/** File is: HostServer.java 10-24-05  Version 1.0
Author Clark Elliott. Status: throw-away code to illustrate some points for class.

NOTE: This is NOT a suggested implementation for your agent platform,
but rather a running example of something that might serve some of
your needs, or provide a way to start thinking about what YOU would like to do.
You may freely use this code as long as you improve it and write your own comments.

TO EXECUTE:
1. Start the HostServer in some shell. >> java HostServer
2. start a web browser and point it to http://localhost:1565. Enter some text and press
   the submit button to simulate playing the animal game.
3. start a second web browser, also pointed to http://localhost:1565 and do the same.
   Note that the two agents do not interfere with one another.
4. To suggest to an agent that it migrate, enter the string "migrate"
in the text box and submit. The agent will migrate to a new port, but keep its old state.

COMMENTS:
This is a simple framework for hosting agents that can migrate from
one server and port, to another server and port. For the example, the
server is always localhost, but the code would work the same on
different, and multiple, hosts.

State is implemented simply as an integer that is incremented. This represents the state
of the animal game for some agent.

The example uses a standard, default, HostListener port of 1565.

DESIGN OVERVIEW
Here is the high-level design, more or less:

HOST SERVER
  Runs on some machine
  Port counter is just a global integer incremented after each assignment
  Loop:
  Accept connection with a request for hosting
  Spawn an Agent Looper/Listener with the new, unique, port

AGENT LOOPER/LISTENER
  Make an initial state, or accept an existing state if this is a migration
  Get an available port from this host server
  Set the port number back to the client which now knows IP address and port of its
  new home.
  Loop:
  Accept connections from web client(s)
  Spawn an agent worker, and pass it the state and the parent socket blocked in this
  loop

AGENT WORKER
  If normal interaction, just update the state, and pretend to play the animal game
  (Migration should be decided autonomously by the agent, but we instigate it here with
  client)
  If Migration:
  Select a new host
  Send server a request for hosting, along with its state
  Get back a new port where it is now already living in its next incarnation
  Send HTML FORM to web client pointing to the new host/port.
Wake up and kill the Parent AgentLooper/Listener by closing the socket
Die

WEB CLIENT
Just a standard web browser pointing to http://localhost:1565 to start.

import java.io.*; // Get the Input Output libraries
import java.net.*; // Get the Java networking libraries
class AgentWorker extends Thread { // Class definition
    Socket sock; // Class member, socket, local to Worker.
    agentHolder parentAgentHolder;
    int localPort;
    // Constructor: assign args s,prt,thr to local values:
    AgentWorker (Socket s, int prt, agentHolder ah)
    {sock = s; localPort = prt; parentAgentHolder = ah;}
    public void run()
    {System.out.println("In agentWorker Thread for agent.");
    PrintStream out = null;
    BufferedReader in = null;
    String NewHost = "localhost";
    int NewHostMainPort = 1565;
    String buf = "";
    int newPort;
    Socket clientSock;
    BufferedReader fromHostServer;
    PrintStream toHostServer;
    try {
        out = new PrintStream(sock.getOutputStream());
        in = new BufferedReader
        (new InputStreamReader(sock.getInputStream()));
        String inLine = in.readLine ();
        /* The decision to MIGRATE should be decided autonomously BY THE AGENT. Here, for
        illustration, we just allow the web client to instigate the migration. We also
        just pick localhost for the destination host for the illustration, but this also
        should be chosen, autonomously, by the agent. */
        if (inLine.indexOf("migrate") > -1)
        /* select server, send server a request for hosting, get port
        back from server, send HTML pointing to new server and
        port, kill parent AgentListener loop, die: */
        clientSock = new Socket(NewHost, NewHostMainPort); //
        fromHostServer = new BufferedReader(new InputStreamReader(clientSock.getInputStream()));
        toHostServer = new PrintStream(clientSock.getOutputStream());
        toHostServer.println("Please host me. Send my port! [State=" +
        parentAgentHolder.agentState + "]");
        toHostServer.flush();
        /* We really only needed the port. The HTML was sent as convenience for starting
        with an initial request to the HostServer from a web client: */
        for (;;) // so read until we find the port number...
        buf = fromHostServer.readLine();
        if (buf.indexOf("[Port=") > -1)
            break;
    }
    String tempbuf =
        buf.substring(buf.indexOf("[Port=")+6, buf.indexOf("]", buf.indexOf("[Port="));
    newPort = Integer.parseInt(tempbuf);
    System.out.println("newPort is: " + newPort); //
    // The new host is just faked here (localhost), agent would select:
    AgentListener.sendHTMLheader(newPort, NewHost, inLine, out);
    out.println("<h3>We are Migrating to " + NewHost + ""," + newPort + "</h3>");
    AgentListener.sendHTMLsubmit(out);
    System.out.println("Killing parent listening loop.");
    ServerSocket ss = parentAgentHolder.sock; ss.close(); // close parent sock by reference.
else {
    parentAgentHolder.agentState++;  // Simple change to state.
    AgentListener.sendHTMLheader(localPort, NewHost, inline, out);
    out.println("<h3> We are playing the animal game with state " +
    parentAgentHolder.agentState + "</h3>");
    AgentListener.sendHTMLsubmit(out);
    sock.close(); // close this connection, but not the server;
} catch (IOException ioe) {System.out.println(ioe);} 
}

