ПРИЛОЖЕНИЕ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

РАЗРАБОТКА ПРОГРАММЫ РАСЧЕТА СТИПЕНДИИ

КОД ПРОГРАММЫ

КП.ПО-9.1-40 01 01

Листов 19

Руководитель А. В. Сааков

Выполнил З. С. Харитонович

Консультант по ЕСПД А. В. Сааков

Брест 2023

MainApplication.java

```
package com.course.project_javafx;
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.stage.Stage;
import java.io.IOException;

public class MainApplication extends Application {
    private static Stage guiStage;
    private static User user;
    public static Stage getStage() {
        return guiStage;
    }
    public static User getUser() {
        return user;
    }
    public static void setUser(User user) {
        MainApplication.user = user;
    }
    @Override
    public void start(Stage stage) throws IOException {
        guiStage = stage;
        changeScene("authentication-view.fxml",    "Авторизация",
320, 240);
        stage.show();
    }
    public static void main(String[] args) {
        launch();
    }
    public static void changeScene(String file, String title, int
v, int v1) throws IOException {
        FXMLLoader           fxmlLoader           =           new
FXMLLoader(MainApplication.class.getResource(file));
        Scene authScene = new Scene(fxmlLoader.load(), v, v1);
        getStage().setTitle(title);
        getStage().setScene(authScene);
    }
}
```

AuthenticationController.java

```
package com.course.project_javafx;
import javafx.fxml.FXML;
import javafx.scene.control.Label;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;

import java.math.BigInteger;
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.sql.ResultSet;

public class AuthenticationController {
```

```
    @FXML
    public Label errorText;
    public TextField loginText;
    public PasswordField passwordText;
    @FXML
    public void onLoginButtonClick() {
        try {
            String login = loginText.getText();
            String password = passwordText.getText();
            MainApplication.setUser(auth(login, password));
            if (MainApplication.getUser() != null) {
                MainApplication.changeScene("menu-view.fxml",
"Меню", 320, 240);
            } else {
                errorText.setText("Неверный логин или пароль");
            }
        } catch (Exception ex) {
            System.out.println(ex);
        }
    }
    private static User auth(String login, String password) throws
NoSuchAlgorithmException {
        User user = null;
        String passwordHash = stringToHash(password);

        try {
ResultSet rs = Database.sqlRequest("SELECT * FROM users WHERE
login=\"" + login + "\";");
            rs.next();
            if (passwordHash.equals(rs.getString("password"))) {
                user    =    new    User(rs.getString("login"),
rs.getBoolean("role"));
                System.out.println("User " + login + " logged
in.");
            }
        } catch (Exception e) {
            System.out.println(e);
        }
        return user;
    }
    public static String stringToHash(String str) throws
NoSuchAlgorithmException {
        MessageDigest digest = MessageDigest.getInstance("SHA-
256"); // хеширование пароля
        byte[]                          hash                    =
digest.digest(str.getBytes(StandardCharsets.UTF_8));
        return String.format("%064x", new BigInteger(1, hash));
    }
}
```

MenuController.java

```
package com.course.project_javafx;
import javafx.event.ActionEvent;
import javafx.scene.control.Label;
```

```
import java.io.IOException;

public class MenuController {
    public Label errorText;
    public void onAccountManagerButtonClick() throws IOException
{
        if (MainApplication.getUser().isRole())
            MainApplication.changeScene("account-manager-
view.fxml", "Управление учётными записями", 1280, 720);
        else errorText.setText("Недостаточно прав.");
    }
    public  void  onScholarshipCalculationButtonClick()  throws
IOException {
        MainApplication.changeScene("scholarship-calculation-
view.fxml", "Расчёт стипендии", 1280, 720);
    }
    public void onExitButtonClick() throws IOException {
        MainApplication.setUser(null);
        MainApplication.changeScene("authentication-view.fxml",
"Авторизация", 320, 240);
    }
}
```

