# 

**TEST DOCUMENT**

Quality Assurance of the solution

[Document information](#h.qwydk7x1nscw)

[Document version](#h.qgvwgi1jjpq)

[Approval List](#h.6r7j8dh65x32)

[Confidentiality Rating](#h.hes98nyepwis)

[General](#h.1agiq4l899wt)

[Stress Performance Test](#h.hk3acqe9xl2n)

[User Acceptance Test](#h.mz9eoe1a5djk)

[Other Tests?](#h.e4pd5f1cycku)

# Document information

## Document version

|  |  |  |
| --- | --- | --- |
| Version | Author e-mail | Description |
| 1.1 | Carina Lamb [cari2873@stud.kea.dk](mailto:cari2873@stud.kea.dk)  Dechen Chodon [dech0003@stud.kea.dk](mailto:dech0003@stud.kea.dk)  Lina Alhajar [lina.alhajar@gmail.com](mailto:lina.alhajar@gmail.com)  Muniba Talha [muni0144@stud.kea.dk](mailto:muni0144@stud.kea.dk) | Final Version |
|  |  |  |

## Approval List

|  |  |  |
| --- | --- | --- |
| Who | Function | E-mail |
| Nikolaj B. Hemmeshøj | Head of Enterprise Architecture | nibh@kea.dk |
| Jarl Tuxen | Chief Information Security Officer | jart@kea.dk |

## Confidentiality Rating

|  |  |
| --- | --- |
| Rating |  |
| Company Confidential | X |
| Non Confidential |  |

# **General**

Testing is the process of executing the program with the intent to find errors in it. This document contains information about our testing processes. The purpose of testing is to ensure:

* There is no error connecting to the system.
* System can communicate and send/receive messages.
* A system is meeting the requirements.

**Testing Objectives**

Unit testing is the process of testing the different parts of the software. Normally unit testing performed as the part of coding and the unit test phase of software lifecycle. But it is not an unusual practice to do coding and unit testing separately. There are various organizational approaches that can be followed to develop a strategy for unit testing.

* Top down
* Bottom up
* Isolation

Whichever approach is used to perform unit testing, at the end of the software development cycle unit testing helps us find that whether the project has met or exceeded all the expectations mentioned in the requirements. And to for see all the potential risks, bugs and issues associated with the software. So in this way the development team can work to fix the issues before the release of the software.

