```
/**********************************************************************
 * mud.MUDClient
 **********************************************************************/

package mud;

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

/*
  MUD game made by Dovydas Pekus, University of Aberdeen

  This is the main game client file, which gets all the inputs from the player
  and displays the output from the server
*/


public class MUDClient {

  // prepare the input reader
  private static BufferedReader in = new BufferedReader(new InputStreamReader(Sy
stem.in));

  // the player's name that they will be using throughout the game
  private static String playerName = "";

  // specify whether the game is running or not
  private static boolean running = false;

  // the name of the MUD that the player is currently on
  private static String mudName = "";

  // remote server interface
  private static MUDServerInterface serv;

  // current location of the player
  private static String currentLocation = "";

  // the items that the player is currently carrying
  private static List<String> inventory = new ArrayList<>();

  // main game class
  public static void main(String args[]) throws RemoteException {

    if (args.length < 2) {
      System.err.println("Usage:\njava MUDClient <host> <port>");
      return;
    }

    String hostname = args[0];
    int port = Integer.parseInt(args[1]);

    System.setProperty("java.security.policy", "mud.policy");
    System.setSecurityManager(new SecurityManager());

    try {
```

```
      String regURL = "rmi://" + hostname + ":" + port + "/MUDServer";
      System.out.println("Looking up " + regURL);

      serv = (MUDServerInterface) Naming.lookup(regURL);

      // prepare the initial MUDs
      serv.initialize();

      // set up the game
      System.out.println("Welcome!");
      System.out.println("What is your name?");
      try {
        System.out.print(">> ");
        playerName = in.readLine();
      } catch (IOException e) {
        System.err.println("I/O error.");
        System.err.println(e.getMessage());
      }
      System.out.println("Nice to meet you, " + playerName);
      System.out.println();
      System.out.println("Let's begin");

      displayAvailableMUDs();

      joinMUD();

      running = true;
      currentLocation = serv.getStartLocation();

      displayOptions();

      runGame();

    } catch (IOException e) {
      System.err.println("I/O error.");
      System.err.println(e.getMessage());
    } catch (java.rmi.NotBoundException e) {
      System.err.println("Server not bound.");
      System.err.println(e.getMessage());
    }
  }

  // runs the whole game, while 'running' variable is true
  // gets the player's input and displays the output
  private static void runGame() throws RemoteException {

    while (running) try {
      System.out.println();
      System.out.print(">> ");
      String playerInput = in.readLine().toLowerCase();
      handlePlayerInput(playerInput);

    } catch (IOException e) {
      System.err.println("I/O error.");
      System.err.println(e.getMessage());
    }
  }

  // handle an input from the player
  private static void handlePlayerInput(String playerInput) throws RemoteExcepti
on {
```

```
    // move the user in a given direction
    if (playerInput.contains("move")) {

      // get the direction where the player wants to move
      String[] directionString = playerInput.split(" ");

      // if the server returns the same location as the player is at right now,
      // it means that there is no path to that direction
      if (currentLocation.equals(serv.moveUser(currentLocation, directionString[
1], playerName))) {
        System.out.println("Sorry, either there isn't a path to this direction,
or this direction is not valid.");
      } else {
        // move the player and display information about the new location
        System.out.println("You are going " + directionString[1] + "...");
        currentLocation = serv.moveUser(currentLocation, directionString[1], pla
yerName);
        System.out.println(serv.getCurrentLocationInfo(currentLocation));
      }
    }

    // pick up an item
    if (playerInput.contains("pick")) {

      // get the name of the item that the player wants to pick up
      String[] itemString = playerInput.split(" ");

      // pick up the item and notify the user about it
      serv.pickUpItem(currentLocation, itemString[1]);
      inventory.add(itemString[1]);
      System.out.println("You have picked up " + itemString[1]);
    }

    // drop an item
    if (playerInput.contains("drop")) {

      // get the name of the item that the player wants to drop
      String[] itemString = playerInput.split(" ");
      serv.dropItem(currentLocation, itemString[1]);
      inventory.remove(itemString[1]);
      System.out.println("You have dropped " + itemString[1]);
    }

    // display the contents of player's inventory
    if (playerInput.equals("inventory")) {

      // if there isn't any items, inform the player that the inventory is empty
      if (inventory.size() < 1) {
        System.out.println("Your inventory is empty.");
      } else {

        //otherwise, list all the items
        System.out.println("You are carrying:");
        for (String item : inventory) {
          System.out.println("* " + item);
        }
      }
    }

    // get the information about player's surroundings
    if (playerInput.equals("location")) {
      System.out.println("You look around...");
```

```
      System.out.println(serv.getCurrentLocationInfo(currentLocation));
    }

    // display the list of players currently playing in the same MUD"
    if (playerInput.equals("players")) {

      // get the list of all players in the MUD
      String[] currentPlayers = serv.getCurrentPlayersInMUD();

      // check if the player is the only player in the MUD and inform them about
 it
      if (currentPlayers.length < 2) {
        System.out.println("You're the only player in this MUD.");
      } else {

        // otherwise, list all the players
        System.out.println("Currently, these players are playing in this MUD: ")
;
        System.out.println();

        for (String name : currentPlayers) {
          System.out.println("* " + name);
        }
      }
    }

    // print the available commands to the player
    if (playerInput.equals("help")) {
      displayOptions();
