МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
Кафедра ИИТ

ЛАБОРАТОРНАЯ РАБОТА №10

По дисциплине: «Современные платформы программирования»

Выполнил:
Студент ФЭИС
3-го курса, группы ПО-5
Белко В. А.
Проверил:
Крощенко А. А.

Брест 2022

Вариант: 2

Цель работы: приобрести практические навыки разработки многооконных приложений на JavaFX для работы с базами данных

На основе БД, разработанной в лабораторной работе No9, реализовать многооконное приложение-клиент, позволяющее выполнять основные операции над таблицей в БД (добавление, удаление, модификацию данных).

Основные требования к приложению:
• Для отображения выбирать таблицу с внешними ключами;
• Осуществлять вывод основных данных в табличном представлении;
• При выводе краткого представления записи в таблице (т.е. если выводятся не все поля), по щелчку мышкой на запись осуществлять вывод всех полей в подготовленные компоненты на форме;
• Для всех полей, представленных внешними ключами, выводить их текстовое представление из связанных таблиц (например, таблица-справочник «Времена года» содержит два поля – идентификатор и название сезона, в связанной таблице «Месяц года» есть внешний ключ на таблицу «Времена года»; в этом случае при выводе таблицы «Месяц года» нужно выводить название сезона, а не его идентификатор);
• При выводе предусмотреть упорядочивание по столбцу;
• Реализовать простейший фильтр данных по одному-двум полям;
• При добавлении новых данных в таблицу использовать дополнительное окно для ввода;
• При модификации данных можно использовать ту же форму, что и для добавления, но с внесенными актуальными значениями полей;
• При добавлении/модификации выводить варианты значений полей с внешним ключом с помощью выпадающего списка;
• При удалении данных осуществлять удаление записи, на которой в данных момент находится фокус.

Код программы:

InfoDialog:

```
package FacultyCompany.Actions;

import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.layout.StackPane;
import javafx.scene.layout.VBox;
import javafx.stage.Modality;
import javafx.stage.Stage;

public class InfoDialog {
    public InfoDialog(Stage primaryStage, String groupname, String subjectname,
                      Integer semesterid, Integer weekday, String lessonTime, String
lecturerFullName){
        StackPane secondaryLayout = new StackPane();

        final VBox vbox = new VBox();

        vbox.setSpacing(5);
```

```
        Label groupLabel = new Label("Group name: " + groupname);
        Label subjectLabel = new Label("Subject name: " + subjectname);
        Label semestrLabel = new Label("Semester id: " + semesterid);
        Label weekLabel = new Label("Week day: " + weekday);
        Label lessonLabel = new Label("Lesson time: " + lessonTime);
        Label lecturerNameLabel = new Label("Lecturer name: " + lecturerFullName);

        vbox.getChildren().addAll(groupLabel, subjectLabel, semestrLabel, weekLabel,
lessonLabel, lecturerNameLabel);
        secondaryLayout.getChildren().addAll(vbox);

        Scene secondScene = new Scene(secondaryLayout, 250, 150);

        // New window (Stage)
        Stage newWindow = new Stage();
        newWindow.setTitle("Information");
        newWindow.setScene(secondScene);

        // Specifies the modality for new window.
        newWindow.initModality(Modality.WINDOW_MODAL);

        // Specifies the owner Window (parent) for new window
        newWindow.initOwner(primaryStage);

        // Set position of second window, related to primary window.
        newWindow.setX(primaryStage.getX() + 200);
        newWindow.setY(primaryStage.getY() + 100);

        newWindow.setHeight(200);
        newWindow.setWidth(300);

        newWindow.setResizable(false);

        newWindow.show();
    }
}
```

TimeTableAddDialog:

```
package FacultyCompany.Actions;

import FacultyCompany.Core.RepositoryManager;
import FacultyCompany.Entities.*;
import javafx.collections.FXCollections;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.StackPane;
import javafx.scene.layout.VBox;
import javafx.stage.Modality;
import javafx.stage.Stage;

import java.sql.SQLException;
import java.util.List;

public class TimeTableAddDialog {
    Label tittleLabel;
    Button addButton;

    Label weekDayLabel;
    TextField weekDayField;

    ComboBox<Subject> subjectComboBox;
    ComboBox<Group> groupComboBox;
    ComboBox<Teacher> teacherComboBox;
    ComboBox<Calendar> calendarComboBox;
```

```
public TimeTableAddDialog(Stage primaryStage) {
    initControls();

    List<Subject> subjects;
    List<Group> groups;
    List<Teacher> teachers;
    List<Calendar> calendars;

    try {
        RepositoryManager repositoryManager = new RepositoryManager();

        subjects = repositoryManager.subjectRepository.GetAll();
        groups = repositoryManager.groupRepository.GetAll();
        teachers = repositoryManager.teacherRepository.GetAll();
        calendars = repositoryManager.calendarRepository.GetAll();

        var subjectObservableList = FXCollections.observableArrayList(subjects);
        var groupObservableList = FXCollections.observableArrayList(groups);
        var teacherObservableList = FXCollections.observableArrayList(teachers);
        var calendarObservableList = FXCollections.observableArrayList(calendars);

        subjectComboBox = new ComboBox<>(subjectObservableList);
        groupComboBox = new ComboBox<>(groupObservableList);
        teacherComboBox = new ComboBox<>(teacherObservableList);
        calendarComboBox = new ComboBox<>(calendarObservableList);

        subjectComboBox.setValue(subjectObservableList.get(0));
        groupComboBox.setValue(groupObservableList.get(0));
        teacherComboBox.setValue(teacherObservableList.get(0));
        calendarComboBox.setValue(calendarObservableList.get(0));
    }
    catch (SQLException throwables) {
        throwables.printStackTrace();
    }

    StackPane secondaryLayout = new StackPane();
    final VBox vbox = new VBox();

    vbox.setSpacing(5);

    vbox.getChildren().addAll(tittleLabel, weekDayLabel, weekDayField,
            new Label("Enter subject"), subjectComboBox,
            new Label("Enter group"), groupComboBox,
            new Label("Enter teacher"), teacherComboBox,
            new Label("Enter lesson"), calendarComboBox, addButton);
    secondaryLayout.getChildren().addAll(vbox);

    Scene secondScene = new Scene(secondaryLayout, 300, 240);

    // New window (Stage)
    Stage newWindow = new Stage();
    newWindow.setTitle("Add new information");
    newWindow.setScene(secondScene);

    // Specifies the modality for new window.
    newWindow.initModality(Modality.WINDOW_MODAL);

    // Specifies the owner Window (parent) for new window
    newWindow.initOwner(primaryStage);

    // Set position of second window, related to primary window.
    newWindow.setX(primaryStage.getX() + 200);
    newWindow.setY(primaryStage.getY() + 200);

    newWindow.setHeight(400);
    newWindow.setWidth(250);
```

```
        newWindow.setResizable(false);

        newWindow.show();

        addButton.setOnAction(event -> {
            Integer weekDay = Integer.parseInt(weekDayField.getText());

            Integer subjectId =
subjectComboBox.getSelectionModel().getSelectedItem().getId();
            Integer groupId =
groupComboBox.getSelectionModel().getSelectedItem().getId();
            Integer teacherId =
teacherComboBox.getSelectionModel().getSelectedItem().getId();
            Integer calendarId =
calendarComboBox.getSelectionModel().getSelectedItem().getId();

            if (weekDay > 7 || weekDay <= 0)
                return;

            try {
                RepositoryManager repositoryManager = new RepositoryManager();
                repositoryManager.timeTableRepository.Add(new TimeTable(groupId,
subjectId, teacherId, weekDay, calendarId));
                newWindow.close();
            }
            catch (SQLException throwables) {
                throwables.printStackTrace();
            }
        });
    }

    public void initControls()
    {
        tittleLabel = new Label("Enter the information:");

        weekDayLabel = new Label("Week day");
        weekDayField = new TextField();

        addButton = new Button("Add the information");

        weekDayField.setMaxSize(50, 50);
    }
}
```

