

西北工业大学
Northwestern Polytechnical University

Object Oriented Programming (U10M12004) Group Project Report

Title	Registration Page
Group No.	3
Leader	Turysbay Dinmukhambet
Member(s)	Preke Daulet

May 19 2021

Introduction to Java

Java is a general-purpose, concurrent, class-based, object-oriented computer programming language that is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that code that runs on one platform does not need to be recompiled to run on another. Java applications are typically compiled to byte code (class file) that can run on any Java virtual machine (JVM) regardless of computer architecture. Java is, as of 2012, one of the most popular programming languages in use, particularly for client-server web applications, with a reported 10 million users. Java was originally developed by James Gosling at Sun Microsystems (which has since merged into Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them.

The Java language has several libraries for creating user interface. One of the most popular is JavaFx. It is on the basis of JavaFx that we built an application with design, database connection and functionality. There is also a Java Swing library, which is

analogous to JavaFx. It makes no sense to use Swing today, because the library is no longer widely used and a new JavaFx library has come to replace it.

Project structure:

When we create an empty JavaFx-based project, we get a project with several folders and files in them at once. Each individual page in the application consists of two files: a design file and a function file. The entire design of the application is recorded in files with the .fxml extension. There can be an unlimited number of such files. A controller class is added to each such file, which is responsible for processing all actions on a specific page. In the controller, you can track various clicks, you can receive data from the fields and perform any manipulations associated with the application itself. Decoration of an app is done via Scene Builder.

So when we start creating an app folder we will have several files and folders over there. Inside source folder we can see sample folder where we store three files:

- 1) Controller: responsible for functionality of our application
- 2) Main: it will run all the methods
- 3) sample.fxml: template which displays main page of our app

Scene Builder enabled us to quickly design the view of our program, first of all we have set width and height of window to be 700 and 400 respectively. Then we put blue color on navbar and white color for content. Navbar has name of our app (label) right on the center and button for “sign up” on the right top corner, and content contains fields for completing login and password and “sign in” button.

Clicking “sign up” button redirects user to Sign Up page, there will be spotted several text fields for name, surname, password, country, login and checkboxes for verifying the gender of user plus “register” button. Apart from this Sign Up page is displayed via signUp.fxml file and all database operations inside it controlled by SignUpController class.

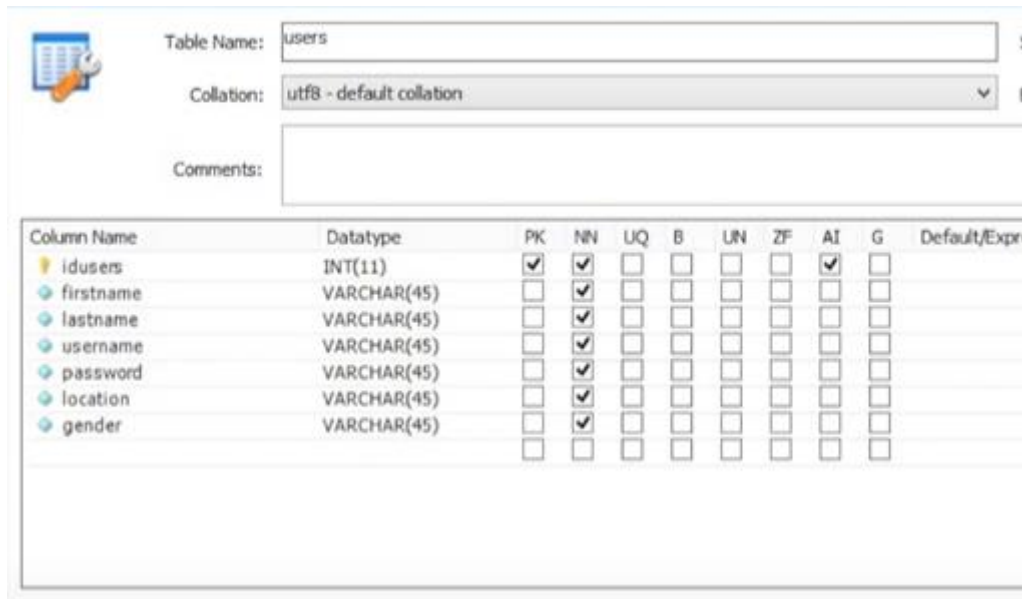
In order to keep track of all user information we downloaded Free MySQL Hosting and phpMyAdmin.

Links: 1) <https://www.freemysqlhosting.net/>

2) <https://www.phpmyadmin.co/>

Specific table is created to save registered user information, it has columns so each particular piece of information is stored in

different vertical cells.



The screenshot shows a database management interface for creating or editing a table named 'users'. The table name is entered in a text box, and the collation is set to 'utf8 - default collation'. Below this, there is a table defining the columns of the 'users' table.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expr
idusers	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
firstname	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
lastname	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
username	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
password	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
location	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
gender	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

As you see, the above photo illustrates structure of “users” table. It contains 7 columns, all are not null that means all fields must be filled in and all have datatype associated with them.

Java uses JDBC technology to work with databases. We always use this driver to connect to any database.

