|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт Информационных технологий | |
|  | |
| Кафедра Математического обеспечения и стандартизации информационных технологий | |
|  | |
|  | |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3** | |
| **по дисциплине** | |
| **«**Объектно-ориентированное программирование**»**  **Тема: «Классы. Диаграммы классов в языке UML»** | |
|  | |
| Выполнил студент группы ИКБО-17-18 | Мисаилиди А.А. |
| Принял преподаватель | Баранова И.А. |

|  |  |  |
| --- | --- | --- |
| Лабораторная работа выполнена | «\_\_»\_\_\_\_\_\_\_201\_\_ г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_201\_\_ г. | *(подпись руководителя)* |

Москва 2019

Задание 1. Разработка программы с применением математической модели

1. Условие задачи

Написать программу для проведения выборов. Избиратели должны предварительно зарегистрироваться в системе. Администратор заполняет список кандидатов. Каждый участник (по сети) или с того же самого компьютера входит в систему и голосует. Данные о проголосовавших накапливаются в базе данных. Дважды проголосовать нельзя. По окончании периода голосования администратор запускает процедуру подсчета голосов и система выдает результат.

2. Постановка задачи

Основные алгоритмы - Подведение итогов выборов.

Интерфейсы:

* Интерфейс для входа в систему.
* Интерфейс для регистрации пользователей.
* Интерфейс для создания голосования.
* Интерфейс для голосования.
* Интерфейс для проверки результатов голосования.

