

ID Zadania: **LAB\_3**

Tytuł zadania: **Implementacja modułu aplikacji bazodanowej**

Termin oddania: **08.06.2020**

Temat: **Dowolny, wybrany przez studenta**

# Patryk KANIEWSKI i Wojciech ZŁOMEK



## Podstawy baz danych

### Treść zadania

Zadanie polega na zaimplementowaniu modułu aplikacji bazodanowej umożliwiającego za pośrednictwem graficznego interfejsu użytkownika:

- *na ocenę dostateczną (3 – dst):*
  - o wyświetlać wszystkie rekordy z dowolnej tabeli relacyjnej bazy danych;
  - o przeprowadzić filtrację pod kątem wybranego atrybutu i wyświetlić wyniki;
- *na ocenę dobrą (4 – db)\*:*
  - o dodawanie nowych rekordów do wybranej tabeli;
  - o modyfikację obecnych rekordów wybranej tabeli;;
- *na ocenę bardzo dobrą (5 – bdb)\*:*
  - o usuwanie rekordów z bazy danych.

**\*wymaga zrealizowania poprzednich podpunktów**

## Przykładowa implementacja modułu aplikacji bazodanowej w języku Java

W niniejszej sekcji opisano proces implementacji szkieletu modułu aplikacji bazodanowej w języku Java wykorzystującego interfejs JDBC oraz biblioteki Swing i AWT.

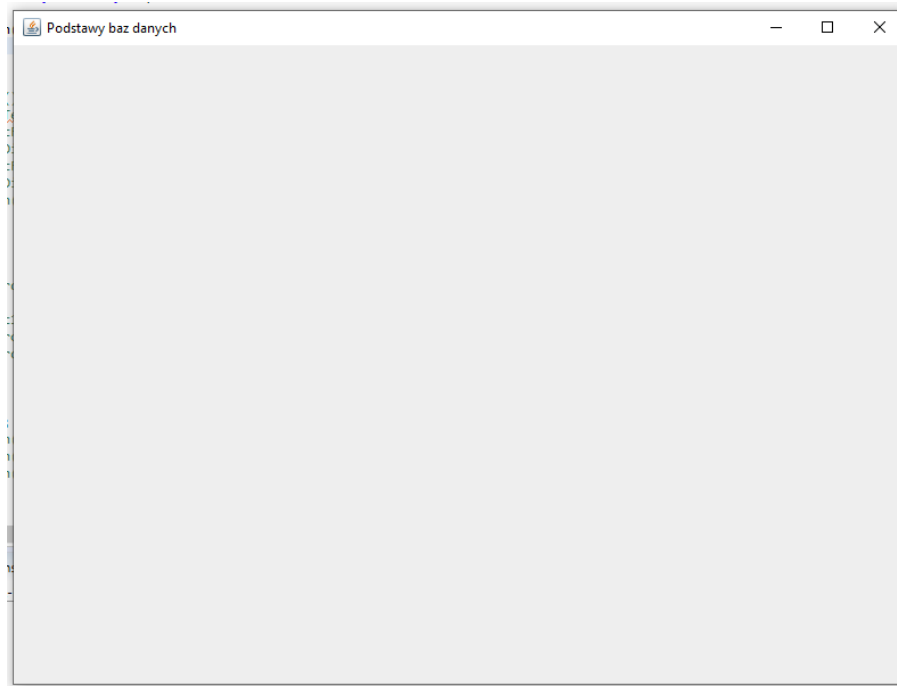
### Implementacja graficznego interfejsu użytkownika

Pierwszym krokiem jest utworzenie głównego okna, co realizuje kod zamieszczony na listingu 1.

```
JFrame f = new JFrame("Podstawy baz danych"); //utworzenie obiektu klasy
JFrame ("okna") o nazwie "Podstawy baz danych"
f.setSize(800, 600); //ustawienie rozmiaru tworzonego "okna" (opcjonalne)
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //aby po wciśnięciu
przycisku zamknięcia, aplikacja została "faktycznie" zamknięta koniecznym
jest ustawienie domyślnego zachowania operacji zamknięcia
//- ustawienie wartości EXIT_ON_CLOSE spowoduje, że po naciśnięciu
przycisku "X" nastąpi wyjście z aplikacji
f.setVisible(true); //wyświetlenie utworzonego "okna" (metodę
setVisible(true) należy wywołać po przygotowaniu całego okna – należy o tym
pamiętać przy dodawaniu wielu komponentów)
```

