Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №5

По дисциплине: «Современные системы программирования»

Тема: «Таблицы»

Выполнил:

Студент 4 курса

Группы ИИ-16 (2)

Пешко А.С.

Проверил:

Коренкович О. Г.

Брест 2021

**Цель работы:** освоить работу с двумерными массивами; познакомиться с классом StringGrid; научиться программировать алгоритмы поиска наибольшего и наименьшего значения, научиться обрабатывать исключительные ситуации.

**Ход работы**

**Задание:**

Создать программу, выполняющую:

– заполнение таблицы случайными целыми числами в диапазоне

от -20 до 20;

– возможность изменения данных вручную;

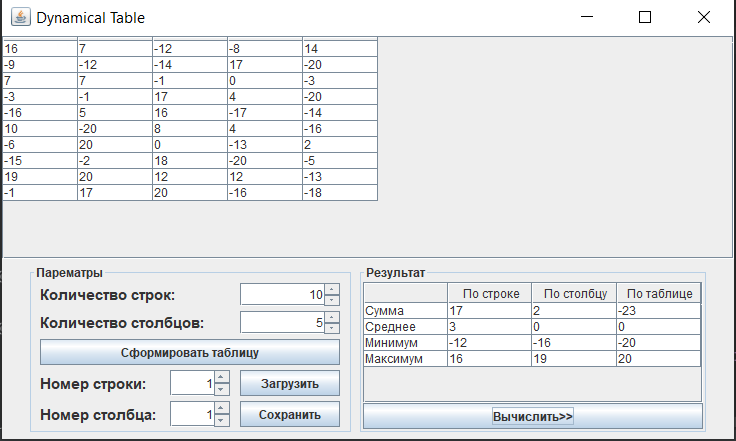
– нахождение сумм, средних, минимальных и максимальных значений по строкам, столбцам и таблице в целом с заполнением таблицы результатов;

- загрузка из xml-файла в таблицу данных документа.

**Код программы:**

public class TableFrame extends JFrame {  
  
 int startVal\_mainRows = 10, // кол-во строк  
 int startVal\_mainCols = 5, // кол-во столбцов  
 int startVal\_minRand = -20, // произв. число минимальное значение  
 int startVal\_maxRand = 20;// произв. число макснимальное значение  
  
 private JScrollPane mainTableSP; // Панель прокрутки для главной таблицы  
 private JTable mainTable; //Главная таблица  
  
 private JPanel bottomPanel; // Нижняя панель с кнопками и таблицей результатов  
  
 // Параметры Инициализации таблицы  
 private JLabel colCountLabel;  
 private JSpinner colCountSelector;  
  
 private JLabel rowCountLabel;  
 private JSpinner rowCountSelector;  
  
 // Кнопка Генерация модели таблицы  
 private JButton generateTM;  
  
 // Параметры расчета по таблице  
 private JLabel colNumberLabel;  
 private JSpinner colNumberSelector;  
  
 private JLabel rowNumberLabel;  
 private JSpinner rowNumberSelector;  
  
 // Кнопка получения результатов  
 private JButton calculate;  
 // Кнопка сохранения в xml  
 private JButton load;  
 private JButton save;  
 private JScrollPane resultTableSP; // Панель прокрутки для таблицы с результатами  
 private JTable resultTable; //Таблица с результатами  
  
 public TableFrame() {  
 this.setDefaultCloseOperation(EXIT\_ON\_CLOSE);  
 this.setLayout(new BorderLayout());  
 this.setSize(750, 450);  
 this.setMinimumSize(getSize());  
 initComponents();  
 }  
  
 private ValuesTableModel initValuesTM(String[][] values) {  
 return new ValuesTableModel(values);  
 }  
  
 private ValuesTableModel initValuesTM(int rowsN, int colsN) {  
 return this.initValuesTM(rowsN, colsN, startVal\_minRand, startVal\_maxRand);  
 }  
  
 private ValuesTableModel initValuesTM(int \_rowsN, int \_colsN, int minRand, int maxRand) {  
  
 int rowsN = Math.max(\_rowsN, 0);  
 int colsN = Math.max(\_colsN, 0);  
  
 String[] colNames = new String[colsN];  
 String[][] colValues = new String[rowsN][colsN];  
  
