*Київський національний університет імені Т. Шевченка*

*Факультет комп’ютерних наук та кібернетики*

Паралельні обчислення

Лабораторна робота №3

Звіт

Виконав: студент групи ІПС-31

Кравчук Павло

*Київ-2020*

***Умова завдання:***

На основі програми, розробленої в лабораторній роботі №2, створити серверну програму, яка забезпечуватиме виконання віддалених запитів управління об’єктами. Свідоцтва про об’єкти повинні зберігатися в базі даних. Розробити клієнтську програму, зо відправлятиме запити на ввід, редагування і отримання інформації про об’єкти. Взаємодії між клієнтом та сервером має працювати за протоколом TCP/IP.

***Варіант 8***

***Предметна область: Футбол***

***Об'єкти: Команди, Гравці***

**Загальні:**

Файл Player.java:

public class Player {  
 public int code;  
 public int teamCode;  
 public String name;  
 public boolean isCaptain;  
 public int salary;  
  
 public String toString() {  
 return name;  
 }  
}

Файл Team.java:

public class Team { //don't use constructor since we're filling elements one by one  
 public int code;  
 public String name;  
  
 public String toString() {  
 return name;  
 }  
}

Аналогічно лабораторній роботі 2.

**Сервер:**

Файл Football.java:

…

Аналогічно лабораторні роботі 2.

Файл Server.java:

import java.io.DataInputStream;  
import java.io.DataOutputStream;  
import java.io.IOException;  
import java.net.ServerSocket;  
import java.net.Socket;  
import java.util.List;  
  
public class Server {  
 private ServerSocket server = null;  
 private Socket sock = null;  
 private DataOutputStream out = null;  
 private DataInputStream in = null;  
 private Football football;  
  
 Server(Football football) {  
 this.football = football;  
 }  
  
 public void start(int port) throws IOException {  
 server = new ServerSocket(port);  
 System.*out*.println("Started...");  
  
 while (true) {  
 sock = server.accept();  
  
 in = new DataInputStream(sock.getInputStream());  
 out = new DataOutputStream(sock.getOutputStream());  
  
 while (processQuery()) ;  
 }  
 }  
  
 private boolean processQuery() {  
 try {  
 int entity = in.readInt(); // 0 - teams, 1 - players  
 int oper = in.readInt();  
  
 int code;  
 String name;  
 Integer isCaptain;  
 Integer salary;  
 Integer teamCode;  
 Integer result;  
  
 switch (entity) {  
 case 0:  
 switch (oper) {  
 case 0: //add  
 code = in.readInt();  
 name = in.readUTF();  
 result = football.addTeam(code, name)? 1 : 0;  
 out.writeInt(result);  
 break;  
 case 1: //update  
 code = in.readInt();  
  
 int status;  
 status = in.readInt();  
 name = (status == 1? in.readUTF() : null); //if status = 1 read, else set null  
  
 result = football.updateTeam(code, name)? 1 : 0;  
 out.writeInt(result);  
 break;  
 case 2: //get  
 code = in.readInt();  
 Team team = football.getTeam(code);  
 if (team != null) {  
 out.writeInt(1);  
 out.writeUTF(team.name);  
 } else {  
 out.writeInt(0);  
 }  
 break;  
 case 3: //delete  
 code = in.readInt();  
 result = football.deleteTeam(code)? 1 : 0;  
 out.writeInt(result);  
 break;  
 case 4: //get all  
 List<Team> teams = football.showTeams();  
 if (teams != null) {  
 out.writeInt(1);  
 out.writeInt(teams.size());  
 for (Team entry : teams) {  
 out.writeInt(entry.code);  
 out.writeUTF(entry.name);  
 }  
 } else {  
 out.writeInt(0);  
 }  
 }  
 break;  
 case 1:  
 switch (oper) {  
 case 0: //add  
 code = in.readInt();  
 name = in.readUTF();  
 isCaptain = in.readInt();  
 salary = in.readInt();  
 teamCode = in.readInt();  
 result = football.addPlayer(code, name, isCaptain == 1, salary, teamCode)? 1 : 0;  
 out.writeInt(result);  
 break;  
 case 1: //update  
 code = in.readInt();  
  
 int status;  
 status = in.readInt();  
 name = (status == 1? in.readUTF() : null);  
  
 status = in.readInt();  
 isCaptain = (status == 1? in.readInt() : null);  
  
 status = in.readInt();  
 salary = (status == 1? in.readInt() : null);  
  
 status = in.readInt();  
 teamCode = (status == 1? in.readInt() : null);  
  