*Listing 1 Kod tworzący nową ramkę o podanej nazwie i określonym rozmiarze*

Na rysunku 1 przedstawiono efekt wykonania powyższego kodu.



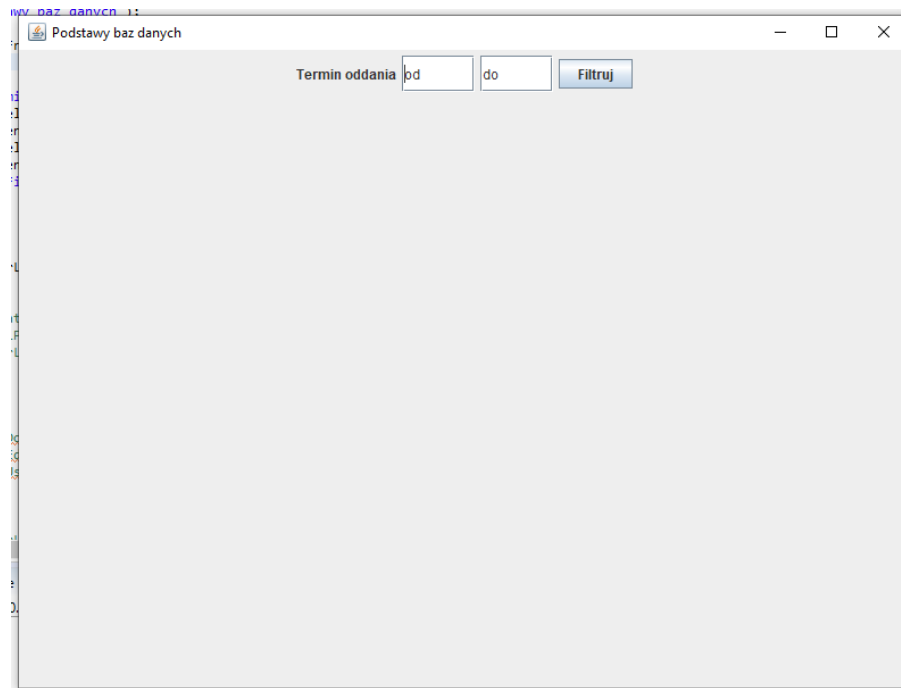
*Rysunek 1 Okno modułu aplikacji, będące efektem wykonania kodu zamieszczonego na listingu 1*

Posiadając gotowe okno możliwe jest przystąpienie do umieszczania w nim potrzebnych elementów interfejsu graficznego, na listingu 2 przedstawiono kod odpowiedzialny za utworzenie przykładowego panelu umożliwiającego filtrację danych:

```
JPanel jp1 = new JPanel(); //utworzenie panelu, który zostanie
wykorzystany jako kontener dla komponentów modułu filtracji
JLabel jl1 = new JLabel("Termin oddania"); //utworzenie etykiety z
tekstem "Termin oddania" (przykład)
JTextField tf1 = new JTextField("od"); //utworzenie pola tekstowego z
domyślną zawartością "od" (przykład)
tf1.setPreferredSize(new Dimension(64, 32)); //ustawienie preferowanego
rozmiaru utworzonego pola tekstowego (szerokość, wysokość)
JTextField tf2 = new JTextField("do");
tf2.setPreferredSize(new Dimension(64, 32));
JButton btn1 = new JButton("Filtruj"); //utworzenie przycisku z napisem
"Filtruj"
jp1.add(jl1); //dodanie etykiety jl1 do panelu jp1
jp1.add(tf1);
jp1.add(tf2);
jp1.add(btn1);
f.getContentPane().add(BorderLayout.NORTH, jp1); //dodanie panelu do okna
w ustalonym szablonie - opcja NORTH umieści panel w górnej części okna
```

*Listing 2 Kod odpowiedzialny za utworzenie przykładowego panelu filtracji danych*

Na rysunku 2 przedstawiono okno aplikacji z dodanym przykładowym panelem filtracji danych.