TimeTableUpdateDialog:

```
package FacultyCompany.Actions;

import FacultyCompany.Core.RepositoryManager;
import FacultyCompany.Entities.*;
import javafx.collections.FXCollections;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.StackPane;
import javafx.scene.layout.VBox;
import javafx.stage.Modality;
import javafx.stage.Stage;

import java.sql.SQLException;
import java.util.List;

public class TimeTableUpdateDialog {
    Label tittleLabel;
    Button addButton;
```

```
    Label weekDayLabel;
    TextField weekDayField;

    ComboBox<Subject> subjectComboBox;
    ComboBox<Group> groupComboBox;
    ComboBox<Teacher> teacherComboBox;
    ComboBox<Calendar> calendarComboBox;

    public TimeTableUpdateDialog(Stage primaryStage, TimeTable table) {
        initControls();

        List<Subject> subjects;
        List<Group> groups;
        List<Teacher> teachers;
        List<Calendar> calendars;

        try {
            RepositoryManager repositoryManager = new RepositoryManager();

            subjects = repositoryManager.subjectRepository.GetAll();
            groups = repositoryManager.groupRepository.GetAll();
            teachers = repositoryManager.teacherRepository.GetAll();
            calendars = repositoryManager.calendarRepository.GetAll();

            var subjectObservableList = FXCollections.observableArrayList(subjects);
            var groupObservableList = FXCollections.observableArrayList(groups);
            var teacherObservableList = FXCollections.observableArrayList(teachers);
            var calendarObservableList = FXCollections.observableArrayList(calendars);

            subjectComboBox = new ComboBox<>(subjectObservableList);
            groupComboBox = new ComboBox<>(groupObservableList);
            teacherComboBox = new ComboBox<>(teacherObservableList);
            calendarComboBox = new ComboBox<>(calendarObservableList);

            subjectComboBox.setValue(subjectObservableList.get(table.getSubjectid() -
1));
            groupComboBox.setValue(groupObservableList.get(table.getGroupid() - 1));
            teacherComboBox.setValue(teacherObservableList.get(table.getTeacherid()));
            calendarComboBox.setValue(calendarObservableList.get(table.getLessonid() -
1));

            String text = Integer.toString(table.getWeekday()) ;
            weekDayField.setText(text);
        }
        catch (SQLException throwables) {
            throwables.printStackTrace();
        }

        StackPane secondaryLayout = new StackPane();
        final VBox vbox = new VBox();

        vbox.setSpacing(5);

        vbox.getChildren().addAll(tittleLabel, weekDayLabel, weekDayField,
                new Label("Enter subject"), subjectComboBox,
                new Label("Enter group"), groupComboBox,
                new Label("Enter teacher"), teacherComboBox,
                new Label("Enter lesson"), calendarComboBox, addButton);
        secondaryLayout.getChildren().addAll(vbox);

        Scene secondScene = new Scene(secondaryLayout, 300, 240);

        // New window (Stage)
        Stage newWindow = new Stage();
        newWindow.setTitle("Update cell");
        newWindow.setScene(secondScene);
```

```
        // Specifies the modality for new window.
        newWindow.initModality(Modality.WINDOW_MODAL);

        // Specifies the owner Window (parent) for new window
        newWindow.initOwner(primaryStage);

        // Set position of second window, related to primary window.
        newWindow.setX(primaryStage.getX() + 200);
        newWindow.setY(primaryStage.getY() + 200);

        newWindow.setHeight(400);
        newWindow.setWidth(250);
        newWindow.setResizable(false);

        newWindow.show();

        addButton.setOnAction(event -> {
            Integer id = table.getId();

            Integer weekDay = Integer.parseInt(weekDayField.getText());
            Integer subjectId =
subjectComboBox.getSelectionModel().getSelectedItem().getId();
            Integer groupId =
groupComboBox.getSelectionModel().getSelectedItem().getId();
            Integer teacherId =
teacherComboBox.getSelectionModel().getSelectedItem().getId();
            Integer calendarId =
calendarComboBox.getSelectionModel().getSelectedItem().getId();

            if (weekDay > 7 || weekDay < 0)
                return;

            try {
                RepositoryManager  repositoryManager = new RepositoryManager();
                repositoryManager.timeTableRepository.Update(new TimeTable(id, groupId,
                        subjectId, teacherId, weekDay, calendarId));
                newWindow.close();
            }
            catch (SQLException throwables) {
                throwables.printStackTrace();
            }
        });
    }

    public void initControls()
    {
        tittleLabel = new Label("Enter the information:");

        weekDayLabel = new Label("Week day");
        weekDayField = new TextField();

        addButton = new Button("Update cell");

        weekDayField.setMaxSize(50, 50);
    }
}
```

Connection:

```
package FacultyCompany.Core;

import java.sql.DriverManager;
import java.sql.SQLException;

public final class Connection {

    public java.sql.Connection GetConnection() throws SQLException {
        final String URL = "jdbc:mysql://localhost:3306/lab2";
```

```
        final String USER = "root";
        final String PASSWORD = "user";

        var connection = DriverManager.getConnection(URL, USER, PASSWORD);
        return connection;
    }
}
```

RepositoryManager:

```
package FacultyCompany.Core;

import FacultyCompany.Core.Connection;
import FacultyCompany.Entities.*;
import FacultyCompany.Interfaces.IBaseRepository;
import FacultyCompany.Repositories.*;

import java.sql.SQLException;

public class RepositoryManager {
    public IBaseRepository<Subject> subjectRepository;
    public IBaseRepository<Group> groupRepository;
    public IBaseRepository<Teacher> teacherRepository;
    public IBaseRepository<Calendar> calendarRepository;
    public IBaseRepository<TimeTable> timeTableRepository;

    private static Connection connection = new Connection();

    public RepositoryManager() throws SQLException {
        this.subjectRepository = new SubjectRepository(connection.GetConnection());
        this.groupRepository = new GroupRepository(connection.GetConnection());
        this.teacherRepository = new TeacherRepository(connection.GetConnection());
        this.calendarRepository = new CalendarRepository(connection.GetConnection());
        this.timeTableRepository = new TimeTableRepository(connection.GetConnection());
    }
}
```