    }

    // exit the game
    if (playerInput.equals("exit")) {

      // drop all the items that the player was carrying
      for (String item : inventory) {
        serv.dropItem(currentLocation, item);
      }
      inventory.clear();

      serv.exit(playerName);
      running = false;
    }

    // display all available MUDs
    if (playerInput.equals("muds")) {
      displayAvailableMUDs();
    }

    // move to another MUD
    if (playerInput.equals("changemud")) {
      displayAvailableMUDs();
      joinMUD();
      currentLocation = serv.getStartLocation();
      displayOptions();
    }

    // create a new MUD
    if (playerInput.equals("createmud")) {
      System.out.println("Enter the name of your new MUD:");
      try {
        System.out.print(">> ");
```

```
        mudName = in.readLine();
        System.out.println();
      } catch (IOException e) {
        System.err.println("I/O error.");
        System.err.println(e.getMessage());
      }
      createNewMUD(mudName);
    }

    // allow the player to change the number of maximum MUDs in real time
    if (playerInput.equals("changemaxmuds")) {
      Integer maxNumberOfMUDs = serv.getMaxNumberOfMuds();
      System.out.println("Currently, the maximum number of MUDs is " + maxNumber
OfMUDs + ", and there are " + serv.getMUDCount() + " MUDs running.");
      System.out.println("Enter the new maximum number of MUDs (remember, it can
not be lower than the current number of MUDs running):");

      try {
        System.out.print(">> ");
        String input = in.readLine();
        Integer newMaxMUDs = Integer.parseInt(input);
        System.out.println();

        // check if the player's chosen maximum number is not smaller than
        // the current number of MUDs running
        if (newMaxMUDs.compareTo(serv.getMUDCount()) >= 0) {
          serv.setNewMaxNumberOfMUDs(newMaxMUDs);
          System.out.println("The new maximum number of MUDs is " + newMaxMUDs);
        } else {
          System.out.println("Sorry, the maximum number of MUDs cannot be lower
than the number of MUDs currently running.");
        }

      } catch (IOException e) {
        System.err.println("I/O error.");
        System.err.println(e.getMessage());
      }


    }
  }

  // displays all possible command options to the player
  private static void displayOptions() {
    System.out.println();
    System.out.println("You can choose from one of these commands:");
    System.out.println();
    System.out.println("* Move <direction>  - move to a selected direction (nort
h, east, south, west)");
    System.out.println("* Pick <item>  - pick up an item from the ground to your
 inventory");
    System.out.println("* Drop <item>  - drop an item from your inventory to the
 ground");
    System.out.println("* Inventory  - see the items you are carrying");
    System.out.println("* Location  - display the information about your surroun
dings");
    System.out.println("* Players  - display the list of players currently playi
ng in the same MUD");
    System.out.println("* Help  - display the available commands");
    System.out.println("* Muds  - display all currently available MUDs");
    System.out.println("* ChangeMUD  - move to another MUD");
    System.out.println("* CreateMUD  - create a new MUD");
```

```
    System.out.println("* Changemaxmuds  - change the number of maximum MUDS");
    System.out.println("* Exit  - exit the game");
  }

  // displays all currently available MUDs to the player
  private static void displayAvailableMUDs() throws RemoteException {
    Integer mudCount = serv.getMUDCount();
    System.out.println("Currently, there are " + mudCount + " MUDs are available
: ");
    System.out.println();
    String[] availableMUDs = serv.getAvailableMUDs();
    for (String mud : availableMUDs) {
      System.out.println("* " + mud);
    }
    System.out.println();
  }

  // allows the player to join one of the existing MUDs
  private static void joinMUD() throws RemoteException {
    // get the name of the MUD that the player wishes to join
    System.out.println("Which MUD would you like to join?");
    System.out.println();
    try {
      System.out.print(">> ");
      mudName = in.readLine();
    } catch (IOException e) {
        System.err.println("I/O error.");
        System.err.println(e.getMessage());
    }

    // check if such MUD exists
    // if not, prompt the player to reenter the name
    if (!serv.checkIfMUDExists(mudName)) {
      System.out.println("Sorry, no such MUD found. Why not try again? (The name
s are case sensitive)");
      System.out.println();
      joinMUD();
    } else {

      // check if the current total number of players online is not exceeding th
e
      // maximum, otherwise terminate the program
      if (!serv.checkIfPlayerLimitNotExceeded()) {
        System.out.println("Sorry, the total number of available players has bee
n exceeded. Please try again later.");
        System.exit(0);
      } else {

        // check if the current number of playing in the chosen MUD is not excee
ding
        // the total, otherwise propmt the player to choose another one
        if (!serv.checkIfPlayerLimitNotExceededInMUD(mudName)) {
          System.out.println("Sorry, the total number of players playing on this
 MUD is exceeding the maximum. Choose another MUD or try again later.");
          joinMUD();
        } else {

          // drop all items that the player is carrying to avoid item duplicatio
n
          for (String item : inventory) {
            serv.dropItem(currentLocation, item);
          }
```

```
          inventory.clear();

          // move the player to another MUD
          System.out.println(serv.createUser(playerName, mudName));
          System.out.println();
        }
      }
    }
  }

  // allow the player to create a new MUD in real time
  private static void createNewMUD(String mudName) throws RemoteException {

    // check if the maximum number of MUDs available has not been exceeded
    if(serv.createNewMUD(mudName)) {
      System.out.println("Your MUD " + mudName + " has been created.");
    } else {
      Integer maxNumberOfMUDs = serv.getMaxNumberOfMuds();
      System.out.println("Sorry, but the maximum number of MUDs (" + maxNumberOf
MUDs + ") has been reached.");
      System.out.println("You can change this number by using the 'changemaxmuds
' command.");
    }
  }
}
```