УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«УНИВЕРСИТЕТ УПРАВЛЕНИЯ «ТИСБИ»

Факультет информационных технологий

Кафедра информационных технологий

**Курсовая работа**

По дисциплине «Базы данных»

Тема: «Проектирование и разработка базы данных информационной системы спортивного клуба»

Выполнил:

студент группы П-711

Мустафин К.Н

Проверил:

доц. Таренко Л.Б.

г. Казань, 2019 г.

**Содержание**

**[Введение](#_Toc26824689)** [3](#_Toc26824689)

[**1.** **Постановка задачи** 4](#_Toc26824690)

[**1.1 Теоретические сведения по проектированию реляционных баз данных.** 5](#_Toc26824691)

[**1.2 Анализ предметной области: выявление основных сущностей и их связей.** 7](#_Toc26824692)

[**1.3. Модель сущность-связь для базы данных** 8](#_Toc26824693)

[**1.4. Схема реляционной базы данных** 9](#_Toc26824694)

[**2. Разработка программного приложения** 11](#_Toc26824695)

[**2.1 Описание структуры и основных методов (процедур и функций) приложения; описание разработки интерфейса приложения** 11](#_Toc26824696)

[**2.2 Результаты тестирования с примерами визуального форм приложения** 13](#_Toc26824697)

[**Список литературы** 28](#_Toc26824698)

[**Приложение 1** 30](#_Toc26824699)

[**Приложение 2** 58](#_Toc26824700)

# **Введение**

В настоящее время любая серьезная организация использует различные базы данных для автоматизации процессов обработки информации, удобства в ее эксплуатации, надежности, для предотвращения избыточности повторяющихся данных, а также в целях экономии времени.

Целью данной курсовой работы является анализ спортивного клуба, а именно определение сущностей и их взаимосвязей, а также создание удобного оконного приложения.

На первом этапе курсовой работы выполняется анализ предметной области и проектирование базы данных спортивного клуба, включающие в себя 5 сущностей: родители, спортсмены, тренера, залы и дисциплины.

На втором этапе курсовой работы разрабатывается оконное приложение для удобного пользования базой данных, в которой можно добавить нового спортсмена, указав его родителя и секцию, которую он посещает, а также добавление нового тренера и секции. Также, помимо добавления, можно их также удалять.

# **Постановка задачи**

**Требования:**

* Система программирования – NetBeans.
* Система управления базами данных (СУБД) – Windows Access 2007 - 2013

Назначение программы:

1. Хранение информации об тренерах, учениках, родителях, залах и секциях

2. Возможность добавления и удаления необходимой информации.

3. Поиск запрашиваемых данных

**Задачи:**

Описать предметную область (описание должно быть кратким, но достаточным для принятия решений по проекту базы данных).

Определить состав и содержание информации, используемой в данной предметной области, в том числе:

* Составить перечень задач и запросов, указать входные и выходные данные
* Определить возможные будущие изменения информационных потребностей пользователей
* Установить уточненные требования к информационным потребностям пользователей

При этом очень важно определить перечень выполняемых системой функций.

Целью курсового проекта является разработка базы данных «Спортивного клуба» и создание оконного приложения для удобного доступа к информации, добавление и удаление всей необходимой информации.

# **1.1 Теоретические сведения по проектированию реляционных баз данных.**

**База данных** – это совокупность логически связанных данных. Базой данных часто упрощённо или ошибочно называют Системы Управления Базами Банных (СУБД). Нужно различать набор данных (собственно БД) и программное обеспечение, предназначенное для организации и ведения баз данных (СУБД). Одним из основных понятий в БД является атрибут – некоторая характеристика, или параметр, представляемого в БД объекта.

Банк данных – это информационная система, включающая в себя математические, технические, информационные, организационные, программные и языковые средства, обеспечивающие в совокупности централизованную поддержку хранимых данным. Банк данных включает в себя БД и СУБД.

На уровне физической модели электронная БД представляет собой файл или их набор в формате TXT, CSV, Excel, DBF, XML либо в специализированном формате конкретной СУБД. Также в СУБД в понятие физической модели включают специализированные виртуальные понятия, существующие в ёё рамках — таблица, табличное пространство, сегмент, куб, кластер и т.д.

В настоящее время наибольшее распространение получили реляционные базы данных. Картотеками пользовались до появления электронных баз данных. Сетевые и иерархические базы данных считаются устаревшими, объектно-ориентированные пока никак не стандартизированы и не получили широкого распространения. Некоторое возрождение получили иерархические базы данных в связи с появлением и распространением формата XML.

**СУБД** — специализированная программа (чаще комплекс программ), предназначенная для манипулирования базой данных. Для создания и управления информационной системой СУБД необходима в той же степени, как для разработки программы на алгоритмическом языке необходим транслятор.

**Основные функции СУБД:**

* Управление данными во внешней памяти (на дисках);
* Управление данными в оперативной памяти;
* Журнализация изменений и восстановление базы данных после сбоев;
* Поддержка языков БД (язык определения данных, язык манипулирования данными).

Обычно современная СУБД содержит следующие компоненты: ядро, которое отвечает за управление данными во внешней и оперативной памяти и журнализацию, процессор языка базы данных, обеспечивающий оптимизацию запросов на извлечение и изменение данных и создание, как правило, машинно-независимого исполняемого внутреннего кода, подсистему поддержки времени исполнения, которая интерпретирует программы манипуляции данными, создающие пользовательский интерфейс с СУБД, а также сервисные программы (внешние утилиты), обеспечивающие ряд дополнительных возможностей по обслуживанию информационной системы.

**Классификация СУБД**

По типу управляемой базы данных СУБД разделяются на:

* Сетевые
* Иерархические
* Реляционные
* Объектно-реляционные
* Объектно-ориентированные

По архитектуре организации хранения данных локальные СУБД (все части локальной СУБД размещаются на одном компьютере) распределенные СУБД (части СУБД могут размещаться на двух и более компьютерах)

# **1.2 Анализ предметной области: выявление основных сущностей и их связей.**

Целью данной работы является разработка и реализация электронной базы данных, обеспечивающей хранение, накопление и предоставление всей необходимой информации про тренеров, родителей, спортсменов, залах и секциях.

**Спортивный клуб** **–** клуб в котором дети, абсолютно бесплатно, занимаются спортивными дисциплинами. Главным пользователем БД будет администратор. Именно он будет добавлять информацию про тренеров, спортсменов, секции, родителей. Вся информация будет храниться в базе данных.

**В результате проведенного анализа предметной области были выявлены следующие сущности:**

1. Родители
2. Секции
3. Спортсмены
4. Тренера
5. Залы

**У каждой сущности выделим ее атрибуты:**

* Родители: id, фамилия, имя, номер телефона.
* Секции: id, название, id-тренера, максимальное количество спортсменов, день и время, id-зала.
* Спортсмены: id, фамилия, имя, день рождения, id-родителя, id-секции.
* Тренера: id, фамилия, имя, номер телефона, образование.
* Залы: id, название(номер) зала.

# **1.3. Модель сущность-связь для базы данных**

1:1

Спортсмен

1:М

Секция

М:1

1:1

Родитель

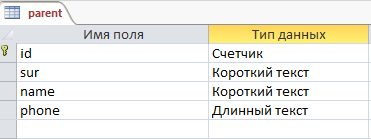
Зал

Тренер

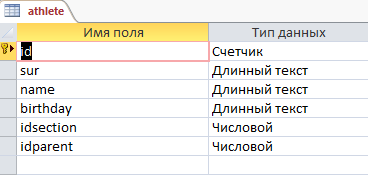
# **1.4. Схема реляционной базы данных**

На основе выполненной работы, разберем каждую таблицу сущность.

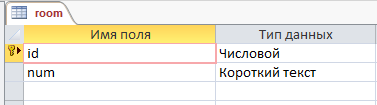
Начнем с сущности родитель: сюда входит id (числового типа, а также он является ключевым значением), фамилия (текстовый), имя (текстовый) и номер телефона(текстовый). Id – ставится автоматически. Фамилия, имя и номер телефона вводится в ручную.



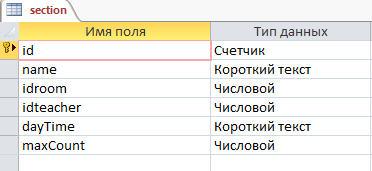
Сущность спортсмен: сюда входит id (числовой, он же ключевой), фамилия (текстовый), имя (текстовый), день рождения (текстовый), id-секции (числовой) и id-родителя (числовой). Id – ставится автоматически. Фамилия, имя и номер вводятся в ручную. Id-секции и id-родителя выбираются из соответствующих таблиц.



Сущность зал: содержит id (числовой, он же ключевой) и номер (текстовый). Id – заполняется автоматически, а номер (или, другими словами, название зала) вводится в ручную.



Сущность секция: содержит id (числовой, он же ключевой), название (текстовый), id-зала (числовой), id-тренера (числовой), день недели и время(текстовый), максимальное количество спортсменов (числовой). Id – заполняется автоматически, название, день недели и время, а также максимальное количество спортсменов нужно вводить в ручную, а id-зала и id-тренера нужно выбирать из соответствующих таблиц.



На конец сущность тренер: id (числовой, он же ключевой), фамилия (текстовый), имя (текстовый), номер телефона (текстовый), образование (текстовый). Id – вводится автоматически, остальные поля нужно заполнять в ручную.

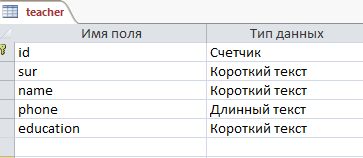
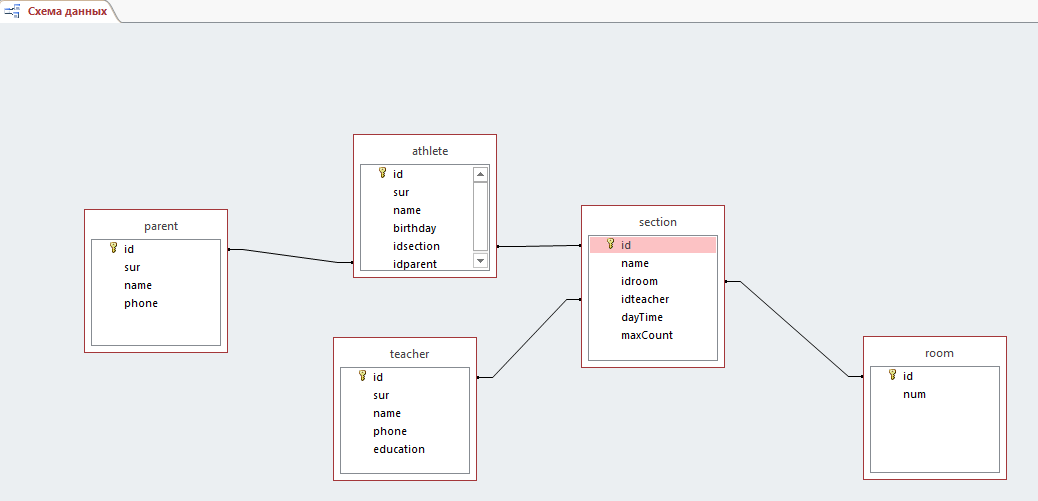


Схема данных:



# **2. Разработка программного приложения**

## **2.1 Описание структуры и основных методов (процедур и функций) приложения; описание разработки интерфейса приложения**

Для установки связи с базой данных MS Access со средой разработки Apache NetBeans 10, используются дополнительные библиотеки: commons-lang-2.6.jar, commons-logging-1.1.3.jar, hsqldb-2.3.1.jar, jackcess-2.1.11.jar и ucanaccess-4.0.4.jar. Эти библиотеки нужны для того, чтобы наше приложения умело получать доступ к базе данных, знала простые команды для работы с базой данных и самое главное – умела отправлять SQL – запросы в базу данных.

