Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №7

По дисциплине: «Современные системы программирования»

Тема: «Разработка пользовательского интерфейса для работы с базами данных»

Выполнил:

Студент 4 курса

Группы ИИ-16 (2)

Пешко А.С.

Проверил:

Коренкович О. Г.

Брест 2021

**Цель работы:** Научиться разрабатывать GUI-приложения с использованием баз данных.

**Ход работы**

**Задание:**

Разработать подсистему учета и регистрации посещений поликлиники больными.

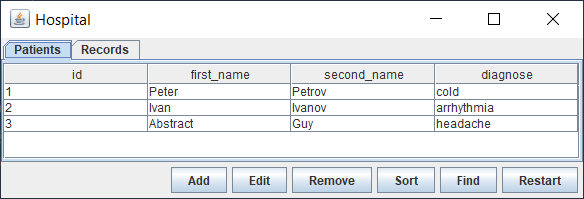
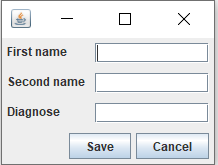
**Код программы:**  
  
public class HospitalForm extends JFrame {  
  
 private final String SELECT\_ALL\_PATIENTS = "SELECT \* FROM patients";  
 private final String SORT\_BY\_SECOND\_NAME\_PATIENTS = "SELECT \* FROM patients ORDER BY second\_name;";  
  
 private final String SELECT\_ALL\_RECORDS = "SELECT \* FROM records";  
 private final String SORT\_BY\_ORDER\_DATE\_RECORDS = "SELECT \* FROM records ORDER BY record\_date DESC;";  
  
 Integer selectedTab = 0;  
 Hospital database;  
 JTabbedPane tabbedPane = new JTabbedPane();  
 ArrayList<JTable> tables;  
 String[] tabs = {"Patients", "Records"};  
  
 JPanel panel = new JPanel(new BorderLayout());  
  
 public HospitalForm() {  
 tables = new ArrayList<>();  
 database = new Hospital();  
 createWindow();  
 tables.add(createTable(SELECT\_ALL\_PATIENTS));  
 tables.add(createTable(SELECT\_ALL\_RECORDS));  
  
  
 for (int i = 0; i < tables.size(); i++) {  
 JScrollPane scroll = new JScrollPane(tables.get(i));  
 tables.get(i).setFillsViewportHeight(true);  
 tabbedPane.add(tabs[i], scroll);  
 }  
 panel.add(bottomPanel(), BorderLayout.SOUTH);  
 panel.add(tabbedPane, BorderLayout.CENTER);  
 tabbedPane.addChangeListener(e -> selectedTab = ((JTabbedPane) e.getSource()).getSelectedIndex());  
 add(panel);  
 setVisible(true);  
 }  
  
 private void createWindow() {  
 setTitle("Hospital");  
 setSize(600, 600);  
 setLocationRelativeTo(null);  
 setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);  
 }  
  
 public void update() {  
 remove(panel);  
 int selectedI = tabbedPane.getSelectedIndex();  
 tabbedPane.removeAll();  
 panel = new JPanel(new BorderLayout());  
 for (int i = 0; i < tables.size(); i++) {  
 JScrollPane scroll = new JScrollPane(tables.get(i));  
 tables.get(i).setFillsViewportHeight(true);  
 tabbedPane.add(tabs[i], scroll);  
 }  
 panel.add(bottomPanel(), BorderLayout.SOUTH);  
 panel.add(tabbedPane, BorderLayout.CENTER);  
 tabbedPane.addChangeListener(e -> selectedTab = ((JTabbedPane) e.getSource()).getSelectedIndex());  
 add(panel);  
 revalidate();  
 tabbedPane.setSelectedIndex(selectedI);  
 repaint();  
 }  
  
 private JTable createTable(String query) {  
 JTable table = null;  
 try {  
 ResultSet resultSet = database.statement.executeQuery(query);  
 ResultSetMetaData resultSetMetaData = resultSet.getMetaData();  
 String[] columnNames = new String[resultSetMetaData.getColumnCount()];  
  
 for (int i = 1; i <= resultSetMetaData.getColumnCount(); i++)  
 columnNames[i - 1] = resultSetMetaData.getColumnLabel(i);  
  
 ArrayList<ArrayList<Object>> dataList = new ArrayList<>();  
 for (int i = 0; resultSet.next(); i++) {  
 dataList.add(new ArrayList<>());  
 for (String value : columnNames) {  
 dataList.get(i).add(resultSet.getObject(value));  
 }  
 }  
  
