# Object Classes

APPENDIX

# D

## IN THIS APPENDIX

This appendix lists the classes defined in the Java API and the custom C++ library described in the book and on this book's Web site. Each class has one of three designations: *Class* (a normal class that you can instantiate), *Abstract Class* (a class that defines a skeleton for derived classes), and *Superclass* (an instantiable parent class).

# C++ Exceptions

This section describes the classes for exception classes defined in the book. The hierarchy holds to the philosophy of doing as little as possible so as to minimize the potential of catastrophic class creations.

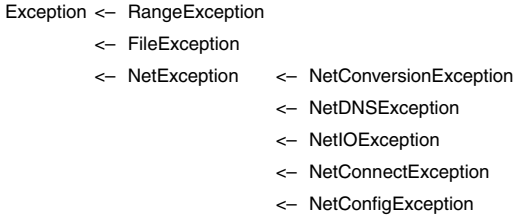


**FIGURE D.1**
*The C++ exceptions class hierarchy.*

## `Exception` (Superclass)

Constructor:

```
Exception(SimpleString s);
```

General Description: Generic exception message with `SimpleString` type.

Method:

| | |
|---|---|
| `const char* GetString()` | Retrieve the string message. |

Child Exceptions:

| | |
|---|---|
| `RangeException` | Any range exception. Used by `MessageGroup` class. |
| `FileException` | Any file exception. Used by `Socket` class. |

## `NetException` (Class)

Constructor:

```
NetException(SimpleString s);
```

General Description: Generic network exception.

Parent Class: `Exception`

Child Exceptions:

| | |
|---|---|
| `NetConversionException` | Host (`inet_ntop/inet_pton`) address conversion exception. Used by `HostAddress` class. |
| `NetDNSException` | Could not resolve hostname exception. Used by `HostAddress`. |
| `NetIOException` | `send()/recv()` exception. Used by `Socket`. |
| `NetConnectException` | Exception when trying to use `bind()`, `connect()`, `listen()`, or `accept()`. Used by `ServerSocket`, `ClientSocket`, and `MessageGroup`. |
| `NetConfigException` | Exception when trying to set or get socket option. Used by all `Socket` classes. |

# C++ Support Classes

This section describes several classes that are related to the framework but are simpler than those of other class libraries. Feel free to replace these as desired with the standard C++ class libraries.



## `SimpleString` (Class)

Constructor:

```
SimpleString(const char* s);
SimpleString(const SimpleString& s);
```

General Description: Very simple and lightweight string type.

Methods:

| | |
|---|---|
| `+(char *)+(Simplestring& )` | Append string to current instance. |
| `const char* GetString()` | Retrieve the string message. |

Exceptions Thrown: (none)

## `HostAddress` (Class)

Constructor:

```
HostAddress(const char* Name=0, ENetwork Network=eIPv4);
HostAddress(HostAddress& Address);
```

General Description: Class to manage host identification.

Methods:

| | |
|---|---|
| `void SetPort(int Port);` | Set the port number. |
| `int GetPort(void) const;` | Get the port number. |
| `ENetwork GetNetwork(void) const;` | Get the network type. |
| `struct sockaddr* GetAddress(void) const;` | Get the actual socket address. |
| `int GetSize(void) const;` | Get socket address size. |
| `int ==(HostAddress& Address) const;` | Compare if equal. |
| `int !=(HostAddress& Address) const;` | Compare if not equal. |
| `const char* GetHost(bool byName=1);` | Retrieve the hostname. |

Exceptions Thrown:

```
Exception
NetConversionException
NetDNSException
```

# C++ Messaging Classes

This class hierarchy lets you define classes that self-package and self-unpackage the internal data. While not as simple or direct as the hierarchy in Java, it's simple and direct.

## `Message` (Abstract Class)

Constructor: (none)

General Description: Message pattern for creating a specific message to send and receive.

Methods:

| | |
|---|---|
| `virtual char* Wrap(int& Bytes) const;` | Interface for packaging object. |
| `bool Unwrap(char* package, int Bytes, int MsgNum);` | Interface for unpackaging object. |

Exceptions Thrown: (none)

## `TextMessage` (Class)

Constructor: (none)

General Description: Message pattern for creating a specific message to send and receive.

