimeout(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/ServerSocket.html#bind(java.net.SocketAddress)) allows the socket to be bound even though a previous connection is in a timeout state.

When a ServerSocket is created the initial setting of SO\_REUSEADDR is not defined. Applications can use [getReuseAddress()](http://docs.google.com/java/net/ServerSocket.html#getReuseAddress()) to determine the initial setting of SO\_REUSEADDR.

The behaviour when SO\_REUSEADDR is enabled or disabled after a socket is bound (See [isBound()](http://docs.google.com/java/net/ServerSocket.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/ServerSocket.html#getReuseAddress()), [bind(SocketAddress)](http://docs.google.com/java/net/ServerSocket.html#bind(java.net.SocketAddress)), [isBound()](http://docs.google.com/java/net/ServerSocket.html#isBound()), [isClosed()](http://docs.google.com/java/net/ServerSocket.html#isClosed())

### 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/ServerSocket.html#setReuseAddress(boolean))

### toString

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

Returns the implementation address and implementation port of this socket as 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.

### setSocketFactory

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

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

When an application creates a new server 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 has already been 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())

### setReceiveBufferSize

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

Sets a default proposed value for the SO\_RCVBUF option for sockets accepted from this ServerSocket. The value actually set in the accepted socket must be determined by calling [Socket.getReceiveBufferSize()](http://docs.google.com/java/net/Socket.html#getReceiveBufferSize()) after the socket is returned by [accept()](http://docs.google.com/java/net/ServerSocket.html#accept()).

The value of SO\_RCVBUF is used both to set the size of the internal socket receive buffer, and to set the size of the TCP receive window that is advertized to the remote peer.

It is possible to change the value subsequently, by calling [Socket.setReceiveBufferSize(int)](http://docs.google.com/java/net/Socket.html#setReceiveBufferSize(int)). However, if the application wishes to allow a receive window larger than 64K bytes, as defined by RFC1323 then the proposed value must be set in the ServerSocket **before** it is bound to a local address. This implies, that the ServerSocket must be created with the no-argument constructor, then setReceiveBufferSize() must be called and lastly the ServerSocket is bound to an address by calling bind().

Failure to do this will not cause an error, and the buffer size may be set to the requested value but the TCP receive window in sockets accepted from this ServerSocket will be no larger than 64K bytes.

**Parameters:**size - the size to which to set the receive 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:** 1.4 **See Also:**[getReceiveBufferSize()](http://docs.google.com/java/net/ServerSocket.html#getReceiveBufferSize())

### getReceiveBufferSize

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

Gets the value of the SO\_RCVBUF option for this ServerSocket, that is the proposed buffer size that will be used for Sockets accepted from this ServerSocket.

Note, the value actually set in the accepted socket is determined by calling [Socket.getReceiveBufferSize()](http://docs.google.com/java/net/Socket.html#getReceiveBufferSize()).

**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:** 1.4 **See Also:**[setReceiveBufferSize(int)](http://docs.google.com/java/net/ServerSocket.html#setReceiveBufferSize(int))

### setPerformancePreferences

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

Sets performance preferences for this ServerSocket.

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. 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 bound will have no effect. This implies that in order to use this capability requires the socket to be created with the no-argument constructor.

**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/ServerSocket.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/SecureCacheResponse.html)   [**NEXT CLASS**](http://docs.google.com/java/net/Socket.html) | [**FRAMES**](http://docs.google.com/index.html?java/net/ServerSocket.html)    [**NO FRAMES**](http://docs.google.com/ServerSocket.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#3rdcrjn) |

[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).