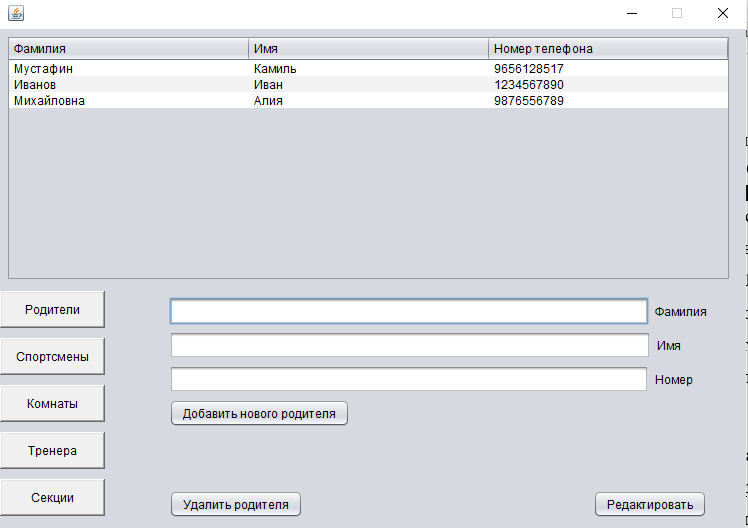
SQL- декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных. В нашем случае данной базой данных является MS Access 2007-2013.

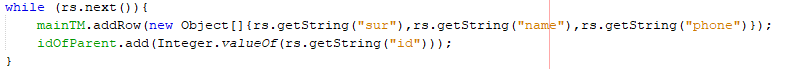
Использовалось 4 основных SQL запроса: это запросы записей по каким-то условиям (или без условий), удаление каких-то записей из таблицы по каким-то условиям (или без условий), обновление записей в базе данных по каким-то условия (или без условий), ну и, конечно же, добавление новых записей в таблицу.

Операторы:

* SELECT выбирает данные, удовлетворяющие заданным условиям,
* INSERT добавляет новые данные,
* UPDATE изменяет существующие данные,
* DELETE удаляет данные;

Для обращения к базе данных использовались только компоненты-кнопки, т.е. нажимая, на определенную кнопку происходил определенный запрос SQL в базу данных. Пример:

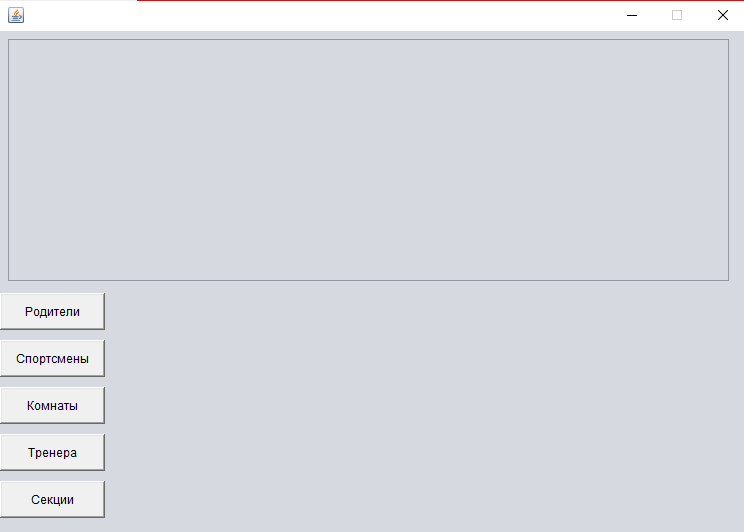


При нажатии на кнопку «Родители» произошел sql запрос: ResultSet rs = st.executeQuery("SELECT \* FROM parent"); который означает, что мы запросили всю информацию из таблицы “parent”, затем в таблицу мы добавляли по новой строчке в которую входили: Фамилия, Имя и номер телефона.

В проекте есть два главных файла: test.java и editForm.java, первый файл нужен для основной работы, второй файл нужен для того, чтобы редактировать какую-то информацию.

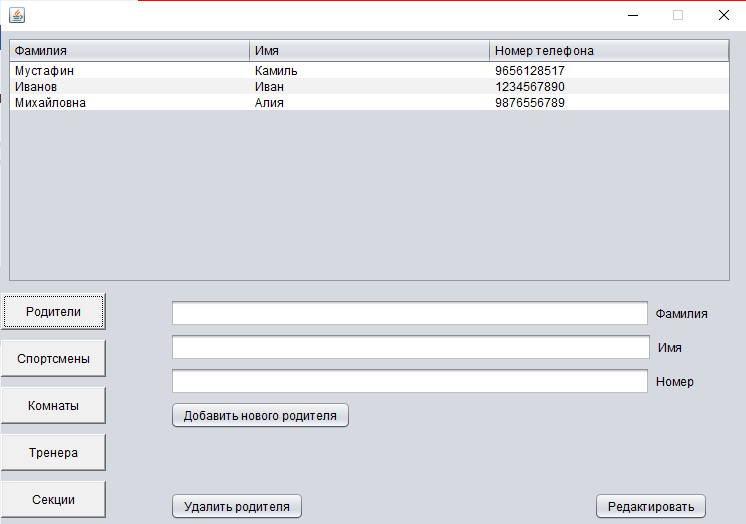
# **2.2 Результаты тестирования с примерами визуального форм приложения**

При первом запуске программы появляется вот такое окно:

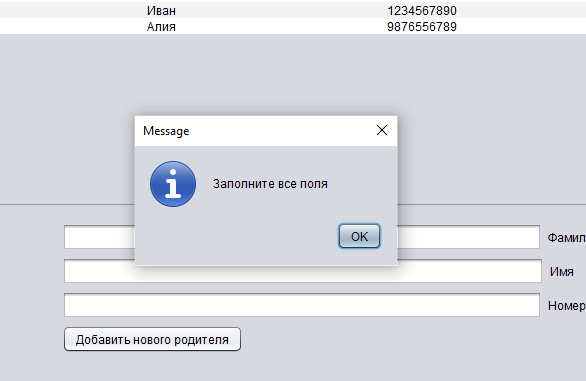


Здесь есть кнопки: родители, спортсмены, комнаты, тренера и секции, а также главная табличка.

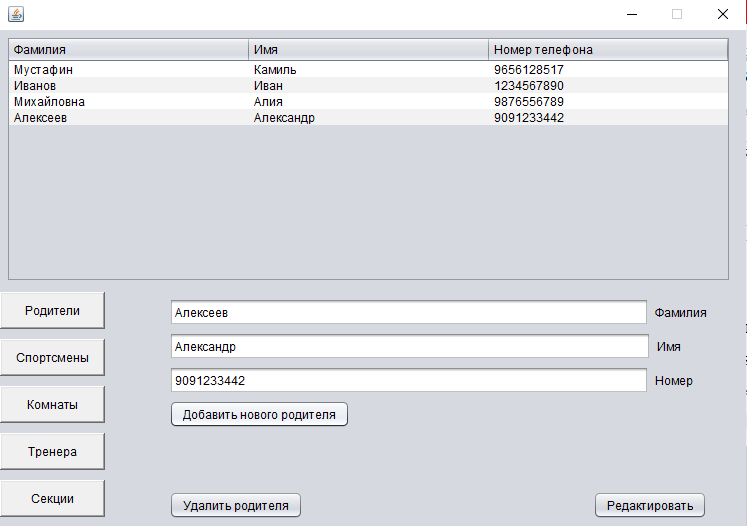
Попробуем нажать на кнопку родители:



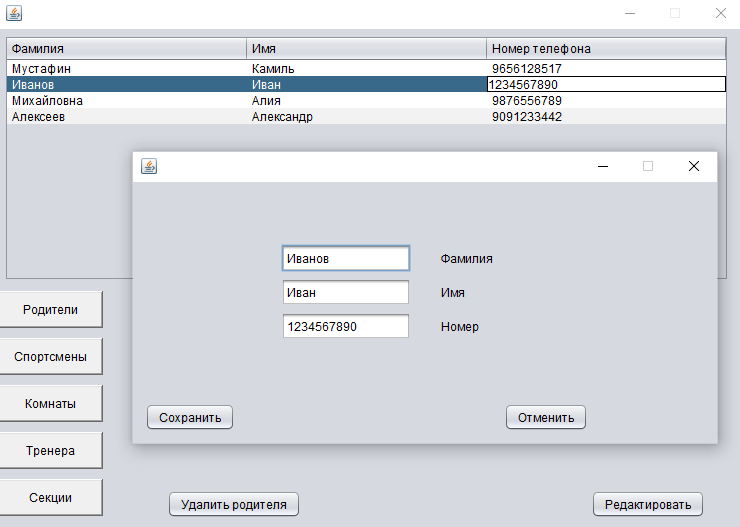
Заполнятся поля с уже существующими родителями в базе данных и откроется скрытая панель где есть несколько текстовых полей и кнопки. Если попробуем добавить нового родителя, то выйдет ошибка:



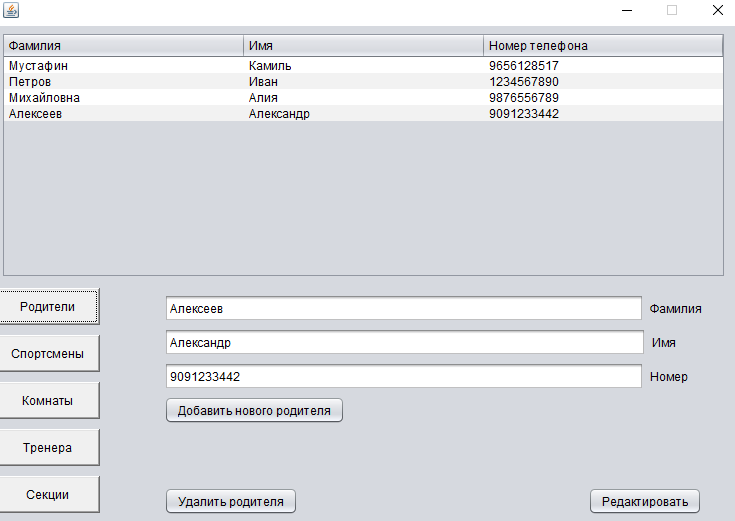
Для добавления нового родителя нужно прежде всего заполнить все поля:



