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

Отчет по выполнению практической работы 9

**Тема. СОЗДАНИЕ ПОЛЬЗОВАТЕЛЬСКИХ**

**ИСКЛЮЧЕНИЙ**

Дисциплина Программирование на языке Джава

Выполнил

|  |  |
| --- | --- |
| студент | Болотов Михаил |
|  | Фамилия И.О. |
| группа | ИКБО-06-19 |
|  | Номер группы |

Москва 2020

**Содержание**

[​ Теория 2](#__RefHeading___Toc4128_811265792)

[​ Создание проверяемых пользовательских исключений 2](#__RefHeading___Toc42645_811265792)

[​ Создание непроверяемых пользовательских исключений 2](#__RefHeading___Toc42647_811265792)

[​ Задание 3](#__RefHeading___Toc4136_811265792)

[​ Код 3](#__RefHeading___Toc35859_811265792)

[​ LabClassDriver.java 3](#__RefHeading___Toc42649_811265792)

[​ Student.java 4](#__RefHeading___Toc42651_811265792)

[​ EmptyStringException.java 6](#__RefHeading___Toc42653_811265792)

[​ LabClassUI.java 6](#__RefHeading___Toc42655_811265792)

[​ LabClass.java 10](#__RefHeading___Toc42657_811265792)

[​ StudentNotFoundException.java 10](#__RefHeading___Toc42659_811265792)

[​ Скриншот 11](#__RefHeading___Toc3748_811265792)

[​ Заключение 11](#__RefHeading___Toc3750_811265792)

[​ Библиографический список 11](#__RefHeading___Toc3752_811265792)

# Теория

Язык Java предоставляет исчерпывающий набор классов исключений, но иногда при разработке программ вам потребуется создавать новые – свои собственные исключения, которые являются специфическими для потребностей именно вашего приложения. В этой практической работе вы научитесь создавать свои собственные пользовательские классы исключений. Как вы уже знаете, в Java есть два вида исключений- проверяемы и непроверяемые. Для начала рассмотрим создание пользовательских проверяемых исключений.

### Создание проверяемых пользовательских исключений

Проверяемые исключения — это исключения, которые необходимо обрабатывать явно.

Чтобы создать собственное пользовательское исключение, мы будем наследоваться от класса java.lang.Exception. Давайте рассмотрим пример как это реализуется на практике и создадим собственный класс для проверяемого исключения с именем BadFileNameException

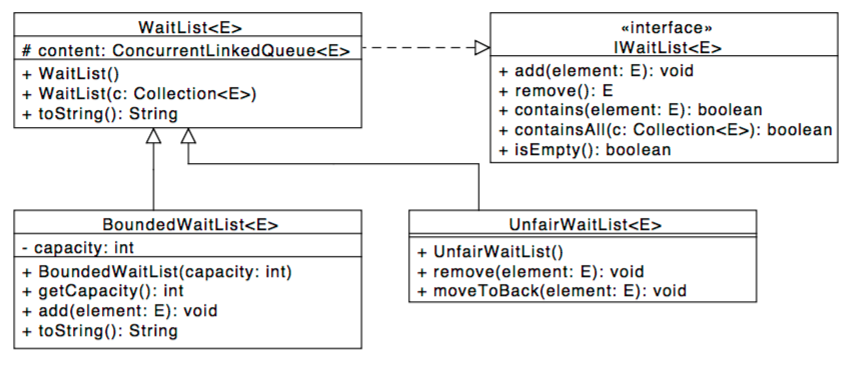
### Создание непроверяемых пользовательских исключений

В том же примере, который мы рассматривали выше предположим, что нам нужно такое пользовательское исключение, в котором обрабатывается ошибка, если файла не содержит расширения.

В этом случае нам как раз понадобится создать пользовательское непроверяемое исключение, похожее на предыдущее, потому что данная ошибка будет обнаружена только во время выполнения участка кода. Чтобы создать собственное непроверяемое исключение, нам нужно наследоваться от класса java.lang.RuntimeException

# Задание

1. Исследуйте UML диаграмму классов на рисунке 1 и понаблюдайте, как она выражает то, что мы говорили выше в словах. Убедитесь, что вы понимаете все аспекты диаграммы.
2. Расширить и модифицировать исходный код WaitList, как необходимо, чтобы полностью реализовать всю схему UML. Включить комментарии Javadoc. Обратите внимание на переключение ролей после реализации каждого интерфейса / класса!
3. Изучение работу метода main(), которая использует ваши новые классы и интерфейс.



