```java
package mud;

import java.util.*;

/**
 * @author stefanrudvin 51549217
 *
 * Class modified from practicals.package practicals.rmishout.ShoutServiceImpl
 */
public class MUDServiceImpl implements MUDServiceInterface {

    /**
        Current instance of the MUD
     */
    private MUD MUDInstance;
    private String MUDInstanceName;

    /**
     * Hash map of all available MUDs
     */
    private Map<String, MUD> Muds = new HashMap<>();

    /**
        Current players in MUD. playerName : <Timeout, Location>
     */
    private Map<String, String[]> Players = new HashMap<>();

    /**
        These values can be changed to allow for server size manipulation.
     */
    public static int MAX_NUM_OF_MUDS = 5;
    public static int MAX_TOTAL_PLAYERS = 100;
    public static int MAX_PLAYERS_PER_MUD = 10;


    /**
     * Stores messages that are returned to the client. E.g. when player disconnects, times out
```

```
37  or joins a mud.
38       */
39      private String broadcastMessage = "";
40
41      public MUDServiceImpl() {
42          /*
43              Create sample MUDs when server starts
44           */
45          Muds.put("sample", new MUD("maps/sample.edg", "maps/sample.msg", "maps/sample.thg"));
46          Muds.put("aberdeen", new MUD("maps/aberdeen.edg", "maps/aberdeen.msg", "maps/aberdeen.
    thg"));
47          Muds.put("aberdeen2", new MUD("maps/aberdeen.edg", "maps/aberdeen.msg", "maps/aberdeen.
    thg"));
48
49          /*
50              Start player timeout function.
51              This runs every 0.5 seconds, and after it runs 10 times without a player ping
52              the player is disconnected from the server,
53              removed from the Players object and the
54              users object in the mud instance.
55           */
56
57          Timer timerObj = new Timer(true);
58          timerObj.scheduleAtFixedRate(new TimerTask() {
59              @Override
60              public void run() {
61                  decrementPlayerTimeOut();
62              }
63          }, 10, 500);
64      }
65
66      public Integer setMaxPlayersPerMud(int maxPlayersPerMud) {
67          MAX_PLAYERS_PER_MUD = maxPlayersPerMud;
68          return MAX_PLAYERS_PER_MUD;
69      }
70
71      public Integer setMaxTotalPlayers(int maxTotalPlayers) {
```

```
 72          MAX_TOTAL_PLAYERS = maxTotalPlayers;
 73          return MAX_TOTAL_PLAYERS;
 74      }
 75
 76      public Integer setMaxNumOfMuds(int maxNumOfMuds) {
 77          MAX_NUM_OF_MUDS = maxNumOfMuds;
 78          return MAX_NUM_OF_MUDS;
 79      }
 80
 81      /**
 82       *  This method runs every 500ms, decrementing the timeout variable of the players object
    to time out players.
 83       */
 84      public void decrementPlayerTimeOut() {
 85
 86          Iterator<Map.Entry<String, String[]>> iter = Players.entrySet().iterator();
 87          while (iter.hasNext()) {
 88              Map.Entry<String, String[]> entry = iter.next();
 89
 90              String playerName = entry.getKey();
 91
 92              Integer timeOut = Integer.parseInt(entry.getValue()[0]);
 93              timeOut--;
 94
 95              entry.getValue()[0] = Integer.toString(timeOut);
 96
 97              String mud = entry.getValue()[1];
 98
 99              if (timeOut <= 0) {
100                  disconnectPlayer(mud, playerName);
101                  iter.remove();
102              }
103          }
104      }
105
106
107      /**
```

```
108       * This method allows the client to refresh the players object so that the player does not
      get timed out.
109       * @param playerName String
110       * @return String
111       */
112      public String refreshPlayerTimeOut(String playerName) {
113          String[] player = Players.get(playerName);
114          player[0] = Integer.toString(10);
115          return broadcastMessage;
116      }
117
118      /**
119       * Disconnect player and send message to all players
120       * @param mud MUD to disconnect from
121       * @param playerName String
122       */
123      private void disconnectPlayer(String mud, String playerName) {
124          broadcastMessage = "Player " + playerName + " has timed out in MUD: " + mud;
125          System.out.println(broadcastMessage);
126          MUDInstance.users.remove(playerName);
127      }
128
129      /**
130       *  Allows client to create a new MUD with a custom name.
