Министерство науки и высшего образования Российской федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №3

по курсу «Программирование на языке Java»

на тему «Обработка исключительных ситуаций»

Вариант 5

Выполнили

студенты группы 21ВВП2:

Лукина Е.Ю.

Ванюшин И.А.

Тарасов Н.А.

Приняли:

Юрова О.В.

Карамышева Н.С.

Пенза 2024

**Цель работы**

Изучить механизм обработки исключительных ситуаций.

**Задание на лабораторную работу**

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.

|  |  |  |
| --- | --- | --- |
| Номер варианта | Функция | Класс-коллекция |
| 5 |  | ArrayList |

**Ход работы**

Написали пользовательский класс, наследующий класс Exception для реализации пользовательских исключений.

Ниже представлен листинг класса IntegralInputException.

public class IntegralInputException extends Exception {

public IntegralInputException() { super(); }

public IntegralInputException(String message) { super(message); }

public IntegralInputException(String message, Throwable cause) { super(message, cause); }

public IntegralInputException(Throwable cause) { super(cause); }

}

Модифицировали классы IntegralCalculator и RecIntegral, добавив в них различные проверки пределов вычислений определенного интеграла, а также добавили блоки Try/Catch в код основной программы, вызывающий всплывающие диалоги с сообщением об ошибке.

**Листинг**

IntegralInputException.java

package lab1;

/\*\*

\*

\* @author User

\*/

public class IntegralInputException extends Exception {

public IntegralInputException() { super(); }

public IntegralInputException(String message) { super(message); }

public IntegralInputException(String message, Throwable cause) { super(message, cause); }

public IntegralInputException(Throwable cause) { super(cause); }

}

IntegralCalculator.java

package lab1;

/\*\*

\*

\* @author User

\*/

public class IntegralCalculator {

public IntegralCalculator()

{

a = 0;

b = 1;

e = 1.0e-4;

}

public IntegralCalculator(double \_a, double \_b, double \_e) throws IntegralInputException

{

if (!(\_a <= 1.0e+6 && \_a >= -1.0e+6)) throw new IntegralInputException("Left limit is out of limit.");

if (!(\_b <= 1.0e+6 && \_b >= -1.0e+6)) throw new IntegralInputException("Right limit is out of limit.");

if (\_e == 0) throw new IntegralInputException("Step cannot be zero.");

if (\_e > Math.abs(\_a) + Math.abs(\_b)) throw new IntegralInputException("Step is greater than integral limit.");

if (\_e <= 1.0e+6 && \_e >= 1.0e-6) e = Math.abs(\_e);

else throw new IntegralInputException("Step is out of limit.");

if (\_a > \_b)

{

a = \_b;

b = \_a;

}

else

{

a = \_a;

b = \_b;

}

}

private double e, a, b;

public double E() { return e; }

public double A() { return a; }

public double B() { return b; }

public void SetE(double \_e) throws IntegralInputException

{

if (\_e == 0) throw new IntegralInputException("Step cannot be zero.");

if (\_e > Math.abs(a) + Math.abs(b)) throw new IntegralInputException("Step is greater than integral limit.");

if (\_e <= 1.0e+6 && \_e >= 1.0e-6) e = Math.abs(\_e);

else throw new IntegralInputException("Step is out of limit.");

}

public void SetArea(double \_a, double \_b) throws IntegralInputException {

if (!(\_a <= 1.0e+6 && \_a >= -1.0e+6)) throw new IntegralInputException("Left limit is out of limit.");

if(!(\_b <= 1.0e+6 && \_b >= -1.0e+6)) throw new IntegralInputException("Right limit is out of limit.");

if (e > Math.abs(\_a) + Math.abs(\_b)) throw new IntegralInputException("Step is greater than integral limit.");

if (\_a > \_b)

{

a = \_b;

b = \_a;

}

else

{

a = \_a;

b = \_b;

}

}

public double F()

{

double result = 0;

double lBound = a, rBound = a + e;

while (rBound < b)

{

result += e \* (Math.exp(-lBound) + Math.exp(-rBound)) / 2;

lBound += e;

rBound += e;

}

if (lBound < b) result += e \* (Math.exp(-lBound) + Math.exp(-b)) / 2;

return result;

}

}

RecIntegral.java

package lab1;

/\*\*

\*

\* @author User

\*/

public class RecIntegral {

public RecIntegral()

