Министерство науки и высшего образования РФ

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №3

по курсу «Программирование на языке JAVA»

Выполнили:

студенты группы 20ВВП2

Киреев Б.П.

Барсуков Н.И.

Проверил:

Юрова О.В.

Карамышева Н.С.

Пенза 2023

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

**Вариант №6:** sqrt(x), LinkedList, Exceptions

**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. Оформление лабораторной работы должно быть выполнено в соответствии с требованиями, приведенными в Приложении 2.

**Листинг:**

Main.java

public class Main {  
 public static void main(String[] args) {  
 new MainForm();  
 }  
}

MainForm.java

import javax.swing.\*;  
import javax.swing.table.DefaultTableModel;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.util.LinkedList;  
import java.util.Vector;  
  
public class MainForm extends JFrame {  
 private JTextField TextField\_UpperLimit;  
 private JTextField TextField\_LowerLimit;  
 private JPanel WorkPlace;  
 private JButton addButton;  
 private JButton deleteButton;  
 private JTable table1;  
 private JButton calculateButton;  
 private JLabel JLabel\_UpperLimit;  
 private JLabel JLabel\_LowerLimit;  
 private JLabel JLabel\_Step;  
 private JTextField TextField\_Step;  
 private JButton fillButton;  
 private JButton clearButton;  
 private DefaultTableModel model;  
  
 public LinkedList<RecIntegral> list = new LinkedList<>();  
  
 private Double Calculate(Double upper, Double low, Double step)  
 {  
 Double result = 0.0;  
 for(int i = 0; i < (upper - low)/step; i ++)  
 {  
 if (low + (i+1) \* step < upper)  
 result += 0.5 \* step \* (Math.*sqrt*(low + i \* step) + Math.*sqrt*(low + (i+1) \* step));  
 else  
 result += 0.5 \* step \* (Math.*sqrt*(low + i \* step) + Math.*sqrt*(upper));  
 }  
 return result;  
 }  
  
 public MainForm(){  
 model = (DefaultTableModel)table1.getModel();  
 model.addColumn("Верхний предел");  
 model.addColumn("Нижний предел");  
 model.addColumn("Шаг");  
 model.addColumn("Результат");  
 table1.setModel(model);  
  
 addButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
  
 try {  
 if (Double.*parseDouble*(TextField\_LowerLimit.getText()) > 1000000 ||  
 Double.*parseDouble*(TextField\_LowerLimit.getText()) < 0.000001)  
 {  
 throw new MyException("Выход за пределы диапазона", "Нижний лимит");  
 }  
  
 if (Double.*parseDouble*(TextField\_UpperLimit.getText()) > 1000000 ||  
 Double.*parseDouble*(TextField\_UpperLimit.getText()) < 0.000001)  
 {  
 throw new MyException("Выход за пределы диапазона", "Верхний лимит");  
 }  
  
 if (Double.*parseDouble*(TextField\_Step.getText()) > 1000000 ||  
 Double.*parseDouble*(TextField\_Step.getText()) < 0.000001)  
 {  
 throw new MyException("Выход за пределы диапазона", "Шаг");  
 }  
  
 if (Double.*parseDouble*(TextField\_Step.getText()) != 0)  
 {  
 throw new MyException("Шаг не может быть равен нулю", "Шаг");  
 }  
 } catch (MyException ex)  
 {  
 ex.ShowMessage(WorkPlace);  
 return;  
 }  
  
 model.addRow(new Object[]{  
 Double.*parseDouble*(TextField\_UpperLimit.getText()),  
 Double.*parseDouble*(TextField\_LowerLimit.getText()),  
 Double.*parseDouble*(TextField\_Step.getText()),  
 0  
 });  
 list.add(new RecIntegral(Double.*parseDouble*(TextField\_UpperLimit.getText()),  
 Double.*parseDouble*(TextField\_LowerLimit.getText()),  
 Double.*parseDouble*(TextField\_Step.getText()),  
 0.0));  
  
