MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
VADYM HETMAN KYIV NATIONAL ECONOMIC UNIVERSITY

Educational and Scientific Institute

"Institute of Information Technologies in Economics"

Department of Systems Analysis and Cybersecurity

Coursework

from the academic discipline

" Object -oriented programming"

on the topic: " Development of a software system for analyzing the student contingent based on requests for marital status "

Made by a student

group IA-201

Valikova Valeria Atnonivna

Checked by: Begun A.V.

Kyiv 2024

Content

Introduction 3
SECTION 1. Purpose of development 4
SECTION 2. Program Requirements 5
2.1. Requirements for functional characteristics 5
2.2. Reliability requirements 5
2.3. Operating conditions 5
2.4. Requirements for the composition of hardware parameters and software compatibility 6
CHAPTER 3. .... Object model of the problem 7
3.1. Defining objects and classes 7
3.2. Defining dependencies 7
CHAPTER 4. .... Dynamic model of the problem 8
4.1. Clarifying scenarios 8
4.2. Construction of state diagrams of objects 9
CHAPTER 5. .... Functional model of the problem 10
5.1. Defining input and output values 10
5.2. Defining optimization criteria 11
CHAPTER 6. .... Software Development 12
6.1. Creating a database 12
6.2. Description of functions 12
6.3. Creating the interface and support features 14
CHAPTER 7. .... Program Code 17
CHAPTER 8. .... Conclusion 23
CHAPTER 9. .... Sources 24

INTRODUCTION

Topic: II.3. Development of a software system for analyzing the student contingent based on requests for marital status.

Goal: create a program that helps in organizing the work of the university. The application is used to populate a database that will contain information about students, indicating their marital status.

To achieve this goal, the following tasks were set:

1. Analyze the features of the research topic using a variety of sources.
2. Research the subject area and analyze the student population by marital status.
3. Decide on the problem statement, formulate product requirements.
4. Develop an optimal database structure.
5. Develop a user-friendly interface.
6. Create a software system.
7. Evaluate the product and test it.

The system will be developed in Intellij. IDE from JetBrains .

Object of research: determining the marital status of students.

Subject of research: software product for analyzing students by marital status.

# РОЗДІЛ 1.	PURPOSE OF DEVELOPMENT

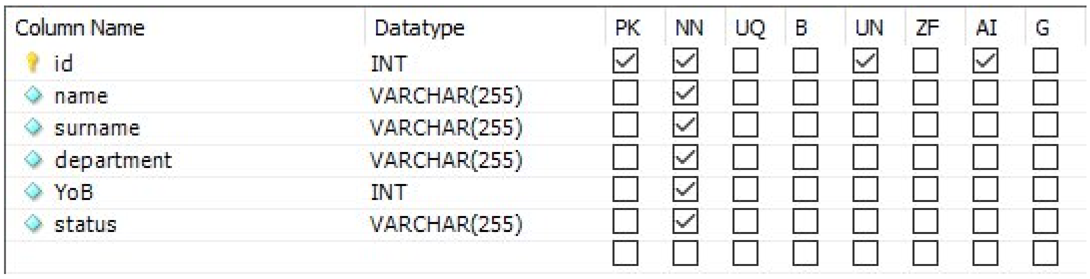
The software product is designed to facilitate the entry, editing, deletion of information about students and searching for their marital status. Access will be via a local network. Data will be taken from a local database created on a local server using MySQL Workbench .

Important achievements of this system will be a clear and accessible presentation of data, the ability to quickly operate on and access them, as well as the creation of other users (students) with their personal information.

РОЗДІЛ 2. PROGRAM REQUIREMENTS

2.1. Functional requirements

The database " student 1 " contains the following information:



Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G
id	INT	✓	✓	☐	☐	✓	☐	✓	☐
name	VARCHAR(255)	☐	✓	☐	☐	☐	☐	☐	☐
surname	VARCHAR(255)	☐	✓	☐	☐	☐	☐	☐	☐
department	VARCHAR(255)	☐	✓	☐	☐	☐	☐	☐	☐
YoB	INT	☐	✓	☐	☐	☐	☐	☐	☐
status	VARCHAR(255)	☐	✓	☐	☐	☐	☐	☐	☐
		☐	☐	☐	☐	☐	☐	☐	☐

Figure 2. 1- Database " student 1"

Where id is the identification number (filled in automatically), name is the student's name, surname is the student's last name, department is the department of study (institute), YoB is the year of birth, status is the family status.

Requirements:

1. View data in a table.
2. Adding, editing, and deleting data.
3. Search and sort capabilities .
4. Updating the database after adding and editing information.

2.2. Reliability requirements

The database system checks the entered information by checking the type of data input. The field must be entered with a valid value, which is why fields with the int data type cannot accept values of the varchar (string) or float type . If incorrect values are entered, the system will issue an error and will not enter the data into the table.

, all form fields must be filled in, as the NN (not null) , which checks the entered value for characters.

2.3. Operating conditions

The software product is intended for users who do not have extensive knowledge of the SQL language . Ensuring stable operation of the database, correcting errors and making changes to the software part of the system is the task of a system administrator or teacher with appropriate qualifications.

2.4. Requirements for the composition of hardware parameters and software compatibility

The application can run on any operating system (Windows, Linux , etc.) in the .exe file format. Also, to work, you need to have access to a database and SQL on your computer. Server .

РОЗДІЛ 3. OBJECT MODEL OF THE PROBLEM

3.1. Defining objects and classes

System classes:

“ FXMLController ” – controls all actions in the table (adding, editing, deleting, searching, sorting).

“ TableViewUser ” – displaying the table and form on the screen.

“ FXMLPart ” – a graphical part constructed using SceneBuilder .

“ student ” – sets the class constructor and assigns values to variables and establishes the ability to set and return values.

“ mySqlConnection ” – provides access to the database and values.