*Rysunek 2 Okno modułu aplikacji, będące efektem wykonania kodu zamieszczonego na listingach 1-2*

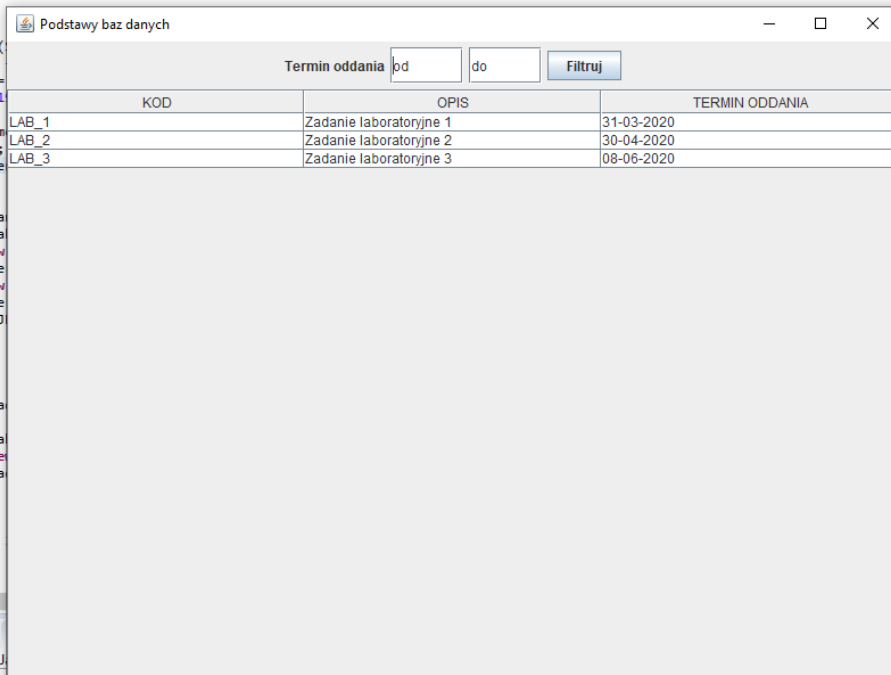
Przedstawione powyżej rozwiązanie nie jest jedynym możliwym. Innym podejściem jest wykorzystanie elementów *JComboBox*, czyli listy rozwijanej zawierającej dostępne wartości filtra.

Kolejnym krokiem w tworzeniu przedmiotowego modułu aplikacji bazodanowej jest dodanie do okna tabeli, w której będą wyświetlane dane znajdujące się w bazie. Kod odpowiedzialny za dodanie do graficznego interfejsu przykładowej tabeli zamieszczono na listingu 3:

```
String jt1Attrs[] = {"KOD", "OPIS", "TERMIN ODDANIA"};
String jt1Data[][] = { {"LAB_1", "Zadanie laboratoryjne 1", "31-03-2020"},
{"LAB_2", "Zadanie laboratoryjne 2", "30-04-2020"}, {"LAB_3", "Zadanie
laboratoryjne 3", "08-06-2020"} };
JTable jt1 = new JTable(jt1Data, jt1Attrs); //utworzenie tabeli, jako
parametr 1. przekazywana jest zawartość dwuwymiarowej tablicy jt1Data,
która będzie stanowiła źródło danych tworzonej tabeli, natomiast jako
parametr 2. przekazywane są wartości tablicy jt1Attrs, które będą stanowiły
nagłówki kolumn tworzonej tabeli
JScrollPane sp1 = new JScrollPane(jt1); //utworzenie obiektu JScrollPane
celem wyświetlenia nagłówków kolumn tabeli
f.getContentPane().add(BorderLayout.CENTER, sp1); //dodanie panelu do
okna w ustalonym szablonie - opcja CENTER umieści panel w środkowej części
okna
```

*Listing 3 Kod odpowiedzialny za utworzenie tabeli*

Na rysunku 3 przedstawiono okno aplikacji z tabelą uzupełnioną przykładowymi danymi.