Calendar:

```
package FacultyCompany.Entities;

public class Calendar {
    private int id;
    private int semesterid;
    private int weekday;
    private int lessonid;
    private String lessontime;

    public Calendar() {}

    public Calendar(int semesterId, int weekDay, int lessonId, String lessonTime) {
        this.semesterid = semesterId;
        this.weekday = weekDay;
        this.lessonid = lessonId;
        this.lessontime = lessonTime;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public int getSemesterid() {
```

```
        return semesterid;
    }

    public void setSemesterid(int semesterId) {
        this.semesterid = semesterId;
    }

    public int getWeekday() {
        return weekday;
    }

    public void setWeekday(int weekDay) {
        this.weekday = weekDay;
    }

    public int getLessonid() {
        return lessonid;
    }

    public void setLessonid(int lessonId) {
        this.lessonid = lessonId;
    }

    public String getLessontime() {
        return lessontime;
    }

    public void setLessontime(String lessonTime) {
        this.lessontime = lessonTime;
    }

    @Override
    public String toString()
    {
        return getLessontime();
    }
}
```

Group:

```
package FacultyCompany.Entities;

public class Group {
    private int id;
    private String groupname;

    public Group() {}

    public Group(String groupName) {
        this.groupname = groupName;
    }

    public String getGroupname() {
        return groupname;
    }

    public void setGroupname(String groupName) {
        this.groupname = groupName;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }
```

```
    @Override
    public String toString()
    {
        return getGroupname();
    }

}
```

Subject:

```
package FacultyCompany.Entities;

public class Subject {
    private int id;
    private String subjectName;

    public Subject() {}

    public Subject(String subjectName) {
        this.subjectName = subjectName;
    }

    public String getSubjectName() {
        return subjectName;
    }

    public void setSubjectName(String subjectName) {
        this.subjectName = subjectName;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    @Override
    public String toString()
    {
        return getSubjectName();
    }

}
```

Teacher:

```
package FacultyCompany.Entities;

public class Teacher {
    private int id;
    private String firstname;
    private String lastname;
    private String patronymic;

    public Teacher() {}

    public Teacher(String firstName, String lastName, String patronymic) {
        this.firstname = firstName;
        this.lastname = lastName;
        this.patronymic = patronymic;
    }

    public int getId() {
        return id;
    }
```

```
    public void setId(int id) {
        this.id = id;
    }

    public String getFirstname() {
        return firstname;
    }

    public void setFirstname(String firstName) {
        this.firstname = firstName;
    }

    public String getLastname() {
        return lastname;
    }

    public void setLastname(String lastName) {
        this.lastname = lastName;
    }

    public String getPatronymic() {
        return patronymic;
    }

    public void setPatronymic(String patronymic) {
        this.patronymic = patronymic;
    }

    @Override
    public String toString()
    {
        return getFirstname() + " " + getLastname();
    }

}
```

TimeTable:

```
package FacultyCompany.Entities;

public class TimeTable {
    private int id;

    private int groupid;
    private Group group;

    private int subjectid;
    private Subject subject;

    private int teacherid;
    private Teacher teacher;

    private int weekday;

    private int lessonid;
    private Calendar calendar;

    public TimeTable() {}

    public TimeTable(int groupId, int subjectId, int teacherId, int weekDay, int lessonId)
{
        this.groupid = groupId;
        this.subjectid = subjectId;
        this.teacherid = teacherId;
        this.weekday = weekDay;
        this.lessonid = lessonId;
    }
```

```
    public TimeTable(int id) {
        this.id = id;
    }

    public TimeTable(int id, int groupId, int subjectId, int teacherId, int weekDay, int
lessonId) {
        this.id = id;
        this.groupid = groupId;
        this.subjectid = subjectId;
        this.teacherid = teacherId;
        this.weekday = weekDay;
        this.lessonid = lessonId;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public int getGroupid() {
        return groupid;
    }

    public void setGroupid(int groupId) {
        this.groupid = groupId;
    }

    public Group getGroup() {
        return group;
    }

    public void setGroup(Group group) {
        this.group = group;
    }

    public int getSubjectid() {
        return subjectid;
    }

    public void setSubjectid(int subjectId) {
        this.subjectid = subjectId;
    }

    public Subject getSubject() {
        return subject;
    }

    public void setSubject(Subject subject) {
        this.subject = subject;
    }

    public int getTeacherid() {
        return teacherid;
    }

    public void setTeacherid(int teacherId) {
        this.teacherid = teacherId;
    }

    public String getTeacherName() {
        return teacher.getFirstname() + " " + teacher.getLastname();
    }

    public Teacher getTeacher() {
```

```
        return teacher;
    }

    public void setTeacher(Teacher teacher) {
        this.teacher = teacher;
    }

    public int getWeekday() {
        return weekday;
    }

    public void setWeekday(int weekDay) {
        this.weekday = weekDay;
    }

    public int getLessonid() {
        return lessonid;
    }

    public void setLessonid(int lessonId) {
        this.lessonid = lessonId;
    }

    public Calendar getCalendar() {
        return calendar;
    }

    public void setCalendar(Calendar calendar) {
        this.calendar = calendar;
    }

}
```

IBaseRepository:

```
package FacultyCompany.Interfaces;

import java.sql.SQLException;
import java.util.ArrayList;

public interface IBaseRepository<T> {
    T Add(T entity) throws SQLException;
    void Update(T entity) throws SQLException;
    void Delete(T entity) throws SQLException;
    T GetByIdOrNull(int id) throws SQLException;
    ArrayList<T> GetAll() throws SQLException;
}
```

CalendarRepository:

```
package FacultyCompany.Repositories;

import FacultyCompany.Entities.Calendar;
import FacultyCompany.Interfaces.IBaseRepository;

import java.sql.Connection;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;

public class CalendarRepository implements IBaseRepository<Calendar> {

    private final Connection connection;

    public CalendarRepository(Connection connection) {
        this.connection = connection;
    }
```

```
    @Override
    public Calendar Add(Calendar entity) throws SQLException {
        var query =
                "INSERT INTO lab2.calendar( " +
                        " semesterid, weekday, lessonid, lessontime) " +
                        " VALUES (?, ?, ?, ?)";

        var statement = connection.prepareStatement(query,
Statement.RETURN_GENERATED_KEYS);
        statement.setInt(1, entity.getSemesterid());
        statement.setInt(2, entity.getWeekday());
        statement.setInt(3, entity.getLessonid());
        statement.setString(4, entity.getLessontime());

        statement.execute();

        var generatedKeys = statement.getGeneratedKeys();
        generatedKeys.next();
        entity.setId(generatedKeys.getInt(1));

        return entity;
    }

    @Override
    public void Update(Calendar entity) throws SQLException {
        var query ="UPDATE lab2.calendar " +
                " SET semesterid=?, weekday=?, lessonid=?, lessontime=?" +
                " WHERE id=?";

        var statement = connection.prepareStatement(query);
        statement.setInt(1, entity.getSemesterid());
        statement.setInt(2, entity.getWeekday());
        statement.setInt(3, entity.getLessonid());
        statement.setString(4, entity.getLessontime());
        statement.setInt(5, entity.getId());

        statement.executeUpdate();
    }

    @Override
    public void Delete(Calendar entity) throws SQLException {
        var query = "DELETE FROM lab2.calendar" +
                " WHERE id=?";
        var statement = connection.prepareStatement(query);
        statement.setInt(1, entity.getId());

        statement.executeUpdate();
    }

    @Override
    public Calendar GetByIdOrNull(int id) throws SQLException {
        var query =
                "SELECT * FROM lab2.calendar" +
                        " WHERE Id = ?";

        var statement = connection.prepareStatement(query);
        statement.setInt(1, id);

        var reader = statement.executeQuery();
        if(reader.next())
        {
            var result = new Calendar();
            result.setId(reader.getInt("id"));
            result.setLessonid(reader.getInt("lessonid"));
            result.setSemesterid(reader.getInt("semesterid"));
            result.setWeekday(reader.getInt("weekday"));
            result.setLessontime(reader.getString("lessontime"));
```

```
            return result;
        }

        return null;
    }

    @Override
    public ArrayList<Calendar> GetAll() throws SQLException {
        var query =
                "SELECT * FROM lab2.calendar Order by id";

        var statement = connection.prepareStatement(query);

        var reader = statement.executeQuery();
        var result = new ArrayList<Calendar>();
        while (reader.next())
        {
            var calendar = new Calendar();
            calendar.setId(reader.getInt("id"));
            calendar.setLessonid(reader.getInt("lessonid"));
            calendar.setSemesterid(reader.getInt("semesterid"));
            calendar.setWeekday(reader.getInt("weekday"));
            calendar.setLessontime(reader.getString("lessontime"));

            result.add(calendar);
        }

        return result;
    }
}
```

