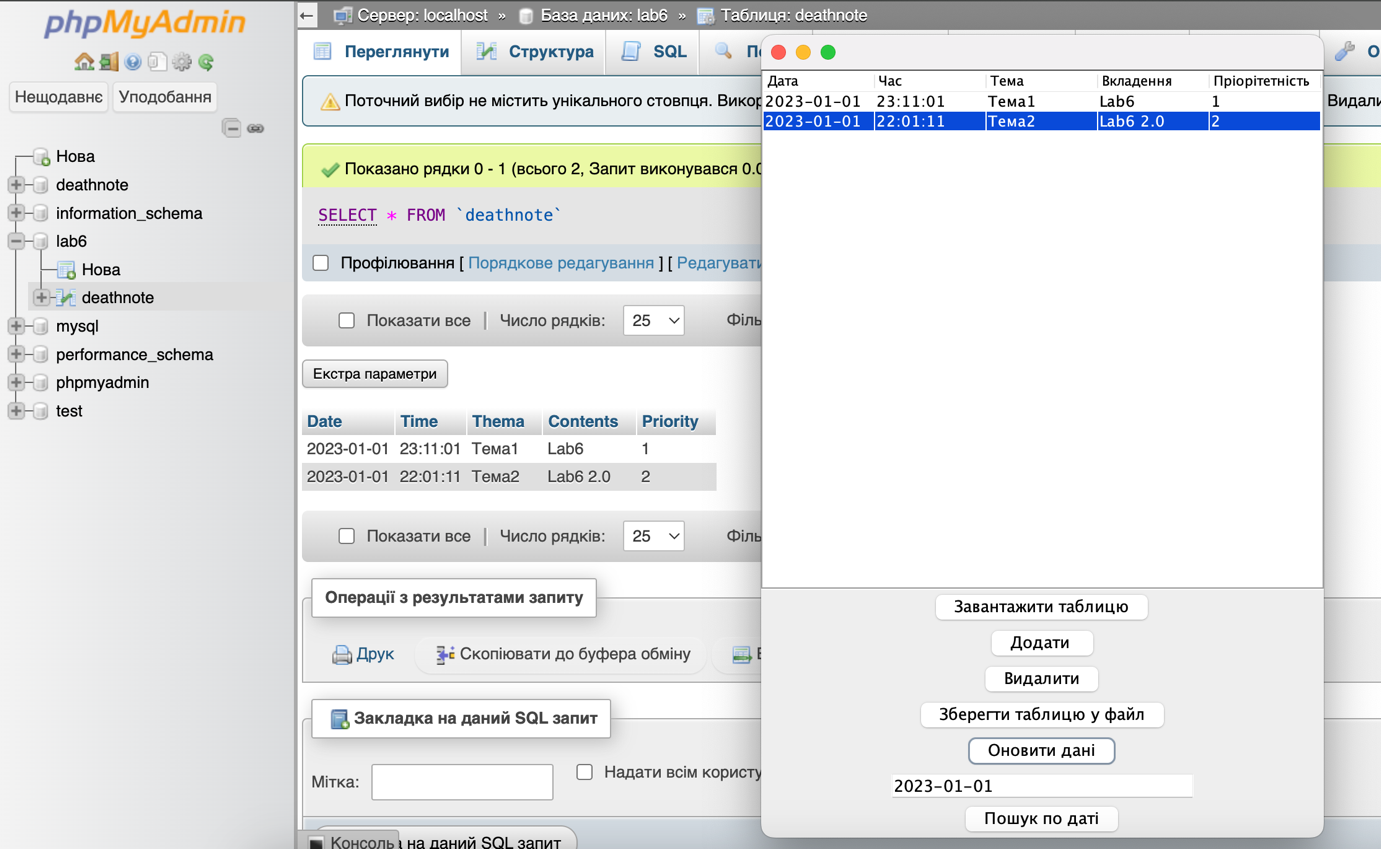
