**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ**

**И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

(СПбГУТ)

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И ПРОГРАММНОЙ ИНЖЕНЕРИИ (**ИТПИ**)

КАФЕДРА ПРОГРАММНОЙ ИНЖЕНЕРИИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ (ПИ И ВТ)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Лабораторная работа №6**

по дисциплине «Разработка фронтенд-приложений управления телекоммуникациями»

Тема «Цепочка ответственности для обработки запросов»

Вариант 24

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент гр. | ИКПИ-23 |  | 09.12.2024 | Харлова А.А. |
| Преподаватель |  |  |  | Белая Т.И. |

г. Санкт-Петербург

2024 г.

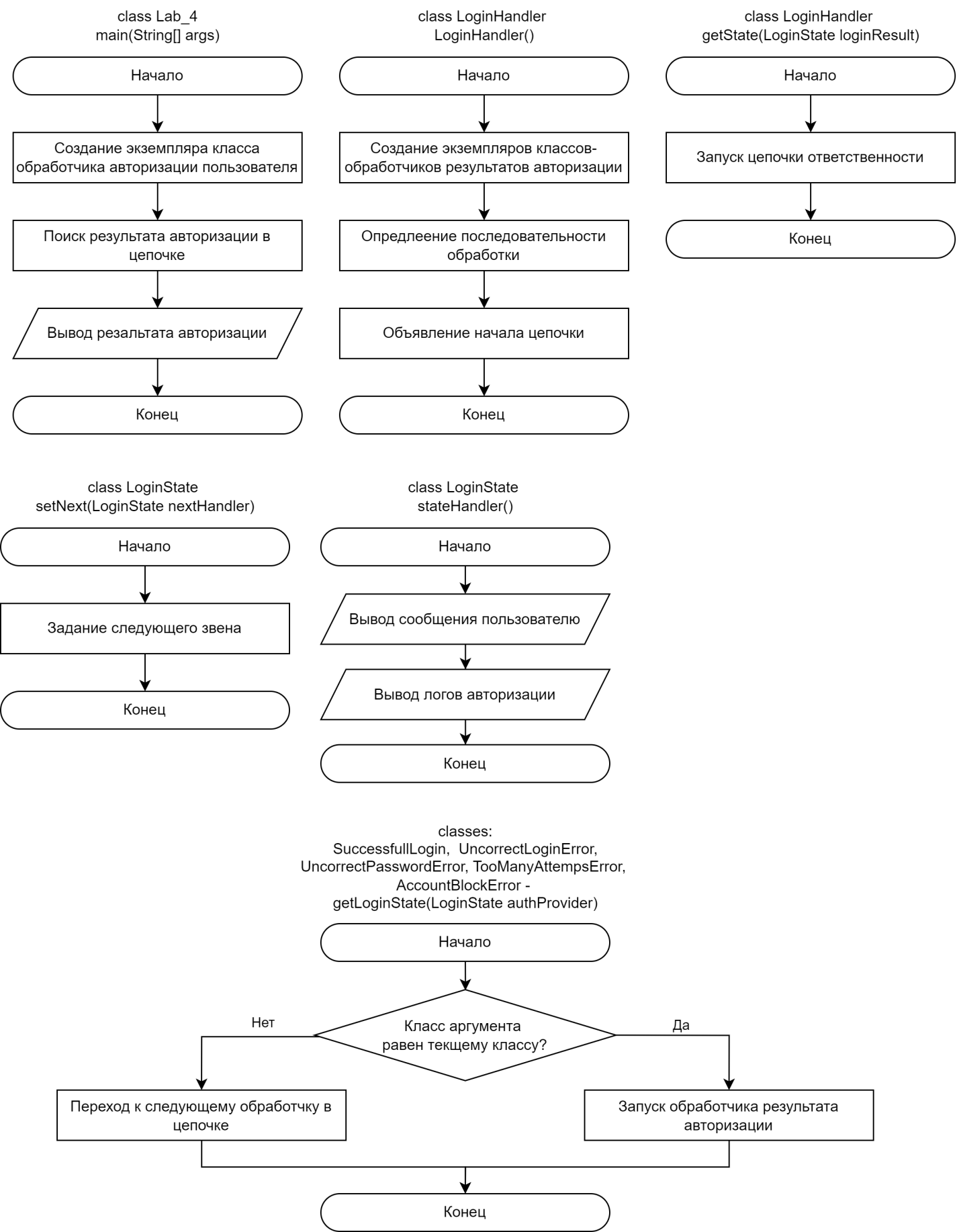
# ПОСТАНОВКА ЗАДАЧИ

В настоящей лабораторной работе следует создать паттерн "Цепочка ответственности" для обработки запросов, позволяя передавать запросы по цепочке обработчиков до тех пор, пока не будет найден подходящий.