 Object[][] data = new Object[0][];  
 if(dataList.size() != 0) {  
 data = new Object[dataList.size()][dataList.get(0).size()];  
 for (int i = 0; i < dataList.size(); i++) {  
 for (int j = 0; j < dataList.get(0).size(); j++) {  
 data[i][j] = dataList.get(i).get(j);  
 }  
 }  
 }  
  
 table = new JTable(data, columnNames);  
 table.setSelectionMode(0);  
 } catch (SQLException e) {  
 System.out.println("Error in creating table : " + e.getMessage());;  
 }  
 return table;  
 }  
 private JPanel bottomPanel() {  
 JPanel panel = new JPanel(new FlowLayout(FlowLayout.RIGHT));  
 JButton addButton = new JButton("Add");  
 addButton.addActionListener(e -> {  
 editForm(-1, -1);  
 });  
 JButton edit = new JButton("Edit");  
 edit.addActionListener(e -> {  
 editForm((Integer) tables.get(selectedTab).getValueAt(tables.get(selectedTab).getSelectedRow(), 0), tables.get(selectedTab).getSelectedRow());  
 });  
 JButton remove = new JButton("Remove");  
 remove.addActionListener(e -> {  
 switch (selectedTab) {  
 case 0: {  
 database.removeNote("patients", (Integer) tables.get(selectedTab).getValueAt(tables.get(selectedTab).getSelectedRow(), 0));  
 tables.set(0, createTable(SELECT\_ALL\_PATIENTS));  
 update();  
 break;  
 }  
 case 1: {  
 database.removeNote("records", (Integer) tables.get(selectedTab).getValueAt(tables.get(selectedTab).getSelectedRow(), 0));  
 tables.set(1, createTable(SELECT\_ALL\_RECORDS));  
 update();  
 break;  
 }  
 }  
 });  
 JButton sort = new JButton("Sort");  
 sort.addActionListener(e -> {  
 switch (selectedTab) {  
 case 0: {  
 tables.set(0, createTable(SORT\_BY\_SECOND\_NAME\_PATIENTS));  
 update();  
 break;  
 }  
 case 1: {  
 tables.set(1, createTable(SORT\_BY\_ORDER\_DATE\_RECORDS));  
 update();  
 break;  
 }  
 }  
 });  
 JButton find = new JButton("Find");  
 find.addActionListener(e -> {  
 findForm();  
 });  
 JButton restart = new JButton("Restart");  
 restart.addActionListener(e -> {  
 switch (selectedTab) {  
 case 0: {  
 tables.set(0, createTable(SELECT\_ALL\_PATIENTS));  
 update();  
 break;  
 }  
 case 1: {  
 tables.set(1, createTable(SELECT\_ALL\_RECORDS));  
 update();  
 break;  
 }  
 }  
 });  
 panel.add(addButton);  
 panel.add(edit);  
 panel.add(remove);  
 panel.add(sort);  
 panel.add(find);  
 panel.add(restart);  
 return panel;  
 }  
 public void func(JPanel p, JComponent c, Integer x, Integer y, Integer w, Integer h, boolean b, GridBagConstraints gbc) {  
 gbc.gridx = x;  
 gbc.gridy = y;  
 gbc.gridwidth = w;  
 gbc.gridheight = h;  
 gbc.fill = b ? GridBagConstraints.HORIZONTAL : GridBagConstraints.VERTICAL;  
 p.add(c, gbc);  
 }  
 private void editForm(Integer index, Integer row) {  
 JFrame frame = new JFrame();  
 GridBagConstraints gbc = new GridBagConstraints();  
 JPanel panel = new JPanel(new BorderLayout());  
 JPanel p1 = new JPanel(new GridBagLayout());  
 gbc.insets = new Insets(5, 5, 5, 5);  
 gbc.anchor = GridBagConstraints.NORTH;  
 JPanel p2 = new JPanel(new FlowLayout(FlowLayout.RIGHT));  
 JButton save = new JButton("Save");  
 JButton cancel = new JButton("Cancel");  
 switch (selectedTab) {  
 // Patients  
 case 0: {  
 JTextField firstName = index.equals(-1) ? new JTextField(10) : new JTextField(tables.get(selectedTab).getValueAt(row, 1).toString(), 10);  
 func(p1, new JLabel("First name"), 0, 0, 1, 1, true, gbc);  
 func(p1, firstName, 1, 0, 4, 1, true, gbc);  
  