# **Stress Performance Test**

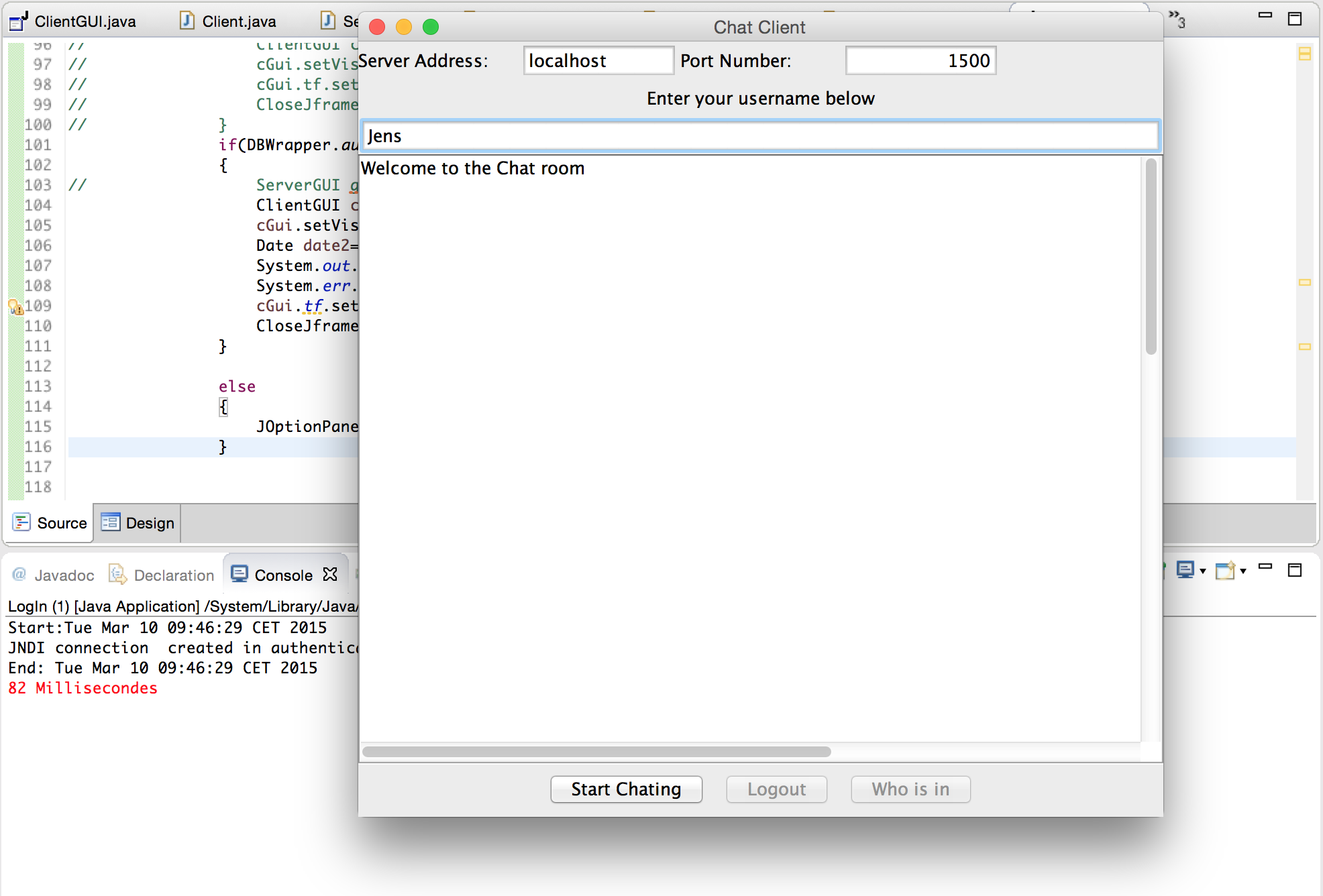
**Load test:**

**Response Time:**

Load test is used to measure the response time of the software when the user makes a demand. Our application is not web based so JMeter testing tool couldn’t prove to be very helpful in getting the results. We tried to test our system by specifying the number of transactions and number of threads. But since we couldn’t measure our system response in JMeter, AppPerfect or any other testing tools, we used (System.***out***.println()) to display the response time.

