```
    1: /**********************************************************************
    2:  * cs3524.solutions.mud.Vertex
    3:  **********************************************************************/
    4:
    5: package cs3524.solutions.mud;
    6:
    7: import java.util.Map;
    8: import java.util.HashMap;
    9: import java.util.List;
   10: import java.util.Vector;
   11: import java.util.Iterator;
   12:
   13: // Represents a location in the MUD (a vertex in the graph).
   14: class Vertex
   15: {
   16:     public String _name;             // Vertex name
   17:     public String _msg = "";         // Message about this location
   18:     public Map<String,Edge> _routes; // Association between direction
   19:                                      // (e.g. "north") and a path
   20:                                      // (Edge)
   21:     public List<String> _things;     // The things (e.g. players) at
   22:                                      // this location
   23:
   24:     public Vertex( String nm )
   25:     {
   26:         _name = nm;
   27:         _routes = new HashMap<String,Edge>(); // Not synchronised
   28:         _things = new Vector<String>();       // Synchronised
   29:     }
   30:
   31:     public String toString()
   32:     {
   33:         String summary = "\n";
   34:         summary += _msg + "\n";
   35:         Iterator iter = _routes.keySet().iterator();
   36:         String direction;
   37:         while (iter.hasNext()) {
   38:             direction = (String)iter.next();
   39:             summary += "To the " + direction + " there is " + ((Edge)_routes.get(
direction )).view + "\n";
   40:         }
   41:         iter = _things.iterator();
   42:         if (iter.hasNext()) {
   43:             summary += "You can see: ";
   44:             do {
   45:                 summary += iter.next() + " ";
   46:             } while (iter.hasNext());
   47:         }
   48:         summary += "\n\n";
   49:         return summary;
   50:     }
   51: }
   52:
```

```
 1: /*
 2:  * To change this license header, choose License Headers in Project Properties.
 3:  * To change this template file, choose Tools | Templates
 4:  * and open the template in the editor.
 5:  */
 6: package cs3524.solutions.mud;
 7:
 8: import java.rmi.Naming;
 9: import java.rmi.RMISecurityManager;
10: import java.rmi.server.UnicastRemoteObject;
11:
12: import java.util.List;
13:
14: import java.io.BufferedReader;
15: import java.io.InputStreamReader;
16:
17: import java.net.InetAddress;
18: import java.util.Iterator;
19:
20: /**
21:  *
22:  * @author darren
23:  */
24: public class MUDClient {
25:
26:     static MUDService service;
27:     static BufferedReader in = new BufferedReader( new InputStreamReader( System.i
n ));
28:     private static String username;
29:     private static String location;
30:     private static String servername = "demo";
31:     private List<String> inv;
32:
33:     public static void main(String args[]) throws Exception{
34:
35:         if(args.length < 2){
36:             System.err.println("Missing arguments <host> <port>");
37:             return;
38:         }
39:
40:         // Parse arguments
41:         String hostname = args[0];
42:         int port = Integer.parseInt(args[1]);
43:
44:         // Setup Security Manager
45:         System.setProperty("java.security.policy", "mud.policy");
46:         System.setSecurityManager( new RMISecurityManager() );
47:
48:         try {
49:
50:             String regURL = "rmi://" + hostname + ":" + port + "/MudService";
51:             service = (MUDService)Naming.lookup(regURL);
52:
53:             setup();
54:
55:         }
56:         catch (java.io.IOException e) {
57:             System.err.println( "I/O error." );
58:             System.err.println( e.getMessage() );
59:         }
60:         catch (java.rmi.NotBoundException e) {
61:             System.err.println( "Server not bound." );
62:             System.err.println( e.getMessage() );
63:         }
64:
65:     }
66:
67:     static void setup() throws Exception {
68:         System.out.println(service.getServers());
69:
70:         System.out.println("Please enter servername:");
71:         servername = in.readLine();
72:
73:         service.changeMUD(servername);
74:
75:         System.out.println(service.introduction());
76:
77:         username = in.readLine();
78:         location = service.getStartLocation();
79:
80:         if(service.addUser(username)){
81:                         systemStarted();
82:
83:         } else {
84:             System.out.println("Sory - this server is currently busy. Please try a
gain later");
85:         }
86:     }
87:
88:
89:     static void systemStarted() throws Exception{
90:
91:         boolean accepting = true;
92:
93:         location(location);
94:
95:         try {
96:
97:             while(accepting){
98:                 String input = in.readLine();
99:
100:                service.changeMUD(servername);
101:
102:                if(input.equals("quit") || input.equals("exit")){
103:                    accepting = false;
104:                } else if(input.equals("whoami?")){
105:                    System.out.println(username);
106:                } else if(input.toLowerCase().contains("move")){
107:                    String moving[] = input.split(" ");
108:                    String direction = moving[1];
109:
110:                    if(direction.equalsIgnoreCase("north") || direction.equalsIgno
reCase("east") || direction.equalsIgnoreCase("south") || direction.equalsIgnoreCase("west
")){
