Министерство образования Российской Федерации

Пензенский государственный университет

**Кафедра «Вычислительная техника»**

**ОТЧЁТ**

По лабораторной работе №3

По дисциплине:

«Программирование на языке JAVA»

На тему:

«Обработка исключительных ситуаций»

Выполнили:

студенты группы 20ВВП1

Верховский М. В.

Лукин В. Д.

Приняли:

к.т.н., доцент, Юрова О.В.

к.т.н., доцент, Карамышева Н.С.

**Пенза 2023**

**Цель работы:** изучить механизм обработки исключительных ситуаций.

**Задание на лабораторную работу:**

**Вариант 4**

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 выполнения лабораторной работы:**

**Листинг**

Table.java

import javax.swing.\*;  
import javax.swing.table.DefaultTableModel;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.util.\*;  
  
import static javax.swing.JOptionPane.ERROR\_MESSAGE;  
  
  
public class Table {  
 private LinkedList<RecIntegral> info = new LinkedList<>();  
 private JButton AddButton;  
 private JButton DelButton;  
 private JButton CalcButton;  
 private JTextField Down;  
 private JTextField Up;  
 private JTextField Step;  
 private JTable MyTable;  
 private DefaultTableModel MyModel;  
 private JPanel MyPanel;  
 private JButton ClearButton;  
 private JButton StuffButton;  
  
 public Table(){  
  
  
 //Убираем возможность редактирования столбца с результатом  
 MyTable.setModel(new DefaultTableModel() {  
  
 @Override  
 public boolean isCellEditable(int rowIndex, int columnIndex) { // Возвращает true, если ячейка в rowIndex и columnIndex доступна для редактирования  
 return columnIndex !=3;  
 }  
 });  
  
 // Чтобы добавить и удалить строки из таблицы, нужно использовать DefaultTableModel  
 MyModel = (DefaultTableModel) MyTable.getModel();  
  
 // создание столбцов  
 MyModel.addColumn("Низ");  
 MyModel.addColumn("Верх");  
 MyModel.addColumn("Шаг");  
 MyModel.addColumn("Результат");  
  
 AddButton.addActionListener(new ActionListener() { // ActionListener хранит в себе метод, который активируется при нажатии кнопки  
 @Override  
 public void actionPerformed(ActionEvent e) { // actionPerformed - метод, который активируется при нажатии кнопки  
 //Обработка исключений с помощью блоков try-catch:  
 try {  
 if (Double.parseDouble(Step.getText()) == 0){  
 throw new Exception1("Шаг");  
 }  
  
 } catch (Exception1 ex) {  
 ex.Message1(MyPanel, ERROR\_MESSAGE);  
 return; }  
  
  
 try {  
 if (Double.parseDouble(Down.getText()) > 1000000 || Double.parseDouble(Down.getText()) < 0.000001){  
 throw new Exception1("Нижний предел");  
 }  
 if (Double.parseDouble(Up.getText()) > 1000000 || Double.parseDouble(Up.getText()) < 0.000001){  
 throw new Exception1("Верхний предел");  
 }  
 if (Double.parseDouble(Step.getText()) > 1000000 || Double.parseDouble(Step.getText()) < 0.000001){  
 throw new Exception1("Шаг");  
 }  
  
 } catch (Exception1 ex) {  
 ex.Message2(MyPanel, ERROR\_MESSAGE);  
 return; }  
  
  
 try {  
 if (Double.parseDouble(Down.getText()) > Double.parseDouble(Up.getText())){  
 throw new Exception1("Пределы");  
 }  
  
 } catch (Exception1 ex) {  
 ex.Message3(MyPanel, ERROR\_MESSAGE);  
 return; }  
  
  
 Double Up = Double.parseDouble(Table.this.Up.getText()); // getText возвращает текст, содержащийся в текстовом поле Up  
 Double Down = Double.parseDouble(Table.this.Down.getText());  
 Double Step = Double.parseDouble(Table.this.Step.getText());  
 MyModel.addRow(new Double[]{Down, Up, Step}); // addRow добавляет строку  
 info.add(new RecIntegral(Down, Up, Step, 0));//Работа с коллекцией  
  
 }  
 });  
  
 CalcButton.addActionListener(new ActionListener() { // ActionListener хранит в себе метод, который активируется при нажатии кнопки  
 @Override  
 public void actionPerformed(ActionEvent e) { // actionPerformed - метод, который активируется при нажатии кнопки  
 int indexRow = MyTable.getSelectedRow(); // Метод getSelectedRow() возвращает индекс первой выбранной строки  
  
 if (indexRow == -1) {  
 return;  
 }  
  
 double area = 0; // площадь = интеграл  
 Vector data = MyModel.getDataVector().get(indexRow);// в data записываем значения выбранной строки (значения могут меняться)  
 double a = (Double) data.get(0);  
 double b = (Double) data.get(1);  
 double h = (Double) data.get(2);  
  
 for (int i = 0; i < (b - a) / h; i++) {  
 if (a+(i+1)\*h<=b){  
 area += h \* (0.5 \* (Math.tan(a+i\*h) + Math.tan(a+(i+1)\*h)));  
 }  
 else {  
 area += h \* (0.5 \* (Math.tan(a+i\*h) + Math.tan(b)));  
 }  
  
