```java
package mud;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.rmi.Naming;
import java.rmi.RMISecurityManager;
import java.rmi.RemoteException;
import java.util.ArrayList;
import java.util.List;
import java.util.Timer;
import java.util.TimerTask;

/**
 * @author stefanrudvin 51549217
 *
 * Class modified from practicals.package practicals.rmishout.ShoutServiceClient
 */
public class MUDClient {

    private static BufferedReader in = new BufferedReader(new InputStreamReader(System.in));

    /**
     * Store variables of current game session
     */
    private static String playerName;
    private static String serverName = "aberdeen";

    private static String playerLocation;
    private static MUDServiceInterface MUDService;

    /**
     * Specify whether the game is running, i.e. accept player input
     */
    private static boolean running = true;

    /**
```

```
38       * Stores all items the player is carrying
39       */
40      private static List<String> items = new ArrayList<>();
41
42      /**
43       * Store current message which shows information about players leaving, timing out or
    joining a MUD
44       */
45      private static String broadcastMessage = "";
46
47      /**
48       * Specify whether the player is currently changing MUD. This stops the client pinging a
    non-existent player.
49       */
50      private static Boolean changingMUD = false;
51
52      /**
53       * Class modified from practicals.rmishout.ShoutServerMainline.java
54       */
55
56      public static void main(String args[]) {
57
58          if (args.length < 2) {
59              System.err.println("Usage:\njava MUDClient <host> <port>");
60              return;
61          }
62
63          String hostname = args[0];
64          int registryPort = Integer.parseInt(args[1]);
65
66          System.setProperty("java.security.policy", "mud.policy");
67          System.setSecurityManager(new RMISecurityManager());
68
69          try {
70              String regURL = "rmi://" + hostname + ":" + registryPort + "/MUDService";
71              System.out.println("Looking up " + regURL);
72
```

```
 73              // Setup service
 74              MUDService = (MUDServiceInterface) Naming.lookup(regURL);
 75
 76              initialize();
 77
 78              runGame();
 79
 80          } catch (java.io.IOException e) {
 81              System.err.println("I/O error.");
 82              System.err.println(e.getMessage());
 83          } catch (java.rmi.NotBoundException e) {
 84              System.err.println("Server not bound.");
 85              System.err.println(e.getMessage());
 86          }
 87      }
 88
 89      /**
 90       * Initialize the game before running
 91       */
 92      private static void initialize() throws RemoteException {
 93
 94          printMainIntro();
 95
 96          selectPlayerName();
 97
 98          showMudsAndJoin();
 99
100          startServerPing();
101
102          printCurrentMudIntro();
103
104          playerLocation = MUDService.getStartLocation();
105      }
106
107      /**
108       * Select playerName throughout session.
