|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| *Федеральное государственное бюджетное образовательное учреждение высшего образования* ***«МИРЭА – Российский технологический университет»***  **РТУ МИРЭА** |

**Институт информационных технологий (ИТ)**

**Кафедра инструментального и прикладного программного обеспечения (ИиППО)**

**Дисциплина «Программирование на языке Джава»**

**ОТЧЕТ**

**ПО ПРАКТИЧЕСКОМУ ЗАДАНИЮ №9**

Выполнил студент группы ИНБО-02-20 Веселкин.И.А

Принял Степанов П.В.

Практические работы выполнены «\_\_\_»\_\_\_\_\_\_\_2021г.

«\_\_\_\_\_\_\_\_\_\_\_\_\_\_» «\_\_\_»\_\_\_\_\_\_\_2021г.

Отметка о выполнении

**Москва – 2021 г.**

## **Задание**

Клиент совершает покупку онлайн. При оформлении заказа у пользователя запрашивается фио и номер ИНН. В программе проверяется, действителен ли номер ИНН для такого клиента. Исключение будет выдано в том случае, если введен недействительный ИНН.

Предлагается модернизировать задачу из предыдущей лабораторной работы (см. методические указания по выполнению лабораторных работ №1-8) – задача сортировки студентов по среднему баллу. Необходимо разработать пользовательский интерфейс для задачи поиска и сортировки (использовать массив интерфейсных ссылок- пример в лекции 5). Дополнить ее поиском студента по фио – в случае отсутствия такого студента необходимо выдавать собственное исключение.

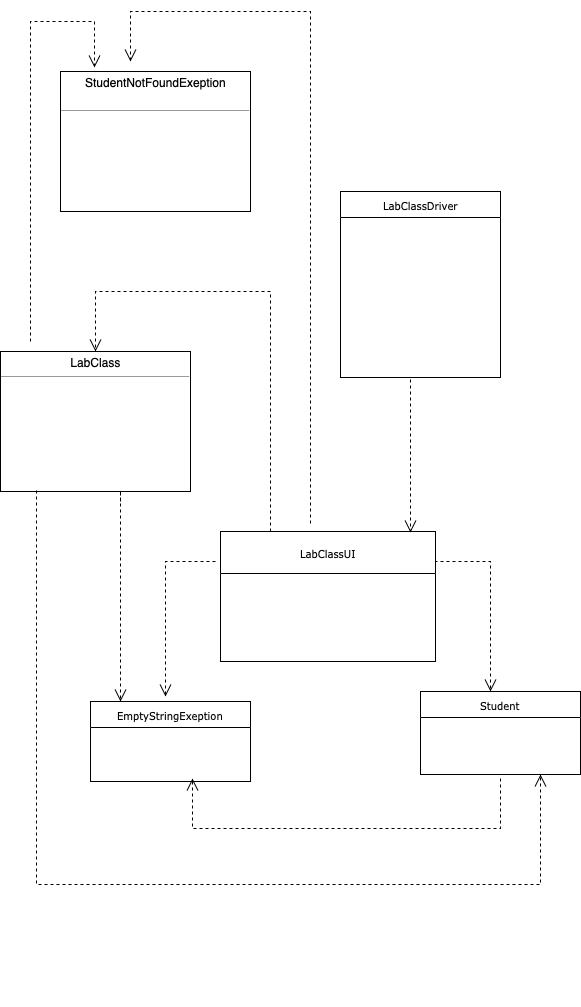


Рисунок 1. - UML диаграмма проекта LabClass с обработкой исключений

## **Ход Работы**

Реализация класса исключений EmptyStringExeption:

public class EmptyStringExeption extends Exception {  
 public EmptyStringExeption(String errorMessage){  
 super(errorMessage);  
 }  
}

Реализация класса исключений EmptyStringExeption:

public class StudentNotFoundException extends Exception{  
 public StudentNotFoundException(String errorMessage){  
 super(errorMessage);  
 }  
}

Реализация класса LabClass:

public class LabClass {  
 Student[] list=new Student[1];  
 public interface Comparator<Student>{  
 boolean compare(Student a, Student b);  
 }  
 public static class StudentsComparator implements LabClass.Comparator<Student> {  
 public boolean compare(Student a, Student b){  
 return a.getGrade()>b.getGrade();  
 }  
 }  
 static LabClass.StudentsComparator *comparator*=new LabClass.StudentsComparator();  
 int partition(Student list[], int low, int high) {  
 Student base = list[high];  
 int i = (low - 1);  
 for (int j = low; j < high; j++) {  
 if (!*comparator*.compare(list[j],base)) {  
 i++;  
 Student temp = list[i];  
 list[i] = list[j];  
 list[j] = temp;  
 }  
 }  
 Student temp = list[i + 1];  
 list[i + 1] = list[high];  
 list[high] = temp;  
 return i + 1;  
 }  
 void quickSort(Student list[], int low, int high)  
 {  
 if (low < high)  
 {  
 int pi = partition(list, low, high);  
 quickSort(list, low, pi-1);  
 quickSort(list, pi+1, high);  
 }  
 }  
 void clone(Student[] bigArray, Student[] smallArray){  
 int i=0;  
 for (Student s:smallArray){  
 bigArray[i]=s;  
 i++;  
 }  
 }  
 void clear(){  
 list=new Student[1];  
 }

Реализация класса LabClassDriver:

public class LabClassDriver {  
 public void run(Student[] list){  
 LabClass sort=new LabClass();  
 sort.quickSort(list,0,list.length-1);  
 }  
}

Реализация класса LabClassUI:

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
public class LabClassUI extends JFrame {  
 private LabClass lab=new LabClass();  
 private int width=350;  
 private int height=450;  
 private int current=0;  
 private JLabel inputName, inputINN, data, result;  
 private JButton button,getResult,refresh;  
 private JTextField name=new JTextField(15);  
 private JTextField INN=new JTextField(15);  
 private JTextArea input=new JTextArea(5,20);  
 private JTextArea output=new JTextArea(5,20);  
 public LabClassUI(){  
 setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 setTitle("Сортировка студентов");  
 setSize(width,height);  
 setResizable(false);  
 setLayout(new FlowLayout());  
 inputName=new JLabel("Введите имя студента: ");  
 inputINN=new JLabel("Введите ИНН студента:");  
 data=new JLabel("Студенты:");  
 result=new JLabel("Ответ: ");  
 button=new JButton("Добавить");  
 getResult=new JButton("Отсортировать");  
 refresh=new JButton("Заново");  
 name.setText("");  
 INN.setText("");  
 input.setAutoscrolls(true);  
 input.setEditable(false);  
 output.setAutoscrolls(true);  
 output.setEditable(false);  
 JScrollPane inScroll=new JScrollPane(input, JScrollPane.*VERTICAL\_SCROLLBAR\_AS\_NEEDED*,JScrollPane.*HORIZONTAL\_SCROLLBAR\_NEVER*);  
 JScrollPane outScroll=new JScrollPane(output, JScrollPane.*VERTICAL\_SCROLLBAR\_AS\_NEEDED*,JScrollPane.*HORIZONTAL\_SCROLLBAR\_NEVER*);  
 JPanel card=new JPanel();  
 JPanel card2=new JPanel();  
 JPanel card3=new JPanel();  
 JPanel card4=new JPanel();  
 JPanel card5=new JPanel();  
 JPanel card6=new JPanel();  
 JPanel card7=new JPanel();  
 card.add(inputName);  
 card.add(name);  
 card2.add(inputINN);  
 card2.add(INN);  
 card3.add(button);  
 card4.add(data);  
 card4.add(inScroll);  
 card5.add(getResult);  
 card6.add(result);  
 card6.add(outScroll);  
 card7.add(refresh);  
 add(card);  
 add(card2);  
 add(card3);  
 add(card4);  
 add(card5);  
 add(card6);  
 add(card7);  
 name.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (name.getText().equals("")){  
 JOptionPane.*showMessageDialog*(LabClassUI.this,"Введите имя студента");  
 }  
 }  
 });  
 INN.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (INN.getText().equals("")){  
 JOptionPane.*showMessageDialog*(LabClassUI.this,"Введите ИНН студента");  
 }  
 }  
 });  
 button.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 boolean isReady=true;  
 if (name.getText().equals("")){  
 JOptionPane.*showMessageDialog*(LabClassUI.this,"Введите имя студента");  
 isReady=false;  
 }  
 if (INN.getText().equals("")){  
 JOptionPane.*showMessageDialog*(LabClassUI.this,"Введите ИНН студента");  
 isReady=false;  
  
 }  
 if (isReady) {  
 input.append(name.getText()+" "+INN.getText()+"\n");  
 Student temp=new Student(name.getText(),Integer.*parseInt*(INN.getText()));  
 if (current<lab.list.length) {  
 lab.list[current] = temp;  
 current++;  
 }  
 else{  
 Student[] tempList=new Student[lab.list.length+1];  
 lab.clone(tempList,lab.list);  
 tempList[current]=temp;  
 lab.list=tempList;  
 current++;  
 }  
 name.setText("");  
 INN.setText("");  
 }  
 }  
 });  
 getResult.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 name.setEditable(false);  
 INN.setEditable(false);  
 button.setEnabled(false);  
 if (input.getText().equals("")){  
 JOptionPane.*showMessageDialog*(LabClassUI.this,"Студенты не найдены");  
 }  
 else {  
 LabClassDriver driver=new LabClassDriver();  
 driver.run(lab.list);  
 for (int i = 0; i < lab.list.length; i++) {  
 output.append(lab.list[i].getName() + " " + lab.list[i].getINN() + "\n");  
 }  
 }  
 }  
 });  
 refresh.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 lab.clear();  
 current=0;  
 name.setEditable(true);  
 INN.setEditable(true);  
 button.setEnabled(true);  
 input.setText("");  
 output.setText("");  
 }  
 });  
 }  
 public static void main(String[] args) {  
 LabClassUI l=new LabClassUI();  
 l.setVisible(true);  
 }  
}

Реализация класса SortingStudents:

import java.util.Comparator;  
  
public class SortingStudents implements Comparator<Student> {  
  
 @Override  
 public int compare(Student o1, Student o2) {  
 return o1.getName().hashCode() - o2.getName().hashCode();  
 }  
}

Реализация класса Student:

public class Student {  
 private String name;  
 private int INN;  
 Student(String name, int INN){  
 this.name=name;  
 this.INN=INN;  
 }  
 public void setINN(int INN) {  
 this.INN = INN;  
 }  
 public void setName(String name) {  
 this.name = name;  
 }  
 public String getName(){  
 return name;  
 }  
 public int getINN(){  
 return INN;  
 }  
 public String toString(){  
 return "Name: "+name+" INN: "+INN;  
 }

## **Вывод**

В ходе выполнения работы я научился работать с исключениями в языке java.

С полным кодом программы можно ознакомится по ссылке: https://github.com/gost214/Practika-INBO-02-20-Veselkin/tree/main/pract9