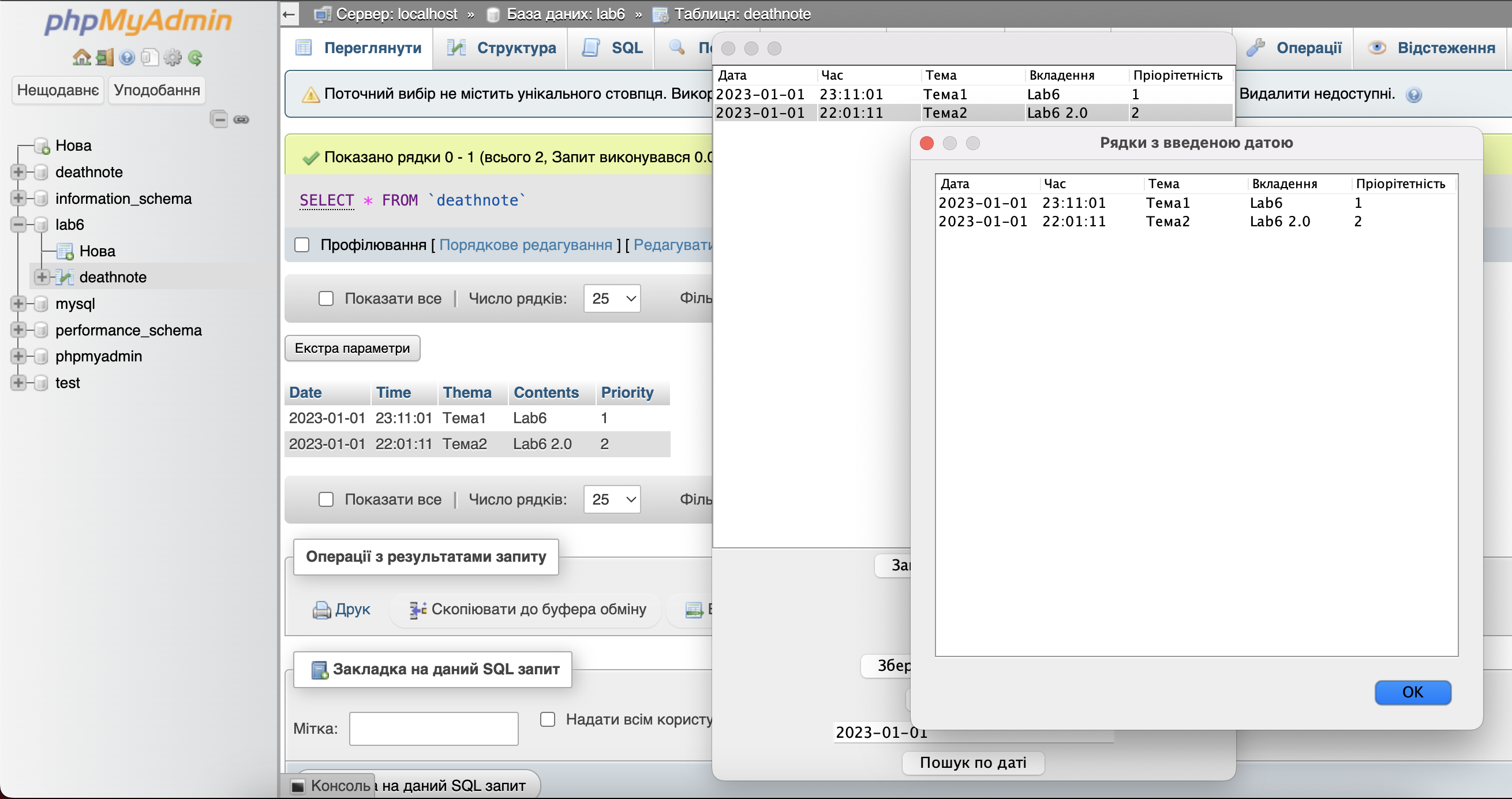
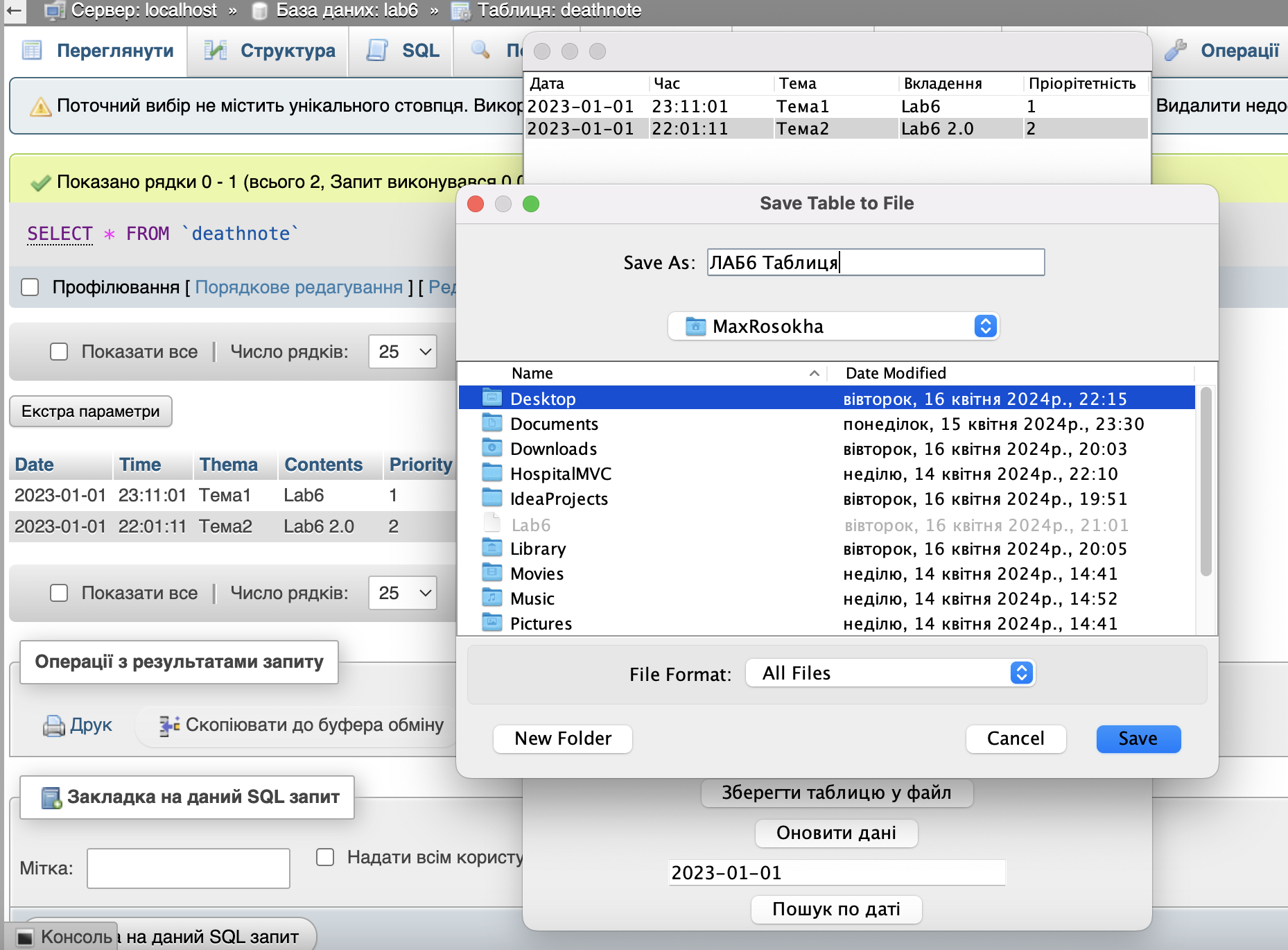
**Лабораторна робота №6(17)**