 Boolean boolIsCap;  
 if (isCaptain == null) boolIsCap = null;  
 else boolIsCap = isCaptain == 1;  
  
 result = football.updatePlayer(code, name, boolIsCap, salary, teamCode)? 1 : 0;  
 out.writeInt(result);  
 break;  
 case 2: //get  
 code = in.readInt();  
 Player player = football.getPlayer(code);  
 if (player != null) {  
 out.writeInt(1);  
 out.writeUTF(player.name);  
 out.writeInt(player.isCaptain? 1 : 0);  
 out.writeInt(player.salary);  
 out.writeInt(player.teamCode);  
 } else {  
 out.writeInt(0);  
 }  
 break;  
 case 3: //delete  
 code = in.readInt();  
 result = football.deletePlayer(code)? 1 : 0;  
 out.writeInt(result);  
 break;  
 case 4: //get all  
 List<Player> players = football.showPlayers();  
 if (players != null) {  
 out.writeInt(1);  
 out.writeInt(players.size());  
 for (Player entry : players) {  
 out.writeInt(entry.code);  
 out.writeUTF(entry.name);  
 out.writeInt(entry.isCaptain? 1 : 0);  
 out.writeInt(entry.salary);  
 out.writeInt(entry.teamCode);  
 }  
 } else {  
 out.writeInt(0);  
 }  
 }  
 break;  
 }  
  
 return true;  
 } catch (IOException e) {  
 return false;  
 }  
 }  
}

Клас що відповідає за ввімкнення сервера та обробки запитів з клієнта.

**Клієнт:**

Файл FootballClient.java:

import java.io.DataInputStream;  
import java.io.DataOutputStream;  
import java.io.IOException;  
import java.net.Socket;  
  
public class FootballClient {  
 private Socket sock = null;  
 private DataOutputStream out = null;  
 private DataInputStream in = null;  
  
  
 public FootballClient(String ip, int port) throws IOException {  
 sock = new Socket(ip, port);  
  
 in = new DataInputStream(sock.getInputStream());  
 out = new DataOutputStream(sock.getOutputStream());  
 }  
  
 public void addTeam(int code, String name) {  
 try {  
 int entity = 0;  
 int oper = 0;  
  
 out.writeInt(entity);  
 out.writeInt(oper);  
  
 out.writeInt(code);  
 out.writeUTF(name);  
  
 int result = in.readInt();  
  
 System.*out*.println(result == 1? "Success" : "Failed");  
 } catch (IOException ignore) {}  
 }  
  
 public void updateTeam(int code, String name) {  
 try {  
 int entity = 0;  
 int oper = 1;  
  
 out.writeInt(entity);  
 out.writeInt(oper);  
  
 out.writeInt(code);  
  
 if (name != null) {  
 out.writeInt(1); //if null send 0, else 1  
 out.writeUTF(name);  
 } else {  
 out.writeInt(0);  
 }  
  
 int result = in.readInt();  
  
 System.*out*.println(result == 1? "Success" : "Failed");  
 } catch (IOException ignore) {}  
 }  
  
 public Team getTeam(int code) {  
 Team team = null;  
  
 try {  
 int entity = 0;  
 int oper = 2;  
  
 out.writeInt(entity);  
 out.writeInt(oper);  
  
 out.writeInt(code);  
  
 int result = in.readInt();  
  
 System.*out*.println(result == 1? "Success" : "Failed");  
  
 if (result == 1) {  
 team = new Team();  
 team.code = code;  
 team.name = in.readUTF();  
 }  
 } catch (IOException ignore) {}  
  
 return team;  
 }  
  
 public void deleteTeam(int code) {  
 try {  
 int entity = 0;  
 int oper = 3;  
  
 out.writeInt(entity);  
 out.writeInt(oper);  
  
 out.writeInt(code);  
  
 int result = in.readInt();  
  
 System.*out*.println(result == 1? "Success" : "Failed");  
 } catch (IOException ignore) {}  
  
 }  
  
 public void showTeams() {  
 try {  
 int entity = 0;  
 int oper = 4;  
  
 out.writeInt(entity);  
 out.writeInt(oper);  
  
 int result = in.readInt();  
  
