Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №3

по курсу «Программирование на языке Java»

на тему: «Работа с коллекциями объектов»

Вариант 9

Выполнили студенты группы 21ВВП2  
Сивущин А. С.

Савицкий М.А.

Ефимущкин А.Н.

Приняла

Юрова О. В.

Пенза 2024

**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:**

Создаем свой класс, унаследованный от класса Exception.

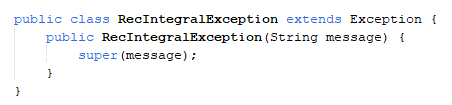


Рисунок 1

Реализуем проверку вводимых данных с использованием механизма исключений

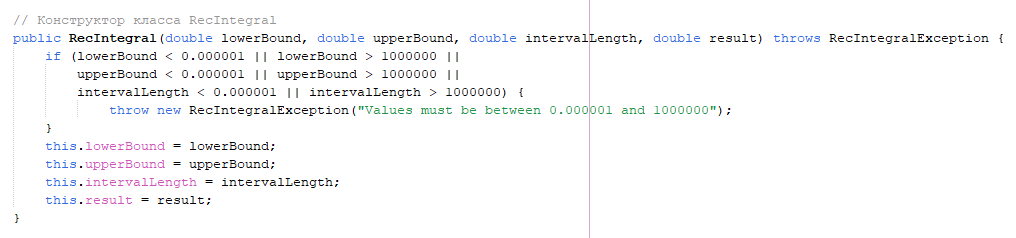


Рисунок 2

Генерируем исключение, если возникает попытка создать экземпляр класса RecIntegral со значениями, не являющимися числами в диапазоне от 0,000001 до 1000000

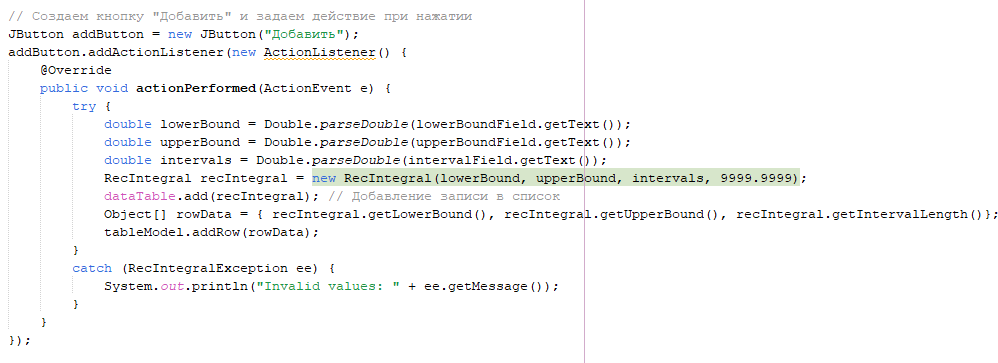


Рисунок 3

**Листинг:**

package com.mycompany.lab3;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import javax.swing.table.TableCellEditor;

import javax.swing.table.TableCellRenderer;

import javax.swing.table.TableColumn;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.ArrayList;

import java.util.List;

public class Lab3 extends JFrame {

private DefaultTableModel tableModel;

private List<RecIntegral> dataTable; // Коллекция для хранения данных таблицы

public Lab3() {

setTitle("Integration Calculator - cos(x^2) ");

// Создаем коллекцию для хранения данных таблицы

dataTable = new ArrayList<>();

// Создаем таблицу

String[] columns = {"Нижняя граница", "Верхняя граница", "Длина интервала", "Результат"};

DefaultTableModel tableModel = new DefaultTableModel(columns, 0){

@Override

public boolean isCellEditable(int row, int column){

// разрешаем редактирование всех столбцов кроме 4го

return column != 3;

}

};

JTable table = new JTable(tableModel){

@Override

public TableCellRenderer getCellRenderer(int row, int column){

if (column == 3){

// запрещаем редактирование 4го столбца

return getDefaultRenderer(Object.class);

}

else{

return super.getCellRenderer(row, column);

}

}

@Override

public TableCellEditor getCellEditor(int row, int column){

if (column == 3){

return getDefaultEditor(Object.class);

}

else{

return super.getCellEditor(row, column);

}

}

};

TableColumn column = table.getColumnModel().getColumn(3);

column.setCellEditor(null);

// Создаем текстовые поля для ввода данных

JTextField lowerBoundField = new JTextField(10);

JTextField upperBoundField = new JTextField(10);

JTextField intervalField = new JTextField(10);

// Создаем кнопку "Добавить" и задаем действие при нажатии

JButton addButton = new JButton("Добавить");

addButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

try {

double lowerBound = Double.parseDouble(lowerBoundField.getText());

double upperBound = Double.parseDouble(upperBoundField.getText());

double intervals = Double.parseDouble(intervalField.getText());

RecIntegral recIntegral = new RecIntegral(lowerBound, upperBound, intervals, 9999.9999);

dataTable.add(recIntegral); // Добавление записи в список

Object[] rowData = { recIntegral.getLowerBound(), recIntegral.getUpperBound(), recIntegral.getIntervalLength()};

tableModel.addRow(rowData);

}

catch (RecIntegralException ee) {

System.out.println("Invalid values: " + ee.getMessage());

}

}

});

// Создаем кнопку "Удалить" и задаем действие при нажатии

JButton deleteButton = new JButton("Удалить");

deleteButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

int selectedRow = table.getSelectedRow();

if (selectedRow != -1) {

tableModel.removeRow(selectedRow);

dataTable.remove(selectedRow);