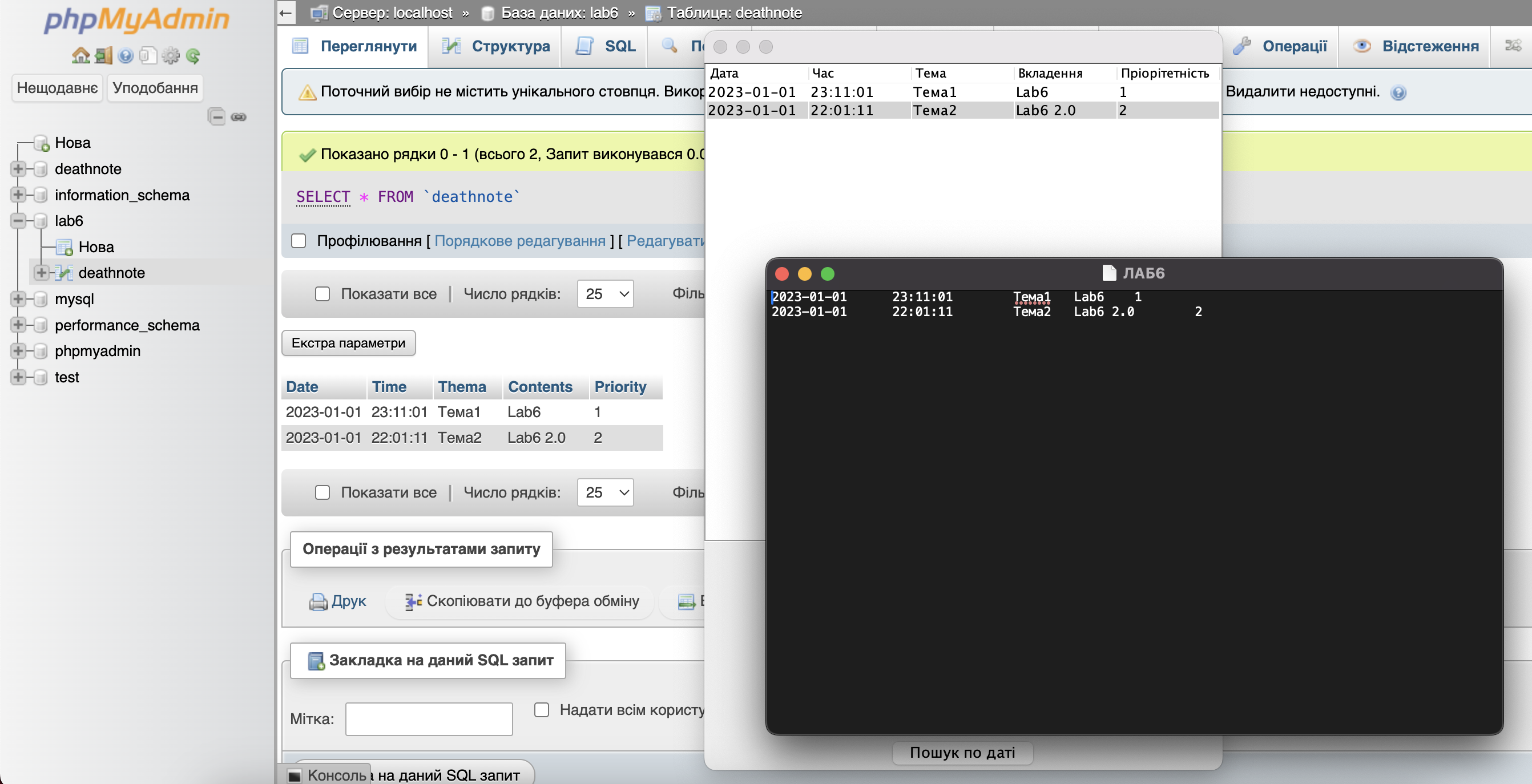
В.2

Створити програму яка виконуватиме функції персонального щоденника. (запис важливих подій за день)









import javax.swing.\*;  
import javax.swing.table.DefaultTableModel;  
import java.awt.\*;  
import java.awt.event.\*;  
import java.io.\*;  
import java.sql.\*;  
  
public class NewJFrame extends javax.swing.JFrame {  
 private DefaultTableModel initialModel;  
 private DefaultTableModel model;  
 private Connection connection;  
  
