**ЛАБОРАТОРНА РОБОТА № 3**

з дисципліни

«Розробка програмного забезпечення на Java»

Шейко Р. О.

Групи КІ-21-2

**Завдання на лабораторну роботу**

1. Додати до таблиці п'ятий стовпець з контактними даними студента, відповідно на головній формі додати компоненти і доопрацювати код, щоб дані додавалися в таблицю.

2. Доопрацювати код моделі даних таблиці так, щоб нумерація в першому стовпці реалізовувалася через модель, і при видаленні елемента із середини списку не було пропуску в нумерації.

3. Створити панель для фільтрації значень в таблиці за вказаним фрагментом. Для фільтрації застососувати RowFilter для вибору стовпця, за яким ведеться пошук. Значення фрагмента застосувати JComboBox. Використовувати компонент для пошуку вводиться користувачем в компонент JTextField.

4. Додати до додатку функціонал редагування даних обраного користувачем рядку в таблиці.

5. Виконати оптимізацію проекту як ї в попередніх роботах. Застосувати компонент JFormattedTextField.

1. Додамо до таблиці п'ятий стовпець з контактними даними студента, відповідно на головній формі додати компоненти і доопрацювати код, щоб дані додавалися в таблицю.

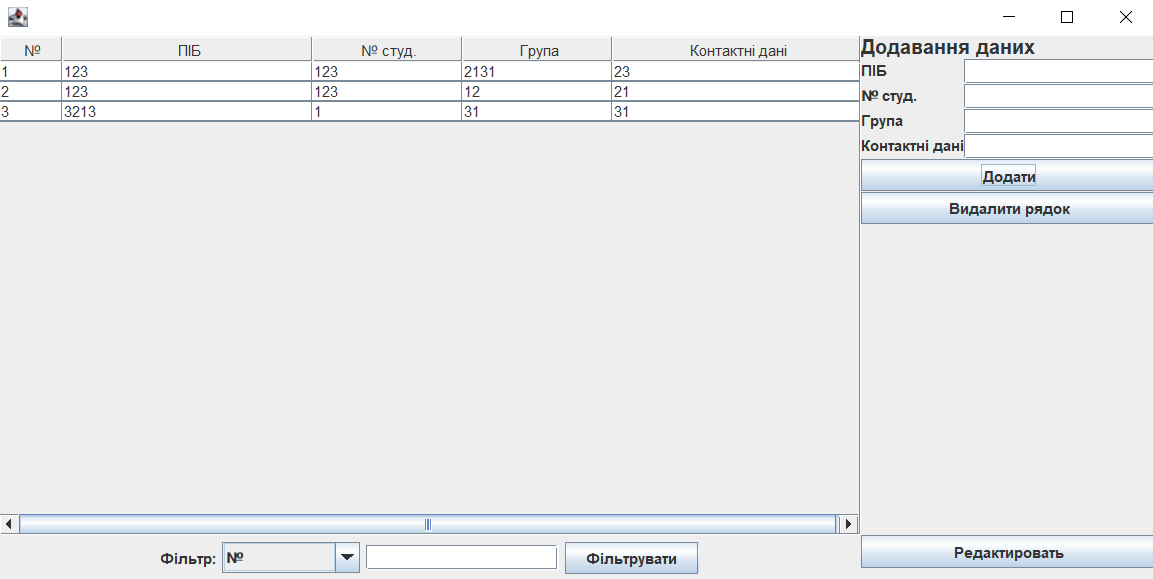


Рис 1. Інтерфейс програми

1. Доопрацювати код моделі даних таблиці так, щоб нумерація в першому стовпці реалізовувалася через модель, і при видаленні елемента із середини списку не було пропуску в нумерації.