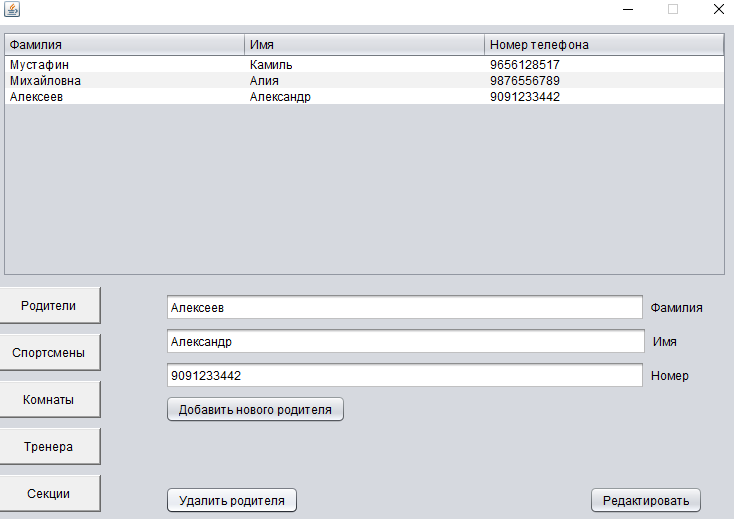
Если мы хотим редактировать какого-то родителя, нужно выделить родителя в таблице и нажать на кнопку редактировать:



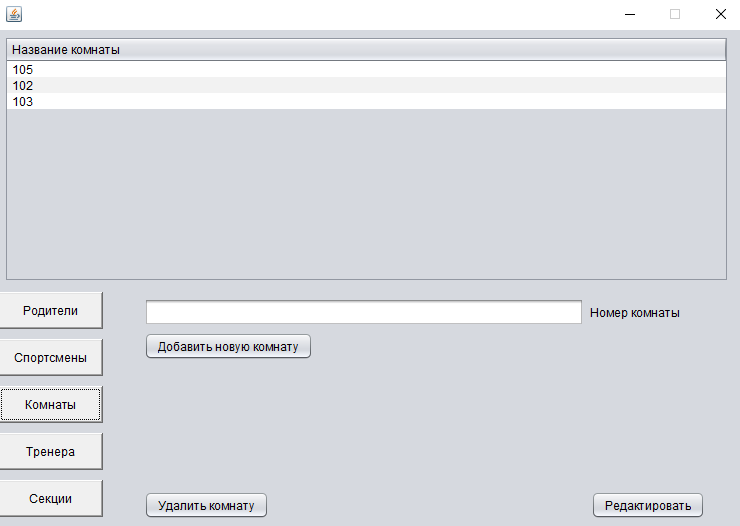
При нажатии на кнопку редактировать появляется новое окно, где можно редактировать все данные связанные с родителем и при нажатии на кнопку сохранить они сохранятся в базе данных, давайте попробуем изменить фамилию на Петров, нажмем кнопку сохранить и обновим, нажав на кнопку «Родители»:



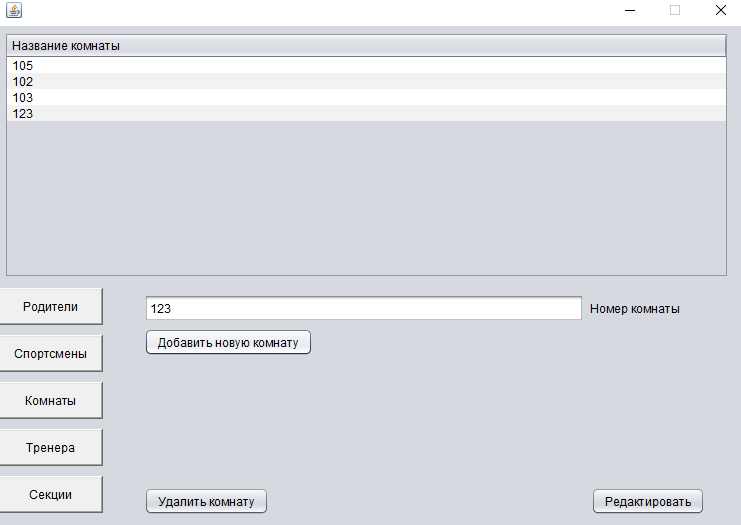
Попробуем удалить Петрова Ивана: для этого нужно выделить его в таблице и нажать удалить родителя:



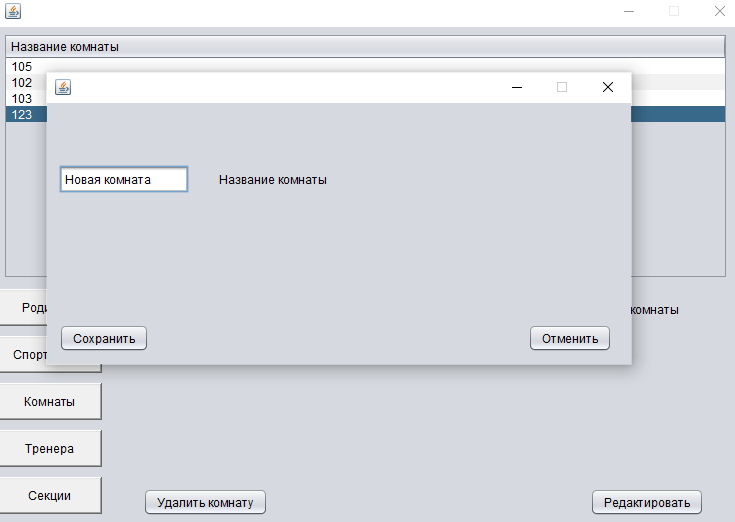
При нажатии на кнопку комнаты откроется:



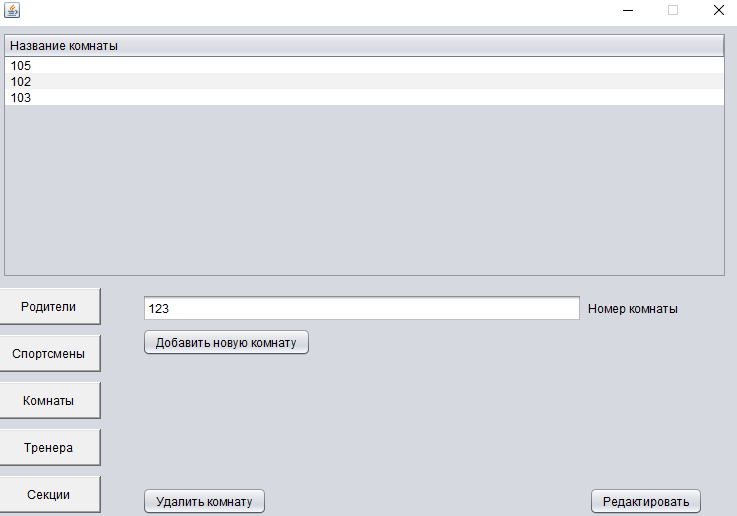
Попробуем добавить новую комнату:



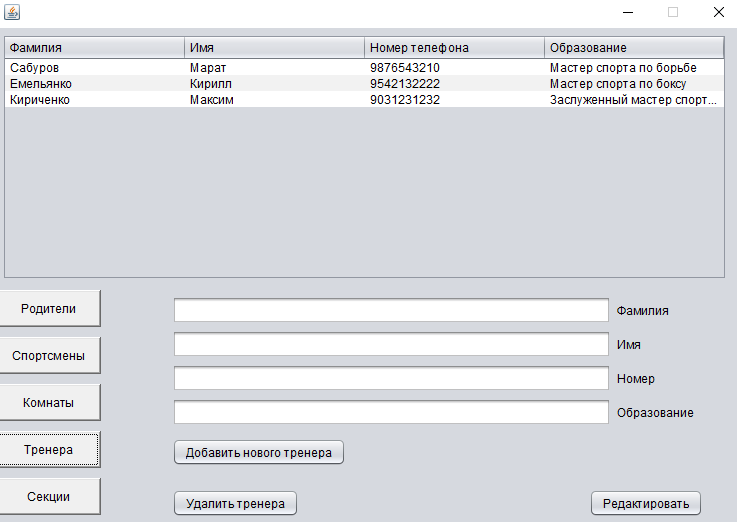
Попробуем отредактировать её:



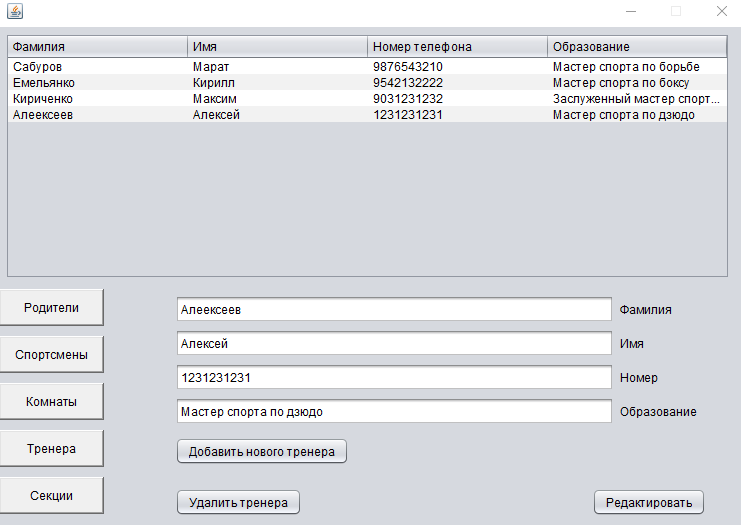
И удалим её:



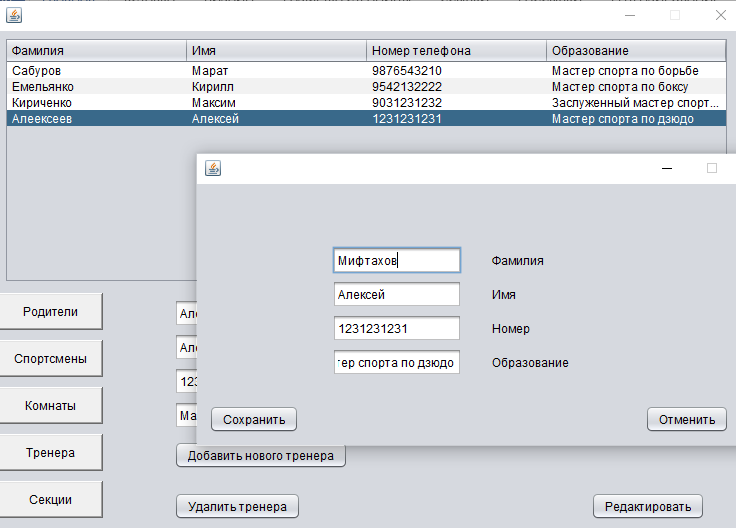
Откроем таблицу тренера:

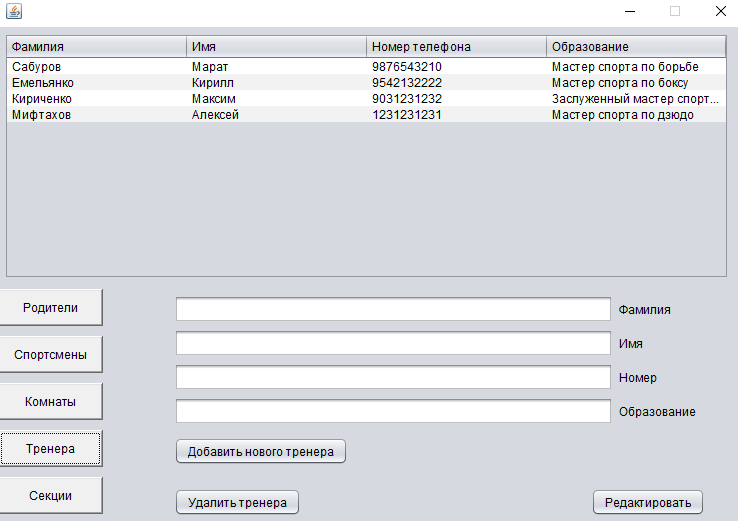


Здесь можно делать все те же действия. Добавим:

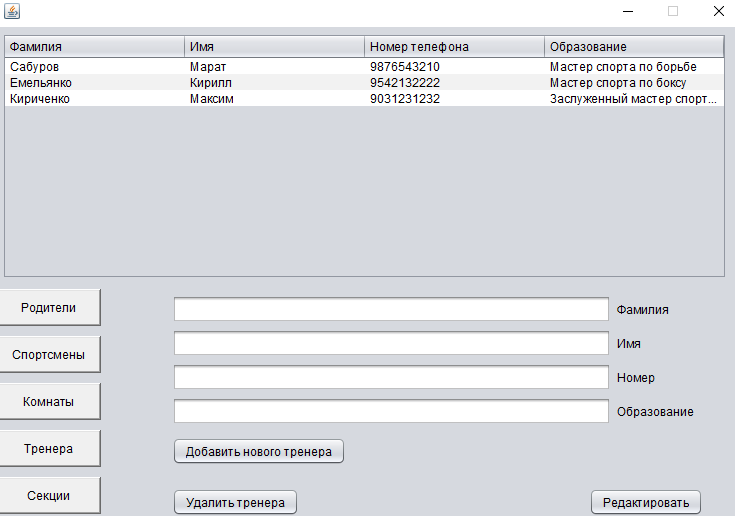


Можно так же редактировать:



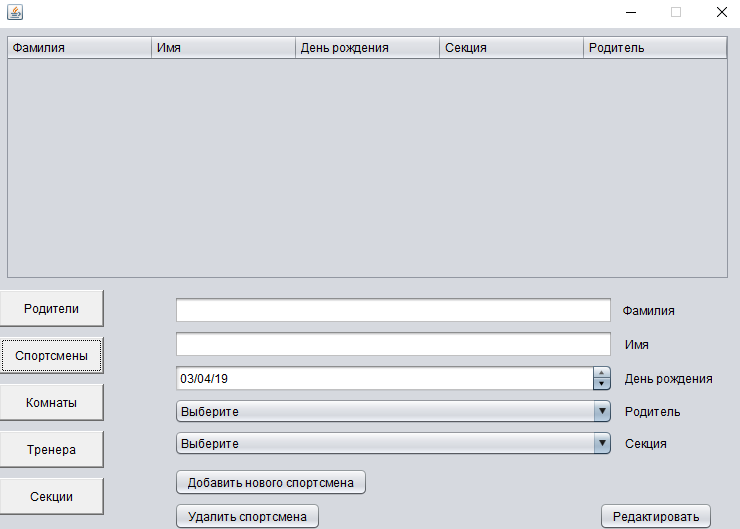


И удалять:



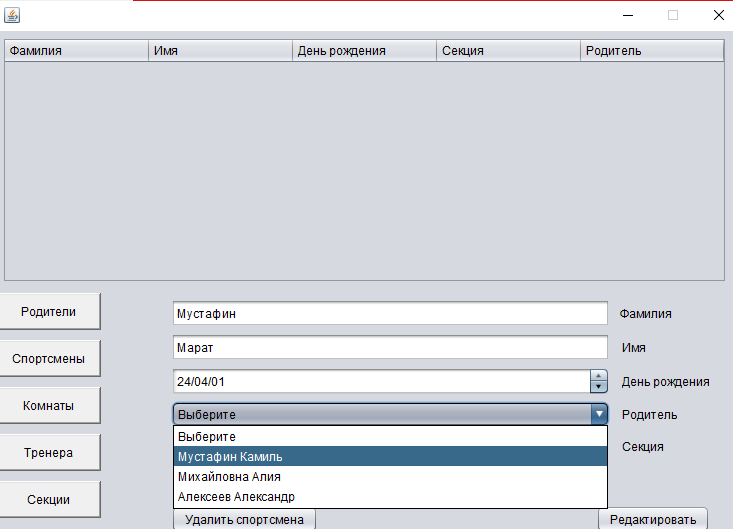
**ОЧЕНЬ ВАЖНО**: чтобы удалять какую-либо информацию в уже выше перечисленных таблицах, нужно, чтобы они были «откреплены» от связанных с ними другими записями! Например: если мы хотим удалить тренера, то нужно сначала указать другого тренера в той секции где он вёл занятия и по такой аналогии для всех сущностей.

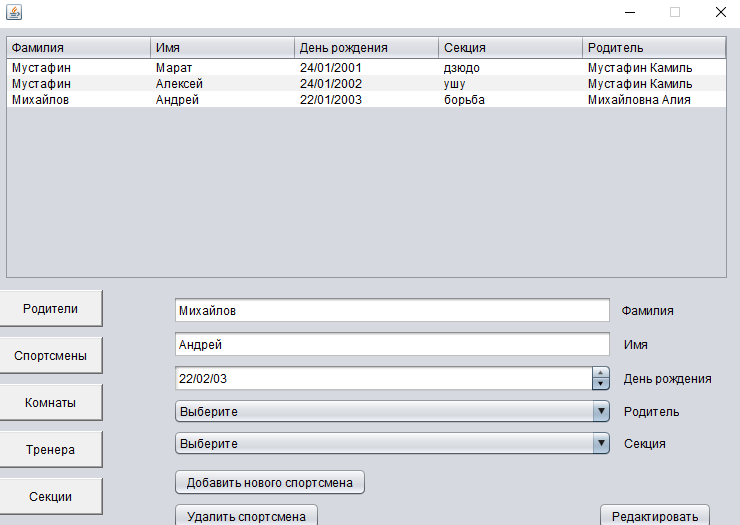
Теперь разберем кнопки спортсмены и секции и начнем с спортсменов:



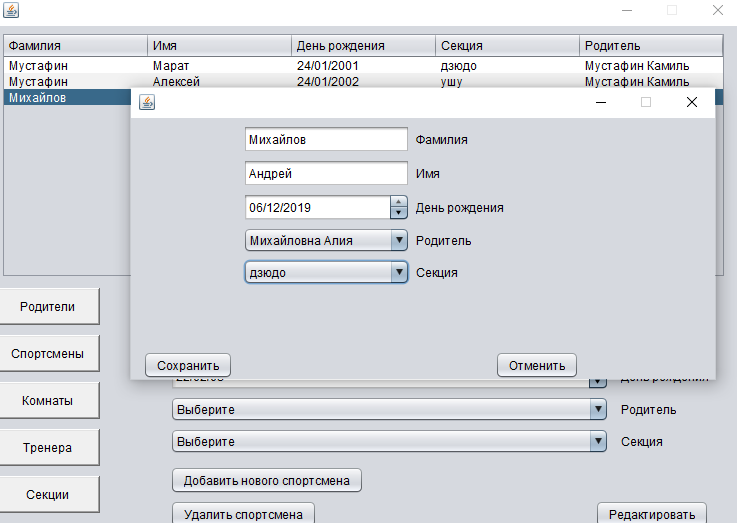
**Для того чтобы добавить нового спортсмена нужно чтобы родители и секции были заранее заполнены, иначе выбирать будет не из чего!**

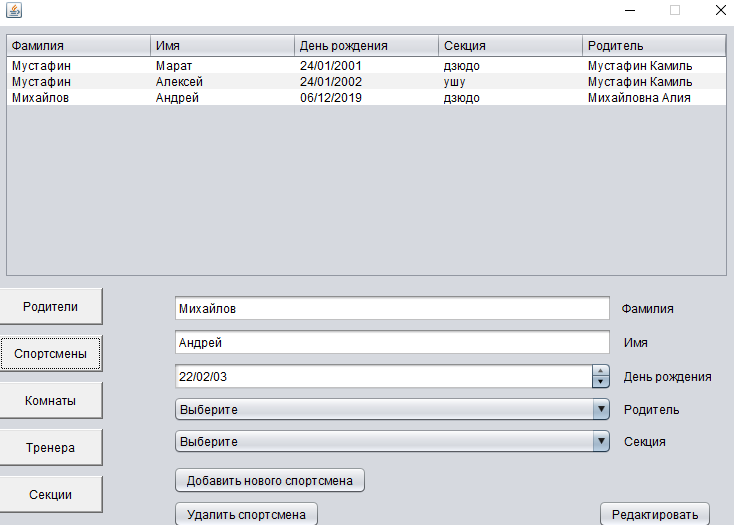
Вводим фамилию и имя, день рождения можно написать самому, а можно нажимать на стрелочки и таким образом выбрать день рождения. Родителя и секцию выбираем из выпадающего списка.



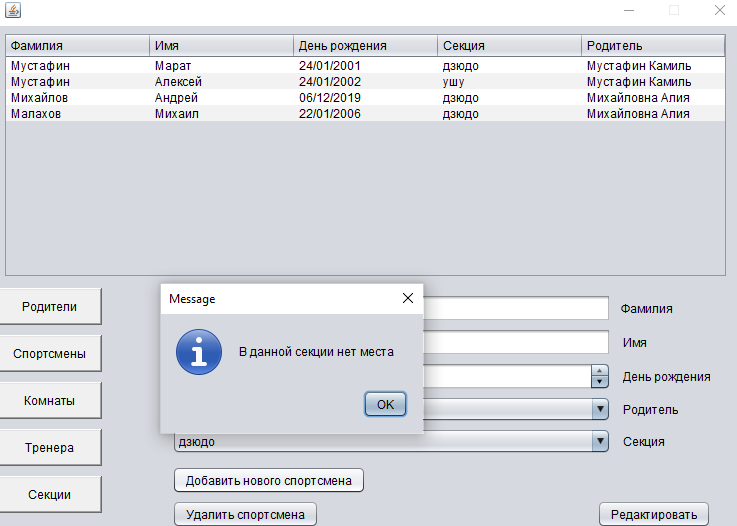


Попробуем отредактировать:

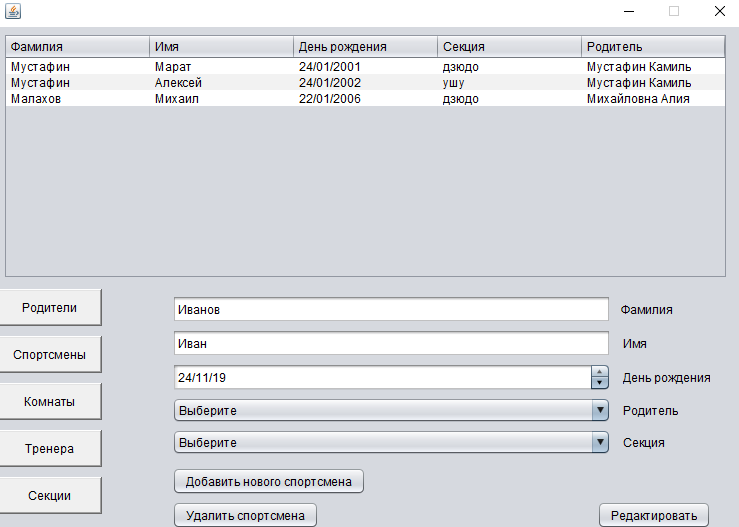




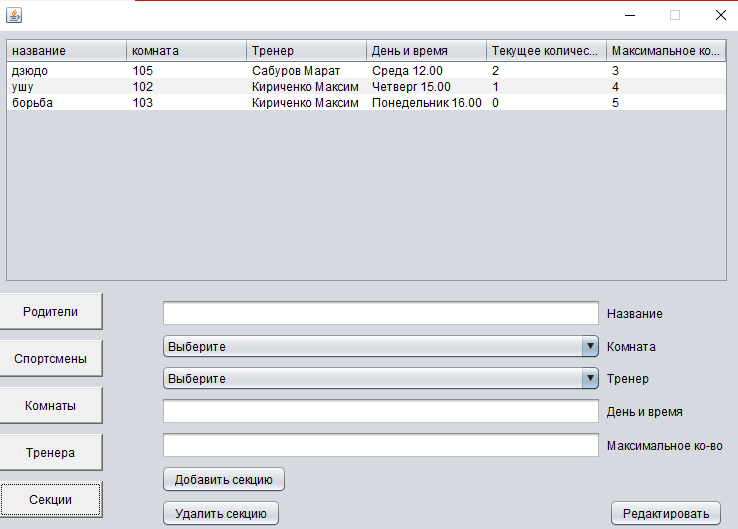
Если попробовать слишком много детей в одну секцию: выйдет ошибка, которая будет говорить, что места в данной секции закончены.