KOD	OPIS	TERMIN ODDANIA
LAB_1	Zadanie laboratoryjne 1	31-03-2020
LAB_2	Zadanie laboratoryjne 2	30-04-2020
LAB_3	Zadanie laboratoryjne 3	08-06-2020

*Rysunek 3 Okno modułu aplikacji, będące efektem wykonania kodu zamieszczonego na listingach 1-3*

Ostatnim elementem potrzebnym do zapewnienia założonych funkcjonalności modułu jest panel zawierający przyciski dodawania, edycji i usuwania rekordów z bazy danych. Kod odpowiedzialny za dodanie do GUI wspomnianych komponentów zamieszczono na listingu 4:

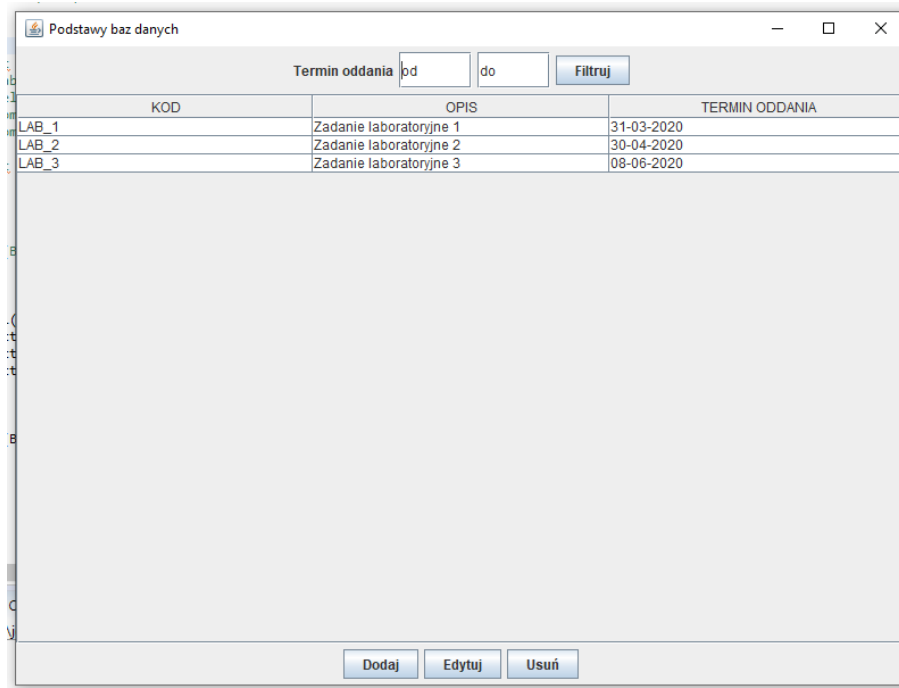
```

JPanel jp2 = new JPanel();
JButton btn2 = new JButton("Dodaj");
JButton btn3 = new JButton("Edytuj");
JButton btn4 = new JButton("Usuń");
jp2.add(btn2);
jp2.add(btn3);
jp2.add(btn4);
f.getContentPane().add(BorderLayout.SOUTH, jp2); //dodanie panelu do okna
w ustalonym szablonie - opcja SOUTH umieści panel w dolnej części okna

```

*Listing 4 Kod odpowiedzialny za utworzenie panelu z przyciskami*

Na rysunku 4 przedstawiono przykładowe okno ze wszystkimi niezbędnymi komponentami do obsługi założonych funkcjonalności implementowanego modułu aplikacji bazodanowej.