# Код

### LabClassDriver.java

package dev.ky3he4ik.pr.pr9;  
  
  
import java.util.Arrays;  
import java.util.Collection;  
  
public class LabClassDriver {  
 public static Student findByName(String name, Collection<Student> students) {  
 for (Student student : students) {  
 if (name.equals(student.getName())) {  
 return student;  
 }  
 }  
 return null;  
 }  
  
 public static void sort(Collection<Student> students) {  
 Student[] objects = students.toArray(new Student[0]);  
 sort(objects, 0, students.size() - 1);  
 students.clear();  
 students.addAll(Arrays.asList(objects));  
 }  
  
 public static void sort(Comparable<Student>[] objects) {  
 sort(objects, 0, objects.length - 1);  
 }  
  
 private static void sort(Comparable<Student>[] objects, final int begin, final int end) {  
 if (begin < end) {  
 int part = partition(objects, begin, end);  
 sort(objects, begin, part);  
 sort(objects, part + 1, end);  
 }  
 }  
  
 private static int partition(Comparable<Student>[] objects, final int begin, final int end) {  
 final Comparable<Student> pivot = objects[(begin + end) / 2];  
 int i = begin, j = end;  
 while (true) {  
 while (objects[i].compareTo((Student) pivot) > 0) {  
 ++i;  
 }  
 while (objects[j].compareTo((Student) pivot) < 0) {  
 --j;  
 }  
  
 if (i >= j) {  
 return j;  
 }  
 Comparable<Student> temp = objects[i];  
 objects[i] = objects[j];  
 objects[j] = temp;  
 ++i;  
 --j;  
 }  
 }  
}

### Student.java

package dev.ky3he4ik.pr.pr9;  
  
import java.util.Arrays;  
  
public class Student implements Comparable<Student> {  
 private String name;  
 private String IDNumber;  
 private byte[] finalMarks;  
  
 public Student(final String name, final String IDNumber, final byte[] finalMarks) {  
 this.finalMarks = finalMarks.clone();  
 this.name = name;  
 this.IDNumber = IDNumber;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getIDNumber() {  
 return IDNumber;  
 }  
  
 public void setIDNumber(String IDNumber) {  
 this.IDNumber = IDNumber;  
 }  
  
 public byte[] getFinalMarks() {  
 return finalMarks;  
 }  
  
 public void setFinalMarks(byte[] finalMarks) {  
 this.finalMarks = finalMarks;  
 }  
  
 @Override  
 public String toString() {  
 return "Student " + name + " with IDNumber " + IDNumber  
 + "; marks: " + Arrays.toString(finalMarks);  
 }  
  
 private static double getAverageMark(final Student student) {  
 double mark = 0;  
 for (final byte item : student.finalMarks) {  
 mark += item;  
 }  
 return mark / student.finalMarks.length;  
 }  
  
 @Override  
 public int compareTo(Student student) {  
 return Double.compare(getAverageMark(this), getAverageMark(student));  
 }  
}

### EmptyStringException.java

package dev.ky3he4ik.pr.pr9;  
  
  
public class EmptyStringException extends Exception {  
 public EmptyStringException() {  
 super("Input string is empty!");  
 }  
}

### LabClassUI.java

package dev.ky3he4ik.pr.pr9;  
  
import javax.swing.\*;  
import javax.swing.table.AbstractTableModel;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.util.Arrays;  
import java.util.Collections;  
  
public class LabClassUI extends JFrame {  
 private final LabTableModel labTableModel;  
 private final LabClass labClass;  
  
 public LabClassUI (LabClass labClass) {  
 super("Lab class");  
 this.labClass = labClass;  
  
 setLayout(new BorderLayout());  
 setSize(500, 500);  
 setMinimumSize(new Dimension(300, 300));  
 setDefaultCloseOperation(EXIT\_ON\_CLOSE);  
  
 labTableModel = new LabTableModel();  
  
 JTable table = new JTable(labTableModel);  
 JScrollPane scrollPane = new JScrollPane(table);  
 add(scrollPane, BorderLayout.CENTER);  
  
 JPanel panel = new JPanel(new GridLayout(2, 2));  
 add(panel, BorderLayout.SOUTH);  
  
 JButton sortButton = new JButton("Sort table");  
 sortButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 labTableModel.sortByGPA();  
 }  
 });  
 panel.add(sortButton);  
  
 JButton shuffleButton = new JButton("Shuffle table");  
 shuffleButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 labTableModel.shuffle();  
 }  
 });  
 panel.add(shuffleButton);  
  
 JTextField textField = new JTextField();  
 textField.setHorizontalAlignment(JTextField.CENTER);  
 panel.add(textField);  
  
 JButton findButton = new JButton("Find student by name");  
 findButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 try {  
 String name = textField.getText();  
 if (name == null || name.isEmpty() || name.isBlank()) {  
 throw new EmptyStringException();  
 }  
 Student student = LabClassDriver.findByName(name, labClass.getStudents());  
 if (student == null) {  
 throw new StudentNotFoundException(name);  
 } else {  
 JOptionPane.showMessageDialog(LabClassUI.this,  
 "Statistics of student:\n" + student);  
 }  
 } catch (EmptyStringException | StudentNotFoundException exception) {  
 JOptionPane.showMessageDialog(LabClassUI.this,  
 exception.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);  
 }  
 }  
 });  
 panel.add(findButton);  
  
 setVisible(true);  
 }  
  
 public void addStudentToTable(Student student) {  
 labTableModel.addRow(student);  
 }  
  
 public void removeStudentFromTable(int studentIndex) {  
 labTableModel.removeRow(studentIndex);  
 }  
  
 private class LabTableModel extends AbstractTableModel {  
 private final String[] columnNames = {  
 "Name", "ID number", "Final marks"  
 };  
  
 @Override  
 public int getRowCount() {  
 return labClass.getStudents().size();  
 }  
  
 @Override  
 public int getColumnCount() {  
 return columnNames.length;  
 }  
  
 @Override  
 public String getColumnName(int columnIndex) {  
 return columnNames[columnIndex];  
 }  
  
 @Override  
 public Class<?> getColumnClass(int columnIndex) {  
 return String.class;  
 }  
  
 @Override  
 public boolean isCellEditable(int rowIndex, int columnIndex) {  
 return false;  
 }  
  
 @Override  
 public Object getValueAt(int rowIndex, int columnIndex) {  
 final Student student = labClass.getStudents().get(rowIndex);  
 switch (columnIndex) {  
 case 0:  
 return student.getName();  
 case 1:  
 return student.getIDNumber();  
 case 2:  
 return Arrays.toString(student.getFinalMarks());  
 default:  
 return "";  
 }  
 }  
  
 public void removeRow(int rowIndex) {  
 labClass.getStudents().remove(rowIndex);  
 fireTableRowsDeleted(rowIndex, rowIndex);  
 }  
  
 public void addRow(Student student) {  
 labClass.getStudents().add(student);  
 final int last = labClass.getStudents().size() - 1;  
 fireTableRowsInserted(last, last);  
 }  
  
 public void sortByGPA() {  
 LabClassDriver.sort(labClass.getStudents());  
 fireTableRowsUpdated(0, labClass.getStudents().size() - 1);  
 }  
  
 public void shuffle() {  
 Collections.shuffle(labClass.getStudents());  
 fireTableRowsUpdated(0, labClass.getStudents().size() - 1);  
 }  
 }  
}

### LabClass.java

package dev.ky3he4ik.pr.pr9;  
  
  
import java.util.Arrays;  
import java.util.LinkedList;  
  
public class LabClass {  
 private final LinkedList<Student> students = new LinkedList<>();  
  
 LabClass() {  
 students.addAll(Arrays.asList(  
 new Student("Abcd", "SQ4695", new byte[] {5, 5, 5, 5}),  
 new Student("Efgh", "PH1357", new byte[] {5, 4, 5, 2}),  
 new Student("Misha", "IT1228", new byte[] {5, 4, 3, 2}),  
 new Student("Dima", "KB9847", new byte[] {5, 4, 4, 1}),  
 new Student("Dmia", "IT1338", new byte[] {2, 2, 2, 5})  
 ));  
 new LabClassUI(this);  
 }  
  
 public LinkedList<Student> getStudents() {  
 return students;  
 }  
  
 public static void main(String[] args) {  
 new LabClass();  
 }  
}

### StudentNotFoundException.java

package dev.ky3he4ik.pr.pr9;  
  
public class StudentNotFoundException extends Exception {  
 public StudentNotFoundException(String name) {  
 super("Student with name `" + name + "` not found!");  
 }  
}

# Скриншот



# Заключение

В данной практической работе я научился создавать собственные исключения.

# Библиографический список

1. Зорина Н.В. Курс лекций по Объектно-ориентированному программированию на Java, МИРЭА, Москва, 2016
2. Программирование на языке Java: работа со строками и массивами. Методические указания. [Электронный ресурс] : Учебно-методические пособия — Электрон. дан. — СПб. : ПГУПС, 2015. — 24 с.
3. Кожомбердиева, Г.И. Программирование на языке Java: создание графического интерфейса пользователя: учеб. пособие. [Электронный ресурс] : Учебные пособия / Г.И. Кожомбердиева, М.И. Гарина. — Электрон. дан. — СПб.: ПГУПС, 2012. — 67 с.
4. Вишневская, Т.И. Технология программирования. Часть 1. [Электронный ресурс] / Т.И. Вишневская, Т.Н. Романова. — Электрон. дан. — М. : МГТУ им. Н.Э. Баумана, 2007. — 59 с.