Спробуємо видалити один елемент:

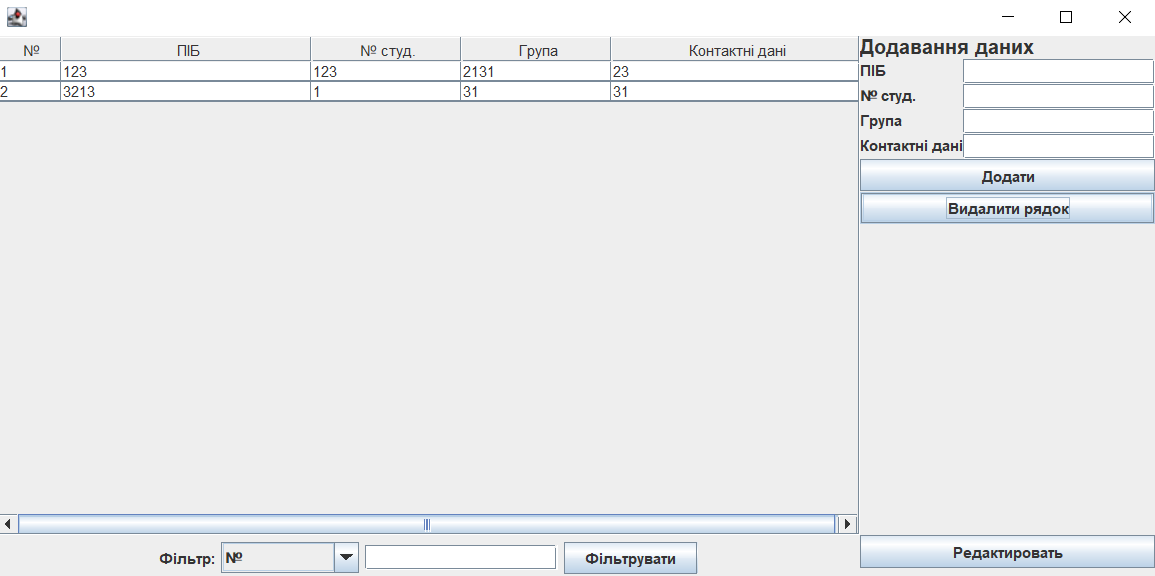


Рис 2. Ілюстрація роботи видалення

1. Створимо панель для фільтрації значень в таблиці за вказаним фрагментом.

|  |  |
| --- | --- |
|  |  |
| Рис 3а. Ілюстрація роботи фільтру | Рис 3б. Ілюстрація роботи фільтру |

1. Додати до додатку функціонал редагування даних обраного користувачем рядку в таблиці.

Мною було додано редагування. В цій ілюстрації було змінено декілька полів.

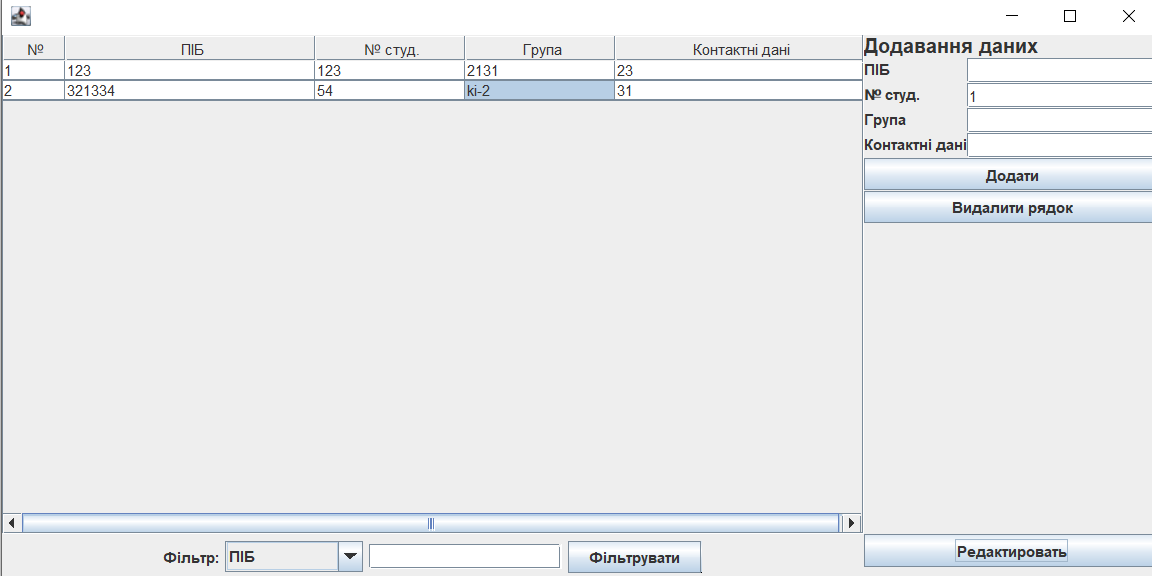


Рис 4. Ілюстрація додатку

1. Застосувати компонент JFormattedTextField. Удосконалити проект: по можливості позбавитись повторюваності коду, за-класти можливості подальшого розвитку проекту (заміна інтерфейсу користувача, бізнес-логіки і т.д.), оптимізувати код. Удосконалення виконувати над копією проекту. Звернути увагу на такі поняття, як інтерфейс, посилання на метод і т.д.

