```
// Client that sends and receives packets to/from a server.
public class ClientSimple extends JFrame   {
  private JTextField enterField; // for entering messages
  private JTextArea displayArea; // for displaying messages
  private DatagramSocket socket; // socket to connect to server

  // set up GUI and DatagramSocket
  public ClientSimple()     {
    super( "Client" );
    enterField = new JTextField( "Type message here" );
    enterField.addActionListener(
      new ActionListener()
      {
        public void actionPerformed( ActionEvent event )
        {
          try // create and send packet
          {
            // get message from textfield
            String message = event.getActionCommand();
            displayArea.append( "\nSending packet containing: " +
              message + "\n" );
            byte data[] = message.getBytes(); // convert to bytes
            // create sendPacket
            DatagramPacket sendPacket = new DatagramPacket( data,
              data.length, InetAddress.getLocalHost(), 5000 );
            socket.send( sendPacket ); // send packet
            displayArea.append( "Packet sent\n" );
            displayArea.setCaretPosition(
```

```
                displayArea.getText().length() );
            } // end try
            catch ( IOException ioException )
            {
              displayMessage( ioException.toString() + "\n" );
               ioException.printStackTrace();
            } // end catch
          } // end actionPerformed
       } // end inner class
    ); // end call to addActionListener
   add( enterField, BorderLayout.NORTH );
    displayArea = new JTextArea();
    add( new JScrollPane( displayArea ), BorderLayout.CENTER );
    setSize( 400, 300 ); // set window size
    setVisible( true ); // show window
    try // create DatagramSocket for sending and receiving packets
    {
      socket = new DatagramSocket();
    } // end try
    catch ( SocketException socketException )
    {
      socketException.printStackTrace();
      System.exit( 1 );
    } // end catch
} // end Client constructor

// wait for packets to arrive from Server, display packet contents
public void waitForPackets()
```

```
{
  while ( true )
  {
    try // receive packet and display contents
    {
      byte data[] = new byte[ 100 ]; // set up packet
      DatagramPacket receivePacket = new DatagramPacket(
        data, data.length );
      socket.receive( receivePacket ); // wait for packet
      // display packet contents
      displayMessage( "\nPacket received:" +
        "\nFrom host: " + receivePacket.getAddress() +
        "\nHost port: " + receivePacket.getPort() +
        "\nLength: " + receivePacket.getLength() +
        "\nContaining:\n\t" + new String( receivePacket.getData(),
          0, receivePacket.getLength() ) );
    } // end try
    catch ( IOException exception )
    {
      displayMessage( exception.toString() + "\n" );
      exception.printStackTrace();
    } // end catch
  } // end while
} // end method waitForPackets

// manipulates displayArea in the event-dispatch thread
private void displayMessage( final String messageToDisplay )
{
```

```
      SwingUtilities.invokeLater(
        new Runnable()
        {
          public void run() // updates displayArea
          {
            displayArea.append( messageToDisplay );
          } // end method run
        } // end inner class
      ); // end call to SwingUtilities.invokeLater
    } // end method displayMessage
}  // end class Client
```

```
// Server that receives and sends packets from/to a client.
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.SocketException;
import java.awt.BorderLayout;
import javax.swing.JFrame;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.SwingUtilities;

public class ServerSimple extends JFrame
{
  private JTextArea displayArea; // displays packets received
  private DatagramSocket socket; // socket to connect to client

  // set up GUI and DatagramSocket
  public ServerSimple()
  {
    super( "Server" );

    displayArea = new JTextArea(); // create displayArea
    add( new JScrollPane( displayArea ), BorderLayout.CENTER );
    setSize( 400, 300 ); // set size of window
    setVisible( true ); // show window

    try // create DatagramSocket for sending and receiving packets
    {
```

```
      socket = new DatagramSocket( 5000 );
    } // end try
    catch ( SocketException socketException )
    {
      socketException.printStackTrace();
      System.exit( 1 );
    } // end catch
} // end Server constructor

// wait for packets to arrive, display data and echo packet to client
public void waitForPackets()
{
    while ( true )
    {
      try // receive packet, display contents, return copy to client
      {
        byte data[] = new byte[ 100 ]; // set up packet
        DatagramPacket receivePacket =
          new DatagramPacket( data, data.length );

        socket.receive( receivePacket ); // wait to receive packet

        // display information from received packet
        displayMessage( "\nPacket received:" +
          "\nFrom host: " + receivePacket.getAddress() +
          "\nHost port: " + receivePacket.getPort() +
          "\nLength: " + receivePacket.getLength() +
          "\nContaining:\n\t" + new String( receivePacket.getData(),
```

```
            0, receivePacket.getLength() ) );

         sendPacketToClient( receivePacket ); // send packet to client
      } // end try
      catch ( IOException ioException )
      {
         displayMessage( ioException.toString() + "\n" );
         ioException.printStackTrace();
      } // end catch
   } // end while
} // end method waitForPackets

// echo packet to client
private void sendPacketToClient( DatagramPacket receivePacket )
   throws IOException
{
   displayMessage( "\n\nEcho data to client..." );

   // create packet to send
   DatagramPacket sendPacket = new DatagramPacket(
      receivePacket.getData(), receivePacket.getLength(),
      receivePacket.getAddress(), receivePacket.getPort() );

   socket.send( sendPacket ); // send packet to client
   displayMessage( "Packet sent\n" );
} // end method sendPacketToClient

// manipulates displayArea in the event-dispatch thread
```

```
   private void displayMessage( final String messageToDisplay )
   {
      SwingUtilities.invokeLater(
         new Runnable()
         {
            public void run() // updates displayArea
            {
               displayArea.append( messageToDisplay ); // display message
            } // end method run
         } // end anonymous inner class
      ); // end call to SwingUtilities.invokeLater
   } // end method displayMessage
} // end class Server
```