 // Створимо новий DefaultTableModel та скопіюємо дані з поточної моделі  
 private DefaultTableModel createCopyTableModel(DefaultTableModel originalModel) {  
 DefaultTableModel copyModel = new DefaultTableModel();  
 // Додаємо колонки  
 for (int columnIndex = 0; columnIndex < originalModel.getColumnCount(); columnIndex++) {  
 copyModel.addColumn(originalModel.getColumnName(columnIndex));  
 }  
 // Додаємо дані  
 for (int rowIndex = 0; rowIndex < originalModel.getRowCount(); rowIndex++) {  
 Object[] rowData = new Object[originalModel.getColumnCount()];  
 for (int columnIndex = 0; columnIndex < originalModel.getColumnCount(); columnIndex++) {  
 rowData[columnIndex] = originalModel.getValueAt(rowIndex, columnIndex);  
 }  
 copyModel.addRow(rowData);  
 }  
 return copyModel;  
 }  
  
 public NewJFrame() {  
 initComponents();  
 connectToDatabase();  
 // Ініціалізуємо початкову модель як копію поточної моделі  
 initialModel = createCopyTableModel(model);  
 }  
  
  
 private void initComponents() {  
 jScrollPane1 = new javax.swing.JScrollPane();  
 jTable1 = new javax.swing.JTable();  
 jButton1 = new javax.swing.JButton();  
 jButton2 = new javax.swing.JButton();  
 jButton3 = new javax.swing.JButton();  
 jButton4 = new javax.swing.JButton();  
 jButton5 = new javax.swing.JButton(); // Кнопка для оновлення даних  
 jTextField1 = new javax.swing.JTextField(); // Поле для введення дати пошуку  
 jButton6 = new javax.swing.JButton(); // Кнопка для пошуку  
  
 setDefaultCloseOperation(javax.swing.WindowConstants.*EXIT\_ON\_CLOSE*);  
  
 model = new DefaultTableModel(  
 new Object [][] {  
  
 },  
 new String [] {  
 "Дата", "Час", "Тема", "Вкладення", "Пріорітетність"  
 }  
 );  
  
 jTable1.setModel(model);  
 jScrollPane1.setViewportView(jTable1);  
  
 jButton1.setText("Завантажити таблицю");  
 jButton1.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 jButton1ActionPerformed(evt);  
 }  
 });  
  
 jButton2.setText("Додати ");  
 jButton2.setEnabled(false);  
 jButton2.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 jButton2ActionPerformed(evt);  
 }  
 });  
  
 jButton3.setText("Видалити");  
 jButton3.setEnabled(false);  
 jButton3.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 jButton3ActionPerformed(evt);  
 }  
 });  
  
 jButton4.setText("Зберегти таблицю у файл");  
 jButton4.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 jButton4ActionPerformed(evt);  
 }  
 });  
  
 jButton5.setText("Оновити дані"); // Текст кнопки "Оновити дані"  
 jButton5.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 jButton5ActionPerformed(evt);  
 }  
 });  
  
  
 jTextField1.setToolTipText("Введіть дату у форматі yyyy-mm-dd");  
 jTextField1.setColumns(20);  
  
 jButton6.setText("Пошук по даті"); // Кнопка для пошуку за датою  
 jButton6.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 jButton6ActionPerformed(evt);  
 }  
 });  
  