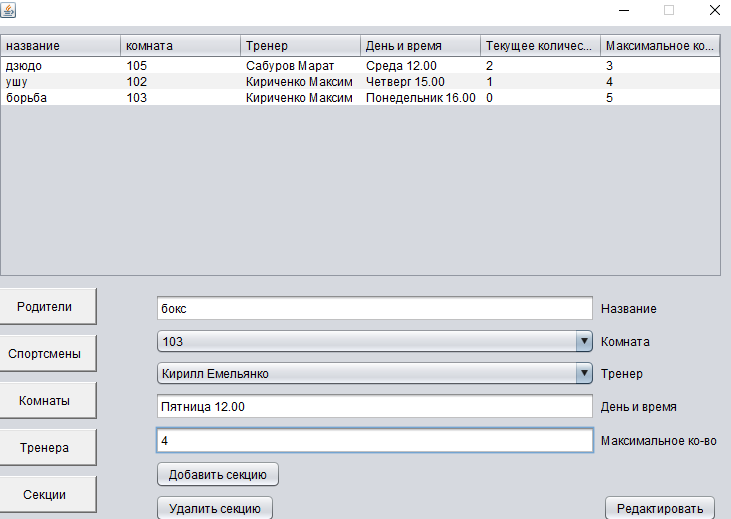
Как и было ранее, для удаления нужно выделить спортсмена и удалить его:

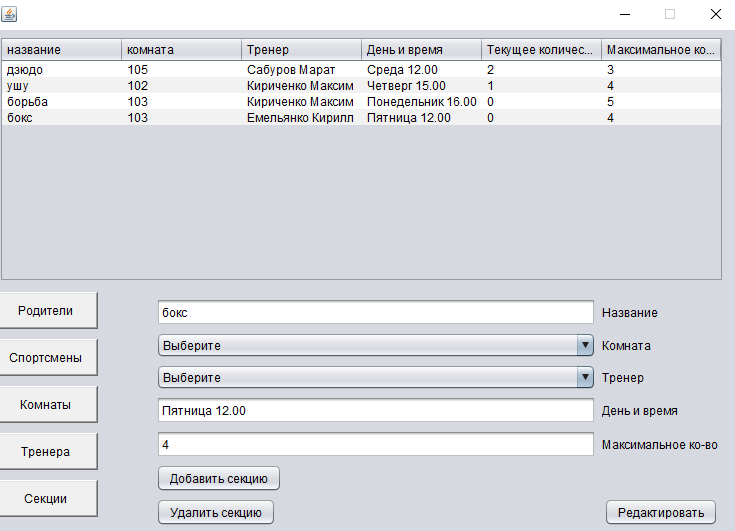


Ну и на конец – секции:

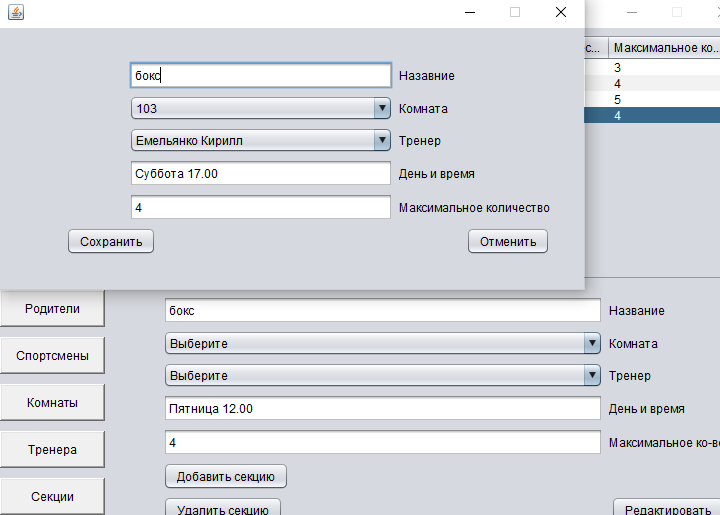


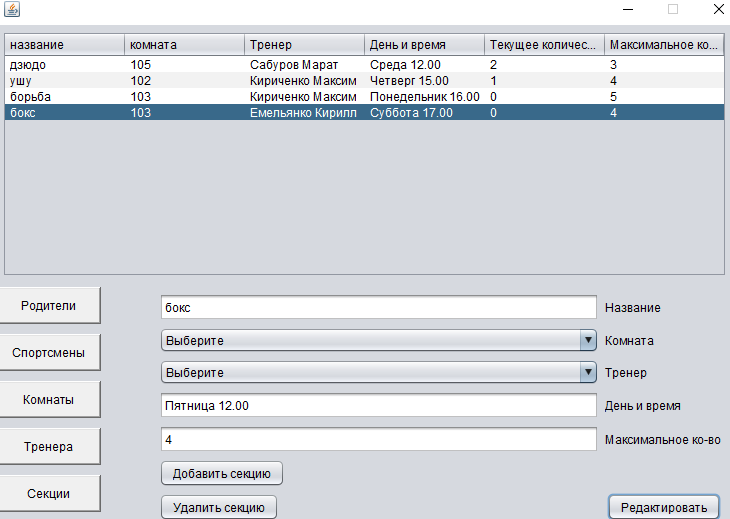
Для добавления новой секции, нужно, чтобы изначально были созданы комнаты и тренера. Для добавления секции пишем название секции, затем выбираем из выпадающих списков комнату и тренера, затем пишем день недели и время, на последнее максимальное количество спортсменов:



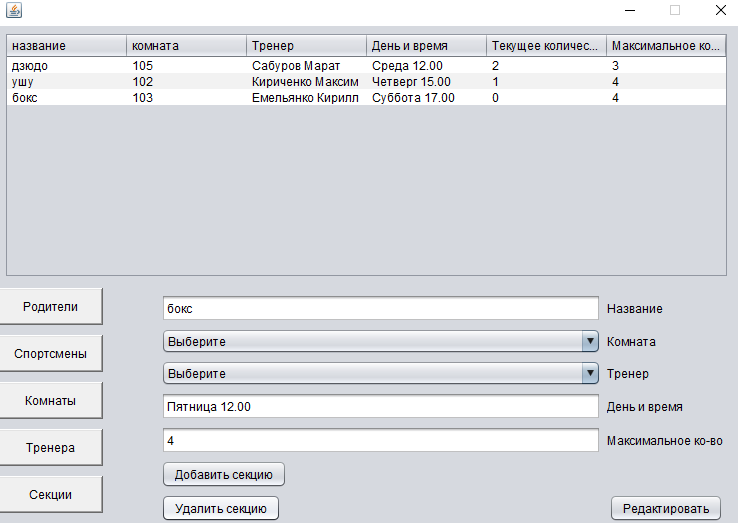


Попробуем отредактировать бокс, поменяв ему день и время:





И попробуем удалить борьбу



Как видно из скриншотов программа отлажена.

# **Список литературы**

1. www.iprbookshop.ru – Электронно-библиотечная система «Университетская библиотека онлайн»

2. Конноли Т. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. . – СПб.:Питер, 2017 г.

3. Туманов, В. Е. Основы проектирования реляционных баз данных / В. Е. Туманов. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 502 c. — ISBN 978-5-94774-713-3. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/52221.html

4. Ульман Д. Системы баз данных. Полный курс. – СПб.:Питер, 2017 г.

5. Кумскова И., Базы данных. Учебник | Кумскова Ирина Александровна, — СПб.:Питер, 2019 c.

6. Марц Н. Большие данные. Принципы и практика построения масштабируемых систем обработки данных в реальном времени - Символ-Плюс, 2018 г.

7. Швецов, В. И. Базы данных / В. И. Швецов. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 218 c. — ISBN 2227-8397. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/52139.html

8. Система ответов и вопросов о программировании Stack Overflow. – Режим доступа: https://stackoverflow.com

9. Осипов, Д. Л. InterBase и Delphi. Клиент-серверные базы данных / Д. Л. Осипов. — Саратов : Профобразование, 2017. — 536 c. — ISBN 978-5-4488-0050-4. — Текст : электронный //Электронно-библиотечная система IPR BOOKS:[сайт]. — URL:http://www.iprbookshop.ru/64056.html

10. Интернет-энциклопедия Wikipedia. – Режим доступа: https://www.wikipedia.org

# **Приложение 1**

**Файл test.java**

package javaapplication1;

import java.awt.event.KeyEvent;

import java.sql.\*;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.swing.JOptionPane;

import javax.swing.table.DefaultTableModel;

public class test extends javax.swing.JFrame {

Connection con;

Statement st;

DefaultTableModel mainTM;

String pattern = "dd/MM/yyyy";

SimpleDateFormat simpleDateFormat = new SimpleDateFormat(pattern);

javax.swing.JPanel lastPanel;

int widthPanel = 550, heightPanel = 200;

ArrayList<Integer> idOfParent, idOfAthlete, idOfRoom, idOfCoach, idOfSection;

//конструктор для класса

public test() {

initComponents();

lastPanel = athletePanel;

athletePanel.setVisible(false);

parentPanel.setVisible(false);

coachPanel.setVisible(false);

roomPanel.setVisible(false);

sectionPanel.setVisible(false);

mainTM = (DefaultTableModel) mainTable.getModel();

try{

Class.forName("net.ucanaccess.jdbc.UcanaccessDriver");

con = DriverManager.getConnection("jdbc:ucanaccess://db.accdb");

st = con.createStatement();

}

catch (Exception e){

System.out.println("not connected");

}

}

//при нажатии на кнопку спортсмены

private void athleteButtonActionPerformed(java.awt.event.ActionEvent evt) {

lastPanel.setVisible(false);

lastPanel = athletePanel;

athletePanel.setVisible(true);

athletePanel.setSize(widthPanel, heightPanel);

//работа с основной таблицей

try {

ResultSet rs = st.executeQuery("SELECT \* FROM athlete");