GroupRepository:

```
package FacultyCompany.Repositories;

import FacultyCompany.Entities.Group;
import FacultyCompany.Interfaces.IBaseRepository;

import java.sql.Connection;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;

public class GroupRepository implements IBaseRepository<Group> {
    private final Connection connection;

    public GroupRepository(Connection connection) {
        this.connection = connection;
    }

    @Override
    public Group Add(Group entity) throws SQLException {
        var query =
                "INSERT INTO lab2.groups(" +
                        " groupname)" +
                        " VALUES (?)";

        var statement = connection.prepareStatement(query,
Statement.RETURN_GENERATED_KEYS);
        statement.setString(1, entity.getGroupname());

        statement.execute();

        var generatedKeys = statement.getGeneratedKeys();
        generatedKeys.next();
        entity.setId(generatedKeys.getInt(1));

        return entity;
```

```
    }

    @Override
    public void Update(Group entity) throws SQLException {
        var query =
                "UPDATE lab2.groupss" +
                        " SET groupname = ?" +
                        " WHERE id = ?";

        var statement = connection.prepareStatement(query);
        statement.setString(1, entity.getGroupname());
        statement.setInt(2, entity.getId());

        statement.executeUpdate();
    }

    @Override
    public void Delete(Group entity) throws SQLException {
        var query = "DELETE FROM lab2.groupss" +
                " WHERE id=?";
        var statement = connection.prepareStatement(query);
        statement.setInt(1, entity.getId());

        statement.executeUpdate();
    }

    @Override
    public Group GetByIdOrNull(int id) throws SQLException {
        var query =
                "SELECT * FROM lab2.groupss" +
                        " WHERE Id = ?";

        var statement = connection.prepareStatement(query);
        statement.setInt(1, id);

        var reader = statement.executeQuery();
        if(reader.next())
        {
            var result = new Group();
            result.setId(reader.getInt("id"));
            result.setGroupname(reader.getString("groupname"));
            return result;
        }

        return null;
    }

    @Override
    public ArrayList<Group> GetAll() throws SQLException {
        var query =
                "SELECT * FROM lab2.groupss Order by id";

        var statement = connection.prepareStatement(query);

        var reader = statement.executeQuery();
        var result = new ArrayList<Group>();
        while (reader.next())
        {
            var group = new Group();
            group.setId(reader.getInt("id"));
            group.setGroupname(reader.getString("groupname"));

            result.add(group);
        }

        return result;
```

```
    }
}
```

SubjectRepository:

```
package FacultyCompany.Repositories;

import FacultyCompany.Entities.Subject;
import FacultyCompany.Interfaces.IBaseRepository;

import java.sql.Connection;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;

public class SubjectRepository implements IBaseRepository<Subject> {

    private final Connection connection;

    public SubjectRepository(Connection connection) {
        this.connection = connection;
    }

    @Override
    public Subject Add(Subject entity) throws SQLException {
        var query =
                "INSERT INTO lab2.subjects(" +
                        " subject_name)" +
                        " VALUES (?)";

        var statement = connection.prepareStatement(query,
Statement.RETURN_GENERATED_KEYS);
        statement.setString(1, entity.getSubjectName());


        statement.execute();

        var generatedKeys = statement.getGeneratedKeys();
        generatedKeys.next();
        entity.setId(generatedKeys.getInt(1));

        return entity;
    }

    @Override
    public void Update(Subject entity) throws SQLException {
        var query =
                "UPDATE lab2.subjects" +
                        " SET subject_name = ?" +
                        " WHERE id = ?";

        var statement = connection.prepareStatement(query);
        statement.setString(1, entity.getSubjectName());
        statement.setInt(2, entity.getId());

        statement.executeUpdate();
    }

    @Override
    public void Delete(Subject entity) throws SQLException {
        var query = "DELETE FROM lab2.subjects" +
                " WHERE id=?";
        var statement = connection.prepareStatement(query);
        statement.setInt(1, entity.getId());

        statement.executeUpdate();
    }
```

```
    @Override
    public Subject GetByIdOrNull(int id) throws SQLException {
        var query =
                "SELECT * FROM lab2.subjects" +
                        " WHERE Id = ? " +
                        "Order by id";

        var statement = connection.prepareStatement(query);
        statement.setInt(1, id);

        var reader = statement.executeQuery();
        if(reader.next())
        {
            var result = new Subject();
            result.setId(reader.getInt("id"));
            result.setSubjectName(reader.getString("subject_name"));
            return result;
        }

        return null;
    }

    @Override
    public ArrayList<Subject> GetAll() throws SQLException {
        var query =
                "SELECT * FROM lab2.subjects Order by id";

        var statement = connection.prepareStatement(query);

        var reader = statement.executeQuery();
        var result = new ArrayList<Subject>();
        while (reader.next())
        {
            var subject = new Subject();
            subject.setId(reader.getInt("id"));
            subject.setSubjectName(reader.getString("subject_name"));

            result.add(subject);
        }

        return result;
    }
}
```