Parent Class: `Message`

Methods:

| | |
|---|---|
| `=(const char* str);`<br>`=(const TextMessage& s);` | Assign a new string to object. |
| `+=(const char* str);`<br>`+=(const TextMessage& s);` | Append string to object. |
| `const char* GetBuffer(void) const;` | Get the text. |
| `char* Wrap(int& Bytes) const;` | Wrap object to send. |
| `bool Unwrap(char* package, int Bytes, int MsgNum);` | Unwrap received object. |
| `GetSize(void) const;` | Get length of string. |
| `void SetSize(int Bytes);` | Set string length. |
| `int GetAvailable(void) const;` | Get available bytes in buffer. |

Exceptions Thrown: (none)

# C++ Socket Classes



This hierarchy defines the classes that make up the socket interfaces. It contains effectively five different classes that you can instantiate: `SocketServer`, `SocketClient`, `Datagram`, `Broadcast`, and `MessageGroup`. You can easily expand this hierarchy with OpenSSL classes (`SSLServer` and `SSLClient`).

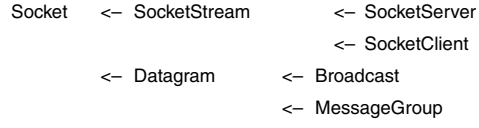


**FIGURE D.2**
*The C++ `Socket` class hierarchy.*

## `Socket` (Superclass)

Constructor:

```
Socket(void);
Socket(int sd);
Socket(ENetwork Network, EProtocol Protocol);
Socket(Socket& sock);
```

General Description: General socket class, not intended for direct instantiation.

Methods:

| | |
|---|---|
| `void Bind(HostAddress& Addr);` | Bind socket to port/interface. |
| `void CloseInput(void) const;` | Close input stream. |
| `void CloseOutput(void) const;` | Close output stream. |
| `int Send(Message& Msg, int Options=0) const;` | Send message to connected site. |
| `int Send(HostAddress& Addr, Message& Msg, int Options=0) const;` | Send directed message. |
| `int Receive(Message& Msg, int Options=0) const;` | Receive message from connection. |
| `int Receive(HostAddress& Addr, Message& Msg, int Options=0) const;` | Receive directed message. |
| `void PermitRoute(bool Setting);` | Allow routable packets. |
| `void KeepAlive(bool Setting);` | Keep connection alive. |
| `void ShareAddress(bool Setting);` | Share port/interface address. |
| `int GetReceiveSize(void);`<br>`void SetReceiveSize(int Bytes);` | Get/set receive buffer size. |
| `int GetSendSize(void);`<br>`void SetSendSize(int Bytes);` | Get/set send buffer size. |
| `int GetMinReceive(void);`<br>`void SetMinReceive(int Bytes);` | Get/set minimum watermark for `SIGIO` receive signal. |
| `int GetMinSend(void);`<br>`void SetMinSend(int Bytes);` | Get/set minimum watermark for `SIGIO` send signal. |
| `struct timeval GetReceiveTimeout(void);`<br>`void SetReceiveTimeout(struct timeval& val);` | Get/set time before aborting a receive. |
| `struct timeval GetSendTimeout(void);`<br>`void SetSendTimeout(struct timeval& val);` | Get/set time before aborting a send. |
| `ENetwork GetType(void);` | Get the socket type (network). |
| `virtual int  GetTTL(void);`<br>`virtual void SetTTL(int Hops);` | Get/set the time-to-live. |
| `int GetError(void);` | Get any pending errors. |

Exceptions Thrown:

`NetException`

`FileException`

```
NetConnectException
NetIOException
NetConfigException
```

## SocketStream (Class)

Constructor:

```
SocketStream(void);
SocketStream(int sd);
SocketStream(ENetwork Network);
SocketStream(SocketStream& sock);
```

General Description: Streaming (`SOCK_STREAM`) socket.

Parent Class: `Socket`

Methods:

| | |
|---|---|
| `int GetMaxSegmentSize(void);`<br>`void SetMaxSegmentSize(short Bytes);` | Get/set the segment size (MSS). |
| `void DontDelay(bool Setting);` | Enable/disable Nagle algorithm. |

Exception Thrown:

```
NetConfigException
```

## SocketServer (Class)

Constructor:

```
SocketServer(int port, ENetwork Network=eIPv4, int QLen=15);
SocketServer(HostAddress& Addr, int QLen=15);
```

General Description: TCP server.

