```
import java.io.IOException;
import java.net.URI;
import java.util.Enumeration;

import net.jxta.discovery.DiscoveryService;
import net.jxta.document.AdvertisementFactory;
import net.jxta.document.Element;
import net.jxta.document.MimeMediaType;
import net.jxta.document.StructuredDocument;
import net.jxta.document.StructuredDocumentFactory;
import net.jxta.endpoint.Message;
import net.jxta.endpoint.StringMessageElement;
import net.jxta.exception.PeerGroupException;
import net.jxta.id.IDFactory;
import net.jxta.peergroup.PeerGroup;
import net.jxta.peergroup.PeerGroupFactory;
import net.jxta.peergroup.PeerGroupID;
import net.jxta.pipe.InputPipe;
import net.jxta.pipe.OutputPipe;
import net.jxta.pipe.PipeService;
import net.jxta.protocol.ModuleImplAdvertisement;
import net.jxta.protocol.PeerGroupAdvertisement;
import net.jxta.protocol.PipeAdvertisement;

// RestoPeer represents a restaurant that receives auction requests
// for french fries from HungryPeers. RestoPeers offers three sizes of
// french fries (small, large, medium). Each restaurant assignes a
// different price to each size. Each restaurant also offers a special
// offering.
//
// Each restaurant is uniquely identified by its brand name.

public class RestoPeer {

    private PeerGroup netpg = null;      // The NetPeerGroup
    private PeerGroup restoNet = null;   // The restoNet Peergroup

    private String brand = "Chez JXTA";         // Brand of this restaurant
    private String specials = "large ($3.00)"; // Current restaurant special


    // Services within the RestoNet peergroup
    private DiscoveryService disco = null;  // Discovery service
    private PipeService pipes = null;       // Pipe service
    private PipeAdvertisement myAdv = null; // My RestoPeer pipe advertise
ment
    private InputPipe pipeIn = null;        // Input pipe that we listen t
o
                                            // for requests from Hungry Pe
ers

    private int timeout = 3000;          // discovery wait timeout
    private int rtimeout = 8000;         // resolver pipe timeout

    static String groupURL = "jxta:uuid−4d6172676572696e204272756e6f202002";

    public static void main(String args[]) {
        RestoPeer myapp = new RestoPeer();
        myapp.startJxta();
        System.exit(0);
    }

    // Method to start the JXTA platform, join the RestoNet peergroup and
    // advertise the RestoPeer service
```

```
    private void startJxta() {
        try {
            //Discover and join (or start) the default peergroup
            netpg = PeerGroupFactory.newNetPeerGroup();
        } catch (PeerGroupException e) {
            //Couldn’t initialize; can’t continue
            System.out.println("Fatal error : creating the NetPeerGroup");
            System.exit(1);
        }

        // Discover (or create) and join the RestoNet peergroup
        try {
            joinRestoNet();
        } catch (Exception e) {
            System.out.println("Can’t join or create RestoNet");
            System.exit(1);
        }

        // Discover (or create) and publish a RestoPeer pipe to receive
        // auction request for fries from HungryPeers
        if (!createRestoPipe()) {
            System.out.println("Aborting due to failure to create RestoPeer pipe");
            System.exit(1);
        }

        // Start the RestoPeer server loop to respond to Hungry peers
        // fries requests.
        handleFriesRequest();
    }

    // Discover (or create) and join the RestoNet peergroup
    private void joinRestoNet() throws Exception {

        int count = 3;   // maximun number of attempts to discover
        System.out.println("Attempting to Discover the RestoNet PeerGroup");

        // Get the discovery service from the NetPeergroup
        DiscoveryService hdisco = netpg.getDiscoveryService();

        Enumeration ae = null;   // Holds the discovered peers

        // Loop until wediscover the RestoNet or
        // until we’ve exhausted the desired number of attempts
        while (count−− > 0) {
            try {
                // search first in the peer local cache to find
                // the RestoNet peergroup advertisement
                ae = hdisco.getLocalAdvertisements(DiscoveryService.GROUP,
                                          "Name", "RestoNet");

                // If we found the RestoNet advertisement we are done
                if ((ae != null) && ae.hasMoreElements())
                    break;

                // If we did not find it, we send a discovery request
                hdisco.getRemoteAdvertisements(null,
                       DiscoveryService.GROUP, "Name", "RestoNet", 1, null)
;

                // Sleep to allow time for peers to respond to the
                // discovery request
                try {
                    Thread.sleep(timeout);
                } catch (InterruptedException ie) {}
            } catch (IOException e){
                // Found nothing! Move on
            }
        }

        PeerGroupAdvertisement restoNetAdv = null;

        // Check if we found the RestoNet advertisement.
        // If we didn’t, then either
        //       we are the first peer to join or
        //       no other RestoNet peers are up.
        // In either case, we must create the RestoNet peergroup

        if (ae == null || !ae.hasMoreElements()) {
            System.out.println(
                 "Could not find the RestoNet peergroup; creating one");
            try {
                // Create a new, all−purpose peergroup.
