Пензенский государственный университет

Факультет вычислительной техники

Кафедра «Вычислительная техника»

Работа с коллекциями объектов

Отчет о лабораторной работе № 2

по дисциплине «Разработка кроссплатформенных приложений»

Выполнили: ст-ты гр. 21ВОЭ1

Кокорин А. А.

Тишаев М. Р.

Приняли: к.т.н., доцент каф. ВТ

Юрова О. В.

к.т.н., доцент каф. ВТ

Карамышева Н. С.

2024

1 Цель работы

Изучить библиотеку стандартных коллекций Java Collections Framework, позволяющую хранить различные структуры данных.

2 Формулировка задания

Модифицировать приложение из предыдущей лабораторной работы, реализовав хранение данных таблицы с использованием библиотеки коллекций. Для этого реализовать класс RecIntegral, способный хранить одну запись таблицы. Для нечетных вариантов в качестве класса-коллекции выбрать ArrayList, для четных - LinkedList. Кроме того, добавить пару кнопок: очистить / заполнить, которые будут очищать таблицу и заполнять ее данными из коллекции соответственно. Оформление лабораторной работы должно быть выполнено в соответствии с требованиями.

3 Графическое описание алгоритма

Алгоритм метода *jButtonAdd* для добавления данных интеграла в таблицу представлен на рисунке 1. Алгоритм начинается с добавления данных из текстовых полей *jTextFieldStep*, *jTextFieldLow*, *jTextFieldTop* в таблицу (блок 2). Затем данные из текстовых полей *jTextFieldStep*, *jTextFieldLow*, *jTextFieldTop* добавляются в конец списка *List* (блок 3). Далее метод завершает свою работу.

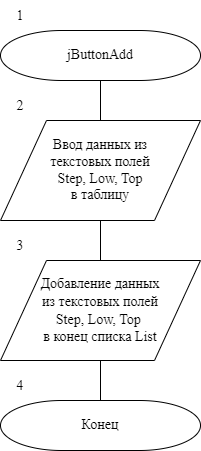


Рисунок 1 – Схема метода *jButtonAdd*

Алгоритм метода *jButtonDel* для удаления выделенной строки с данными из таблицы представлен на рисунке 2. Алгоритм начинается с ввода *row* – количество строк в таблице (блок 2). Затем проверяется условие *row = 0* (блок 3). Если условие истинно, то программа завершает свою работу. В противном случае осуществляется удаление строки с номером *row* из таблицы (блок 4). Далее происходит удаление элемента с номером *row* из списка *List* (блок 5). Затем метод завершает свою работу.

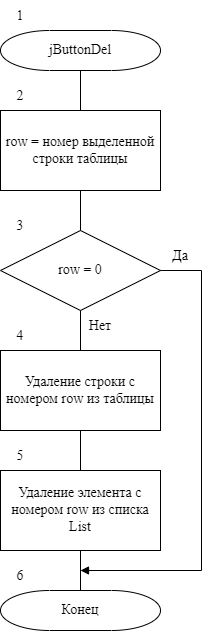


Рисунок 2 – Схема метода *jButtonDel*

Алгоритм метода *jButtonFill* для удаления выделенной строки с данными из таблицы представлен на рисунке 3. Алгоритм начинается с проверки условия, количество элементов в списке *List = 0* (блок 2). Если условие истинно, то программа завершает свою работу. В противном случае все данные из списка *List* добавляются в таблицу (блок 3). Затем метод завершает свою работу.



Рисунок 3 – Схема метода *jButtonFill*

4 Описание программы

Программа построена на основе лабораторной работы № 1.

Текст программы приведен в приложении А.

На Форму были добавлены следующие объекты:

кнопки *jButtonFill* для создания обработчика события с целью добавления данных из списка *List* в таблицу и *jButtonClear* для создания обработчика события с целью очистки таблицы. В свойстве *text* этих объектов помещены пояснения для пользователя программы:

*jButtonFill.text* — «Заполнить»;

*jButtonClear.text* — «Очистить».

В программу были добавлены два обработчика событий: обработчик события клика на кнопку формы *jButtonAdd* — *jButtonAddActionPerformed*, обработчик события клика на кнопку формы *jButtonDel* — *jButtonDelActionPerformed*, обработчик событития. А также был добавлен двусвязный список *List* с элементами класса *RecIntegral* для записей таблицы.

Был добавлен класс *RecIntegral* для хранения одной записи таблицы.

В классе *RecIntegral* были описаны:

закрытые поля вещественного типа *double*:

*a* – нижняя граница интегрирования,

*b* – верхняя граница интегрирования,

*h* – шаг интегрирования;

конструктор *RecIntegral(a, b, h)*, который переносит значения из параметров в соответствующие переменные класса;

методы, которые возвращают значения вещественного типа:

*geta()* – возвращает значение переменной *a*,

*getb()* – возвращает значение переменной *b*,

*geth()* – возвращает значение переменной *h*.

5 Программа и методика испытаний

Для проверки правильности работы программы подготовлен вручную тестовый набор данных в таблице 1.