**РАЗРАБОТКА АЛГОРИТМА**

Для реализации паттерна были выбраны запросы авторизации пользователя.

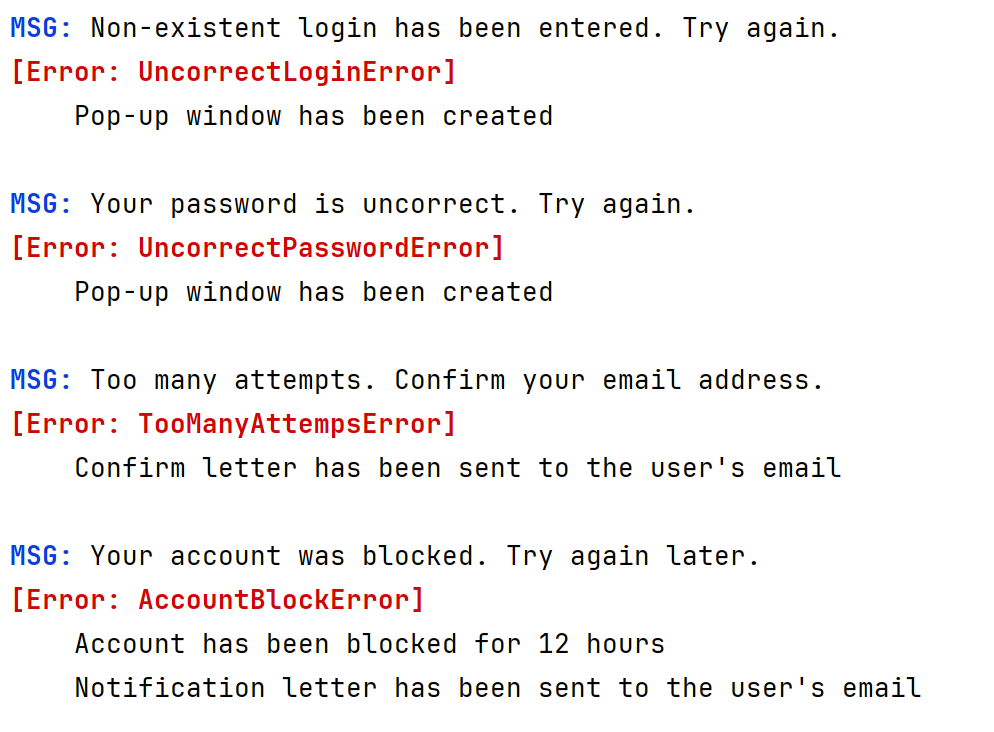
1. Создание точки входа в программу. Разработка класса "lab\_4" с функцией "main", создающего экземпляр цепочки-обработчика авторизации и экземпляры классов возможных результатов попытки авторизации пользователя для тестирования работы программы.
2. Создание класса цепочки ответственности "LoginHandler", определяющего порядок функций обработки авторизации пользователя и возвращающего результат входа в систему.
3. Создание класса "LoginState", хранящего информацию о текущем и следующем звене цепочки ответственности.
4. Создание классов "SuccessfulLogin", "TooManyAttempsError", "UncorrectLoginError", "UncorrectPasswordError", "AccountBlockError" различных типов ошибок авторизации, наследуемых от "LoginState".