KOD	OPIS	TERMIN ODDANIA
LAB_1	Zadanie laboratoryjne 1	31-03-2020
LAB_2	Zadanie laboratoryjne 2	30-04-2020
LAB_3	Zadanie laboratoryjne 3	08-06-2020

Rysunek 4 Okno modułu aplikacji, będące efektem wykonania kodu zamieszczonego na listingach 1-4

## Implementacja i testowanie połączenia z bazą danych

Połączenie z bazą danych w implementowanym module aplikacji zrealizowano za pośrednictwem interfejsu JDBC, kod odpowiedzialny za nawiązanie komunikacji z przykładową bazą danych zamieszczono na listingu 5:

```
Connection c = null;
Statement s = null;

try {
    c = DriverManager.getConnection("jdbc:sqlite:C:/...ścieżka do bazy
danych"); //sqlite wskazuje RDBMS, w którym baza, z którą zostanie
nawiązane połączenie, została utworzona (należy dostosować)
    if (c.isValid(0)) System.out.println("Połączono z bazą danych!");
    else System.out.println("Brak połączenia z baza danych!");
    c.close();
} catch (SQLException e) {
    e.printStackTrace();
}
```

Listing 5 Kod realizujący połączenie z przykładową bazą danych

## **Wariant na ocenę dostateczną (3 - *dst*)**



**Wariant na ocenę dobrą (4 - db)**

## Wariant na ocenę bardzo dobrą (5 - bdb)

```
//ten program znajduje się w $CWD/src/bazylab3/*.java

package bazylab3;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Scanner;

public class Main
{
    public static String ip = "192.168.0.97";

    public static void main(String[] args)
    {
        int i = 1;

        if (i == 1)
        {
            GUI gui = new GUI();
        } else
        {
            Login user1 = new Login("admin",
"pwsz");
            Mysql db1 = new Mysql(user1, ip,
3306);
            String test[][];
            String test2[];
            String test3[];
            String test4[];
            test = db1.select("select * from
filmy;");
            test2 = db1.getColumns("select *
from filmy;");
            test3 = db1.getTables();
            test4 = db1.getColumnTypes("select
* from filmy;");
            printArray(test2);
            printArray(test4);
            System.out.print("\n");
            printArrayArray(test);
            System.out.print("\n");
            printArray(test3);
            db1.close();
        }
    }

    public static void printArrayArray(String[][] tablica)
    {
        // evil copypaste magic
https://stackoverflow.com/a/46018033
        System.out.println(
Arrays.deepToString(tablica).replace("], ", "]\n").replace("[[",
"[").replace("]]", "]]"));
    }
}
```

```
    }

    public static void printArray(String[] tablica)
    {
System.out.println(Arrays.deepToString(tablica));
    }

}

package bazylab3;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.*;

public class Login
{
    // to tylko struct ktory bedzie przechowywał użytkownika
    String username;
    String password;
    String ip;
    int port;
    Boolean gonext = false;

    static JFrame okno = null;

    public Login(String username, String password, String ip,
int port)
    {
        this.username = username;
        this.password = password;
        this.ip = ip;
        this.port = port;
    }

    public Login()
    {
        okno = new JFrame("Połącz sie do bazy
MySQL");
        okno.setSize(1024, 780);

        JPanel jp2 = new JPanel();
        JButton jbconnect = new JButton("polacz");

        final JTextField tfip = new
JTextField("192.168.0.97");
        final JTextField tfport = new
JTextField("3306");
        final JTextField tfuser = new
JTextField("admin");
        final JTextField tfpass = new
JTextField("pwsz");

        jp2.add(tfip);
        jp2.add(tfport);
        jp2.add(tfuser);
        jp2.add(tfpass);
        jp2.add(jbconnect);
    }
}
```

```
        okno.getContentPane().add(jp2);

        okno.setVisible(true);

        jconnect.addActionListener(new
ActionListener()
        {
            public void
actionPerformed(ActionEvent evt)
            {
                ip =
                port =
                username =
                password =
                gonext =
                true;

                System.out.print(ip + port + username + password);
            }
        });