 System.*out*.println(result == 1? "Success" : "Failed");  
  
 if (result == 1) {  
 int size = in.readInt();  
 System.*out*.println("Teams:");  
  
 for (int i = 0; i < size; i++) {  
 System.*out*.println("ID: " + in.readInt() + " name: " + in.readUTF());  
 }  
 }  
 } catch (IOException ignore) {}  
 }  
  
 public void addPlayer(int code, String name, boolean isCaptain, int salary, int teamCode) {  
 try {  
 int entity = 1;  
 int oper = 0;  
  
 out.writeInt(entity);  
 out.writeInt(oper);  
  
 out.writeInt(code);  
 out.writeUTF(name);  
 out.writeInt(isCaptain? 1 : 0);  
 out.writeInt(salary);  
 out.writeInt(teamCode);  
  
 int result = in.readInt();  
  
 System.*out*.println(result == 1? "Success" : "Failed");  
 } catch (IOException ignore) {}  
 }  
  
 public void updatePlayer(int code, String name, Boolean isCaptain, Integer salary, Integer teamCode) {  
 try {  
 int entity = 1;  
 int oper = 1;  
  
 out.writeInt(entity);  
 out.writeInt(oper);  
  
 out.writeInt(code);  
  
 if (name != null) {  
 out.writeInt(1);  
 out.writeUTF(name);  
 } else {  
 out.writeInt(0);  
 }  
  
 if (isCaptain != null) {  
 out.writeInt(1); //if null send 0, else 1  
 out.writeInt(isCaptain? 1 : 0);  
 } else {  
 out.writeInt(0);  
 }  
  
 if (salary != null) {  
 out.writeInt(1);  
 out.writeInt(salary);  
 } else {  
 out.writeInt(0);  
 }  
  
 if (teamCode != null) {  
 out.writeInt(1);  
 out.writeInt(teamCode);  
 } else {  
 out.writeInt(0);  
 }  
  
 int result = in.readInt();  
  
 System.*out*.println(result == 1? "Success" : "Failed");  
 } catch (IOException ignore) {}  
 }  
  
 public Player getPlayer(int code) {  
 Player player = null;  
  
 try {  
 int entity = 1;  
 int oper = 2;  
  
 out.writeInt(entity);  
 out.writeInt(oper);  
  
 out.writeInt(code);  
  
 int result = in.readInt();  
  
 System.*out*.println(result == 1? "Success" : "Failed");  
  
 if (result == 1) {  
 player = new Player();  
  
 player.code = code;  
 player.name = in.readUTF();  
 player.isCaptain = in.readInt() == 1;  
 player.salary = in.readInt();  
 player.teamCode = in.readInt();  
 }  
 } catch (IOException ignore) {}  
  
 return player;  
 }  
  
 public void deletePlayer(int code) {  
 try {  
 int entity = 1;  
 int oper = 3;  
  
 out.writeInt(entity);  
 out.writeInt(oper);  
  
 out.writeInt(code);  
  
 int result = in.readInt();  
  
 System.*out*.println(result == 1? "Success" : "Failed");  
 } catch (IOException ignore) {}  
 }  
  
  
  
 public void showPlayers() {  
 try {  
 int entity = 1;  
 int oper = 4;  
  
 out.writeInt(entity);  
 out.writeInt(oper);  
  
 int result = in.readInt();  
  
 System.*out*.println(result == 1 ? "Success" : "Failed");  
  
 if (result == 1) {  
 int size = in.readInt();  
 System.*out*.println("Players:");  
  
 for (int i = 0; i < size; i++) {  
 System.*out*.println("ID: " + in.readInt() + " name: " + in.readUTF() +  
 " captain: " + (in.readInt() == 1 ? "1" : "0") + " salary: " +  
 in.readInt() + " teamCode: " + in.readInt());  
 }  
 }  
 } catch (IOException ignore) { }  
 }  
}

Клас, методи якого збігаються по назві та функціоналу з методами класу Football сервера, проте замість запитів до СУБД відбувається запити до сервера і надсилання додаткових необхідних даних, після чого отримання відповідей про вдалість операції та залежно від операції додаткових даних.

**Запити та відповіді:**

Структура запиту клієнта:

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип даних** | **Опис** |
| Номер таблиці | Int | 0 – працює на таблицею команд;  1 – над таблицею гравців |
| Номер команди | Int | 0 – Додати  1 – змінити існуючий  2 – отримати за ID  3 – видалити за ID  4 – отримати всі |
| Додаткові | Int або string | Додаткові дані, необхідні, наприклад, при створені нового запису |

Структура відповіді сервера:

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип даних** | **Опис** |
| Статус операції | Int | 0 – операція не вдалась;  1 – операція успішна |
| Додаткові | Int або string | Додаткові дані, якщо клієнт запитав дані з таблиці |