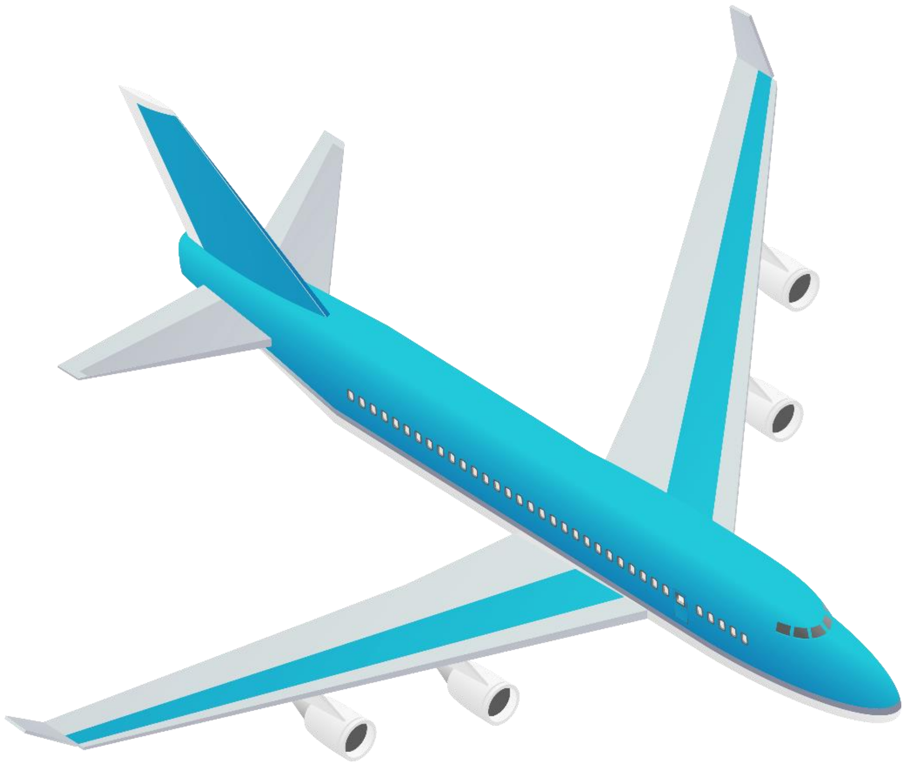
Plane ticket manager

Object oriented programming mini project



Student: Ciausu Ioan-Calin

Group: 30423

Coordinator: Garleanu Stefan Alexandru

Table of content

Abstract

In this project we want to create a simple GUI Java application, so we have chosen a user-friendly plane ticket management system. The main functionalities are searching for flights, buying, and modifying tickets. Moreover, it supports multiple users.