        //TODO czekac na to az uzytkownik kliknie
        zanim jechac dalej
        while(!gonext)
        {
        }
    }

package bazylab3;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Arrays;
import java.util.List;

public class Mysql
{
    Connection baza;
    Login user;
    String url;

    public Mysql(Login login)
    {
        this.user = login;
        this.url = "jdbc:mysql://" + user.ip + ":"
+ user.port + "/lab3?serverTimezone=UTC";
        System.out.print("probuje: " + url + "\n");
        connect();
    }
}
```

```
        public void connect()
        {
            try
            {
                baza =
                DriverManager.getConnection(url, user.username, user.password);
            } catch (SQLException ex)
            {
                debug(ex);
            }
            System.out.print("połączono z baza\n");
        }

        public String[] getColumns(String query)
        {
            Statement zapytanie;
            try
            {
                zapytanie = baza.createStatement();
                ResultSet wynik =
                zapytanie.executeQuery(query);
                ResultSetMetaData wynikMeta =
                wynik.getMetaData();
                int kolumny =
                wynikMeta.getColumnCount();
                String output[] = new
                String[kolumny];
                for (int i = 1; i <= kolumny; i++)
                {
                    wynikMeta.getColumnName(i);
                    output[i - 1] =
                }
                return output;
            } catch (SQLException ex)
            {
                debug(ex);
            }
            return null; // just in case
        }

        public String[] getColumnTypes(String query)
        {
            Statement zapytanie;
            try
            {
                zapytanie = baza.createStatement();
                ResultSet wynik =
                zapytanie.executeQuery(query);
                ResultSetMetaData wynikMeta =
                wynik.getMetaData();
                int kolumny =
                wynikMeta.getColumnCount();
                String output[] = new
                String[kolumny];
                for (int i = 1; i <= kolumny; i++)
                {
```

```
        output[i - 1] =
wynikMeta.getColumnClassName(i);
    }
    } catch (SQLException ex)
    {
        debug(ex);
    }

    return null; // just in case
}

public String[] getTables()
{
    String temp[][] = select("show tables;");
    String kolumna[] = new String[temp.length];
    for (int i = 0; i < kolumna.length; i++)
    {
        kolumna[i] = temp[i][0];
    }
    return kolumna;
}

public String[][] select(String query)
{
    try
    {
        PreparedStatement zapytanie =
baza.prepareStatement(query, ResultSet.TYPE_SCROLL_SENSITIVE,
ResultSet.CONCUR_UPDATABLE);

        ResultSet wynik =
        ResultSetMetaData wynikMeta =
        int kolumny =
        int n = 0;

        // tutaj jest hack zeby dostac
        // liczbe rekordow zwrzonych, w sumie to nie wiem
        // czy da sie lepiej zrobic
        while (wynik.next())
        {
            n++;
        }
        while (wynik.previous())
        {
            assert true; // NOOP
        }

        String output[][] = new String[n]
        [kolumny];

        n = 0;
        while (wynik.next())
        {
            for (int i = 1; i <=
            kolumny; i++)
            {
                String pole =
                wynik.getString(i);

                output[n][i - 1] =
                pole;
            }
            n++;
        }
    }
    catch (SQLException ex)
    {
        debug(ex);
    }
    return output;
}
```

```
        }
        n++;
    }
    return output;
} catch (SQLException ex)
{
    debug(ex);
}
return null; // just incase
}

public String arrayToString(String array[])
{
    String output = new String();
    for (int i = 0; i < array.length; i++)
    {
        if (i != 0)
            output = output + ",";
        output = output + "\"" + array[i] +
"\\";
    }
    //System.out.print("array :\n" + output +
"\n");
