package dao;  
  
import com.mongodb.MongoClient;  
import com.mongodb.MongoException;  
import com.mongodb.client.FindIterable;  
import com.mongodb.client.MongoCollection;  
import com.mongodb.client.MongoCursor;  
import com.mongodb.client.MongoDatabase;  
import org.bson.Document;  
  
import java.net.UnknownHostException;  
import java.util.ArrayList;  
import java.util.List;  
  
public class query2 {  
 private static MongoCollection<Document> collection;  
  
 public static void main(String args[]) {  
  
 try {  
 initConnection();  
 String targetID = "20";  
 query(targetID);  
  
 } catch (Exception e) {  
 System.*err*.println(e.getClass().getName() + ": " + e.getMessage());  
 }  
 }  
  
 private static void initConnection() throws UnknownHostException, MongoException {  
 *// 连接到 mongodb 服务* MongoClient mongoClient = new MongoClient("localhost", 27017);  
 MongoDatabase mongoDatabase = mongoClient.getDatabase("DA");  
 System.*out*.println("Connect to database successfully");  
  
*// 创建  
// mongoDatabase.createCollection("assignment2\_tags");  
// System.out.println("集合创建成功");  
  
 //提取* collection = mongoDatabase.getCollection("assignment2\_tags");  
 System.*out*.println("集合 tags 选择成功");  
  
 }  
  
 private static void loadData() throws Exception {  
 */\*\*  
 \* 1. 获取迭代器FindIterable<Document>  
 \* 2. 获取游标MongoCursor<Document>  
 \* 3. 通过游标遍历检索出的文档集合  
 \* \*/* FindIterable<Document> findIterable = collection.find();  
 MongoCursor<Document> mongoCursor = findIterable.iterator();  
 while (mongoCursor.hasNext()) {  
 System.*out*.println(mongoCursor.next().toJson());  
  
 }  
 }  
  
 private static void query(String targetID) {  
 ArrayList<List<String>> movieList = new ArrayList<List<String>>(1000);  
 ArrayList<String> user = new ArrayList<String>();  
 ArrayList<List<String>> userList = new ArrayList<List<String>>(1000);  
 ArrayList<String> movie = new ArrayList<String>();  
  
 FindIterable<Document> findIterable = collection.find();  
 MongoCursor<Document> mongoCursor = findIterable.iterator();  
 while (mongoCursor.hasNext()) {  
 movieList.add(new ArrayList<String>(){});  
 userList.add(new ArrayList<String>(){});  
 Object E = mongoCursor.next();  
 String User = (((Document) E).get("UserID")).toString();  
 String Movie = (((Document) E).get("MovieID")).toString();  
 if (!user.contains(User)) {  
 user.add(User);  
 }  
 movieList.get(user.indexOf(User)).add(Movie);  
 if (!movie.contains(Movie)) {  
 movie.add(Movie);  
 }  
 userList.get(movie.indexOf(Movie)).add(User);  
 }  
  
*// for(String s : movieList.get(user.indexOf(targetID))){  
// System.out.println(userList.get(movie.indexOf(s)).toString());  
// }* ArrayList<String> resUser = new ArrayList<String>();  
 for(String m : movieList.get(user.indexOf(targetID))){  
 for(String u : userList.get(movie.indexOf(m))) {  
 if(!resUser.contains(u)){  
 resUser.add(u);  
 }  
 }  
 }  
 System.*out*.println(resUser.toString());  
 }  
}

[20, 49, 850, 1056, 1282, 3058, 10555, 17288, 17312, 29421, 32828, 32978, 33384, 35051, 37698, 39065, 59285, 60713, 61869, 65713, 66650, 66740, 67033, 67070, 67119, 67330, 69374, 69663, 70806, 71291, 3307, 8041, 9316, 11613, 14706, 17653, 19885, 23172, 30627, 32861, 33235, 43699, 47780, 48717, 49265, 51118, 52723, 56472, 57249, 57973, 62012, 65436, 67252, 67401, 68363, 68494, 68628, 70200, 71315, 71424, 71497, 71556, 146, 181, 16610, 22095, 25520, 27744, 34745, 43113, 43213, 48600, 54316, 57962, 64821, 66129, 66149, 67921, 68304, 69820, 70047, 70974, 1751, 2456, 2471, 2606, 2854, 3844, 4341, 6393, 10154, 11368, 11912, 14160, 14918, 15264, 16419, 17044, 17647, 18015, 19854, 21188, 21374, 22302, 23032, 23388, 23423, 23619, 24132, 25244, 25907, 26704, 27808, 28892, 28914, 29538, 29850, 31175, 33305, 34704, 35141, 40895, 41838, 42500, 45290, 46804, 47448, 48011, 48743, 50607, 53859, 56345, 57786, 59444, 63507, 63604, 67422, 68046, 68172, 68178, 68441, 68618, 69195, 69576, 70120, 70253, 70775, 71331, 477, 1017, 1225, 3489, 5031, 13086, 14653, 25450, 35533, 35721, 40800, 41130, 53192, 59092, 60863, 61274, 64363, 68637, 70299]