Пензенский государственный университет

Факультет вычислительной техники

Кафедра «Вычислительная техника»

Отчет о лабораторной работе №3

по дисциплине «Разработка кроссплатформенных приложений»

на тему «Обработка исключительных ситуаций»

Выполнили: ст-ты гр. 22ВОЭ1

Старёхин Б.А.

Николаев Д.А.

Проверили: к.т.н., доцент каф. ВТ

Юрова О.В.

Карамышева Н.С.

2025

Цель работы

Изучить механизм обработки исключительных ситуаций.

Задание

Модифицировать приложение из предыдущей лабораторной работы, реализовав проверку вводимых данных с использованием механизма исключений. Необходимо создать свой класс, унаследованный от класса Exception, и генерировать исключение, если возникает попытка создать экземпляр класса RecIntegral со значениями, не являющимися числами в диапазоне от 0,000001 до 1000000. В качестве обработки исключения необходимо выводить диалог, содержащий предупреждение о некорректности введенных данных.

Ход работы

Листинг программы

package my.contacteditor;

import javax.swing.table.DefaultTableModel;

import javax.swing.JOptionPane;

import java.util.ArrayList;

import java.util.Scanner;

public class ContactEditorUI extends javax.swing.JFrame {

ArrayList<RecIntegral> arrInteg = new ArrayList<> ();

public ContactEditorUI() {

initComponents();

}

@SuppressWarnings("unchecked")

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

private void initComponents() {

jPanel1 = new javax.swing.JPanel();

jLabel1 = new javax.swing.JLabel();

jTextField1 = new javax.swing.JTextField();

jLabel2 = new javax.swing.JLabel();

jTextField2 = new javax.swing.JTextField();

jLabel3 = new javax.swing.JLabel();

jTextField3 = new javax.swing.JTextField();

jButton1 = new javax.swing.JButton();

jButton2 = new javax.swing.JButton();

jButton3 = new javax.swing.JButton();

jScrollPane1 = new javax.swing.JScrollPane();

jTable1 = new javax.swing.JTable();

jButton4 = new javax.swing.JButton();

jButton5 = new javax.swing.JButton();

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

jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder(null, "", javax.swing.border.TitledBorder.DEFAULT\_JUSTIFICATION, javax.swing.border.TitledBorder.DEFAULT\_POSITION, new java.awt.Font("Tahoma", 1, 12))); // NOI18N

jLabel1.setText("верхний предел");

jTextField1.addActionListener(new java.awt.event.ActionListener() {

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

jTextField1ActionPerformed(evt);

}

});

jLabel2.setText("нижний предел");

jLabel3.setText("шаг");

jButton1.setText("добавить");

jButton1.addActionListener(new java.awt.event.ActionListener() {

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

jButton1ActionPerformed(evt);

}

});

jButton2.setText("вычислить");

jButton2.addActionListener(new java.awt.event.ActionListener() {

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

jButton2ActionPerformed(evt);

}

});

jButton3.setText("удалить");

jButton3.addActionListener(new java.awt.event.ActionListener() {

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

jButton3ActionPerformed(evt);

}

});

jTable1.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N

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

new Object [][] {

},

new String [] {

"верх.пред", "ниж.пред", "шаг", "результат"

}

) {

boolean[] canEdit = new boolean [] {

false, false, false, false

};

public boolean isCellEditable(int rowIndex, int columnIndex) {

return canEdit [columnIndex];

}

});

jScrollPane1.setViewportView(jTable1);

jButton4.setText("заполнить");

jButton4.addActionListener(new java.awt.event.ActionListener() {

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

jButton4ActionPerformed(evt);

}

});

jButton5.setText("очистить");

jButton5.addActionListener(new java.awt.event.ActionListener() {

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

jButton5ActionPerformed(evt);

}

});

javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);

jPanel1.setLayout(jPanel1Layout);

jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addContainerGap()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT\_SIZE, 466, Short.MAX\_VALUE)

.addContainerGap())

.addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

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

.addComponent(jTextField1)

.addComponent(jLabel2)

.addComponent(jTextField2)

.addComponent(jLabel3)

.addComponent(jTextField3))