 Random random = new Random();  
  
 for (int row = 0; row < rowsN; row++) {  
 colValues[row] = new String[colsN];  
 for (int col = 0; col < colsN; col++) {  
 colValues[row][col] = String.valueOf((random.nextInt((maxRand + 1) - minRand) + minRand));  
 }  
 }  
  
 return new ValuesTableModel(colNames, colValues);  
 }  
  
 private void initComponents() {  
  
 mainTableSP = new JScrollPane();  
 mainTable = new JTable(initValuesTM(startVal\_mainRows, startVal\_mainCols, startVal\_minRand, startVal\_maxRand));  
 mainTableSP.setViewportView(mainTable);//создает область просмотра главной таблицы  
 mainTable.getTableHeader().setReorderingAllowed(false); // фиксация табличного пространства (запрет перемещения столбцов) главной таблицы  
  
 mainTable.setShowGrid(true);  
 mainTable.setAutoResizeMode(JTable.AUTO\_RESIZE\_OFF);  
  
  
 ListSelectionModel mainTableSelectionModel = mainTable.getSelectionModel();  
 mainTableSelectionModel.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);  
  
 mainTable.setCellSelectionEnabled(true);  
 mainTable.setRowSelectionAllowed(true);  
 mainTable.setColumnSelectionAllowed(true);  
  
 mainTableSelectionModel.addListSelectionListener(new ListSelectionListener() {  
 public void valueChanged(ListSelectionEvent e) {  
 mainTable\_handleSelectionEvent(e);  
 }  
 });  
  
 mainTable.getColumnModel().getSelectionModel().addListSelectionListener(new ListSelectionListener() {  
 @Override  
 public void valueChanged(ListSelectionEvent e) {  
 mainTable\_handleSelectionEvent(e);  
 }  
 });  
  
 this.add(mainTableSP, BorderLayout.CENTER);  
  
 bottomPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 5, 5));  
 bottomPanel.setBorder(BorderFactory.createEmptyBorder(0, 5, 0, 5));  
  
 Dimension bottomPanelDimension = new Dimension(700, 180);  
 bottomPanel.setSize(bottomPanelDimension);  
 bottomPanel.setPreferredSize(bottomPanelDimension);  
 bottomPanel.setMinimumSize(bottomPanelDimension);  
 bottomPanel.setMaximumSize(bottomPanelDimension);  
  
   
 bottomPanel.add(new JPanel() {{  
 rowCountLabel = new JLabel();  
 rowCountSelector = new JSpinner();  
 rowCountSelector.setValue(startVal\_mainRows);  
 colCountLabel = new JLabel();  
 colCountSelector = new JSpinner();  
 colCountSelector.setValue(startVal\_mainCols);  
 generateTM = new JButton();  
 rowNumberLabel = new JLabel();  
 rowNumberSelector = new JSpinner();  
 load = new JButton();  
 colNumberLabel = new JLabel();  
 colNumberSelector = new JSpinner();  
 save = new JButton();  
  
 this.setLayout(new GridBagLayout());  
 this.setBorder(new TitledBorder("Парематры"));  
  
 Dimension optionsPanelDimension = new Dimension(325, bottomPanelDimension.height - 10);  
 this.setSize(optionsPanelDimension);  
 this.setPreferredSize(optionsPanelDimension);  
 this.setMinimumSize(optionsPanelDimension);  
 this.setMaximumSize(optionsPanelDimension);  
  
 ((GridBagLayout) getLayout()).columnWidths = new int[]{130, 70, 100, 0};  
 ((GridBagLayout) getLayout()).rowHeights = new int[]{28, 28, 28, 0, 24, 24};  
  
 //---- rowCountLabel ----  
 rowCountLabel.setText("Количество строк: ");  
 rowCountLabel.setFont(rowCountLabel.getFont().deriveFont(rowCountLabel.getFont().getStyle() | Font.BOLD, rowCountLabel.getFont().getSize() + 3f));  
 add(rowCountLabel, new GridBagConstraints(0, 0, 2, 1, 0.0, 0.0,  
 GridBagConstraints.CENTER, GridBagConstraints.BOTH,  
 new Insets(0, 0, 5, 10), 0, 0));  
 add(rowCountSelector, new GridBagConstraints(2, 0, 1, 1, 0.0, 0.0,  
 GridBagConstraints.CENTER, GridBagConstraints.BOTH,  
 new Insets(0, 0, 5, 0), 0, 0));  
  