3.2. Defining dependencies

Depicting a dependency using a diagram

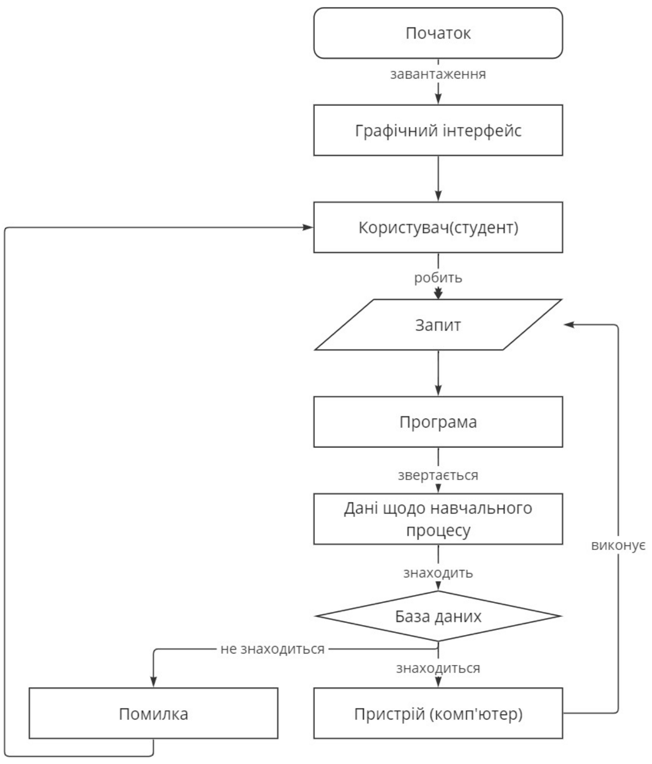


Figure 3. 1- Program dependency flowchart

РОЗДІЛ 4. PROBLEM MODEL

4.1. Clarifying scenarios

I offer to your attention one of the possible scenarios for the system's operation:

1. The user (student) launches the software system.
2. The user (student) selects the type of work (adding a new user, adjusting an already added user, deleting an existing user, searching for a user by marital status, exiting the program).
3. The user (student) enters information about the new user.
4. The user (student) adds information about the new user and automatically updates the database.
5. The user (student) sorts the table by parameters available in the table itself (identification number, first name, last name, department, year of birth, or marital status).
6. The user (student) enters a subtype of family status and searches by the family status parameter.
7. The user (student) selects the added user by right-clicking.
8. The user (student) selects the type of work (edit or delete).
9. User (student) corrects user
10. The user (student) adds the corrected data + the data table is automatically updated.
11. The user (student) selects the user.
12. User (student) deletes user + automatically updates data table.
13. The user (student) closes the software system.
14. The user (student) completes the work.

4.2. Building object state diagrams

Client-server. Data is entered into a form and immediately enters the database. Text data is transmitted and processed by the server. The user (student) initiates CRUD operations, which are performed by the server. The server processes the data received from the user (student) and updates the database.

The system uses JDBC to directly connect a JavaFX application to MySQL .

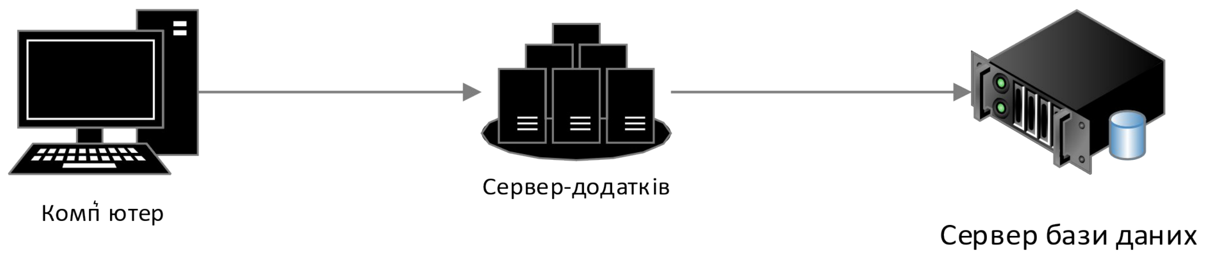


Figure 4. 1- Client-server architecture

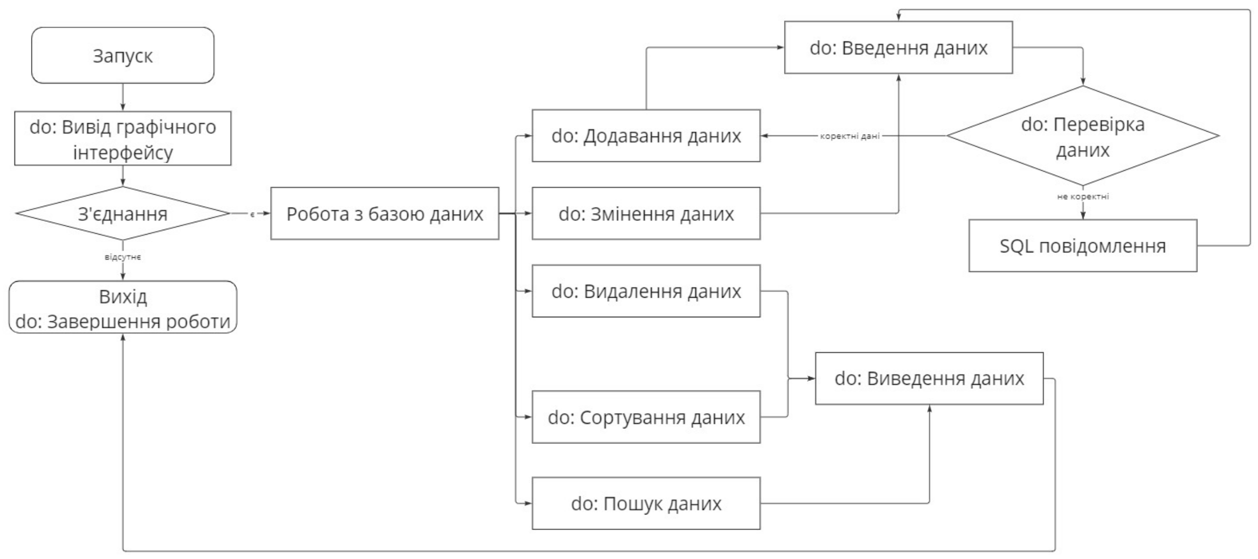


Figure 4. 2- Block diagram of software system states

РОЗДІЛ 5. FUNCTIONAL MODEL OF THE PROBLEM

5.1. Defining input and output values

Input values are queries and constraints that the user sets for the system. Output values are the result of executing these queries and the information values obtained from the database.