AccountManagerController.java

```
package com.course.project_javafx;
import javafx.event.ActionEvent;
import javafx.scene.control.*;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.scene.input.MouseEvent;
import java.io.IOException;
import java.sql.ResultSet;
import java.util.Optional;

public class AccountManagerController {

    public TableView<User> table;
    private      final      ObservableList<User>      data      =
FXCollections.observableArrayList();
    public Label addLabel;
    public TextField loginField;
    public TextField passwordField;
    public ChoiceBox roleChoiceBox;
    public TextField editLoginField;
    public TextField editPasswordField;
    public ChoiceBox editRoleChoiceBox;
    public Label editLabel;

    public void updateTable() {

table.setColumnResizePolicy(TableView.CONSTRAINED_RESIZE_POLICY)
;
        try {
```

```
            ResultSet rs = Database.sqlRequest("SELECT * FROM
users;");
            data.clear();
            while (rs.next()) {
                User curUser = new User(rs.getString("login"),
rs.getString("password"),
                        rs.getBoolean("role"));
                data.add(curUser);
            }
            table.setItems(data);
        } catch (Exception e) {
            System.out.println(e);
        }
    }
    public void onBackButtonClick() throws IOException {
        MainApplication.changeScene("menu-view.fxml",     "Меню",
320, 240);
    }
    public void onAddButtonClick() {
        if          (loginField.getText().equals("")          ||
passwordField.getText().equals("")) {
            addLabel.setText("Заполните все поля");
            return;
        }
        try {
            ResultSet rs = Database.sqlRequest("SELECT * FROM
users WHERE login=\"" + loginField.getText() + "\";");
            if (!rs.next()) {
                String uRole;
                if
(roleChoiceBox.getValue().equals("Пользователь")) {
                    uRole = "0";
                }                       else                    if
(roleChoiceBox.getValue().equals("Администратор")) {
                    uRole = "1";
                } else {
                    addLabel.setText("Что-то пошло не так.");
                    return;
                }
                Database.sqlUpdate("INSERT INTO users values(\""
+ loginField.getText() + "\", \""
                        +
AuthenticationController.stringToHash(passwordField.getText())  +
"\", " + uRole + ");");
                addLabel.setText("Пользователь           добавлен
успешно.");
                loginField.setText("");
                passwordField.setText("");
                roleChoiceBox.setValue("Пользователь");
                this.updateTable();
            } else {
                addLabel.setText("Пользователь  с  таким  логином
уже существует.");
```

```
            }
        } catch (Exception ex) {
            addLabel.setText("Что-то пошло не так.");
            System.out.println(ex);
        }
    }
    public void onTableClicked(MouseEvent mouseEvent) {
        User user = table.getSelectionModel().getSelectedItem();
        if (user == null) return;
        editLoginField.setText(user.getLogin());
        editLabel.setText("");
        if (user.isRole()) {
            editRoleChoiceBox.setValue("Администратор");
        } else {
            editRoleChoiceBox.setValue("Пользователь");
        }
    }
    public void onDeleteButtonClicked() {
        if (table.getSelectionModel().getSelectedItem() == null)
{
            editLabel.setText("Выберите запись.");
            return;
        }
        Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
        alert.setTitle("Подтверждение");
        alert.setHeaderText("Удаление записи пользователя " +
table.getSelectionModel().getSelectedItem().getLogin());
        alert.setContentText("Вы точно хотите удалить эту
запись?");
        ButtonType buttonYes = new ButtonType("Да");
        ButtonType buttonCancel = new ButtonType("Отмена",
ButtonBar.ButtonData.CANCEL_CLOSE);
        alert.getButtonTypes().setAll(buttonYes, buttonCancel);
        Optional<ButtonType> result = alert.showAndWait();
        if (result.get() == buttonYes) {
            try {
                Database.sqlUpdate("DELETE FROM users WHERE
login=\""                                                    +
table.getSelectionModel().getSelectedItem().getLogin() + "\";");
                this.updateTable();
                editLabel.setText("Запись удалена.");
            } catch (Exception e) {
                editLabel.setText("Что-то пошло не так.");
                System.out.println(e);
            }
        }
    }
    public void onEditButtonClicked() {
        if (table.getSelectionModel().getSelectedItem() == null)
{
            editLabel.setText("Выберите запись.");
            return;
        }
```

```
        if        (editLoginField.getText().equals("")        ||
editRoleChoiceBox.getValue().equals("")) {
            editLabel.setText("Недостаточно данных.");
            return;
        }
        String role;
        if (editRoleChoiceBox.getValue().equals("Администратор"))
{
            role = "1";
        } else {
            role = "0";
        }
        String newLogin = editLoginField.getText();
        if
(!newLogin.equals(table.getSelectionModel().getSelectedItem().ge
tLogin())) {
            try {
                ResultSet rs = Database.sqlRequest("SELECT * FROM
users WHERE login=\"" + newLogin + "\";");
                if (rs.next()) {
                    editLabel.setText("Пользователь            с
таким\nлогином уже существует.");
                    return;
                }
            } catch (Exception e) {
                editLabel.setText("Что-то пошло не так.");
                System.out.println(e);
                return;
            }
        }
        String newPassword = new String();
        if (!editPasswordField.getText().equals("")) {
            try {
                newPassword                                    =
AuthenticationController.stringToHash(editPasswordField.getText(
));
            } catch (Exception e) {
            }
        } else {
            newPassword                                        =
table.getSelectionModel().getSelectedItem().getPassword();
        }
        try {
            Database.sqlUpdate("UPDATE  users  SET  login=\""  +
newLogin + "\", password=\"" + newPassword
                    + "\", role=" + role +
                    "          WHERE          login=\""          +
table.getSelectionModel().getSelectedItem().getLogin() + "\";");
            editLabel.setText("Запись обновлена.");
            editLoginField.setText("");
            editPasswordField.setText("");
            editRoleChoiceBox.setValue("            ");
            this.updateTable();
```

```
        } catch (Exception e) {
            editLabel.setText("Что-то пошло не так.");
            System.out.println(e);
            ;
        }
    }
}
```