 //---- colCountLabel ----  
 colCountLabel.setText("Количество столбцов: ");  
 colCountLabel.setFont(colCountLabel.getFont().deriveFont(colCountLabel.getFont().getStyle() | Font.BOLD, colCountLabel.getFont().getSize() + 3f));  
 add(colCountLabel, new GridBagConstraints(0, 1, 2, 1, 0.0, 0.0,  
 GridBagConstraints.CENTER, GridBagConstraints.BOTH,  
 new Insets(0, 0, 5, 10), 0, 0));  
 add(colCountSelector, new GridBagConstraints(2, 1, 1, 1, 0.0, 0.0,  
 GridBagConstraints.CENTER, GridBagConstraints.BOTH,  
 new Insets(0, 0, 5, 0), 0, 0));  
  
 //---- generateTM ----  
 add(generateTM, new GridBagConstraints(0, 2, 3, 1, 0.0, 0.0,  
 GridBagConstraints.CENTER, GridBagConstraints.BOTH,  
 new Insets(0, 0, 5, 0), 0, 0));  
 generateTM.setAction(new AbstractAction("Сформировать таблицу") {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 generateTM\_handleAction(e);  
 }  
 });  
  
 rowNumberLabel.setText("Номер строки: ");  
 rowNumberLabel.setFont(rowNumberLabel.getFont().deriveFont(rowNumberLabel.getFont().getStyle() | Font.BOLD, rowNumberLabel.getFont().getSize() + 3f));  
 add(rowNumberLabel, new GridBagConstraints(0, 4, 1, 1, 0.0, 0.0,  
 GridBagConstraints.CENTER, GridBagConstraints.BOTH,  
 new Insets(0, 0, 5, 10), 0, 0));  
 add(rowNumberSelector, new GridBagConstraints(1, 4, 1, 1, 0.0, 0.0,  
 GridBagConstraints.CENTER, GridBagConstraints.BOTH,  
 new Insets(0, 0, 5, 10), 0, 0));  
  
 add(load, new GridBagConstraints(2, 4, 1, 1, 0.0, 0.0,  
 GridBagConstraints.CENTER, GridBagConstraints.BOTH,  
 new Insets(0, 0, 5, 0), 0, 0));  
 load.setAction(new AbstractAction("Загрузить") {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 load\_handleAction(e);  
 }  
 });  
  
 colNumberLabel.setText("Номер столбца: ");  
 colNumberLabel.setFont(colNumberLabel.getFont().deriveFont(colNumberLabel.getFont().getStyle() | Font.BOLD, colNumberLabel.getFont().getSize() + 3f));  
 add(colNumberLabel, new GridBagConstraints(0, 5, 1, 1, 0.0, 0.0,  
 GridBagConstraints.CENTER, GridBagConstraints.BOTH,  
 new Insets(0, 0, 0, 10), 0, 0));  
 add(colNumberSelector, new GridBagConstraints(1, 5, 1, 1, 0.0, 0.0,  
 GridBagConstraints.CENTER, GridBagConstraints.BOTH,  
 new Insets(0, 0, 0, 10), 0, 0));  
  
 //---- CalculateCol ----  
 add(save, new GridBagConstraints(2, 5, 1, 1, 0.0, 0.0,  
 GridBagConstraints.CENTER, GridBagConstraints.BOTH,  
 new Insets(0, 0, 0, 0), 0, 0));  
 save.setAction(new AbstractAction("Сохранить") {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 save\_handleAction(e);  
 }  
 });  
  
 }}, BorderLayout.WEST);  
  
 bottomPanel.add(new JPanel() {{  
 this.setBorder(new TitledBorder("Результат"));  
 this.setLayout(new BorderLayout());  
  
 Dimension resultPanelDimension = new Dimension(350, bottomPanelDimension.height - 10);  
 this.setSize(resultPanelDimension);  
 this.setPreferredSize(resultPanelDimension);  
 this.setMinimumSize(resultPanelDimension);  
 this.setMaximumSize(resultPanelDimension);  
  
 resultTableSP = new JScrollPane();  
 resultTable = new JTable(new ResultTableModel());  
 resultTableSP.setViewportView(resultTable);//создает область просмотра таблицы результатов  
 resultTable.getTableHeader().setReorderingAllowed(false); // фиксация табличного пространства (запрет перемещения столбцов) таблицы результатов  
 calculate = new JButton();  
  
 this.add(resultTableSP, BorderLayout.CENTER);  
  
 calculate.setAction(new AbstractAction("Вычислить>>") {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 calculate\_handleAction(e);  
 }  
 });  
  
 this.add(calculate, BorderLayout.SOUTH);  
  
