|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |
| **Институт информационных технологий** |
| **Кафедра корпоративных информационных систем**  **Лабораторная работа №3**  по дисциплине  Объектно-ориентированное программирование  **Тема лабораторной работы:** Классы. Диаграммы классов в языке UML   |  |  | | --- | --- | | Студент группы ИКБО-08-18 | Валяев Данила Андреевич | |  |  | |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) | |  |  | |  |  | | Работа представлена к защите | «11» ноября 2019 г. | |  |  | |  |  | |
|  |
| **Содержание**  [**Задание** 3](#_Toc27908433)  [**Заключение** 7](#_Toc27908434)  [**Список используемых источников** 8](#_Toc27908435)  [**Приложение** 9](#_Toc27908436) |

# **Задание:** Реализовать классы в соответствии с предложенной диаграммой.

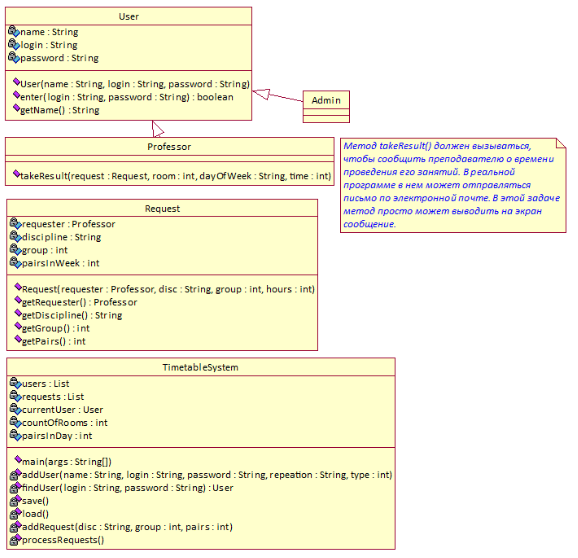


Рисунок 1.1 – Диаграмма варианта

Интерфейсы:

- Интерфейс для входа в систему.

- Интерфейс для регистрации пользователей (преподавателей).

- Интерфейс администратора (для указания числа аудиторий и др.)

- Интерфейс для приема заявок.

- Интерфейс для составления расписания.

Таблицы данных:

- Пользователи

- Заявки

- Информация об университете (число аудиторий и др.)

Реализация класса пользователя:

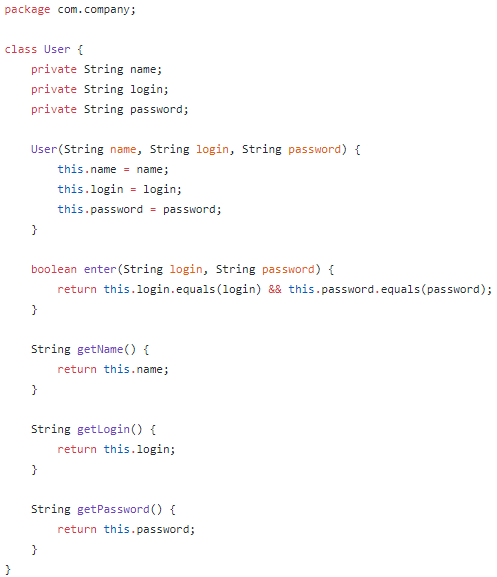


Рисунок 1.2 – Класс пользователя

Реализация класса преподавателя:

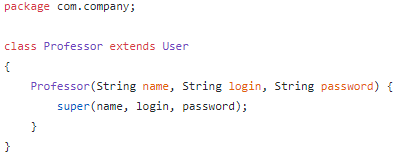


Рисунок 1.3 – Класс преподавателя

**UML** – унифицированный язык моделирования (Unified Modeling Language) – это система обозначений, которую можно применять для объектно-ориентированного анализа и проектирования. Его можно использовать для визуализации, спецификации, конструирования и документирования программных систем.

Реализация класса запроса:

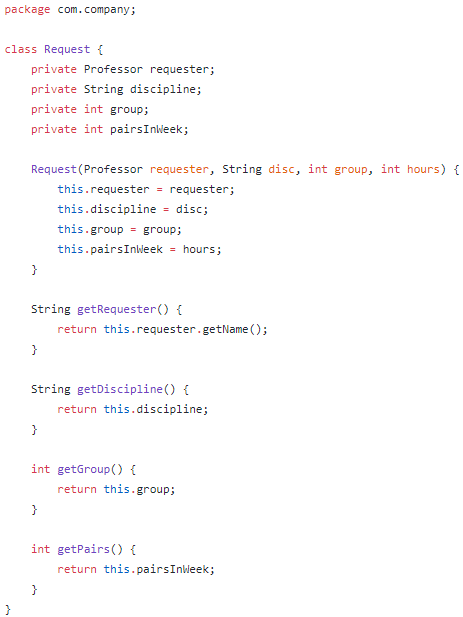


Рисунок 1.4 – Класс запроса

Реализация функции входа пользователя:

private boolean loginUser() {

Scanner in = new Scanner(System.in);

print("Введите логин: ");

String login = in.nextLine();

print("Введите пароль: ");

String password = in.nextLine();

Timetable.currentUser = this.findUser(login, password);

return Timetable.currentUser.enter(login, password);

}

Реализация функции построения расписания:



Рисунок 1.5 – Функция построения расписания

# **Заключение**

В процессе выполнения лабораторной работы мы познакомились с UML диаграммами.

# **Список используемых источников**

1. UML[Электронный ресурс]. – URL: https://prog-cpp.ru/uml-classes/

# **Приложение**

TimeTable

package com.company;

import java.util.Scanner;

import java.util.ArrayList;

public class Timetable

{

private static ArrayList<Professor> users;

private static ArrayList<Request> requests;

private static Pairs[][] pairs;

private static Professor currentUser;

private int rooms;

private int groups;

public static void main(String[] args) {

Timetable timetbl = new Timetable();

for (int i = 0; i < 6; ++i) {

for (int j = 0; j < 6; ++j) {

Timetable.pairs[i][j] = new Pairs();

}

}

Scanner in = new Scanner(System.in);

int menu = -1;

while (menu != 0) {

menu = -1;

println("\nСистема автоматического составления расписания");

println("[0] - Выход из программы");

println("[1] - Вход");

println("[2] - Регистрация");

println("[3] - Меню администратора");

try {

menu = in.nextInt();

}

catch (IllegalArgumentException e) {

println("Некорректный ввод");

continue;

}

switch (menu) {

case 1: {

try{

if (timetbl.loginUser()) {

while (menu != 0) {

println("Здравствуйте, " + Timetable.currentUser.getName() + "!");

println("[0] - Выход в главное меню");

println("[1] - Подача заявки");

println("[2] - Ваше расписание");

Scanner in4 = new Scanner(System.in);

menu = in4.nextInt();

switch (menu) {

case 1: {

Scanner in5 = new Scanner(System.in);

print("Введите название дисциплины: ");

String disc = in5.nextLine();

print("Введите группу: ");

int group = in5.nextInt();

print("Введите количество пар: ");

int pairs = in5.nextInt();

timetbl.addRequest(disc, group, pairs);

continue;

}

case 2: {

timetbl.printPairs(Timetable.currentUser);

}

case 0: {

continue;

}

default: {

println("Некорректный ввод");

}

}

}

menu = -1;

continue;

}

}

catch (NullPointerException e){

println("Данный пользователь не зарегистрирован");

continue;

}

}

case 2: {

Scanner in3 = new Scanner(System.in);

print("Введите Ваше имя: ");

String name = in3.nextLine();

print("Введите Ваш логин: ");

String login = in3.nextLine();

print("Введите Ваш пароль: ");

String pass = in3.nextLine();

timetbl.addUser(name, login, pass);

continue;

}

case 3: {

Scanner in2 = new Scanner(System.in);

print("Введите логин: ");

String login = in2.next();

print("Введите пароль: ");

String pass = in2.next();

if (login.equals("admin") && pass.equals("admin")) {

while (menu != 0) {

println("Здравствуйте, администратор!");

println("[0] - Выход в главное меню");

println("[1] - Указать количество аудиторий");

println("[2] - Указать количество групп");

menu = in2.nextInt();

switch (menu) {

case 1: {

println("Введите количество аудиторий: ");

int rooms = in2.nextInt();

timetbl.setRooms(rooms);

continue;

}

case 2: {

println("Введите количество групп: ");

int groups = in2.nextInt();

timetbl.setGroups(groups);

continue;

}

case 0: {

timetbl.processRequests();

println("Расписание составлено!");

}

}

}

}

menu = -1;

continue;

}

case 0: {

println("Работа завершена");

break;

}

default: {

println("Некорректный ввод");

menu = -1;

}

}

}

}

private boolean loginUser() {

Scanner in = new Scanner(System.in);

print("Введите логин: ");

String login = in.nextLine();

print("Введите пароль: ");

String password = in.nextLine();

Timetable.currentUser = this.findUser(login, password);

return Timetable.currentUser.enter(login, password);

}

private void addUser(String name, String login, String password) {

if (this.findUser(login, password) != null) {

println("Пользователь уже зарегистрирован");

return;

}

Timetable.currentUser = new Professor(name, login, password);

Timetable.users.add(Timetable.currentUser);

}

private void addRequest(String disc, int group, int pairs) {

Request r = new Request(Timetable.currentUser, disc, group, pairs);

Timetable.requests.add(r);

}

private void processRequests() {

for (Request r : Timetable.requests) {

int number = r.getPairs();

for (int i = 0; i < 6; ++i) {

for (int j = 0; j < 6; ++j) {

int room = 1;

if (Timetable.pairs[i][j].getRooms() < this.rooms && number > 0) {

while (Timetable.pairs[i][j].checkRoom(room)) {

++room;

}

if (!Timetable.pairs[i][j].checkGroup(r.getGroup()) && !Timetable.pairs[i][j].checkProfessor(r.getRequester())) {

Timetable.pairs[i][j].setPair(r.getRequester(), r.getDiscipline(), room, r.getGroup(), j, i);

Timetable.pairs[i][j].Group\_(r.getGroup());

Timetable.pairs[i][j].Prof\_(r.getRequester());

Timetable.pairs[i][j].Room\_(room);

--number;

}

}

}

}

}

}

private Professor findUser(String login, String password) {

for (Professor i : Timetable.users) {

if (i.getLogin().equals(login) && i.getPassword().equals(password)) {

return i;

}

}

return null;

}

private void printPairs(Professor user) {

println("Ваше расписание:");

for (int i = 0; i < 6; ++i) {

for (int j = 0; j < 6; ++j) {

Timetable.pairs[i][j].printPair(user.getName());

}

}

}

private void setRooms(int rooms) {

this.rooms = rooms;

}

private void setGroups(int groups) {

this.groups = groups;

}

private static void println(String text){

System.out.println(text);

}

private static void print(String text){

System.out.print(text);

}

static {

Timetable.users = new ArrayList<>();

Timetable.requests = new ArrayList<>();

Timetable.pairs = new Pairs[6][6];

}

}

User

package com.company;

class User {

private String name;

private String login;

private String password;

User(String name, String login, String password) {

this.name = name;

this.login = login;

this.password = password;

}

boolean enter(String login, String password) {

return this.login.equals(login) && this.password.equals(password);

}

String getName() {

return this.name;

}

String getLogin() {

return this.login;

}

String getPassword() {

return this.password;

}

}

Professor

package com.company;

class Professor extends User

{

Professor(String name, String login, String password) {

super(name, login, password);

}

}

Pairs

package com.company;

import java.util.ArrayList;

class Pairs

{

private String[] days;

private ArrayList<Integer> room;

private ArrayList<Integer> group;

private ArrayList<String> professor;

private ArrayList<String> pair;

Pairs() {

this.days = new String[] { "Понедельник", "Вторник", "Среда", "Четверг", "Пятница", "Суббота" };

this.room = new ArrayList<Integer>();

this.group = new ArrayList<Integer>();

this.professor = new ArrayList<String>();

this.pair = new ArrayList<String>();

}

void Room\_(int room) {

this.room.add(room);

}

void Group\_(int group) {

this.group.add(group);

}

void Prof\_(String professor) {

this.professor.add(professor);

}

boolean checkRoom(int room) {

return this.room.contains(room);

}

boolean checkGroup(int group) {

return this.group.contains(group);

}

boolean checkProfessor(String professor) {

return this.professor.contains(professor);

}

int getRooms() {

return this.room.size();

}

void setPair(String professor, String disc, int room, int group, int hour, int day) {

this.pair.add("группа: " + group + ", профессор: " + professor + ", дисциплина: " + disc + ", аудитория: " +

room + ", пара:" + (hour + 1) + ", день: " + this.days[day]);

}

void printPair(String prof) {

for (String i : this.pair) {

if (i.contains(prof)) {

System.out.println(i);

}

}

}

}

Request

package com.company;

class Request {

private Professor requester;

private String discipline;

private int group;

private int pairsInWeek;

Request(Professor requester, String disc, int group, int hours) {

this.requester = requester;

this.discipline = disc;

this.group = group;

this.pairsInWeek = hours;

}

String getRequester() {

return this.requester.getName();

}

String getDiscipline() {

return this.discipline;

}

int getGroup() {

return this.group;

}

int getPairs() {

return this.pairsInWeek;

}

}