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

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jButton4, javax.swing.GroupLayout.Alignment.TRAILING)

.addComponent(jButton5, javax.swing.GroupLayout.Alignment.TRAILING))

.addGap(30, 30, 30)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(jButton2, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jButton3, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jButton1, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

.addGap(92, 92, 92))))

);

jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addContainerGap()

.addComponent(jLabel1)

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

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jButton1)

.addComponent(jButton4))

.addGap(18, 18, 18)

.addComponent(jLabel2)

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

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jButton2)

.addComponent(jButton5))

.addGap(18, 18, 18)

.addComponent(jLabel3)

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

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jTextField3, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jButton3))

.addGap(59, 59, 59)

.addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT\_SIZE, 126, Short.MAX\_VALUE)

.addContainerGap())

);

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

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

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

.addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

);

layout.setVerticalGroup(

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

.addGroup(layout.createSequentialGroup()

.addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(0, 6, Short.MAX\_VALUE))

);

pack();

}// </editor-fold>

private void jTextField1ActionPerformed(java.awt.event.ActionEvent evt) {

}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

double vpred;

double npred;

double step;

try {

vpred = Double.parseDouble(jTextField1.getText());

npred = Double.parseDouble(jTextField2.getText());

step = Double.parseDouble(jTextField3.getText());

if (vpred < 0.000001 || vpred > 1000000 ||

npred < 0.000001 || npred > 1000000 ||

step < 0.000001 || step > 1000000) {

throw new CustomException("Значения должны находиться в диапазоне от 0.000001 до 1000000.");

}

if (vpred < npred){

throw new CustomException("Нижний предел должен быть меньше верхнего предела.");

}

if ((vpred - npred) < step){

throw new CustomException("Шаг интегрирования должен быть меньше интервала.");

}

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

tModel.addRow(new Object[] {vpred, npred, step});

arrInteg.add(new RecIntegral(vpred, npred, step));

jTextField1.setText("");

jTextField2.setText("");

jTextField3.setText("");

} catch (CustomException ex) {

JOptionPane.showMessageDialog(this, ex.getMessage(),

"Ошибка диапазона", JOptionPane.WARNING\_MESSAGE);

} catch (NumberFormatException ex) {

JOptionPane.showMessageDialog(this, "Некорректное значение введено. Попробуйте снова.",

"Ошибка ввода", JOptionPane.ERROR\_MESSAGE);

}

}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

double vp;

double np;

double st;

double result;

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

int rowNom = jTable1.getSelectedRow();

if (rowNom == -1) {

JOptionPane.showMessageDialog(null, "выберите строку!");

}

else {

vp = Double.parseDouble(tModel.getValueAt(rowNom, 0).toString());

np = Double.parseDouble(tModel.getValueAt(rowNom, 1).toString());

st = Double.parseDouble(tModel.getValueAt(rowNom, 2).toString());

result = CalcInt (vp,np,st);

tModel.setValueAt(result, rowNom, 3);

arrInteg.set (rowNom, new RecIntegral (np, vp, st, result));

}

}

public double CalcInt (double vp, double npr, double step){

double start, h, sumS=0;

start = vp;

while (start > npr) {

h = Math.min(step, start - npr);

sumS += h \* (Math.exp(-start) + Math.exp(-(start - h))) / 2;

start -= h;

}

return sumS;

}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {

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

int rowNom = jTable1.getSelectedRow();

if (rowNom == -1) {

JOptionPane.showMessageDialog(null, "выберите строку!");

}

else {

tModel.removeRow(rowNom);

}

}

private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {

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

tModel.setNumRows(0);

}

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

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

tModel.setNumRows(0);

for (RecIntegral item: arrInteg) {

tModel.addRow(new Object[] {item.getVP(),item.getNP(),item.getST(),item.getRES()});

}

}

public static void main(String args[]) {

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

public void run() {

new ContactEditorUI().setVisible(true);

}

});

}

// Variables declaration - do not modify

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;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JPanel jPanel1;

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JTable jTable1;

private javax.swing.JTextField jTextField1;

private javax.swing.JTextField jTextField2;

private javax.swing.JTextField jTextField3;