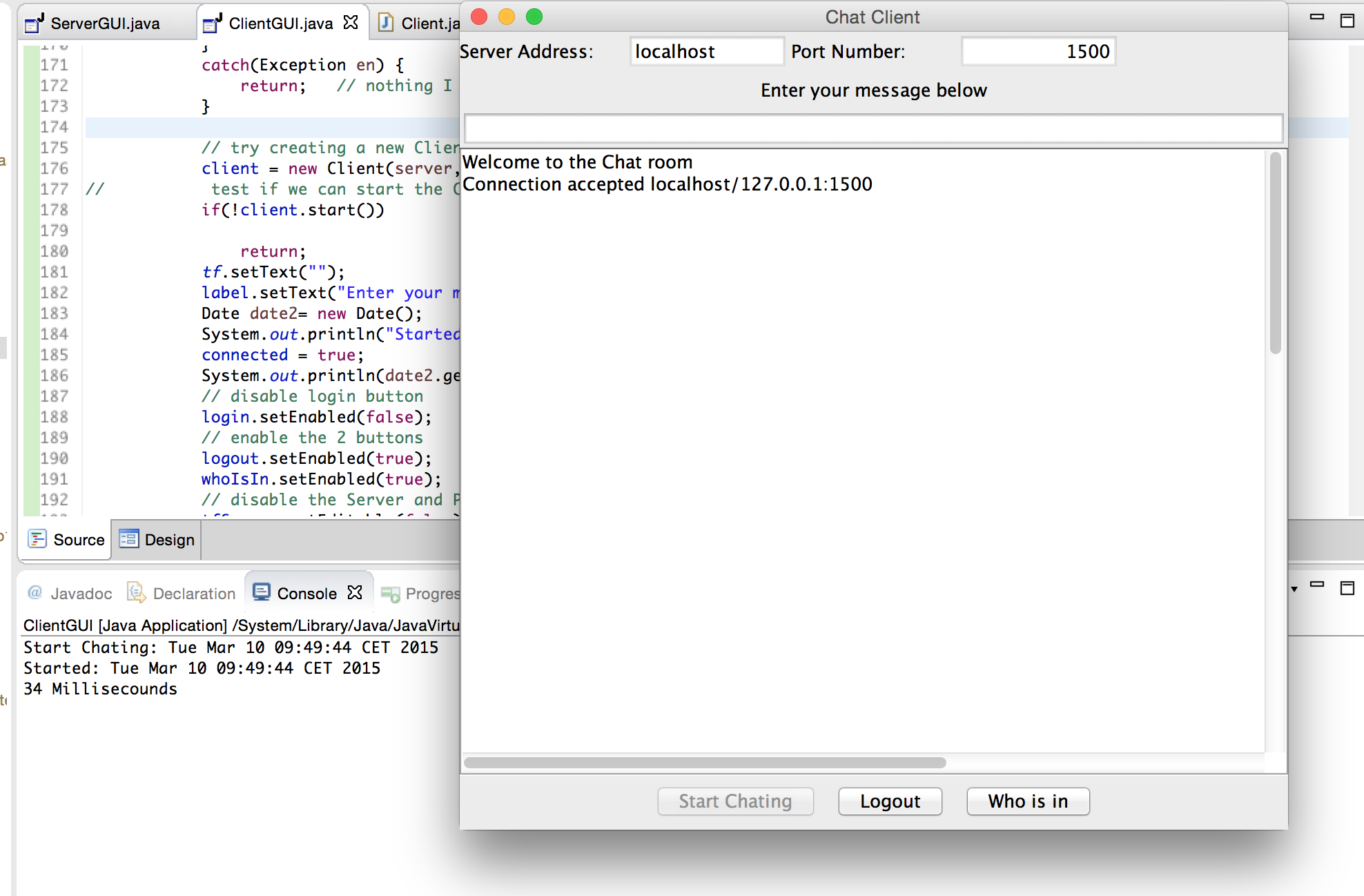
**Case-1: User Log in**

We have tested how long it does takes after the user entered his/her username and password and clicked Log on until he/she entered the Chatting. We have used (System.***out***.println()) at the very start when the user click log on, and (System.***out***.println()) when the chatting started, as shown in console in the picture below.



**Case-2: Start Chatting**

Showing the time it takes between the users clicking start chatting until the chat starts. It takes 34 milliseconds.