Building a data flow diagram:

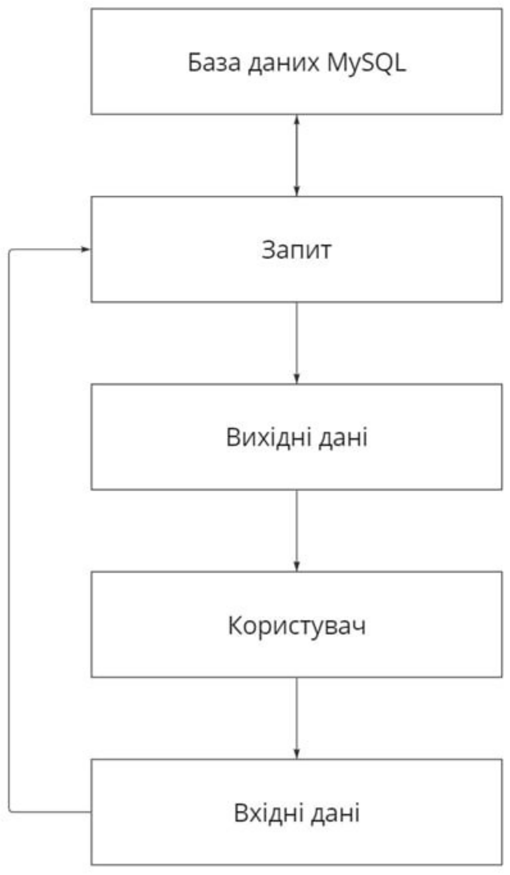


Figure 5. 1- Data flow diagram

The user enters input data (for example: first name, last name, etc.) and sends a request to add, edit, or delete data. The system reports the request (success or error) and returns a table with the entered data.

Description of restrictions

The system has the following limitations:

- Fields must be filled with data of the appropriate type. For int - numbers, for varchar - words.

- It is impossible to duplicate an identification number. Duplicate surnames and first names are possible.

- All fields must be filled in.

5.2. Defining optimization criteria

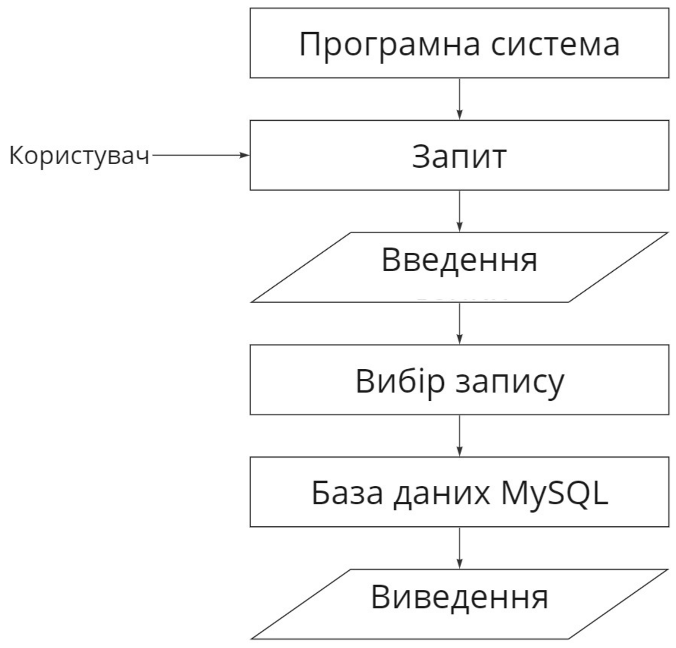


Figure 5. 2- Flowchart for the query execution process

First, the system connects to the database and performs certain manipulations specified by the program (search, edit, delete, add). After changes, the system automatically updates the database view and displays a new version after corrections.

РОЗДІЛ 6. SOFTWARE DEVELOPMENT

6.1. Creating a database

The database was created in MySQL. Workbench

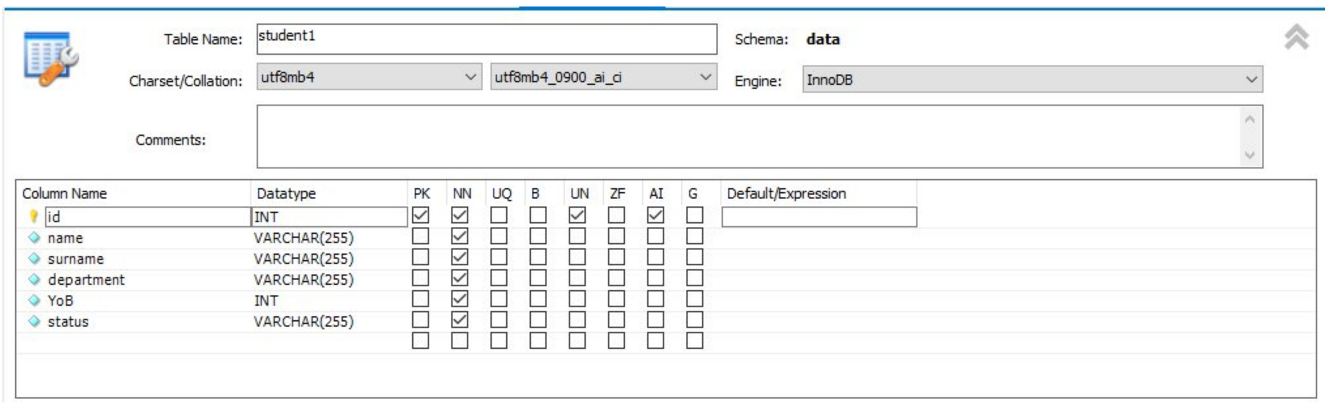


Table Name: student1
Schema: data
Charset/Collation: utf8mb4 | utf8mb4_0900_ai_ci
Engine: InnoDB
Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT	✓	✓			✓		✓		
name	VARCHAR(255)		✓							
surname	VARCHAR(255)		✓							
department	VARCHAR(255)		✓							
YoB	INT		✓							
status	VARCHAR(255)		✓							