*Рисунок 1 - Алгоритм программы*



*Рисунок 2 - Пример работы программы*



*Рисунок 3 - Пример вывода-обработки некорреткных входных данных*

**ЛИСТИНГ ПРОГРАММЫ**

**«lab\_4.java»**

package lab\_4;  
  
public class lab\_4 {  
  
 public static void main(String[] args) {  
 *// обработчик результатов авторизации* LoginHandler handler = new LoginHandler();  
  
 *// создание экземпляров классов всех возможных результатов попытки авторизации пользователя* LoginState success = new SuccessfulLogin();  
 LoginState uncorrectLogin = new UncorrectLoginError();  
 LoginState uncorrectPassword = new UncorrectPasswordError();  
 LoginState tooManyAttemps = new TooManyAttempsError();  
 LoginState accountBlock = new AccountBlockError();  
  
 *// тестированаие* handler.getState(success);  
 handler.getState(uncorrectLogin);  
 handler.getState(uncorrectPassword);  
 handler.getState(tooManyAttemps);  
 handler.getState(accountBlock);  
 }  
}

**«LoginHandler.java»**

package lab\_4;  
  
*// класс-обработчик результата авторизации пользователя*public class LoginHandler {  
 */\*  
 объявление шаблона "цепочки ответственности"  
 для обработки успешного входа в систему  
 и возможных ошибок авторизации пользователя  
 \*/* private final LoginState chainOfResponsibility;  
  
 LoginHandler() {  
 */\*  
 создание экземпляров классов-обработчиков возможных результатов авторизации  
 ( звенья "цепочки ответственности")  
 \*/* LoginState success = new SuccessfulLogin();  
 LoginState uncorrectLogin = new UncorrectLoginError();  
 LoginState uncorrectPassword = new UncorrectPasswordError();  
 LoginState tooManyTryings = new TooManyAttempsError();  
 LoginState accountBlock = new AccountBlockError();  
  
 *// определение последовательности обработки* success.setNext(uncorrectLogin);  
 uncorrectLogin.setNext(uncorrectPassword);  
 uncorrectPassword.setNext(tooManyTryings);  
 tooManyTryings.setNext(accountBlock);  
  
 *// объявление начала "цепочки"* this.chainOfResponsibility = success;  
 }  
  
 *// функция получения результата авторизации* public void getState(LoginState loginResult) {  
 *// запуск "цепочки ответственности"* chainOfResponsibility.getLoginState(loginResult);  
 }  
}

**«LoginState.java»**

package lab\_4;  
  
*// абстрактный класс-результат авторизации пользователя  
// (звено "цепочки отвественности")*public abstract class LoginState {  
 *// переменная, указывающая на следущее звено  
 // "цепочки отвественности"* public LoginState nextLoginState;  
  
 *// функция поиска и обработки нужного звена  
 // "цепочки отвественности"* abstract public void getLoginState(LoginState authProvider);  
  
 *// задание следующего возможного звена  
 // "цепочки отвественности"* public void setNext(LoginState nextHandler) {  
 this.nextLoginState = nextHandler;  
 }  
  
 *// обработчик результата авторизации* public void stateHandler() {  
 showMessage();  
 showActionsLog();  
 System.*out*.print('\n');  
 }  
  
 *// вывод сообщения для пользователя* abstract public void showMessage();  
  
 *// вывод логов дальнейших действий* abstract public void showActionsLog();  
}

**«AccountBlockError.java»**

package lab\_4;  
  
*// Класс-обработчик блокировки аккаунта пользователя  
// из-за большого количества неуспешных попыток входа в аккаунт*public class AccountBlockError extends LoginState {  
 @Override  
 public void getLoginState(LoginState authProvider) {  
 if (authProvider instanceof AccountBlockError) {  
 this.stateHandler();  
 } else if (this.nextLoginState != null) {  
 this.nextLoginState.getLoginState(authProvider);  
 }  
 }  
  