 JPanel centerPanel = new JPanel(new GridBagLayout());  
 GridBagConstraints gbc = new GridBagConstraints();  
 gbc.gridx = 0;  
 gbc.gridy = 0;  
 centerPanel.add(jScrollPane1, gbc);  
 gbc.gridy++;  
 centerPanel.add(jButton1, gbc);  
 gbc.gridy++;  
 centerPanel.add(jButton2, gbc);  
 gbc.gridy++;  
 centerPanel.add(jButton3, gbc);  
 gbc.gridy++;  
 centerPanel.add(jButton4, gbc);  
 gbc.gridy++;  
 centerPanel.add(jButton5, gbc); // Додана кнопка "Оновити дані"  
 gbc.gridy++;  
 centerPanel.add(jTextField1, gbc); // Додане поле для введення дати  
 gbc.gridy++;  
 centerPanel.add(jButton6, gbc); // Додана кнопка для пошуку за датою  
  
 getContentPane().add(centerPanel, BorderLayout.*CENTER*);  
  
 pack();  
 }  
  
 private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
 loadTableDataFromServer();  
 }  
  
 private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {  
 model.addRow(new Object[]{"Дата", "Час", "Тема", "Вкладення", "Пріорітетність"});  
 }  
  
 private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {  
 int selectedRowIndex = jTable1.getSelectedRow();  
 if (selectedRowIndex >= 0) {  
 model.removeRow(selectedRowIndex);  
 deleteRowFromServer(selectedRowIndex);  
 } else {  
 JOptionPane.*showMessageDialog*(this, "Виберіть рядок для видалення", "Помилка", JOptionPane.*ERROR\_MESSAGE*);  
 }  
 }  
  
 private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {  
 saveTableToFile();  
 }  
  
 private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {  
 updateDataOnServer();  
 }  
  
 private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {  
 String searchDate = jTextField1.getText();  
 searchByDate(searchDate);  
 }  
  
  
 private void jTable1MouseClicked(java.awt.event.MouseEvent evt) {  
 int lastSelectedRowIndex = jTable1.getSelectedRow();  
 }  
  
 private void jTable1KeyReleased(java.awt.event.KeyEvent evt) {  
 int lastSelectedRowIndex = jTable1.getSelectedRow();  
 }  
  
 private void jTable1InputMethodTextChanged(java.awt.event.InputMethodEvent evt) {  
 int lastSelectedRowIndex = jTable1.getSelectedRow();  
 }  
  
  
  
 private void connectToDatabase() {  
 try {  
 String url = "jdbc:mysql://localhost:3306/lab6";  
 String username = "root";  
 String password = "";  
 connection = DriverManager.*getConnection*(url, username, password);  
 } catch (SQLException ex) {  
 ex.printStackTrace();  
 JOptionPane.*showMessageDialog*(this, "Error connecting to database: " + ex.getMessage(), "Error", JOptionPane.*ERROR\_MESSAGE*);  
 }  
 }  
  
 public void loadTableDataFromServer() {  
 model.setRowCount(0);  
  
 try {  
 String sql = "SELECT Date, Time, Thema, Contents, Priority FROM deathnote";  
 Statement statement = connection.createStatement();  
 ResultSet resultSet = statement.executeQuery(sql);  
  
 while (resultSet.next()) {  
 Object[] rowData = {  
 resultSet.getDate("Date"),  
 resultSet.getTime("Time"),  
 resultSet.getString("Thema"),  
 resultSet.getString("Contents"),  
 resultSet.getString("Priority")  
 };  
 model.addRow(rowData);  
 }  
  
 jButton2.setEnabled(true);  
 jButton3.setEnabled(true);  
 } catch (SQLException ex) {  
 ex.printStackTrace();  
 JOptionPane.*showMessageDialog*(this, "Error loading data: " + ex.getMessage(), "Error", JOptionPane.*ERROR\_MESSAGE*);  
 }  
 }  
  