131       * @param name String
132       * @return boolean
133       */
134      public boolean createMUD(String name) {
135          if (MAX_NUM_OF_MUDS > Muds.size()) {
136              Muds.put(name, new MUD("maps/aberdeen.edg", "maps/aberdeen.msg", "maps/aberdeen.
      thg"));
137              return true;
138          }
139          return false;
140      }
141
142      /**
```

```
143       * Get total number of players
144       * @return String
145       */
146      public String getPlayerTotal() {
147          return Integer.toString(Players.size());
148      }
149
150      /**
151       * Get total number of MUDs running
152       * @return String
153       */
154      public String getMUDTotal() {
155          return Integer.toString(Muds.size());
156      }
157
158      /**
159       * Get total number of players in current MUD
160       * @return String
161       */
162      public String getMUDPlayerTotal() {
163          return Integer.toString(MUDInstance.users.size());
164      }
165
166      /**
167       * Get start location of player in current MUD
168       * @return String
169       */
170      public String getStartLocation() {
171          return MUDInstance.startLocation();
172      }
173
174      /**
175       * Get information about the current location of the player
176       * @param location String
177       * @return String
178       */
179      public String getLocationInfo(String location) {
```

```
180         return MUDInstance.getVertex(location).toString();
181     }
182 
183     /**
184      * Move player in a specific direction
185      * @param currentLocation String
186      * @param direction String
187      * @param playerName String
188      * @return String
189      */
190     public String moveDirection(String currentLocation, String direction, String playerName) {
191         return MUDInstance.moveThing(currentLocation, direction, playerName);
192     }
193 
194     /**
195      * Initialize the player in the service and send a message to all players
196      * @param playerName String
197      * @param serverName String
198      * @return boolean
199      */
200     public boolean initializePlayer(String playerName, String serverName) {
201         if ((MUDInstance.users.size() < MAX_PLAYERS_PER_MUD) && (Players.size() <
    MAX_TOTAL_PLAYERS)) {
202 
203             String[] playerInfoArray = {"10", serverName};
204 
205             Players.put(playerName, playerInfoArray);
206 
207             broadcastMessage = "Player " + playerName + " has joined the game.";
208 
209             MUDInstance.addThing(MUDInstance.startLocation(), playerName);
210             MUDInstance.users.put(playerName, MUDInstance.startLocation());
211             return true;
212         }
213         return false;
214     }
215 
```

```
216     /**
217      * Get all objects at current player location
218      * @param location String
219      * @return String
220      */
221     public String getObjectsAtLocation(String location) {
222         return MUDInstance.locationInfo(location);
223     }
224 
225     /**
226      * Take a selected item at a location
227      * @param item String
228      * @param location String
229      */
230     public void takeItem(String item, String location) {
231         MUDInstance.deleteThing(location, item);
232     }
233 
234     /**
235      * Drop an item to a location
236      * @param item String
237      * @param location String
238      */
239     public void dropItem(String item, String location) {
240         MUDInstance.addThing(location, item);
241     }
242 
243     /**
244      * Change the current MUD to another specified one
245      * @param MUDName String
246      * @return String
247      */
248     public String changeMUD(String MUDName, String playerName) {
249         MUDInstanceName = MUDName;
250         MUDInstance = Muds.get(MUDName);
251 
252         broadcastMessage = "Player " + playerName + " has joined MUD: " + MUDName;
```

```
253
254         return MUDName;
255     }
256
257     /**
258      *  Exit the current MUD, send message
259      * @param playerName String
260      * @param location String
261      */
262     public void exitMUD(String playerName, String location) {
263         broadcastMessage = "Player " + playerName + " has left the game.";
264         Players.remove(playerName);
265         MUDInstance.users.remove(playerName);
266         takeItem(playerName, location);
267     }
268
269     /**
270      * Return a string
271      * @return String
272      */
273     public String getMudsString() {
274         return Muds.keySet().toString().replaceAll("[\\[]]", "").replaceAll(",", " | ");
275     }
276
277     /**
278      *  Show welcome string specific to current MUDInstance
279      * @return String
280      */
281     public String welcome(){
282         return
    "=================================================================================
    ==================================== \n" +
283                 "                                                    Welcome to " +
    MUDInstanceName + "!                                          \n" +
284                 "                                                        " + getPlayerTotal() +
    " Total players online                                \n" +
285                 "                                                      " + getMUDPlayerTotal() +
```

```
285 " Players online in this MUD                                    \n" +
286
    "=============================================================================================
    ====================================";
287     }
288 }
289
```