    return output;
}

public String objectToString(List<Object> object)
{
    String output = new String();
    output = object.toString().replace("[",
").replace("]", "");
    return output;
}

public Boolean insertInto(String tabela, String into[],
String[] values)
{
    String query = "insert into " + tabela + "
values(" + arrayToString(values) + ")";
    System.out.print(query + "\n");

    Statement zapytanie;
    try
    {
        zapytanie = baza.createStatement();
        int wynik =
zapytanie.executeUpdate(query);

        return true;
    } catch (SQLException ex)
    {
        debug(ex);
    }

    return false;
}

public Boolean deleteRow(String tabela, String[] kolumny,
String[] values)
{
    // DELETE FROM Customers WHERE
CustomerName='Alfreds Futterkiste';
```

```
String query = "delete from " + tabela + "
where ";

    for (int i = 0; i < values.length; i++)
    {
        if (i != 0)
            query = query + " AND ";
        query = query + kolumny[i] + "
= \"" + values[i] + "\"";
    }

    System.out.print(query + "\n"); // debug

    Statement zapytanie;
    try
    {
        zapytanie = baza.createStatement();
        int wynik =
zapytanie.executeUpdate(query);

        return true;
    } catch (SQLException ex)
    {
        debug(ex);
    }

    return false;
}

    public Boolean alterRow(String tabela, String[] kolumny,
String[] values, String[] delete)
    {
        deleteRow(tabela, kolumny, delete);
        insertInto(tabela, kolumny, values);
        return false;
    }

    public void close()
    {
        try
        {
            System.out.printf("zamykam baze\
n");

            baza.close();
        } catch (SQLException ex)
        {
            debug(ex);
        }
    }

    public void debug(SQLException ex) // bo mnie wkurzaja te
try catche
    {
        System.out.println("SQLException: " +
ex.getMessage());
        System.out.println("SQLState: " +
ex.getSQLState());
        System.out.println("VendorError: " +
ex.getErrorCode());
    }
}
```



```
package bazylab3;

import javax.swing.*.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class GUI
{
    static String currentQuery = "select * from aktorzy";

    static Login user = null;
    static Mysql db1 = null;

    static String data[][] = null;
    static String kolumny[] = null;
    static JTable table = null;

    static JFrame okno = null;

    static JComboBox<String> jComboPola = new
JComboBox<String>();

    public void connect()
    {
        user = new
Login("admin", "pwsz", "192.168.0.97", 3306);
        //user = new Login();

        db1 = new Mysql(user);
    }
    public GUI()
    {
        connect();
        createTable();
        updateTable();
        displayWindow();
    }

    protected void finilize()
    {
        db1.close();
    }

    private void createTable()
    {
        table = new JTable();
    }

    public static void updateTable()
    {
        // zapytanie sql
        System.out.print("aktualizacja query " +
currentQuery + "\n");

        data = db1.select(currentQuery);
        kolumny = db1.getColumns(currentQuery);

        // naniesc zmiany do tabeli
    }
}
```

```
DefaultTableModel model = new
DefaultTableModel(data, kolumny);
model.addRow(new Object[kolumny.length]);
table.setModel(model);

// naprawic filtracje
if (jComboPola.getItemCount() > 0)
    jComboPola.removeAllItems();

for (int i = 0; i < kolumny.length; i++)
{
    jComboPola.addItem(kolumny[i]);
}

}

public static void displayWindow()
{
    // combobox do wybrania tabeli
    JComboBox<String> jComboTabele = new
JComboBox<String>(db1.getTables());

    jComboTabele.updateUI();

    // tworzenie okna
    okno = new JFrame("LAB3");
    okno.setSize(1024, 780);

    okno.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    //JOptionPane.showMessageDialog(null,
"Połączono z baza");

    // tworzenie tabeli
    JScrollPane scrollPane = new
JScrollPane(table);

    table.setFillViewportHeight(true);

    okno.getContentPane().add(BorderLayout.CENTER, table);

    okno.getContentPane().add(BorderLayout.SOUTH, jComboTabele);