}

}

});

// Создаем кнопку "Очистить" и задаем действие при нажатии

JButton clearButton = new JButton("Очистить");

clearButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

tableModel.setRowCount(0);

}

});

// Создаем кнопку "Заполнить" и задаем действие при нажатии

JButton fillButton = new JButton("Заполнить");

fillButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

for (RecIntegral recIntegral : dataTable)

tableModel.addRow(recIntegral.getDataAsStringArray());

}

});

JButton calculateButton = new JButton("Вычислить");

calculateButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

int selectedRow = table.getSelectedRow();

double lowerBound = Double.parseDouble(tableModel.getValueAt(selectedRow, 0).toString());

double upperBound = Double.parseDouble(tableModel.getValueAt(selectedRow, 1).toString());

double intervals = Double.parseDouble(tableModel.getValueAt(selectedRow, 2).toString());

double result = calculateIntegration(lowerBound, upperBound, intervals);

if (selectedRow != -1) {

tableModel.setValueAt(result, selectedRow, 3);

// Получаем объект RecIntegral из коллекции

RecIntegral recIntegral = dataTable.get(selectedRow);

// Изменяем первое значение интегрирования

double newValue = result;

recIntegral.setResult(newValue);

}

}

});

// Создаем панель для компонентов

JPanel panel = new JPanel();

panel.add(lowerBoundField);

panel.add(upperBoundField);

panel.add(intervalField);

panel.add(addButton);

panel.add(deleteButton);

panel.add(calculateButton);

panel.add(clearButton);

panel.add(fillButton);

// Добавляем панель и таблицу на окно

add(panel, "North");

JScrollPane scrollPane = new JScrollPane(table);

add(scrollPane);

setSize(1000, 400);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setVisible(true);

}

public class RecIntegralException extends Exception {

public RecIntegralException(String message) {

super(message);

}

}

// Класс для хранения записи таблицы

private class RecIntegral {

private double lowerBound;

private double upperBound;

private double intervalLength;

private double result;

// Конструктор класса RecIntegral

public RecIntegral(double lowerBound, double upperBound, double intervalLength, double result) throws RecIntegralException {

if (lowerBound < 0.000001 || lowerBound > 1000000 ||

upperBound < 0.000001 || upperBound > 1000000 ||

intervalLength < 0.000001 || intervalLength > 1000000) {

throw new RecIntegralException("Values must be between 0.000001 and 1000000");

}

this.lowerBound = lowerBound;

this.upperBound = upperBound;

this.intervalLength = intervalLength;

this.result = result;

}

// Возвращает значение нижней границы интегрирования

public double getLowerBound() {

return lowerBound;

}

// Возвращает значение верхней границы интегрирования

public double getUpperBound() {

return upperBound;

}

// Возвращает значение длины шага интегрирования

public double getIntervalLength() {

return intervalLength;

}

// Возвращает результат вычисления

public double getResult() {

return result;

}

private void setResult(double newResult) {

this.result = newResult;

}

public String[] getDataAsStringArray() {

String[] data = new String[4];

data[0] = String.valueOf(lowerBound);

data[1] = String.valueOf(upperBound);

data[2] = String.valueOf(intervalLength);

data[3] = String.valueOf(result);

return data;

}

}

// Метод для вычисления интеграла функции cos(x^2) методом трапеций

private double calculateIntegration(double lowerBound, double upperBound, double step) {

double sum = 0.0;

double x = lowerBound;

while (x < upperBound) {

double fx1 = Math.cos(x \* x); // значение функции в левой точке отрезка

double fx2 = Math.cos(Math.min(x + step, upperBound) \* Math.min(x + step, upperBound)); // значение функции в правой точке отрезка

sum += (fx1 + fx2) \* Math.min(step, upperBound - x) / 2;

x += step;

}

//округлим ответ

int decimalPlaces = 4; // количество знаков после запятой, до которого нужно округлить

sum = Math.round(sum \* Math.pow(10, decimalPlaces)) / Math.pow(10, decimalPlaces);

return sum; // возвращаем значение интеграла

}

public static void main(String[] args) {

new Lab3(); // создаем экземпляр приложения

}

}

**Результаты работы программы:**

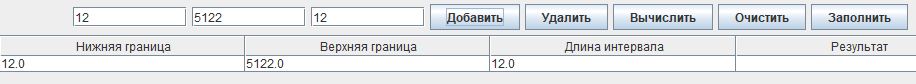


Рисунок 4 - спокойно добавляем значения в разрешенном диапазоне

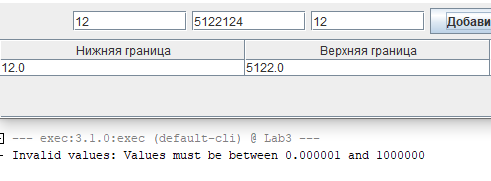


Рисунок 5 - не попадаем в разрешенный диапазон

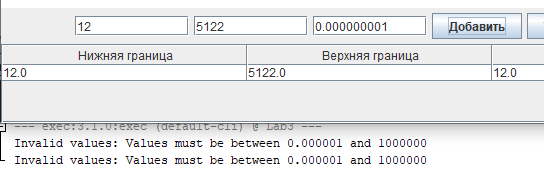


Рисунок 6 - не попадаем в разрешенный диапазон

**Вывод:** нами был изучен механизм обработки исключительных ситуаций.