109       */
```

```
110     private static void selectPlayerName() {
111         System.out.println("Enter your playerName:");
112         try {
113             printCarets();
114             playerName = in.readLine();
115         } catch (IOException e) {
116             e.printStackTrace();
117         }
118         System.out.println("");
119     }
120 
121     /**
122      * Show available muds and allow player to join one
123      */
124     private static void showMudsAndJoin() throws RemoteException {
125         showMuds();
126 
127         joinMUD();
128     }
129 
130     /**
131      * Show current MUDS
132      */
133     private static void showMuds() throws RemoteException {
134         System.out.println("Here are the available MUDs:");
135         System.out.println(MUDService.getMudsString());
136     }
137 
138     /**
139      * Allow player to join mud. If it is full, allow player to wait or try again indefinitely
140      */
141     private static void joinMUD() throws RemoteException {
142         System.out.println("Select which MUD to join:");
143 
144         boolean accepted = false;
145 
146         while (!accepted) {
```

```
147             selectMud();
148             // Try to join until accepted
149             if (tryJoinMUD()) {
150                 accepted = true;
151             } else {
152                 System.out.println("Sorry, this or all servers are full. Please wait or try another:");
153             }
154         }
155     }
156
157     /**
158      * Allow player to select which MUD to join
159      */
160     private static void selectMud() {
161         try {
162             printCarets();
163             serverName = in.readLine();
164             MUDService.changeMUD(serverName, playerName);
165         } catch (IOException e) {
166             e.printStackTrace();
167         }
168     }
169
170     /**
171      * Try to join a mud, and return boolean on whether join was successful
172      */
173     private static boolean tryJoinMUD() throws RemoteException {
174         return MUDService.initializePlayer(playerName, serverName);
175     }
176
177     /**
178      * Begin pinging player object in server to avoid timeout. This is paused when player changes MUDs with changingMUD
179      */
180     private static void startServerPing() {
181         Timer timerObj = new Timer(true);
```

```
182         timerObj.scheduleAtFixedRate(new TimerTask() {
183             @Override
184             public void run() {
185                 try {
186                     if (!changingMUD) {
187                         refreshTimeOutAndRetrieveBroadcast();
188                     }
189 
190                 } catch (RemoteException e) {
191                     e.printStackTrace();
192                 }
193             }
194         }, 10, 500);
195     }
196 
197     /**
198      * Refresh player timeout and printout broadcast message if it is different form the current one.
199      */
200     private static void refreshTimeOutAndRetrieveBroadcast() throws RemoteException {
201         String message = MUDService.refreshPlayerTimeOut(playerName);
202 
203         if (!message.equals(broadcastMessage) && !message.equals("")) {
204             broadcastMessage = message;
205             System.out.println(message);
206             printCarets();
207         }
208     }
209 
210     /**
211      * Print intro for the current Mud
212      */
213     private static void printCurrentMudIntro() throws RemoteException {
214         clearConsole();
215 
216         System.out.println(MUDService.welcome());
217     }
```

```
218
219     /**
220      * Change the current mud player is playing in
221      */
222     private static void changeMUD() throws RemoteException {
223
224         showMudsAndJoin();
225
226         changingMUD = false;
227
228         printCurrentMudIntro();
229
230         playerLocation = MUDService.getStartLocation();
231     }
232
233     /**
234      * Main game loop for playing
235      */
236     private static void runGame() throws RemoteException {
237
238         printHelp();
239
240         printCurrentLocationInfo();
241
242         while (running) try {
243             System.out.println("");
244             printCarets();
245             String choice = in.readLine().toLowerCase();
246
247             handlePlayerActions(choice);
248
249         } catch (IOException e) {
250             e.printStackTrace();
251         }
252
253         System.out.println("Game exit.");
254         System.out.println("Catch you next time!");
```

```
255             printMainIntro();
256         }
257 
258         /**
259          * Handle user taking an object
260          * @param choice String
261          * @return Boolean
262          */
263         private static Boolean handleTakeAction(String choice) throws RemoteException  {