Parent Class: `SocketStream`

Methods:

| | |
|---|---|
| `void Accept(void (*Servlet)`<br>`(const Socket& Client));` | Accept a connection and call `Servlet` with `Socket` handle. |
| `void Accept(HostAddress& Addr,`<br>`void (*Server)(const Socket& Client));` | Accept connection and capture host ID. |

Exceptions Thrown:

```
Exception
NetConnectException
```

## `SocketClient` (Class)

Constructor:

```
SocketClient(ENetwork Network=eIPv4);
SocketClient(HostAddress& Host, ENetwork Network=eIPv4); //auto-connect
```

General Description: TCP client.

Parent Class: `SocketStream`

Method:

| | |
|---|---|
| `void Connect(HostAddress& Addr);` | Connect to host at `Addr`. |

Exception Thrown:

`NetConnectException`

## `Datagram` (Class)

Constructor:

```
Datagram(HostAddress& Me, ENetwork Network=eIPv4,
    EProtocol Protocol=eDatagram);
Datagram(ENetwork Network=eIPv4, EProtocol
    Protocol=eDatagram);
```

General Description: General datagram (UDP) socket.

Parent Class: `Socket`

Methods:

| | |
|---|---|
| `void MinimizeDelay(bool Setting);` | Request minimal packet delay. |
| `void MaximizeThroughput(bool Setting);` | Request maximum network throughput. |
| `void MaximizeReliability(bool Setting);` | Request maximum reliability. |
| `void MinimizeCost(bool Setting);` | Request minimal cost. |
| `void PermitFragNegotiation(EFrag Setting);` | Set fragmentation negotiation. |

Exception Thrown:

`NetConfigException`

## `Broadcast` (Class)

Constructor:

```
Broadcast(HostAddress& Me);
```

General Description: Broadcast socket for subnets.

Parent Class: `Datagram`

Methods: (none)

Exception Thrown:

```
NetConfigException
```

## `MessageGroup` (Class)

Constructor:

```
MessageGroup(HostAddress& Me, ENetwork Network=eIPv4);
```

General Description: Multicast socket.

Parent Class: `Datagram`

Methods:

| | |
|---|---|
| `Connect(HostAddress& Address);` | Connect to multicast group address. |
| `void Join(HostAddress& Address, int IFIndex=0);` | Join multicast group. |
| `void Drop(HostAddress& Address);` | Drop multicast group. |

Exceptions Thrown:

```
NetConfigException
NetConnectException
RangeException
```



# Java Exceptions

This section describes all the relevant exceptions that a Java program may generate while working with its sockets.

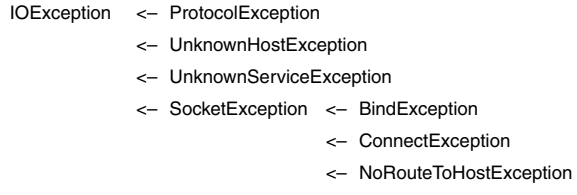


**FIGURE D.3**
*The Java exceptions class hierarchy.*

### `java.io.IOException` (Class)

Constructor:

```
IOException();
IOException(String msg);
```

General Description: General exceptions during input and output.

Parent Class: `Exception`

Child Exceptions:

| | |
|---|---|
| `java.net.ProtocolException` | Protocol error in `Socket`. |
| `java.net.UnknownHostException` | Hostname not found in DNS. |
| `java.net.UnknownServiceException` | Unsupported service attempted. |

### `java.net.SocketException` (Class)

Constructor:

```
SocketException();
SocketException(String msg);
```

General Description: Exception when trying to use `bind()`, `connect()`, `listen()`, or `accept()`. Used by `ServerSocket`, `ClientSocket`, and `MessageGroup`.

Parent Class: `IOException`

Child Exceptions:

| | |
|---|---|
| `java.net.BindException` | Could not bind to address/port (often because it is already in use by another process). |
| `java.net.ConnectException` | Host unavailable, not found, not responding, or not process listening on designated port. |
| `java.net.NoRouteToHostException` | Route to the destination could not be established. |

## Java Support Classes

Like the C++ framework, Java uses several support classes to interface with its socket API. This section describes only those that are directly relevant to sockets.