ScholarshipCalculationController.java

```
package com.course.project_javafx;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.scene.control.ChoiceBox;
import javafx.scene.control.Label;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;
import javafx.scene.input.MouseEvent;
import java.io.IOException;
import java.sql.ResultSet;

public class ScholarshipCalculationController {

    public TableView<Student> table;
    private final ObservableList<Student> data =
FXCollections.observableArrayList();
    public ChoiceBox searchChoiceBox;
    public TextField searchTextField;
    public Label errorLabel;
    public TextField scholarshipTextField;

    public void updateTable() {
table.setColumnResizePolicy(TableView.CONSTRAINED_RESIZE_POLICY)
;
        try {
            ResultSet rs = Database.sqlRequest("SELECT * FROM
students;");
            data.clear();
            while (rs.next()) {
                data.add(new Student(rs));
            }
            table.setItems(data);
        } catch (Exception e) {
            System.out.println(e);
        }
    }
    public void onBackButtonClicked() throws IOException {
        MainApplication.changeScene("menu-view.fxml", "Меню",
320, 240);
    }
    public void onSearchButtonClicked() {
        updateTable();
        ObservableList<Student> newData =
FXCollections.observableArrayList();
        Object value = searchChoiceBox.getValue();
```

```
        for (Student curStudent : data) {
            if (value.equals("по всем полям")) {
                if
(curStudent.toString().contains(searchTextField.getText())) {
                    newData.add(curStudent);
                }
            } else if (value.equals("id")) {
                if
(curStudent.getId().contains(searchTextField.getText())) {
                    newData.add(curStudent);
                }
            } else if (value.equals("ФИО")) {
                if
(curStudent.getNSP().contains(searchTextField.getText())) {
                    newData.add(curStudent);
                }
            } else if (value.equals("группа")) {
                if
(curStudent.getGroup().contains(searchTextField.getText())) {
                    newData.add(curStudent);
                }
            } else if (value.equals("форма")) {
                if
(curStudent.getEduForm().contains(searchTextField.getText())) {
                    newData.add(curStudent);
                }
            } else if (value.equals("общ. деятельность")) {
                if
(curStudent.getSocWork().contains(searchTextField.getText())) {
                    newData.add(curStudent);
                }
            }
        }
        table.setItems(newData);
    }
    public void onCalculateButtonClick() {
        for (Student cur : data) {
            if (cur.getEduFormRaw()) {
                int sum = 0;
                boolean isExc = true;
                for (int exam : cur.getExams()) {
                    if (exam < 9) {
                        isExc = false;
                    }
                    sum += exam;
                }
                if (sum / 4.0 < 5.0) {
                    cur.setScholarship(0.0);
                    continue;
                }
                double k = 1.0;
                if (isExc) {
                    k += 0.25;
```

```
                    if (cur.getSocWorkRaw()) {
                        k += 0.25;
                    }
                }
                for (boolean credit : cur.getCredits()) {
                    if (!credit) {
                        k = 0;
                        break;
                    }
                }
cur.setScholarship(Double.parseDouble(scholarshipTextField.getTe
xt()) * k);
            }
            Database.sqlUpdate("UPDATE students SET scholarship="
+ cur.getScholarship() + " WHERE id=" + cur.getId());
        }
        updateTable();
    }
    public void onAdminButtonClicked() throws IOException {
        if (MainApplication.getUser().isRole()) {
            MainApplication.changeScene("scholarship-admin-
view.fxml", "Расчёт стипендии", 1280, 720);
        } else {
            errorLabel.setText("Недостаточно прав.");
        }
    }
}
```