111:                        String newlocation = service.moveDirection(location, direct
ion);
112:                        if(newlocation.equals(location)){
113:                            System.out.println("Can not move " + direction);
114:                        } else {
115:                            location = newlocation;
116:                            location(location);
117:                            service.updateUserLocation(username, location);
118:                        }
119:                    } else {
120:                        System.out.println("Unknown Direction " + direction);
121:                    }
122:
123:                } else if(input.equals("who")){
124:                    System.out.println(service.getPlayersAtLocation(location) + "\
n");
125:                } else if(input.contains("take")){
126:                    String splt[] = input.split(" ");
127:                    String item = splt[1];
128:
```

```
129:                     if(service.takeItem(item, location)){
130:                         System.out.println("You now own the " + item+"\n");
131:                     } else {
132:                         System.out.println("Could not take the " + item+"\n");
133:                     }
134:                 }
135:             }
136:
137:         } catch(Exception e){
138:             return;
139:         }
140:
141:     }
142:
143:     static void location(String locationname) throws Exception{
144:
145:         System.out.println(service.location(locationname));
146:
147:     }
148:
149: }
```

```
 1: /**********************************************************************
 2:  * cs3524.solutions.mud.Edge
 3:  **********************************************************************/
 4:
 5: package cs3524.solutions.mud;
 6:
 7: // Represents an path in the MUD (an edge in a graph).
 8: class Edge
 9: {
10:     public Vertex _dest;   // Your destination if you walk down this path
11:     public String _view;   // What you see if you look down this path
12:
13:     public Edge( Vertex d, String v )
14:     {
15:         _dest = d;
16:         _view = v;
17:     }
18: }
19:
```

```
    1: /*
    2:  * To change this license header, choose License Headers in Project Properties.
    3:  * To change this template file, choose Tools | Templates
    4:  * and open the template in the editor.
    5:  */
    6: package cs3524.solutions.mud;
    7:
    8: import java.io.BufferedReader;
    9: import java.io.InputStreamReader;
   10: import java.net.InetAddress;
   11: import java.rmi.Naming;
   12: import java.rmi.RMISecurityManager;
   13: import java.rmi.server.UnicastRemoteObject;
   14:
   15: /**
   16:  *
   17:  * @author darren
   18:  */
   19: public class MUDServerMainline {
   20:
   21:     static BufferedReader in = new BufferedReader( new InputStreamReader( System.i
n ));
   22:
   23:
   24:     public static void main(String args[]){
   25:
   26:         if(args.length < 2){
   27:             System.err.println("You must provide two arguments: <regport> <serverp
ort>");
   28:             return;
   29:         }
   30:
   31:         int registryPort = Integer.parseInt(args[0]);
   32:         int serverPort = Integer.parseInt(args[1]);
   33:
   34:         System.out.println("Attempting to start server running on port " + Integer
.toString(registryPort));
   35:
   36:         try {
   37:
   38:             String hostname = (InetAddress.getLocalHost()).getCanonicalHostName();
   39:
   40:             // Setup Security Manager
   41:             System.setProperty("java.security.policy", "mud.policy");
   42:             System.setSecurityManager( new RMISecurityManager() );
   43:
   44:             // Generate the remote objects
   45:             MUDServiceImpl mudservice = new MUDServiceImpl();
   46:             MUDService mudstub = (MUDService)UnicastRemoteObject.exportObject(muds
ervice, serverPort);
   47:
   48:             String regURL = "rmi://" + hostname + ":" + registryPort + "/MudServic
e";
   49:
   50:             try {
   51:             Naming.rebind(regURL, mudstub);
   52:
   53:             } catch (Exception e){
   54:                 System.out.println(e.getMessage());
   55:             }
   56:             System.out.println("Server is running at "+regURL);
   57:             System.out.println("Launching Admin Mode");
   58:
   59:
   60:             while(true){
   61:
   62:                 String input = in.readLine();
   63:
   64:                 if(input.contains("create")){
   65:                     String[] arguments = input.split(" ");
   66:
   67:                     if(mudservice.Servers.size() < 5){
   68:                         System.out.println("Create a mud with the name " + argumen
ts[1]);
   69:
   70:                         MUD newmud = new MUD(arguments[2],arguments[3],arguments[4
]);
   71:                         mudservice.Servers.put(arguments[1], newmud);
   72:                     } else {
   73:                         System.out.println("Sorry - you can only have 5 MUD's runn
ing at a time");
   74:                     }
   75:                 }
   76:
   77:
   78:             }
   79:
   80:
   81:         }
   82:         catch (java.net.UnknownHostException e) {
   83:             System.err.println("Gannot get local host name.");
   84:         }
   85:         catch (java.io.IOException e){
   86:             System.err.println("Failed to regitser.");
   87:         }
   88:
   89:     }
   90:
   91: }
```

```
    1: /*
    2:  * To change this license header, choose License Headers in Project Properties.