### `java.net.DatagramPacket` (Class)

Constructor:

```
DatagramPacket(byte[] buf, int len);
DatagramPacket(byte[] buf, int len, InetAddress addr, int port);
```

```
DatagramPacket(byte[] buf, int Offset, int len);
DatagramPacket(byte[] buf, int Offset, int len, InetAddress addr, int port);
```

General Description: Basic message carriers for receiving and sending messages.

Methods:

| | |
|---|---|
| `InetAddress getAddress();`<br>`void setAddress(InetAddress addr);` | Get or set the source or destination address of the packet. |
| `byte[] getData();`<br>`void setData(byte[] buf);`<br>`void setData(byte[] buf, int offset, int len);` | Get or set the message data. |
| `int getLength();`<br>`void setLength(int length);` | Get or set the message data length. |
| `int getOffset();` | Get the offset of the data to be sent or received. |
| `int getPort();`<br>`void setPort(int port);` | Get or set the source or destination port of the packet. |

Exceptions Thrown: (none)

## `java.net.InetAddress` (Class)

Constructor: (none)

General Description: Internet address socket. This class does not have a constructor. Instead, use one of the static methods.

Static Methods:

| | |
|---|---|
| `InetAddress getByName(String host);` | Return an InetAddress for host. |
| `InetAddress getAllByName(String host);` | Return all InetAddresses for host. |
| `InetAddress getLocalHost();` | Get the localhost IP address. |

Methods:

| | |
|---|---|
| `String getHostAddress();` | Get the numeric address. |
| `byte[] getAddress();` | Get the binary address. |
| `boolean isMulticastAddress();` | Check to see if the address is in the multicast range. |
| `String getHostName();` | Get the host's actual name. |

Exception Thrown:

`UnknownHostException`

# Java I/O Classes

Java has an outstanding set of classes that work with various I/O. Unfortunately, they are not very intuitive, and connecting them together is similar to working with a puzzle. Please read Chapter 12, "Using Java's Networking API," for more information on how to interlock these pieces into useful streams.

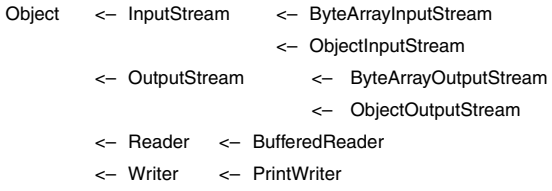


**FIGURE D.4**
*The Java I/O class hierarchy.*

## `java.io.InputStream` (Abstract Class)

Constructor:

```
InputStream();
```

General Description: A general class for basic stream input.

Parent Class: `Object`

Methods:

| | |
|---|---|
| `int available();` | Return the number of bytes you can read without blocking. |
| `void close();` | Close the channel. |
| `void mark(int readlimit);` | Set the maximum number of bytes to buffer for `mark()` and `reset()`. |
| `boolean markSupported();` | Check to see if stream supports `mark()`/`reset()`. |
| `int read();` | Read a single byte from the stream. |
| `int read(byte[] arr);` | Read an array of bytes into `arr`. |
| `int read(byte[] arr, int offset, int length);` | Read an array of bytes into `arr` beginning at `offset` for `length` bytes. |
| `void reset();` | Return to last `mark`ed place. |
| `long skip(long n);` | Skip `n` bytes forward in the stream. |

Exceptions Thrown:

`IOException`

## `java.io.ByteArrayInputStream` (Class)

Constructor:

```
ByteArrayInputStream(byte[] buf);
ByteArrayInputStream(byte[] buf, int offset, int length);
```

General Description: Allows you to create a virtual input stream from an array of bytes. Sometimes you get a block of data (such as from `DatagramSocket`); this class takes the role of streaming that information.

Parent Class: `InputStream`

Methods: (none; many overridden methods from `InputStream`)

Exceptions Thrown: (none)

## `java.io.ObjectInputStream` (Class)



Constructor:

```
ObjectInputStream(InputStream o);
```

General Description: Using this class, you read transmitted or stored objects. You create this object with an `InputStream` (available in the `Socket` class).