 @Override  
 public void stateHandler() {  
 this.showMessage();  
 this.showActionsLog();  
 }  
  
 @Override  
 public void showMessage() {  
 System.*out*.println("\033[1;34mMSG:\033[0m Your account was blocked. Try again later.");  
 }  
  
 @Override  
 public void showActionsLog() {  
 System.*out*.println("\033[1;31m[Error: AccountBlockError]\033[0m");  
 System.*out*.println("\tAccount has been blocked for 12 hours\n" +  
 "\tNotification letter has been sent to the user's email");  
 }  
}

**«SuccessfulLogin.java»**

package lab\_4;  
  
*// Класс-обработчик успешной авторизации пользовател*public class SuccessfulLogin extends LoginState {  
 @Override  
 public void getLoginState(LoginState authProvider) {  
 if (authProvider instanceof SuccessfulLogin) {  
 this.stateHandler();  
 } else if (this.nextLoginState != null) {  
 this.nextLoginState.getLoginState(authProvider);  
 }  
 }  
  
 @Override  
 public void showMessage() {  
 System.*out*.println("\033[1;34mMSG:\033[0m Welcome!");  
 }  
  
 @Override  
 public void showActionsLog() {  
 System.*out*.println("\033[1;32m[The user <User@name> is logged in]\033[0m");  
 }  
}

**«TooManyAttempsError.java»**

package lab\_4;  
  
*// Класс-обработчик большого количства неуспешных попыток входа в аккаунт*public class TooManyAttempsError extends LoginState {  
 @Override  
 public void getLoginState(LoginState authProvider) {  
 if (authProvider instanceof TooManyAttempsError) {  
 this.stateHandler();  
 } else if (this.nextLoginState != null) {  
 this.nextLoginState.getLoginState(authProvider);  
 }  
 }  
  
 @Override  
 public void showMessage() {  
 System.*out*.println("\033[1;34mMSG:\033[0m Too many attempts. Confirm your email address.");  
 }  
  
 @Override  
 public void showActionsLog() {  
 System.*out*.println("\033[1;31m[Error: TooManyAttempsError]\033[0m");  
 System.*out*.println("\tConfirm letter has been sent to the user's email");  
 }  
}

**«UncorrectLoginError.java»**

package lab\_4;  
  
public class UncorrectLoginError extends LoginState {  
 @Override  
 public void getLoginState(LoginState authProvider) {  
 if (authProvider instanceof UncorrectLoginError) {  
 this.stateHandler();  
 } else if (this.nextLoginState != null) {  
 this.nextLoginState.getLoginState(authProvider);  
 }  
 }  
  
 @Override  
 public void showMessage() {  
 System.*out*.println("\033[1;34mMSG:\033[0m Non-existent login has been entered. Try again.");  
 }  
  
 @Override  
 public void showActionsLog() {  
 System.*out*.println("\033[1;31m[Error: UncorrectLoginError]\033[0m");  
 System.*out*.println("\tPop-up window has been created");  
 }  
}

**«UncorrectPasswordError.java»**

package lab\_4;  
  
*// Класс-обработчик пользовательского ввода некорректного пароля*public class UncorrectPasswordError extends LoginState {  
  
 @Override  
 public void getLoginState(LoginState authProvider) {  
 if (authProvider instanceof UncorrectPasswordError) {  
 this.stateHandler();  
 } else if (this.nextLoginState != null) {  
 this.nextLoginState.getLoginState(authProvider);  
 }  
 }  
  
 @Override  
 public void showMessage() {  
 System.*out*.println("\033[1;34mMSG:\033[0m Your password is uncorrect. Try again.");  
 }  
  
 @Override  
 public void showActionsLog() {  
 System.*out*.println("\033[1;31m[Error: UncorrectPasswordError]\033[0m");  
 System.*out*.println("\tPop-up window has been created");  
 }  
}