Для початку застосуємо JFormattedTextField. Ілюстрація роботи показана нижче. Була спроба створити масив даних з кожним полем name, але поле робить автозаміну на 1.

|  |  |
| --- | --- |
|  |  |
| Рис 5а. Ілюстрація додатку | Рис 5б. Ілюстрація додатку |

Для початку в проекті були винесені головні налаштування у клас Main. Зміни зображені нижче.

public static int width = 800;

public static int height = 600;

public static Font font = new Font("Arial", Font.BOLD, 16);

Були додані зміни до скролл панелі.

scrollPane.setPreferredSize(new Dimension(t1.getPreferredSize().width, 400));

Далі для більш зручної зміни інтерфейсу було додано можливість редагування columnWidth. Тобто, при зміні цієї статичної зміни проходить і зміна інтерфейсу. Також це дозволило скоротити код: тепер працює цикл замість ручного введення розміру колон.

for(int i = 0; i < columnWidth.length; i++)

t1.getColumnModel().getColumn(i).setPreferredWidth(columnWidth[i]);

Також була додана змінна localizationUA, що дозволяє зробити редагування інтефейсу більш швидким, та зміну мови більш лаконічною для програміста.

Код до всієї роботи знаходиться у додатках А і Б.

**Висновок**

В рамках лабораторної роботи була розроблена програма для керування і відображення інформації про студентів. Основна функціональність була розширена та вдосконалена згідно з вимогами завдань. Основні зміни та стилізація були внесені в наступні аспекти:

**1. Додавання стовпця з контактними даними:** Був доданий п'ятий стовпець до таблиці для відображення контактних даних студентів. Додавання даних до таблиці було впорядковане через модель `TableModel`.

**2. Оптимізація моделі даних:** Модель даних таблиці була оптимізована для правильної нумерації студентів у першому стовпці. Це дозволило уникнути пропусків у нумерації при видаленні елементів.

**3. Фільтрація даних в таблиці:** Була створена панель для фільтрації значень в таблиці. Для фільтрації використовувався RowFilter з використанням компонента JComboBox для вибору стовпця та JTextField для введення користувачем значень фільтра.

**4. Редагування даних студентів:** Доданий функціонал редагування даних обраного користувачем рядку в таблиці. Користувач може вносити зміни в дані, натискаючи на відповідну комірку таблиці.

**5. Оптимізація та використання JFormattedTextField:** Для полегшення введення та валідації числових даних був використаний JFormattedTextField. Це дозволяє користувачам вводити числа в обраному форматі.

Загалом, завдяки цим змінам та додаткам програма стала більш функціональною та користувацьки зручною, дозволяючи легко вводити, редагувати, фільтрувати та відображати інформацію про студентів.

Додаток А

*Лістинг файлу «Сonverter.java»*

import javax.swing.\*;

import javax.swing.JFormattedTextField;

import javax.swing.text.NumberFormatter;

import javax.swing.table.TableRowSorter;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.text.NumberFormat;

public class Main {

public static int width = 800;

public static int height = 600;

public static Font font = new Font("Arial", Font.BOLD, 16);

public static int columnNumber = 15;

public static int[] columnWidth = {

50, 200, 120, 120, 200

};

public static String[] localizationUA = {

"Додавання даних",

"Додати",

"ПІБ",

"№ студ.", //3

"Контактні дані",

"Група",

"Фільтр:", //6

"Фільтрувати",

"Видалити рядок",

"Редагувати", //9

"Закінчити редагування",

};

public static void main(String[] args) {

JFrame frame = new JFrame();

frame.setSize(width, height);

frame.setLocationRelativeTo(null);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setLayout(new GridBagLayout());

GridBagConstraints c = new GridBagConstraints();

TableModel tm = new TableModel();

JTable t1 = new JTable(tm);