TeacherRepository:

```
package FacultyCompany.Repositories;

import FacultyCompany.Entities.Teacher;
import FacultyCompany.Interfaces.IBaseRepository;

import java.sql.Connection;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;

public class TeacherRepository implements IBaseRepository<Teacher> {

    private final Connection connection;

    public TeacherRepository(Connection connection) {
        this.connection = connection;
    }

    @Override
    public Teacher Add(Teacher entity) throws SQLException {
        var query =
```

```
                "INSERT INTO lab2.teachers(" +
                        "firstname, lastname, patronymic)" +
                        " VALUES (?, ?, ?)";

        var statement = connection.prepareStatement(query,
Statement.RETURN_GENERATED_KEYS);
        statement.setString(1, entity.getFirstname());
        statement.setString(2, entity.getLastname());
        statement.setString(3, entity.getPatronymic());

        statement.execute();

        var generatedKeys = statement.getGeneratedKeys();
        generatedKeys.next();
        entity.setId(generatedKeys.getInt(1));

        return entity;
    }

    @Override
    public void Update(Teacher entity) throws SQLException {
        var query =
                "UPDATE lab2.teachers" +
                        " SET firstname=?, lastname=?, patronymic=?" +
                        " WHERE id=?";


        var statement = connection.prepareStatement(query);
        statement.setString(1, entity.getFirstname());
        statement.setString(2, entity.getLastname());
        statement.setString(3, entity.getPatronymic());
        statement.setInt(4, entity.getId());

        statement.executeUpdate();
    }

    @Override
    public void Delete(Teacher entity) throws SQLException {
        var query = "DELETE FROM lab2.teachers" +
                " WHERE id=?";
        var statement = connection.prepareStatement(query);
        statement.setInt(1, entity.getId());

        statement.executeUpdate();
    }

    @Override
    public Teacher GetByIdOrNull(int id) throws SQLException {
        var query =
                "SELECT * FROM lab2.teachers" +
                        " WHERE Id = ?";

        var statement = connection.prepareStatement(query);
        statement.setInt(1, id);

        var reader = statement.executeQuery();
        if(reader.next())
        {
            var result = new Teacher();
            result.setId(reader.getInt("id"));
            result.setFirstname(reader.getString("firstname"));
            result.setLastname(reader.getString("lastname"));
            result.setPatronymic(reader.getString("patronymic"));
            return result;
        }

        return null;
```

```
    }

    @Override
    public ArrayList<Teacher> GetAll() throws SQLException {
        var query =
                "SELECT * FROM lab2.teachers Order by id";

        var statement = connection.prepareStatement(query);

        var reader = statement.executeQuery();
        var result = new ArrayList<Teacher>();
        while (reader.next())
        {
            var teacher = new Teacher();
            teacher.setId(reader.getInt("id"));
            teacher.setFirstname(reader.getString("firstname"));
            teacher.setLastname(reader.getString("lastname"));
            teacher.setPatronymic(reader.getString("patronymic"));

            result.add(teacher);
        }

        return result;
    }
}
```

TimeTableRepository:

```
package FacultyCompany.Repositories;

import FacultyCompany.Entities.TimeTable;
import FacultyCompany.Interfaces.IBaseRepository;

import java.sql.Connection;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;

public class TimeTableRepository implements IBaseRepository<TimeTable> {

    private final Connection connection;

    public TimeTableRepository(Connection connection) {
        this.connection = connection;
    }

    @Override
    public TimeTable Add(TimeTable entity) throws SQLException {
        var query =
                "INSERT INTO lab2.timetable(" +
                        "groupid, subjectid, teacherid, weekday, lessonid)" +
                        " VALUES (?, ?, ?, ?, ?);";

        var statement = connection.prepareStatement(query,
Statement.RETURN_GENERATED_KEYS);
        statement.setInt(1, entity.getGroupid());
        statement.setInt(2, entity.getSubjectid());
        statement.setInt(3, entity.getTeacherid());
        statement.setInt(4, entity.getWeekday());
        statement.setInt(5, entity.getLessonid());

        statement.execute();

        var generatedKeys = statement.getGeneratedKeys();
        generatedKeys.next();
        entity.setId(generatedKeys.getInt(1));

        return entity;
```

```
    }

    @Override
    public void Update(TimeTable entity) throws SQLException {

        var query = "UPDATE lab2.timetable " +
                " SET groupid=?, subjectid=?, teacherid=?, weekday=?, lessonid=? " +
                " WHERE id=?";

        var statement = connection.prepareStatement(query);
        statement.setInt(1, entity.getGroupid());
        statement.setInt(2, entity.getSubjectid());
        statement.setInt(3, entity.getTeacherid());
        statement.setInt(4, entity.getWeekday());
        statement.setInt(5, entity.getLessonid());
        statement.setInt(6, entity.getId());

        statement.executeUpdate();
        return;
    }

    @Override
    public void Delete(TimeTable entity) throws SQLException {
        var query = "DELETE FROM lab2.timetable" +
                " WHERE id=?";

        var statement = connection.prepareStatement(query);
        statement.setInt(1, entity.getId());

        statement.executeUpdate();
    }

    @Override
    public TimeTable GetByIdOrNull(int id) throws SQLException {
        var query =
                "SELECT * FROM lab2.timetable" +
                        " WHERE Id = ?";

        var statement = connection.prepareStatement(query);
        statement.setInt(1, id);

        var reader = statement.executeQuery();
        if(reader.next())
        {
            var result = new TimeTable();
            result.setId(reader.getInt("id"));
            result.setLessonid(reader.getInt("lessonid"));
            result.setWeekday(reader.getInt("weekday"));
            result.setGroupid(reader.getInt("groupid"));
            result.setSubjectid(reader.getInt("subjectid"));
            return result;
        }

        return null;
    }

    @Override
    public ArrayList<TimeTable> GetAll() throws SQLException {
        var query =
                "SELECT * FROM lab2.timetable Order by id";

        var statement = connection.prepareStatement(query);

        var reader = statement.executeQuery();
        var result = new ArrayList<TimeTable>();
        while (reader.next())
        {
```

```
            var timeTable = new TimeTable();
            timeTable.setId(reader.getInt("id"));
            timeTable.setGroupid(reader.getInt("groupid"));
            timeTable.setSubjectid(reader.getInt("subjectid"));
            timeTable.setTeacherid(reader.getInt("teacherid"));
            timeTable.setWeekday(reader.getInt("weekday"));
            timeTable.setLessonid(reader.getInt("lessonid"));

            result.add(timeTable);
        }

        return result;
    }
}
```

TimeTableViewModel:

```
package FacultyCompany.ViewModels;

import FacultyCompany.Entities.TimeTable;

public class TimeTableViewModel {
    private int id;

    private String groupname;

    private String subjectName;

    private String teacherFullName;

    private int semesterid;
    private int weekday;
    private String lessontime;

    private int groupid;
    private int subjectid;
    private int teacherid;
    private int lessonid;

    public TimeTableViewModel(TimeTable table) {
        this.id = table.getId();
        this.groupname = table.getGroup().getGroupname();
        this.subjectName = table.getSubject().getSubjectName();
        this.semesterid = table.getCalendar().getSemesterid();
        this.weekday = table.getWeekday();
        this.lessontime = table.getCalendar().getLessontime();
        this.groupid = table.getGroupid();
        this.subjectid = table.getSubjectid();
        this.lessonid = table.getLessonid();
        this.teacherFullName = table.getTeacherName();
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getGroupname() {
        return groupname;
    }

    public void setGroupname(String groupname) {
        this.groupname = groupname;
    }
```

```
    public String getSubjectName() {
        return subjectName;
    }

    public void setSubjectName(String subjectName) {
        this.subjectName = subjectName;
    }

    public int getSemesterid() {
        return semesterid;
    }

    public void setSemesterid(int semesterid) {
        this.semesterid = semesterid;
    }

    public int getWeekday() {
        return weekday;
    }

    public void setWeekday(int weekday) {
        this.weekday = weekday;
    }

    public int getLessonid() {
        return lessonid;
    }

    public void setLessonid(int lessonid) {
        this.lessonid = lessonid;
    }

    public String getLessontime() {
        return lessontime;
    }

    public void setLessontime(String lessontime) {
        this.lessontime = lessontime;
    }

    public String getTeacherFullName() {
        return teacherFullName;
    }
}
```