// End of variables declaration

}

package my.contacteditor;

public class RecIntegral {

private double vp;

private double np;

private double st;

private double result;

public RecIntegral (double np, double vp, double st){

this.vp = vp;

this.np = np;

this.st = st;

this.result = 0.0;

}

public RecIntegral (double np, double vp, double st, double result){

this.vp = vp;

this.np = np;

this.st = st;

this.result = result;

}

public RecIntegral (double np, double vp, double st, String someAdditionalParam) throws CustomException {

if (np < 0.000001 || np > 1000000) {

throw new CustomException ("Значение должно быть в диапазоне от 0.000001 до 1000000.");

}

if (vp < 0.000001 || vp > 1000000) {

throw new CustomException ("Значение должно быть в диапазоне от 0.000001 до 1000000.");

}

if (st < 0.000001 || st > 1000000) {

throw new CustomException ("Значение должно быть в диапазоне от 0.000001 до 1000000.");

}

if (np > vp) {

throw new CustomException ("Нижний предел должен быть меньше верхнего предела.");

}

if ((vp - np) < st) {

throw new CustomException ("Шаг интегрирования должен быть меньше интервала.");

}

this.vp = vp;

this.np = np;

this.st = st;

}

public double getNP(){

return this.np;

}

public double getVP(){

return this.vp;

}

public double getST(){

return this.st;

}

public double getRES(){

return result;

}

public void setRES(double result){

this.result = result;

}

}

package my.contacteditor;

public class CustomException extends Exception {

public CustomException(String message) {

super (message);

}

}

Словесное описание алгоритма

1. Объявление метода, который обрабатывает событие нажатия кнопки jButton1.
2. Объявление трех переменных типа double: vpred, npred и step, которые будут использоваться для хранения значений.
3. Получение текста из текстового поля jTextField1, преобразование его в число типа double и присвоение переменной vpred.
4. Получение текста из jTextField2, преобразование в double и присвоение переменной npred.
5. Получение текста из jTextField3, преобразование в double и присвоение переменной step.
6. Проверяем, находятся ли значения vpred, npred и step в заданном диапазоне.
7. Проверяем, npred должен быть меньше чем vpred.
8. Проверяем, шаг интегрирования должен быть меньше интервла.
9. Если хотя бы одно из значений не корректно, то выбрасываем исключение.
10. Получение модели таблицы из jTable1 и приведение ее к типу DefaultTableModel, чтобы можно было добавлять строки.
11. Добавление новой строки в модель таблицы с тремя значениями: vpred, npred и step.
12. Добавление нового объекта RecIntegral в коллекцию arrInteg.
13. Очистка текстовых полей после добавления данных в таблицу.
14. Блок catch обрабатывает исключение типа CustomException, которое может быть выброшено в предыдущем блоке try, переменная ex будет содержать информацию об этом исключении.
15. Здесь вызывается метод showMessageDialog класса JOptionPane, который отображает диалоговое окно с сообщением, this указывает на текущий объект, ex.getMessage() возвращает сообщение об ошибке, которое было передано при создании исключения CustomException, JOptionPane.WARNING\_MESSAGE указывает, что это предупреждающее сообщение, что визуально выделяет его в диалоговом окне.
16. Блок catch обрабатывает исключение типа NumberFormatException, которое возникает, если введенные данные не могут быть преобразованы в число.
17. Вызывается метод showMessageDialog, чтобы отобразить сообщение об ошибке, сообщение информирует пользователя о том, что введенное значение некорректно и предлагает попробовать снова, JOptionPane.ERROR\_MESSAGE указывает, что это сообщение об ошибке, что также визуально выделяет его в диалоговом окне.
18. Объявление метода, который обрабатывает событие нажатия кнопки jButton2.
19. Объявление четырех переменных типа double: vp, np, st и result, которые будут использоваться для хранения значений из выбранной строки таблицы и результата вычислений.
20. Получение модели таблицы из jTable1 и приведение ее к типу DefaultTableModel.
21. Получение номера выбранной строки в таблице. Если строка не выбрана, rowNom будет равен -1.
22. Проверка, выбрана ли строка. Если нет, отображается сообщение с просьбой выбрать строку.
23. Если строка выбрана, извлечение значений из выбранной строки таблицы и преобразование их в double. Значения присваиваются переменным vp, np и st.
24. Вызов метода CalcInt, который выполняет вычисления с использованием значений vp, np и st, и присвоение результата переменной result.
25. Установка значения result в четвертую колонку (индекс 3) выбранной строки таблицы.
26. Происходит вызов метода set у объекта arrInteg. Новое значение, которое устанавливается по индексу, является объект класса RecIntegral, которому передаются четыре параметра: np, vp, st и result.
27. Объявление метода CalcInt, который принимает три параметра: vp, npr и step. Метод возвращает значение типа double.
28. Объявление переменных:

start — начальное значение, инициализируется значением vp.

h — переменная для хранения текущего шага.

sumS — переменная для накопления суммы, инициализируется нулем.

1. Инициализация переменной start значением vp.
2. Начало цикла do-while, который будет выполняться хотя бы один раз.
3. Вычисление значения h, которое будет равно меньшему из step и разности npr и start. Это гарантирует, что шаг не превысит оставшееся расстояние до npr.
4. Обновление sumS с использованием метода трапеций для численного интегрирования. Здесь вычисляется площадь трапеции, основанной на текущем start и start + h.
5. Увеличение start на значение h, чтобы перейти к следующему интервалу.
6. Условие продолжения цикла: пока start меньше npr, цикл будет выполняться.
7. Возврат накопленной суммы sumS, которая представляет собой результат интегрирования.
8. Объявление метода, который обрабатывает событие нажатия кнопки jButton3.
9. Получение модели таблицы jTable1 и приведение её к типу DefaultTableModel, чтобы можно было изменять данные в таблице.
10. Получение номера выбранной строки в таблице. Если строка не выбрана, rowNom будет равен -1.
11. Проверка, выбрана ли строка.
12. Показ сообщения пользователю с просьбой выбрать строку.
13. Удаление выбранной строки из модели таблицы.
14. Объявление метода, который обрабатывает событие нажатия кнопки jButton5.
15. Получение модели таблицы jTable1 и приведение её к типу DefaultTableModel, чтобы можно было изменять данные в таблице.
16. Устанавливает количество строк в таблице в 0, что фактически очищает все текущие данные, отображаемые в jTable1.
17. Объявление метода, который обрабатывает событие нажатия кнопки jButton4.
18. Получение модели таблицы jTable1 и приведение её к типу DefaultTableModel, чтобы можно было изменять данные в таблице.
19. Устанавливает количество строк в таблице в 0, что фактически очищает все текущие данные, отображаемые в jTable1.
20. Цикл for, который проходит по всем элементам arrInteg, каждый элемент представляет собой объект RecIntegral.
21. Внутри цикла добавляем новую строку в таблицу для каждого элемента, для каждого item извлекаем значения, вызывая методы getNP(), getVP(), getST(), и getRES(). Это передается методу addRow, который добавляет данные в таблицу jTable1.
22. Объявляет пакет, к которому принадлежит класс.
23. Объявляет новый класс RecIntegral.
24. Объявляются четыре переменные, они будут использоваться для хранения значений в каждом объекте класса RecIntegral.
25. Объявляет конструктор класса. Он принимает три параметра: np, vp и st.
26. this используется для разграничения между полями класса и параметрами конструктора, значения параметров присваиваются соответствующим переменным экземпляра. Переменной result присваивается значение 0.0.
27. Второй конструктор класса, который принимает четыре параметра, позволяет создавать объект с уже инициализированным значением result.
28. result получает значение, переданное в конструктор.
29. Метод RecIntegral является конструктором класса RecIntegral. Он принимает четыре параметра: np, vp, st и someAdditionalParam. Конструктор может выбрасывать исключение CustomException.
30. Если значение np меньше 0.000001 или больше 1000000, выбрасывается исключение CustomException с сообщением о том, что значение должно находиться в указанном диапазоне.
31. Если значение vp меньше 0.000001 или больше 1000000, выбрасывается исключение CustomException с сообщением о том, что значение должно находиться в указанном диапазоне.
32. Если значение st меньше 0.000001 или больше 1000000, выбрасывается исключение CustomException с сообщением о том, что значение должно находиться в указанном диапазоне.
33. Если значение np меньше чем vp, то выбрасывается исключение CustomException с сообщением о том, что нижний предел должен быть меньше верхнего предела.
34. Если значение vp - np меньше чем st, то выбрасывается исключение CustomException с сообщением о том, шаг интегрирования должен быть меньше чем интервал.
35. Если все проверки пройдены, значения vp, np и st присваиваются соответствующим полям класса, что позволяет сохранить их для дальнейшего использования в экземпляре класса.
36. геттер, который возвращает значение переменной vp.
37. геттер, который возвращает значение переменной np.
38. геттер, который возвращает значение переменной st.
39. геттер, который возвращает значение переменной res.
40. Объявляем пакет, к которому принадлежит класс.
41. Объявляем новый класс CustomException, который является подклассом стандартного класса Exception. Это означает, что CustomException будет представлять собой пользовательское исключение, которое можно выбрасывать и обрабатывать в программе.
42. Конструктор класса CustomException. Он принимает строку message, которая будет содержать сообщение об ошибке, связанной с исключением.
43. Вызываем конструктор суперкласса Exception с помощью ключевого слова super. Это передает сообщение об ошибке в класс Exception, чтобы оно могло быть использовано при обработке исключения.