 JTextField secondName = index.equals(-1) ? new JTextField(10) : new JTextField(tables.get(selectedTab).getValueAt(row, 2).toString(), 10);  
 func(p1, new JLabel("Second name"), 0, 1, 1, 1, true, gbc);  
 func(p1, secondName, 1, 1, 4, 1, false, gbc);  
  
 JTextField diagnose = index.equals(-1) ? new JTextField(10) : new JTextField(tables.get(selectedTab).getValueAt(row, 3).toString(), 10);  
 func(p1, new JLabel("Diagnose"), 0, 2, 1, 1, true, gbc);  
 func(p1, diagnose, 1, 2, 4, 1, false, gbc);  
  
 save.addActionListener(e -> {  
 try {  
 if (firstName.getText().equals("") || secondName.getText().equals("") || diagnose.getText().equals("")) {  
 throw new Exception();  
 }  
 if (index.equals(-1)) {  
 database.addNote("patients", new Object[]{  
 firstName.getText(),  
 secondName.getText(),  
 diagnose.getText()  
 });  
 } else {  
 database.updateNote("patients", new Object[]{  
 firstName.getText(),  
 secondName.getText(),  
 diagnose.getText()  
 }, index);  
 }  
 frame.dispose();  
 frame.setVisible(false);  
 tables.set(0, createTable(SELECT\_ALL\_PATIENTS));  
 update();  
 } catch (Exception generalException) {  
 JOptionPane.showMessageDialog(this, "Empty fields", "Error", JOptionPane.OK\_OPTION);  
 }  
 });  
 cancel.addActionListener(e -> {  
 frame.dispose();  
 frame.setVisible(false);  
 });  
 break;  
 }  
 // Records  
 case 1: {  
 LinkedHashMap<String, Integer> drivers = new LinkedHashMap<>();  
  
 JComboBox patient = new JComboBox();  
  
 try {  
 ResultSet resultSet = database.statement.executeQuery(SELECT\_ALL\_PATIENTS);  
 ResultSetMetaData resultSetMetaData = resultSet.getMetaData();  
 String[] columnNames = new String[resultSetMetaData.getColumnCount()];  
  
 for (int i = 1; i <= resultSetMetaData.getColumnCount(); i++) {  
 columnNames[i - 1] = resultSetMetaData.getColumnLabel(i);  
 }  
 columnNames = new String[]{columnNames[0], columnNames[2]};  
 for (int i = 0; resultSet.next(); i++) {  
 drivers.put(resultSet.getString(columnNames[1]), resultSet.getInt(columnNames[0]));  
 patient.addItem(resultSet.getString(columnNames[1]));  
 }  
  
 } catch (SQLException e) {  
 System.out.println("Records table, getting initial content error : " + e.getMessage());  
 }  
  
 if (!index.equals(-1)) {  
 patient.setSelectedItem(tables.get(selectedTab).getValueAt(row, 3).toString());  
 }  
 func(p1, new JLabel("Patient"), 0, 2, 1, 1, true, gbc);  
 func(p1, patient, 1, 2, 4, 1, false, gbc);  
  
 JTextField date = index.equals(-1) ? new JTextField("2021-01-01", 10) : new JTextField(tables.get(selectedTab).getValueAt(row, 2).toString(), 10);  
 func(p1, new JLabel("Date"), 0, 3, 1, 1, true, gbc);  
 func(p1, date, 1, 3, 4, 1, false, gbc);  
  
 JTextField description = index.equals(-1) ? new JTextField(10) : new JTextField(tables.get(selectedTab).getValueAt(row, 3).toString(), 10);  
 func(p1, new JLabel("Description"), 0, 0, 1, 1, true, gbc);  
 func(p1, description, 1, 0, 4, 1, true, gbc);  
  
