

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧЕРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
КАФЕДРА ИИТ

Лабораторная работа №4
по дисциплине «Современные платформы программирования»

Выполнила:
Андросюк М.М.
Группа: ПО-5
Проверил:
Крощенко А.А.

Брест 2022

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

Задание: На основе БД, разработанной в лабораторной работе № 3, реализовать многооконное приложение-клиент, позволяющее выполнять основные операции над таблицей в БД (добавление, удаление, модификацию данных).

Текст программы:

```
public final class CreateController implements

    Initializable { @FXML
    private TextField groupNameTextField;
    @FXML
    private TextField startYearTextField;
    @FXML
    private TextField endYearTextField;
    @FXML
    private ChoiceBox<Faculty> facultyChoiceBox;
    @FXML
    private ChoiceBox<Student> headmanChoiceBox;

    private DB db = null;

    @Override
    public void initialize(URL arg0, ResourceBundle
        arg1) { try {
        this.db = new DB();

        ObservableList<Faculty> faculties = FXCollections.observableArrayList();
        ObservableList<Student> students = FXCollections.observableArrayList();

        ResultSet facultiesSet = this.db.getAll(DB.FACULTIES_TABLE);
        ResultSet studentsSet = this.db.getAll(DB.STUDENTS_TABLE);

        while
            (Objects.requireNonNull(facultiesSet).next()) {
            faculties.add(new
                Faculty(facultiesSet.getInt("id"),
facultiesSet.getString("faculty_name")));
            }

            while (Objects.requireNonNull(studentsSet).next()) {
            students.add(new Student(studentsSet.getInt("id"),
                studentsSet.getString("first_name"), studentsSet.getString("last_name")));
            }

            this.facultyChoiceBox.setItems(faculties);
            this.headmanChoiceBox.setItems(students);
        } catch (final Exception exception) {
            Logger.getLogger(CreateController.class.getName()).log(Level.SEVERE, null,
                exception);
        }
    }
}
```

```

    }

    @FXML
    private void create() {
        if (this.groupNameTextField.getText().isEmpty() ||
            this.facultyChoiceBox.getSelectionModel().isEmpty()
            || this.headmanChoiceBox.getSelectionModel().isEmpty()) {
            return;
        }

        try {
            this.db.createGroup(new Group(null,
            this.headmanChoiceBox.getSelectionModel().getSelectedItem().getId(),
                this.facultyChoiceBox.getSelectionModel().getSelectedItem().getId(),
            this.groupNameTextField.getText(),
                Short.parseShort(startYearTextField.getText()),
            Short.parseShort(endYearTextField.getText())));
        } catch (NumberFormatException ignore) { }
    }
}

```

```

public final class UpdateController implements Initializable {

```

```

    @FXML
    private TextField groupNameTextField;
    @FXML
    private TextField startYearTextField;
    @FXML
    private TextField endYearTextField;
    @FXML
    private ChoiceBox<Faculty> facultyChoiceBox;
    @FXML
    private ChoiceBox<Student> headmanChoiceBox;

    private DB db = null;
    private Integer updatingId = null;
    private ObservableList<Faculty> faculties =
    null; private ObservableList<Student> students
    = null;

    @Override
    public void initialize(URL arg0, ResourceBundle
    arg1) { try {
        this.db = new DB();

        faculties = FXCollections.observableArrayList();
        students = FXCollections.observableArrayList();

        ResultSet facultiesSet = this.db.getAll(DB.FACULTIES_TABLE);
        ResultSet studentsSet = this.db.getAll(DB.STUDENTS_TABLE);

        while (Objects.requireNonNull(facultiesSet).next()) {

```

```

        faculties.add(new
Faculty(facultiesSet.getInt("id"),
facultiesSet.getString("faculty_name")));
    }

    while (Objects.requireNonNull(studentsSet).next()) {
        students.add(new Student(studentsSet.getInt("id"),
            studentsSet.getString("first_name"), studentsSet.getString("last_name")));
    }

    this.facultyChoiceBox.setItems(faculties);
    this.headmanChoiceBox.setItems(students);
} catch (final Exception exception) {
    Logger.getLogger(CreateController.class.getName()).log(Level.SEVERE, null,
exception);
}
}

public void setUpdatingId(final Integer
updatingId) { try {
    this.updatingId = updatingId;

    ResultSet group = this.db.getGroups(null, null, this.updatingId);
    Objects.requireNonNull(group).next();

    this.groupNameTextField.setText(group.getString("name"));
    this.startYearTextField.setText(String.valueOf(group.getShort("start_year")));
    this.endYearTextField.setText(String.valueOf(group.getShort("end_year")));

    int facultyId = group.getInt("faculty_id");
    int headmanId = group.getInt("headman_id");

    Faculty groupFaculty = this.faculties.stream().filter(faculty ->
faculty.getId() == facultyId).findFirst().get();
    Student groupHeadman = this.students.stream().filter(student ->
student.getId() == headmanId).findFirst().get();

    this.facultyChoiceBox.setValue(groupFaculty);
    this.headmanChoiceBox.setValue(groupHeadman);

} catch (final Exception exception) {
    Logger.getLogger(CreateController.class.getName()).log(Level.SEVERE, null,
exception);
}
}

@FXML
private void update() {
    if (this.groupNameTextField.getText().isEmpty() ||
this.facultyChoiceBox.getSelectionModel().isEmpty()
|| this.headmanChoiceBox.getSelectionModel().isEmpty()) {
        return;
    }

    try {

```

```

        this.db.updateGroup(new Group(this.updatingId,
this.headmanChoiceBox.getSelectionModel().getSelectedItem().getId(),
        this.facultyChoiceBox.getSelectionModel().getSelectedItem().getId(),
this.groupNameTextField.getText(),
        Short.parseShort(startYearTextField.getText()),
Short.parseShort(endYearTextField.getText())));
    } catch (NumberFormatException ignore) { }
}
}
}

```

```

public final class MainController implements Initializable {

```

```

    @FXML
    private TableView<Group> groupsTableView; @FXML
    private TableColumn<Group, String> groupsTableViewId;
    @FXML
    private TableColumn<Group, String> groupsTableViewName; @FXML
    private TableColumn<Group, String> groupsTableViewStartYear;
    @FXML
    private TableColumn<Group, Date> groupsTableViewEndYear;

```

```

    @FXML
    private TextField
    facultyNameTextField; @FXML
    private TextField headmanNameTextField;

```

```

    @FXML
    private TextField groupNameTextField;
    @FXML
    private TextField startYearTextField;

```

```

    private ObservableList<Group> groups =
    null; private DB db = null;

```

```

    String groupNameFilter =
    null; Short startYearFilter
    = null;

```

```

    @Override
    public void initialize(URL arg0, ResourceBundle
    arg1) { this.groups =
    FXCollections.observableArrayList(); this.db =
    new DB();

    this.groupsTableViewId.setCellValueFactory(new PropertyValueFactory<>("id"));
    this.groupsTableViewName.setCellValueFactory(new PropertyValueFactory<>("name"));
    this.groupsTableViewStartYear.setCellValueFactory(new
PropertyValuFactory<>("startYear"));
    this.groupsTableViewEndYear.setCellValueFactory(new
PropertyValuFactory<>("endYear"));

    ChangeListener<Object> listener = (obs, oldValue, newValue) -
    > { try {

```

```

        Group group = this.groupsTableView.getSelectionModel().getSelectedItem();

        if (group == null) {
            return;
        }

        ResultSet groupDetailsSet = this.db.getGroupDetails(group.getId());
        Objects.requireNonNull(groupDetailsSet).next();

        this.facultyNameTextField.setText(groupDetailsSet.getString("faculty_name"));
        this.headmanNameTextField.setText(groupDetailsSet.getString("headman_name"));
    } catch (final Exception exception) {
        Logger.getLogger(MainController.class.getName()).log(Level.SEVERE, null,
            exception);
    }
};

this.groupsTableView.focusedProperty().addListener(listener);
this.groupsTableView.getSelectionModel().selectedItemProperty().addListener(listener);

read();
}

```

@FXML

private void

```

    create() { try {
        Parent parent = App.loadFXML("create");
        Stage stage = new Stage();
        stage.setScene(new Scene(parent));
        stage.initStyle(StageStyle.UTILITY);
        stage.show();
    } catch (final IOException exception) {
        Logger.getLogger(MainController.class.getName()).log(Level.SEVERE, null,
            exception);
    }
}

```

@FXML

private void read()

```

    { try {
        this.clearDetailFields();
        this.groups.clear();

        ResultSet groupsSet = this.db.getGroups(this.groupNameFilter,

            this.startYearFilter, null); while (Objects.requireNonNull(groupsSet).next())

        {
            this.groups.add(new Group(groupsSet.getInt("id"), null, null,
groupsSet.getString("name"),
                groupsSet.getShort("start_year"),
                groupsSet.getShort("end_year")));
            this.groupsTableView.setItems(this.groups);
        }
    } catch (final Exception exception) {
        Logger.getLogger(MainController.class.getName()).log(Level.SEVERE, null,
            exception);
    }
}

```

```

}

@FXML
private void update()
{ try {
    Group group = this.groupsTableView.getSelectionModel().getSelectedItem();

    if (group ==
        null) { return;
    }

    FXMLLoader loader = new FXMLLoader(getClass().getResource("update.fxml"));
    Parent parent = loader.load();

    UpdateController updateController = loader.getController();
    updateController.setUpdatingId(group.getId());

    Stage stage = new Stage();
    stage.setScene(new
    Scene(parent));
    stage.initStyle(StageStyle.UTILITY
    ); stage.show();
} catch (final Exception exception) {
    Logger.getLogger(MainController.class.getName()).log(Level.SEVERE, null,
    exception);
}
}

```

```

@FXML
private void delete() {
    Group group = this.groupsTableView.getSelectionModel().getSelectedItem();

    if (group ==
        null) { return;
    }

    this.db.deleteGroupById(group.getId());
}

```

```

@FXML
private void search() {
    this.groupNameFilter = this.groupNameTextField.getText();

    try {
        this.startYearFilter = Short.parseShort(this.startYearTextField.getText());
    } catch (NumberFormatException ignore) {
        this.startYearFilter = null;
    }

    this.read();
}

```

```

private void clearDetailFields() {
    this.facultyNameTextField.clear();
}

```

```

        this.headmanNameTextField.clear();
    }
}

public final class DB {

    private Connection connection;
    private static final String GROUPS_TABLE = "groups";

    public static final String FACULTIES_TABLE =
        "faculty"; public static final String STUDENTS_TABLE =
        "students";

    public DB() {
        try {
            connection =
DriverManager.getConnection("jdbc:postgresql://localhost:5432/spp", "postgres",
"admin");
        } catch (final SQLException exception) {
            Logger.getLogger(DB.class.getName()).log(Level.SEVERE, null, exception);
        }
    }

    public ResultSet getAll(final String
        tableName) { try {
            Statement statement = this.connection.createStatement();
            statement.closeOnCompletion();

            return statement.executeQuery("SELECT * FROM " + tableName + " ORDER BY id");
        } catch (final SQLException exception) {
            Logger.getLogger(DB.class.getName()).log(Level.SEVERE, null, exception);
        }

        return null;
    }

    public ResultSet getGroups(final String nameFilter, final Short startYearFilter,
        final Integer id) { try {
            Statement statement = this.connection.createStatement();
            statement.closeOnCompletion();

            String sql = "SELECT id, name, faculty_id, headman_id, start_year, end_year
FROM " + GROUPS_TABLE;

            if (id != null) {
                sql += " WHERE id = " + id;

                return statement.executeQuery(sql);
            }

            if (nameFilter != null) {
                sql += " WHERE name LIKE '%" + nameFilter + "%'";
            }
        }
    }
}

```



```

        if (startYearFilter != null) {
            sql += nameFilter == null ? " WHERE" : "
            AND"; sql += " start_year = " +
            startYearFilter;
        }

        return statement.executeQuery(sql);
    } catch (final SQLException exception) {
        Logger.getLogger(DB.class.getName()).log(Level.SEVERE, null, exception);
    }

    return null;
}

public ResultSet getGroupDetails(final int
id) { try {
    Statement statement = this.connection.createStatement();
    statement.closeOnCompletion();

    return statement.executeQuery("SELECT " + FACULTIES_TABLE + ".faculty_name,
CONCAT(" + STUDENTS_TABLE + ".first_name, ' ', " + STUDENTS_TABLE + ".last_name) AS
headman_name "
                                + "FROM " + GROUPS_TABLE + " "
                                + "INNER JOIN " + FACULTIES_TABLE + " ON " + GROUPS_TABLE +
".faculty_id = " + FACULTIES_TABLE + ".id "
                                + "INNER JOIN " + STUDENTS_TABLE + " ON " + GROUPS_TABLE +
".headman_id = " + STUDENTS_TABLE + ".id "
                                + "WHERE " + GROUPS_TABLE + ".id = " + id);
    } catch (final SQLException exception) {
        Logger.getLogger(DB.class.getName()).log(Level.SEVERE, null, exception);
    }

    return null;
}

public void deleteGroupById(final int
id) { try {
    Statement statement = this.connection.createStatement();
    statement.closeOnCompletion();
    statement.executeUpdate("DELETE FROM " + GROUPS_TABLE + " WHERE id = " +
id)
;    } catch (final SQLException exception) {
        Logger.getLogger(DB.class.getName()).log(Level.SEVERE, null,
exception);
    }
}

public void createGroup(final Group group)
{ try {
    Statement statement = this.connection.createStatement();
    statement.closeOnCompletion();

    String query = "INSERT INTO " + GROUPS_TABLE + "(name, faculty_id, headman_id,
start_year, end_year) "

```

```

        + "VALUES ('" + group.getName() + "', " + group.getFacultyId() + ", " +
group.getHeadmanId() + ", " + group.getStartYear() + ", " + group.getEndYear() + ")";
        statement.executeUpdate(query);
    } catch (final SQLException exception) {
        Logger.getLogger(DB.class.getName()).log(Level.SEVERE, null, exception);
    }
}

```

```

public void updateGroup(final Group group)
{ try {
    Statement statement = this.connection.createStatement();
    statement.closeOnCompletion();

    String query = "UPDATE " + GROUPS_TABLE + " "
        + "SET name = '" + group.getName() + "', faculty_id = " +
group.getFacultyId() + ", headman_id = " + group.getHeadmanId() + ", start_year = " +
group.getStartYear() + ", end_year = " + group.getEndYear() + " "
        + "WHERE id = " + group.getId();
    statement.executeUpdate(query);
} catch (final SQLException exception) {
    Logger.getLogger(DB.class.getName()).log(Level.SEVERE, null, exception);
}
}
}

```

```

public final class Faculty {

    private int id;
    private final String name;

    public Faculty(final int id, final String
        name) { this.id = id;
        this.name = name;
    }

    public int getId() {
        return id;
    }

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

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

```

```

public final class Group

{ private Integer id;

```

```

private final Integer
headmanId; private final
Integer facultyId; private
final String name; private
final short startYear;
private final short endYear;

public Group(final Integer id, final Integer headmanId, final Integer facultyId, final
    String name, final short startYear, short endYear) {
    this.id = id;
    this.headmanId =
    headmanId; this.facultyId
    = facultyId; this.name =
    name; this.startYear =
    startYear; this.endYear =
    endYear;
}

public int getId() {
    return id;
}

public int getHeadmanId() {
    return headmanId;
}

public int getFacultyId() {
    return facultyId;
}

public String getName() {
    return name;
}

public short getStartYear()
{ return startYear;
}

public short getEndYear() {
    return endYear;
}

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

public final class Student {

    private int id;
    private final String
    firstName; private final
    String lastName;

    public Student(final int id, final String firstName, final String lastName) {

```

```

        this.id = id;
        this.firstName = firstName;
        this.lastName = lastName;
    }

    public int getId() {
        return id;
    }

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

    @Override
    public String toString() {
        return firstName + " " + lastName;
    }
}

```

ID	Name	Start year	End year	
1	ПО-5	2019	2023	
4	ПО-4	2018	2022	

Faculty name:
Headman name:

Search by group name:
Search by start year

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