Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №3

по курсу «Программные средства проектирования цифровых устройств»

на тему «Обработка исключительных ситуаций»

**Выполнили студенты группы 21ввв1:**

Лобанов Р.И.

Антропов Д.В.

**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.

**Листинг:**

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.HashMap;

import java.util.Iterator;

import java.util.Map;

public class NewForm {

double lowBorder = 0; //down

double highBorder = 0; //upp

double step = 0; //step

private JPanel panel1;

private JTextField lowBorderTextField;

private JTextField highBorderTextField;

private JTextField stepTextField;

private JButton CalcButton;

private JButton AddButton;

private JButton DelButton;

private JTable table1;

private JScrollPane scrollPane1;

private JButton AddRecToTableButton;

private JButton DeleteRecFromCollectionButton;

private final HashMap<RecIntegral,Double> collectionHashMap = new HashMap();

private final HashMap<RecIntegral,Double> tableHashMap = new HashMap();

DefaultTableModel model = (DefaultTableModel) table1.getModel();

public NewForm() {

CalcButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

for(int row = 0; row < table1.getRowCount(); row++){

if(table1.getValueAt(row,0) != "" && table1.getValueAt(row,1) != "" && table1.getValueAt(row,2) != ""){

step = Double.parseDouble(table1.getValueAt(row, 0).toString());

highBorder = Double.parseDouble(table1.getValueAt(row, 1).toString());

lowBorder = Double.parseDouble(table1.getValueAt(row, 2).toString());

RecIntegral rec = new RecIntegral(lowBorder,highBorder,step);

double answer = integrate(rec);

table1.setValueAt(answer, row, 3);

tableHashMap.put(rec,answer);

collectionHashMap.put(rec,answer);

}

}

}

});

AddButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String step = stepTextField.getText();

String highBorder = highBorderTextField.getText();

String lowBorder = lowBorderTextField.getText();

RecIntegral rec = new RecIntegral(

Double.parseDouble(lowBorder),

Double.parseDouble(highBorder),

Double.parseDouble(step));

if(!tableHashMap.containsKey(rec)){

tableHashMap.put(rec,null);

model.addRow(new Object[]{step, highBorder, lowBorder});

}

}

});

DelButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

int selectedRow = table1.getSelectedRow();

if (selectedRow != -1){

tableHashMap.remove(

new RecIntegral(

Double.parseDouble(table1.getValueAt(selectedRow,2).toString()),

Double.parseDouble(table1.getValueAt(selectedRow,1).toString()),

Double.parseDouble(table1.getValueAt(selectedRow,0).toString())

)

);

model.removeRow(selectedRow);

}

}

});

AddRecToTableButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

for (Map.Entry<RecIntegral,Double> entry:collectionHashMap.entrySet()){

RecIntegral rec = entry.getKey();

if(!tableHashMap.containsKey(rec)){

tableHashMap.put(rec,entry.getValue());

model.addRow(new Object[]{Double.toString(rec.step),

Double.toString(rec.highBorder),

Double.toString(rec.lowBorder),

Double.toString(entry.getValue())});

}

else{

if(tableHashMap.get(rec)==null){

for(int i = 0; i< model.getRowCount();i++){

if(model.getValueAt(i,0).toString().equals(Double.toString(rec.step)))

continue;

if(model.getValueAt(i,1).toString().equals(Double.toString(rec.highBorder)))

continue;

if(model.getValueAt(i,2).toString().equals(Double.toString(rec.lowBorder)))

continue;

model.removeRow(i);

tableHashMap.put(rec,entry.getValue());

model.addRow(new Object[]{Double.toString(rec.step),

Double.toString(rec.highBorder),

Double.toString(rec.lowBorder),

Double.toString(entry.getValue())});

break;

}

}

}

}

}

});

DeleteRecFromCollectionButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

collectionHashMap.clear();

}

});

}

public static void main(String[] args) {

JFrame frame = new JFrame("Integrator √x");

frame.setLocationRelativeTo(null);

frame.setContentPane(new NewForm().panel1);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.pack();

frame.setVisible(true);

}

public static double integrate(double lowBorder,double highBorder,double step){

double square;

double sum = 0;

double x;

sum += Math.exp(-lowBorder);

for (x = lowBorder+step;x < highBorder;x+=step){

if (x >= highBorder){

x-=step;

break;