//очищащем таблицу

mainTM.setColumnCount(0);

mainTM.addColumn("Фамилия");

mainTM.addColumn("Имя");

mainTM.addColumn("День рождения");

mainTM.addColumn("Секция");

mainTM.addColumn("Родитель");

mainTM.setRowCount(0);

idOfAthlete = new ArrayList<Integer>();

//заполняем таблицу

while (rs.next()){

String str = rs.getString("idsection");

ResultSet sec = st.executeQuery("SELECT name FROM section WHERE id = '" + str + "'"); //sectionName

sec.next();

String idp = rs.getString("idparent");

ResultSet parent = st.executeQuery("select name, sur from parent where id = '" + idp + "'");

parent.next();

mainTM.addRow(new Object[]{rs.getString("sur"),rs.getString("name"),rs.getString("birthday"), sec.getString("name"), parent.getString("sur") + " " + parent.getString("name")});

idOfAthlete.add(Integer.valueOf(rs.getString("id")));

}

} catch (Exception ex) { //если ошибка

Logger.getLogger(test.class.getName()).log(Level.SEVERE, null, ex);

}

//работа с нижней панелькой

try{

//заполняем выбор родителей

ResultSet parent = st.executeQuery("SELECT \* FROM parent");

parentComboBox.removeAllItems();

parentComboBox.addItem("Выберите");

while (parent.next()){

parentComboBox.addItem(parent.getString("SUR") + " " + parent.getString("name"));

}

//заполняем выбор секции

ResultSet section = st.executeQuery("SELECT \* FROM section");

sectionComboBox.removeAllItems();

sectionComboBox.addItem("Выберите");

while (section.next()){

sectionComboBox.addItem(section.getString("name"));

}

} catch (Exception ex){

Logger.getLogger(test.class.getName()).log(Level.SEVERE, null, ex);

}

}

//при нажатии на кнопку тренера

private void coachButtonActionPerformed(java.awt.event.ActionEvent evt) {

lastPanel.setVisible(false);

lastPanel = coachPanel;

coachPanel.setVisible(true);

coachPanel.setSize(widthPanel, heightPanel);

try {

ResultSet rs = st.executeQuery("SELECT \* FROM teacher");

//очищащем таблицу

mainTM.setColumnCount(0);

mainTM.addColumn("Фамилия");

mainTM.addColumn("Имя");

mainTM.addColumn("Номер телефона");

mainTM.addColumn("Образование");

mainTM.setRowCount(0);

idOfCoach = new ArrayList<Integer>();

//заполняем таблицу

while (rs.next()){

mainTM.addRow(new Object[]{rs.getString("sur"),rs.getString("name"),rs.getString("phone"), rs.getString("education")});

idOfCoach.add(Integer.valueOf(rs.getString("id")));

}

} catch (Exception ex) { //если ошибка

Logger.getLogger(test.class.getName()).log(Level.SEVERE, null, ex);

}

}

//при нажатии на кнопку секции

private void sectionButtonActionPerformed(java.awt.event.ActionEvent evt) {

lastPanel.setVisible(false);

lastPanel = sectionPanel;

sectionPanel.setVisible(true);

try {

ResultSet rs = st.executeQuery("SELECT \* FROM section");

//очищащем таблицу

mainTM.setColumnCount(0);

mainTM.addColumn("название");

mainTM.addColumn("комната");

mainTM.addColumn("Тренер");

mainTM.addColumn("День и время");

mainTM.addColumn("Текущее количество");

mainTM.addColumn("Максимальное количество");

mainTM.setRowCount(0);

idOfSection = new ArrayList<Integer>();

//заполняем таблицу

while (rs.next()){

//название комнаты

String str = rs.getString("idroom");

ResultSet room = st.executeQuery("SELECT num FROM room WHERE id = '" + str + "'");

room.next();

//тренер

String str1 = rs.getString("idteacher");

ResultSet t = st.executeQuery("SELECT sur, name FROM teacher WHERE id = '" + str1 + "'");

t.next();

//текущее количество спортсменов на данной секции

ResultSet count = st.executeQuery("SELECT COUNT(\*) FROM athlete WHERE idsection = '" + rs.getString("id") +"'");

count.next();

//заполняем

mainTM.addRow(new Object[]{

rs.getString("name"),

room.getString("num"),

t.getString("sur") + " " + t.getString("name"),

rs.getString("dayTime"),

count.getString("C1"),

rs.getString("maxCount")});

idOfSection.add(Integer.valueOf(rs.getString("id")));

}

} catch (Exception ex) { //если ошибка

Logger.getLogger(test.class.getName()).log(Level.SEVERE, null, ex);

}

try{

//заполняем выбор комнат

ResultSet room = st.executeQuery("SELECT \* FROM room");

roomSecComboBox.removeAllItems();

roomSecComboBox.addItem("Выберите");

while (room.next()){

roomSecComboBox.addItem(room.getString("num"));

}

//заполняем выбор тренеров

ResultSet teacher = st.executeQuery("SELECT \* FROM teacher");

coachSecComboBox.removeAllItems();

coachSecComboBox.addItem("Выберите");

while (teacher.next()){

coachSecComboBox.addItem(teacher.getString("name") + " " + teacher.getString("sur"));

}

} catch (Exception ex){

Logger.getLogger(test.class.getName()).log(Level.SEVERE, null, ex);

}

}

//при нажатии на кнопку родители

private void parentButtonActionPerformed(java.awt.event.ActionEvent evt) {

lastPanel.setVisible(false);

lastPanel = parentPanel;

parentPanel.setVisible(true);

parentPanel.setSize(widthPanel, heightPanel);

try {

ResultSet rs = st.executeQuery("SELECT \* FROM parent");

//очищащем таблицу

mainTM.setColumnCount(0);

mainTM.addColumn("Фамилия");

mainTM.addColumn("Имя");

mainTM.addColumn("Номер телефона");

mainTM.setRowCount(0);

idOfParent = new ArrayList<Integer>();

//заполняем таблицу

while (rs.next()){

mainTM.addRow(new Object[]{rs.getString("sur"),rs.getString("name"),rs.getString("phone")});

idOfParent.add(Integer.valueOf(rs.getString("id")));

}

} catch (Exception ex) { //если ошибка

Logger.getLogger(test.class.getName()).log(Level.SEVERE, null, ex);

}

}

//при нажатии на кнопку комнаты

private void roomButtonActionPerformed(java.awt.event.ActionEvent evt) {

lastPanel.setVisible(false);

lastPanel = roomPanel;

roomPanel.setVisible(true);

roomPanel.setSize(widthPanel, heightPanel);

try {

ResultSet rs = st.executeQuery("SELECT \* FROM room");

//очищащем таблицу

mainTM.setColumnCount(0);

mainTM.addColumn("Название комнаты");

mainTM.setRowCount(0);

idOfRoom = new ArrayList<Integer>();

//заполняем таблицу

while (rs.next()){

mainTM.addRow(new Object[]{rs.getString("num")});

idOfRoom.add(Integer.valueOf(rs.getString("id")));

}

} catch (Exception ex) { //если ошибка

Logger.getLogger(test.class.getName()).log(Level.SEVERE, null, ex);

}

}

//ОБРАБОТЧИК для контроля ввода цифр

private void numParEditKeyTyped(java.awt.event.KeyEvent evt) {

char c = evt.getKeyChar();

if (!(Character.isDigit(c)

|| (c == KeyEvent.VK\_DELETE)

|| (c == KeyEvent.VK\_BACK\_SPACE)

|| Character.isWhitespace(c))){

evt.consume();

JOptionPane.showMessageDialog(null, "Можно вводить только цифры");

}

}

//кнопка для редактирования спортсмена

private void editAthleteButtonActionPerformed(java.awt.event.ActionEvent evt) {

int a = mainTable.getSelectedRow();

if (a > -1){

try {

ResultSet parents = st.executeQuery("select \* from parent");

ResultSet section = st.executeQuery("select \* from section");

String q = "SELECT id FROM athlete WHERE name='" + mainTM.getValueAt(a, 1).toString()+"' AND sur = '" + mainTM.getValueAt(a, 0) + "'";

System.out.println(q);

ResultSet childId = st.executeQuery(q);

childId.next();

editForm edit = new editForm();

if (!edit.editAthPanel.isVisible())

edit.editAthPanel.setVisible(true);

edit.idAthleteLabel.setText("id = " + childId.getString("id"));

edit.jTextField1.setText(mainTM.getValueAt(a, 1).toString());

edit.jTextField2.setText(mainTM.getValueAt(a, 0).toString());

edit.parentComboBox.removeAllItems();

edit.parentComboBox.addItem("Выберите");

while (parents.next()){

edit.parentComboBox.addItem(parents.getString("sur") + " " + parents.getString("name"));

}

edit.parentComboBox.setSelectedItem(mainTM.getValueAt(a, 4));

edit.sectionComboBox.removeAllItems();

edit.sectionComboBox.addItem("Выберите");

while (section.next()){

edit.sectionComboBox.addItem(section.getString("name"));

}

edit.sectionComboBox.setSelectedItem(mainTM.getValueAt(a, 3));

edit.editAthPanel.setVisible(true);

edit.setVisible(true);

} catch (Exception ex){

System.out.println(ex);

}

} else {

JOptionPane.showMessageDialog(this, "Выберите спортсмена");

}

}

//кнопка для удаления спортсмена

private void deleteAthleteButtonActionPerformed(java.awt.event.ActionEvent evt) {

int a = mainTable.getSelectedRow();

if (a > -1){

try {

st.executeUpdate("delete from athlete where id = " + idOfAthlete.get(a));

athleteButtonActionPerformed(evt);

} catch (Exception ex){

Logger.getLogger(test.class.getName()).log(Level.SEVERE, null, ex);

}

} else {

JOptionPane.showMessageDialog(this, "Выберите спорстмена");

}

}

//кнопка для добавления нового спортсмена

private void addNewAthleteButtonActionPerformed(java.awt.event.ActionEvent evt) {

//если все поля заполнены

if (!(surAthEdit.getText().equals("") || nameAthEdit.getText().equals("") || sectionComboBox.getSelectedIndex() == 0 || parentComboBox.getSelectedIndex() == 0)){

String selectedSection = sectionComboBox.getSelectedItem().toString();

try {

ResultSet idSec = st.executeQuery("SELECT id FROM section where name = '" + selectedSection + "'");

idSec.next();

String idSection = idSec.getString("ID");

ResultSet count = st.executeQuery("SELECT COUNT(\*) FROM athlete WHERE idsection = '" + idSection +"'");

count.next();

ResultSet maxCount = st.executeQuery("SELECT \* FROM section where id = '" + idSection + "'");

maxCount.next();

String selectedParent = parentComboBox.getSelectedItem().toString();

int s = selectedParent.indexOf(" ");

String parName = selectedParent.substring(s+1, selectedParent.length());

String parSur = selectedParent.substring(0, s);

String qq = "select id from parent where sur = '"+parSur+"' and name = '" + parName + "'";

ResultSet idParent = st.executeQuery(qq);

idParent.next();

String s1 = count.getString("C1"), s2 = maxCount.getString("maxcount");

String date = simpleDateFormat.format(birthdaySpinner.getValue());

String values = "'"

+surAthEdit.getText()+"','"

+nameAthEdit.getText()+"','"

+date+"','"

+idSection+"','"

+idParent.getString("id")+"'";