    3:  * To change this template file, choose Tools | Templates
    4:  * and open the template in the editor.
    5:  */
    6: package cs3524.solutions.mud;
    7:
    8: import java.rmi.Remote;
    9: import java.rmi.RemoteException;
   10:
   11: public interface MUDService extends Remote
   12: {
   13:
   14:     public String introduction() throws RemoteException;
   15:
   16:
   17:     public String getStartLocation() throws RemoteException;
   18:     public String location(String location) throws RemoteException;
   19:     public String moveDirection(String current, String direction) throws RemoteExc
eption;
   20:     public boolean addUser(String username) throws RemoteException;
   21:     public void updateUserLocation(String username, String location) throws Remote
Exception ;
   22:     public String getPlayersAtLocation(String location) throws RemoteException;
   23:     public boolean takeItem(String item, String location) throws RemoteException;
   24:     public void changeMUD(String name) throws RemoteException;
   25:     public String getServers() throws RemoteException;
   26:
   27:
   28: }
```

```
/************************************************************************
 * cs3524.solutions.mud.MUD
 ************************************************************************/

package cs3524.solutions.mud;

import java.io.FileReader;
import java.io.BufferedReader;
import java.io.IOException;
import java.util.StringTokenizer;

import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.Vector;
import java.util.HashMap;

/**
 * A class that can be used to represent a MUD; essenially, this is a
 * graph.
 */

public class MUD
{
    /**
     * Private stuff
     */

    public boolean _setup = false;

    // A record of all the vertices in the MUD graph. HashMaps are not
    // synchronized, but we don't really need this to be synchronised.
    public Map<String,Vertex> vertexMap = new HashMap<String,Vertex>();

    private String _startLocation = "";

    public Map<String,String> users = new HashMap<String,String>();

    /**
     * Add a new edge to the graph.
     */
    private void addEdge( String sourceName,
                          String destName,
                          String direction,
                          String view )
    {
        Vertex v = getOrCreateVertex( sourceName );
        Vertex w = getOrCreateVertex( destName );
        v._routes.put( direction, new Edge( w, view ) );
    }

    /**
     * Create a new thing at a location.
     */
    private void createThing( String loc,
                              String thing )
    {
        Vertex v = getOrCreateVertex( loc );
        v._things.add( thing );
    }

    /**
     * Change the message associated with a location.
     */
    private void changeMessage( String loc, String msg )
    {
        Vertex v = getOrCreateVertex( loc );
        v._msg = msg;
    }

    /**
     * If vertexName is not present, add it to vertexMap.  In either
     * case, return the Vertex. Used only for creating the MUD.
     */
    private Vertex getOrCreateVertex( String vertexName )
    {
        Vertex v = vertexMap.get( vertexName );
        if (v == null) {
            v = new Vertex( vertexName );
            vertexMap.put( vertexName, v );
        }
        return v;
    }

    /**
     *
     */
    public Vertex getVertex( String vertexName )
    {
        return vertexMap.get( vertexName );
    }

    /**
     * Creates the edges of the graph on the basis of a file with the
     * following fromat:
     * source direction destination message
     */
    private void createEdges( String edgesfile )
    {
        try {
            FileReader fin = new FileReader( edgesfile );
            BufferedReader edges = new BufferedReader( fin );
            String line;
            while((line = edges.readLine()) != null) {
                StringTokenizer st = new StringTokenizer( line );
                if( st.countTokens( ) < 3 ) {
                    System.err.println( "Skipping ill-formatted line " + line );
                    continue;
                }
                String source = st.nextToken();
                String dir    = st.nextToken();
                String dest   = st.nextToken();
                String msg = "";
                while (st.hasMoreTokens()) {
                    msg = msg + st.nextToken() + " ";
                }
                addEdge( source, dest, dir, msg );
            }
        }
        catch( IOException e ) {
            System.err.println( "Graph.createEdges( String " +
                                edgesfile + ")\n" + e.getMessage() );
        }
    }

    /**
     * Records the messages assocated with vertices in the graph on
     * the basis of a file with the following format:
     * location message
     * The first location is assumed to be the starting point for
     * users joining the MUD.