{

limitL = "";

limitR = "";

dx = "";

result = "";

}

public RecIntegral(IntegralCalculator src)

{

limitL = String.valueOf(src.A());

limitR = String.valueOf(src.B());

dx = String.valueOf(src.E());

result = String.valueOf(src.F());

}

public RecIntegral(double \_limitL, double \_limitR, double \_dx, double \_result) throws IntegralInputException

{

if (!(\_limitL <= 1.0e+6 && \_limitL >= -1.0e+6)) throw new IntegralInputException("Left limit is out of limit.");

if (!(\_limitR <= 1.0e+6 && \_limitR >= -1.0e+6)) throw new IntegralInputException("Right limit is out of limit.");

if (\_dx == 0) throw new IntegralInputException("Step cannot be zero.");

if (\_dx > Math.abs(\_limitL) + Math.abs(\_limitR)) throw new IntegralInputException("Step is greater than integral limit.");

if (!(\_dx <= 1.0e+6 && \_dx >= 1.0e-6) ) throw new IntegralInputException("Step is out of limit.");

if (!(\_result <= 1.0e+6 && \_result >= -1.0e+6)) throw new IntegralInputException("Result is out of limit.");

limitL = String.valueOf(\_limitL);

limitR = String.valueOf(\_limitR);

dx = String.valueOf(\_dx);

result = String.valueOf(\_result);

}

private String limitL, limitR, dx, result;

public String LimitL() { return limitL; }

public String LimitR() { return limitR; }

public String Dx() { return dx; }

public String Result() { return result; }

public void SetLimitL(double \_limitL) throws IntegralInputException

{

if (!(\_limitL <= 1.0e+6 && \_limitL >= -1.0e+6)) throw new IntegralInputException("Left limit is out of limit.");

if (Double.parseDouble(dx) > Math.abs(\_limitL) + Math.abs(Double.parseDouble(limitR))) throw new IntegralInputException("Step is greater than integral limit.");

limitL = String.valueOf(\_limitL);

}

public void SetLimitR(double \_limitR) throws IntegralInputException

{

if (!(\_limitR <= 1.0e+6 && \_limitR >= -1.0e+6)) throw new IntegralInputException("Right limit is out of limit.");

if (Double.parseDouble(dx) > Math.abs(Double.parseDouble(limitL)) + Math.abs(\_limitR)) throw new IntegralInputException("Step is greater than integral limit.");

limitR = String.valueOf(\_limitR);

}

public void SetDx(double \_dx) throws IntegralInputException

{

if (\_dx == 0) throw new IntegralInputException("Step cannot be zero.");

if (!(\_dx <= 1.0e+6 && \_dx >= 1.0e-6) ) throw new IntegralInputException("Step is out of limit.");

if (\_dx > Math.abs(Double.parseDouble(limitL)) + Math.abs(Double.parseDouble(limitR))) throw new IntegralInputException("Step is greater than integral limit.");

dx = String.valueOf(\_dx);

}

public void SetResult(double \_result) throws IntegralInputException

{

if (!(\_result <= 1.0e+6 && \_result >= -1.0e+6)) throw new IntegralInputException("Result is out of limit.");

result = String.valueOf(\_result);

}

public void Clear()

{

limitL = "";

limitR = "";

dx = "";

result = "";

}

}

MyWin.java

package lab1;

import java.awt.Container;

import java.awt.Dimension;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.ArrayList;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.JTextField;

import javax.swing.table.DefaultTableModel;

import javax.swing.JOptionPane;

import javax.swing.JDialog;

public class MyWin extends JFrame implements ActionListener

{

// Some window's elements

private final JTextField jtfA;

private final JTextField jtfB;

private final JTextField jtfE;

private final DefaultTableModel dtResultsModel;

private final JTable jtResults;

private final JDialog jdFrame;

private final ArrayList alRecs;

// Класс для вычисления определенного интеграла

private final IntegralCalculator ic;

// Classes serial number

private static final long serialVersionUID = 1L;

public MyWin()

{

// Initialization block

jdFrame = new JDialog();

alRecs = new ArrayList();

ic = new IntegralCalculator();

jtfA = new JTextField();

jtfB = new JTextField();

jtfE = new JTextField();

//----------------==Table Filling==------------------------------

Object[] columnNames = new String[]{"L limit", "R limit", "Step", "Result"};