System.out.println(values);

if (!s1.equals(s2)){

st.executeUpdate("insert into athlete(sur,name,birthday,idsection,idparent) values ("+values+")");

athleteButtonActionPerformed(evt);

} else {

JOptionPane.showMessageDialog(this, "В данной секции нет места");

}

}catch (Exception ex){

Logger.getLogger(test.class.getName()).log(Level.SEVERE, null, ex);

}

} else {

JOptionPane.showMessageDialog(this, "Заполните все поля");

}

}

//ОБРАБОТЧИК вводим данных для слов

private void surAthEditKeyTyped(java.awt.event.KeyEvent evt) {

char c = evt.getKeyChar();

if (!(Character.isDigit(c) || Character.isAlphabetic(c)

|| (c == KeyEvent.VK\_DELETE)

|| (c == KeyEvent.VK\_BACK\_SPACE)

|| (c == KeyEvent.VK\_PERIOD)

|| Character.isWhitespace(c))){

evt.consume();

JOptionPane.showMessageDialog(null, "Можно вводить только латинские буквы, кириллицу и цифры");

}

}

//кнопка для добавления родителя

private void addParButtonActionPerformed(java.awt.event.ActionEvent evt) {

if (!(nameParEdit.getText().equals("") || surParEdit.getText().equals("") || numParEdit.getText().equals(""))){

try {

String values = "'" + nameParEdit.getText() + "', '" + surParEdit.getText() + "', '" + numParEdit.getText()+"'";

System.out.println(values);

st.executeUpdate("insert into parent(name, sur, phone) values("+values+")");

parentButtonActionPerformed(evt);

} catch (Exception e){

System.out.print(e);

}

} else {

JOptionPane.showMessageDialog(this, "Заполните все поля");

}

}

//кнопка для удаления родителя

private void deletParButtonActionPerformed(java.awt.event.ActionEvent evt) {

int a = mainTable.getSelectedRow();

if (a > -1){

try {

st.executeUpdate("delete from parent where id = " + idOfParent.get(a));

parentButtonActionPerformed(evt);

} catch (Exception e){

System.out.println(e);

}

} else {

JOptionPane.showMessageDialog(this, "Выберите родителя");

}

}

//кнопка для редактирования родителя

private void editParButtonActionPerformed(java.awt.event.ActionEvent evt) {

int a = mainTable.getSelectedRow();

if (a > -1){

try{

editForm edit = new editForm();

edit.idParLabel.setText(idOfParent.get(a).toString());

edit.surParEdit.setText(mainTM.getValueAt(a, 0).toString());

edit.nameParEdit.setText(mainTM.getValueAt(a, 1).toString());

edit.numParEdit.setText(mainTM.getValueAt(a, 2).toString());

edit.editParPanel.setVisible(true);

edit.setVisible(true);

} catch (Exception e){

System.out.print(e);

}

} else {

JOptionPane.showMessageDialog(this, "Выберите родителя");

}

}

//кнопка для добавления тренера

private void addCoachButtonActionPerformed(java.awt.event.ActionEvent evt) {

if (!(nameCoachEdit.getText().equals("") || surCoachEdit.getText().equals("") || numCoachEdit.getText().equals(""))){

try {

String values = "'" + surCoachEdit.getText() + "', '" + nameCoachEdit.getText() + "', '" + numCoachEdit.getText()+"','"+edCoachEdit.getText()+"'";

System.out.println(values);

st.executeUpdate("insert into teacher(name, sur, phone, education) values("+values+")");

coachButtonActionPerformed(evt);

} catch (Exception e){

System.out.print(e);

}

} else {

JOptionPane.showMessageDialog(this, "Заполните все поля");

}

}

//кнопка для удаления тренера

private void delCoachButtonActionPerformed(java.awt.event.ActionEvent evt) {

int a = mainTable.getSelectedRow();

if (a > -1){

try {

st.executeUpdate("delete from teacher where id = " + idOfCoach.get(a));

coachButtonActionPerformed(evt);

} catch (Exception e){

System.out.println(e);

}

} else {

JOptionPane.showMessageDialog(this, "Выберите тренера");

}

}

//кнопка для редактирования тренера

private void editCoachButtonActionPerformed(java.awt.event.ActionEvent evt) {

int a = mainTable.getSelectedRow();

if (a > -1){

editForm edit = new editForm();

edit.idCoachLabel1.setText(idOfCoach.get(a).toString());

edit.surCoachEdit1.setText(mainTM.getValueAt(a, 0).toString());

edit.nameCoachEdit1.setText(mainTM.getValueAt(a, 1).toString());

edit.numCoachEdit1.setText(mainTM.getValueAt(a, 2).toString());

edit.edCoachEdit1.setText(mainTM.getValueAt(a, 3).toString());

edit.editCoachPanel.setVisible(true);

edit.setVisible(true);

} else {

JOptionPane.showMessageDialog(this, "Выберите трененера");

}

}

//кнопка для добавления комнаты

private void addRoomButtonActionPerformed(java.awt.event.ActionEvent evt) {

if (!(nameRoomEdit.getText().equals(""))){

try {

String values = "'" + nameRoomEdit.getText() +"'";

System.out.println(values);

st.executeUpdate("insert into room(num) values("+values+")");

roomButtonActionPerformed(evt);

} catch (Exception e){

System.out.print(e);

}

} else {

JOptionPane.showMessageDialog(this, "Заполните все поля");

}

}

//кнопка для удаления комнаты

private void delRoomButtonActionPerformed(java.awt.event.ActionEvent evt) {

int a = mainTable.getSelectedRow();

if (a > -1){

try {

st.executeUpdate("delete from room where id = " + idOfRoom.get(a));

roomButtonActionPerformed(evt);

} catch (Exception e){

System.out.println(e);

}

} else {

JOptionPane.showMessageDialog(this, "Выберите комнату");

}

}

//кнопка для редактирования комнаты

private void editRoomButtonActionPerformed(java.awt.event.ActionEvent evt) {

int a = mainTable.getSelectedRow();

if (a > -1){

editForm edit = new editForm();

edit.idRoomLabel.setText(idOfRoom.get(a).toString());

edit.nameRoomEdit.setText(mainTM.getValueAt(a, 0).toString());

edit.editRoomPanel.setVisible(true);

edit.setVisible(true);

} else {

JOptionPane.showMessageDialog(this, "Выберите комнату");

}

}

//кнопка для добавления секции

private void addSecButtonActionPerformed(java.awt.event.ActionEvent evt) {

if (!(nameSecEdit.getText().equals("") || dayTimeSecEdit.getText().equals("") || maxCountSecEdit.getText().equals("") || coachSecComboBox.getSelectedIndex() == 0 || roomSecComboBox.getSelectedIndex() == 0)){

try {

String selectedCoach = coachSecComboBox.getSelectedItem().toString();

int s = selectedCoach.indexOf(" ");

String coachSur = selectedCoach.substring(s+1, selectedCoach.length());

String coachName = selectedCoach.substring(0, s);

String selectedRoom = roomSecComboBox.getSelectedItem().toString();

try {

ResultSet idCoach = st.executeQuery("select id from teacher where sur = '" + coachSur + "' and name = '" + coachName + "'");

idCoach.next();

ResultSet idRoom = st.executeQuery("select id from room where num = '" + selectedRoom + "'");

idRoom.next();

String values = "'" + nameSecEdit.getText() +"', '" + dayTimeSecEdit.getText() + "', '" + maxCountSecEdit.getText() + "', '" + idCoach.getString("id") + "', '" + idRoom.getString("id") + "'";

System.out.println(values);

st.executeUpdate("insert into section(name, dayTime, maxCount, idteacher, idroom) values("+values+")");

sectionButtonActionPerformed(evt);

} catch (Exception ex){

System.out.println(ex);

}

} catch (Exception e){

System.out.print(e);

}

} else {

JOptionPane.showMessageDialog(this, "Заполните все поля");

}

}

//кнопка для удаления секции

private void delSecButtonActionPerformed(java.awt.event.ActionEvent evt) {

int a = mainTable.getSelectedRow();

if (a > -1){

try {

st.executeUpdate("delete from section where id = " + idOfSection.get(a));

sectionButtonActionPerformed(evt);

} catch (Exception e){

System.out.println(e);

}

} else {

JOptionPane.showMessageDialog(this, "Выберите секцию");

}

}

//кнопка для обработки секции

private void editSecButtonActionPerformed(java.awt.event.ActionEvent evt) {

int a = mainTable.getSelectedRow();

if (a > -1){

try {

editForm edit = new editForm();

edit.idSecLabel.setText(idOfSection.get(a).toString());

edit.nameSecEdit.setText(mainTM.getValueAt(a, 0).toString());

edit.dayTimeSecEdit.setText(mainTM.getValueAt(a, 3).toString());

edit.maxCountSecEdit.setText(mainTM.getValueAt(a, 5).toString());

ResultSet room = st.executeQuery("select \* from room");

edit.roomComboBox.removeAllItems();

edit.roomComboBox.addItem("Выберите");

while(room.next()){

edit.roomComboBox.addItem(room.getString("num"));

}

edit.roomComboBox.setSelectedItem(mainTM.getValueAt(a, 1));

ResultSet coach = st.executeQuery("select \* from teacher");

edit.coachComboBox.removeAllItems();

edit.coachComboBox.addItem("Выберите");

while(coach.next()){

edit.coachComboBox.addItem(coach.getString("sur") + " " + coach.getString("name"));

}

edit.coachComboBox.setSelectedItem(mainTM.getValueAt(a, 2));

edit.editSectionPanel.setVisible(true);

edit.setVisible(true);

} catch (Exception e){

System.out.println(e);

}

} else {

JOptionPane.showMessageDialog(this, "Выберите секцию");

}

}

//Дальше идут созданные компоненты

// Variables declaration - do not modify

private javax.swing.JButton addCoachButton;

private javax.swing.JButton addNewAthleteButton;

private javax.swing.JButton addParButton;

private javax.swing.JButton addRoomButton;

private javax.swing.JButton addSecButton;

public java.awt.Button athleteButton;

private javax.swing.JPanel athletePanel;

private javax.swing.JSpinner birthdaySpinner;

private java.awt.Button coachButton;

private javax.swing.JPanel coachPanel;

private javax.swing.JComboBox<String> coachSecComboBox;

private javax.swing.JTextField dayTimeSecEdit;

private javax.swing.JButton delCoachButton;

private javax.swing.JButton delRoomButton;

private javax.swing.JButton delSecButton;

private javax.swing.JButton deletParButton;

private javax.swing.JButton deleteAthleteButton;

private javax.swing.JTextField edCoachEdit;

private javax.swing.JButton editAthleteButton;

private javax.swing.JButton editCoachButton;

private javax.swing.JButton editParButton;

private javax.swing.JButton editRoomButton;

private javax.swing.JButton editSecButton;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel10;

private javax.swing.JLabel jLabel11;

private javax.swing.JLabel jLabel12;

private javax.swing.JLabel jLabel13;

private javax.swing.JLabel jLabel14;

private javax.swing.JLabel jLabel15;

private javax.swing.JLabel jLabel16;

private javax.swing.JLabel jLabel17;

private javax.swing.JLabel jLabel18;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JLabel jLabel4;

private javax.swing.JLabel jLabel5;

private javax.swing.JLabel jLabel6;

private javax.swing.JLabel jLabel7;

private javax.swing.JLabel jLabel8;

private javax.swing.JLabel jLabel9;

private javax.swing.JPanel jPanel1;