 private void deleteRowFromServer(int rowIndex) {  
 try {  
 String sql = "DELETE FROM deathnote WHERE Date=? AND Time=? AND Thema=? AND Contents=? AND Priority=?";  
 PreparedStatement statement = connection.prepareStatement(sql);  
 statement.setDate(1, (Date) model.getValueAt(rowIndex, 0));  
 statement.setTime(2, (Time) model.getValueAt(rowIndex, 1));  
 statement.setString(3, (String) model.getValueAt(rowIndex, 2));  
 statement.setString(4, (String) model.getValueAt(rowIndex, 3));  
 statement.setString(5, (String) model.getValueAt(rowIndex, 4));  
 statement.executeUpdate();  
 } catch (SQLException ex) {  
 ex.printStackTrace();  
 JOptionPane.*showMessageDialog*(this, "Error deleting data from server: " + ex.getMessage(), "Error", JOptionPane.*ERROR\_MESSAGE*);  
 }  
 }  
  
 public void saveTableToFile() {  
 JFileChooser fileChooser = new JFileChooser();  
 fileChooser.setDialogTitle("Save Table to File");  
 int userSelection = fileChooser.showSaveDialog(this);  
 if (userSelection == JFileChooser.*APPROVE\_OPTION*) {  
 File fileToSave = fileChooser.getSelectedFile();  
 try (PrintWriter writer = new PrintWriter(fileToSave)) {  
 for (int i = 0; i < model.getRowCount(); i++) {  
 for (int j = 0; j < model.getColumnCount(); j++) {  
 writer.print(model.getValueAt(i, j));  
 if (j < model.getColumnCount() - 1) {  
 writer.print("\t");  
 }  
 }  
 writer.println();  
 }  
 JOptionPane.*showMessageDialog*(this, "Table saved successfully!", "Success", JOptionPane.*INFORMATION\_MESSAGE*);  
 } catch (IOException ex) {  
 ex.printStackTrace();  
 JOptionPane.*showMessageDialog*(this, "Error saving table: " + ex.getMessage(), "Error", JOptionPane.*ERROR\_MESSAGE*);  
 }  
 }  
 }  
  
 private void updateDataOnServer() {  
 // Оновлення всіх даних на сервері  
 try {  
 // Очистимо таблицю на сервері  
 String clearSql = "TRUNCATE TABLE deathnote";  
 PreparedStatement clearStatement = connection.prepareStatement(clearSql);  
 clearStatement.executeUpdate();  
  
 // Додамо всі рядки з моделі до бази даних  
 String insertSql = "INSERT INTO deathnote (Date, Time, Thema, Contents, Priority) VALUES (?, ?, ?, ?, ?)";  
 PreparedStatement insertStatement = connection.prepareStatement(insertSql);  
 for (int i = 0; i < model.getRowCount(); i++) {  
 // Перевіряємо, чи всі обов'язкові поля заповнені  
 if (model.getValueAt(i, 0) != null && model.getValueAt(i, 1) != null &&  
 model.getValueAt(i, 2) != null && model.getValueAt(i, 3) != null &&  
 model.getValueAt(i, 4) != null) {  
 // Перевіряємо, чи дані відповідають очікуваним типам даних  
 if (model.getValueAt(i, 0) instanceof String &&  
 model.getValueAt(i, 1) instanceof String &&  
 model.getValueAt(i, 2) instanceof String &&  
 model.getValueAt(i, 3) instanceof String &&  
 model.getValueAt(i, 4) instanceof String) {  
 insertStatement.setDate(1, java.sql.Date.*valueOf*((String) model.getValueAt(i, 0)));  
 insertStatement.setTime(2, java.sql.Time.*valueOf*((String) model.getValueAt(i, 1)));  
 insertStatement.setString(3, (String) model.getValueAt(i, 2));  
 insertStatement.setString(4, (String) model.getValueAt(i, 3));  
 insertStatement.setString(5, (String) model.getValueAt(i, 4));  
 insertStatement.executeUpdate();  
 }  
 }  
 }  
  