Parent Class: `InputStream`

Methods:

| | |
|---|---|
| `int available();` | Return the number of bytes you can read without blocking. |
| `void close();` | Close this channel. |
| `void defaultReadObject();` | Read the current class's non-static and non-transient fields from this stream. |
| `int read();`<br>`int read(byte[] arr,`<br>`  int offset, int len);`<br>`readFully(byte[] arr);`<br>`readFully(byte[] arr,`<br>`  int offset, int len);` | Read a byte or an array of bytes beginning at `offset` for `len` bytes. `readFully()` reads all the bytes to fill the array, blocking as needed. |

| | |
|---|---|
| `boolean readboolean();`<br>`byte readByte();`<br>`char readChar();`<br>`double readDouble();`<br>`float readFloat();`<br>`int readInt();`<br>`long readLong();`<br>`short readShort();`<br>`int readUnsignedByte();`<br>`int readUnsignedShort();`<br>`String readUTF();` | Read the designated type. |
| `Object readObject();` | Read `Object` instance. You can discover the type and then convert it later with the casting operators. |

Exceptions Thrown:

`IOException`

`ClassNotFoundException`

`NotActiveException`

`OptionalDataException`

`InvalidObjectException`

`SecurityException`

`StreamCorruptedException`

## `java.io.OutputStream` (Abstract Class)

Constructor:

```
OutputStream();
```

General Description: A general class for basic stream input.

Parent Class: `Object`

Methods:

| | |
|---|---|
| `void close();` | Close the channel. |
| `void flush();` | Flush written data from buffers. |
| `void write(byte b);` | Write a single byte to the stream. |
| `int write(byte[] arr);` | Write an array of bytes (`arr`) to the stream. |
| `int write(byte[] arr, int offset, int len);` | Write an array of bytes (`arr`) beginning at `offset` for `len` bytes. |

Exception Thrown:

`IOException`

## java.io.ByteArrayOutputStream (Class)

Constructor:

```
ByteArrayOutputStream();
ByteArrayOutputStream(int size);
```

General Description: Allows you to stream data into an array of bytes. Classes like `DatagramSocket` work only with blocks of data; this class takes the role of streaming information.

Parent Class: `OutputStream`

Methods:

| | |
|---|---|
| `void reset();` | Clear buffers and empty array. |
| `int write(byte[] arr, int offset, int len);` | Write an array of bytes (`arr`) beginning at `offset` for `len` bytes. |
| `byte[] toByteArray();` | Return the array of streamed data. |
| `int size();` | Return the current size of the buffer. |
| `String toString(String encoder);` | Create string, translating chars with `encoder`. |
| `void write(int b);` | Write a single byte to the stream. |
| `void write(OutputStream o);` | Send the array of data through `OutputStream`. |

Exceptions Thrown: (none)

## java.io.ObjectOutputStream (Class)

Constructor:

```
ObjectOutputStream(OutputStream o);
```

General Description: Using this class, you transmit or store objects. You create this object with an `OutputStream` (available in the `Socket` class).

Parent Class: `OutputStream`

Methods:

| | |
|---|---|
| `void close();` | Close this channel. |
| `void defaultWriteObject();` | Write the current class's non-static and non-transient fields to this stream. You can call this only while in the `writeObject()` method during serialization. |
| `int flush();` | Flush written data from buffers. |

| | |
|---|---|
| `int reset();` | Toss the information written to the stream. |
| `void useProtocolVersion(int version);` | Force earlier serialization version. |
| `void write(byte b);`<br>`int write(byte[] arr);`<br>`int write(byte[] arr,`<br>`  int offset, int len);` | Write a byte or an array of bytes beginning at `offset` for `len` bytes. |
| `void writeboolean(boolean b);`<br>`void writeByte(byte b);`<br>`void writeBytes(String s);`<br>`void writeChar(int c);`<br>`void writeChars(String s);`<br>`void writeDouble(double d);`<br>`void writeFloat(float f);`<br>`void writeInt(int i);`<br>`void writeLong(long l);`<br>`void writeShort(int us);` | Write the designated type. |
| `void writeUTF(String s);`<br>`int writeFields();` | Write buffered fields to stream. |
| `void writeObject(Object o);` | Write `Object` instance. |

Exceptions Thrown:

`IOException`

`SecurityException`

## `java.io.BufferedReader` (Class)

Constructor:

```
BufferedReader(Reader i);
BufferedReader(Reader i, int size);
```

General Description: Keeps buffers for improved performance. Does some translation of types for line recognition. `size` specifies the size of the input buffers.