 }}, BorderLayout.EAST);  
  
 this.add(bottomPanel, BorderLayout.SOUTH);  
 }  
  
 private void load\_handleAction(ActionEvent e) {  
 DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();  
 try {  
 String[][] result;  
 dbf.setFeature(XMLConstants.FEATURE\_SECURE\_PROCESSING, true);  
 DocumentBuilder db = dbf.newDocumentBuilder();  
 Document doc = db.parse(new File("./saved\_table.xml"));  
 doc.getDocumentElement().normalize();  
 NodeList row\_list = doc.getElementsByTagName("row");  
 result = new String[row\_list.getLength()][];  
 for (int row = 0; row < row\_list.getLength(); row++) {  
 Node row\_node = row\_list.item(row);  
 if (row\_node.getNodeType() == Node.ELEMENT\_NODE) {  
 Element row\_element = (Element) row\_node;  
 int row\_element\_id = (Integer.parseInt(row\_element.getAttribute("id")) - 1);  
 NodeList cell\_list = row\_element.getElementsByTagName("cell");  
 result[row\_element\_id] = new String[cell\_list.getLength()];  
 for (int cell = 0; cell < cell\_list.getLength(); cell++) {  
 Node cell\_node = cell\_list.item(cell);  
 if (cell\_node.getNodeType() == Node.ELEMENT\_NODE) {  
 Element cell\_element = (Element) cell\_node;  
 int cell\_element\_id = (Integer.parseInt(cell\_element.getAttribute("id")) - 1);  
 result[row\_element\_id][cell\_element\_id] = cell\_element.getTextContent();  
 }  
 }  
 }  
 }  
 mainTable\_updateTM(this.initValuesTM(result));  
 } catch (ParserConfigurationException | SAXException | IOException ex) {  
 System.err.println("Can't load xml data: " + ex.getLocalizedMessage());  
 }  
 }  
  
 private void save\_handleAction(ActionEvent e) {  
 final String[][] values = ((ValuesTableModel) mainTable.getModel()).getValues();  
 try {  
 DocumentBuilderFactory dFact = DocumentBuilderFactory.newInstance();  
 DocumentBuilder build = dFact.newDocumentBuilder();  
 Document doc = build.newDocument();  
 Element root = doc.createElement("table");  
 doc.appendChild(root);  
 int rowN = 0;  
 for (String[] row : values) {  
 rowN++;  
 Element rowElement = doc.createElement("row");  
 rowElement.setAttribute("id", String.valueOf(rowN));  
 int cellN = 0;  
 for (String cell : row) {  
 cellN++;  
 Element cellElement = doc.createElement("cell");  
 cellElement.setAttribute("id", String.valueOf(cellN));  
 cellElement.setTextContent(cell);  
 rowElement.appendChild(cellElement);  
 }  
 root.appendChild(rowElement);  
 }  
 TransformerFactory tranFactory = TransformerFactory.newInstance();  
 Transformer aTransformer = tranFactory.newTransformer();  
 aTransformer.setOutputProperty(OutputKeys.ENCODING, "UTF-8");  
 aTransformer.setOutputProperty(  
 "{http://xml.apache.org/xslt}indent-amount", "2");  
 aTransformer.setOutputProperty(OutputKeys.INDENT, "yes");  
 DOMSource domSource = new DOMSource(doc);  
 try {  
 FileWriter fileWriter = new FileWriter("./saved\_table.xml");  
 StreamResult streamResult = new StreamResult(fileWriter);  
 aTransformer.transform(domSource, streamResult);  
 } catch (IOException ex) {  
 ex.printStackTrace();  
 }  
 } catch (TransformerException | ParserConfigurationException ex) {  
 System.err.println("Can't build the document: " + ex.getLocalizedMessage());  
 }  
 }  
  
 private void calculate\_handleAction(ActionEvent e) {  
  
 int selectedRow = Integer.parseInt(rowNumberSelector.getValue().toString()) - 1,  
 selectedColumn = Integer.parseInt(colNumberSelector.getValue().toString()) - 1;  
  
 final String[][] values = ((ValuesTableModel) mainTable.getModel()).getValues();  
  
