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 ЗАНЯТИЮ №9

Выполнил студент группы ИНБО-02-20 Деревянкин Н.А.

Принял Степанов П.В.

Практическая работа выполнена «___»_____2021 г.

«_____» «___»_____2021 г.

Отметка о выполнении

Москва – 2021 г.

СОДЕРЖАНИЕ

Цель работы

Научиться создавать собственные исключения

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.

Выполнение работы

Приступив к выполнению, я создал все необходимые классы, список для студентов и собственные исключения, которые будут появляться при необходимых условиях.

Код выполненной работы

Здесь в нескольких скриншотах можно увидеть, как выглядит код полученного задания и его вывод.

```java
public class LabClassDriver {
    LabClass labClass;

    public LabClassDriver(LabClass labClass) { this.labClass = labClass; }

    public void input() {
        String name;
        int grade;
        System.out.println("Введите имя и оценку студента('0', чтобы закончить ввод): ");
        Scanner in = new Scanner(System.in);
        name = in.next();
        while (!name.equals("0")) {
            grade = in.nextInt();
            labClass.addStudent(new Student(name, grade));
            System.out.println("Введите имя и оценку студента('0', чтобы закончить ввод): ");
            name = in.next();
        }
    }
}
```

Рисунок 1 – Класс LabClassDriver

```java
public class LabClass {
    private final ArrayList<Student> studentsList;

    public LabClass() { this.studentsList = new ArrayList<>(); }

    public void addStudent(Student student) {
        studentsList.add(student);
        studentsList.sort(Student::compareTo);
    }

    public Student search(String studentsName) throws NoStudentException {
        for (Student student : studentsList) {
            if (student.getName().equals(studentsName)) {
                return student;
            }
        }
        throw new NoStudentException("Student " + studentsName + " was not found");
    }

    public Student remove() { return studentsList.remove( index: 0); }

    public boolean isEmpty() { return studentsList.isEmpty(); }
}
```

Рисунок 2 – Класс LabClass

```java
public record Student(String name, int grade) implements Comparable<Student> {
    @Override
    public String toString() {
        return "Student{" +
                "grade=" + grade +
                ", name='" + name + '\'' +
                '}';
    }

    public String getName() { return name; }

    public int getGrade() { return grade; }

    @Override
    public int compareTo(Student o) { return Integer.compare(this.getGrade(), o.getGrade()); }
}
```

Рисунок 3 – Класс Student

```java
public class LabClassUI {
    LabClass labClass;

    public LabClassUI() {
        labClass = new LabClass();
        LabClassDriver driver = new LabClassDriver(labClass);
        driver.input();
    }

    public void searchForStudent() {
        String name = null;
        Scanner in = new Scanner(System.in);
        System.out.println("Введите имя студента, которого хотите найти:");
        try {
            name = in.next();
            if (name.isBlank())
                throw new EmptyStringException("Empty string!");
        } catch (EmptyStringException e) {
            System.err.println(e.getMessage());
            System.out.println();
            searchForStudent();
        }
        try {
            System.out.println(labClass.search(name).toString());
        } catch (NoStudentException e) {
            System.err.println(e.getMessage());
        }
    }

    public static void main(String[] args) {
        LabClassUI start = new LabClassUI();
        start.searchForStudent();
```

Рисунок 4 – Класс LabClassUI

```java
public class NoStudentException extends Exception {
    public NoStudentException(String errMessage) {
        super(errMessage);
    }
}
```

Рисунок 5 – Исключение NoStudentException

```java
public class EmptyStringException extends Exception {
    public EmptyStringException(String errMessage) {
        super(errMessage);
    }
}
```

Рисунок 6 – Исключение EmptyStringException

Вывод

В результате выполнения данной практической работы я научился создать и работать с собственными исключениям.

GitHub - https://github.com/dronikosha/JavaPractice