                ModuleImplAdvertisement implAdv =
                    netpg.getAllPurposePeerGroupImplAdvertisement();

                restoNet = netpg.newGroup(
                                mkGroupID(),      // Assign new group ID
                                implAdv,          // The implem. adv
                                "RestoNet",       // Name of peergroup
                                "RestoNet, Inc.");// Description of peergroup

                // Get the PeerGroup Advertisement
                restoNetAdv = netpg.getPeerGroupAdvertisement();

            } catch (Exception e) {
                System.out.println("Error in creating RestoNet Peergroup");
                throw e;
            }
        } else {
            // The RestoNet advertisement was found in the cache;
            // that means we can join the existing RestoNet peergroup

            try {
                restoNetAdv = (PeerGroupAdvertisement) ae.nextElement();
                restoNet = netpg.newGroup(restoNetAdv);
                System.out.println(
                    "Found the RestoNet Peergroup advertisement");
            } catch (Exception e) {
                System.out.println("Error in creating RestoNet PeerGroup from existing adv
");
                throw e;
            }
        }

        try {
            // Get the discovery and pipe services for the RestoNet Peergr
oup
            disco = restoNet.getDiscoveryService();
            pipes = restoNet.getPipeService();
        } catch (Exception e) {
            System.out.println("Error getting services from RestoNet");
            throw e;
        }

        System.out.println("RestoNet Restaurant (" + brand + ") is on−line");
        return;
    }
```

```
    // Method to handle fries auction requests from HungryPeers.
    // The method waits for HungryPeer requests pipe messages to arrive.
    // Incoming requests contain a pipe advertisement to respond to
    // the HungryPeers requester and a fries size.  The method
    // generates a bid offer for the request, opens an output pipe to
    // the HungryPeer requester, and send the response.
    private void handleFriesRequest() {

    //    InputStream ip = null;               // Input Stream to read mes
sage
        PipeAdvertisement hungryPipe = null; // HungryPeer Requester pipe

        StructuredDocument request = null;   // Request document
        StructuredDocument bid = null;       // Response document
        // Document mime types
        MimeMediaType mimeType = new MimeMediaType("text", "xml");
        Element el = null;                   // Element in document
        String name = null;                  // Name of the sender
        String size = null;                  // Fries size Requested
        OutputPipe pipeOut = null;           // Output pipe to respond to
                                             // HungryPeer requester

        System.out.println("RestoNet Restaurant (" + brand +
                           ") waiting for HungryPeer requests");

        // Loop waiting for HungryPeer Requests
        while (true) {
            Message msg = null;          // Incoming pipe message
            try {
                // Block until a message arrive on the RestoPeer pipe
                msg = pipeIn.waitForMessage();
                // If message is null discard message
                if (msg == null) {
                   if (Thread.interrupted()) {
                        // We have been asked to stop
                        System.out.println("Abort: RestoPeer interrupted");
                        return;
                    }
                }

                // We received a message; extract the request
                try {
                    // Extract the HungryPipe pipe information
                    // to reply to the sender
         //           ip = msg.getMessageElement("HungryPeerPipe");

                    // Construct the associated pipe advertisement
                    // via the AdvertisementFactory
                    hungryPipe = (PipeAdvertisement)
                    AdvertisementFactory.newAdvertisement( msg.getMessageE
lement("HungryPeerPipe").toString());

                    // Extract the sender name and fries size requested
                    // building a StructuredDocument
            //        ip = msg.getMessageElement("Request").getStream();
                    request = StructuredDocumentFactory.newStructuredDocum
ent(mimeType,msg.getMessageElement("Request").toString());


                    // Extract the fields from the structured Document
                    Enumeration enumeration = request.getChildren();

                    // Loop over all the elements of the document
                    while (enumeration.hasMoreElements()) {
```

```java
                    el = (Element) enumeration.nextElement();
                    String attr = (String) el.getKey();
                    String value = (String) el.getValue();

                    // Extract the HungryPeer Requester Name
                    if (attr.equals("Name")) {
                        name = value;
                        continue;
                    }

                    // Extract the Fries  size requested
                    else if (attr.equals("Fries")) {
                        size = value;
                        continue;
                    }
                }
            } catch (Exception e) {
                continue; // Broken content; silently discard
            }

            System.out.println("Received Request from HungryPeer "
                               + name
                               + " for "
                               + size
                               + " Fries.");

            // The auction request is valid. We can
            // create the output pipe to send the response bid to
            // the HungryPeer requester
            try {
                System.out.println(
                    "Attempting to create Output Pipe to HungryPeer " +
                    name);

                // Create an output pipe connection to the HungryPeer
                pipeOut = pipes.createOutputPipe(hungryPipe,
                                                 rtimeout);
                // Check if we have a pipe
                if (pipeOut == null) {
                    // Cannot conect the pipe
                    System.out.println("Could not find HungryPeer pipe");
                    continue;
                }
            } catch (Exception e) {
                // Pipe creation exception
                System.out.println("HungryPeer may not be listening anymore");
                continue;
            }

            // We have a pipe connection to the HungryPeer.