 TextField\_UpperLimit.setText("");  
 TextField\_LowerLimit.setText("");  
 TextField\_Step.setText("");  
 }  
 });  
  
 deleteButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (table1.getSelectedRow() != -1){  
 list.remove(table1.getSelectedRow());  
 model.removeRow(table1.getSelectedRow());  
 }  
 }  
 });  
  
 calculateButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 for (int i = 0; i < model.getRowCount(); i++)  
 {  
 Vector data = model.getDataVector().get(i);  
  
 Double upper = (Double)data.get(0);  
 Double low = (Double)data.get(1);  
 Double step = (Double)data.get(2);  
  
 Double result = Calculate(upper, low, step);  
 data.set(3, result);  
  
 list.get(i).res = result;  
 }  
  
 table1.repaint();  
 }  
 });  
  
 setContentPane(WorkPlace);  
 setSize(1000, 600);  
 setVisible(true);  
 setDefaultCloseOperation(WindowConstants.*EXIT\_ON\_CLOSE*);  
  
 fillButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 for (RecIntegral memList :  
 list) {  
 model.addRow(new Object[]{  
 memList.upLim,  
 memList.lowLim,  
 memList.st,  
 memList.res  
 });  
 }  
 }  
 });  
 clearButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 int modelSize = model.getRowCount() - 1;  
 for (int i = modelSize; i >= 0; i--)  
 {  
 model.removeRow(i);  
 }  
 }  
 });  
 }  
  
 private void createUIComponents() {  
 table1 = new JTable() {  
 @Override  
 public boolean isCellEditable(int row, int col) {  
 return (col != 3);  
 }  
 };  
 }  
  
};