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JScrollPane jScrollPane2;

private javax.swing.JTable jTable1;

private javax.swing.JTable mainTable;

private javax.swing.JTextField maxCountSecEdit;

private javax.swing.JTextField nameAthEdit;

private javax.swing.JTextField nameCoachEdit;

private javax.swing.JTextField nameParEdit;

private javax.swing.JTextField nameRoomEdit;

private javax.swing.JTextField nameSecEdit;

private javax.swing.JTextField numCoachEdit;

private javax.swing.JTextField numParEdit;

private java.awt.Button parentButton;

private javax.swing.JComboBox<String> parentComboBox;

private javax.swing.JPanel parentPanel;

private java.awt.Button roomButton;

private javax.swing.JPanel roomPanel;

private javax.swing.JComboBox<String> roomSecComboBox;

private java.awt.Button sectionButton;

private javax.swing.JComboBox<String> sectionComboBox;

private javax.swing.JPanel sectionPanel;

private javax.swing.JTextField surAthEdit;

private javax.swing.JTextField surCoachEdit;

private javax.swing.JTextField surParEdit;

// End of variables declaration

}

**Файл editForm.java**

package javaapplication1;

import java.awt.event.KeyEvent;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.Statement;

import java.text.SimpleDateFormat;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.swing.JOptionPane;

public class editForm extends javax.swing.JFrame {

Connection con;

Statement st;

public editForm() {

initComponents();

editAthPanel.setVisible(false);

editParPanel.setVisible(false);

editRoomPanel.setVisible(false);

editCoachPanel.setVisible(false);

editSectionPanel.setVisible(false);

try{

Class.forName("net.ucanaccess.jdbc.UcanaccessDriver");

con = DriverManager.getConnection("jdbc:ucanaccess://db.accdb");

st = con.createStatement();

}

catch (Exception e){

System.out.println("not connected");

}

}

//при нажатии на кнопку сохранить в редактировании спортсмена

private void okButtonActionPerformed(java.awt.event.ActionEvent evt) {

if (!(jTextField1.getText().equals("") || jTextField2.getText().equals("") || sectionComboBox.getSelectedIndex() == 0 || parentComboBox.getSelectedIndex() == 0)){

try{

ResultSet idSec = st.executeQuery("SELECT id FROM section where name = '" + sectionComboBox.getSelectedItem().toString() + "'");

idSec.next();

String idSection = idSec.getString("ID");

String surnameOfParent = parentComboBox.getSelectedItem().toString();

int space = surnameOfParent.indexOf(" ");

surnameOfParent = surnameOfParent.substring(0, space);

ResultSet idParent = st.executeQuery("select id from parent where sur = '"+surnameOfParent+"'");

idParent.next();

String pattern = "dd/MM/yyyy";

SimpleDateFormat simpleDateFormat = new SimpleDateFormat(pattern);

String date = simpleDateFormat.format(birthdaySpinner.getValue());

String idAthlete = idAthleteLabel.getText();

idAthlete = idAthlete.substring(5, idAthlete.length());

String q = "update athlete set name = '" + jTextField1.getText() +"', sur = '"+jTextField2.getText()

+"', birthday = '" + date + "', idsection = '" + idSection + "', idparent = '" + idParent.getString("id")+"' "

+ "where id = '" + idAthlete + "'";

System.out.println(q);

st.executeUpdate(q);

this.setVisible(false);

}catch (Exception ex){

Logger.getLogger(test.class.getName()).log(Level.SEVERE, null, ex);

}

} else {

JOptionPane.showMessageDialog(this, "Заполните все поля");

}

}

// ОБРАБОТЧИК вводимых данных для цифр и слов

private void jTextField1KeyTyped(java.awt.event.KeyEvent evt) {

char c = evt.getKeyChar();

if (!(Character.isDigit(c) || Character.isAlphabetic(c)

|| (c == KeyEvent.VK\_DELETE)

|| (c == KeyEvent.VK\_BACK\_SPACE)

|| (c == KeyEvent.VK\_PERIOD)

|| Character.isWhitespace(c))){

evt.consume();

JOptionPane.showMessageDialog(null, "Можно вводить только латинские буквы, кириллицу и цифры");

}

}

//при нажатии на отмену во всех панелях

private void cancelButtonActionPerformed(java.awt.event.ActionEvent evt) {

this.setVisible(false);

}

//при нажатии на кнопку сохранить в родителе

private void okParButtonActionPerformed(java.awt.event.ActionEvent evt) {

if (!(surParEdit.getText().equals("") || nameParEdit.getText().equals("") || numParEdit.getText().equals(""))){

try{

String idParent = idParLabel.getText();

String q = "update parent set name = '" + nameParEdit.getText() +"', sur = '"

+surParEdit.getText() + "', phone = '" + numParEdit.getText() +"' "

+ "where id = '" + idParent + "'";

System.out.println(q);

st.executeUpdate(q);

this.setVisible(false);

}catch (Exception ex){

Logger.getLogger(test.class.getName()).log(Level.SEVERE, null, ex);

}

} else {

JOptionPane.showMessageDialog(this, "Заполните все поля");

}

}

// ОБРАБОТЧИК вводимых данных для цифр

private void numParEditKeyTyped(java.awt.event.KeyEvent evt) {

char c = evt.getKeyChar();

if (!(Character.isDigit(c)

|| (c == KeyEvent.VK\_DELETE)

|| (c == KeyEvent.VK\_BACK\_SPACE)

|| Character.isWhitespace(c))){

evt.consume();

JOptionPane.showMessageDialog(null, "Можно вводить только цифры");

}

}

//при нажатии на кнопку сохранить в комнате

private void okRoomButtonActionPerformed(java.awt.event.ActionEvent evt) {

if (!(nameRoomEdit.getText().equals(""))){

try{

String idRoom = idRoomLabel.getText();

String q = "update room set num = '" + nameRoomEdit.getText()

+ "' where id = '" + idRoom + "'";

System.out.println(q);

st.executeUpdate(q);

this.setVisible(false);

}catch (Exception ex){

Logger.getLogger(test.class.getName()).log(Level.SEVERE, null, ex);

}

} else {

JOptionPane.showMessageDialog(this, "Заполните поле");

}

}

//при нажатии на кнопку сохранить в тренере

private void okCouchButtonActionPerformed(java.awt.event.ActionEvent evt) {

if (!(surCoachEdit1.getText().equals("") || nameCoachEdit1.getText().equals("") || numCoachEdit1.getText().equals("") || edCoachEdit1.getText().equals(""))){

try{

String idCoach = idCoachLabel1.getText();

String q = "update teacher set name = '" + nameCoachEdit1.getText() +"', sur = '"

+surCoachEdit1.getText() + "', phone = '" + numCoachEdit1.getText() +"', education = '" + edCoachEdit1.getText() +"' "

+ "where id = '" + idCoach + "'";

System.out.println(q);

st.executeUpdate(q);

this.setVisible(false);

}catch (Exception ex){

Logger.getLogger(test.class.getName()).log(Level.SEVERE, null, ex);

}

} else {

JOptionPane.showMessageDialog(this, "Заполните все поля");

}

}

//при нажатии на кнопку сохранить в секции

private void okSecButtonActionPerformed(java.awt.event.ActionEvent evt) {

if (!(nameSecEdit.getText().equals("") || roomComboBox.getSelectedIndex() == 0 || coachComboBox.getSelectedIndex() == 0 || dayTimeSecEdit.getText().equals("") || maxCountSecEdit.getText().equals(""))){

try{

String selectedCoach = coachComboBox.getSelectedItem().toString();

int s = selectedCoach.indexOf(" ");

String coachName = selectedCoach.substring(s+1, selectedCoach.length());

String coachSur = selectedCoach.substring(0, s);

ResultSet idCoach = st.executeQuery("select id from teacher where sur = '" + coachSur + "' and name = '" + coachName + "'");

idCoach.next();

String selectedRoom = roomComboBox.getSelectedItem().toString();

ResultSet idRoom = st.executeQuery("select id from room where num = '" + selectedRoom + "'");

idRoom.next();

String idSec = idSecLabel.getText();

String q = "update section set name = '" + nameSecEdit.getText() +"', idRoom = '"

+ idRoom.getString("id") + "', idteacher = '" + idCoach.getString("id") +"', dayTime = '" + dayTimeSecEdit.getText() +"', maxCount = '" + maxCountSecEdit.getText() +"' "

+ "where id = '" + idSec + "'";

System.out.println(q);

st.executeUpdate(q);

this.setVisible(false);

}catch (Exception ex){

Logger.getLogger(test.class.getName()).log(Level.SEVERE, null, ex);

}

} else {

JOptionPane.showMessageDialog(this, "Заполните все поля");

}

}

//описываются все использованные компоненты

// Variables declaration - do not modify

public javax.swing.JSpinner birthdaySpinner;

private javax.swing.JButton cancelButton;

private javax.swing.JButton cancelButton1;

private javax.swing.JButton cancelButton2;

private javax.swing.JButton cancelButton3;

private javax.swing.JButton cancelSecButton;

public javax.swing.JComboBox<String> coachComboBox;

public javax.swing.JTextField dayTimeSecEdit;

public javax.swing.JTextField edCoachEdit1;

public javax.swing.JPanel editAthPanel;

public javax.swing.JPanel editCoachPanel;

public javax.swing.JPanel editParPanel;

public javax.swing.JPanel editRoomPanel;

public javax.swing.JPanel editSectionPanel;

public java.awt.Label idAthleteLabel;

public java.awt.Label idCoachLabel1;

public java.awt.Label idParLabel;

public java.awt.Label idRoomLabel;

public java.awt.Label idSecLabel;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel10;

private javax.swing.JLabel jLabel11;

private javax.swing.JLabel jLabel12;

private javax.swing.JLabel jLabel13;

private javax.swing.JLabel jLabel14;

private javax.swing.JLabel jLabel15;

private javax.swing.JLabel jLabel16;

private javax.swing.JLabel jLabel17;

private javax.swing.JLabel jLabel18;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JLabel jLabel4;

private javax.swing.JLabel jLabel5;

private javax.swing.JLabel jLabel6;

private javax.swing.JLabel jLabel7;

private javax.swing.JLabel jLabel8;

private javax.swing.JLabel jLabel9;

public javax.swing.JTextField jTextField1;

public javax.swing.JTextField jTextField2;

public javax.swing.JTextField maxCountSecEdit;

public javax.swing.JTextField nameCoachEdit1;

public javax.swing.JTextField nameParEdit;

public javax.swing.JTextField nameRoomEdit;

public javax.swing.JTextField nameSecEdit;

public javax.swing.JTextField numCoachEdit1;

public javax.swing.JTextField numParEdit;

private javax.swing.JButton okButton;

private javax.swing.JButton okCouchButton;

private javax.swing.JButton okParButton;

private javax.swing.JButton okRoomButton;

private javax.swing.JButton okSecButton;

public javax.swing.JComboBox<String> parentComboBox;

public javax.swing.JComboBox<String> roomComboBox;

public javax.swing.JComboBox<String> sectionComboBox;

public javax.swing.JTextField surCoachEdit1;

public javax.swing.JTextField surParEdit;

// End of variables declaration

}

# **Приложение 2**



