Київський національний університет імені Т. Шевченка
Факультет комп'ютерних наук та кібернетики

Паралельні обчислення
Лабораторна робота №2
Звіт

Виконав: студент групи ІПС-31
Кравчук Павло

Київ-2020

Умова завдання:

Розробити програму, що забезпечує ввід і редагування інформації про об’єкти згідно з заданою предметною областю. Інформація про об’єкти має зберігатись в окремій базі даних. Доступ до даних отримується з використанням можливостей JDBC або ODBC.

Варіант 8

Предметна область: Футбол

Об'єкти: Команди, Гравці

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;
    }
}
```

Клас гравця, містить поля для унікального ідентифікатору, імені, статусу капітана команди, зарплатні та коду команди до якої він належить. Методи гравця: звичайний метод приведення до строки, який повертає лише ім’я.

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;
    }
}
```

Клас команди, містить поля для унікального ідентифікатору та назви команді. Методи команди: звичайний метод приведення до строки, який повертає лише назву.

Файл Football.java:

```java
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class Football {
    private Connection con;
    private Statement stmt;

    Football(Connection con) throws SQLException {
        this.con = con;
        this.stmt = con.createStatement();
    }

    public void stop() throws SQLException {
        con.close();
    }

    public void addTeam(int code, String name) {
        String sql = "INSERT INTO TEAMS (ID_T, NAME)" +
                "VALUES ("+code+", '"+name+"')";
        try
        {
            stmt.executeUpdate(sql);
            System.out.println("Team "+name+
                    " added");
        } catch (SQLException e)
        {
            System.out.println("Error: team  "+name+
                    " was not added");
            System.out.println(" >> "+e.getMessage());
        }

    }

    public void updateTeam(int code, String name) {
        String sql = "UPDATE TEAMS " +
                "SET ";
        boolean previous = false;
        if (name != null) {
            sql += "NAME = '"+name+"'";
            previous = true;
        }

        if (!previous) {
            System.out.println("Nothing to update");
            return;
        }

        sql += " WHERE ID_T = " + code;

        try
        {
            stmt.executeUpdate(sql);
            System.out.println("Team with id "+code+
                    " updated");
        } catch (SQLException e)
        {
            System.out.println("Error: team  "+name+
                    " was not updated");
            System.out.println(" >> "+e.getMessage());
        }
    }
```

```
    public Team getTeam(int code) {
        String sql = "SELECT *" +
                "FROM TEAMS T1" +
                "WHERE T1.ID_T = " + code;

        Team team = new Team();
        try
        {
            ResultSet rs = stmt.executeQuery(sql);
            if (rs.next())
            {
                int id = rs.getInt("ID_T");
                team.code = id;
                String name = rs.getString("NAME");
                team.name = name;
                System.out.println(" >> "+ id + " - " + name);
            } else {
                System.out.println("Team with id " + code + " not found");
            }

        } catch (SQLException e)
        {
            System.out.println("Error while looking for a team");
            System.out.println(" >> "+e.getMessage());
        }

        return team;
    }

    public void deleteTeam(int code) {
        String sql1 = "DELETE FROM PLAYERS WHERE ID_T = "+code;

        String sql = "DELETE FROM TEAMS WHERE ID_T = "+code;
        try
        {
            int c1 = stmt.executeUpdate(sql1);  //delete players of the team
            int c = stmt.executeUpdate(sql);
            if (c>0)
            {
                System.out.println("Team with id "
                        + code +" deleted successfully");
            }
            else
            {
                System.out.println("Team with id "
                        + code +" was not found");
            }
        } catch (SQLException e)
        {
            System.out.println(
                    "Error while deleting team with id "+code);
            System.out.println(" >> "+e.getMessage());
        }

    }

    public void showTeams() {
        String sql = "SELECT ID_T, NAME FROM TEAMS";
        try
        {
            ResultSet rs = stmt.executeQuery(sql);
            System.out.println("Teams:");
            while (rs.next())
            {
                int id = rs.getInt("ID_T");
                String name = rs.getString("NAME");
                System.out.println(" >> "+ id + " - " + name);
```

```
            }
            rs.close();
        } catch (SQLException e)
        {
            System.out.println(
                    "Error while getting teams");
            System.out.println(" >> "+e.getMessage());
        }