User manual

Opening screen

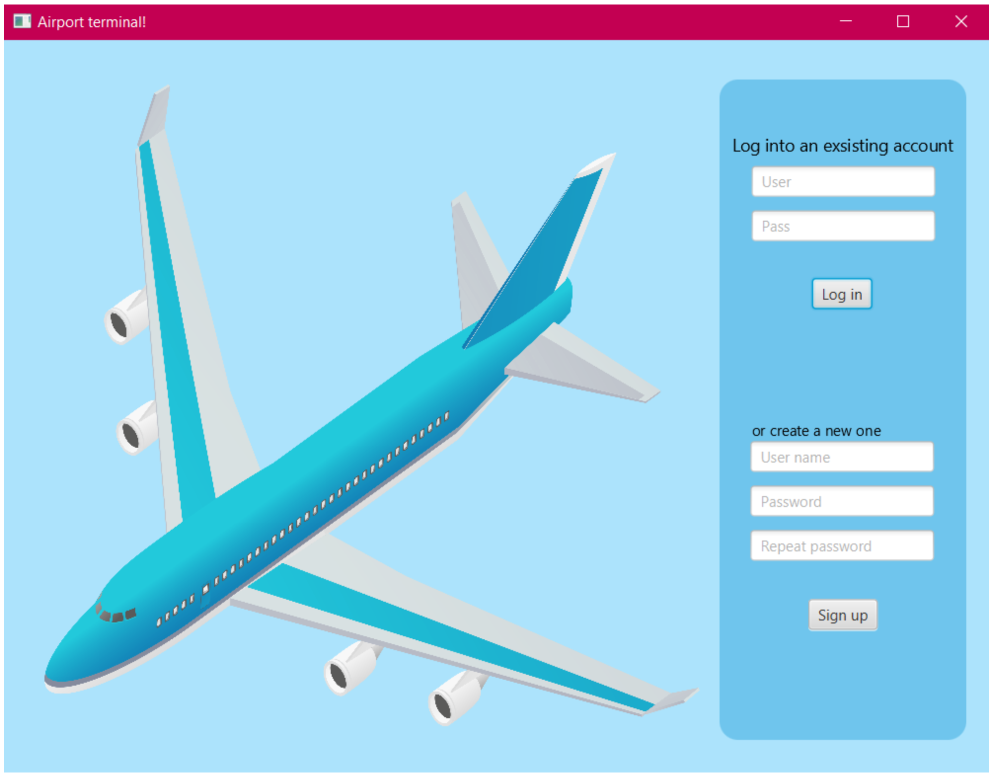


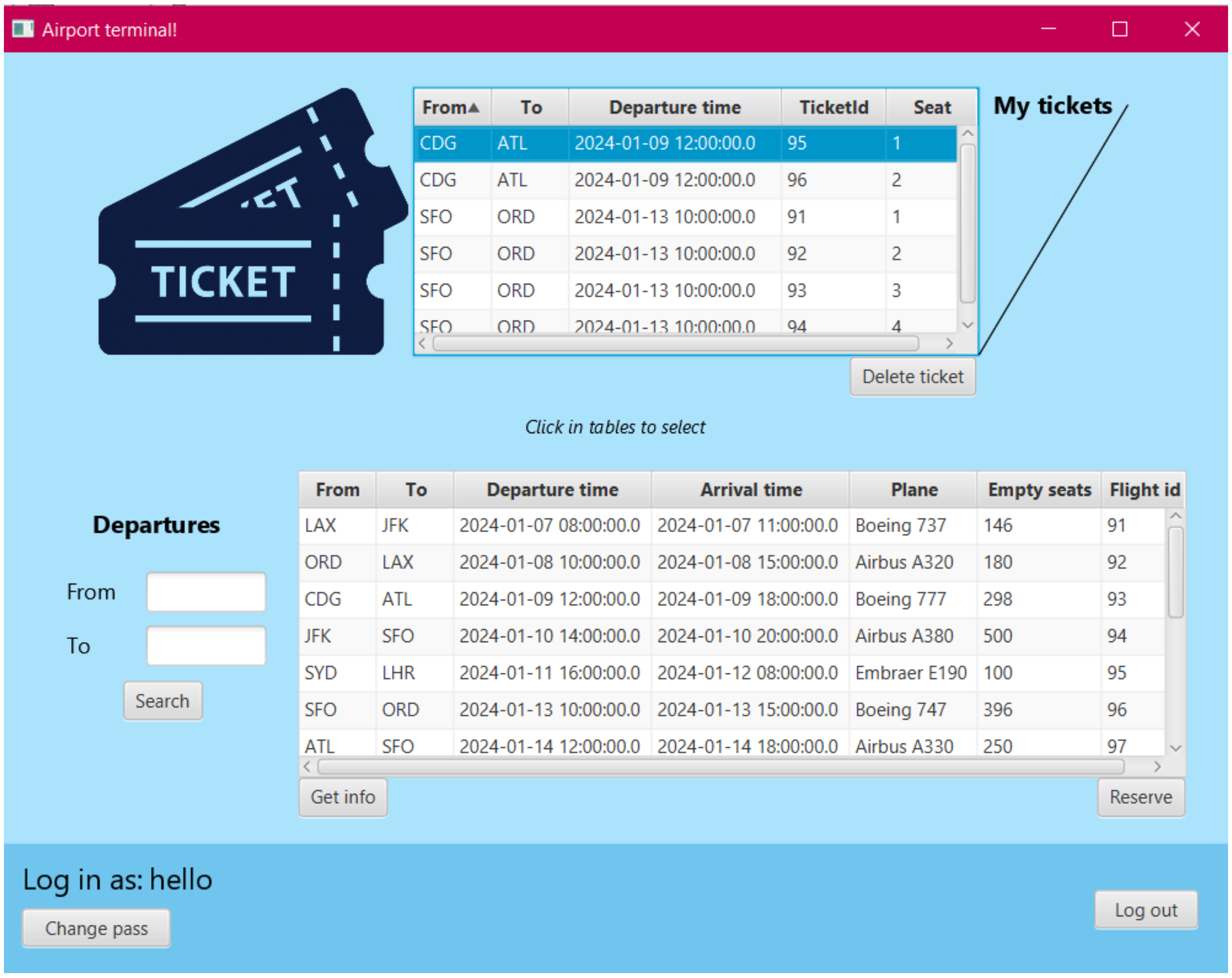
Image of the opening screen

Old users can log in easily by introducing their username and password in the top fields, while the new ones can easily sign up by choosing a username and a password. If the username is already in use or the two copies of the passwords do not match, the user will receive an error message.

For testing purpose, we recommend:

Username: *John*, Password: *hello*

Main screen



Airport terminal!

From▲	To	Departure time	TicketId	Seat
CDG	ATL	2024-01-09 12:00:00.0	95	1
CDG	ATL	2024-01-09 12:00:00.0	96	2
SFO	ORD	2024-01-13 10:00:00.0	91	1
SFO	ORD	2024-01-13 10:00:00.0	92	2
SFO	ORD	2024-01-13 10:00:00.0	93	3
SFO	ORD	2024-01-13 10:00:00.0	94	4

My tickets

Click in tables to select

Departures

From	To	Departure time	Arrival time	Plane	Empty seats	Flight id
LAX	JFK	2024-01-07 08:00:00.0	2024-01-07 11:00:00.0	Boeing 737	146	91
ORD	LAX	2024-01-08 10:00:00.0	2024-01-08 15:00:00.0	Airbus A320	180	92
CDG	ATL	2024-01-09 12:00:00.0	2024-01-09 18:00:00.0	Boeing 777	298	93
JFK	SFO	2024-01-10 14:00:00.0	2024-01-10 20:00:00.0	Airbus A380	500	94
SYD	LHR	2024-01-11 16:00:00.0	2024-01-12 08:00:00.0	Embraer E190	100	95
SFO	ORD	2024-01-13 10:00:00.0	2024-01-13 15:00:00.0	Boeing 747	396	96
ATL	SFO	2024-01-14 12:00:00.0	2024-01-14 18:00:00.0	Airbus A330	250	97

Log in as: hello

Image of the main screen

The main screen has many functions:

1. Search flights
   Fill in the *From* and *To* fields the departure and landing airports. To allow for any airport leave the associated field empty.
   Click the head of the table to sort the results of the search.
   Click on a flight to select it.
2. Getting information about a specific flight
   When clicking *Get info* while a flight is selected, a new window will open containing more details about it.
3. Buying tickets
   Click *Reserve* after you selected it from the bottom table.
4. Viewing already bought tickets.
   Just look in *My tickets* table.
   Click the head of the table to sort after the corresponding column.

5. Deleting tickets

   Select a ticket from the *My tickets* table and click Delete.

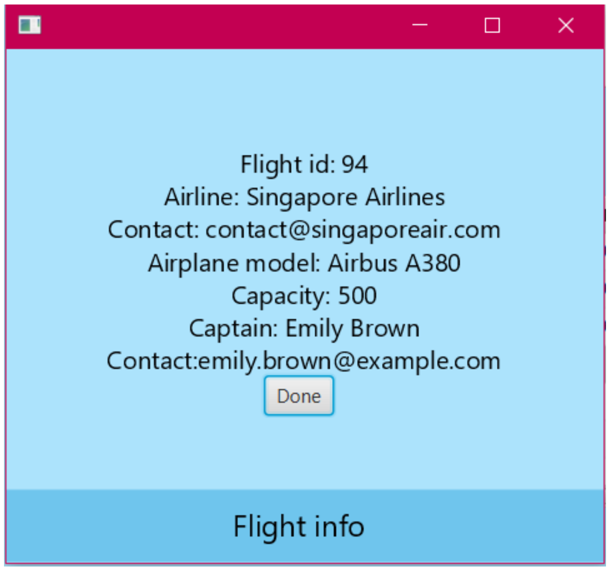
6. Changing password

   Click the *Change pass* button and a new window will pop up.

7. Logging out

   Click the *Log out* button.

Flight info screen



Flight id: 94
Airline: Singapore Airlines
Contact: contact@singaporeair.com
Airplane model: Airbus A380
Capacity: 500
Captain: Emily Brown
Contact:emily.brown@example.com

Image of the flight info screen

This window contains information about the selected flight. To close it click *Done*.

Change password screen

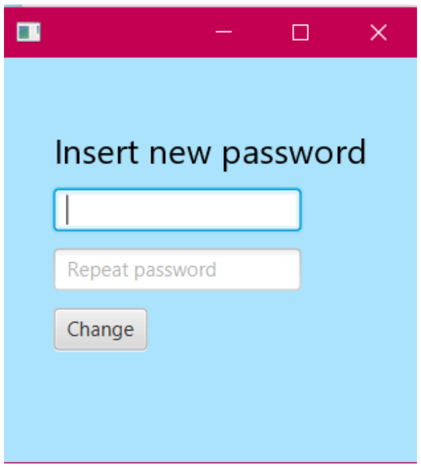


Image of the change password screen

Write the new password twice and click *Change*.

If the 2 copies of the password do not match an error message will be show.

Technologies used

The frontend is build using JavaFX, for a modern looking and responsive UI. On the backend, the application integrates with a PostgreSQL database to store and retrieve essential data such as user profiles, flight information, and booking details.

The PostgreSQL database is hosted locally, and the access information are hardcoded due to the fact that the current application is intended as a demo.

An important implementation detail is that the connection object is passed between the view controllers, instead of being regenerated in each one.

UML Diagram

To keep the diagram intelligible, we show only the action listener of the JavaFX objects.

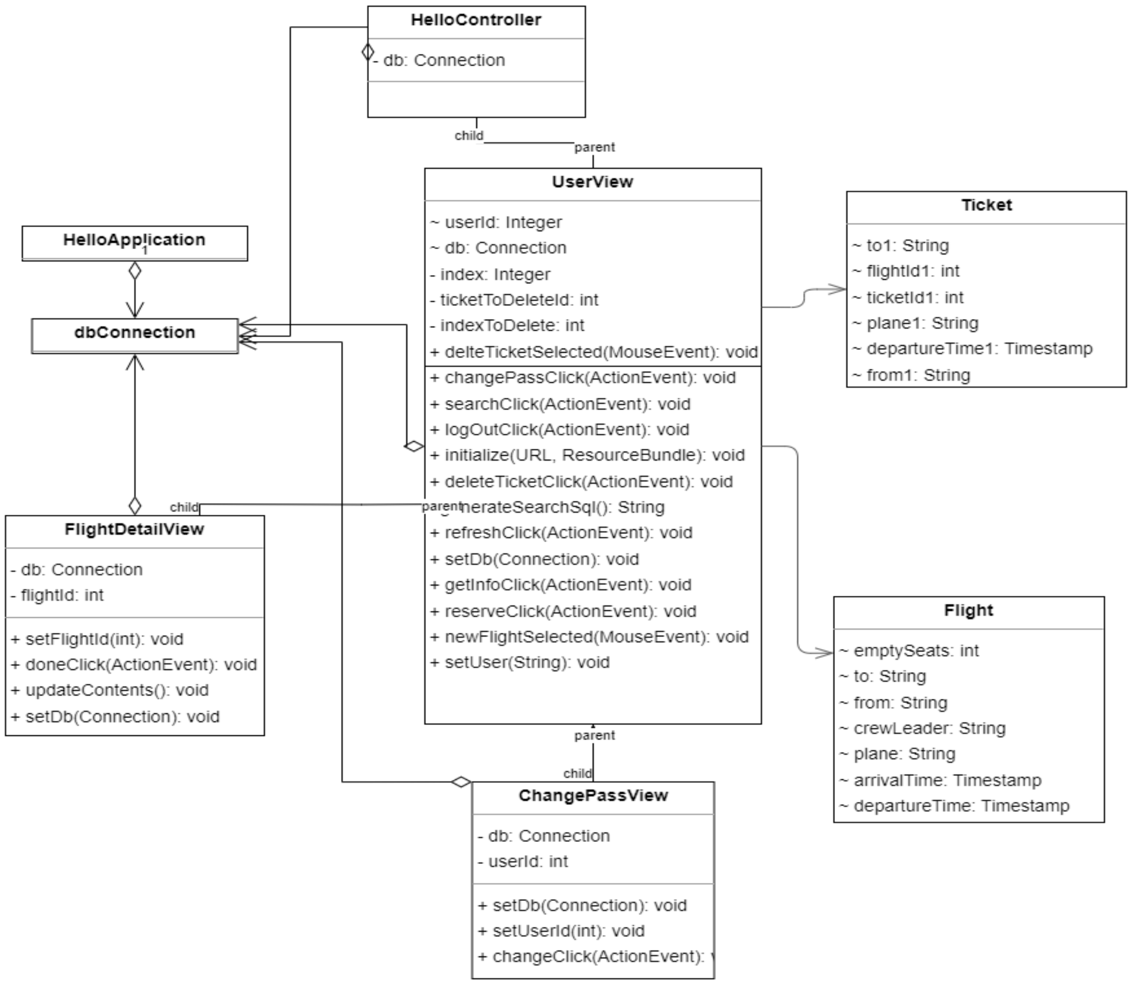
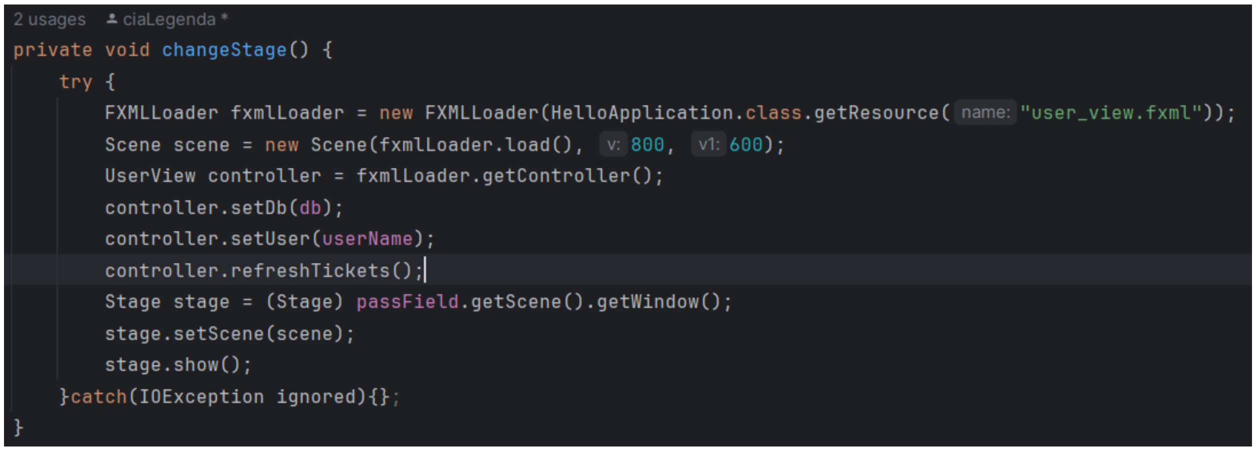

HelloController
- db: Connection
child
parent
UserView
~ userId: Integer
~ db: Connection
- index: Integer
- ticketToDeleteId: int
- indexToDelete: int
+ delteTicketSelected(MouseEvent): void
+ changePassClick(ActionEvent): void
+ searchClick(ActionEvent): void
+ logOutClick(ActionEvent): void
+ initialize(URL, ResourceBundle): void
+ deleteTicketClick(ActionEvent): void
generateSearchSql(): String
+ refreshClick(ActionEvent): void
+ setDb(Connection): void
+ getInfoClick(ActionEvent): void
+ reserveClick(ActionEvent): void
+ newFlightSelected(MouseEvent): void
+ setUser(String): void
Ticket
~ to1: String
~ flightId1: int
~ ticketId1: int
~ plane1: String
~ departureTime1: Timestamp
~ from1: String
HelloApplication
dbConnection
FlightDetailView
- db: Connection
- flightId: int
+ setFlightId(int): void
+ doneClick(ActionEvent): void
+ updateContents(): void
+ setDb(Connection): void
Flight
~ emptySeats: int
~ to: String
~ from: String
~ crewLeader: String
~ plane: String
~ arrivalTime: Timestamp
~ departureTime: Timestamp
ChangePassView
- db: Connection
- userId: int
+ setDb(Connection): void
+ setUserId(int): void
+ changeClick(ActionEvent):

Fig. UML Diagram

Important technical notes

When we need to change the content of the window or to open a new one, we create a new scene, and we associate it with the controller class of the required window. Also, we use setter methods to inject the database connection and username when required by the context.



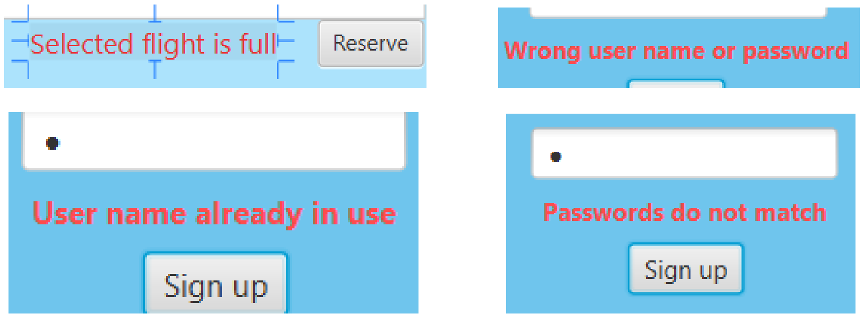
```
2 usages   ciaLegenda *
private void changeStage() {
    try {
        FXMLLoader fxmlLoader = new FXMLLoader(HelloApplication.class.getResource( name: "user_view.fxml"));
        Scene scene = new Scene(fxmlLoader.load(),  v: 800,  v1: 600);
        UserView controller = fxmlLoader.getController();
        controller.setDb(db);
        controller.setUser(userName);
        controller.refreshTickets();
        Stage stage = (Stage) passField.getScene().getWindow();
        stage.setScene(scene);
        stage.show();
    }catch(IOException ignored){};
}
```

Fig. Sniped of code used to transition from the staring page to the main one

The application can detect possible errors in the user's inputs and solve them properly. In particular, the following scenarios are considered:

1. Wrong username-password combo
2. Different copies of the password when duplicates are required
3. Creating an account with an already existing username
4. Pressing *Reserve* with no flight selected
5. Pressing *Delete* with no ticket selected
6. Reserving a ticket on a flight with no empty seats

Regarding 1, 2,3 and 6 an appropriate error message will be displayed in red, while for 3 and 4 the program just ignores the action.



Selected flight is full

Wrong user name or password

User name already in use

Passwords do not match

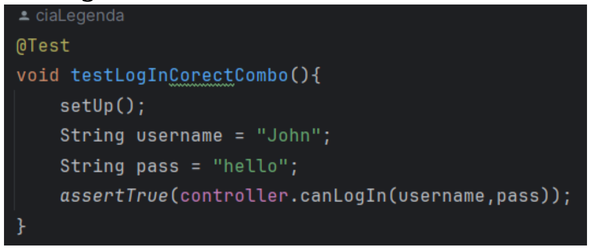
Fig. error messages from the app

The SQL queries use *Statement* instead of *PreparedStatement* presenting a high security risk which should be solved for any real-world implementation!

Tests

The tests are implemented using JUnit and revolve around the log in/sign up process. They include:

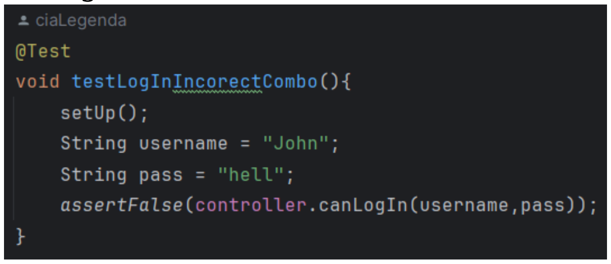
- *testLogInCorrectCombo*



```
ciaLegenda
@Test
void testLogInCorectCombo(){
    setUp();
    String username = "John";
    String pass = "hello";
    assertTrue(controller.canLogIn(username,pass));
}
```

We use a known username-password combo to test the log in functionality.

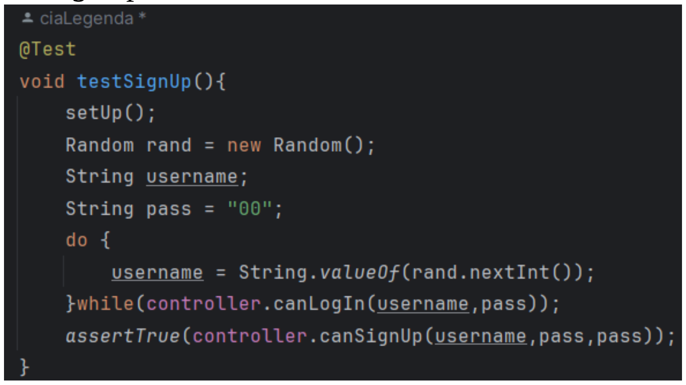
- *testLogInIncorectCombo*



```
ciaLegenda
@Test
void testLogInIncorectCombo(){
    setUp();
    String username = "John";
    String pass = "hell";
    assertFalse(controller.canLogIn(username,pass));
}
```

We use a misspelling of a known password.

- *testSignUp*



```
ciaLegenda *
@Test
void testSignUp(){
    setUp();
    Random rand = new Random();
    String username;
    String pass = "00";
    do {
        username = String.valueOf(rand.nextInt());
    }while(controller.canLogIn(username,pass));
    assertTrue(controller.canSignUp(username,pass,pass));
}
```

This test is a little bit more complicated. To begin with, we generate random usernames until we find one that is not in use. Finally, we test if we can create a new user using the found username and giving two identical copies of the password.