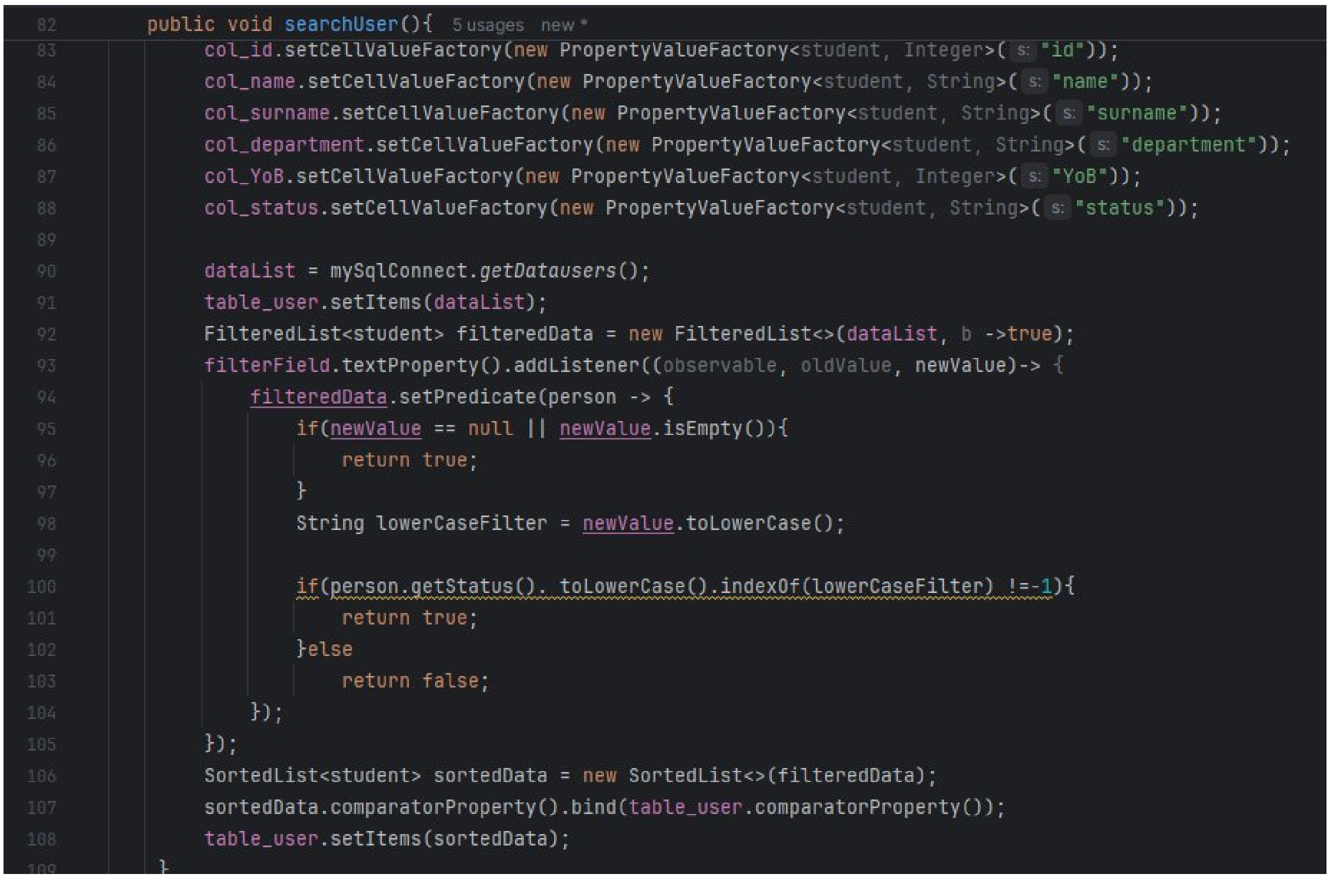
Figure 6. 1- Database

6.2. Description of functions

Using database queries and functions, the interface of the software complex is provided with the necessary information for working with the software product and analyzing data.

The following functions are implemented in the software system:

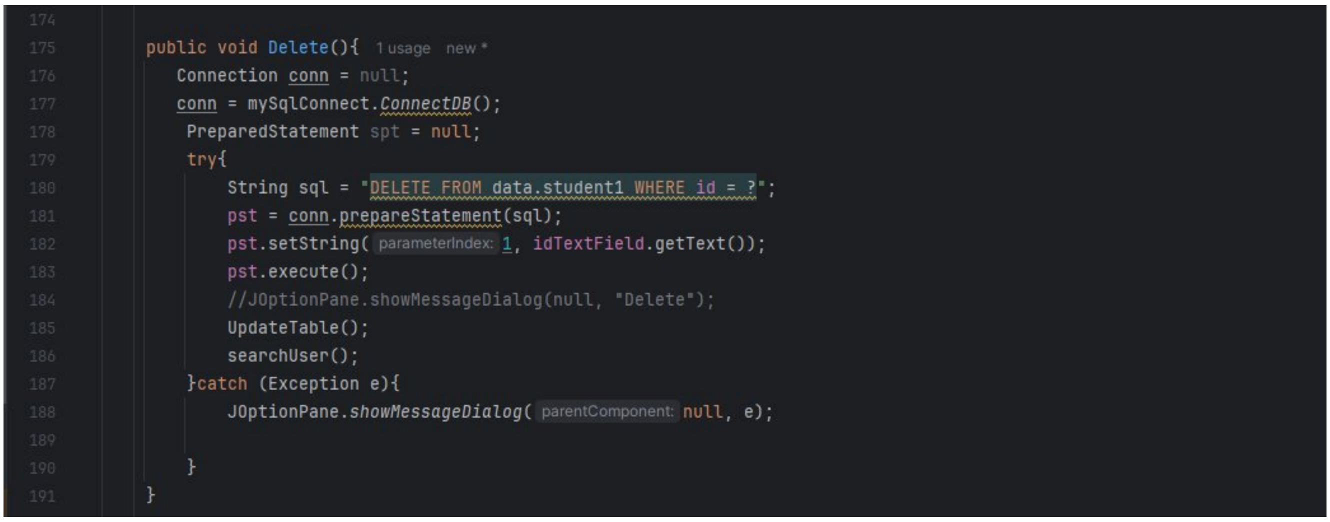
1. Search function – the user enters a keyword and searches for matching entries. There is also a lowercase recognition function for simplified input.



```java
    public void searchUser(){
        col_id.setCellValueFactory(new PropertyValueFactory<student, Integer>("id"));
        col_name.setCellValueFactory(new PropertyValueFactory<student, String>("name"));
        col_surname.setCellValueFactory(new PropertyValueFactory<student, String>("surname"));
        col_department.setCellValueFactory(new PropertyValueFactory<student, String>("department"));
        col_YoB.setCellValueFactory(new PropertyValueFactory<student, Integer>("YoB"));
        col_status.setCellValueFactory(new PropertyValueFactory<student, String>("status"));

        dataList = mySqlConnect.getDatausers();
        table_user.setItems(dataList);
        FilteredList<student> filteredData = new FilteredList<>(dataList, b ->true);
        filterField.textProperty().addListener((observable, oldValue, newValue)-> {
            filteredData.setPredicate(person -> {
                if(newValue == null || newValue.isEmpty()){
                    return true;
                }
                String lowerCaseFilter = newValue.toLowerCase();

                if(person.getStatus(). toLowerCase().indexOf(lowerCaseFilter) !=-1){
                    return true;
                }else
                    return false;
            });
        });
        SortedList<student> sortedData = new SortedList<>(filteredData);
        sortedData.comparatorProperty().bind(table_user.comparatorProperty());
        table_user.setItems(sortedData);
    }
```

