МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Российский государственный университет им. А.Н. Косыгина (Технологии. Дизайн. Искусство)»

Кафедра автоматизированных систем обработки информации и управления

Отчет по лабораторной работе № 4

по дисциплине «Программирование»

Тема: «Табличный компонент»

Выполнил: Гоянов Р.Р., группа МВА-220

Проверил: Самойлова Т.А.

Цель работы

Целью работы является ознакомление со средствами вывода, редактирования и сортировки табличных данных

Задание

Отобразить данные класса, созданного в лабораторной работе №1 (Pen), в виде таблицы. Количество полей в классе должно быть не меньше 5, хотя бы одно из них - типа Boolean. Предусмотреть возможность добавления и сортировки данных в таблице. Добавить 2 произвольных отрисовщика.

Код класса Pen

package com.goyanov.lab4;

public class Pen

{

enum Color

{

СИНИЙ, ЧЁРНЫЙ, КРАСНЫЙ, ЗЕЛЁНЫЙ;

}

private String name;

private String fabricator;

private Color color;

private boolean automatic;

private double cost;

public Pen(String name, String fabricator, Color color, boolean automatic, double cost)

{

this.name = name;

this.fabricator = fabricator;

this.color = color;

this.automatic = automatic;

this.cost = cost;

}

public String getName()

{

return name;

}

public String getFabricator()

{

return fabricator;

}

public Color getColor()

{

return color;

}

public boolean isAutomatic()

{

return automatic;

}

public double getCost()

{

return cost;

}

}

Код программы

package com.goyanov.lab4;

import javax.swing.\*;

import javax.swing.table.DefaultTableCellRenderer;

import javax.swing.table.DefaultTableModel;

import javax.swing.table.TableModel;

import javax.swing.table.TableRowSorter;

import java.awt.\*;

import java.util.Arrays;

public class Main

{

static class CostRenderer extends DefaultTableCellRenderer

{

@Override

public Component getTableCellRendererComponent(JTable table, Object value, boolean isSelected, boolean hasFocus, int row, int column)

{

Double cost = (Double) table.getValueAt(row, column);

Component component = super.getTableCellRendererComponent(table, value, isSelected, hasFocus, row, column);

if (cost > 50) component.setBackground(new Color(227, 38, 38));

else if (cost > 10) component.setBackground(new Color(239, 234, 127));

else component.setBackground(new Color(70, 183, 55));

return component;

}

}

static class ColorRenderer extends DefaultTableCellRenderer

{

@Override

public Component getTableCellRendererComponent(JTable table, Object value, boolean isSelected, boolean hasFocus, int row, int column)

{

Pen.Color color = (Pen.Color) table.getValueAt(row, column);

Component component = super.getTableCellRendererComponent(table, value, isSelected, hasFocus, row, column);

if (color == Pen.Color.*ЧЁРНЫЙ*) component.setBackground(new Color(180, 180, 180));

else if (color == Pen.Color.*СИНИЙ*) component.setBackground(new Color(90, 202, 238));

else if (color == Pen.Color.*КРАСНЫЙ*) component.setBackground(new Color(255, 138, 138));

else component.setBackground(new Color(143, 255, 115));

return component;

}

}

public static void main(String[] args)

{

Pen[] pen = new Pen[10];

pen[0] = new Pen("Ручка шариковая", "Erich Krause", Pen.Color.*ЧЁРНЫЙ*, false, 4.22);

pen[1] = new Pen("Ручка автоматическая", "Fellowes", Pen.Color.*ЗЕЛЁНЫЙ*, true, 12.23);

pen[2] = new Pen("Ручка PEN-2011", "Комус", Pen.Color.*ЧЁРНЫЙ*, true, 32.13);

pen[3] = new Pen("Ручка красная", "BiC", Pen.Color.*КРАСНЫЙ*, false, 6.99);

pen[4] = new Pen("Ручка синяя", "BiC", Pen.Color.*СИНИЙ*, false, 5.89);

pen[5] = new Pen("Ручка-Крикучка", "Комус", Pen.Color.*ЧЁРНЫЙ*, true, 59.99);

pen[6] = new Pen("Автоматическая ручка AU-20", "Auschan", Pen.Color.*ЗЕЛЁНЫЙ*, false, 45.99);

pen[7] = new Pen("Ручка зелёная", "EI Casco", Pen.Color.*ЗЕЛЁНЫЙ*, false, 12.22);

pen[8] = new Pen("Шариковая ручка", "Fellowes", Pen.Color.*КРАСНЫЙ*, true, 21.89);

pen[9] = new Pen("Ручка PILOT", "Moleskine", Pen.Color.*СИНИЙ*, true, 13.22);

JFrame frame = new JFrame("Таблица ручек");

frame.setSize(870, 280);

frame.setResizable(false);

frame.setDefaultCloseOperation(WindowConstants.*EXIT\_ON\_CLOSE*);

JPanel mainPanel = new JPanel();

mainPanel.setLayout(new BorderLayout());

Object[] headers = {"Название", "Производитель", "Цвет", "Автоматическая", "Цена, руб."};