Parent Class: `Reader`

Methods:

| | |
|---|---|
| `void close();` | Close the channel. |
| `void mark(int readlimit);` | Set the maximum number of bytes to buffer for `mark()` and `reset()`. |

| | |
|---|---|
| `boolean markSupported();` | Check to see if stream supports `mark()`/`reset()`. |
| `int read();` | Read a single byte from the stream. |
| `int read(byte[] arr, int offset, int length);` | Read an array of bytes into `arr` beginning at `offset` for `length` bytes. |
| `String readLine();` | Read up to newline and return `String`. |
| `boolean ready();` | Return true if ready to read. |
| `void reset();` | Return to last `marked` place. |
| `long skip(long n);` | Skip `n` bytes forward in the stream. |

Exception Thrown:

`IOException`

## `java.io.PrintWriter` (Class)

Constructor:

```
PrintWriter(Writer o);
PrintWriter(Writer o, boolean autoFlush);
PrintWriter(OutputStream o);
PrintWriter(OutputStream o, boolean autoFlush);
```

General Description: does some translation from data types into readable text. The `autoFlush` flag forces a flush when the program calls `println()`.

Parent Class: `Writer`

Methods:

| | |
|---|---|
| `boolean checkError();` | Flush stream and check for any errors. |
| `void close();` | Close this channel. |
| `void defaultWriteObject();` | Write the current class's non-static and non-transient fields to this stream. You can only call this while in the `writeObject()` method during serialization. |
| `int flush();` | Flush written data from buffers. |
| `int reset();` | Toss the information written to the stream. |
| `void write(byte b);`<br>`int write(byte[] arr);`<br>`int write(byte[] arr, int offset, int len);` | Write a byte or an array of bytes beginning at `offset` for `len` bytes. |

| | |
|---|---|
| `void print(boolean b);`<br>`void print(char c);`<br>`void print(char[] s);`<br>`void print(double d);`<br>`void print(float f);`<br>`void print(int i);`<br>`void print(long l);`<br>`void print(Object obj);`<br>`void print(String s);` | Print the designated type. The `Object` type uses the `String.valueOf()` method to convert data. |
| `void println();`<br>`void println(boolean b);`<br>`void println(char c);`<br>`void println(char[]s);`<br>`void println(double d);`<br>`void println(float f);`<br>`void println(int i);`<br>`void println(long l);`<br>`void println(Object obj);`<br>`void println(String s);` | Print the designated type and terminate line with newline. If `autoFlush` is enabled, flush the stream. |
| `void write(int b);` | Write a single byte to the stream. |
| `int write(char[] arr);` | Write an array of chars (`arr`) to stream. |
| `int write(char[] arr,`<br>`int offset, int len);` | Write an array of chars (`arr`) beginning at `offset` for `len` bytes. |
| `int write(String s);` | Write string to stream. |
| `int write(String s,`<br>`int offset, int len);` | Write string to stream beginning at `offset` for `len` bytes. |

Exceptions Thrown:

`IOException`

`SecurityException`

# Java Socket Classes

The Java Socket API supports four basic IPv4 classes: `Socket`, `ServerSocket`, `DatagramSocket`, and `MulticastSocket`. This section describes the interface for each of these classes.

```
Object    <– Socket
          <– ServerSocket
          <– DatagramSocket   <– MulticastSocket
```

**FIGURE D.5**
*The Java socket class hierarchy.*

## `java.net.Socket` (Class)

Constructor:

```
Socket(String host, int port);
Socket(InetAddress addr, int port);
Socket(String host, int port, InetAddress lAddr, int lPort);
Socket(InetAddress addr, int port, InetAddress lAddr, int lPort);
```

General Description: This is the basic communication interface (TCP) for all network traffic.