Figure 6. 2- Search function

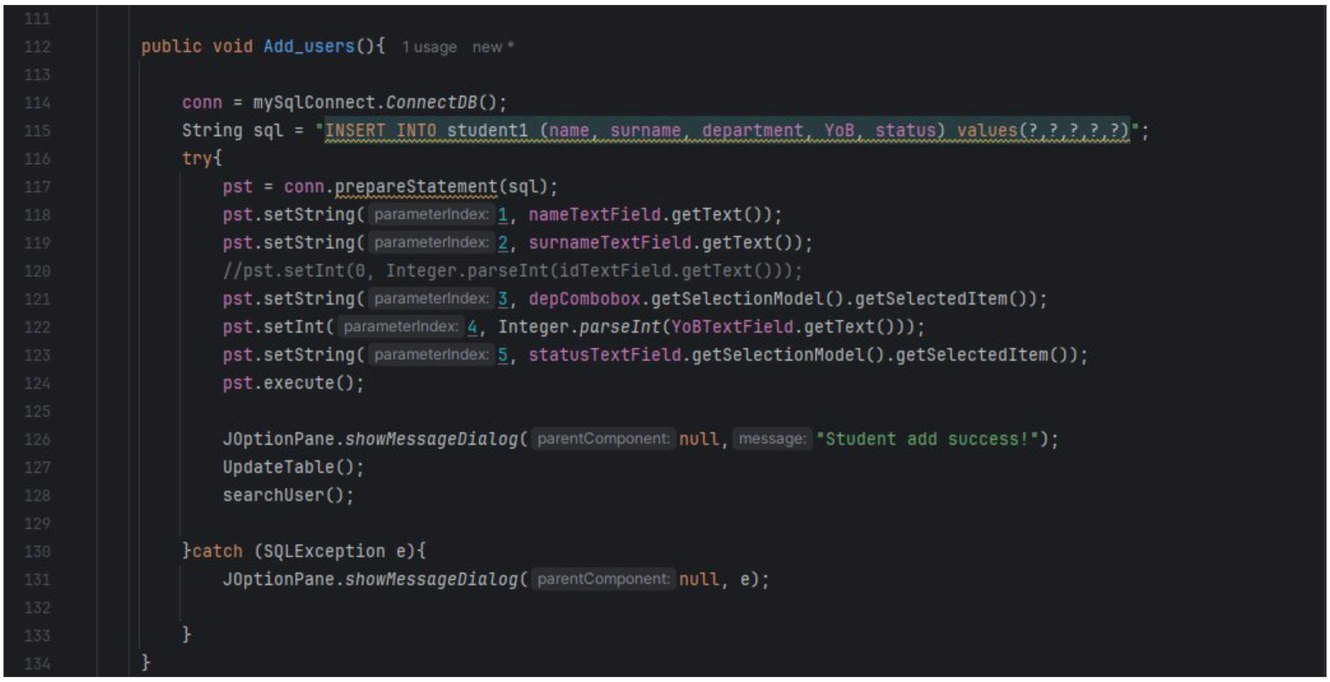
2. Deleting data – the user selects the row with the record to be deleted and deletes it completely.



```java
    public void Delete(){
        Connection conn = null;
        conn = mySqlConnect.ConnectDB();
         PreparedStatement spt = null;
         try{
             String sql = "DELETE FROM data.student1 WHERE id = ?";
             pst = conn.prepareStatement(sql);
             pst.setString(1, idTextField.getText());
             pst.execute();
             //JOptionPane.showMessageDialog(null, "Delete");
             UpdateTable();
             searchUser();
         }catch (Exception e){
             JOptionPane.showMessageDialog(null, e);

         }
    }
```

Figure 6. 3- Delete function

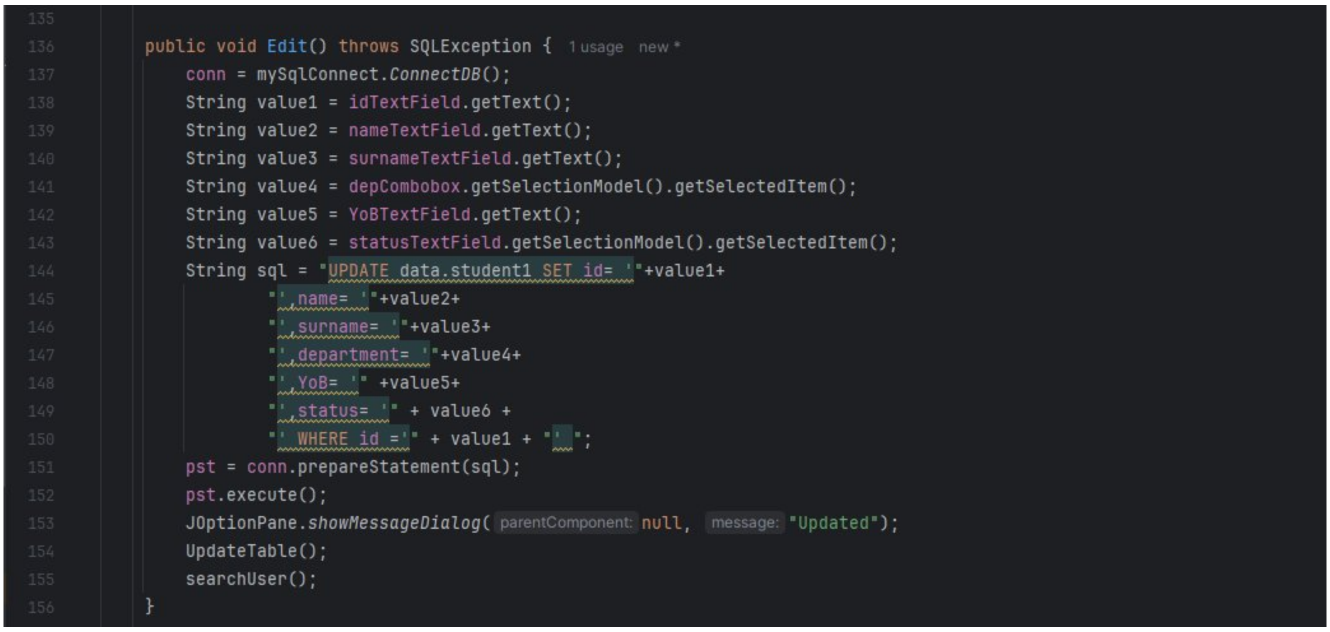
3. Adding data – the user enters new data; if all fields are filled in and the entered data matches the field types, it is added to the database.



```java

    public void Add_users(){ 1 usage  new *

        conn = mySqlConnect.ConnectDB();
        String sql = "INSERT INTO student1 (name, surname, department, YoB, status) values(?,?,?,?,?)";
        try{
            pst = conn.prepareStatement(sql);
            pst.setString( parameterIndex: 1, nameTextField.getText());
            pst.setString( parameterIndex: 2, surnameTextField.getText());
            //pst.setInt(0, Integer.parseInt(idTextField.getText()));
            pst.setString( parameterIndex: 3, depCombobox.getSelectionModel().getSelectedItem());
            pst.setInt( parameterIndex: 4, Integer.parseInt(YoBTextField.getText()));
            pst.setString( parameterIndex: 5, statusTextField.getSelectionModel().getSelectedItem());
            pst.execute();

            JOptionPane.showMessageDialog( parentComponent: null, message: "Student add success!");
            UpdateTable();
            searchUser();

        }catch (SQLException e){
            JOptionPane.showMessageDialog( parentComponent: null, e);

        }
    }
```