class agentHolder {
    ServerSocket sock; // Pointer to the parent looping listener for the agent
    int agentState;  // This might be the animal data structure, just an int for example
    agentHolder(ServerSocket s){sock = s;}
}

class AgentListener extends Thread {    // Class definition
    Socket sock;  // Class member, socket, local to AgentListener
    int localPort;
    // Constructor, assign args As,prt to local values:
    AgentListener (Socket As, int prt) {sock = As; localPort = prt;}
    int agentState = 0;//If state exists, set it, otherwise start at zero.
    public void run(){
        BufferedReader in = null;
        PrintStream out = null;
        String NewHost = "localhost"; // this depends on what the host really is
        try{
            String buf;
            out = new PrintStream(sock.getOutputStream());
            // The I WANT TO COME connection:
            in = new BufferedReader(new InputStreamReader(sock.getInputStream()));
            buf = in.readLine();
            if (buf.indexOf("[State=") > -1) {
                String tempbuf =
                    buf.substring(buf.indexOf("[State=")+7, buf.indexOf(""]);
                agentState = Integer.parseInt(tempbuf);
                System.out.println("agentState is: " + agentState);
            }
            System.out.println(buf);
            sendHTMLheader(localPort, NewHost, buf, out);
            out.println("Now in Agent Looper starting Agent Listening Loop
<br>
[Port=" + localPort + "]<br>");
            sendHTMLsubmit(out);
            ServerSocket servsock = new ServerSocket(localPort, 2);
            agentHolder agenthold = new agentHolder(servsock);
            agenthold.agentState = agentState;
            while (true) {
                sock = servsock.accept();
                System.out.println("Got a connection to agent at port " + localPort);
                new AgentWorker (sock, localPort, agenthold).start();
            }
        } catch (IOException ioe) {
            System.out.println("Either connection failed, or just killed Listener Loop for agent at port " +
            localPort);
            System.out.println(ioe);
        }
        static void sendHTMLheader (int localPort, String NewHost, String inLine, PrintStream out)
        {
            out.println("HTTP/1.1 200 OK");
            out.println("Content-Length: 700"); // who knows how long, faking it.
```
out.println("Content-Type: text/html");
out.println("<html><head> </head><body> 

<h2> This is for submission to PORT " + localPort + " on " + NewHost + "</h2>
<h3> You sent:" + inLine + "</h3>

<form method="GET" action="http://" + NewHost + ":" + localPort + ":">
  Enter text or <i>migrate</i>:
  <input type="text" name="person" size=20 value="YourTextInput"><p>
</form></body></html>

public class HostServer {
  public static int NextPort = 30000;

  public static void main(String a[]) throws IOException {
    int q_len = 6; /* Number of requests for OpSys to queue */
    int port = 1565;
    Socket sock;

    ServerSocket servsock = new ServerSocket(port, q_len);

    System.out.println("DIA Master receiver started at 1565.
    
    while (true) {
      NextPort = NextPort + 1;
      sock = servsock.accept();
      System.out.println("Starting an AgentListener at port " + NextPort);
      new AgentListener (sock, NextPort).start();
    }
  }
}
```