264             if (choice.contains("take")) {
265                 String item = parseChoice(choice);
266 
267                 MUDService.takeItem(item, playerLocation);
268 
269                 System.out.println("You have picked up: " + item);
270                 items.add(item);
271 
272                 return true;
273             }
274             return false;
275         }
276 
277         /**
278          * Handle user moving in the game
279          * @param choice String
280          * @return Boolean
281          */
282         private static Boolean handleMoveAction(String choice) throws RemoteException {
283             if (choice.contains("move")) {
284 
285                 String direction = parseChoice(choice);
286 
287                 String location = MUDService.moveDirection(playerLocation, direction, playerName);
288 
289                 if (location.equals(playerLocation)) {
290                     System.out.println("There is an obstacle in the way. Try another direction:");
291                     return true;
```

```
292
293             } else {
294                 playerLocation = location;
295                 System.out.println("You have moved to location: " + location);
296                 printCurrentLocationInfo();
297                 return true;
298             }
299         }
300         return false;
301     }
302
303     /**
304      * Handle user dropping an item
305      * @param choice String
306      * @return Boolean
307      */
308     private static Boolean handleDropAction(String choice) throws RemoteException {
309         if (choice.contains("drop")) {
310             String item = parseChoice(choice);
311
312             MUDService.dropItem(item, playerLocation);
313
314             System.out.println("You dropped: " + item);
315             items.remove(item);
316             return true;
317         }
318         return false;
319     }
320
321     /**
322      * Handle all the available actions a player can make
323      */
324     private static void handlePlayerActions(String choice) throws IOException {
325
326         if (handleMoveAction(choice) || handleTakeAction(choice) || handleDropAction(choice))
    {
327             return;
```

```
328         }
329
330         switch (choice) {
331             case ("help"):
332                 printHelp();
333                 break;
334             case "whoami":
335                 System.out.println(playerName);
336                 break;
337             case "look":
338                 System.out.println(MUDService.getObjectsAtLocation(playerLocation));
339                 break;
340             case "items":
341                 System.out.println("Contents of inventory: ");
342                 items.forEach(System.out::println);
343                 break;
344             case "exit":
345                 exitMUD();
346                 running = false;
347                 break;
348             case "setmaxtotalplayers":
349                 handleSetMaxTotalPlayers();
350                 break;
351             case "setmaxmudplayers":
352                 handleSetMaxMudPlayers();
353                 break;
354             case "setmaxmuds":
355                 handleSetMaxMuds();
356                 break;
357             case "mudlist":
358                 showMuds();
359                 break;
360             case "mudmigrate":
361                 exitMUD();
362                 changeMUD();
363                 printCurrentLocationInfo();
364                 break;
```

```
365                 case "mudcreate":
366                     handleCreateMud();
367                     break;
368                 case "mudtotal":
369                     System.out.println(MUDService.getMUDTotal());
370                     break;
371                 case "mudplayers":
372                     System.out.println(MUDService.getMUDPlayerTotal());
373                     break;
374                 case "mudtotalplayers":
375                     System.out.println(MUDService.getPlayerTotal());
376                     break;
377                 default:
378                     printHelp();
379                     break;
380             }
381         }
382 
383         private static void handleSetMaxTotalPlayers() throws IOException {
384             System.out.println("Enter maximum total player allowance: ");
385 
386             printCarets();
387 
388             String choice = in.readLine().toLowerCase();
389 
390             System.out.println("Maximum player allowance set to: " + MUDService.setMaxTotalPlayers
    (Integer.parseInt(choice)));
391         }
392 
393         private static void handleSetMaxMudPlayers() throws IOException {
394             System.out.println("Enter maximum total player allowance per MUD: ");
395 
396             printCarets();