Figure 6. 4- User add function

4. Editing data – the user selects the row with the record that needs to be changed, updates the necessary data, and saves the changes to the database.

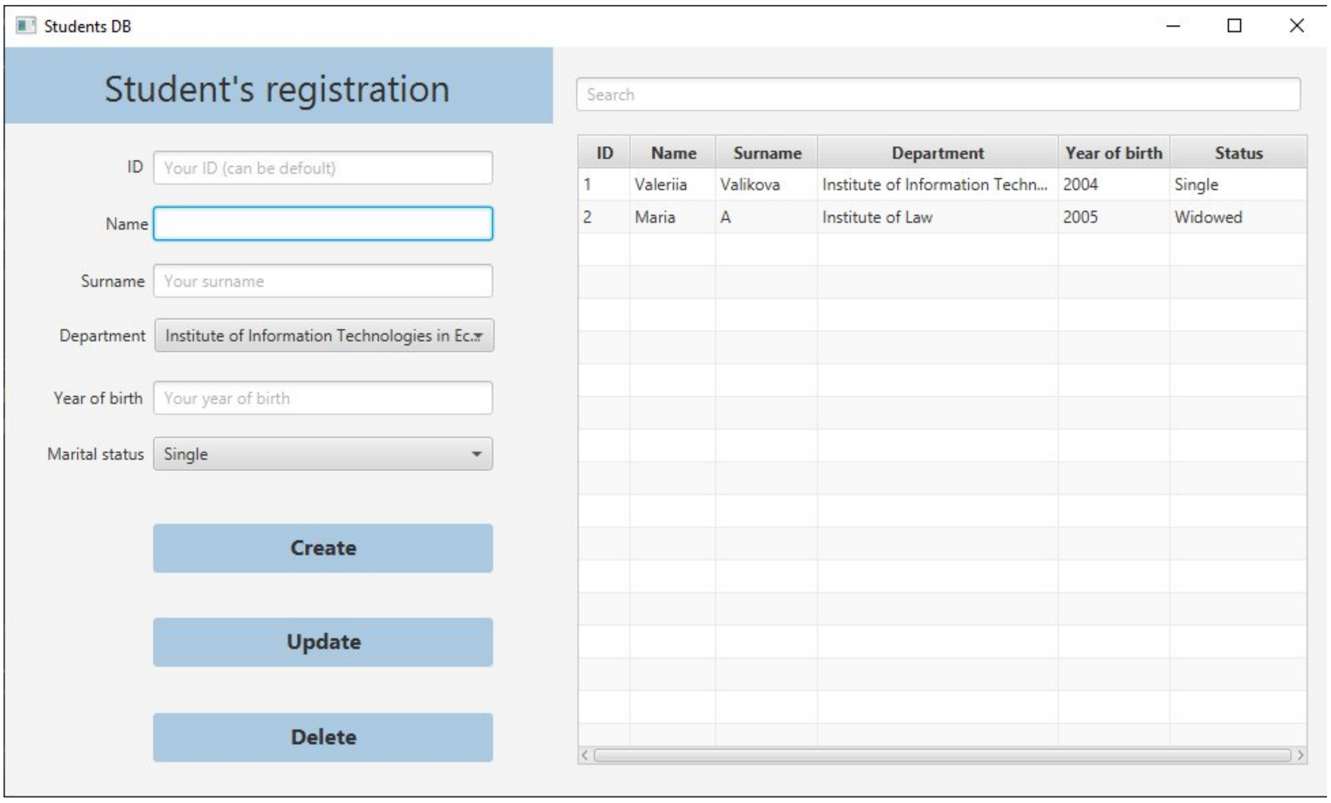


```java

    public void Edit() throws SQLException { 1 usage  new *
        conn = mySqlConnect.ConnectDB();
        String value1 = idTextField.getText();
        String value2 = nameTextField.getText();
        String value3 = surnameTextField.getText();
        String value4 = depCombobox.getSelectionModel().getSelectedItem();
        String value5 = YoBTextField.getText();
        String value6 = statusTextField.getSelectionModel().getSelectedItem();
        String sql = "UPDATE data.student1 SET id= '"+value1+
                "',name= '"+value2+
                "',surname= '"+value3+
                "',department= '"+value4+
                "',YoB= '" +value5+
                "',status= '" + value6 +
                "' WHERE id ='" + value1 + "' ";
        pst = conn.prepareStatement(sql);
        pst.execute();
        JOptionPane.showMessageDialog( parentComponent: null,  message: "Updated");
        UpdateTable();
        searchUser();
    }
```

Figure 6. 5- Editing function

6.3. Creating the interface and support features

The software system I present consists of one window, but logically divided into two parts. The left part is a form for entering data, editing and deleting. The right part is a table display with a possible search for a user (student) by status.