Результат выполнения программы

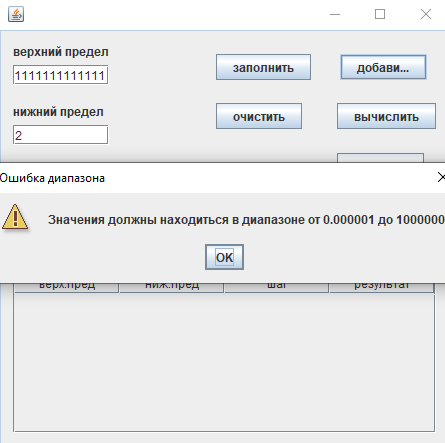


Рисунок 1 — Ввод некорректных данных

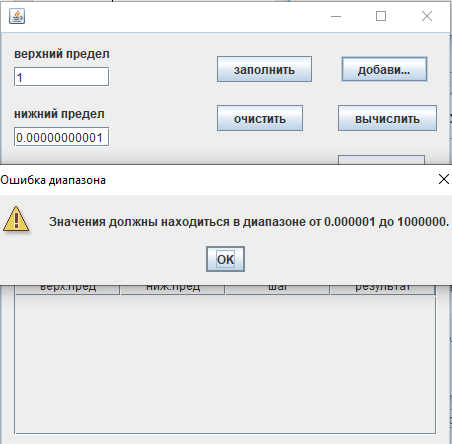


Рисунок 2 — Ввод некорректных данных

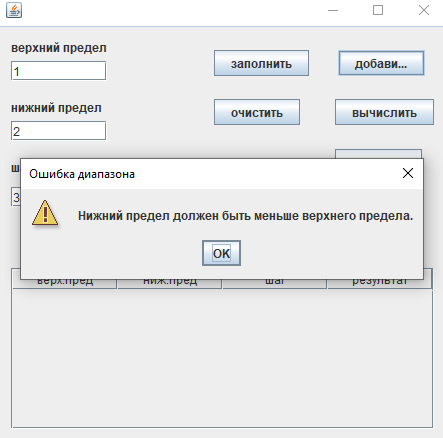


Рисунок 3 — Ввод некорректных данных

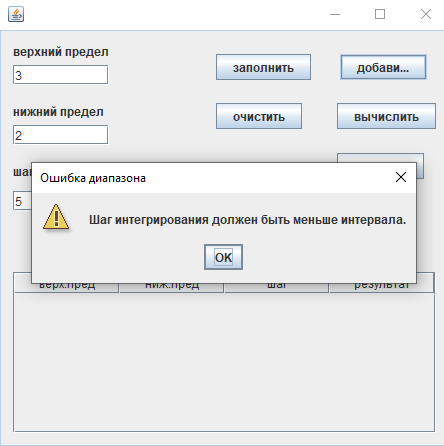


Рисунок 4 — Ввод некорректных данных

Вывод

В данной лабораторной работе был изучен механизм обработки исключительных ситуаций.