Object[][] data = new String[][]{};

dtResultsModel = new DefaultTableModel(data, columnNames);

dtResultsModel.setColumnIdentifiers(columnNames);

jtResults = new JTable(dtResultsModel);

JScrollPane sp = new JScrollPane(jtResults);

Container c = getContentPane(); // Client rect

c.setLayout(null); // НЕНАВИЖУ КОМПАНОВЩИКИ

//----------------==Left Panel's Filling==-----------------------

// Creates a left sided panel

JPanel jpLeftPanel = new JPanel();

jpLeftPanel.setLayout(null); // АГРЕССИЯ И ЗУБЫ СКРИПЯТ

jpLeftPanel.setBounds(0, 0, 200, 480); // Resizes left panel

// Creates buttons.

// The one that Adds new records & the other that Clears all inputs.

JButton jbtnAdd = new JButton("Add");

JButton jbtnClear = new JButton("Clear");

JButton jbtnDelete = new JButton("Delete");

JButton jbtnLoad = new JButton("Load");

// Buttons's resize.

jbtnAdd.setBounds(10, 200, 180, 50);

jbtnClear.setBounds(10, 260, 180, 50);

jbtnLoad.setBounds(10, 320, 180, 50);

jbtnDelete.setBounds(10, 380, 180, 50);

// Adds action listeners for the objects.

jbtnAdd.addActionListener(this);

jbtnClear.addActionListener(this);

jbtnDelete.addActionListener(this);

jbtnLoad.addActionListener(this);

jtfA.setBounds(10, 10, 180, 50);

jtfB.setBounds(10, 70, 180, 50);

jtfE.setBounds(10, 130, 180, 50);

jtfA.setText(Double.toString(ic.A()));

jtfB.setText(Double.toString(ic.B()));

jtfE.setText(Double.toString(ic.E()));

sp.setBounds(200, 0, 427, 480);

// Fills panel with early created buttons & txtFields.

jpLeftPanel.add(jtfA);

jpLeftPanel.add(jtfB);

jpLeftPanel.add(jtfE);

jpLeftPanel.add(jbtnAdd);

jpLeftPanel.add(jbtnClear);

jpLeftPanel.add(jbtnDelete);

jpLeftPanel.add(jbtnLoad);

//----------------==Client rect's Filling==-----------------------

// Adds left panel to actually left side of client rect.

c.add(jpLeftPanel);

c.add(sp);

//----------------==Setting window's properties==-----------------

// Window's properties.

setTitle("Integral Calculator"); // Window's title. Obviously.

// Sets preffered size to window.

setPreferredSize(new Dimension(640, 480));

// Exit application on Exit button.

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

pack(); // Setting preffered sizes.

setVisible(true); // Makes window visible.

}

@Override

public void actionPerformed(ActionEvent e) {

// "Calculate" case. Calculates the integral with setted parameters.

if (e.getActionCommand().equals("Add"))

{

try

{

ic.SetArea(Double.parseDouble(jtfA.getText()), Double.parseDouble(jtfB.getText()));

ic.SetE(Double.parseDouble(jtfE.getText()));

alRecs.add(new RecIntegral(ic));

jtfE.setText(((RecIntegral)alRecs.getLast()).Dx());

dtResultsModel.addRow(new String[]

{

String.valueOf(ic.A()),

String.valueOf(ic.B()),

String.valueOf(ic.E()),

((RecIntegral)alRecs.getLast()).Result()

});

}

catch(IntegralInputException exc)

{

JOptionPane.showMessageDialog(

jdFrame,

exc.getMessage(),

"Error",

JOptionPane.ERROR\_MESSAGE

);

}

catch(NumberFormatException exc)

{

JOptionPane.showMessageDialog(

jdFrame,

exc.getMessage(),

"Error",

JOptionPane.ERROR\_MESSAGE

);

}

}

else if (e.getActionCommand().equals("Clear"))

{

dtResultsModel.setRowCount(0);

}

else if (e.getActionCommand().equals("Load"))

{

dtResultsModel.setRowCount(0);

for (int i = 0; i < alRecs.size(); i++)

{

dtResultsModel.addRow(new String[]

{

((RecIntegral)alRecs.get(i)).LimitL(),

((RecIntegral)alRecs.get(i)).LimitR(),

((RecIntegral)alRecs.get(i)).Dx(),

((RecIntegral)alRecs.get(i)).Result()

});

}

}

else if (e.getActionCommand().equals("Delete"))

{

int i = jtResults.getSelectedRow();

if (i != -1)

{

dtResultsModel.removeRow(i);

alRecs.remove(i);

}

}

else throw new UnsupportedOperationException("Not supported yet."); // Unprocessed action case.