}

sum += 2\*Math.exp(-x);

}

square = step\*sum/2;

square += (highBorder-x)\*Math.exp(-highBorder)/2;

return square;

}

public static double integrate(RecIntegral rec){

return integrate(rec.lowBorder, rec.highBorder, rec.step);

}

private void createUIComponents() {

scrollPane1 = new JScrollPane(table1);

String[] columnNames = {"Шаг", "Верхний", "Нижний", "Результат"};

String[][] data = {

};

table1 = new JTable(data, columnNames);

table1.setModel(new DefaultTableModel(data, columnNames)

{public boolean isCellEditable(int row, int column){

return column != 3;

}}

);

}

}

Class RecIntegral:

import java.util.Objects;

public class RecIntegral {

double lowBorder;

double highBorder;

double step;

RecIntegral(){

}

RecIntegral(double lowBorder, double highBorder,double step){

this.lowBorder = lowBorder;

this.highBorder = highBorder;

this.step = step;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (!(o instanceof RecIntegral)) return false;

RecIntegral that = (RecIntegral) o;

return Double.compare(that.lowBorder, lowBorder) == 0 && Double.compare(that.highBorder, highBorder) == 0 && Double.compare(that.step, step) == 0;

}

@Override

public int hashCode() {

return Objects.hash(lowBorder, highBorder, step);

}

}

Class MyException:

import javax.swing.\*;

public class MyException extends Exception{

public MyException(){super();}

public MyException(double lowBorder,double highBorder,double step){

super();

String message ="";

if(lowBorder<0.000001 || lowBorder>1000000)

message+="Нижняя граница должна быть больше 0.000001 и меньше 1000000\n";

if(highBorder<0.000001 || highBorder>1000000)

message+="Верхняя граница должна быть больше 0.000001 и меньше 1000000\n";

if(step<0.000001 || step>1000000)

message+="Шаг должен быть больше 0.000001 и меньше 1000000\n";

JOptionPane.showMessageDialog(new JOptionPane(),

message,

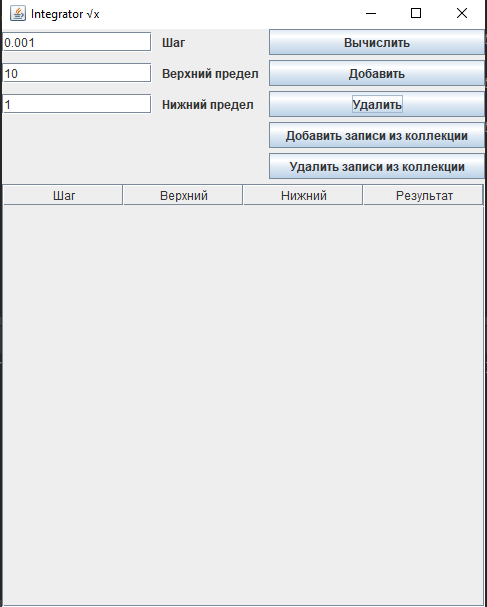
"MyException",

JOptionPane.ERROR\_MESSAGE);

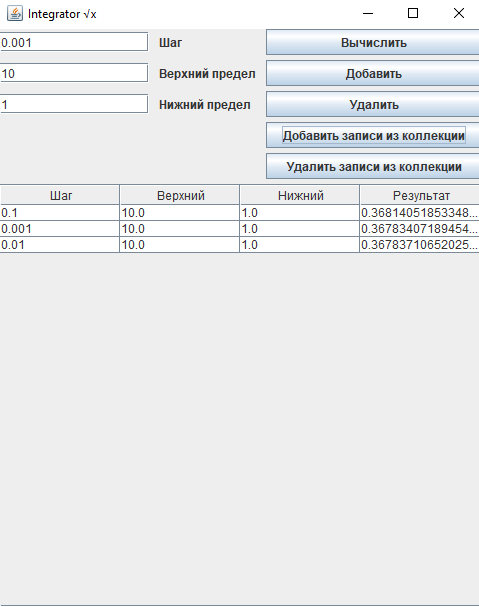
}

}

Вывод программы до нажатия кнопки “Добавить записи из коллекции”



Вывод программы после нажатия кнопки “Добавить записи из коллекции”



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