Parent Class: `Object`

Methods:

| | |
|---|---|
| `void close();` | Close the socket. |
| `InetAddress getInetAddress();` | Get the host address of the peer. |
| `InputStream getInputStream();` | Get the `InputStream` for receiving messages. |
| `boolean getKeepAlive();`<br>`void setKeepAlive(boolean on);` | Keep the connection alive. |
| `InetAddress getLocalAddress();` | Get the local address the socket is connected to. |
| `int getLocalPort();` | Get the local port. |
| `OutputStream getOutputStream();` | Get the `OutputStream` for sending messages. |
| `int getPort();` | Get the peer's port number. |
| `int getReceiveBufferSize();`<br>`void setReceiveBufferSize(int size);` | Get/set receive buffer's size. |
| `int getSendBufferSize();`<br>`void setSendBufferSize(int size);` | Get/set send buffer's size. |
| `int getSoLinger();`<br>`void setSoLinger(boolean on, int linger);` | Get/set the socket linger time (in seconds). |
| `int getSoTimeout();`<br>`void setSoTimeout(int timeout);` | Get/set the timeout for I/O. If enabled, reading the pipe aborts after specified time. |

| | |
|---|---|
| `boolean getTcpNoDelay();`<br>`void setTcpNoDelay(boolean on);` | Enable/disable the Nagle algorithm, which determines the process for sending information. If disabled, the computer sends the data before it receives any confirmation. |
| `void shutdownInput();` | Close the input channel. |
| `void shutdownOutput();` | Close the output channel. |

Exceptions Thrown:

`IOException`

`SocketException`

## `java.net.ServerSocket` (Class)

Constructor:

```
ServerSocket(int port);
ServerSocket(int port, int backLog);
ServerSocket(int port, int backLog, InetAddress bindAddr);
```

General Description: This specialized TCP socket creates a listening server socket.

Parent Class: `Object`

Static Method:

| | |
|---|---|
| `setSocketFactory(SocketImplFactory fac);` | Set the `Socket` implementation factory. |

Methods:

| | |
|---|---|
| `Socket accept();` | Accept a client connection and return a `Socket`. |
| `void close();` | Close the socket. |
| `InetAddress getInetAddress();` | Get the local address the socket is connected to. |
| `int getLocalPort();` | Get the local port. |
| `int getSoTimeout();`<br>`void setSoTimeout(int timeout);` | Get/set the timeout for I/O. If enabled, reading the pipe aborts after specified time. |

Exceptions Thrown:

`IOException`

`SocketException`

## java.net.DatagramSocket (Class)

Constructor:

```
DatagramSocket();
DatagramSocket(int port);
DatagramSocket(int port, InetAddress bindAddr);
```

General Description: This is the general datagram (UDP) socket for message passing.

Parent Class: `Object`

Methods:

| | |
|---|---|
| `void close();` | Close the socket. |
| `void connect(InetAddress addr, int port);` | Connect peers for implicit sending. |
| `void disconnect();` | Disconnect connected peers. |
| `InetAddress getInetAddress();` | Get the host address of the peer. |
| `InetAddress getLocalAddress();` | Get the local address the socket is connected to. |
| `int getLocalPort();` | Get the local port. |
| `int getPort();` | Get the peer's port number. |
| `int getReceiveBufferSize();`<br>`void setReceiveBufferSize(int size);` | Get/set receive buffer's size. |
| `int getSendBufferSize();`<br>`void setSendBufferSize(int size);` | Get/set send buffer's size. |
| `int getSoTimeout();`<br>`void setSoTimeout(int timeout);` | Get/set the timeout for I/O.<br>If enabled, reading the pipe aborts after specified time. |
| `void receive(DatagramPacket p);` | Receive message. |
| `void send(DatagramPacket p);` | Send message. |

Exceptions Thrown:

`IOException`

`SocketException`

## java.net.MulticastSocket (Class)

Constructor:

```
MulticastSocket();
MulticastSocket(int port);
```

General Description: This is the general datagram (UDP) socket for unconnected messages.

Parent Class: `DatagramSocket`

Methods:

| | |
|---|---|
| `InetAddress getInterface();`<br>`void setInterface(InetAddress addr);` | Get/set the local address the socket is connected to. |
| `int getTimeToLive();`<br>`void setTimeToLive(int TTL);` | Get/set the time-to-live for each message. |
| `void joinGroup(InetAddress addr);` | Leave multicast group. |
| `void leaveGroup(InetAddress addr);` | Join multicast group. |
| `void send(DatagramPacket p, int TTL);` | Send message with specific TTL. |

Exceptions Thrown:

`IOException`

`SocketException`