}

// запуск оконного приложения

public static void main(String args[]) {

new MyWin();

}

}

**Результат работы программы**

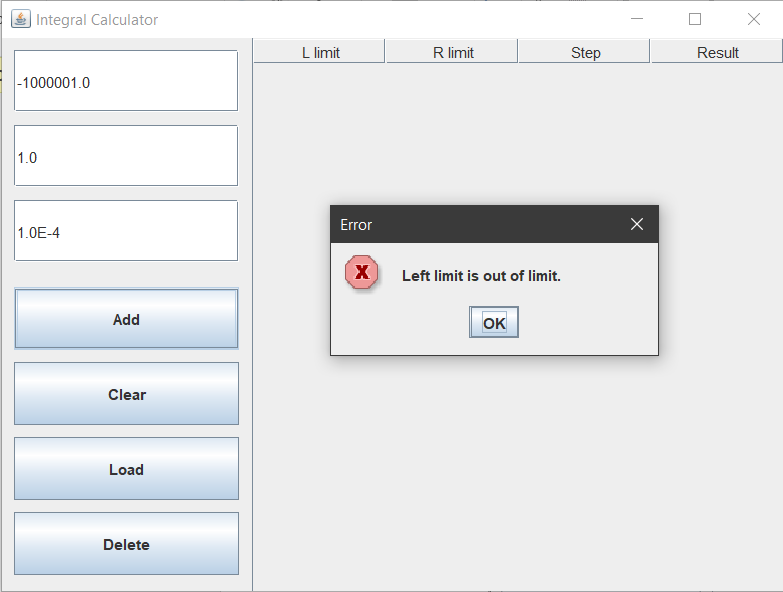
Результаты работы программы представлены на рисунках 1-5. 