ScholarshipAdminController.java

```
package com.course.project_javafx;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.scene.control.*;
import javafx.scene.input.MouseEvent;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Optional;

public       class       ScholarshipAdminController       extends
ScholarshipCalculationController {

    public TableView<Student> table;
    private     final     ObservableList<Student>     data     =
FXCollections.observableArrayList();
    public ChoiceBox searchChoiceBox;
    public TextField searchTextField;
    public TextField editNSPField;
    public Label editLabel;
    public TextField editGroupField;
    public ChoiceBox editEduFormChoiceBox;
    public CheckBox credit0CheckBox;
    public CheckBox credit1CheckBox;
    public CheckBox credit2CheckBox;
    public CheckBox credit3CheckBox;
```

```
    public CheckBox credit4CheckBox;
    public ChoiceBox exam0ChoiceBox;
    public ChoiceBox exam1ChoiceBox;
    public ChoiceBox exam2ChoiceBox;
    public ChoiceBox exam3ChoiceBox;
    public ChoiceBox editSocWorkChoiceBox;
    public TextField scholarshipTextField;
    @Override
    public void onBackButtonClicked() throws IOException {
        MainApplication.changeScene("scholarship-calculation-
view.fxml", "Расчёт стипендии", 1280, 720);
    }
    public void onAddButtonClicked() {
        String[] fields = formToStrings();
        if (fields == null) return;
        try {
            Database.sqlUpdate("INSERT INTO students values
(NULL, \"" + fields[0] + "\", \"" + fields[1] + "\", " +
                    fields[2] + ", \"" + fields[3] + "\", \"" +
fields[4] + "\", " + fields[5] + ", 0);");
            updateTable();
            clearForm();
        } catch (Exception e) {
            System.out.println(e);
            editLabel.setText("Что-то пошло не так.");
        }
    }
public void onDeleteButtonClicked() {
        if (table.getSelectionModel().getSelectedItem() == null)
{
            editLabel.setText("Выберите запись.");
            return;
        }
        Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
        alert.setTitle("Подтверждение");
        alert.setHeaderText("Удаление записи студента " +
table.getSelectionModel().getSelectedItem().getNSP());
        alert.setContentText("Вы точно хотите удалить эту
запись?");
        ButtonType buttonYes = new ButtonType("Да");
        ButtonType buttonCancel = new ButtonType("Отмена",
ButtonBar.ButtonData.CANCEL_CLOSE);
        alert.getButtonTypes().setAll(buttonYes, buttonCancel);
        Optional<ButtonType> result = alert.showAndWait();
        if (result.get() == buttonYes) {
            try {
                Database.sqlUpdate("DELETE FROM students WHERE
id=\""  +  table.getSelectionModel().getSelectedItem().getId()  +
"\";");
                updateTable();
                editLabel.setText("Запись удалена.");
                clearForm();
            } catch (Exception e) {
```

```
                editLabel.setText("Что-то пошло не так.");
                System.out.println(e);
            }
        }
    }
    public void onEditButtonClicked() {
        if (table.getSelectionModel().getSelectedItem() == null)
{
            editLabel.setText("Выберите запись.");
            return;
        }
        final String[] fields = formToStrings();
        if (fields == null) return;
        try {
            Database.sqlUpdate("UPDATE students SET nsp=\"" +
fields[0] + "\", grp=\"" + fields[1] + "\", eduForm="
                    + fields[2] + ", credits=\"" + fields[3] +
"\", exams=\"" + fields[4] + "\", socWork=" + fields[5] +
                    "          WHERE          id="          +
table.getSelectionModel().getSelectedItem().getId() + ";");
            updateTable();
            editLabel.setText("Запись изменена.");
            clearForm();
        } catch (Exception e) {
            System.out.println(e);
            editLabel.setText("Что-то пошло не так.");
        }
    }
public void clearForm() {
        editNSPField.setText("");
        editGroupField.setText("");
        editSocWorkChoiceBox.setValue("");
        editEduFormChoiceBox.setValue("");
        credit0CheckBox.setSelected(false);
        credit1CheckBox.setSelected(false);
        credit2CheckBox.setSelected(false);
        credit3CheckBox.setSelected(false);
        credit4CheckBox.setSelected(false);
        exam0ChoiceBox.setValue("");