App:

```
package FacultyCompany;

import FacultyCompany.Actions.InfoDialog;
import FacultyCompany.Actions.TimeTableAddDialog;
import FacultyCompany.Actions.TimeTableUpdateDialog;
import FacultyCompany.Core.RepositoryManager;
import FacultyCompany.Entities.*;
import FacultyCompany.ViewModels.TimeTableViewModel;
import javafx.application.Application;
import javafx.collections.FXCollections;
import javafx.fxml.FXMLLoader;
import javafx.geometry.Insets;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.VBox;
```

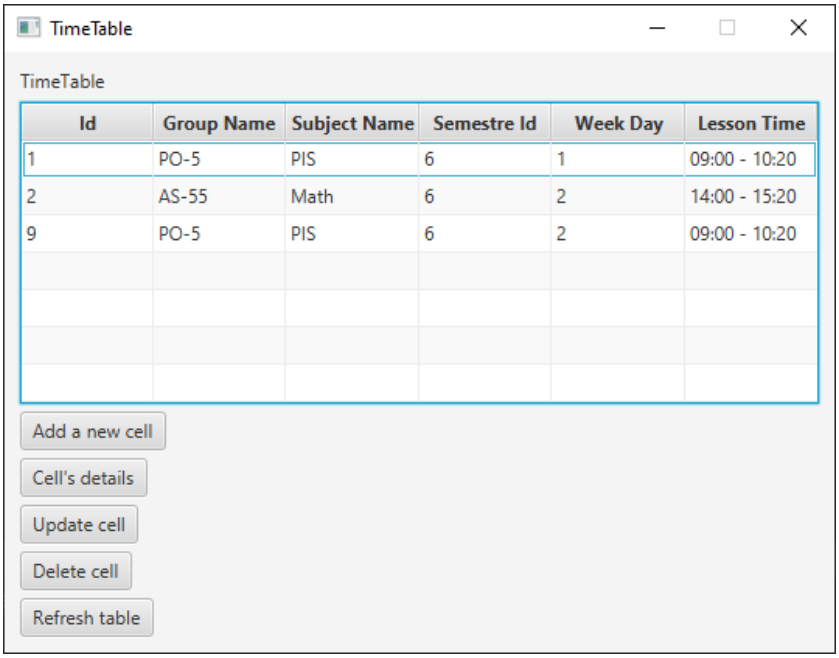
```
import javafx.stage.Stage;

import java.sql.SQLException;

import java.util.ArrayList;

public class App extends Application {
    private static RepositoryManager repositoryManager;

    public App() throws SQLException {
        repositoryManager = new RepositoryManager();
    }

    TableView<TimeTableViewModel> tableTimeTables;

    @Override
    public void start(Stage primaryStage) throws Exception {
        var timeTables= repositoryManager.timeTableRepository.GetAll();

        var timeTablesWithData = convertWithData(timeTables);

        ArrayList<TimeTableViewModel> VmTimeTable = new ArrayList<>();

        timeTablesWithData.forEach(t -> {
            VmTimeTable.add(new TimeTableViewModel(t));
        });

        var observableTimeTables = FXCollections.observableArrayList(VmTimeTable);

        tableTimeTables = new TableView<>(observableTimeTables);

        tableTimeTables.setColumnResizePolicy(TableView.CONSTRAINED_RESIZE_POLICY);
        tableTimeTables.setPrefSize(720,200);

        TableColumn<TimeTableViewModel, Integer> idColumn = new TableColumn<>("Id");
        idColumn.setCellValueFactory(new PropertyValueFactory<>("id"));
        tableTimeTables.getColumns().add(idColumn);

        TableColumn<TimeTableViewModel, String> groupnameColumn = new
TableColumn<>("Group Name");
        groupnameColumn.setCellValueFactory(new PropertyValueFactory<>("groupname"));
        tableTimeTables.getColumns().add(groupnameColumn);

        TableColumn<TimeTableViewModel, String> subjectNameColumn = new
TableColumn<>("Subject Name");
        subjectNameColumn.setCellValueFactory(new
PropertyValueFactory<>("subjectName"));
        tableTimeTables.getColumns().add(subjectNameColumn);

        TableColumn<TimeTableViewModel, Integer> semesteridColumn = new
TableColumn<>("Semestre Id");
        semesteridColumn.setCellValueFactory(new PropertyValueFactory<>("semesterid"));
        tableTimeTables.getColumns().add(semesteridColumn);

        TableColumn<TimeTableViewModel, Integer> weekdayColumn = new TableColumn<>("Week
Day");
        weekdayColumn.setCellValueFactory(new PropertyValueFactory<>("weekday"));
        tableTimeTables.getColumns().add(weekdayColumn);

        TableColumn<TimeTableViewModel, String> lessontimeColumn = new
TableColumn<>("Lesson Time");
        lessontimeColumn.setCellValueFactory(new PropertyValueFactory<>("lessontime"));
        tableTimeTables.getColumns().add(lessontimeColumn);

        executeRefresh();

        final VBox vbox = new VBox();
```

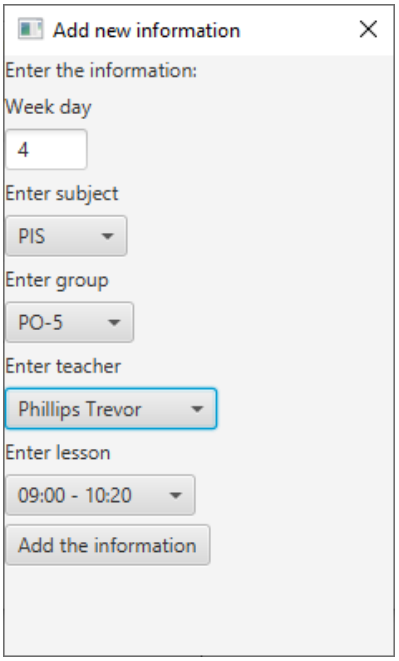
```
        vbox.setSpacing(5);
        vbox.setPadding(new Insets(10, 10, 10, 10));
        vbox.getChildren().addAll(new Label("TimeTable"), tableTimeTables);

        Button addCellButton = new Button();
        addCellButton.setText("Add a new cell");

        addCellButton.setOnAction(event -> {
            new TimeTableAddDialog(primaryStage);
        });

        vbox.getChildren().addAll(addCellButton);

        Button infoButton = new Button();
        infoButton.setText("Cell's details");

        infoButton.setOnAction(event -> {
            TimeTableViewModel infoTimeTable =
tableTimeTables.getSelectionModel().getSelectedItems().get(0);
            new InfoDialog(primaryStage, infoTimeTable.getGroupname(),
infoTimeTable.getSubjectName(), infoTimeTable.getSemesterid(),
                    infoTimeTable.getWeekday(), infoTimeTable.getLessontime(),
infoTimeTable.getTeacherFullName());
        });

        vbox.getChildren().addAll(infoButton);

        Button updateButton = new Button();
        updateButton.setText("Update cell");

        updateButton.setOnAction(event -> {
            TimeTableViewModel updateTimeTable =
tableTimeTables.getSelectionModel().getSelectedItems().get(0);
            TimeTable vmtable = new TimeTable();

            try {
                vmtable =
repositoryManager.timeTableRepository.GetByIdOrNull(updateTimeTable.getId());
            }
            catch (SQLException throwables) {
                throwables.printStackTrace();
            }

            new TimeTableUpdateDialog(primaryStage, vmtable);
        });

        vbox.getChildren().addAll(updateButton);

        Button deleteCellButton = new Button();
        deleteCellButton.setText("Delete cell");

        deleteCellButton.setOnAction(event -> {
            try {
                executeDelete();
                executeRefresh();
            }
            catch (SQLException throwables) {
                throwables.printStackTrace();
            }
        });

        vbox.getChildren().addAll(deleteCellButton);

        Button refreshCellButton = new Button();
        refreshCellButton.setText("Refresh table");

        refreshCellButton.setOnAction(event -> {
```