In addition, since we work with MySQL, it is necessary to download a special connector that combines Java JDBC and the MySQL database.

To access the database, we need to establish a connection, specify the driver, and also specify the connection parameters (host, login, password, and so on). The following code illustrates implementation:

```
Connection dbConnection;

public Connection getDbConnection() throws ClassNotFoundException, SQLException
{
    String connectionString = "jdbc:mysql://" + dbHost + ":" + dbPort + "/"
+ dbName;

    Class.forName("com.mysql.jdbc.Driver");

    dbConnection = DriverManager.getConnection(connectionString, dbUser,
dbPass);

    return dbConnection;
}
```

Names of tables, fields, databases, etc. are stored in variables and separate classes. This approach will make it easier to carry out the transfer of a project from one database to another.

User Registration:

To authorize a user, it is necessary to select values from the database, iterate over them and check for match with the entered data. This can be done using a special SQL query, which will return an array of received data in the format of the ResultSet class. This is how it works:

```
ResultSet result = dbHelper.getUser(user);

try {
    while(result.next()) {
        // Получаем имя пользователей
        String name = result.getString("firstname");
        System.out.println("User name is - " + name);
    }
} catch (SQLException e) {
    e.printStackTrace();
}
```

Once user registers and logs in, home page with “questionnaire” button shows up, and if it is clicked, it redirects user to “TEST” page and several questions pop up based on geography subject.

Below code with comments describes the functionality of its controller:

```
package sample;

import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.RadioButton;
import javafx.scene.control.ToggleGroup;
import javafx.scene.text.Text;

import java.util.Arrays;
import java.util.Collections;
import java.util.List;

public class Controller {

    // Fields that refer to objects of page
    @FXML
    private ToggleGroup answers;

    @FXML
```

```
private Text question_text;

@FXML
private RadioButton radio_btn_1;

@FXML
private RadioButton radio_btn_2;

@FXML
private RadioButton radio_btn_3;

@FXML
private RadioButton radio_btn_4;

@FXML
private Button answerBtn;

// An array based on the Questions class. Each object is a question with a set of possible answers
private Questions[] questions = new Questions[] {
    new Questions("What is the capital of Ireland", new String[] {"London", "Berlin",
    "Ottawa", "Dublin"}),
    new Questions("What is the capital of Italy", new String[] {"Milan", "Paris", "Madrid",
    "Rome"}),
    new Questions("What is the capital of Belgium", new String[] {"Amsterdam", "Astana",
    "Lisbon", "Brussels"}),
    new Questions("What is the capital of Spain", new String[] {"Shanghai", "Valencia",
    "Berlin", "Madrid"}),
    new Questions("What is the capital of Thailand", new String[] {"Delhi", "Kuala Lumpur",
    "Hanoi", "Bangkok"}),
    new Questions("What is the capital of Peru", new String[] {"Buenos Aires", "Santiago",
    "Bogota", "Lima"})
};

// Variables for storing the current question number and for counting the number of correct answers
private int nowQuestion = 0, correctAnswers;
// this variable consists the correct answer for current question
private String nowCorrectAnswer;

@FXML
public void initialize() {
    // assign correct answer to nowCorrectAnswer variable
    nowCorrectAnswer = questions[nowQuestion].correctAnswer();

    // this method works once user presses button to answer question
```



```
answerBtn.setOnAction(e -> {
    // obtain selected button by user
    RadioButton selectedRadioButton =(RadioButton) answers.getSelectedToggle();
    // statement works just if user selected option
    if(selectedRadioButton != null) {
        // get the text of chosen answer
        String toggleGroupValue = selectedRadioButton.getText();

        // Checks for identity
        if(toggleGroupValue.equals(nowCorrectAnswer)) {
            // Выводим информацию и увеличиваем количество верных ответов
            System.out.println("Верный ответ");
            correctAnswers++;
        } else
            System.out.println("Не верный ответ");

        // if user has finished questionnaire, it is time to hide all fields
        if(nowQuestion + 1 == questions.length) {
            radio_btn_1.setVisible(false);
            radio_btn_2.setVisible(false);
            radio_btn_3.setVisible(false);
            radio_btn_4.setVisible(false);
            answerBtn.setVisible(false);

            // Show the number of correctly answered questions
            question_text.setText("You have" + correctAnswers + "correct answers out of"
+ questions.length + " questions!");
        } else { // if it is not the end of questionnaire, next question is shown
            // index for next question
            nowQuestion++;
            nowCorrectAnswer = questions[nowQuestion].correctAnswer();

            // Next question is shown
            question_text.setText(questions[nowQuestion].getQuestion());
            // String array consisting answers for current questions
            String[] answers = questions[nowQuestion].getAnswers();

            // Converting it into list enables us to do quick shuffle
            List<String> intList = Arrays.asList(answers);

            // answers placed in random order
            Collections.shuffle(intList);

            // set answer to radio buttons
```

```
        radio_btn_1.setText(intList.get(0));
        radio_btn_2.setText(intList.get(1));
        radio_btn_3.setText(intList.get(2));
        radio_btn_4.setText(intList.get(3));

        selectedRadioButton.setSelected(false);
    }

}

});

}

}
```