**Section 28.1 Introduction**

• Java provides stream sockets and datagram sockets (p. 2). With stream sockets (p. 2), a process establishes a connection (p. 2) to another process. While the connection is in place, data flows between the processes in streams. Stream sockets are said to provide a connection-oriented service (p. 2). The protocol used for transmission is the popular TCP (Transmission Control Protocol; p. 2).

• With datagram sockets (datagram socket), individual packets of information are transmitted. UDP (User Datagram Protocol; p. 2) is a connectionless service that does not guarantee that packets will not be lost, duplicated or arrive out of sequence.

#### Section 28.2 Reading a File on a Web Server

• JEditorPane (p. 3) method setPage (p. 5) downloads the document specified by its argument and displays it.

• Typically, an HTML document contains hyperlinks that link to other documents on the web. If an HTML document is displayed in an uneditable JEditorPaneand the user clicks a hyperlink (p. 5), a HyperlinkEvent (p. 5) occurs and theHyperlinkListeners are notified.

• HyperlinkEvent method getEventType (p. 5) determines the event type.HyperlinkEvent contains nested class EventType (p. 5), which declares event types ACTIVATED, ENTERED and EXITED. HyperlinkEvent method getURL (p. 5) obtains the URL represented by the hyperlink.

#### Section 28.3 Establishing a Simple Server Using Stream Sockets

• Stream-based connections (p. 2) are managed with Socket objects (p. 6).

• A ServerSocket object (p. 6) establishes the port (p. 6) where a server (p. 2) waits for connections from clients (p. 2). ServerSocket method accept (p. 6) waits indefinitely for a connection from a client and returns a Socket object when a connection is established.

• Socket methods getOutputStream and getInputStream (p. 7) get references to a Socket’s OutputStream and InputStream, respectively. Method close (p. 7) terminates a connection.

#### Section 28.4 Establishing a Simple Client Using Stream Sockets

• A server name and port number (p. 6) are specified when creating a Socketobject to enable it to connect a client to the server. A failed connection attempt throws an IOException.

• InetAddress method getByName (p. 20) returns an InetAddress object (p. 14) containing the IP address of the specified computer. InetAddress methodgetLocalHost (p. 20) returns an InetAddress object containing the IP address of the local computer executing the program.

#### Section 28.6 Datagrams: Connectionless Client/Server Interaction

• Connection-oriented transmission is like the telephone system—you dial and are given a connection to the telephone of the person with whom you wish to communicate. The connection is maintained for the duration of your phone call, even when you aren’t talking.

• Connectionless transmission (p. 20) with datagrams is similar to mail carried via the postal service. A large message that will not fit in one envelope can be broken into separate message pieces that are placed in separate, sequentially numbered envelopes. All the letters are then mailed at once. They could arrive in order, out of order or not at all.

• DatagramPacket objects store packets of data that are to be sent or that are received by an application. DatagramSockets send and receiveDatagramPackets.

• The DatagramSocket constructor that takes no arguments binds theDatagramSocket to a port chosen by the computer executing the program. The one that takes an integer port-number argument binds the DatagramSocket to the specified port. If a DatagramSocket constructor fails to bind theDatagramSocket to a port, a SocketException occurs (p. 21). DatagramSocketmethod receive (p. 24) blocks (waits) until a packet arrives, then stores the packet in its argument.

• DatagramPacket method getAddress (p. 24) returns an InetAddress object containing information about the computer from or to which the packet was sent. Method getPort (p. 24) returns an integer specifying the port number (p. 6) through which the DatagramPacket was sent or received. Method getLength(getLength) returns the number of bytes of data in a DatagramPacket. MethodgetData (p. 24) returns a byte array containing the data.

• The DatagramPacket constructor for a packet to be sent takes four arguments—the byte array to be sent, the number of bytes to be sent, the client address to which the packet will be sent and the port number where the client is waiting to receive packets.

• DatagramSocket method send (p. 24) sends a DatagramPacket out over the network.

• If an error occurs when receiving or sending a DatagramPacket, anIOException occurs.