Object[][] content = new Object[10][];

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

{

content[i] = new Object[]{pen[i].getName(), pen[i].getFabricator(), pen[i].getColor(), pen[i].isAutomatic(), pen[i].getCost()};

}

DefaultTableModel model = new DefaultTableModel(content, headers)

{

@Override

public Class<?> getColumnClass(int columnIndex)

{

return getValueAt(0, columnIndex).getClass();

}

@Override

public boolean isCellEditable(int row, int column)

{

return false;

}

};

JTable table = new JTable(model);

RowSorter<TableModel> sorter = new TableRowSorter<>(model);

table.setRowSorter(sorter);

table.setPreferredScrollableViewportSize(new Dimension(250,80));

JScrollPane scrollPane = new JScrollPane(table);

table.setDefaultRenderer(Double.class, new CostRenderer());

table.setDefaultRenderer(Pen.Color.class, new ColorRenderer());

mainPanel.add(scrollPane, BorderLayout.*CENTER*);

JPanel functionalPanel = new JPanel();

functionalPanel.setLayout(new FlowLayout(FlowLayout.*CENTER*, 10, 5));

JTextField name = new JTextField(13);

functionalPanel.add(name);

JTextField fabricator = new JTextField(13);

functionalPanel.add(fabricator);

JTextField color = new JTextField(13);

functionalPanel.add(color);

JCheckBox automatic = new JCheckBox();

functionalPanel.add(automatic);

JTextField cost = new JTextField(5);

functionalPanel.add(cost);

JButton addButton = new JButton("Добавить");

addButton.addActionListener((event)->

{

try {

model.addRow(new Object[]{name.getText(), fabricator.getText(), Pen.Color.*valueOf*(color.getText().toUpperCase()), automatic.isSelected(), Double.*parseDouble*(cost.getText())});

} catch (Exception e) { System.*out*.println("Ошибка в переданных аргументах. Проверьте данные."); }

});

functionalPanel.add(addButton);

JButton clearButton = new JButton("Очистить");

clearButton.addActionListener((event) ->

{

Arrays.*stream*(functionalPanel.getComponents()).filter(e -> e instanceof JTextField).map(JTextField.class::cast).forEach(e -> e.setText(""));

automatic.setSelected(false);

});

functionalPanel.add(clearButton);

mainPanel.add(functionalPanel, BorderLayout.*SOUTH*);

frame.add(mainPanel);

frame.setVisible(true);

}

}

Результаты выполнения программы

Результаты выполнения программы представлены на рис. 1-9

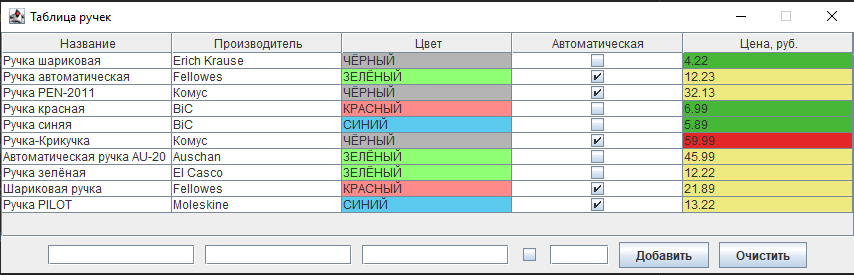


рис. 1 - Начальное состояние таблицы

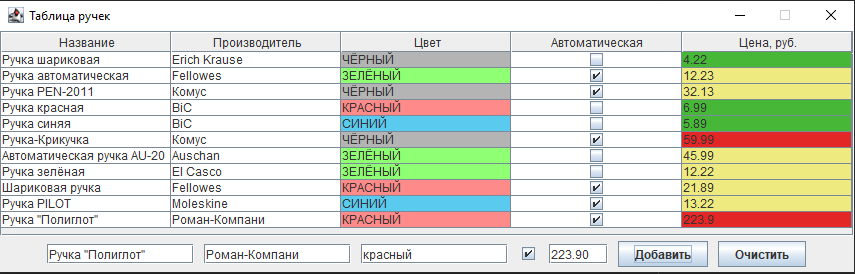


рис. 2 - Состояние таблицы после добавления новой строки

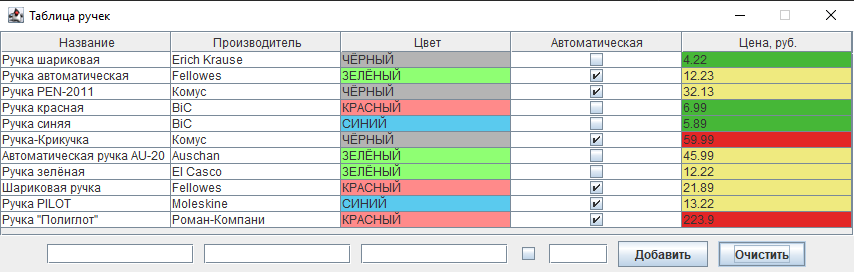


рис. 3 - Состояние полей для ввода значений после нажатия на кнопку “Очистить”