DefaultCellEditor cellEditor = new DefaultCellEditor(new JTextField());

t1.setDefaultEditor(Object.class, cellEditor);

t1.setAutoResizeMode(JTable.AUTO\_RESIZE\_OFF);

for(int i = 0; i < columnWidth.length; i++)

t1.getColumnModel().getColumn(i).setPreferredWidth(columnWidth[i]);

JScrollPane scrollPane = new JScrollPane(t1);

scrollPane.setPreferredSize(new Dimension(t1.getPreferredSize().width, 400));

c.gridx = 0;

c.gridy = 0;

c.gridwidth = 1;

c.gridheight = 2;

c.fill = GridBagConstraints.BOTH;

frame.add(scrollPane, c);

JPanel panel1 = new JPanel();

JLabel panel\_name = new JLabel(localizationUA[0]);

panel\_name.setFont(font);

panel1.setLayout(new GridBagLayout());

JButton add\_data = new JButton(localizationUA[1]);

JLabel fio\_lab = new JLabel(localizationUA[2]);

final JTextField fio\_tf = new JTextField(columnNumber);

JLabel id\_lab = new JLabel(localizationUA[3]);

NumberFormatter formatter = new NumberFormatter(NumberFormat.getIntegerInstance());

formatter.setValueClass(Integer.class); // Укажите класс значений (в данном случае Integer)

formatter.setMinimum(0); // Минимальное значение

formatter.setMaximum(Integer.MAX\_VALUE); // Максимальное значение

JFormattedTextField id\_tf = new JFormattedTextField(formatter);

id\_tf.setColumns(columnNumber);

JLabel number\_lab = new JLabel(localizationUA[4]);

final JTextField number\_tf = new JTextField(columnNumber);

JLabel group\_lab = new JLabel(localizationUA[5]);

final JTextField group\_tf = new JTextField(columnNumber);

// Додайте фільтраційну панель

JPanel filterPanel = new JPanel();

JLabel filterLabel = new JLabel(localizationUA[6]);

JComboBox<String> filterColumn = new JComboBox<>(tm.getColumnNames());

JTextField filterText = new JTextField(columnNumber);

JButton filterButton = new JButton(localizationUA[7]);

filterPanel.add(filterLabel);

filterPanel.add(filterColumn);

filterPanel.add(filterText);

filterPanel.add(filterButton);

c.gridx = 0;

c.gridy = 2;

c.gridwidth = 1;

frame.add(filterPanel, c);

// Створіть фільтрувальник для таблиці

DefaultRowSorter<TableModel, Integer> sorter = new TableRowSorter<>(tm);

t1.setRowSorter(sorter);

filterButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String selectedColumn = filterColumn.getSelectedItem().toString();

String filterValue = filterText.getText();

sorter.setRowFilter(RowFilter.regexFilter(filterValue, tm.getColumnIndexByName(selectedColumn)));

}

});

c.gridwidth = 1;

c.gridheight = 1;

c.fill = GridBagConstraints.HORIZONTAL;

c.gridwidth = 2;

c.gridx = 0;

c.gridy = 0;

panel1.add(panel\_name, c);

c.gridwidth = 1;

c.gridx = 0;

c.gridy = 1;

c.anchor = GridBagConstraints.NORTH;

panel1.add(fio\_lab, c);

c.gridx = 1;

c.gridy = 1;

panel1.add(fio\_tf, c);

c.gridx = 0;

c.gridy = 2;

panel1.add(id\_lab, c);

c.gridx = 1;

c.gridy = 2;

panel1.add(id\_tf, c);

c.gridx = 0;

c.gridy = 3;

panel1.add(group\_lab, c);

c.gridx = 1;

c.gridy = 3;

panel1.add(group\_tf, c);

c.gridx = 0;

c.gridy = 4;

panel1.add(number\_lab, c);

c.gridx = 1;

c.gridy = 4;

panel1.add(number\_tf, c);

c.gridx = 0;

c.gridy = 5;

c.gridwidth = 2;

panel1.add(add\_data, c);

c.gridx = 1;

c.gridy = 0;

c.gridwidth = 1;

frame.add(panel1, c);