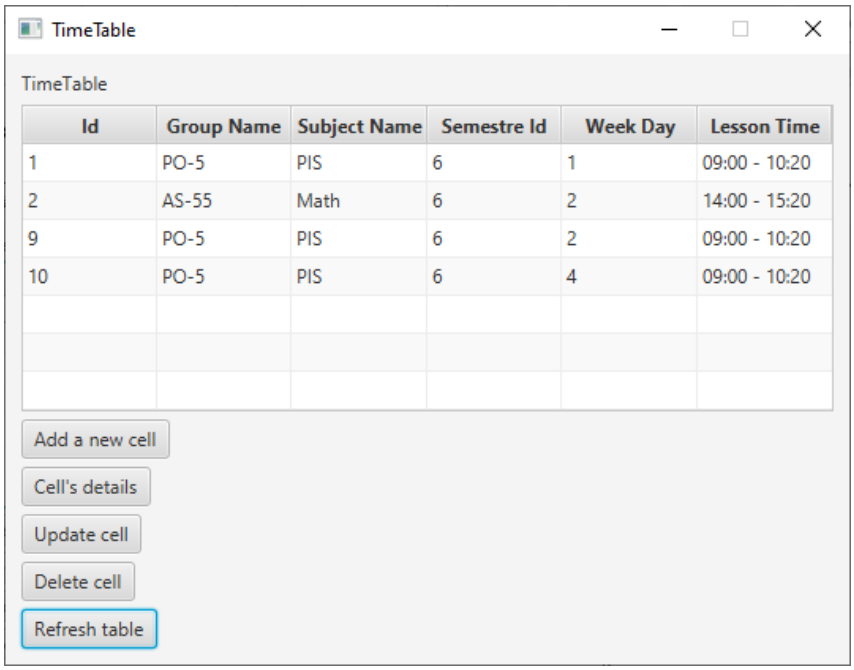
```
            try {
                executeRefresh();
            }
            catch (SQLException throwables) {
                throwables.printStackTrace();
            }
        });
        vbox.getChildren().addAll(refreshCellButton);
        Parent root = FXMLLoader.load(getClass().getResource("/sample.fxml"));
        primaryStage.setTitle("TimeTable");

        Scene scene = new Scene(root, 500, 500);
        ((GridPane) scene.getRoot()).getChildren().addAll(vbox);
        primaryStage.setScene(scene);

        primaryStage.setHeight(425);
        primaryStage.setWidth(550);

        primaryStage.setResizable(false);

        primaryStage.show();
    }

    public static void main(String[] args) {
        launch(args);
    }

    private void executeRefresh() throws SQLException {
        tableTimeTables.getItems().clear();

        var timeTables= repositoryManager.timeTableRepository.GetAll();

        var timeTablesWithData = convertWithData(timeTables);
        ArrayList<TimeTableViewModel> VmTimeTable = new ArrayList<>();

        timeTablesWithData.forEach(t -> {
            VmTimeTable.add(new TimeTableViewModel(t));
        });

        var observableTimeTables = FXCollections.observableArrayList(VmTimeTable);

        tableTimeTables.getItems().addAll(observableTimeTables);
    }

    private void executeDelete() throws SQLException {
        TimeTableViewModel deletedTimeTable =
tableTimeTables.getSelectionModel().getSelectedItems().get(0);
        repositoryManager.timeTableRepository.Delete(new
TimeTable(deletedTimeTable.getId()));
    }

    public static ArrayList<TimeTable> convertWithData(ArrayList<TimeTable> table) {
        table.forEach(t -> {
            try {

t.setGroup(repositoryManager.groupRepository.GetByIdOrNull(t.getGroupid()));

t.setSubject(repositoryManager.subjectRepository.GetByIdOrNull(t.getSubjectid()));

t.setTeacher(repositoryManager.teacherRepository.GetByIdOrNull(t.getTeacherid()));

t.setCalendar(repositoryManager.calendarRepository.GetByIdOrNull(t.getLessonid()));
            }
            catch (SQLException throwables) {
                throwables.printStackTrace();
            }
        });
```

