```
/*
* Copyright (c) Joan-Manuel Marques 2013. All rights reserved.
* DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
*
* This file is part of the practical assignment of Distributed Systems
course.
*
* This code is free software: you can redistribute it and/or modify
* it under the terms of the GNU General Public License as published by
* the Free Software Foundation, either version 3 of the License, or
* (at your option) any later version.
*
* This code is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
* along with this code.  If not, see <http://www.gnu.org/licenses/>.
*/

package udp.client;


import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;
import java.net.UnknownHostException;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import edu.uoc.dpcs.lsim.logger.LoggerManager.Level;
import lsim.library.api.LSimLogger;


/**
 * @author Joan-Manuel Marques
 *
 */

public class RemoteMapUDPclient {

      public RemoteMapUDPclient() {
      }

      public Map<String, String> getMap (List<Key> keys) {
            Map<String, String> map = new HashMap<String, String>();
            int i = 1;
            for (Key key : keys) {
                  LSimLogger.log(
                              Level.TRACE,
                              "["+i+"] Query for key "+key.getKey()+"
at "+ key.getServerAddress() +":"+key.getServerPort()
                              );

                  String value = get(key.getKey(),
key.getServerAddress(), key.getServerPort());
```

```
                  LSimLogger.log(Level.TRACE, "["+i+"]
RemoteMap("+key.getKey()+"): "+ value);
                  i++;
                  map.put(key.getKey(), value);
            }

            return map;
      }

      private String get(String key, String server_address, int
server_port){
            LSimLogger.log(Level.INFO, "inici RemoteMapUDPclient.get
");
            LSimLogger.log(Level.TRACE, "key: " + key);
            LSimLogger.log(Level.TRACE, "server_address: " +
server_address);
            LSimLogger.log(Level.TRACE, "server_port: " + server_port);

            String resposta = null;

            /* TODO: implementació de la part client UDP / implement
UDP client's side / implementación de la parte cliente UDP */

            String posicion = new String(key);
            //buffer donde se almacenara los mensajes
            byte[] buffer   = new byte[256];

            try {

                  //Creo el socket de UDP
                  DatagramSocket socketUDP = new DatagramSocket ();

                  //Obtengo la localizacion del servidor
                  InetAddress direccionServidor =
InetAddress.getByName(server_address);

                  //Convierto la key a bytes
                  //buffer = posicion.getBytes();

                  // Construimos un datagrama para enviar el mensaje al
servidor
                  DatagramPacket pregunta = new DatagramPacket
(posicion.getBytes(), posicion.getBytes().length, direccionServidor,
server_port);

                  // Enviamos el datagrama
                  socketUDP.send(pregunta);

                  // Construimos el DatagramPacket que contendrá la
respuesta
                  DatagramPacket respuesta = new DatagramPacket
(buffer, buffer.length);
                  socketUDP.receive(respuesta);

                  resposta  = new String(respuesta.getData()).trim();

                  // Cerramos el socket
                  socketUDP.close();

            } catch (SocketException e) {
                  LSimLogger.log(Level.ERROR, "SocketException");
```

```
                        e.printStackTrace();
                } catch (UnknownHostException e) {
                    LSimLogger.log(Level.ERROR, "UnknownHostException");
                    e.printStackTrace();
                } catch (IOException e) {
                    LSimLogger.log(Level.ERROR, "IOException");
                    e.printStackTrace();
                }

                    return resposta;
              }
        }
```