```
/*******************************************************************
 * mud.MUDServerImpl
 *******************************************************************/

package mud;

import java.rmi.RemoteException;
import java.util.*;

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

public class MUDServerImpl implements MUDServerInterface {

  // all available MUDs
  private Map<String, MUD> MUDs = new HashMap<String, MUD>();

  // stores the current MUD that the player is on
  private MUD currentInstance;

  // number of players currently online throughout all MUDs
  // stores the name of the player and the name of the MUD that the player is on
  private Map<String, String> currentPlayers = new HashMap<String, String>();

  // maximum number of MUDs running at the same time
  private static int maxNumberOfMUDs = 5;

  // maximum number of player currently online
  private static int maxNumberOfPlayers = 10;

  // maximum number of player currently playing in a single MUD
  private static int maxNumberOfPlayersInMUD = 5;

  public MUDServerImpl() throws RemoteException {
  }

  // create two MUDs
  public void initialize() {
    MUDs.put("myMUD", new MUD("mymud.edg","mymud.msg","mymud.thg"));
    MUDs.put("myMUD2", new MUD("mymud.edg","mymud.msg","mymud.thg"));
  }

  // create user
  // returns the location information about the players starting location
  public String createUser(String playerName, String mudName) {
    System.out.println("The player " + playerName + " has joined the " + mudName
 + " MUD.");
    currentInstance = MUDs.get(mudName);
    currentInstance.addThing(currentInstance.startLocation(), playerName);
    currentInstance.players.put(playerName, currentInstance.startLocation());

    // if changing MUD, remove the player from the current players list
    if (Arrays.stream(currentPlayers.keySet().toArray(new String[currentPlayers.
keySet().size()])).anyMatch(playerName::equals)) {
      currentPlayers.remove(playerName);
    }
    // add the player to the current players list with the MUD that they are cur
rently on
    currentPlayers.put(playerName, mudName);

    return currentInstance.locationInfo(currentInstance.startLocation());
```

```
  }

  // checks if the number of player in a given MUD does not exceed the maximum
  // number of players allowed to play in a single MUD
  public boolean checkIfPlayerLimitNotExceededInMUD(String mudName) {
    currentInstance = MUDs.get(mudName);
    if (currentInstance.players.size() < maxNumberOfPlayersInMUD) {
      return true;
    } else {
      return false;
    }
  }

  // checks if the current number of players online is not exceeding the maximum
 number of players
  public boolean checkIfPlayerLimitNotExceeded() {
    if (currentPlayers.size() < maxNumberOfPlayers) {
      return true;
    } else {
      return false;
    }
  }

  // move the user to a give direction
  // returns a string telling that the player has moved a given direction
  public String moveUser(String currentLocation, String direction, String player
Name) {
    currentInstance.players.remove(playerName);
    currentInstance.players.put(playerName, direction);
    return currentInstance.moveThing(currentLocation, direction, playerName);
  }

  // returns the start location of the MUD
  public String getStartLocation() {
    return currentInstance.startLocation();
  }

  // returns the information about the player's current location
  public String getCurrentLocationInfo(String currentLocation) {
    return currentInstance.locationInfo(currentLocation);
  }

  // deletes an object from the location by putting it in the player's inventory
  public void pickUpItem(String currentLocation, String item) {
    currentInstance.delThing(currentLocation, item);
  }

  // add an object to the location by dropping it from the player's inventory
  public void dropItem(String currentLocation, String item) {
    currentInstance.addThing(currentLocation, item);
  }

  // returns a list of all players that are currently playing in the same MUD
  public String[] getCurrentPlayersInMUD() {
    return currentInstance.players.keySet().toArray(new String[currentInstance.p
layers.keySet().size()]);
  }

  // handles the player exiting the MUD by removing it from the players' list
  public void exit(String playerName) {
    currentInstance.players.remove(playerName);
    currentPlayers.remove(playerName);
```

```
    System.out.println("The player " + playerName + " has left the server.");
  }

  // returns a list of all currently available MUDs
  public String[] getAvailableMUDs() {
    return MUDs.keySet().toArray(new String[MUDs.keySet().size()]);
  }

  // returns a number of currently available MUDs
  public Integer getMUDCount() {
    return MUDs.size();
  }

  // checks if a MUD of a given name currently exists
  public boolean checkIfMUDExists(String mudName) {
    return Arrays.stream(getAvailableMUDs()).anyMatch(mudName::equals);
  }

  // creates a new MUD in real time
  public boolean createNewMUD(String mudName) {
    // check if the current number of MUDs are not exceeding the maximum number
    if (maxNumberOfMUDs > MUDs.size()) {
      MUDs.put(mudName, new MUD("mymud.edg","mymud.msg","mymud.thg"));
      return true;
    } else {
      return false;
    }
  }

  // returns a number of current maximum number of MUDs
  public Integer getMaxNumberOfMuds() {
    return maxNumberOfMUDs;
  }

  // sets the new maximum number of MUDs
  public void setNewMaxNumberOfMUDs(Integer newMaxMUDS) {
    maxNumberOfMUDs = newMaxMUDS;
  }
}
```