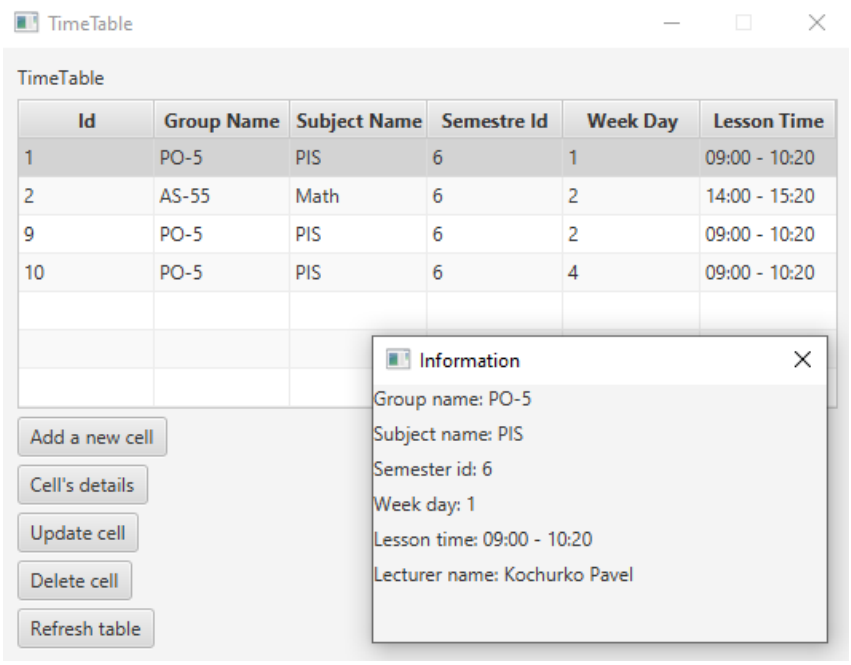
```
        return table;
    }
}
```

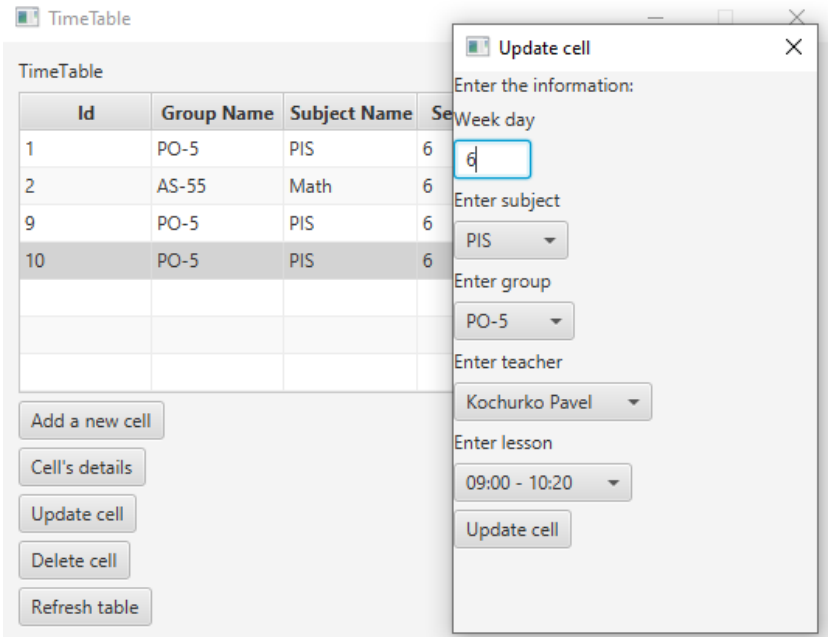
Результат выполнения программы:

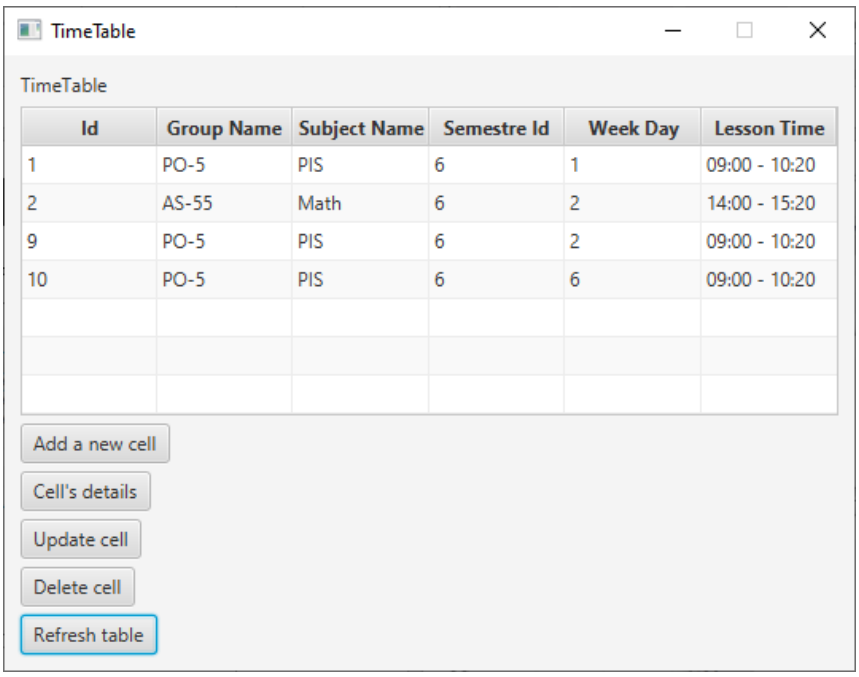


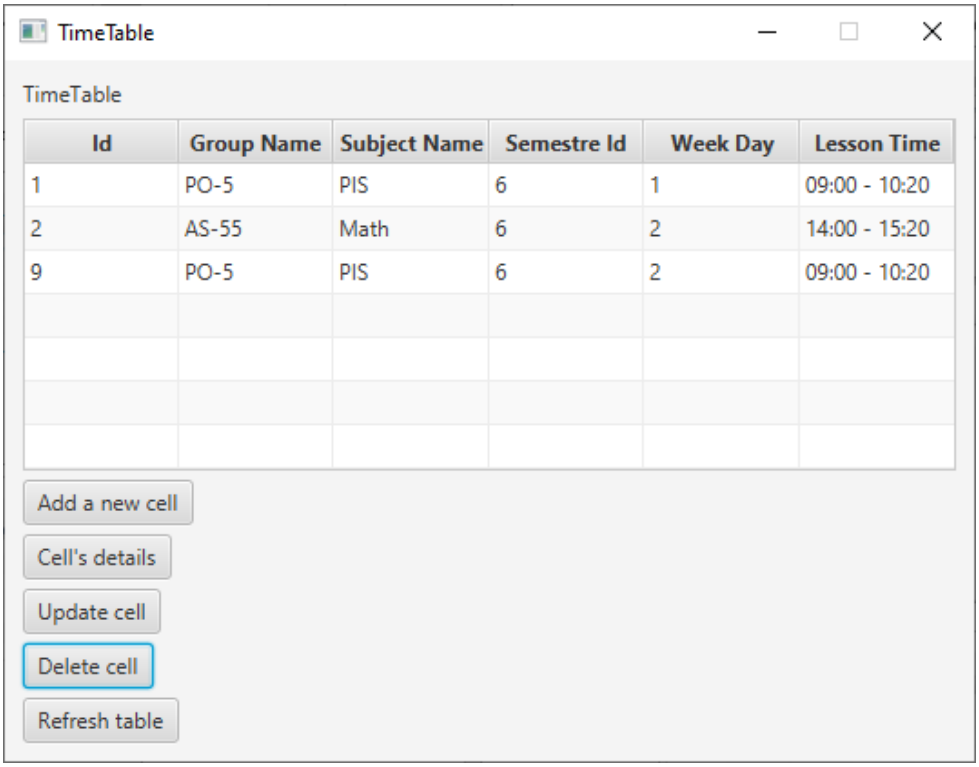





TimeTable
TimeTable
Id
Group Name
Subject Name
Semestre Id
Week Day
Lesson Time
1 PO-5 PIS 6 1 09:00 - 10:20
2 AS-55 Math 6 2 14:00 - 15:20
9 PO-5 PIS 6 2 09:00 - 10:20
10 PO-5 PIS 6 4 09:00 - 10:20
Add a new cell
Cell's details
Update cell
Delete cell
Refresh table
Information
Group name: PO-5
Subject name: PIS
Semester id: 6
Week day: 1
Lesson time: 09:00 - 10:20
Lecturer name: Kochurko Pavel


TimeTable
TimeTable
Id
Group Name
Subject Name
1 PO-5 PIS 6
2 AS-55 Math 6
9 PO-5 PIS 6
10 PO-5 PIS 6
Add a new cell
Cell's details
Update cell
Delete cell
Refresh table
Update cell
Enter the information:
Week day
6
Enter subject
PIS
Enter group
PO-5
Enter teacher
Kochurko Pavel
Enter lesson
09:00 - 10:20
Update cell


TimeTable
TimeTable
Id
Group Name
Subject Name
Semestre Id
Week Day
Lesson Time
1 PO-5 PIS 6 1 09:00 - 10:20
2 AS-55 Math 6 2 14:00 - 15:20
9 PO-5 PIS 6 2 09:00 - 10:20
10 PO-5 PIS 6 6 09:00 - 10:20
Add a new cell
Cell's details
Update cell
Delete cell
Refresh table



Вывод: приобрёл практические навыки разработки многооконных приложений на JavaFX для работы с базами данных