ID	Your ID (can be defoult)
Name	
Surname	Your surname
Department	Institute of Information Technologies in Ec...
Year of birth	Your year of birth
Marital status	Single

Search

ID	Name	Surname	Department	Year of birth	Status
1	Valeriia	Valikova	Institute of Information Techn...	2004	Single
2	Maria	A	Institute of Law	2005	Widowed

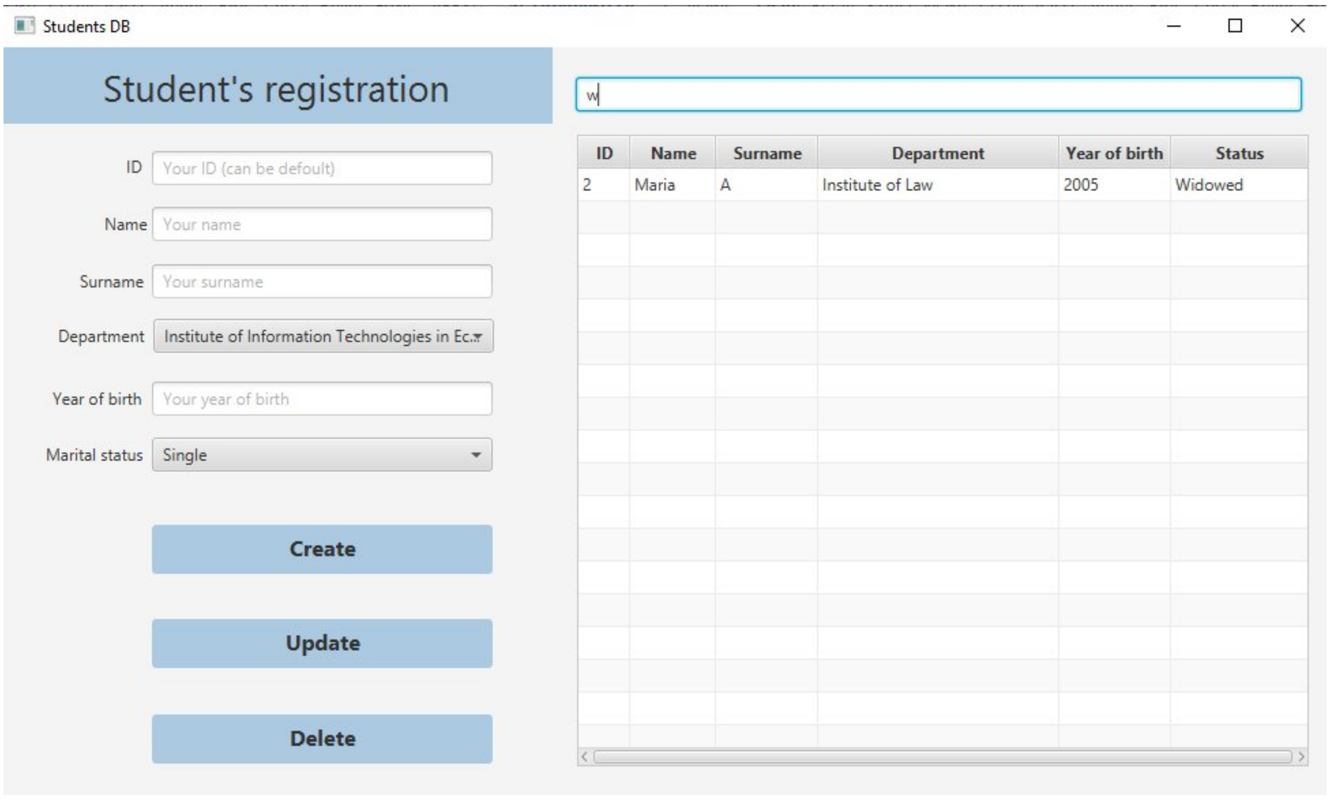
Figure 6. 6- Main program interface

First, the user can enter the new user's details and add them to the table.

If the data is entered incorrectly, an error will be raised with the reason. In other cases, the data is added to the table and the code causes an automatic update.

To delete a user, you need to select a row, all data will be displayed on the right side of the form, and click “ Delete ”. After that, the code will also cause the table to be updated.

If the user wants to find only students with a certain family status, then they need to start entering the query into the line and the necessary lines will gradually be displayed from the first letters.



ID	Name	Surname	Department	Year of birth	Status
2	Maria	A	Institute of Law	2005	Widowed

Figure 6. 7- Example of a search function by marital status

РОЗДІЛ 7. PROGRAM CODE

This section demonstrates the most important parts of the program code.

1. MySQL database and returns a list of students from the student1 table.

```
public class mySqlConnect {
  Connection conn = null ;

  public static java.sql.Connection ConnectDB (){
    try {
      Class.forName (" com.mysql.cj.jdbc.Driver ");
      Connection conn = DriverManager.getConnection ("
jdbc:mysql ://localhost:3306/ data ", " root ", "************");
// JOptionPane.showMessageDialog ( null , " Connection " Established ");
      return conn ;

} catch ( Exception e){
      JOptionPane.showMessageDialog ( null , e);
      return null ;
}
}

  public static ObservableList < student > getDatausers (){
    Connection conn = ConnectDB ();
    ObservableList < student > list = FXCollections.observableArrayList ();
    try {
      PreparedStatement pst = conn.prepareStatement ("SELECT * FROM
student1");
      ResultSet rs = pst.executeQuery ();

      while ( rs.next ()){
        list.add ( new student (
                    rs.getInt (" id "),
                    rs.getString (" name "),
                    rs.getString (" surname "),
                    rs.getString (" department "),
                    rs.getInt (" YoB "),
                    rs.getString (" status ")
        )
        );
        }
        } catch ( Exception e){

        }
            return list ;
        }
```

2. Code that loads the UI from an FXML file and displays it in a window named " Students DB"

```
        public class TableViewUser extends Application {
        @Override
          public void start ( Stage stage ) throws Exception {

            Parent root =
FXMLLoader.load(TableViewUser.class.getResource("FXMLPart.fxml"));
            Scene scene = new Scene ( root );
            stage.setTitle (" Students DB");
            stage.setScene ( scene );
            stage.show ();
        }

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

3. Code to remove users from the list