        exam1ChoiceBox.setValue("");
        exam2ChoiceBox.setValue("");
        exam3ChoiceBox.setValue("");
    }
private String[] formToStrings() {
        String NSP = editNSPField.getText();
        String group = editGroupField.getText();

        String socWork = new String();
        if
(editSocWorkChoiceBox.getValue().toString().equals("Активная")) {
            socWork = "1";
```

```
        }                           else                            if
(editSocWorkChoiceBox.getValue().toString().equals("Неактивная")
) {
            socWork = "0";
        }
        String eduForm = new String();

if(editEduFormChoiceBox.getValue().toString().equals("Бюджетная"
){
            eduForm = "1";
        }                           else                            if
(editEduFormChoiceBox.getValue().toString().equals("Платная")) {
            eduForm = "0";
        }
        if     (NSP.equals("")     ||     group.equals("")     ||
socWork.equals("")        ||        eduForm.equals("")        ||
exam0ChoiceBox.getValue() == null || exam1ChoiceBox.getValue() ==
null     ||     exam2ChoiceBox.getValue()     ==     null     ||
exam3ChoiceBox.getValue() == null) {
            editLabel.setText("Заполните все поля.");
            return null;
        }
        ArrayList<CheckBox>     creditsCheckBoxes     =     new
ArrayList<>(){{
                add(credit0CheckBox);
                add(credit1CheckBox);
                add(credit2CheckBox);
                add(credit3CheckBox);
                add(credit4CheckBox);}};
        String credits = new String();
        for (CheckBox cur : creditsCheckBoxes) {
            if (cur.isSelected()) {
                credits += "1, ";
            } else {
                credits += "0, ";
            }
        }
        credits = credits.substring(0, credits.length() - 2);
        ArrayList<ChoiceBox>     examsChoiceBoxes     =     new
ArrayList<>(){
            {add(exam0ChoiceBox);
                add(exam1ChoiceBox);
                add(exam2ChoiceBox);
                add(exam3ChoiceBox);}};
        String exams = new String();
        for (ChoiceBox cur : examsChoiceBoxes) {
            exams += cur.getValue().toString() + ", ";
        }
        exams = exams.substring(0, exams.length() - 2);
        String[] strings = {NSP, group, eduForm, credits, exams,
socWork};
        return strings;
    }
```

```
    @Override
    public void onTableClicked(MouseEvent mouseEvent) {
        Student                    student                    =
table.getSelectionModel().getSelectedItem();
        if (student == null) return;
        editNSPField.setText(student.getNSP());
        editGroupField.setText(student.getGroup());
        if (student.getSocWork().equals("1")) {
            editSocWorkChoiceBox.setValue("Активная");
        } else if (student.getSocWork().equals("0")) {
            editSocWorkChoiceBox.setValue("Неактивная");
        }
        if (student.getEduForm().equals("Б")) {
            editEduFormChoiceBox.setValue("Бюджетная");
        } else if (student.getEduForm().equals("П")) {
            editEduFormChoiceBox.setValue("Платная");
        }
        if (student.getCredit0().equals("Зачт.")) {
            credit0CheckBox.setSelected(true);
        } else {
            credit0CheckBox.setSelected(false);
        }
        if (student.getCredit1().equals("Зачт.")) {
            credit1CheckBox.setSelected(true);
        } else {
            credit1CheckBox.setSelected(false);
        }
        if (student.getCredit2().equals("Зачт.")) {
            credit2CheckBox.setSelected(true);
        } else {
            credit2CheckBox.setSelected(false);
        }
        if (student.getCredit3().equals("Зачт.")) {
            credit3CheckBox.setSelected(true);
        } else {
            credit3CheckBox.setSelected(false);
        }
        if (student.getCredit4().equals("Зачт.")) {
            credit4CheckBox.setSelected(true);
        } else {
            credit4CheckBox.setSelected(false);
        }
        exam0ChoiceBox.setValue(student.getExam0());
        exam1ChoiceBox.setValue(student.getExam1());
        exam2ChoiceBox.setValue(student.getExam2());
        exam3ChoiceBox.setValue(student.getExam3());
        if (student.getSocWork().equals("Акт.")) {
            editSocWorkChoiceBox.setValue("Активная");
        } else {
            editSocWorkChoiceBox.setValue("Неактивная");
        }
    }
}
```