 JOptionPane.*showMessageDialog*(this, "Data updated successfully!", "Success", JOptionPane.*INFORMATION\_MESSAGE*);  
 } catch (SQLException ex) {  
 ex.printStackTrace();  
 JOptionPane.*showMessageDialog*(this, "Error updating data on server: " + ex.getMessage(), "Error", JOptionPane.*ERROR\_MESSAGE*);  
 }  
 }  
  
 private void searchByDate(String searchDate) {  
 if (searchDate.isEmpty()) {  
 JOptionPane.*showMessageDialog*(this, "Введіть дату для пошуку", "Помилка", JOptionPane.*ERROR\_MESSAGE*);  
 return;  
 }  
  
 DefaultTableModel matchingRowsModel = new DefaultTableModel(  
 new Object[][]{},  
 new String[]{"Дата", "Час", "Тема", "Вкладення", "Пріорітетність"}  
 );  
  
 for (int i = 0; i < model.getRowCount(); i++) {  
 if (model.getValueAt(i, 0).toString().equals(searchDate)) {  
 matchingRowsModel.addRow(new Object[]{  
 model.getValueAt(i, 0),  
 model.getValueAt(i, 1),  
 model.getValueAt(i, 2),  
 model.getValueAt(i, 3),  
 model.getValueAt(i, 4)  
 });  
 }  
 }  
  
 if (matchingRowsModel.getRowCount() > 0) {  
 // Виведемо вікно з усіма рядками, що мають таку саму дату  
 JTable matchingRowsTable = new JTable(matchingRowsModel);  
 JScrollPane scrollPane = new JScrollPane(matchingRowsTable);  
 JOptionPane.*showMessageDialog*(this, scrollPane, "Рядки з введеною датою", JOptionPane.*PLAIN\_MESSAGE*);  
 } else {  
 JOptionPane.*showMessageDialog*(this, "Даної дати немає у таблиці", "Повідомлення", JOptionPane.*INFORMATION\_MESSAGE*);  
 }  
 }  
  
  
 // Метод для відновлення початкової моделі після виконання пошуку  
 private void restoreInitialModel() {  
 jTable1.setModel(initialModel);  
 enableButtons(true); // Включаємо кнопки після відновлення початкової моделі  
 }  
  
 // Метод для включення або виключення кнопок  
 private void enableButtons(boolean enabled) {  
 jButton2.setEnabled(enabled);  
 jButton3.setEnabled(enabled);  
 }  
  
 private void showSearchResults(DefaultTableModel searchModel) {  
 jTable1.setModel(searchModel);  
 enableButtons(true); // Включаємо кнопки після встановлення нової моделі  
 }  
  
  
  
 public static void main(String args[]) {  
 java.awt.EventQueue.*invokeLater*(new Runnable() {  
 public void run() {  
 new NewJFrame().setVisible(true);  
 }  
 });  
 }  
  
 private javax.swing.JButton jButton1;  
 private javax.swing.JButton jButton2;  
 private javax.swing.JButton jButton3;  
 private javax.swing.JButton jButton4;  
 private javax.swing.JButton jButton5; // Додана кнопка для оновлення даних  
 private javax.swing.JButton jButton6; // Додана кнопка для пошуку за датою  
 private javax.swing.JTextField jTextField1; // Додане поле для введення дати  
 private javax.swing.JScrollPane jScrollPane1;  
 private javax.swing.JTable jTable1;  
}

**Висновки**: Цей код представляє простий додаток для управління таблицею даних, яка зберігається в базі даних MySQL. Основні функції додатку включають завантаження таблиці з бази даних, додавання нових рядків, видалення рядків та збереження таблиці у текстовий файл.

Код розділений на декілька методів для кожної функціональності, що полегшує читабельність та розуміння коду. Для кожного методу надано коментар Javadoc, який пояснює його призначення та вхідні/вихідні параметри.

Додаток також містить обробники подій для кнопок, які виконують відповідні дії, такі як завантаження, додавання, видалення та збереження таблиці. Крім того, передбачені винятки та помилки, що можуть виникнути під час виконання програми, і їх обробка за допомогою діалогових вікон повідомлень.

Цей код демонструє базовий функціонал для роботи з таблицями даних у Java Swing та взаємодії з базою даних MySQL, а також надає можливість збереження даних у текстовий файл.