    }

    public void addPlayer(int code, String name, boolean isCaptain, int salary, int
teamCode) {
        String sql = "INSERT INTO PLAYERS (ID_P, ID_T, NAME, ISCAPTAIN, SALARY)" +
                "VALUES ("+code+", "+teamCode+", '"+name+"', "+(isCaptain?"1":"0")+",
"+salary+")";
        try
        {
            stmt.executeUpdate(sql);
            System.out.println("Player "+name+
                    " added");
        } catch (SQLException e)
        {
            System.out.println("Error: team  "+name+
                    " was not added");
            System.out.println(" >> "+e.getMessage());
        }
    }

    public void updatePlayer(int code, String name, Boolean isCaptain, Integer
salary, Integer teamCode) {
        String sql = "UPDATE PLAYERS " +
                "SET ";
        boolean previous = false;
        if (name != null) {
            sql += "NAME = '"+name+"'";
            previous = true;
        }
        if (isCaptain != null) {
            if (previous) sql += ", ";
            sql += "ISCAPTAIN = " + (isCaptain?"1":"0");
            previous = true;
        }
        if (salary != null) {
            if (previous) sql += ", ";
            sql += "SALARY = " + salary;
            previous = true;
        }
        if (teamCode != null) {
            if (previous) sql += ", ";
            sql += "ID_T = " + teamCode;
            previous = true;
        }

        if (!previous) {
            System.out.println("Nothing to update");
            return;
        }

        sql += " WHERE ID_P = " + code;

        try
        {
            stmt.executeUpdate(sql);
            System.out.println("Player with id "+code+
                    " updated");
        } catch (SQLException e)
        {
```

```
            System.out.println("Error: player  "+name+
                    " was not updated");
            System.out.println(" >> "+e.getMessage());
        }
    }

    public Player getPlayer(int code) {
        String sql = "SELECT *" +
                "FROM PLAYERS T1" +
                "WHERE T1.ID_P = " + code;

        Player player = new Player();
        try
        {
            ResultSet rs = stmt.executeQuery(sql);
            if (rs.next())
            {
                int id = rs.getInt("ID_P");
                player.code = id;

                int idT = rs.getInt("ID_T");
                player.teamCode = idT;

                String name = rs.getString("NAME");
                player.name = name;

                boolean isCaptain = (rs.getString("ISCAPTAIN").equals("1"));
                player.isCaptain = isCaptain;

                int salary = rs.getInt("SALARY");
                player.salary = salary;

                System.out.println(" >> "+ id + " - " + idT + " - " + name + " - " +
(isCaptain?"1":"0") +
                        " - " + salary);
            } else {
                System.out.println("Player with id " + code + " not found");
            }

        } catch (SQLException e)
        {
            System.out.println("Error while looking for a player");
            System.out.println(" >> "+e.getMessage());
        }

        return player;
    }

    public void deletePlayer(int code) {
        String sql = "DELETE FROM PLAYERS WHERE ID_P = "+code;
        try
        {
            int c = stmt.executeUpdate(sql);
            if (c>0)
            {
                System.out.println("Player with id "
                        + code +" deleted successfully");
            }
            else
            {
                System.out.println("Player with id "
                        + code +" was not found");
            }
        } catch (SQLException e)
        {
            System.out.println(
                    "Error while deleting player with id "+code);
```

```
            System.out.println(" >> "+e.getMessage());
        }
    }



    public void showPlayers() {
        String sql = "SELECT ID_P, ID_T, NAME, ISCAPTAIN, SALARY FROM PLAYERS";
        try
        {
            ResultSet rs = stmt.executeQuery(sql);
            System.out.println("Players:");
            while (rs.next())
            {
                int id = rs.getInt("ID_P");
                int idT = rs.getInt("ID_T");
                String name = rs.getString("NAME");
                boolean isCaptain = (rs.getString("ISCAPTAIN").equals("1"));
                int salary = rs.getInt("SALARY");

                System.out.println(" >> "+ id + " - " + idT + " - " + name + " - " +
(isCaptain?"1":"0") +
                        " - " + salary);
            }
            rs.close();
        } catch (SQLException e)
        {
            System.out.println(
                    "Error while getting players");
            System.out.println(" >> "+e.getMessage());
        }
    }
}
```

Клас звертається до СУБД, отримує та керує даними з неї. Наявні функції зберігання, видалення, зміни характеристик та виведення всіх гравців та команд.

ER: