Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Институт информационных технологий и анализа данных

Центр программной инженерии

ОТЧЕТ

к лабораторной работе №4 по дисциплине:

|  |
| --- |
| «Объектно-ориентированное программирование» |
| Разработка классов с использованием механизмов  наследования, полиморфизма и инкапсуляции  И |

наименование темы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил студент группы: | ИСТб-19-2 |  |  |  | Карбушев И.В. |
|  | шифр группы |  | подпись |  | Фамилия ИО |
| Проверил: | доцент |  |  |  | Маланова Т.В. |
|  | должность |  | подпись |  | Фамилия ИО |

Иркутск. 2020 г.

Оглавление

[Оглавление 2](#_Toc58451550)

[1 Постановка задачи 3](#_Toc58451551)

[2 Проектирование классов 4](#_Toc58451552)

[3 Структура меню 5](#_Toc58451553)

[4 Спецификации 6](#_Toc58451554)

[4.1 Описание класса Attestation 6](#_Toc58451555)

[4.2 Описание класса Exam 7](#_Toc58451556)

[4.3 Описание класса Test 8](#_Toc58451557)

[4.4 Описание интерфейса Event 8](#_Toc58451558)

[4.5 Описание класса Challenge 9](#_Toc58451559)

[4.6 Описание класса Student 9](#_Toc58451560)

[4.7 Описание класса UniversityList 11](#_Toc58451561)

[4.8 Описание класса InteractMenu 12](#_Toc58451562)

[5 Проектирование тестов 13](#_Toc58451563)

[6 Результаты тестирования 14](#_Toc58451564)

[7 Листинг программы 19](#_Toc58451565)

[7 Список использованных источников 40](#_Toc58451566)

1 Постановка задачи

В соответствии с индивидуальным заданием описать иерархию классов, для каждого класса описать поля и соответствующие методы доступа к ним.

В зависимости от задания из этих классов (как минимум один) являются абстрактными классами и служат для выделения общих данных и поведения для других классов. Абстрактный класс должен содержать как минимум один абстрактный метод, реализация которого у его наследников должна различаться.

Помимо этого в общую часть задания входит разработка класса, группирующего объекты описанных в соответствии с заданием классов.

Для выполнения задания необходимо создать некоторое количество объектов, добавить их в группу используя предусмотренные методы класса «группы» и для каждого из них унаследованный метод.

Индивидуальное задание

Резализовать следующего классы:

7. аттестация, зачёт, экзамен, студент;

Разработать класс-группу. Описать иерархию данных классов.

2 Проектирование классов

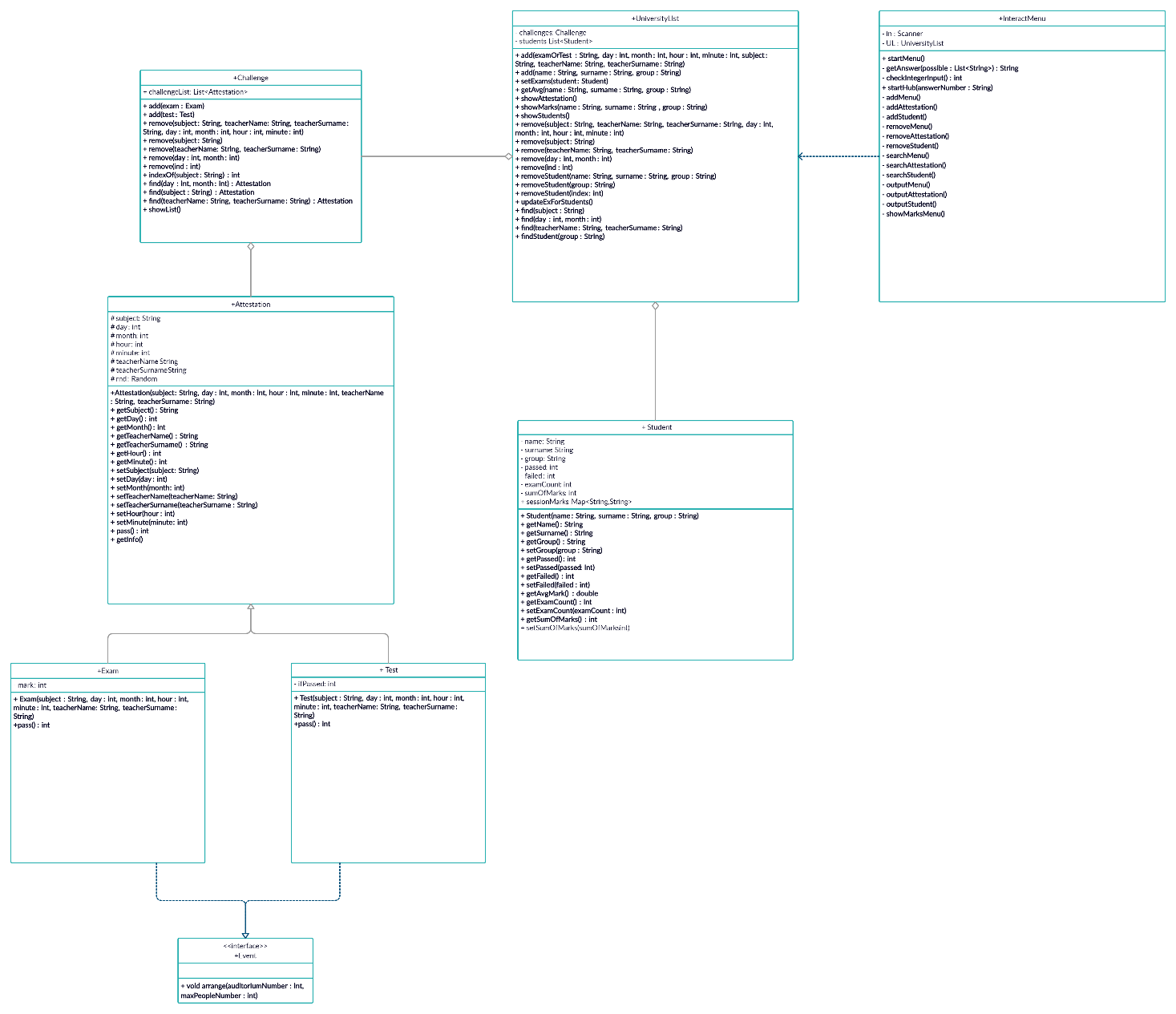
На рисунке 1 представлена UML диаграмма классов

Рисунок 1 – UML диаграмма классов

Class Attestation – родительский класс для аттестаций в университете

Class Exam – экзамен, наследник Attestation

Class Test – зачет, наследник Attestation

Class Student – класс описания студента

Class Challenge – класс-группа для всех объектов класса Attestation

Class UniversityList – класс-группа для объектов класса Challenge и Student

Class InteractMenu – интерактивное меню

3 Структура меню

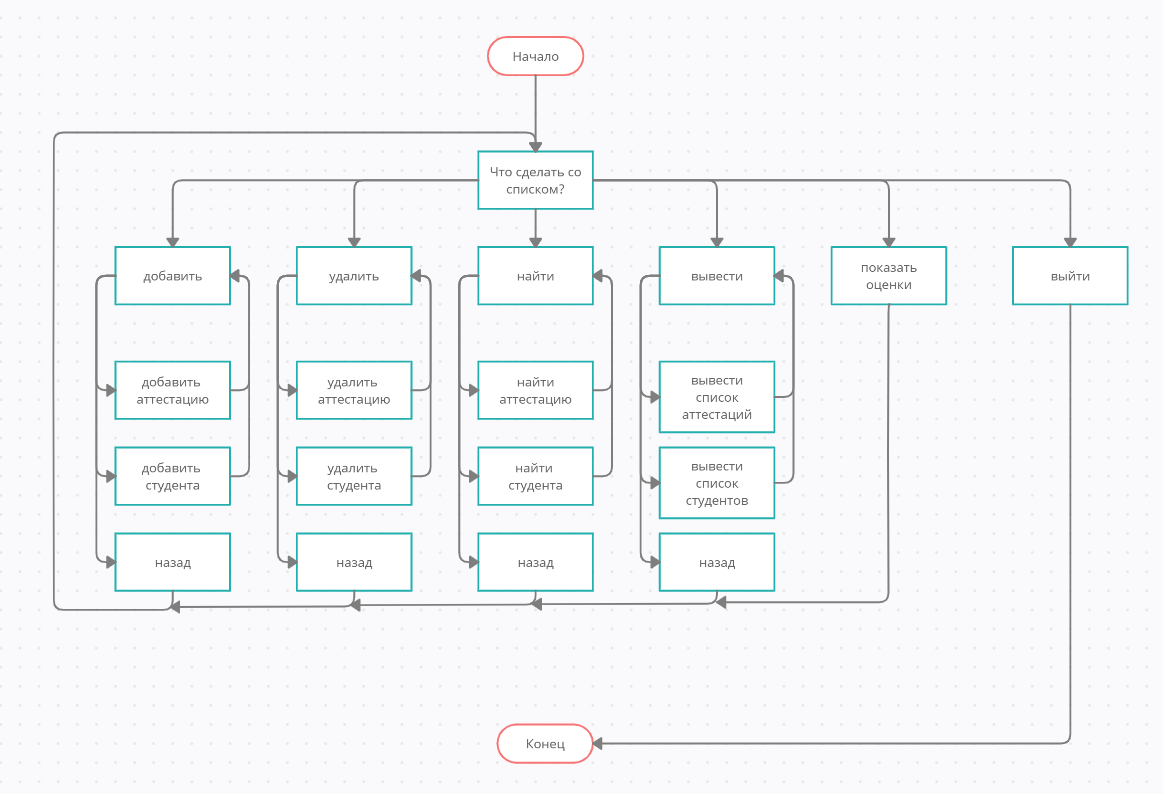
На рисунке 2 представлена структура пользовательского меню

Рисунок 2 – Структура пользовательского меню

Классы, реализующие пользовательское меню:

Class InteractMenu – главный класс пользовательского меню, предоставляющий интерфейс взаимодействия программы с пользователем

4 Спецификации

**4.1 Описание класса Attestation**

Таблица 1 – Поля абстрактного класса Attestation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Назначение | Тип данных | Тип доступа |
| 1 | Subject | Предмет | String | protected |
| 2 | Day | День | Int | protected |
| 3 | Month | месяц | Int | protected |
| 4 | Hour | Час | Int | protected |
| 5 | Minute | Минута | Int | protected |
| 6 | teacherName | Имя преподавателя | String | protected |
| 7 | teacherSurname | Фамилия  Преподавателя | String | protected |
| 8 | rnd | Слуайное число | Random | protected |

Таблица 2 – Конструкторы абстрактного класса Attestation

|  |  |
| --- | --- |
| № | Параметры |
| 1 | String subject |
| Int day |
| Int month |
| Int hour |
| Int minute |
| String teacherName |
| String teacherSurname |

Таблица 3 – Методы абстрактного класса Attestation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Назначение | Тип данных | Тип доступа |
| 1 | getSubject | Получить предмет | String | public |
| 2 | getDay | Получить день | Int | public |
| 3 | getMonth | Получить месяц | Int | public |
| 4 | getHour | Получить час | Int | public |
| 5 | getMinute | Получить минуту | Int | public |
| 6 | getTeacherName | Получить имя преподавателя | String | public |
| 7 | getTeacherSurname | Получить фамилию  преподавателя | String | public |
| 8 | setSubject | Установить предмет | Void | public |
| 9 | setDay | Установить день | Void | public |
| 10 | setMonth | Установить месяц | Void | public |
| 11 | setHour | Установить час | Void | public |
| 12 | setMinute | Установить минуту | Void | public |
| 13 | setTeacherName | Установить имя преподавателя | Void | public |
| 14 | setTeacherSurname | Установить фамилию  преподавателя | void | public |
| 15 | pass | Абстрактный метод «сдать» | Int | public |
| 16 | getInfo | Получить информацию о аттестации | Void | public |

**4.2 Описание класса Exam**

Таблица 4 – Поля класса Exam, наследника класса Attestation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Назначение | Тип данных | Тип доступа |
| 1 | Mark | Оценка (1-5) | int | private |

Таблица 5 – Конструкторы класса Exam

|  |  |
| --- | --- |
| № | Параметры |
| 1 | String subject |
| Int day |
| Int month |
| Int hour |
| Int minute |
| String teacherName |
| String teacherSurname |

Таблица 6 – Методы класса Exam, наследника класса Attestation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Назначение | Тип данных | Тип доступа |
| 1 | pass | Сдать экзамен | int | public |
| 2 | arrange | Получить информацию о проведении экзамена | int | public |

**4.3 Описание класса Test**

Таблица 7 – Поля класса Test, наследника класса Attestation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Назначение | Тип данных | Тип доступа |
| 1 | ifPassed | Зачтено или нет | int | private |

Таблица 8 – Конструкторы класса Test, наследника класса Attestation

|  |  |
| --- | --- |
| № | Параметры |
| 1 | String subject |
| Int day |
| Int month |
| Int hour |
| Int minute |
| String teacherName |
| String teacherSurname |

Таблица 9 – Методы класса Test, наследника класса Attestation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Назначение | Тип данных | Тип доступа |
| 1 | pass | Сдать зачет | String | public |
| 2 | arrange | Получить информацию о проведении зачета | Int | public |

**4.4 Описание интерфейса Event**

Таблица 10 – Методы интерфейса Event

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Назначение | Тип данных | Тип доступа |
| 1 | arrange | Получить информацию о проведении какого-либо события | void |  |

**4.5 Описание класса Challenge**

Таблица 11 – Поля класса-группы Challenge

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Назначение | Тип данных | Тип доступа |
| 1 | challengeList | Список объектов класса Attestation | List<Attestation> | public |

Таблица 12 – Методы класса-группы Challenge

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название функции и ее перегрузок | Назначение | Тип данных | Тип доступа |
| 1 | Add | Добавить аттестацию в список | void | public |
| 2 | remove | Удалить аттестацию из списка | void | public |
| 3 | find | Найти аттестацию | Attestation | public |
| 4 | showList | Вывести список аттестаций | void | public |
| 5 | indexOf | Найти индекс аттестации в списке | int | public |

**4.6 Описание класса Student**

Таблица 13 – Поля класса Student

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Назначение | Тип данных | Тип доступа |
| 1 | Name | Имя студента | String | private |
| 2 | Surname | Фамилия студента | String | private |
| 3 | Group | Группа студента | String | private |
| 4 | Passed | Сдано аттестаций | Int | private |
| 5 | Failed | Завалено аттестаций | Int | private |
| 6 | examCount | Сколько всего экзаменов | Int | private |
| 7 | sumOfMarks | Сумма оценок | int | private |
| 8 | sessionMarks | Оценки по аттестациям студента | Map<String, String> | public |

Таблица 14 – Конструкторы класса Student

|  |  |
| --- | --- |
| № | Параметры |
| 1 | String name |
| String surname |
| String group |

Таблица 15 – Методы класса Student

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Назначение | Тип данных | Тип доступа |
| 1 | getName | Получить имя студента | String | public |
| 2 | getSurname | Получить фамилию студента | String | public |
| 3 | getGroup | Получить группу студента | String | public |
| 4 | setGroup | Установить группу студента | Void | public |
| 5 | getPassed | Получить количество сданных аттестаций | Int | public |
| 6 | setPassed | Установить количество сданных аттестаций | Void | public |
| 7 | getFailed | Получить количество заваленных аттестаций | Int | public |
| 8 | setFailed | Установить количество заваленных аттестаций | Void | public |
| 9 | getAvgMark | Получить средний балл студента | Double | public |
| 10 | getExamCount | Получить количество экзаменов студента | Int | public |
| 11 | setExamCount | Установить количество экзаменов студента | Void | public |
| 12 | getSumOfMarks | Получить сумму всех оценок студента | Int | public |
| 13 | setSumOfMarks | Установить сумму всех оценок студента | Void | public |

**4.7 Описание класса UniversityList**

Таблица 16 – Поля UniversityList

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Назначение | Тип данных | Тип доступа |
| 1 | challenges | Список аттестаций | Challenge | Private |
| 2 | students | Список студентов | Student | Private |

Таблица 17 – Методы UniversityList

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название методов и их перегрузок | Назначение | Тип данных | Тип доступа |
| 1 | add | Добавить в список студента или аттестацию | void | public |
| 2 | setExams | Установить аттестации студентам | void | private |
| 3 | getAvg | Получить средний балл студента | double | public |
| 4 | showAttestation | Показать список аттестаций | Void | public |
| 5 | showMarks | Показать оценки студента | Void | public |
| 6 | showStudents | Показать список студентов | Void | public |
| 7 | remove | Удалить аттестацию из списка | Void | public |
| 8 | removeStudent | Удалить студента из списка | Void | public |
| 9 | updateExForStudents | Обновить аттестации для студентов | Void | private |
| 10 | Find | Найти аттестацию | Void | public |
| 11 | findStudent | Найти студента | Void | public |

**4.8 Описание класса InteractMenu**

Таблица 18 – Поля абстрактного класса Attestation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Назначение | Тип данных | Тип доступа |
| 1 | In | Ввод с консоли | Scanner | private |
| 2 | UL | Список аттестаций и студентов | UniversityList | private |

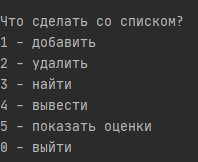
Таблица 19 – Методы абстрактного класса Attestation

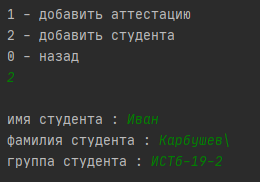
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Назначение | Тип данных | Тип доступа |
| 1 | startMenu | Вывести главное меню | Void | public |
| 2 | getAnswer | Получить ответ с консоли | String | private |
| 3 | checkIntegerInput | Проверить число на корректность | Int | private |
| 4 | startHub | Связывает элементы меню с главным меню | Void | private |
| 5 | addMenu | Меню добавления | Void | private |
| 6 | addAttestation | Добавить аттестацию | Void | private |
| 7 | addStudent | Добавить студента | void | private |
| 8 | removeMenu | Меню удаления | Void | private |
| 9 | removeAttestation | Удалить аттестацию | Void | private |
| 10 | removeStudent | Удалить студента | Void | private |
| 11 | searchMenu | Меню поиска | Void | private |
| 12 | searchAttestation | Найти аттестацию | Void | private |
| 13 | searchStudent | Найти студента | Void | private |
| 14 | outputMenu | Меню вывода | void | private |
| 15 | outputAttestation | Вывести аттестации | Void | private |
| 16 | outputStudent | Вывести студентов | Void | private |
| 17 | showMarksMenu | Показать оценки студента | Void | private |

5 Проектирование тестов

Таблица 20 – Тестирование программы

|  |  |  |
| --- | --- | --- |
| **№** | **Действие** | **Результат действия** |
| 1 | Первоначальный запуск программы | Вывод меню старта |
| 2 | Создание нового студента | Создан студент в списке |
| 3 | Создание экзамена | Создан экзамен в списке |
| 4 | Создание зачета | Создан зачет в списке |
| 5 | Поиск заданной аттестации | Вывод на экран аттестации |
| 6 | Поиск заданного студента | Вывод на экран студента |
| 7 | Вывод списка аттестаций | Вывод на экран списка аттестаций |
| 8 | Вывод списка студентов | Вывод на экран списка студентов |
| 9 | Показ оценок студента | Вывод на экран оценок студента |
| 10 | Удаление аттестации | Удалена аттестация |
| 11 | Удалении студента | Удален студент |
| 12 | Завершение работы программы | Программа закончилась |

6 Результаты тестирования

Рисунок 3 – Тест 1

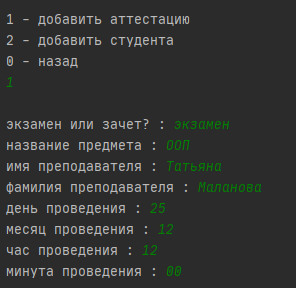
Рисунок 4 – Тест 2

Рисунок 5 – Тест 3

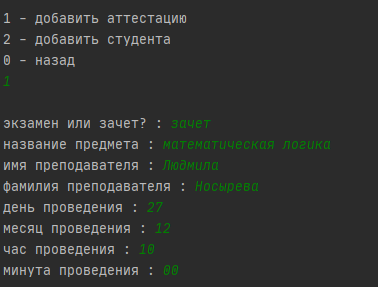
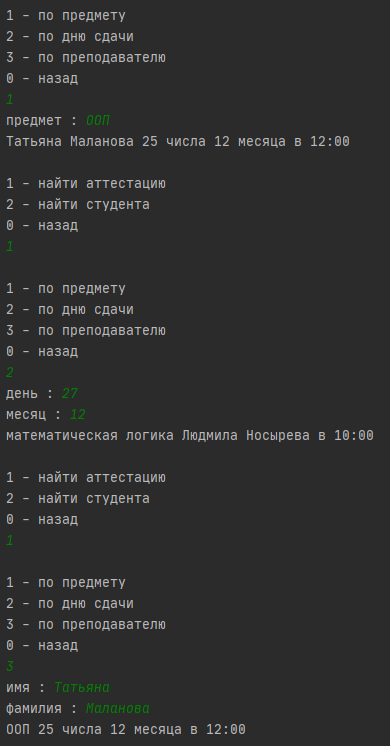
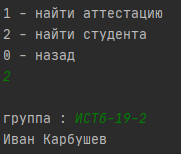
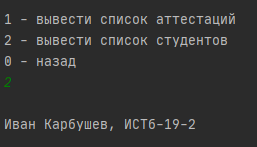
Рисунок 6 – Тест 4

Рисунок 7 – Тест 5

Рисунок 8 – Тест 6

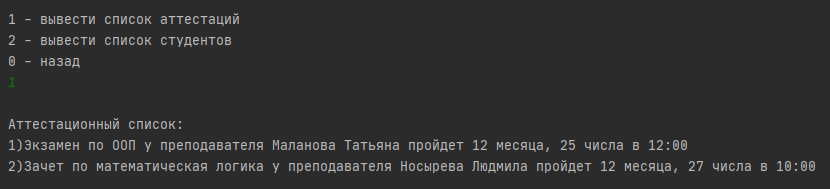
Рисунок 9 – Тест 7

Рисунок 10 – Тест 8

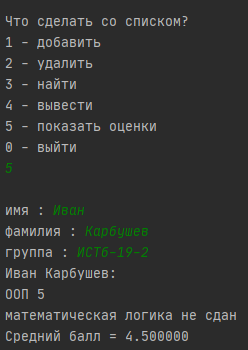


Рисунок 11 – Тест 9

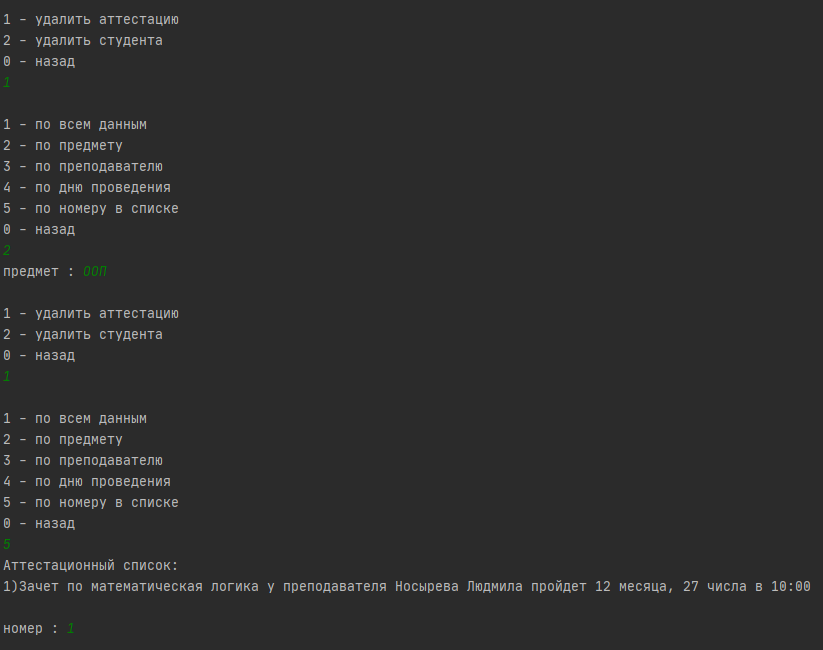


Рисунок 12 – Тест 10

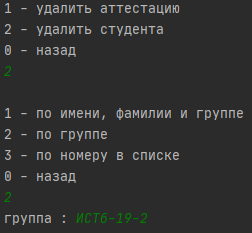


Рисунок 13 – Тест 11

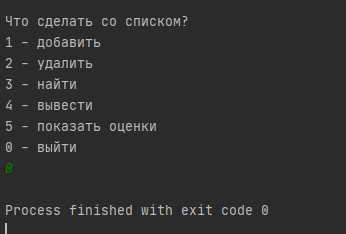


Рисунок 14 – Тест 12

7 Листинг программы

Class Main

package com.company;  
  
public class Main {  
  
 public static void main(String[] args) {  
 InteractMenu.*startMenu*();  
 }  
}

class Attestation

package com.company;  
  
import java.util.Random;  
  
public abstract class Attestation {  
 protected String subject;  
 protected int day;  
 protected int month;  
 protected int hour;  
 protected int minute;  
 protected String teacherName;  
 protected String teacherSurname;  
 protected static Random *rnd* = new Random();  
  
 public Attestation(String subject, int day, int month, int hour, int minute,  
 String teacherName, String teacherSurname) {  
 this.subject = subject;  
 this.day = day;  
 this.month = month;  
 this.hour = hour;  
 this.minute = minute;  
 this.teacherName = teacherName;  
 this.teacherSurname = teacherSurname;  
 }  
  
 public String getSubject() {  
 return subject;  
 }  
  
 public int getDay() {  
 return day;  
 }  
  
 public int getMonth() {  
 return month;  
 }  
  
 public String getTeacherName() {  
 return teacherName;  
 }  
  
 public String getTeacherSurname() {  
 return teacherSurname;  
 }  
  
 public int getHour() {  
 return hour;  
 }  
  
 public int getMinute() {  
 return minute;  
 }  
  
 public void setSubject(String subject) {  
 this.subject = subject;  
 }  
  
 public void setDay(int day) {  
 this.day = day;  
 }  
  
 public void setMonth(int month) {  
 this.month = month;  
 }  
  
 public void setTeacherFirstName(String teacherName) {  
 this.teacherName = teacherName;  
 }  
  
 public void setTeacherSecondName(String teacherSurname) {  
 this.teacherSurname = teacherSurname;  
 }  
  
 public void setHour(int hour) {  
 this.hour = hour;  
 }  
  
 public void setMinute(int minute) {  
 this.minute = minute;  
 }  
  
 public abstract int pass();  
  
 public void getInfo() {  
 System.*out*.printf("Аттестация по предмету - %s%n", subject);  
 System.*out*.printf("Начало в %d%n:%d%n %d%n месяца, %d%n числа", hour, minute, month, day);  
 System.*out*.printf("Проведет аттестацию %s%n %s%n", teacherSurname, teacherName);  
 }  
}

**interface Event**

package com.company;  
  
public interface Event {  
 void arrange(int auditoriumNumber, int maxPeopleNumber);  
}

**Class Exam**

package com.company;  
  
public class Exam extends Attestation implements Event {  
 private int mark;  
  
 public Exam(String subject, int day, int month, int hour, int minute, String teacherFirstName, String teacherSecondName) {  
 super(subject, day, month, hour, minute, teacherFirstName, teacherSecondName);  
 }  
  
 @Override  
 public int pass() {  
 mark = *rnd*.nextInt(5) + 1;  
 return mark;  
 }  
  
 @Override  
 public void arrange(int auditoriumNumber, int maxPeopleNumber) {  
 System.*out*.printf("Экзамен пройдет в аудитории %d вместительностью %d человек%n", auditoriumNumber, maxPeopleNumber);  
 }  
}

**class Test**

package com.company;  
  
public class Test extends Attestation implements Event {  
 private int ifPassed;  
  
 public Test(String subject, int day, int month, int hour, int minute, String teacherFirstName, String teacherSecondName) {  
 super(subject, day, month, hour, minute, teacherFirstName, teacherSecondName);  
 }  
  
 @Override  
 public int pass() {  
 ifPassed = *rnd*.nextInt(2);  
 return ifPassed;  
 }  
  
 @Override  
 public void arrange(int auditoriumNumber, int maxPeopleNumber) {  
 System.*out*.printf("Зачет пройдет в аудитории %d вместительностью %d человек%n", auditoriumNumber, maxPeopleNumber);  
 }  
}

**class Challenge**

package com.company;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class Challenge {  
 public final List<Attestation> challengeList = new ArrayList<>();  
  
 public void add(Exam exam) {  
 challengeList.add(exam);  
 }  
  
 public void add(Test test) {  
 challengeList.add(test);  
 }  
  
 public void remove(String subject, String teacherName, String teacherSurname, int day, int month, int hour, int minute) {  
 challengeList.removeIf(item -> item.getSubject().equals(subject) && item.getTeacherName().equals(teacherName)  
 && item.getTeacherSurname().equals(teacherSurname) && item.getDay() == day && item.getMonth() == month  
 && item.getHour() == hour && item.getMinute() == minute);  
 }  
  
 public void remove(String subject) {  
 challengeList.removeIf(item -> item.getSubject().equals(subject));  
 }  
  
 public void remove(String teacherName, String teacherSurname) {  
 challengeList.removeIf(item -> item.getTeacherName().equals(teacherName) && item.getTeacherSurname().equals(teacherSurname));  
 }  
  
 public void remove(int day, int month) {  
 challengeList.removeIf(item -> item.getDay() == day && item.getMonth() == month);  
 }  
  
 public void remove(int ind) {  
 challengeList.remove(ind);  
 }  
  
 /\*public int indexOf(String subject, String teacherName, String teacherSurname, int day, int month, int hour, int minute) {  
 for (var item : challengeList) {  
 if (item.getSubject().equals(subject) && item.getTeacherName().equals(teacherName)  
 && item.getTeacherSurname().equals(teacherSurname) && item.getDay() == day && item.getMonth() == month  
 && item.getHour() == hour && item.getMinute() == minute)  
 return challengeList.indexOf(item);  
 }  
 return -1;  
 }\*/  
  
 public int indexOf(String subject) {  
 for (var item : challengeList) {  
 if (item.getSubject().equals(subject))  
 return challengeList.indexOf(item);  
 }  
 return -1;  
 }  
  
 public Attestation find(int day, int month) {  
 for (var item : challengeList) {  
 if (item.getDay() == day && item.getMonth() == month) {  
 String addZero = "";  
 if (item.getMinute() < 10) addZero = "0";  
 System.*out*.printf("%s %s %s в %d:%s%d%n",  
 item.getSubject(), item.getTeacherName(), item.getTeacherSurname(), item.getHour(), addZero, item.getMinute());  
 }  
 }  
 return null;  
 }  
  
 public Attestation find(String subject) {  
 for (var item : challengeList) {  
 if (item.getSubject().equals(subject)) {  
 String addZero = "";  
 if (item.getMinute() < 10) addZero = "0";  
 System.*out*.printf("%s %s %d числа %d месяца в %d:%s%d%n",  
 item.getTeacherName(), item.getTeacherSurname(), item.getDay(), item.getMonth(), item.getHour(), addZero, item.getMinute());  
 }  
 }  
 return null;  
 }  
  
 public Attestation find(String teacherName, String teacherSurname) {  
 for (var item : challengeList) {  
 if (item.getTeacherName().equals(teacherName) && item.getTeacherSurname().equals(teacherSurname)) {  
 String addZero = "";  
 if (item.getMinute() < 10) addZero = "0";  
 System.*out*.printf("%s %d числа %d месяца в %d:%s%d%n",  
 item.getSubject(), item.getDay(), item.getMonth(), item.getHour(), addZero, item.getMinute());  
 }  
 }  
 return null;  
 }  
  
 public void showList() {  
 int count = 1;  
 System.*out*.println("Аттестационный список:");  
 for (var item : challengeList) {  
 String testOrExam;  
 if (item instanceof Exam) {  
 testOrExam = "Экзамен";  
 } else {  
 testOrExam = "Зачет";  
 }  
 String addZero = "";  
 if (item.getMinute() < 10) addZero = "0";  
 System.*out*.printf("%d)%s по %s у преподавателя %s %s пройдет %d месяца, %d числа в %d:%s%d%n",  
 count++, testOrExam, item.getSubject(), item.getTeacherSurname(), item.getTeacherName(), item.getMonth(), item.getDay(), item.getHour(), addZero, item.getMinute());  
  
 }  
 }  
}

**class UniversityList**

package com.company;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Random;  
  
public class UniversityList {  
 private final static Challenge *challenges* = new Challenge();  
 private final static List<Student> *students* = new ArrayList<>();  
  
 public void add(String examOrTest, String subject, int day, int month, int hour, int minute,  
 String teacherName, String teacherSurname) {  
 //потому что если нет студентов, то некому сдавать экзамены и зачеты  
 if (*students*.isEmpty()) {  
 System.*out*.println("Нет студентов");  
 return;  
 }  
 if (examOrTest.toLowerCase().equals("экзамен")) *challenges*.add(new Exam(subject, day, month, hour, minute, teacherName, teacherSurname));  
 else if (examOrTest.toLowerCase().equals("зачет")) *challenges*.add(new Test(subject, day, month, hour, minute, teacherName, teacherSurname));  
 else return;  
 for (var student : *students*) {  
 setExams(student);  
 }  
 }  
  
 public void add(String name, String surname, String group) {  
 Student student = new Student(name, surname, group);  
 *students*.add(student);  
 setExams(student);  
 }  
  
 /\*public void add() {  
 Random rnd = new Random();  
 int num = rnd.nextInt(2);  
 if (num == 1) challenges.challengeList.add(new Exam());  
 else challenges.challengeList.add(new Test());  
 }\*/  
  
 private void setExams(Student student) {  
 for (var exam : *challenges*.challengeList) {  
 if (student.sessionMarks.get(exam.getSubject()) != null) continue;  
 int mark = exam.pass();  
 String examMark;  
 if (exam instanceof Exam) {  
 if (mark < 3) { examMark = "не сдан"; student.setFailed(student.getFailed() + 1); }  
 else { examMark = String.*valueOf*(mark); student.setPassed(student.getPassed() + 1); }  
 student.setSumOfMarks(student.getSumOfMarks() + mark);  
 student.setExamCount(student.getExamCount() + 1);  
 }  
 else {  
 if (mark == 1) { examMark = "сдан"; student.setPassed(student.getPassed() + 1); }  
 else { examMark = "не сдан"; student.setFailed(student.getFailed() + 1); }  
 }  
 student.sessionMarks.put(exam.getSubject(), examMark);  
 }  
 }  
  
 public double getAvg(String name, String surname) {  
 for (var student : *students*) {  
 if (student.getName().equals(name) && student.getSurname().equals(surname)) return student.getAvgMark();  
 }  
 return 0;  
 }  
  
 public void showAttestation() {  
 *challenges*.showList();  
 }  
  
 public void showMarks(String name, String surname, String group) {  
 System.*out*.printf("%s %s:%n", name, surname);  
 for (var student : *students*) {  
 if (student.getName().equals(name) && student.getSurname().equals(surname) && student.getGroup().equals(group)) {  
 List<String> keys = new ArrayList<>(student.sessionMarks.keySet());  
 for (String key : keys) {  
 String value = student.sessionMarks.get(key);  
 System.*out*.println(key + " " + value);  
 }  
 System.*out*.printf("Средний балл = %f%n", student.getAvgMark());  
 return;  
 }  
 }  
 System.*out*.println("Нет такого студента");  
 }  
  
 public void showStudents() {  
 for (var student : *students*) {  
 System.*out*.printf("%s %s, %s%n", student.getName(), student.getSurname(), student.getGroup());  
 }  
 }  
  
 public void remove(String subject, String teacherName, String teacherSurname, int day, int month, int hour, int minute) {  
 *challenges*.remove(subject, teacherName, teacherSurname, day, month, hour, minute);  
  
 updateExForStudents();  
 }  
  
 public void remove(String subject) {  
 boolean check = false;  
 for (var item : *challenges*.challengeList) {  
 if (item.getSubject().equals(subject)) {  
 check = true;  
 break;  
 }  
 }  
 if (!check) {  
 System.*out*.printf("Нет аттестации по предмету %s%n", subject);  
 return;  
 }  
 *challenges*.remove(subject);  
 updateExForStudents();  
 }  
  
 public void remove(String teacherName, String teacherSurname) {  
 boolean check = false;  
 for (var item : *challenges*.challengeList) {  
 if (item.getTeacherName().equals(teacherName) && item.getTeacherSurname().equals(teacherSurname)) {  
 check = true;  
 break;  
 }  
 }  
 if (!check) {  
 System.*out*.printf("У преподавателя %s %s нет никаких аттестаций%n", teacherName, teacherSurname);  
 return;  
 }  
 *challenges*.remove(teacherName, teacherSurname);  
 updateExForStudents();  
 }  
  
 public void remove(int day, int month) {  
 boolean check = false;  
 for (var item : *challenges*.challengeList) {  
 if (item.getDay() == day && item.getMonth() == month) {  
 check = true;  
 break;  
 }  
 }  
 if (!check) {  
 System.*out*.printf("%d числа %d месяца нет никаких аттестаций%n", day, month);  
 return;  
 }  
 *challenges*.remove(day, month);  
 updateExForStudents();  
 }  
  
 public void remove(int ind) {  
 if (*challenges*.challengeList.size() <= ind-1) {  
 System.*out*.printf("Нет такой аттестации с индексом %d%n", ind);  
 return;  
 }  
 *challenges*.remove(ind-1);  
 updateExForStudents();  
 }  
  
 public void removeStudent(String name, String surname, String group) {  
 boolean check = false;  
 for (var student : *students*) {  
 if (student.getName().equals(name) && student.getSurname().equals(surname) && student.getGroup().equals(group)) {  
 check = true;  
 break;  
 }  
 }  
 if (check)  
 *students*.removeIf(student -> student.getName().equals(name) && student.getSurname().equals(surname) && student.getGroup().equals(group));  
 else System.*out*.printf("Нет студента %s %s в группе %s%n", name, surname, group);  
 }  
  
 public void removeStudent(String group) {  
 boolean check = false;  
 for (var student : *students*) {  
 if (student.getGroup().equals(group)) {  
 check = true;  
 break;  
 }  
 }  
 if(check)  
 *students*.removeIf(student -> student.getGroup().equals(group));  
 else System.*out*.printf("Нет такой группы - %s%n", group);  
 }  
  
 public void removeStudent(int index) {  
 if (*students*.size() <= index-1) {  
 System.*out*.printf("Нет такого студента с индексом %d%n", index);  
 return;  
 }  
 *students*.remove(index);  
 }  
  
 private void updateExForStudents() {  
 for (var student : *students*) {  
 List<String> keys = new ArrayList<>(student.sessionMarks.keySet());  
 for (String key : keys) {  
 if (*challenges*.indexOf(key) == -1) {  
 student.sessionMarks.remove(key);  
 }  
 }  
 }  
 }  
  
 /\*  
 public int indexOf(String subject, String teacherName, String teacherSurname, int day, int month, int hour, int minute) {  
 return challenges.indexOf(subject, teacherName, teacherSurname, day, month, hour, minute);  
 }  
 \*/  
  
 public void find(String subject) {  
 *challenges*.find(subject);  
 }  
  
 public void find(int day, int month) {  
 *challenges*.find(day, month);  
 }  
  
 public void find(String teacherName, String teacherSurname) {  
 *challenges*.find(teacherName, teacherSurname);  
 }  
  
 public void findStudent(String group) {  
 for (var student : *students*) {  
 if (student.getGroup().equals(group)) System.*out*.printf("%s %s%n", student.getName(), student.getSurname());  
 }  
 }  
}

**class Student**

package com.company;  
  
import java.util.HashMap;  
import java.util.Map;  
  
public class Student {  
 private String name;  
 private String surname;  
 private String group;  
 private int passed = 0;  
 private int failed = 0;  
 private int examCount = 0;  
 private int sumOfMarks = 0;  
 public final Map<String, String> sessionMarks = new HashMap();  
  
 public Student(String name, String surname, String group) {  
 this.name = name;  
 this.surname = surname;  
 this.group = group;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public String getSurname() {  
 return surname;  
 }  
  
 public String getGroup() {  
 return group;  
 }  
  
 public void setGroup(String group) {  
 this.group = group;  
 }  
  
 public int getPassed() {  
 return passed;  
 }  
  
 public void setPassed(int passed) {  
 this.passed = passed;  
 }  
  
 public int getFailed() {  
 return failed;  
 }  
  
 public void setFailed(int failed) {  
 this.failed = failed;  
 }  
  
 public double getAvgMark() {  
 if (examCount == 0) return -1;  
 return (double)sumOfMarks / examCount;  
 }  
  
 public int getExamCount() {  
 return examCount;  
 }  
  
 public void setExamCount(int examCount) {  
 this.examCount = examCount;  
 }  
  
 public int getSumOfMarks() {  
 return sumOfMarks;  
 }  
  
 public void setSumOfMarks(int sumOfMarks) {  
 this.sumOfMarks = sumOfMarks;  
 }  
}

**class InteractMenu**

package com.company;  
  
import java.util.Arrays;  
import java.util.List;  
import java.util.Scanner;  
  
public class InteractMenu {  
 private final static Scanner *in* = new Scanner(System.*in*);  
 private final static UniversityList *UL* = new UniversityList();  
  
 public static void startMenu() {  
 System.*out*.println();  
 System.*out*.println("Что сделать со списком?");  
 System.*out*.println("1 - добавить");  
 System.*out*.println("2 - удалить");  
 System.*out*.println("3 - найти");  
 System.*out*.println("4 - вывести");  
 System.*out*.println("5 - показать оценки");  
 System.*out*.println("0 - выйти");  
  
 String answer = *getAnswer*(Arrays.*asList*("1", "2", "3", "4", "5", "0"));  
 *startHub*(answer);  
 }  
  
 private static String getAnswer(List<String> possible) {  
 String answer = *in*.nextLine();  
 while (!possible.contains(answer)) {  
 System.*out*.println("нет такого варианта ответа, повторите ввод");  
 answer = *in*.nextLine();  
 }  
 return answer;  
 }  
  
 private static int checkIntegerInput() {  
 //проверка стринга на инт  
 String input = *in*.nextLine();  
 int result = -1;  
 try {  
 result = Integer.*parseInt*(input);  
 }  
 catch (NumberFormatException ex) {  
 System.*out*.print("Вы ввели не число, повторите ввод : ");  
 *checkIntegerInput*();  
 }  
 return result;  
 }  
  
 private static void startHub(String answerNumber) {  
 switch (answerNumber) {  
 case "1":  
 *addMenu*();  
 break;  
 case "2":  
 *removeMenu*();  
 break;  
 case "3":  
 *searchMenu*();  
 break;  
 case "4":  
 *outputMenu*();  
 break;  
 case "5":  
 *showMarksMenu*();  
 break;  
 case "0":  
 System.*exit*(0);  
 }  
 }  
  
 private static void addMenu() {  
 System.*out*.println();  
 System.*out*.println("1 - добавить аттестацию");  
 System.*out*.println("2 - добавить студента");  
 System.*out*.println("0 - назад");  
  
 String answer = *getAnswer*(Arrays.*asList*("1", "2", "0"));  
 switch (answer) {  
 case "1":  
 *addAttestation*();  
 break;  
 case "2":  
 *addStudent*();  
 break;  
 }  
 *startMenu*();  
 }  
  
 private static void addAttestation() {  
 System.*out*.println();  
 String examOrTest, subject, teacherName, teacherSurname;  
 int day, month, hour, minute;  
 System.*out*.print("экзамен или зачет? : "); examOrTest = *in*.nextLine();  
 System.*out*.print("название предмета : "); subject = *in*.nextLine();  
 System.*out*.print("имя преподавателя : "); teacherName = *in*.nextLine();  
 System.*out*.print("фамилия преподавателя : "); teacherSurname = *in*.nextLine();  
 System.*out*.print("день проведения : "); day = *checkIntegerInput*();  
 System.*out*.print("месяц проведения : "); month = *checkIntegerInput*();  
 System.*out*.print("час проведения : "); hour = *checkIntegerInput*();  
 System.*out*.print("минута проведения : "); minute = *checkIntegerInput*();  
 *UL*.add(examOrTest, subject, day, month, hour, minute, teacherName, teacherSurname);  
 *addMenu*();  
 }  
  
 private static void addStudent() {  
 System.*out*.println();  
 String name, surname, group;  
 System.*out*.print("имя студента : "); name = *in*.nextLine();  
 System.*out*.print("фамилия студента : "); surname = *in*.nextLine();  
 System.*out*.print("группа студента : "); group = *in*.nextLine();  
 *UL*.add(name, surname, group);  
 *addMenu*();  
 }  
  
 private static void removeMenu() {  
 System.*out*.println();  
 System.*out*.println("1 - удалить аттестацию");  
 System.*out*.println("2 - удалить студента");  
 System.*out*.println("0 - назад");  
  
 String answer = *getAnswer*(Arrays.*asList*("1", "2", "0"));  
 switch (answer) {  
 case "1":  
 *removeAttestation*();  
 break;  
 case "2":  
 *removeStudent*();  
 break;  
 }  
 *startMenu*();  
 }  
  
 private static void removeAttestation() {  
 System.*out*.println();  
 System.*out*.println("1 - по всем данным");  
 System.*out*.println("2 - по предмету");  
 System.*out*.println("3 - по преподавателю");  
 System.*out*.println("4 - по дню проведения");  
 System.*out*.println("5 - по номеру в списке");  
 System.*out*.println("0 - назад");  
 String examOrTest, subject, teacherName, teacherSurname;  
 int day, month, hour, minute;  
  
 String answer = *getAnswer*(Arrays.*asList*("1", "2", "3", "4", "5", "0"));  
 switch (answer) {  
 case "1":  
 System.*out*.print("название предмета : "); subject = *in*.nextLine();  
 System.*out*.print("имя преподавателя : "); teacherName = *in*.nextLine();  
 System.*out*.print("фамилия преподавателя : "); teacherSurname = *in*.nextLine();  
 System.*out*.print("день проведения : "); day = *checkIntegerInput*();  
 System.*out*.print("месяц проведения : "); month = *checkIntegerInput*();  
 System.*out*.print("час проведения : "); hour = *checkIntegerInput*();  
 System.*out*.print("минута проведения : "); minute = *checkIntegerInput*();  
 *UL*.remove(subject, teacherName, teacherSurname, day, month, hour, minute);  
 break;  
 case "2":  
 System.*out*.print("предмет : "); subject = *in*.nextLine();  
 *UL*.remove(subject);  
 break;  
 case "3":  
 System.*out*.print("имя : "); teacherName = *in*.nextLine();  
 System.*out*.print("фамилия : "); teacherSurname = *in*.nextLine();  
 *UL*.remove(teacherName, teacherSurname);  
 break;  
 case "4":  
 System.*out*.print("день : "); day = *checkIntegerInput*();  
 System.*out*.print("месяц : "); month = *checkIntegerInput*();  
 *UL*.remove(day, month);  
 break;  
 case "5":  
 *UL*.showAttestation();  
 System.*out*.println();  
 System.*out*.print("номер : "); int index = *checkIntegerInput*();  
 *UL*.remove(index);  
 }  
 *removeMenu*();  
 }  
  
 private static void removeStudent() {  
 System.*out*.println();  
 System.*out*.println("1 - по имени, фамилии и группе");  
 System.*out*.println("2 - по группе");  
 System.*out*.println("3 - по номеру в списке");  
 System.*out*.println("0 - назад");  
 String name, surname, group;  
  
 String answer = *getAnswer*(Arrays.*asList*("1", "2", "3", "0"));  
 switch (answer) {  
 case "1":  
 System.*out*.print("имя : "); name = *in*.nextLine();  
 System.*out*.print("фамилия : "); surname = *in*.nextLine();  
 System.*out*.print("группа : "); group = *in*.nextLine();  
 *UL*.removeStudent(name, surname, group);  
 break;  
 case "2":  
 System.*out*.print("группа : "); group = *in*.nextLine();  
 *UL*.removeStudent(group);  
 break;  
 case "3":  
 *UL*.showStudents();  
 System.*out*.println();  
 System.*out*.println("индекс : "); int index = *checkIntegerInput*();  
 *UL*.removeStudent(index);  
 break;  
 }  
 *removeMenu*();  
 }  
  
 private static void searchMenu() {  
 System.*out*.println();  
 System.*out*.println("1 - найти аттестацию");  
 System.*out*.println("2 - найти студента");  
 System.*out*.println("0 - назад");  
  
 String answer = *getAnswer*(Arrays.*asList*("1", "2", "0"));  
 switch (answer) {  
 case "1":  
 *searchAttestation*();  
 break;  
 case "2":  
 *searchStudent*();  
 break;  
 }  
 *startMenu*();  
 }  
  
 private static void searchAttestation() {  
 System.*out*.println();  
 System.*out*.println("1 - по предмету");  
 System.*out*.println("2 - по дню сдачи");  
 System.*out*.println("3 - по преподавателю");  
 System.*out*.println("0 - назад");  
  
 String answer = *getAnswer*(Arrays.*asList*("1", "2", "3", "0"));  
 switch (answer) {  
 case "1":  
 String subject;  
 System.*out*.print("предмет : "); subject = *in*.nextLine();  
 *UL*.find(subject);  
 break;  
 case "2":  
 int day, month;  
 System.*out*.print("день : "); day = *checkIntegerInput*();  
 System.*out*.print("месяц : "); month = *checkIntegerInput*();  
 *UL*.find(day, month);  
 break;  
 case "3":  
 String teacherName, teacherSurname;  
 System.*out*.print("имя : "); teacherName = *in*.nextLine();  
 System.*out*.print("фамилия : "); teacherSurname = *in*.nextLine();  
 *UL*.find(teacherName, teacherSurname);  
 break;  
 }  
 *searchMenu*();  
 }  
  
 private static void searchStudent() {  
 System.*out*.println();  
 String group;  
 System.*out*.print("группа : "); group = *in*.nextLine();  
 *UL*.findStudent(group);  
 *searchMenu*();  
 }  
  
 private static void outputMenu() {  
 System.*out*.println();  
 System.*out*.println("1 - вывести список аттестаций");  
 System.*out*.println("2 - вывести список студентов");  
 System.*out*.println("0 - назад");  
  
 String answer = *getAnswer*(Arrays.*asList*("1", "2", "0"));  
 switch (answer) {  
 case "1":  
 *outputAttestation*();  
 break;  
 case "2":  
 *outputStudent*();  
 break;  
 case "0":  
 *startMenu*();  
 break;  
 }  
 }  
  
 private static void outputAttestation() {  
 System.*out*.println();  
 *UL*.showAttestation();  
 *outputMenu*();  
 }  
  
 private static void outputStudent() {  
 System.*out*.println();  
 *UL*.showStudents();  
 *outputMenu*();  
 }  
  
 private static void showMarksMenu() {  
 System.*out*.println();  
 String name, surname, group;  
 System.*out*.print("имя : "); name = *in*.nextLine();  
 System.*out*.print("фамилия : "); surname = *in*.nextLine();  
 System.*out*.print("группа : "); group = *in*.nextLine();  
 *UL*.showMarks(name, surname, group);  
 *startMenu*();  
 }  
}

7 Список использованных источников

1. Объектно-ориентированное программирование. Метод. указания по

выполнению лабораторных работ / сост.: В.Л. Аршинский. – Иркутск : Изд-во

ИРНИТУ, 2017. – 24 c.