```
public void Delete (){
      Connection conn = null ;
      conn = mySqlConnect.ConnectDB ();
       PreparedStatement spt = null ;
       try {
          String sql = "DELETE FROM data.student1 WHERE id = ?";
          pst = conn. prepareStatement ( sql );
          pst.setString (1, idTextField.getText ());
          pst.execute ();
// JOptionPane.showMessageDialog ( null , " Delete ");
          UpdateTable ();
          searchUser ();
} catch ( Exception e){
          JOptionPane.showMessageDialog ( null , e);


}
}
```

4. Code for adding users from a list

```
public void Add_users (){


       conn = mySqlConnect.ConnectDB ();
       String sql = "INSERT INTO student1 ( name , surname , department , YoB ,
status ) values (?,?,?,?,?)";
       try {
          pst = conn. prepareStatement ( sql );
          pst.setString (1, nameTextField.getText ());
          pst.setString (2, surnameTextField.getText ());
// pst.setInt (0, Integer.parseInt ( idTextField.getText ()));
```

```
                pst.setString (3, depCombobox.getSelectionModel (). getSelectedItem ());
                pst.setInt (4, Integer.parseInt ( YoBTextField.getText ()));
                pst.setString (5, statusTextField.getSelectionModel (). getSelectedItem
());
                pst.execute ();

                JOptionPane.showMessageDialog ( null ," Student add success !");
                UpdateTable ();
                searchUser ();

        } catch ( SQLException e){
                JOptionPane.showMessageDialog ( null , e);

        }
        }
```

5. Code for editing users from the list

```
public void Edit () throws SQLException {
        conn = mySqlConnect.ConnectDB ();
        String value1 = idTextField.getText ();
        String value2 = nameTextField.getText ();
        String value3 = surnameTextField.getText ();
        String value4 = depCombobox.getSelectionModel (). getSelectedItem ();
        String value5 = YoBTextField.getText ();
        String value6 = statusTextField.getSelectionModel (). getSelectedItem ();
        String sql = "UPDATE data.student1 SET id = '"+value1+
"', name = '"+value2+
"', surname = '"+value3+
"', department = '"+value4+
"', YoB = '" +value5+
"', status = '" + value6 +
''' WHERE id ='" + value1 + "' ";
        pst = conn. prepareStatement ( sql );
        pst.execute ();
        JOptionPane.showMessageDialog ( null , " Updated ");
        UpdateTable ();
        searchUser ();
}
```

6. Code to select users from a list and display data in specific form cells

```
@FXML
    void getSelected ( MouseEvent event ) {
        index = table_user.getSelectionModel (). getSelectedIndex ();
        if ( index <=-1){
            return ;
}
        conn = mySqlConnect.ConnectDB ();
        idTextField.setText ( col_id.getCellData ( index ). toString ());
        nameTextField.setText ( col_name.getCellData ( index ));
        surnameTextField.setText ( col_surname.getCellData ( index ));
depCombobox.getSelectionModel().select(col_department.getCellData(index));
        YoBTextField.setText ( col_YoB.getCellData ( index ). toString ());
statusTextField.getSelectionModel().select(col_status.getCellData(index));
}
```

7. Search code

```
public void searchUser ( ){
        col_ id.setCellValueFactory (new PropertyValueFactory <student,
Integer>("id"));
        col_ name. setCellValueFactory (new PropertyValueFactory <student,
String>("name"));
        col_ surname.setCellValueFactory (new PropertyValueFactory <student,
String>("surname"));
```

```
            col_ department.setCellValueFactory (new PropertyValueFactory <student,
String>("department"));
            col_YoB.setCellValueFactory (new PropertyValueFactory <student,
Integer>(" YoB "));
            col_ status.setCellValueFactory (new PropertyValueFactory <student,
String>("status"));

            dataList = mySqlConnect.getDatausers ();
            table_ user.setItems ( dataList );
            FilteredList <student> filteredData = new FilteredList < >( dataList , b
->true);
            filterField.textProperty ( ). addListener ((observable, oldValue ,
newValue )-> {
              filteredData.setPredicate (person -> {
                if( newValue == null || newValue.isEmpty ()){
       return true;
       }
       String lowerCaseFilter = newValue.toLowerCase ();

       if( person.getStatus (). toLowerCase ( ). indexOf ( lowerCaseFilter ) !=-1){
       return true;
                }else
       return false;
       });
       });
            SortedList <student> sortedData = new SortedList <>( filteredData );
       sortedData.comparatorProperty( ).bind (table_user.comparatorProperty());
            table_ user.setItems ( sortedData );
       }
```

РОЗДІЛ 8. CONCLUSION

This project considered the development of a software system based on the example of the Vadym Hetman Kyiv National Economic University. The main focus was on creating an electronic system to monitor the marital status of teachers. The following tasks were completed:

1. The necessary literature on the development of relational database systems was studied, including the analysis of programs with similar logic that are available in the open access.
2. The design of a relational database at the logical and physical levels has been completed.
3. An application has been created that works flawlessly with the database.

This program greatly simplifies the work of organizing and monitoring teachers' marital status, as well as other necessary information. It reduces the time spent searching for teachers and their personal information, helping to organize teacher data in one place with a convenient and understandable interface.

The goal of the coursework was achieved successfully.

РОЗДІЛ 9. SOURCES

1. Shvachych G.G., V.M. Pasynkov , G.A. Pavlenko and others. Construction of block diagrams: Textbook . – Dnipropetrovsk: NMetAU , 2004. – 24 p.
2. JDBC Documentation . *Software Download | Oracle* . URL: https://download.oracle.com/otn_hosted_doc/jdeveloper/904preview/jdk14doc/docs/guide/jdbc/index.html .
3. JavaFX . *Oracle | Cloud Applications and Cloud Platform* . URL: https://www.oracle.com/java/technologies/javase/javafx-overview.html .
4. Java Documentation - Get Started . *Oracle Help Center* . URL: https://docs.oracle.com/en/java/ .
5. javax.swing (Java Platform SE 8). *Moved* . URL: https://docs.oracle.com/javase/8/docs/api/javax/swing/package-summary.html .
6. tookookotek . JavaFX Tutorial | Search Bar and TableView filter result as you search , 2021. *YouTube* . URL: https://www.youtube.com/watch?v=2M0L6w3tMOY .
7. JavaFX Scene Builder User Guide : About This User Guide | JavaFX 2 Tutorials and Documentation . Moved . URL: https://docs.oracle.com/javafx/scenebuilder/1/user_guide/jsbpub-user_guide.htm .