 }  
  
 MyModel.setValueAt(area, indexRow, 3); // установка значения интеграла в строку с индексом в столбце №3  
 info.get(indexRow).Res = area;//Работа с коллекцией  
  
 }  
 });  
  
 DelButton.addActionListener(new ActionListener() { // ActionListener хранит в себе метод, который активируется при нажатии кнопки  
 @Override  
 public void actionPerformed(ActionEvent e) { // actionPerformed - метод, который активируется при нажатии кнопки  
 int indexRow = MyTable.getSelectedRow(); // Метод getSelectedRow() возвращает индекс первой выбранной строки  
 if (indexRow != -1){  
 MyModel.removeRow(indexRow); // Метод removeRow() имеет параметр ind - индекс строки, которая будет удалена  
 info.remove(indexRow);//Работа с коллекцией  
 }  
 }  
 });  
  
  
 //Очистить таблицу  
 ClearButton.addActionListener(new ActionListener() { // ActionListener хранит в себе метод, который активируется при нажатии кнопки  
 @Override  
 public void actionPerformed(ActionEvent e) { // actionPerformed - метод, который активируется при нажатии кнопки  
 //Каждый раз, когда удаляем строку, количество строк будет меняться. Продолжаем цикл до тех пор, пока не останется строк  
 while (MyModel.getRowCount()>0)  
 {  
 MyModel.removeRow(0);  
 }  
 }  
 });  
  
 //Заполнить таблицу данными из коллекции  
 StuffButton.addActionListener(new ActionListener() { // ActionListener хранит в себе метод, который активируется при нажатии кнопки  
 @Override  
 public void actionPerformed(ActionEvent e) { // actionPerformed - метод, который активируется при нажатии кнопки  
 while (MyModel.getRowCount() > 0) {  
 MyModel.removeRow(0);  
 }  
 //Заполнение таблицы данными из коллекции  
 for (RecIntegral data : info) {  
 MyModel.addRow(new Object[]{data.Down1,data.Up1,data.Step1,data.Res});  
 }  
  
 }  
 });  
  
 }  
  
 public static void main(String[] args) {  
 JFrame frame = new JFrame("Интегральная гипербола кибернетики"); // создаём окно с рамкой  
 frame.setVisible(true); // показывает или скрывает это окно в зависимости от значения параметра  
 frame.setContentPane(new Table().MyPanel);//задаёт содержимое окна с рамкой  
 frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);//Устанавливает операцию при закрытии окна закончить работу приложения,  
 // которая будет выполняться по умолчанию, когда пользователь инициирует «закрытие» этого окна  
 frame.setSize(450,600); // размер окна  
 frame.setLocation(120,100); // расположение окна  
  
  
 }  
  
}

RecIntegral.java

public class RecIntegral {  
  
 public double Up1;  
 public double Down1;  
 public double Step1;  
 public double Res;  
  
 public RecIntegral(double Down1, double Up1, double Step1, double Res){  
 this.Down1 = Down1;  
 this.Up1 = Up1;  
 this.Step1 = Step1;  
 this.Res = Res;  
  
 }  
  
}

Exception1.java

import java.awt.\*;  
import static javax.swing.JOptionPane.showMessageDialog;  
  
//класс Exception1, унаследованный от класса Exception:  
public class Exception1 extends Exception{  
  
 private String title;  
  
 public Exception1(String title) {  
 this.title = title;  
 }  
  
 //Диалоговое окно с заголовком, cообщением и типом сообщения  
 public void Message1(Component parent, int optionType) {  
 showMessageDialog(parent, "Введён нулевой шаг", title, optionType);  
 }  
 public void Message2(Component parent, int optionType) {  
 showMessageDialog(parent, "Выход за пределы диапазона (0.000001; 1000000)", title, optionType);  
 }  
  
 public void Message3(Component parent, int optionType) {  
 showMessageDialog(parent, "Верхний предел не может быть меньше нижнего предела", title, optionType);  
 }  
}