### Self-Review Exercises

[**28.1**](http://proquest.safaribooksonline.com/9780133813036/ch28lev1sec12_html#ch28ans01)Fill in the blanks in each of the following statements:

a) Exception \_\_\_\_\_\_\_\_\_\_ occurs when an input/output error occurs when closing a socket.

b) Exception \_\_\_\_\_\_\_\_\_\_ occurs when a hostname indicated by a client cannot be resolved to an address.

c) If a DatagramSocket constructor fails to set up a DatagramSocket properly, an exception of type \_\_\_\_\_\_\_\_\_\_ occurs.

d) Many of Java’s networking classes are contained in package \_\_\_\_\_\_\_\_\_\_.

e) Class \_\_\_\_\_\_\_\_\_\_ binds the application to a port for datagram transmission.

f) An object of class \_\_\_\_\_\_\_\_\_\_ contains an IP address.

g) The two types of sockets we discussed in this chapter are \_\_\_\_\_\_\_\_\_\_ and \_\_\_\_\_\_\_\_\_\_.

h) Method getLocalHost returns a(n) \_\_\_\_\_\_\_\_\_\_ object containing the local IP address of the computer on which the program is executing.

i) The URL constructor determines whether its String argument is a valid URL. If so, the URLobject is initialized with that location. If not, a(n) \_\_\_\_\_\_\_\_\_\_ exception occurs.

[**28.2**](http://proquest.safaribooksonline.com/9780133813036/ch28lev1sec12_html#ch28ans02)State whether each of the following is true or false. If false, explain why.

a) UDP is a connection-oriented protocol.

b) With stream sockets a process establishes a connection to another process.

c) A server waits at a port for connections from a client.

d) Datagram packet transmission over a network is reliable—packets are guaranteed to arrive in sequence.

### Answers to Self-Review Exercises

[**28.1**](http://proquest.safaribooksonline.com/9780133813036/ch28lev1sec11_html#ch28que01) a) IOException. b) UnknownHostException. c) SocketException. d) java.net. e)DatagramSocket. f) InetAddress. g) stream sockets, datagram sockets. h) InetAddress. i)MalformedURLException.

[**28.2**](http://proquest.safaribooksonline.com/9780133813036/ch28lev1sec11_html#ch28que02)a) False; UDP is a connectionless protocol and TCP is a connection-oriented protocol. b) True. c) True. d) False; packets can be lost, arrive out of order or be duplicated.

### Exercises

**28.3** Distinguish between connection-oriented and connectionless network services.

**28.4** How does a client determine the hostname of the client computer?

**28.5** Under what circumstances would a SocketException be thrown?

**28.6** How can a client get a line of text from a server?

**28.7** Describe how a client connects to a server.

**28.8** Describe how a server sends data to a client.

**28.9** Describe how to prepare a server to receive a stream-based connection from a single client.

**28.10** How does a server listen for streams-based socket connections at a port?

**28.11** What determines how many connect requests from clients can wait in a queue to connect to a server?

**28.12** As described in the text, what reasons might cause a server to refuse a connection request from a client?

**28.13** Use a socket connection to allow a client to specify a filename of a text file and have the server send the contents of the file or indicate that the file does not exist.

**28.14** Modify [Exercise 28.13](http://proquest.safaribooksonline.com/9780133813036/ch28lev1sec13_html#ch28que13) to allow the client to modify the contents of the file and send the file back to the server for storage. The user can edit the file in a JTextArea, then click a save changes button to send the file back to the server.

**28.15** ***(Multithreaded Server)*** Multithreaded servers are quite popular today, especially because of the increasing use of multi-core servers. Modify the simple server application presented in [Section 28.5](http://proquest.safaribooksonline.com/9780133813036/ch28lev1sec5_html#ch28lev1sec5) to be a multithreaded server. Then use several client applications and have each of them connect to the server simultaneously. Use an ArrayList to store the client threads. ArrayList provides several methods to use in this exercise. Method size determines the number of elements in an ArrayList. Method get returns the element in the location specified by its argument. Method add places its argument at the end of the ArrayList. Methodremove deletes its argument from the ArrayList.

**28.16** ***(Checkers Game)*** In the text, we presented a Tic-Tac-Toe program controlled by a multithreaded server. Develop a checkers program modeled after the Tic-Tac-Toe program. The two users should alternate making moves. Your program should mediate the players’ moves, determining whose turn it is and allowing only valid moves. The players themselves will determine when the game is over.

**28.17** ***(Chess Game)*** Develop a chess-playing program modeled after [Exercise 28.16](http://proquest.safaribooksonline.com/9780133813036/ch28lev1sec13_html#ch28que16).

**28.18** ***(Blackjack Game)*** Develop a blackjack card game program in which the server application deals cards to each of the clients. The server should deal additional cards (per the rules of the game) to each player as requested.

**28.19** ***(Poker Game)*** Develop a poker game in which the server application deals cards to each client. The server should deal additional cards (per the rules of the game) to each player as requested.