JButton select\_remove = new JButton(localizationUA[8]);

c.gridx = 1;

c.gridy = 1;

c.gridwidth = 1;

c.gridheight = 1;

frame.add(select\_remove, c);

add\_data.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String[] st = {Integer.toString(tm.getRowCount() + 1),

fio\_tf.getText(), id\_tf.getText(), group\_tf.getText(), number\_tf.getText()};

tm.addData(st);

fio\_tf.setText("");

id\_tf.setText("");

group\_tf.setText("");

number\_tf.setText("");

t1.revalidate();

t1.repaint();

}

});

final boolean[] editMode = {false}; // Declare as an array to make it effectively final

JButton editButton = new JButton(localizationUA[9]);

c.gridx = 1;

c.gridy = 2;

frame.add(editButton, c);

editButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

editMode[0] = !editMode[0];

if (editMode[0]) {

t1.setEnabled(true);

editButton.setText(localizationUA[10]);

} else {

t1.setEnabled(false);

editButton.setText(localizationUA[9]);

}

}

});

// Rest of the code remains the same

// Добавьте слушатель для редактирования данных в таблице

t1.addMouseListener(new MouseAdapter() {

@Override

public void mouseClicked(MouseEvent e) {

if (editMode[0] && e.getClickCount() == 2) {

int selectedRow = t1.getSelectedRow();

int selectedColumn = t1.getSelectedColumn();

if (selectedRow != -1 && selectedColumn != -1) {

t1.editCellAt(selectedRow, selectedColumn);

t1.getEditorComponent().requestFocus();

}

}

}

});

select\_remove.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

int selectedRow = t1.getSelectedRow();

if (selectedRow != -1) {

int modelRow = t1.convertRowIndexToModel(selectedRow);

tm.removeRow(modelRow);

tm.fireTableDataChanged(); // Обновить таблицу

}

}

});

t1.setRowSelectionAllowed(false);

t1.setColumnSelectionAllowed(false);

t1.setCellSelectionEnabled(true);

frame.pack();

frame.setVisible(true);

}

}

*Додаток Б*

*Лістинг файлу «Conversion.java»*

import javax.swing.table.AbstractTableModel;

import java.util.ArrayList;

import javax.swing.RowFilter;

public class TableModel extends AbstractTableModel {

protected ArrayList<String[]> data;

protected int columnCount = 5;

public TableModel() {

data = new ArrayList<String[]>();

}

@Override

public int getRowCount() {

return data.size();

}

@Override

public int getColumnCount() {

return columnCount;

}

@Override

public Object getValueAt(int rowIndex, int columnIndex) {

String[] row = data.get(rowIndex);

return row[columnIndex];

}

// Добавьте следующий метод для разрешения редактирования ячейки

@Override

public boolean isCellEditable(int rowIndex, int columnIndex) {

// Возвращайте true только для тех столбцов, которые вы хотите разрешить редактировать.

return columnIndex != 0; // Например, не разрешайте редактировать столбец "№"

}

// Добавьте следующий метод для сохранения изменений в ячейке

@Override

public void setValueAt(Object value, int rowIndex, int columnIndex) {

String[] row = data.get(rowIndex);

row[columnIndex] = value.toString();

fireTableCellUpdated(rowIndex, columnIndex);

}

public void removeRow(int rowIndex) {

if (rowIndex >= 0 && rowIndex < data.size()) {

data.remove(rowIndex);

fireTableRowsDeleted(rowIndex, rowIndex);

changeNumbers(rowIndex);

}

}

public void addData(String[] row) {

data.add(row);

fireTableDataChanged();

}

public String getColumnName(int columnIndex) {

switch (columnIndex) {

case 0:

return "№";

case 1:

return Main.localizationUA[2];

case 2:

return Main.localizationUA[3];

case 3:

return Main.localizationUA[5];

case 4:

return Main.localizationUA[4];

}

return "";

}

public void changeNumbers(int rowDeleteIndex) {

for (int i = rowDeleteIndex; i < data.size(); i++) {

String currentId = data.get(i)[0];

data.get(i)[0] = Integer.toString(i + 1);

}

}

public int getColumnIndexByName(String columnName) {

for (int i = 0; i < getColumnCount(); i++) {

if (getColumnName(i).equals(columnName)) {

return i;

}

}

return -1;

}

public String[] getColumnNames() {

String[] columnNames = new String[getColumnCount()];

for (int i = 0; i < getColumnCount(); i++) {

columnNames[i] = getColumnName(i);

}

return columnNames;

}

}