**Результаты выполнения программы**

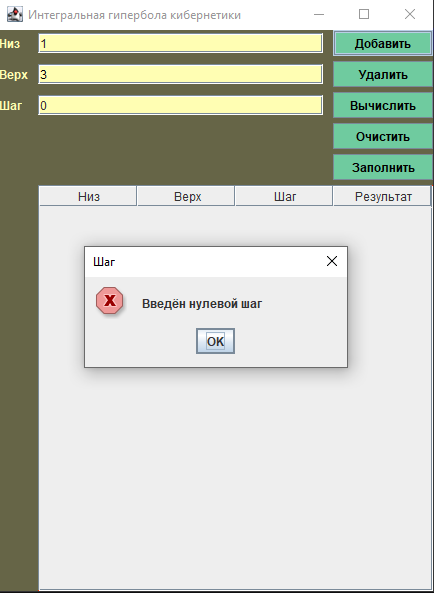


Рисунок №1 – Вывод сообщения о том, что нельзя вводить шаг, равный нулю

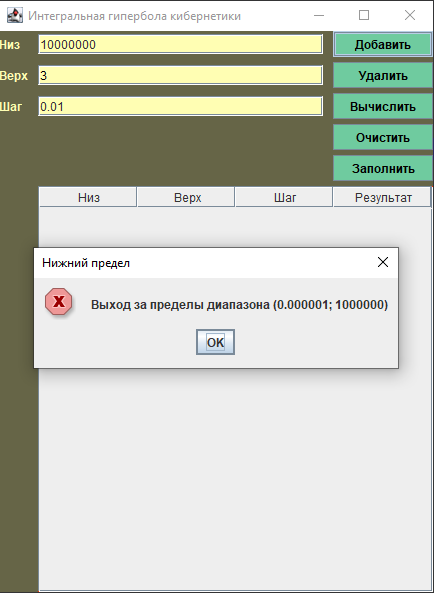


Рисунок №2 - Вывод сообщения о том, что нельзя вводить значение нижнего предела, которое выходит за пределы разрешённого интервала (0.000001; 1000000)

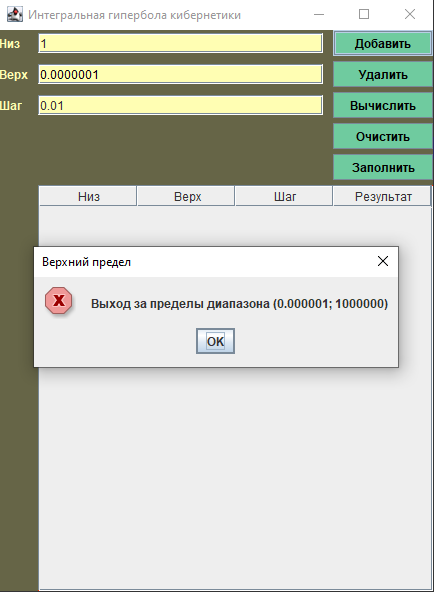


Рисунок №3 - Вывод сообщения о том, что нельзя вводить значение верхнего предела, которое выходит за пределы разрешённого интервала (0.000001; 1000000)

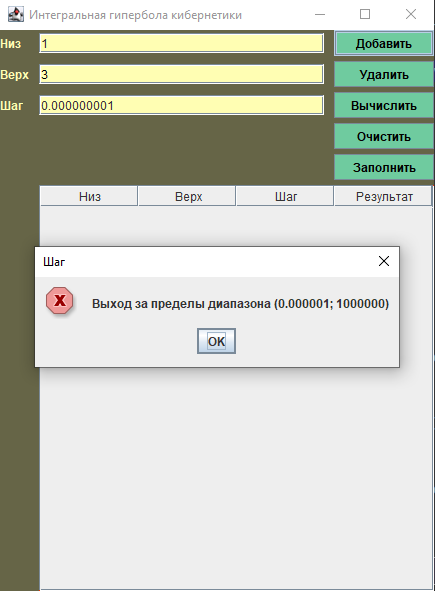


Рисунок №4 - Вывод сообщения о том, что нельзя вводить значение шага, которое выходит за пределы разрешённого интервала (0.000001; 1000000)

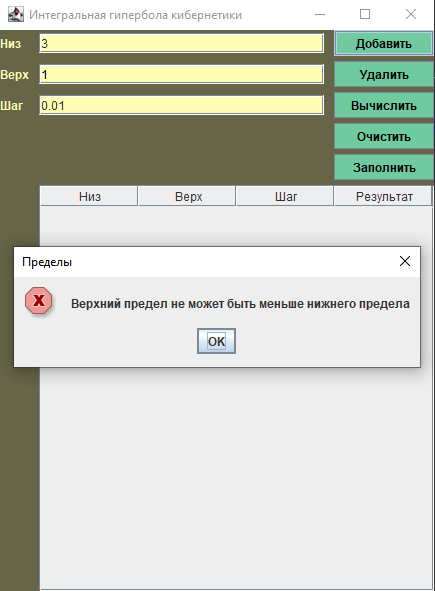


Рисунок №5 - Вывод сообщения о том, что нельзя вводить значения пределов так, чтобы верхний предел был меньше нижнего предела

**Вывод:** Мы изучили механизм обработки исключительных ситуаций.