            // Now create the Bid Response document
            try {
                // Construct the Response document
                bid = StructuredDocumentFactory.newStructuredDocument(
                                 mimeType,
                                 "RestoNet:Bid");

                // Set the Bid values (Brand, price, special)
                // in the response document
                el = bid.createElement("Brand", brand);
                bid.appendChild(el);
                el = bid.createElement("Price", friesPrice(size));
                bid.appendChild(el);
                el = bid.createElement("Specials", specials);
```

```java
                    bid.appendChild(el);

                    // Create a new pipe message
                   // msg = pipes.createMessage();
                    msg = new Message();
                    StringMessageElement bide = new StringMessageElement("
Bid", bid.toString(),null);
                    // Push the Bid offer in the message
                   // msg.addElement(msg.newMessageElement(
                    //                 "Bid",mimeType, bid.getStream()));
                    msg.addMessageElement(bide);
                    // Send the message
                    pipeOut.send(msg);

                    // Close the output pipe connection
                    pipeOut.close();
                } catch (Exception ex) {
                    System.out.println(
                        "Error sending bid offer to HungryPeer " + name);
                    continue;
                }

                System.out.println("Sent Bid Offer to HungryPeer (" + name +
                       ") Fries price = "  + friesPrice(size) +
                       ", special = " + specials);
            } catch (Exception e) {
              System.out.println("Abort RestoPeer interrupted");
              return;
            }
        }
    }

    // Determine the price of the French fries depending on the size
    private String friesPrice(String size) {
        if (size.equals("small"))
            return "$1.50";
        if (size.equals("medium"))
            return "2.50";
        if (size.equals("large"))
            return "3.00";
        return "error";
    }

    // Create the resto pipe associated with this RestoPeer.
    // Discover first if a pipe advertisement exists, if
    // not create and publish it.
    private boolean createRestoPipe() {

        int count = 3;           // Discovery retry count
        Enumeration ae = null;   // Discovery response enumeration

        try {
            System.out.println(
                "Attempting to Discover the Restaurant RestoPipe");

            // Check if I have already published myself
            while (count-- > 0) {
                try {
                    // Check first locally if we have the advertisement ca
ched
                    ae = disco.getLocalAdvertisements(DiscoveryService.ADV
,
                                       "name", "RestoNet:RestoPipe:" + brand);
                    // If we found our pipe advertisement we are done
                    if (ae != null && ae.hasMoreElements())
                        break;

                    // We did not find the advertisement locally;
                    // send a remote request
                    disco.getRemoteAdvertisements(null,
                          DiscoveryService.ADV, "name",
                          "RestoNet:RestoPipe:" + brand, 1, null);

                    // Sleep to allow time for peers to respond to the
                    // discovery request
                    try {
                        Thread.sleep(timeout);
                    } catch (InterruptedException e) {}
                } catch (IOException e) {
                    // Found nothing! Move on
                }
            }

            if (ae == null || !ae.hasMoreElements()) {
                // We did not find the pipe advertisement, so create one
                System.out.println(
                    "Could not find the Restaurant Pipe Advertisement");

                // Create a pipe advertisement for our RestoPeer
                myAdv = (PipeAdvertisement)
                    AdvertisementFactory.newAdvertisement(
                        PipeAdvertisement.getAdvertisementType());

                // Assign a unique ID to the pipe
                myAdv.setPipeID(
                        IDFactory.newPipeID(restoNet.getPeerGroupID() ));

                // The symbolic name of the pipe is built from
                // the brand name of RestoPeer;  each RestoPeer
                // must therefore have a unique name.
                myAdv.setName("RestoNet:RestoPipe:" + brand);

                // Set the type of the pipe to be unidirectional
                myAdv.setType(PipeService.UnicastType);

                // We have the advertisement; publish it
                // into our local cache and to the RestoNet PeerGroup.
                // We use the default lifetime and the default
                // expiration time for remote publishing
                disco.publish(myAdv,
                              PeerGroup.DEFAULT_LIFETIME,
                              PeerGroup.DEFAULT_EXPIRATION);
                disco.remotePublish(myAdv,
                              PeerGroup.DEFAULT_EXPIRATION);

                System.out.println(
                    "Created the Restaurant Pipe Advertisement");
            } else {
                // We found an existing pipe advertisement
                myAdv = (PipeAdvertisement) ae.nextElement();
                System.out.println("Found Restaurant Pipe Advertisement");
            }

            // Create my input pipe to listen for hungry peers
            // requests
            pipeIn = pipes.createInputPipe(myAdv);
        } catch (Exception e) {
```

```
            System.out.println("Could not initialize the Restaurant pipe");
            return false;
        }
        return true;
    }

    private PeerGroupID mkGroupID() throws Exception {
        return (PeerGroupID) IDFactory.fromURI( new URI ("urn" + ":" + grou
pURL));
    }
}
```