    JPanel jp2 = new JPanel(); // tworzymy
panel
    JLabel jl2 = new JLabel("filtracja"); //
dajemy etykiety filtracja, żeby użytkownik wiedział co to

    // przyciski
    JButton jbadd = new JButton("dodaj");
    JButton jbdel = new JButton("usun");
    JButton jbfiler = new JButton("Filtruj");
    JButton jbalter = new JButton("zmien");
    JButton jbdelfiler = new JButton("usun
filtr");

    final JTextField tfod = new
JTextField("od");

    final JTextField tfdo = new
JTextField("do");

    jp2.add(jl2);
```

```
        jp2.add(jComboPola);
        jp2.add(tfod);
        jp2.add(tfdo);

        jp2.add(jbfilter);
        jp2.add(jbdel);
        jp2.add(jbadd);
        jp2.add(jbalter);
        jp2.add(jbdefilter);

okno.getContentPane().add(BorderLayout.NORTH, jp2);

        okno.setVisible(true);
        table.setVisible(true);

        // Wybieranie tabeli
        jComboTabele.addActionListener(new
ActionListener()
        {
            public void
actionPerformed(ActionEvent e)
            {
                currentQuery = "SELECT * FROM " + jComboTabele.getSelectedItem() + ";";
                updateTable();
            }
        });
        jbdelfilter.addActionListener(new
ActionListener()
        {
            public void
actionPerformed(ActionEvent e)
            {
                currentQuery = "SELECT * FROM " + jComboTabele.getSelectedItem() + ";";
                updateTable();
            }
        });

        // wybieranie pola do filtrowania
        jbfilter.addActionListener(new
ActionListener()
        {
            public void
actionPerformed(ActionEvent evt)
            {
                currentQuery = "SELECT * FROM " + jComboTabele.getSelectedItem() + " WHERE "
+ jComboPola.getSelectedItem() + " BETWEEN " + tfod.getText() + " AND "
+ tfdo.getText() + ";";
                updateTable();
            }
        });
    });
}
```

```
//usuwanie zaznaczonego rzędu
jbdel.addActionListener(new
ActionListener()
{
    public void
actionPerformed(ActionEvent evt)
{
    if
    { // jak
        int
        temp = table.getSelectedRow(); // tutaj numer rzędu se zapisujemy(juz chyba
        nie
        // potrzebne, ale jest
        String stringDEL[] = new String[kolumny.length];
        for
        (int i = 0; i < kolumny.length; i++)
        {
            stringDEL[i] = data[temp][i];
        }
        //
        Main.printArray(stringiDEL);
        db1.deleteRow(jComboTabele.getSelectedItem().toString(), kolumny,
        stringDEL);
        updateTable();
    }
});
//dodawanie nowego rzędu
jbadd.addActionListener(new
ActionListener()
{
    public void
actionPerformed(ActionEvent evt)
{
        String
        stringADD[] = new String[kolumny.length];
        for (int i
        = 0; i < kolumny.length; i++)
        {
            stringADD[i] = (String) table.getValueAt(data.length, i);
        }
        //
        Main.printArray(stringADD);
        db1.insertInto(jComboTabele.getSelectedItem().toString(), kolumny,
        stringADD);
        updateTable();
    }
});
jbalter.addActionListener(new
ActionListener()
```

```
{  
    public void  
    actionPerformed(ActionEvent evt)  
    {  
        if  
        {  
            int  
            temp = table.getSelectedRow();  
            String stringDEL[] = new String[kolumny.length];  
            String stringADD[] = new String[kolumny.length];  
  
            for  
            {  
                stringDEL[i] = data[temp][i];  
                stringADD[i] = (String) table.getValueAt(temp, i);  
            }  
            //  
            Main.printArray(stringiDEL);  
  
            db1.alterRow(jComboTabele.getSelectedItem().toString(), kolumny, stringADD,  
            stringDEL);  
            updateTable();  
        }  
    }  
});  
};;
```

## Literatura

- [1] Oracle: *Trail: JDBC™ Database Access*,  
<https://docs.oracle.com/javase/tutorial/jdbc/index.html>
- [2] Oracle: *Using Swing Components*,  
<https://docs.oracle.com/javase/tutorial/uiswing/components/index.html>
- [3] Oracle: *Writing Event Listeners*,  
<https://docs.oracle.com/javase/tutorial/uiswing/events/index.html>