     */
    private void recordMessages( String messagesfile )
    {
```

```
        try {
            FileReader fin = new FileReader( messagesfile );
            BufferedReader messages = new BufferedReader( fin );
            String line;
            boolean first = true; // For recording the start location.
            while((line = messages.readLine()) != null) {
                StringTokenizer st = new StringTokenizer( line );
                if( st.countTokens( ) < 2 ) {
                    System.err.println( "Skipping ill-formatted line " + line );
                    continue;
                }
                String loc = st.nextToken();
                String msg = "";
                while (st.hasMoreTokens()) {
                    msg = msg + st.nextToken() + " ";
                }
                changeMessage( loc, msg );
                if (first) {      // Record the start location.
                    _startLocation = loc;
                    System.out.println(_startLocation);
                    first = false;
                }
            }
        }
        catch( IOException e ) {
            System.err.println( "Graph.recordMessages( String " +
                                messagesfile + ")\n" + e.getMessage() );
        }
    }

    /**
     * Records the things assocated with vertices in the graph on
     * the basis of a file with the following format:
     * location thing1 thing2 ...
     */
    private void recordThings( String thingsfile )
    {
        try {
            FileReader fin = new FileReader( thingsfile );
            BufferedReader things = new BufferedReader( fin );
            String line;
            while((line = things.readLine()) != null) {
                StringTokenizer st = new StringTokenizer( line );
                if( st.countTokens( ) < 2 ) {
                    System.err.println( "Skipping ill-formatted line " + line );
                    continue;
                }
                String loc = st.nextToken();
                while (st.hasMoreTokens()) {
                    addThing( loc, st.nextToken());
                }
            }
        }
        catch( IOException e ) {
            System.err.println( "Graph.recordThings( String " +
                                thingsfile + ")\n" + e.getMessage() );
        }
    }

    /**
     * All the public stuff. These methods are designed to hide the
     * internal structure of the MUD. Could declare these on an
     * interface and have external objects interact with the MUD via
     * the interface.
     */

    /**
     * A constructor that creates the MUD.
     */
    public MUD( String edgesfile, String messagesfile, String thingsfile )
    {
        _setup = true;
        createEdges( edgesfile );
        recordMessages( messagesfile );
        recordThings( thingsfile );

        System.out.println( "Files read..." );
        System.out.println( vertexMap.size( ) + " vertices\n" );
    }

    // This method enables us to display the entire MUD (mostly used
    // for testing purposes so that we can check that the structure
    // defined has been successfully parsed.
    public String toString()
    {
        String summary = "";
        Iterator iter = vertexMap.keySet().iterator();
        String loc;
        while (iter.hasNext()) {
            loc = (String)iter.next();
            summary = summary + "Node: " + loc;
            summary += ((Vertex)vertexMap.get( loc )).toString();
        }
        summary += "Start location = " + _startLocation;
        return summary;
    }

    /**
     * A method to provide a string describing a particular location.
     */
    public String locationInfo( String loc )
    {
        return getVertex( loc ).toString();
    }

    /**
     * Get the start location for new MUD users.
     */
    public String startLocation()
    {
        return _startLocation;
    }

    /**
     * Add a thing to a location; used to enable us to add new users.
     */
    public void addThing( String loc,
                          String thing )
    {
        Vertex v = getVertex( loc );
        v._things.add( thing );
    }

    /**
     * Remove a thing from a location.
     */
    public void delThing( String loc,
                          String thing )
    {
        Vertex v = getVertex( loc );
        v._things.remove( thing );
    }

    /**
```

```
  269:      * A method to enable a player to move through the MUD (a player
  270:      * is a thing). Checks that there is a route to travel on. Returns
  271:      * the location moved to.
  272:      */
  273:     public String moveThing( String loc, String dir, String thing )
  274:     {
  275:         Vertex v = getVertex( loc );
  276:         Edge e = v._routes.get( dir );
  277:         if (e == null)   // if there is no route in that direction
  278:             return loc;  // no move is made; return current location.
  279:         v._things.remove( thing );
  280:         e._dest._things.add( thing );
  281:         return e._dest._name;
  282:     }
  283:
  284:     /**
  285:      * A main method that can be used to testing purposes to ensure
  286:      * that the MUD is specified correctly.