397 
398             String choice = in.readLine().toLowerCase();
399 
400             System.out.println("Maximum player allowance per MUD set to: " + MUDService.
```

```
400 setMaxPlayersPerMud(Integer.parseInt(choice)));
401     }
402 
403     private static void handleSetMaxMuds() throws IOException {
404         System.out.println("Enter maximum MUD allowance: ");
405 
406         printCarets();
407 
408         String choice = in.readLine().toLowerCase();
409 
410         System.out.println("Maximum MUD allowance set to: " + MUDService.setMaxNumOfMuds(
    Integer.parseInt(choice)));
411     }
412 
413     /**
414      *  Handle player exciting a MUD - drops all items, sends request to server and pauses
    timeout ping
415      */
416     private static void exitMUD() throws RemoteException {
417 
418         // Drop all items
419         for (String item : items) {
420             MUDService.dropItem(item, playerLocation);
421         }
422         changingMUD = true;
423 
424         // Remove from players and items in MUD
425         MUDService.exitMUD(playerName, playerLocation);
426     }
427 
428     /**
429      * Return second parameter of String
430      * @param choice String
431      */
432     private static String parseChoice(String choice) {
433         String[] splitChoice = choice.split("\\s+");
434         return splitChoice[1];
```

```
435     }
436
437     /**
438      * Allow player to create a new MUD
439      */
440     private static void handleCreateMud() {
441         System.out.println("Select name of new MUD:");
442
443         try {
444             printCarets();
445             String name = in.readLine();
446
447             if (MUDService.createMUD(name)) {
448                 System.out.println("New MUD created with name: " + name + ".");
449             } else {
450                 System.out.println("Maximum MUD limit reached.");
451             }
452
453         } catch (IOException e) {
454             e.printStackTrace();
455         }
456     }
457
458     /**
459      * Print information about current player location
460      */
461     private static void printCurrentLocationInfo() throws RemoteException {
462         System.out.println(MUDService.getLocationInfo(playerLocation));
463     }
464
465     /**
466      * Print help menu to show available commands
467      */
468     private static void printHelp() {
469         System.out.println("Available commands:");
470         System.out.println("Move <direction> - move to a selected direction (North/East/South/
    West)");
```

```
471         System.out.println("Take <item>       - Take selected item into your inventory");
472         System.out.println("Drop <item>       - Drop selected item to the ground");
473         System.out.println("Items             - Show items in your inventory");
474         System.out.println("Whoami            - Show your player playerName");
475         System.out.println("Look              - Show items, players and paths at current
    location");
476         System.out.println("Help              - Show this help menu");
477         System.out.println("Exit              - Exit the game");
478 
479         System.out.println("");
480 
481         System.out.println("MUDlist           - Show list of current MUDs");
482         System.out.println("MUDmigrate        - Move to another MUD");
483         System.out.println("MUDcreate         - Create a whole new MUD");
484 
485         System.out.println("MUDtotal          - Show total number of MUDs");
486         System.out.println("MUDplayers        - Show number of players in current MUD");
487         System.out.println("MUDtotalplayers   - Show total player number throughout all MUDs");
488 
489         System.out.println("");
490 
491         System.out.println("setmaxmuds          - Set maximum number of MUDs");
492         System.out.println("setmaxmudplayers    - Set maximum total players per MUD");
493         System.out.println("setmaxtotalplayers - Set maximum total players in the game");
494     }
495 
496     /**
497      * Draw a set of carets to aid in player input
498      */
499     private static void printCarets() {
500         System.out.print(">> ");
501     }
502 
503     /**
504      * Clear console window. May not work on windows machines.
505      */
506     private static void clearConsole() {
```

```
507         System.out.print("\033[H\033[2J");
508     }
509 
510     /**
511      * Print 'From dusk till dawn' ASCII art intro
512      */
513     private static void printMainIntro() {
514         clearConsole();
515         System.out.println(
    "=======================================================================================
    ====================================");
516 
517         System.out.println(
518                 "____________                                 _________                  __      ____________.
    __.__   .__     _________                                      \n" +
519                 "\\_    _____/_____  ____   _____    \\_______ \\   __ __   _____|   | __
    \\__     ___/|__|   | |   |    \\_______ \\ _____ __   _  _______   \n" +
520                 " |     __) \\_  __ \\/  _ \\ /     \\    |     |  \\|   |  \\/  ___
    /   |/ /    |     |    |   |  | |   |     |     |  \\\\__  \\\\ \\/ \\/ /     \\ \n" +
521                 " |      \\    |   | \\(  <_> )  Y Y  \\   |     `    \\   |   /\\___ \\
    |     <     |     |    |   |_|   |__   |     `    \\/ __ \\\\      /    |   \\\n" +
522                 " \\___   /    |__|    \\____/|__|_|   / /________   /____//____   >__|_ \\
       |____|    |__|____/____/ /________   (____   /\\/\\_/|___|   /\n" +
523                 "     \\/                          \\/           \\/            \\/      \\
    /                                  \\/      \\/            \\/ ");
524 
525         System.out.println(
    "=======================================================================================
    ====================================");
526     }
527 
528 }
529 
```