Рис.1 – Состояние программы при превышении левой границы интегрирования

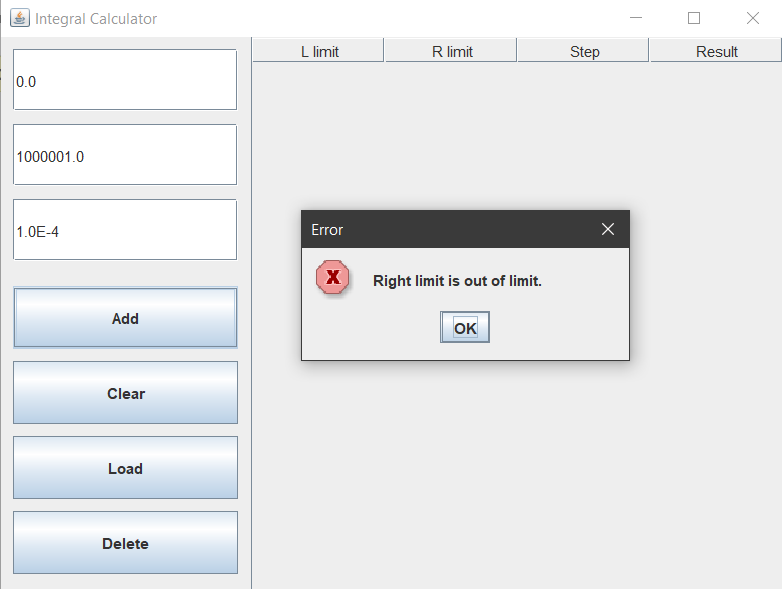


Рис.2 – Состояние программы при превышении правой границы интегрирования

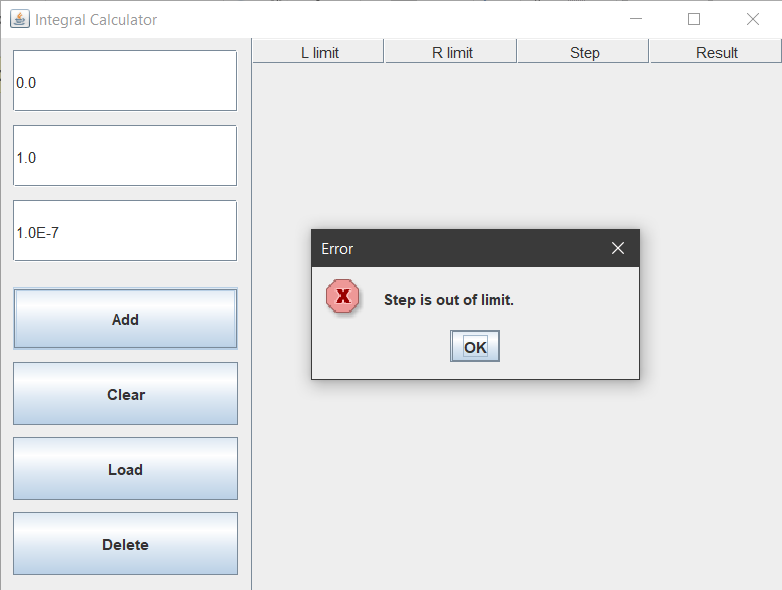


Рис.3 – Состояние программы при превышении точности шага

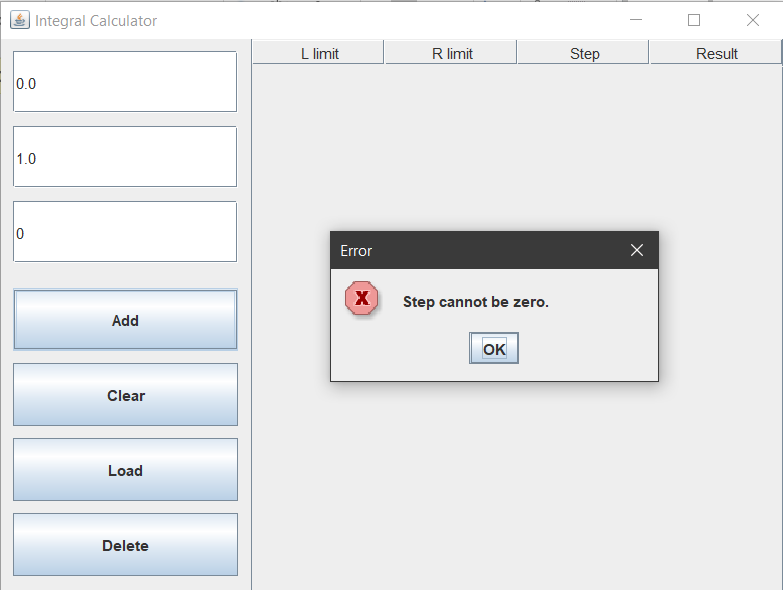


Рис.4 – Состояние программы при попытке задать шагу значение нуля

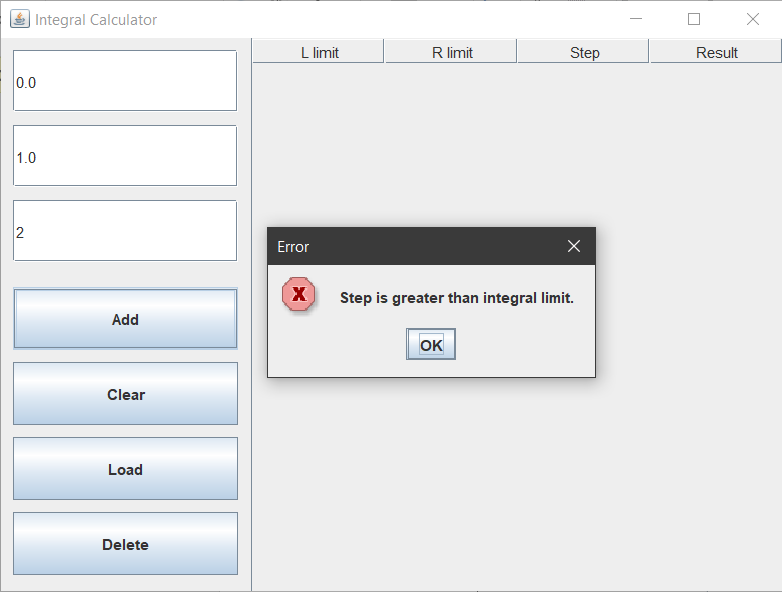


Рис.5 – Состояние программы при попытке задать шагу значение больше модуля разницы границ интегрирования

Результат программы совпал с ожидаемым, следовательно, программа составлена верно.

**Вывод**

В результате выполнения методических указаний к лабораторной работе были получены навыки работы с графической оболочкой Java swing и классом исключений.