# **3. UML-диаграмма вариантов использования**

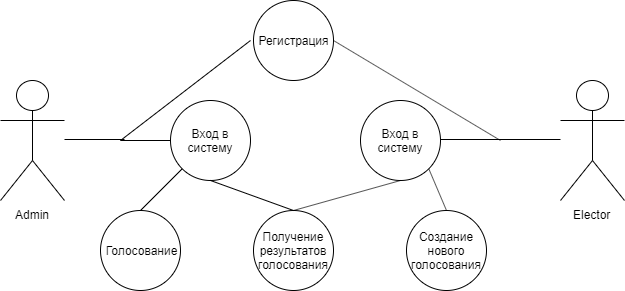


Рис. 1 – диаграмма вариантов использования

# **4. UML-диаграмма последовательности**

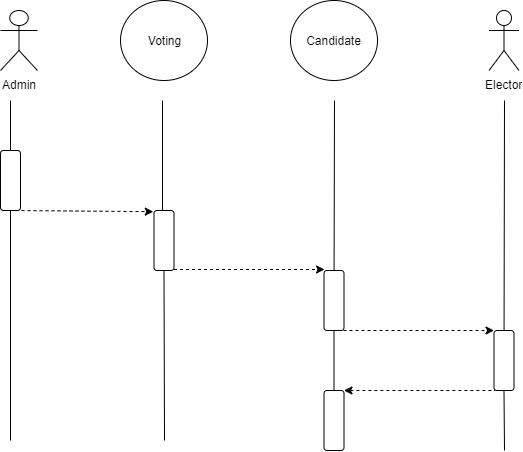


Рис. 2 – диаграмма последовательности действий

**5. UML-диаграмма классов**

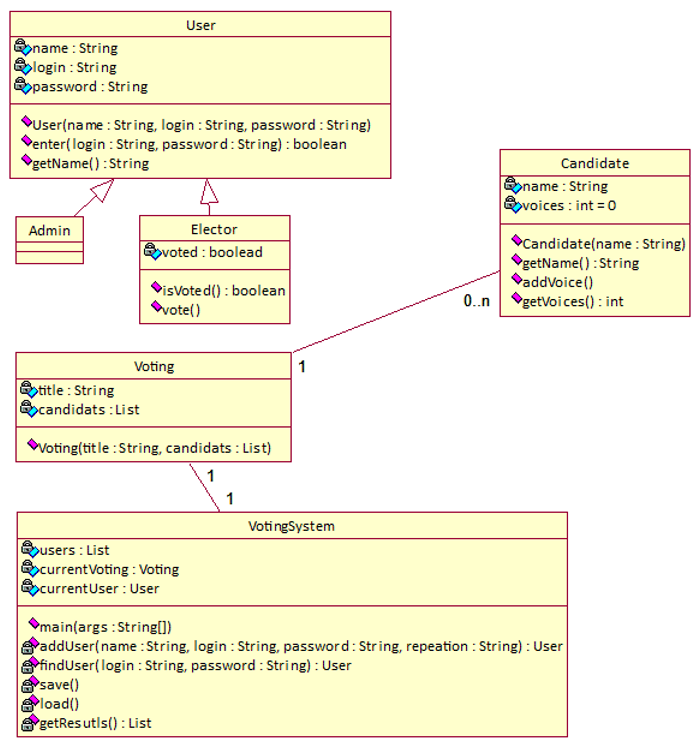


Рис. 3 – диаграмма вариантов использования UML

6. Разработка тестов

1. тест – регистрация нового пользователя

1. Login

2. Registration

0. Exit

Choose variant: 2

Enter your name: 1

Enter your login: 1

Enter your password: 1

Enter your password again: 1

1. Admin

2. Elector

Choose your role: 1

New admin registered

2. тест – вход в аккаунт

1. Login

2. Registration

0. Exit

Choose variant: 1

Enter your login: 1

Enter your password: 1

Logined as admin – 1

3. тест – проведение новых выборов

1. Hold new elections

2. Add new candidate in the list

3. Election results

0. Exit account

Choose variant: 1

Enter elections title: president

4. тест – добавление кандидата

1. Hold new elections

2. Add new candidate in the list

3. Election results

0. Exit account

Choose variant: 2

Enter candidate's name: puten

5. тест – голосование

1. Vote

2. Election results

0. Exit account

Choose variant: 1

Elections: president

1. puten

2. obama

0. Vote for none

Choose the candidate from the list: 1

6. тест – проверка результатов

1. Vote

2. Election results

0. Exit account

Choose variant: 2

Elections: president

1. puten: 1

2. obama: 0

7. тест – выход из аккаунта

1. Vote

2. Election results

0. Exit account

Choose variant: 0

Exiting the account

8. тест – завершение программы

1. Login

2. Registration

0. Exit

Choose variant: 0

Ending the program...

**Работа с Git**

Для того, чтобы добавить изменение в репозиторий на GitHub, нужно выполнить несколько операций:

* Добавление изменений
* Добавление коммита
* Пуш коммита в репозиторий OOP\_lab03-kazzart

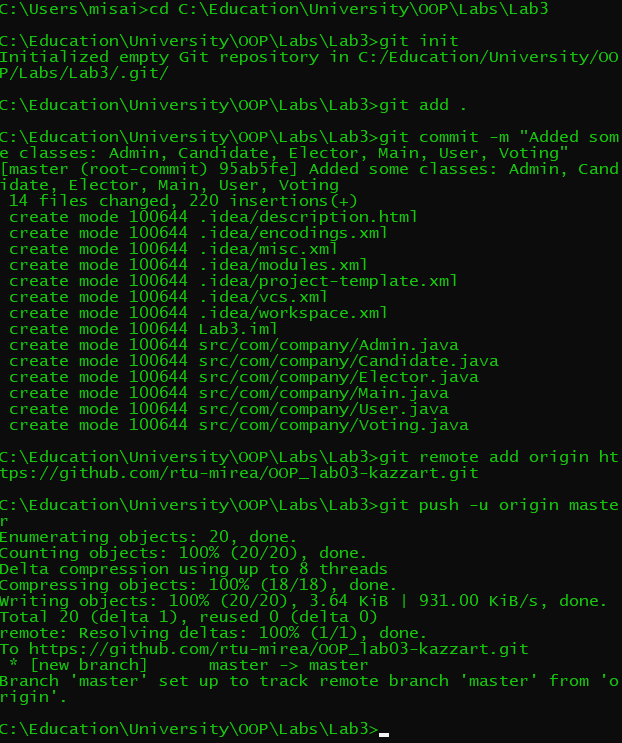


Рис. 4 - Работа с гит. Добавление первого коммита «Added some classes:…»