Таблица 1 – Тестовый набор данных

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Данные, введенные пользователем | | | Данные, полученные с помощью кнопки «Заполнить» | | |
| Нижняя граница | Верхняя граница | Шаг | Нижняя граница | Верхняя граница | Шаг |
| 2 | 4 | 0.2 | 2 | 4 | 0.2 |
| 0 | 3 | 0.3 | 0 | 3 | 0.3 |
| 100 | 100000 | 0.01 | 100 | 100000 | 0.01 |
| 24 | 200 | 0.02 | 24 | 200 | 0.02 |

Для проверки правильности программы необходимо проверить совпадают ли данные таблицы, полученные путем нажатия на кнопку «Заполнить», с ранее введенными данными пользователем.

При работе программы с тестовыми данными получены результаты, приведенные в Приложении Б. Результат, приведенный в таблице 1, совпадают с результатом работы программы на рисунке Б.2, Б.3, Б.4, Б.5. Таким образом, можно сделать вывод, что программа работает правильно.

Вывод

Результатом выполнения данной лабораторной работы является изучение библиотеки стандартных коллекций Java Collections Framework, которая позволяет хранить различные структуры данных, и добавление в программу кнопок очистить / заполнить, которые очищают таблицу и заполняют ее данными из коллекции соответственно.

ТЕКСТ ПРОГРАММЫ

Приложение А

(обязательное)

Класс NewJFrame

package my.javaapplication1;

import javax.swing.table.DefaultTableModel;

import java.util.LinkedList;

public class NewJFrame extends javax.swing.JFrame {

LinkedList<RecIntegral> List = new LinkedList<RecIntegral>();

public NewJFrame() {

initComponents();

}

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

jButtonAdd = new javax.swing.JButton();

jButtonDel = new javax.swing.JButton();

jButtonFind = new javax.swing.JButton();

jTextFieldLow = new javax.swing.JTextField();

jTextFieldTop = new javax.swing.JTextField();

jTextFieldStep = new javax.swing.JTextField();

jScrollPane1 = new javax.swing.JScrollPane();

jTable1 = new javax.swing.JTable();

jLabel1 = new javax.swing.JLabel();

jLabel2 = new javax.swing.JLabel();

jLabel3 = new javax.swing.JLabel();

jButtonClear = new javax.swing.JButton();

jButtonFill = new javax.swing.JButton();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

jButtonAdd.setText("Добавить");

jButtonAdd.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButtonAddActionPerformed(evt);

}

});

jButtonDel.setText("Удалить");

jButtonDel.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButtonDelActionPerformed(evt);

}

});

jButtonFind.setText("Вычислить");

jButtonFind.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButtonFindActionPerformed(evt);

}

});

jTable1.setModel(new javax.swing.table.DefaultTableModel(

new Object [][] {

},

new String [] {

"Нижняя граница", "Верхняя граница", "Шаг", "Результат"

}

) {

boolean[] canEdit = new boolean [] {

true, true, true, false

};

public boolean isCellEditable(int rowIndex, int columnIndex) {

return canEdit [columnIndex];

}

});

jTable1.getTableHeader().setReorderingAllowed(false);

jScrollPane1.setViewportView(jTable1);

if (jTable1.getColumnModel().getColumnCount() > 0) {

jTable1.getColumnModel().getColumn(0).setResizable(false);

jTable1.getColumnModel().getColumn(1).setResizable(false);

jTable1.getColumnModel().getColumn(2).setResizable(false);

jTable1.getColumnModel().getColumn(3).setResizable(false);

}

jLabel1.setText("Нижняя граница");

jLabel2.setText("Верхняя граница");

jLabel3.setText("Шаг");

jButtonClear.setText("Очистить");

jButtonClear.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButtonClearActionPerformed(evt);

}

});

jButtonFill.setText("Заполнить");

jButtonFill.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButtonFillActionPerformed(evt);

}

});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(jButtonFind, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jButtonAdd, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jButtonDel, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

.addGap(30, 30, 30)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(jLabel2, javax.swing.GroupLayout.DEFAULT\_SIZE, 94, Short.MAX\_VALUE)

.addGroup(layout.createSequentialGroup()

.addComponent(jLabel3)

.addGap(0, 0, Short.MAX\_VALUE))

.addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jTextFieldStep)

.addComponent(jTextFieldTop)

.addComponent(jTextFieldLow))

.addGap(64, 64, 64)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(jButtonFill, javax.swing.GroupLayout.DEFAULT\_SIZE, 100, Short.MAX\_VALUE)

.addComponent(jButtonClear, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

.addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

.addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT\_SIZE, 551, Short.MAX\_VALUE)

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addComponent(jLabel1)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jButtonAdd)

.addComponent(jTextFieldLow, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jButtonFill))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addComponent(jLabel2)

.addGap(6, 6, 6)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jButtonDel)

.addComponent(jTextFieldTop, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jButtonClear))

.addGap(17, 17, 17)