 save.addActionListener(e -> {  
 try {  
 if (description.getText().equals("") || date.getText().equals("")) {  
 throw new Exception();  
 }  
 if (index.equals(-1)) {  
 database.addNote("records", new Object[]{  
 description.getText(),  
 drivers.get(patient.getSelectedItem()),  
 date.getText()  
 });  
 } else {  
 database.updateNote("records", new Object[]{  
 description.getText(),  
 drivers.get(patient.getSelectedItem()),  
 date.getText()  
 }, index);  
 }  
 frame.dispose();  
 frame.setVisible(false);  
 tables.set(1, createTable(SELECT\_ALL\_RECORDS));  
 update();  
 } catch (NumberFormatException exception) {  
 JOptionPane.showMessageDialog(this, "Fields must be number", "Error", JOptionPane.OK\_OPTION);  
 } catch (Exception generalException) {  
 JOptionPane.showMessageDialog(this, "Empty fields", "Error", JOptionPane.OK\_OPTION);  
 }  
 });  
 cancel.addActionListener(e -> {  
 frame.dispose();  
 frame.setVisible(false);  
 });  
 break;  
 }  
 }  
 p2.add(save);  
 p2.add(cancel);  
 p1.setSize(700, 700);  
 panel.add(p1, BorderLayout.CENTER);  
 panel.add(p2, BorderLayout.SOUTH);  
 frame.add(panel);  
 frame.setSize(300, 300);  
 frame.setLocationRelativeTo(null);  
 frame.pack();  
 frame.setVisible(true);  
 }  
  
 private void findForm() {  
  
 JFrame frame = new JFrame();  
 GridBagConstraints gbc = new GridBagConstraints();  
 JPanel panel = new JPanel(new BorderLayout());  
 JPanel p1 = new JPanel(new GridBagLayout());  
 gbc.insets = new Insets(5, 5, 5, 5);  
 gbc.anchor = GridBagConstraints.NORTH;  
 JPanel p2 = new JPanel(new FlowLayout(FlowLayout.RIGHT));  
 JButton find = new JButton("Find");  
 JButton cancel = new JButton("Cancel");  
 switch (selectedTab) {  
 // Patients  
 case 0: {  
 JTextField secondName = new JTextField(10);  
 func(p1, new JLabel("Second name"), 0, 1, 1, 1, true, gbc);  
 func(p1, secondName, 1, 1, 4, 1, false, gbc);  
 find.addActionListener(e -> {  
 try {  
 if (secondName.getText().equals("")) {  
 throw new Exception();  
 }  
 frame.dispose();  
 frame.setVisible(false);  
 tables.set(0, createTable("SELECT \* FROM patients WHERE second\_name = '" + secondName.getText() +"'"));  
 update();  
 } catch (Exception generalException) {  
 System.out.println("Error in finding patient : " + generalException.getMessage());  
 JOptionPane.showMessageDialog(this, "Empty fields", "Error", JOptionPane.OK\_OPTION);  
 }  
 });  
 cancel.addActionListener(e -> {  
 frame.dispose();  
 frame.setVisible(false);  
 });  
 break;  
 }  
 // Records  
 case 1: {  
 JTextField date = new JTextField("2021-01-01", 10);  
 func(p1, new JLabel("Date"), 0, 3, 1, 1, true, gbc);  
 func(p1, date, 1, 3, 4, 1, false, gbc);  
 find.addActionListener(e -> {  
 try {  
 if (date.getText().equals("")) {  
 throw new Exception();  
 }  
 frame.dispose();  
 frame.setVisible(false);  
 tables.set(1, createTable("SELECT \* FROM records WHERE record\_date = '" + date.getText() +"'"));  
 update();  
 } catch (Exception generalException) {  
 JOptionPane.showMessageDialog(this, "Empty fields", "Error", JOptionPane.OK\_OPTION);  
 }  
 });  
 cancel.addActionListener(e -> {  
 frame.dispose();  
 frame.setVisible(false);  
 });  
 break;  
 }  
 }

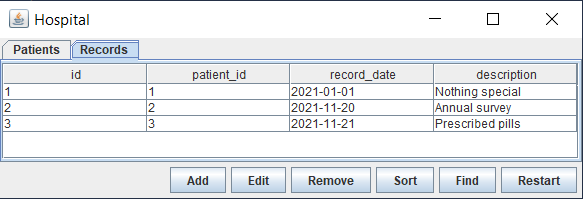
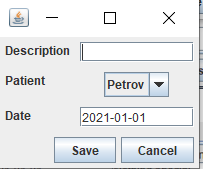
p2.add(find);  
 p2.add(cancel);

p1.setSize(700, 700);  
 panel.add(p1, BorderLayout.CENTER);  
 panel.add(p2, BorderLayout.SOUTH);  
  
 frame.add(panel);  
 frame.setSize(300, 300);  
 frame.setLocationRelativeTo(null);  
 frame.pack();  
 frame.setVisible(true);  
 }

public static void main(String[] args) {  
 new HospitalForm();  
 }  
}

**Результат работы прогаммы:**

**Вывод:** научились разрабатывать GUI-приложения с использованием баз данных..