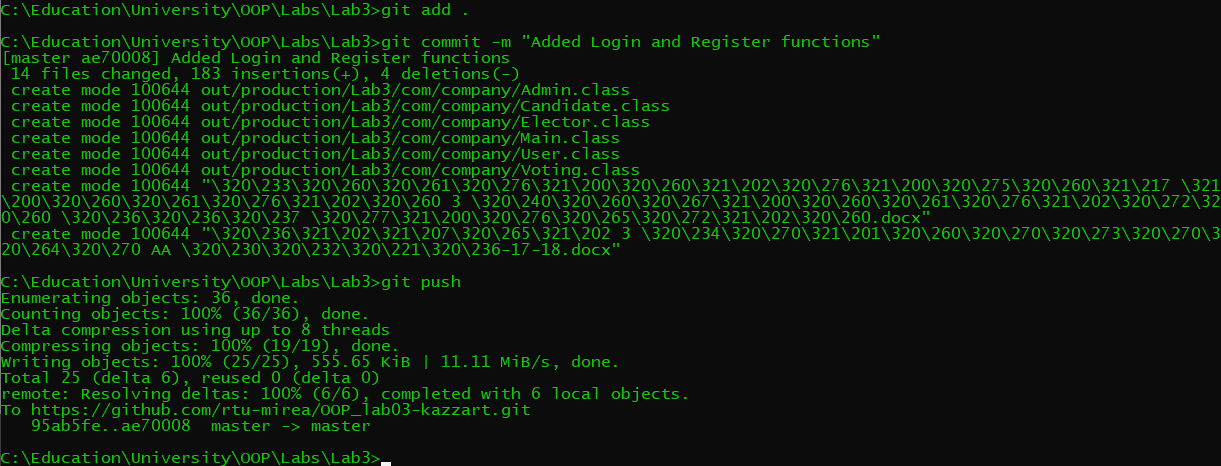


Рис. 5 - Работа с гит. Добавление второго коммита «Added Login and Register functions»

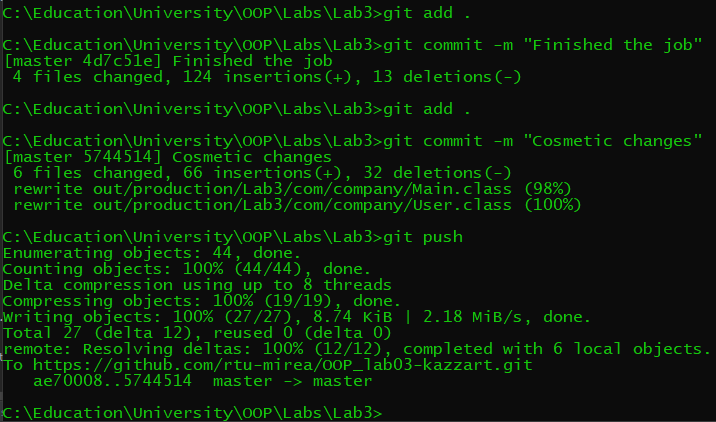


Рис. 6 - Работа с гит. Добавление первого коммита «Finished the job»

**Исходный код программы**

**Main.java**

package com.company;

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class Main {

private static List<User> users = new ArrayList<>();

private static Voting currentVoting;

private static User currentUser;

public static void main(String[] args) {

int comm0 = -1;

Scanner in = new Scanner(System.in);

while (comm0 != 0) {

outputMenu0();

System.out.print("Choose variant: ");

comm0 = in.nextInt();

System.out.println();

switch (comm0) {

case 1:

findUserProcess();

if (currentUser != null && currentUser.access == 0) {

int comm1 = -1;

while (comm1 != 0) {

outputMenu1\_1();

System.out.print("Choose variant: ");

comm1 = in.nextInt();

System.out.println();

switch (comm1) {

case 1:

vote();

break;

case 2:

getResults();

break;

case 0:

System.out.println("Exiting the account");

System.out.println();

currentUser = null;

break;

default:

System.out.println("Entered wrong command");

System.out.println();

break;

}

}

} else if(currentUser != null && currentUser.access == 1) {

int comm1 = -1;

while (comm1 != 0) {

outputMenu1\_2();

System.out.print("Choose variant: ");

comm1 = in.nextInt();

System.out.println();

switch (comm1) {

case 1:

addVoting();

break;

case 2:

addNewCandidate();

break;

case 3:

getResults();

break;

case 0:

System.out.println("Exiting the account");

System.out.println();

currentUser = null;

break;

default:

System.out.println("Entered wrong command");

System.out.println();

break;

}

}

}

break;

case 2:

addUserProcess();

break;

case 0:

System.out.println("Ending the program...");

break;

default:

System.out.println("Entered wrong command");

break;

}

}

}

private static User addUser(String name, String login, String password, String role) throws Exception {

if (!users.isEmpty()) {

for (User user: users) {

if (user.login.equals(login)) {

System.out.println("0");

throw new Exception("Login already exists");

}

}

}

if (role.equals("admin")) {

Admin user = new Admin(name, login, password);

users.add(user);

return user;

} else if (role.equals("elector")) {

Elector user = new Elector(name, login, password);

users.add(user);

return user;

}

throw new Exception("Wrong role!");

}

private static void addUserProcess() {

boolean done = false;

while (!done) {

Scanner in = new Scanner(System.in);

System.out.print("Enter your name: ");

String name = in.nextLine();

System.out.print("Enter your login: ");

String login = in.nextLine();

System.out.print("Enter your password: ");

String password1 = in.nextLine();

System.out.print("Enter your password again: ");

String password2 = in.nextLine();

if (password1.equals(password2)) {

int roleN = 0;

System.out.println("1. Admin");

System.out.println("2. Elector");

while (roleN < 1) {

System.out.print("Choose your role: ");

roleN = in.nextInt();

System.out.println();

switch (roleN) {

case 1:

try {

addUser(name, login, password1, "admin");

System.out.println("New admin registered");

System.out.println();

done = true;

} catch (Exception e) {

System.out.println(e.getMessage());

}

break;

case 2:

try {

addUser(name, login, password1, "elector");

System.out.println("New elector registered");

System.out.println();

done = true;

} catch (Exception e) {

System.out.println(e.getMessage());

}

break;

default:

System.out.println("Entered wrong command");

break;

}

}

} else {

System.out.println("Passwords are different");

}

}

}

private static User findUser(String login, String password) throws Exception {

if (!users.isEmpty()) {

for (User user: users) {

if (user.login.equals(login)) {

if (user.password.equals(password)) {

return user;

} else {

throw new Exception("Wrong password!");

}

}

}

}

throw new Exception("Login not found!");

}

private static void findUserProcess() {

Scanner in = new Scanner(System.in);

System.out.print("Enter your login: ");

String login = in.nextLine();

System.out.print("Enter your password: ");

String password1 = in.nextLine();

System.out.println();

try {

currentUser = findUser(login, password1);

if (currentUser.access == 1) {

System.out.println("Logined as admin - " + currentUser.login);

System.out.println();

} else {

System.out.println("Logined as elector - " + currentUser.login);

System.out.println();

}

} catch (Exception e) {

System.out.println(e.getMessage());

}

}

private static void addVoting() {

Scanner in = new Scanner(System.in);

System.out.print("Enter elections title: ");

String title = in.nextLine();

System.out.println();

currentVoting = new Voting(title, new ArrayList<>());

}

private static void addNewCandidate() {

if (currentVoting != null) {

Scanner in = new Scanner(System.in);

System.out.print("Enter candidate's name: ");

String name = in.nextLine();

System.out.println();

if (!currentVoting.candidates.isEmpty()) {

for (Candidate candidate : currentVoting.candidates) {

if (candidate.getName().equals(name)) {

System.out.println("Candidate is already on the list");

return;