.addComponent(jLabel3)

.addGap(3, 3, 3)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jButtonFind)

.addComponent(jTextFieldStep, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 176, javax.swing.GroupLayout.PREFERRED\_SIZE))

);

pack();

}// </editor-fold>

private void jButtonAddActionPerformed(java.awt.event.ActionEvent evt) {

DefaultTableModel TModel = (DefaultTableModel) jTable1.getModel();

TModel.addRow(new Object[] {jTextFieldLow.getText(), jTextFieldTop.getText(), jTextFieldStep.getText()});

List.addLast(new RecIntegral(Double.valueOf(jTextFieldLow.getText()), Double.valueOf(jTextFieldTop.getText()), Double.valueOf(jTextFieldStep.getText())));

}

private void jButtonDelActionPerformed(java.awt.event.ActionEvent evt) {

int row = jTable1.getSelectedRow();

if (row != -1) {

DefaultTableModel TModel = (DefaultTableModel) jTable1.getModel();

TModel.removeRow(row);

List.remove(row - 1);

}

}

private void jButtonFindActionPerformed(java.awt.event.ActionEvent evt) {

int row = jTable1.getSelectedRow();

if (row != -1) {

double h, a, b, res = 0;

a = Double.parseDouble(jTable1.getValueAt(row, 0).toString());

b = Double.parseDouble(jTable1.getValueAt(row, 1).toString());

h = Double.parseDouble(jTable1.getValueAt(row, 2).toString());

if ((a < b) && ((b - a) % h < 0.0001 || (b - a) % h > h - 0.0001)) {

for (double i = a; i < b; i += h)

res += ((Math.sin(i \* Math.PI / 180) + Math.sin((i + h) \* Math.PI / 180)) \* (h / 2));

jTable1.setValueAt(String.format("%.4f",res), row, 3);

}

}

}

private void jButtonFillActionPerformed(java.awt.event.ActionEvent evt) {

if (List.size() != -1) {

DefaultTableModel TModel = (DefaultTableModel) jTable1.getModel();

for (RecIntegral Rec: List)

TModel.addRow(new Object[] {Rec.geta(), Rec.getb(), Rec.geth()});

}

}

private void jButtonClearActionPerformed(java.awt.event.ActionEvent evt) {

DefaultTableModel TModel = (DefaultTableModel) jTable1.getModel();

TModel.setRowCount(0);

}

public static void main(String args[]) {

try {

for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break;

}

}

} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(NewJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(NewJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(NewJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(NewJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new NewJFrame().setVisible(true);

}

});

}

// Variables declaration - do not modify

private javax.swing.JButton jButtonAdd;

private javax.swing.JButton jButtonClear;

private javax.swing.JButton jButtonDel;

private javax.swing.JButton jButtonFill;

private javax.swing.JButton jButtonFind;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JTable jTable1;

private javax.swing.JTextField jTextFieldLow;

private javax.swing.JTextField jTextFieldStep;

private javax.swing.JTextField jTextFieldTop;

// End of variables declaration

}

Класс RecIntegral

package my.javaapplication1;

public class RecIntegral {

private double a, b, h;

public RecIntegral(double a, double b, double h) {

this.a = a;

this.b = b;

this.h = h;

}

public double geta(){

return this.a;

}

public double getb(){

return this.b;

}

public double geth(){

return this.h;

}

}

РЕЗУЛЬТАТЫ ИСПЫТАНИЙ

Приложение Б

(обязательное)

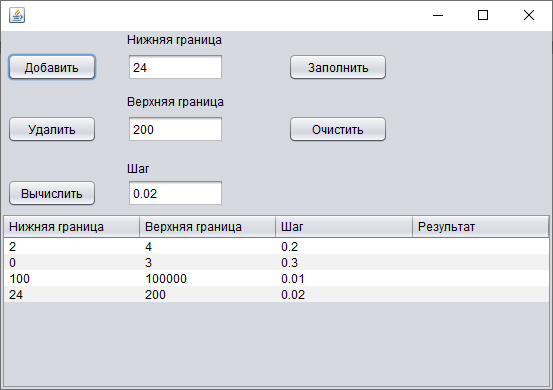


Рисунок Б.1 – Данные, введенные пользователем

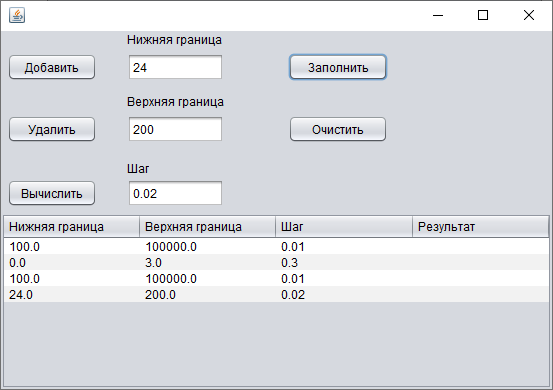


Рисунок Б.2 – Данные, полученные с помощью кнопки «Заполнить»