Database.java

```
package com.course.project_javafx;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class Database {
    static final String DB_URL =
"jdbc:mysql://127.0.0.1:3306/student";
    static final String USER = "root";
    static final String PASS = "toor";
    public static ResultSet sqlRequest(String query) {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection conn = DriverManager.getConnection(DB_URL,
USER, PASS);
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery(query);

            return rs;
        } catch (Exception e) {
            System.out.println(e);
            return null;
        }
    }
    public static void sqlUpdate(String query) {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection conn = DriverManager.getConnection(DB_URL,
USER, PASS);
            Statement stmt = conn.createStatement();
            stmt.executeUpdate(query);
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

User.java

```
package com.course.project_javafx;

public class User {
    private String login;
    private String password;
    private boolean role;
    User(String login, boolean role) {
        this.login = login;
        this.password = null;
        this.role = role;
    }
    User(String login, String password, boolean role) {
        this.login = login;
        this.password = password;
```

```
        this.role = role;
    }
    public String getLogin() {
        return login;
    }
    public void setLogin(String login) {
        this.login = login;
    }
    public String getRole() {
        if (role) {
            return "1";
        } else {
            return "0";
        }
    }
    public boolean isRole() {
        return role;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    public void setRole(boolean role) {
        this.role = role;
    }
}
```

Student.java

```
package com.course.project_javafx;
import java.sql.ResultSet;
import java.sql.SQLException;

public class Student {
    private int id;
    private String NSP; // ФИО
    private String group; // группа
    private boolean eduForm; // форма обучения
    private boolean[] credits; // зачёты ??
    private int[] exams; // экзамены
    private boolean socWork; // участие в общественной работе
    private double scholarship; // размер стипендии
    Student(ResultSet rs) throws SQLException {
        id = rs.getInt("id");
        group = rs.getString("grp");
        NSP = rs.getString("nsp");
        eduForm = rs.getBoolean("eduForm");
        credits = new boolean[5];
        String[] words = rs.getString("credits").split(", ");
        for (int i = 0; i < 5; i++) {
            if (words[i].equals("1")) {
                credits[i] = true;
            } else {
```

```
                credits[i] = false;
            }
        }
        exams = new int[4];
        words = rs.getString("exams").split(", ");
        for (int i = 0; i < 4; i++) {
            exams[i] = Integer.parseInt(words[i]);
        }
        socWork = rs.getBoolean("socWork");
        scholarship = rs.getDouble("scholarship");
    }
    public String getId() {
        return Integer.toString(id);
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getGroup() {
        return group;
    }
    public void setGroup(String group) {
        this.group = group;
    }
    public String getNSP() {
        return NSP;
    }
    public void setNSP(String NSP) {
        this.NSP = NSP;
    }
    public String getEduForm() {
        if (eduForm) {
            return "Б";
        } else {
            return "П";
        }
    }
    public boolean getEduFormRaw() {
        return eduForm;
    }
    public void setEduForm(boolean eduForm) {
        this.eduForm = eduForm;
    }
    public String getCredit0() {
        if (credits[0]) {
            return "Зачт.";
        } else {
            return "Н/З";
        }
    }
    public String getCredit1() {
        if (credits[1]) {
            return "Зачт.";
        } else {
```

```
            return "Н/З";
        }
    }
    public String getCredit2() {
        if (credits[2]) {
            return "Зачт.";
        } else {
            return "Н/З";
        }
    }
    public String getCredit3() {
        if (credits[3]) {
            return "Зачт.";
        } else {
            return "Н/З";
        }
    }
    public String getCredit4() {
        if (credits[4]) {
            return "Зачт.";
        } else {
            return "Н/З";
        }
    }
    public void setCredits(boolean[] credits) {
        this.credits = credits;
    }
    public int[] getExams() {
        return exams;
    }
    public String getExam0() {
        return Integer.toString(exams[0]);
    }
    public String getExam1() {
        return Integer.toString(exams[1]);
    }
    public String getExam2() {
        return Integer.toString(exams[2]);
    }
    public String getExam3() {
        return Integer.toString(exams[3]);
    }
    public void setExams(int[] exams) {
        this.exams = exams;
    }
    public String getSocWork() {
        if (socWork) {
            return "Акт.";
        } else {
            return "Неакт.";
        }
    }
    public boolean getSocWorkRaw() {
```

```
        return socWork;
    }
    public void setSocWork(boolean socWork) {
        this.socWork = socWork;
    }
    public boolean[] getCredits() {
        return credits;
    }
    public String getScholarship() {
        return Double.toString(scholarship);
    }
    public void setScholarship(double scholarship) {
        this.scholarship = scholarship;
    }

    public String toString() {
        return getId() + " " + getNSP() + " " + getGroup() + " "
+ getEduForm() + " " + getSocWork();
    }
}
```