}

}

}

currentVoting.candidates.add(new Candidate(name));

} else {

System.out.println("There is no any elections right now");

}

}

private static void vote() {

Scanner in = new Scanner(System.in);

if (!currentVoting.candidates.isEmpty()) {

if (currentUser.lastElection != currentVoting) {

System.out.println("Elections: " + currentVoting.title);

for (int i = 0; i < currentVoting.candidates.size(); i++) {

System.out.println(Integer.toString(i + 1) + ". " + currentVoting.candidates.get(i).getName());

}

System.out.println("0. Vote for none");

System.out.println();

int choiceDone = -1;

while (choiceDone < 0 || choiceDone > currentVoting.candidates.size()) {

System.out.print("Choose the candidate from the list: ");

choiceDone = in.nextInt();

System.out.println();

if (choiceDone >= 0 && choiceDone <= currentVoting.candidates.size()) {

if (choiceDone == 0) {

currentUser.lastElection = currentVoting;

} else {

Candidate votedCandidate = currentVoting.candidates.get(choiceDone - 1);

votedCandidate.addVoice();

currentVoting.candidates.set(choiceDone - 1, votedCandidate);

currentUser.lastElection = currentVoting;

}

} else {

System.out.println("Wrong input!");

choiceDone = -1;

}

}

} else {

System.out.println("You cannot vote in this elections anymore");

}

} else {

System.out.println("There is no any person in the candidates list");

}

}

private static void getResults() {

Scanner in = new Scanner(System.in);

if (currentVoting != null && !currentVoting.candidates.isEmpty()) {

System.out.println("Elections: " + currentVoting.title);

for (int i = 0; i < currentVoting.candidates.size(); i++) {

System.out.println(Integer.toString(i + 1) + ". " + currentVoting.candidates.get(i).getName() + ": " +

currentVoting.candidates.get(i).getVoices());

}

System.out.println();

} else {

System.out.println("There is no any person in the candidates list");

System.out.println();

}

}

private static void outputMenu0() {

System.out.println("1. Login");

System.out.println("2. Registration");

System.out.println("0. Exit");

}

private static void outputMenu1\_1() {

System.out.println("1. Vote");

System.out.println("2. Election results");

System.out.println("0. Exit account");

}

private static void outputMenu1\_2() {

System.out.println("1. Hold new elections");

System.out.println("2. Add new candidate in the list");

System.out.println("3. Election results");

System.out.println("0. Exit account");

}

}

**User.java**

package com.company;

public class User {

protected String name;

protected String login;

protected String password;

public int access;

public Voting lastElection;

public User() {

}

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

setName(name);

setLogin(login);

setPassword(password);

}

public void setLogin(String login) {

this.login = login;

}

public void setName(String name) {

this.name = name;

}

public void setPassword(String password) {

this.password = password;

}

public String getLogin() {

return login;

}

public String getName() {

return name;

}

public String getPassword() {

return password;

}

public boolean enter(String login, String password) {

if (this.login == login && this.password == password) {

return true;

} else {

return false;

}

}

}

**Admin.java**

package com.company;

public class Admin extends User {

public Admin(String name, String login, String password) {

super(name, login, password);

access = 1;

}

}

**Elector.java**

package com.company;

public class Elector extends User {

private boolean voted = false;

public Elector(String name, String login, String password) {

super(name, login, password);

access = 0;

}

public boolean isVoted() {

return voted;

}

public void vote() {

voted = true;

}

}

**Candidate.java**

package com.company;

public class Candidate {

private String name;

private int voices = 0;

public Candidate(String name) {

this.name = name;

}

public String getName() {

return name;

}

public void addVoice() {

voices++;

}

public int getVoices() {

return voices;

}

}

**Voting.java**

package com.company;

import java.util.ArrayList;

import java.util.List;

public class Voting {

public String title;

public List<Candidate> candidates;

public Voting(String title, List<Candidate> candidates) {

this.title = title;

this.candidates = candidates;

}

}