**28.20** ***(Modifications to the Multithreaded Tic-Tac-Toe Program)*** The programs in [Figs. 28.11](http://proquest.safaribooksonline.com/9780133813036/ch28lev2sec15_html#ch28fig11) and [28.13](http://proquest.safaribooksonline.com/9780133813036/ch28lev2sec15_html#ch28fig13) implemented a multithreaded, client/server version of the game of Tic-Tac-Toe. Our goal in developing this game was to demonstrate a multithreaded server that could process multiple connections from clients at the same time. The server in the example is really a mediator between the two clients—it makes sure that each move is valid and that each client moves in the proper order. The server does not determine who won or lost or whether there was a draw. Also, there’s no capability to allow a new game to be played or to terminate an existing game.

The following is a list of suggested modifications to [Figs. 28.11](http://proquest.safaribooksonline.com/9780133813036/ch28lev2sec15_html#ch28fig11) and [28.13](http://proquest.safaribooksonline.com/9780133813036/ch28lev2sec15_html#ch28fig13):

a) Modify the TicTacToeServer class to test for a win, loss or draw after each move. Send a message to each client that indicates the result of the game when the game is over.

b) Modify the TicTacToeClient class to display a button that when clicked allows the client to play another game. The button should be enabled only when a game completes. Both class TicTacToeClient and class TicTacToeServer must be modified to reset the board and all state information. Also, the other TicTacToeClient should be notified that a new game is about to begin so that its board and state can be reset.

c) Modify the TicTacToeClient class to provide a button that allows a client to terminate the program at any time. When the user clicks the button, the server and the other client should be notified. The server should then wait for a connection from another client so that a new game can begin.

d) Modify the TicTacToeClient class and the TicTacToeServer class so that the winner of a game can choose game piece X or O for the next game. Remember: X always goes first.

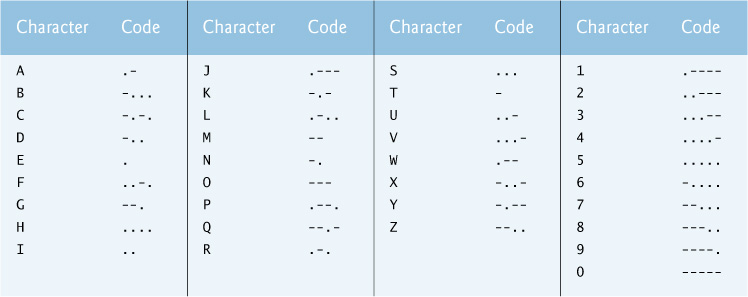
e) If you’d like to be ambitious, allow a client to play against the server while the server waits for a connection from another client.

**28.21** ***(3-D Multithreaded Tic-Tac-Toe)*** Modify the multithreaded, client/server Tic-Tac-Toe program to implement a three-dimensional 4-by-4-by-4 version of the game. Implement the server application to mediate between the two clients. Display the three-dimensional board as four boards containing four rows and four columns each. If you’re ambitious, try the following modifications:

a) Draw the board in a three-dimensional manner.

b) Allow the server to test for a win, loss or draw. Beware! There are many possible ways to win on a 4-by-4-by-4 board!

**28.22** ***(Networked Morse Code)*** Perhaps the most famous of all coding schemes is the Morse code, developed by Samuel Morse in 1832 for use with the telegraph system. The Morse code assigns a series of dots and dashes to each letter of the alphabet, each digit, and a few special characters (e.g., period, comma, colon and semicolon). In sound-oriented systems, the dot represents a short sound and the dash a long sound. Other representations of dots and dashes are used with light-oriented systems and signal-flag systems. Separation between words is indicated by a space or, simply, the absence of a dot or dash. In a sound-oriented system, a space is indicated by a short time during which no sound is transmitted. The international version of the Morse code appears in [Fig. 28.16](http://proquest.safaribooksonline.com/9780133813036/ch28lev1sec13_html#ch28fig16).



**Fig. 28.16** | Letters and digits in international Morse code.

Write a client/server application in which two clients can send Morse-code messages to each other through a multithreaded server application. The client application should allow the user totype English-language phrases in a JTextArea. When the user sends the message, the client application encodes the text into Morse code and sends the coded message through the server to the other client. Use one blank between each Morse-coded letter and three blanks between each Morse-coded word. When messages are received, they should be decoded and displayed as normal characters and as Morse code. The client should have one JTextField for typing and oneJTextArea for displaying the other client’s messages.