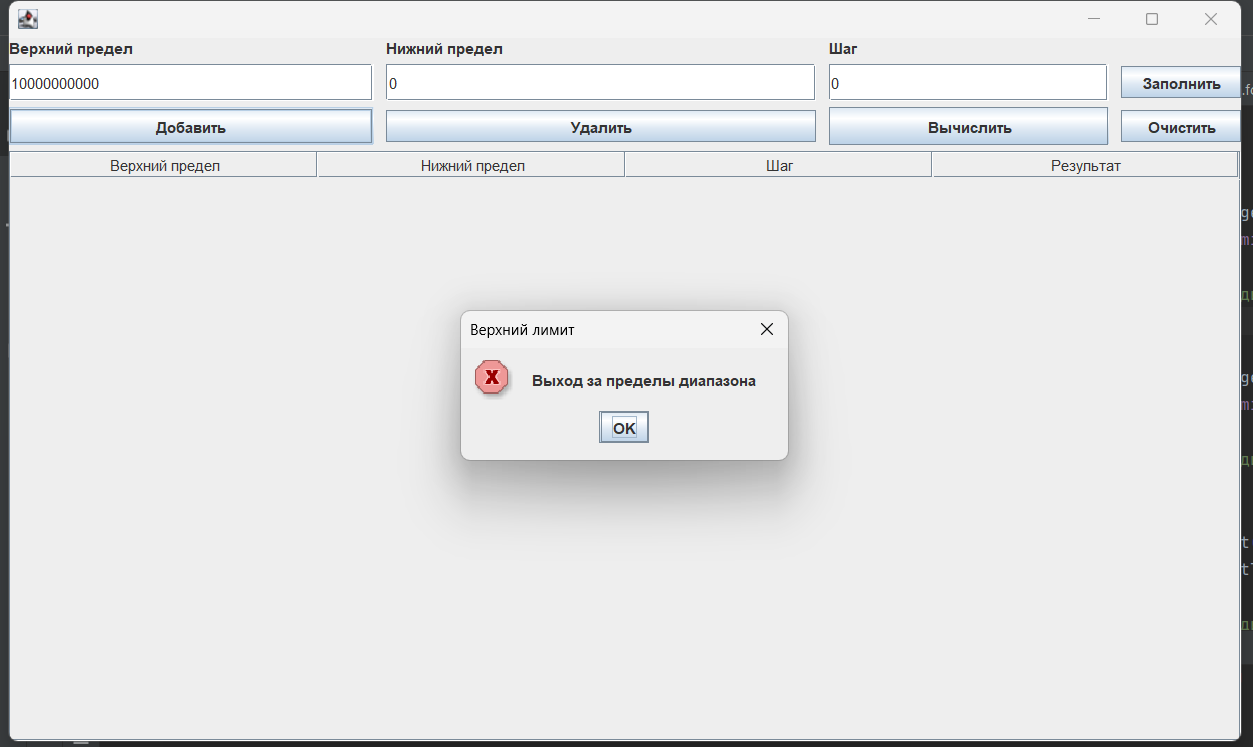
RecIntegral.java

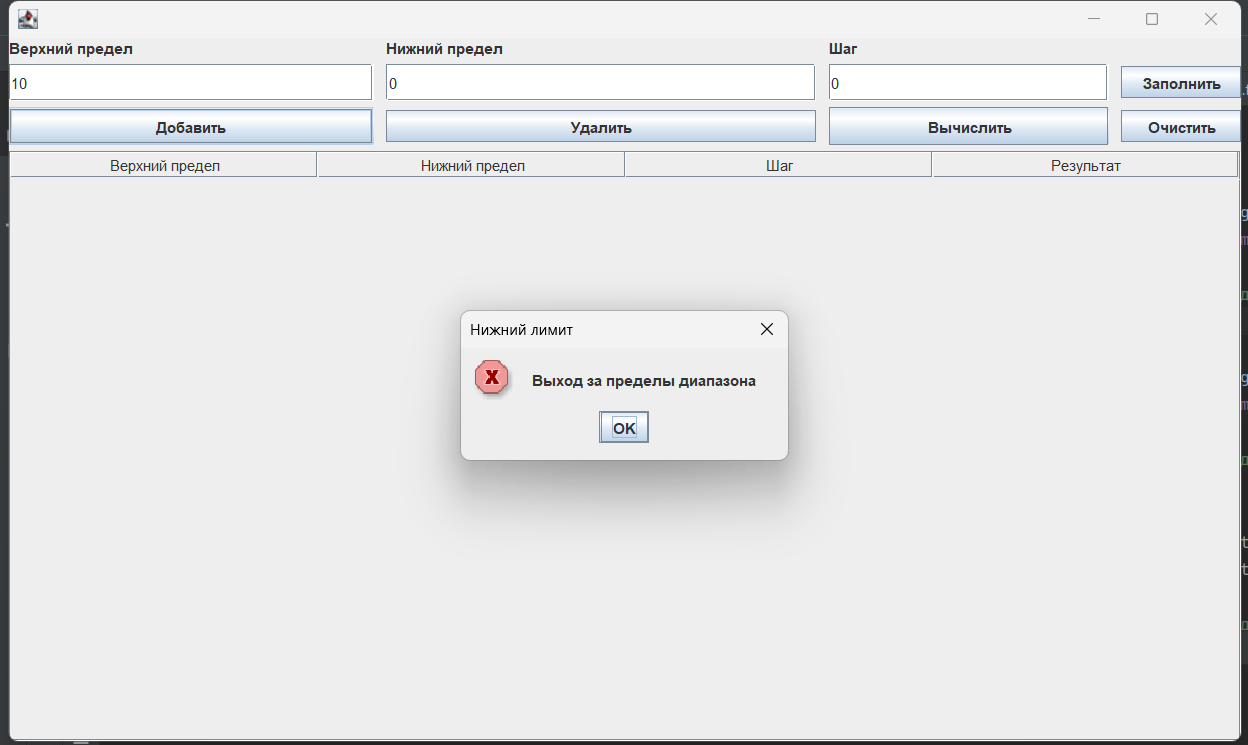
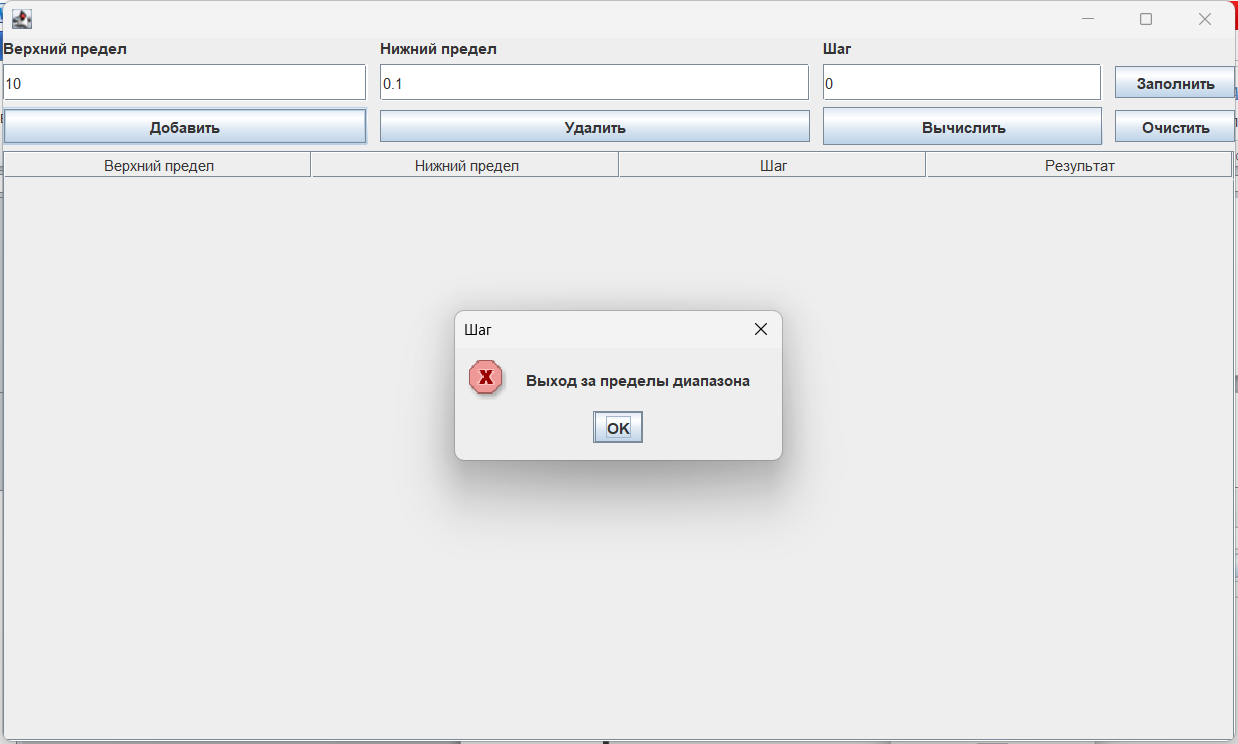
public class RecIntegral {  
 public Double upLim;  
 public Double lowLim;  
 public Double st;  
 public Double res;  
  
 public RecIntegral(Double upLim, Double lowLim, Double st, Double res) {  
 this.upLim = upLim;  
 this.lowLim = lowLim;  
 this.st = st;  
 this.res = res;  
 }  
}

MyException.java

import javax.swing.\*;  
import java.awt.\*;  
  
public class MyException extends Exception {  
 private String message;  
 private String title;  
  
 public MyException(String message, String title) {  
 this.message = message;  
 this.title = title;  
 }  
  
 public void ShowMessage(Component parent)  
 {  
 JOptionPane.*showMessageDialog*(parent, message, title, JOptionPane.*ERROR\_MESSAGE*);  
 }  
}

**Результат работы:**



**Вывод:** в ходе выполнения лабораторной работы мы изучили механизм обработки исключительных ситуаций.