 Integer  
 summ\_byTable = 0,  
 summ\_byCol = 0,  
 summ\_byRow = 0;  
 int  
 count4average\_byTable = 0,  
 count4average\_byCol = 0,  
 count4average\_byRow = 0;  
 Integer  
 min\_byTable = Integer.MAX\_VALUE,  
 min\_byCol = Integer.MAX\_VALUE,  
 min\_byRow = Integer.MAX\_VALUE;  
 Integer  
 max\_byTable = Integer.MIN\_VALUE,  
 max\_byCol = Integer.MIN\_VALUE,  
 max\_byRow = Integer.MIN\_VALUE;  
  
 for (int row = values.length - 1; -1 < row; row--) {  
 for (int col = values[row].length - 1; -1 < col; col--) {  
 try {  
 int val = Integer.parseInt(values[row][col]);  
  
 if (row == selectedRow) {  
 summ\_byRow += val;  
 count4average\_byRow = count4average\_byRow + 1;  
 if (val < min\_byRow) {  
 min\_byRow = val;  
 }  
 if (val > max\_byRow) {  
 max\_byRow = val;  
 }  
 }  
  
 if (col == selectedColumn) {  
 summ\_byCol += val;  
 count4average\_byCol = count4average\_byCol + 1;  
 if (val < min\_byCol) {  
 min\_byCol = val;  
 }  
 if (val > max\_byCol) {  
 max\_byCol = val;  
 }  
 }  
  
 summ\_byTable += val;  
 count4average\_byTable = count4average\_byTable + 1;  
  
 if (val < min\_byTable) {  
 min\_byTable = val;  
 }  
 if (val > max\_byTable) {  
 max\_byTable = val;  
 }  
  
 } catch (NumberFormatException ignored){}  
 }  
 }  
  
 ResultTableModel resultTableModel = (ResultTableModel) resultTable.getModel();  
  
 resultTableModel.setResultSumm(  
 String.valueOf(summ\_byRow),  
 String.valueOf(summ\_byCol),  
 String.valueOf(summ\_byTable)  
 );  
 resultTableModel.setResultAverage(  
 count4average\_byRow > 0 ? String.valueOf(summ\_byRow / count4average\_byRow) : "-",  
 count4average\_byCol > 0 ? String.valueOf(summ\_byCol / count4average\_byCol) : "-",  
 count4average\_byTable > 0 ? String.valueOf(summ\_byTable / count4average\_byTable) : "-"  
 );  
 resultTableModel.setResultMinimum(  
 min\_byRow != Integer.MAX\_VALUE ? String.valueOf(min\_byRow) : "-",  
 min\_byCol != Integer.MAX\_VALUE ? String.valueOf(min\_byCol) : "-",  
 min\_byTable != Integer.MAX\_VALUE ? String.valueOf(min\_byTable) : "-"  
 );  
 resultTableModel.setResultMaximum(  
 max\_byRow != Integer.MIN\_VALUE ? String.valueOf(max\_byRow) : "-",  
 max\_byCol != Integer.MIN\_VALUE ? String.valueOf(max\_byCol) : "-",  
 max\_byTable != Integer.MIN\_VALUE ? String.valueOf(max\_byTable) : "-"  
 );  
  
 resultTableModel.fireTableDataChanged();  
 resultTable.updateUI();  
 }  
  
 private void generateTM\_handleAction(ActionEvent e) {  
 mainTable\_updateTM(initValuesTM(Integer.parseInt(rowCountSelector.getValue().toString()), Integer.parseInt(colCountSelector.getValue().toString())));  
 }  
  
 private void mainTable\_handleSelectionEvent(ListSelectionEvent e) {  
 colNumberSelector.setEnabled(false);  
 rowNumberSelector.setEnabled(false);  
  
 int selectedRow = mainTable.getSelectedRow();  
 rowNumberSelector.setValue(selectedRow >= 0 ? selectedRow + 1 : 0);  
  
 int selectedColumns = mainTable.getSelectedColumn();  
 colNumberSelector.setValue(selectedColumns >= 0 ? selectedColumns + 1 : 0);  
  
 colNumberSelector.setEnabled(true);  
 rowNumberSelector.setEnabled(true);  
 }  
  
 private void mainTable\_updateTM(TableModel tm) {  
 mainTable.setEnabled(false);  
 mainTable.setModel(tm);  
 mainTable.updateUI();  
 mainTable.setEnabled(true);  
 }  
  
}

**Результат работы прогаммы:**



**Вывод:** освоили работу с двумерными массивами; познакомились с классом StringGrid; научились программировать алгоритмы поиска наибольшего и наименьшего значения, научились обрабатывать исключительные ситуации.