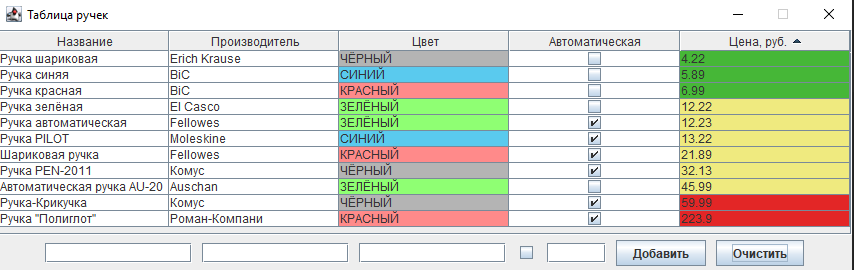


рис. 4 - Сортировка по цене

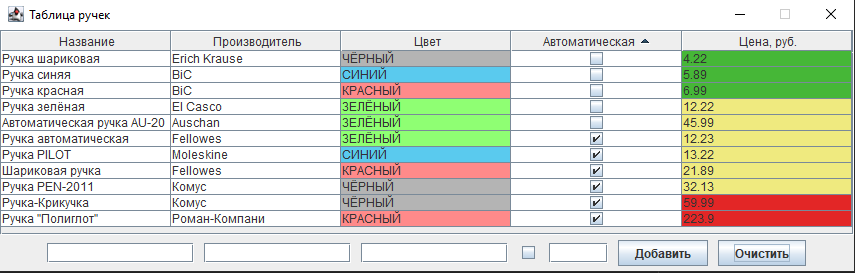


рис. 5 - Сортировка по критерию “Автоматическая”

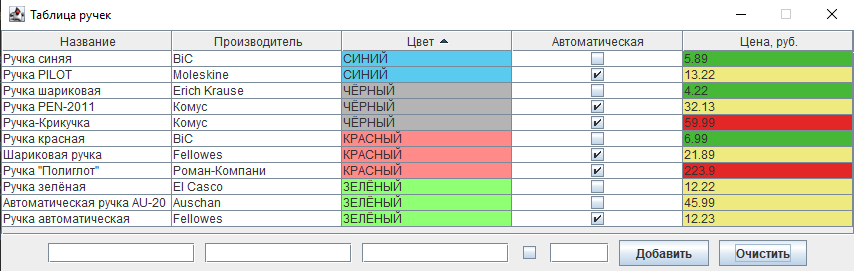


рис. 6 - Сортировка по цвету

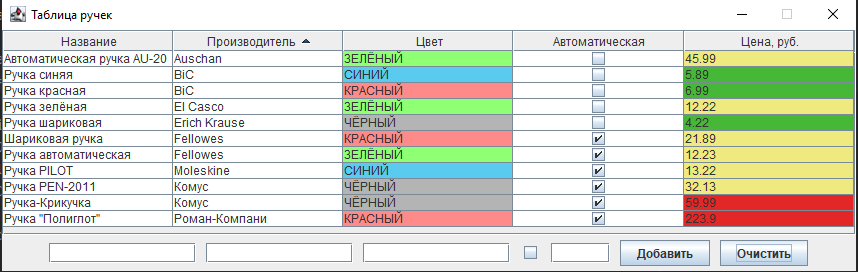


рис. 7 - Сортировка по производителю

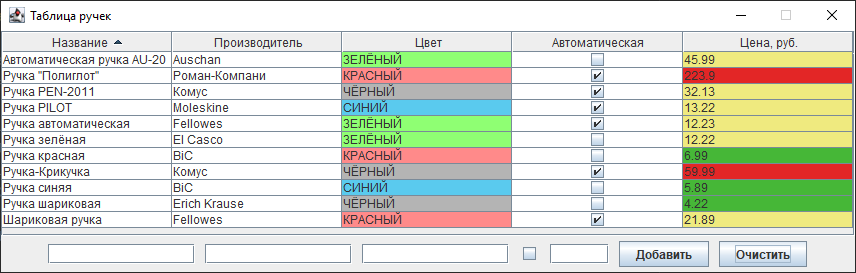


рис. 8 - Сортировка по названию

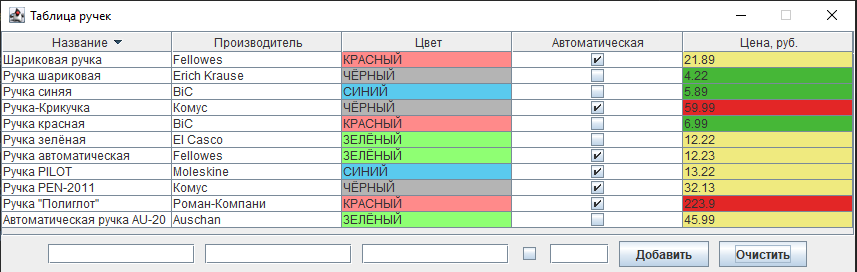


рис. 9 - Сортировка работает как по возрастанию, так и по убыванию