  287:      */
  288:     public static void main(String[] args)
  289:     {
  290:         if (args.length != 3) {
  291:             System.err.println("Usage: java Graph <edgesfile> <messagesfile> <thin
gsfile>");
  292:             return;
  293:         }
  294:         MUD m = new MUD( args[0], args[1], args[2] );
  295:         System.out.println( m.toString() );
  296:     }
  297: }
```

```
    1: /*
    2:  * To change this license header, choose License Headers in Project Properties.
    3:  * To change this template file, choose Tools | Templates
    4:  * and open the template in the editor.
    5:  */
    6: package cs3524.solutions.mud;
    7:
    8: import java.rmi.*;
    9: import java.util.ArrayList;
   10: import java.util.HashMap;
   11: import java.util.Iterator;
   12: import java.util.List;
   13: import java.util.Map;
   14:
   15: /**
   16:  *
   17:  * @author darren
   18:  */
   19: public class MUDServiceImpl implements MUDService {
   20:
   21:     private MUD m;
   22:     public Map<String, MUD> Servers = new HashMap<String, MUD>();
   23:
   24:
   25:     public MUDServiceImpl()    throws RemoteException
   26:     {
   27:         Servers.put("demo", new MUD("mymud.edg","mymud.msg","mymud.thg"));
   28:         Servers.put("demo2", new MUD("mymud.edg","mymud.msg","mymud.thg"));
   29:     }
   30:
   31:     public String introduction() throws RemoteException
   32:     {
   33:         if(m==null){
   34:             m = Servers.get("demo");
   35:         }
   36:         String output = ( "============================== \n \n Welcome to the MU
D Server! \n \n============================== \n" );
   37:         output += "Please enter a username: ";
   38:
   39:         return output;
   40:
   41:     }
   42:
   43:     public String getStartLocation() throws RemoteException {
   44:         return m.startLocation();
   45:     }
   46:
   47:
   48:     public String location(String location) throws RemoteException{
   49:
   50:         return m.getVertex(location).toString();
   51:
   52:     }
   53:
   54:     public String moveDirection(String current, String direction) throws RemoteExc
eption{
   55:         Vertex currentVertex = m.getVertex(current);
   56:         if(currentVertex._routes.containsKey(direction)){
   57:             Edge newLocation = currentVertex._routes.get(direction);
   58:             Vertex newVert = (newLocation._dest);
   59:             //System.out.print(newVert._name);
   60:
   61:         return newVert._name;
   62:         } else {
   63:             return current;
   64:         }
   65:     }
   66:
   67:
   68:     public boolean addUser(String username) throws RemoteException {
   69:
   70:         if(m.users.size() < 10){
   71:             m.users.put(username, m.startLocation());
   72:             return true;
   73:         } else {
   74:             return false;
   75:         }
   76:     }
   77:
   78:     public void updateUserLocation(String username, String location) throws Remote
Exception {
   79:         m.users.remove(username);
   80:         m.users.put(username, location);
   81:
   82:         //System.out.println(m.users);
   83:
   84:     }
   85:
   86:     public String getPlayersAtLocation(String location) throws RemoteException{
   87:
   88:         ArrayList<String> Players = new ArrayList<String>();
   89:         String username;
   90:
   91:         StringBuilder sb = new StringBuilder();
   92:
   93:         Iterator itter = m.users.keySet().iterator();
   94:
   95:         while (itter.hasNext()) {
   96:             username = itter.next().toString();
   97:             if(m.users.get(username).equalsIgnoreCase(location)){
   98:                 Players.add(username);
   99:                 sb.append(username);
  100:                 sb.append(", ");
  101:             }
  102:
  103:         }
  104:
  105:         sb.setLength(sb.length() - 2);
  106:
  107:         return "You can see: " + sb.toString();
  108:
  109:     }
  110:
  111:     public boolean takeItem(String item, String location) throws RemoteException {
  112:         Vertex currentVertex = m.getVertex(location);
  113:         List<String> things = currentVertex._things;
  114:         if(things.contains(item)){
  115:             m.delThing(location, item);
  116:
  117:             return true;
  118:         }
  119:
  120:         return false;
  121:     }
  122:
  123:     public void changeMUD(String name) throws RemoteException {
  124:         //System.out.println("Server is changing to " + name);
  125:         m = Servers.get(name);
  126:
  127:     }
  128:
  129:     public String getServers() throws RemoteException{
  130:
  131:         StringBuilder sb = new StringBuilder();
```

```
132:         Iterator it = Servers.keySet().iterator();
133:
134:         while(it.hasNext()){
135:             sb.append(it.next().toString());
136:             sb.append(", ");
137:         }
138:
139:         sb.setLength(sb.length() - 2);
140:
141:         return "Currently running servers: " + sb.toString();
142:     }
143:
144:
145:
146: }
```