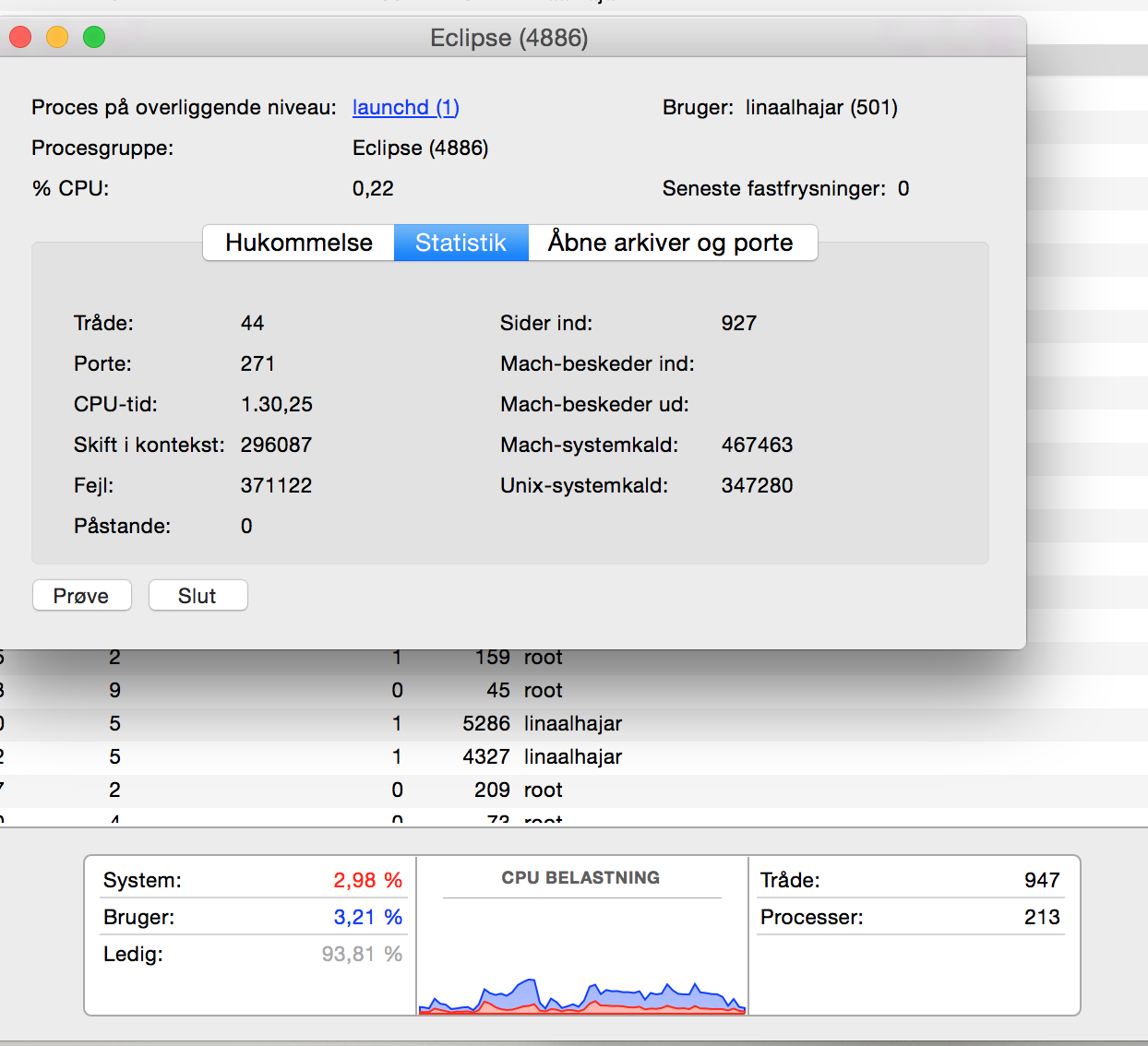
****

**Stress Test**

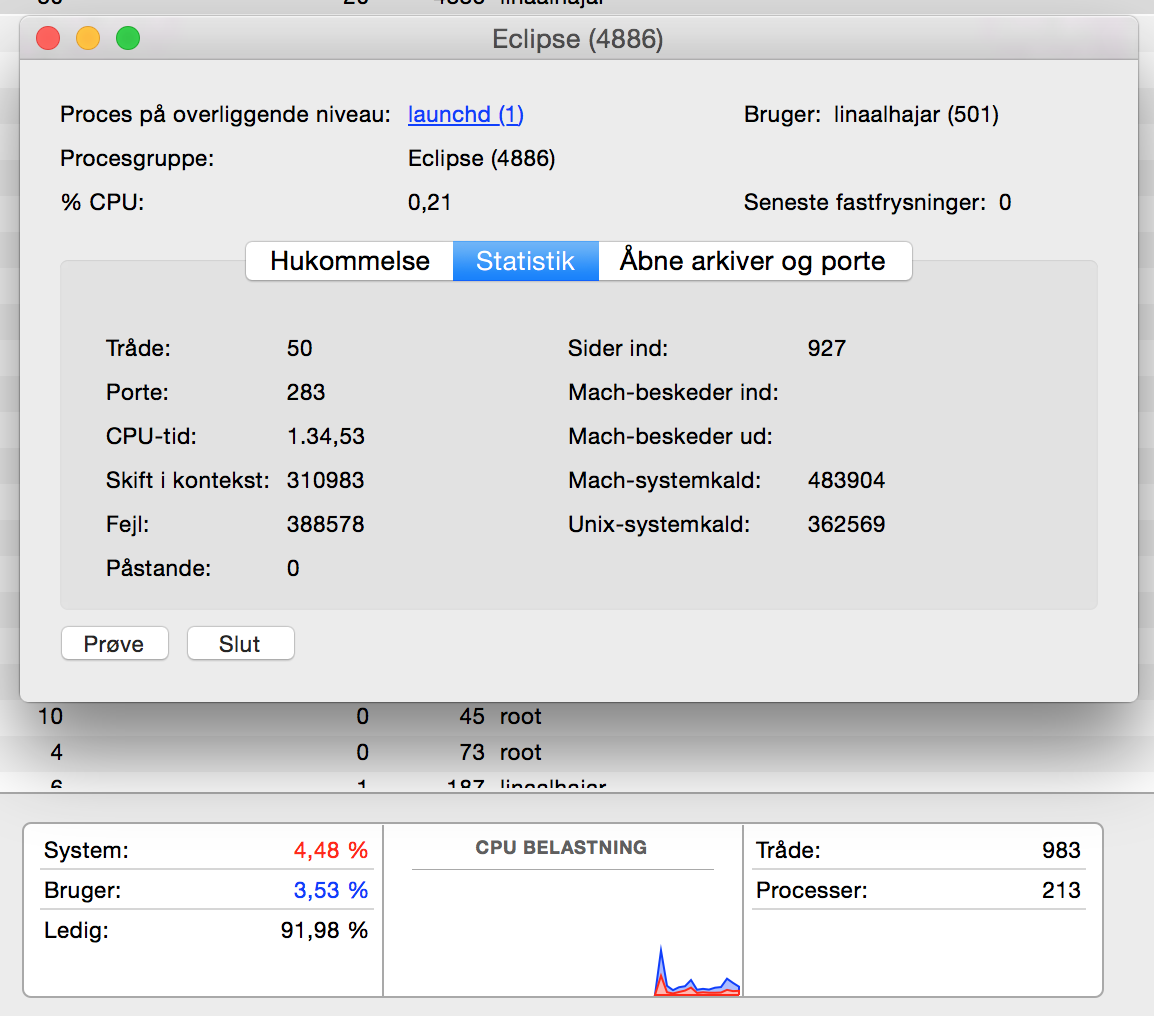
**Stress testing** is a form of deliberately intense or thorough testing used to determine the stability of a given system or entity [wiki]. JMeter testing tool could not be used to perform stress testing on our system. There are tools available to perform Stress Performance Testing (SPT) on desktop-based applications but they are not freeware so we could not use those either.

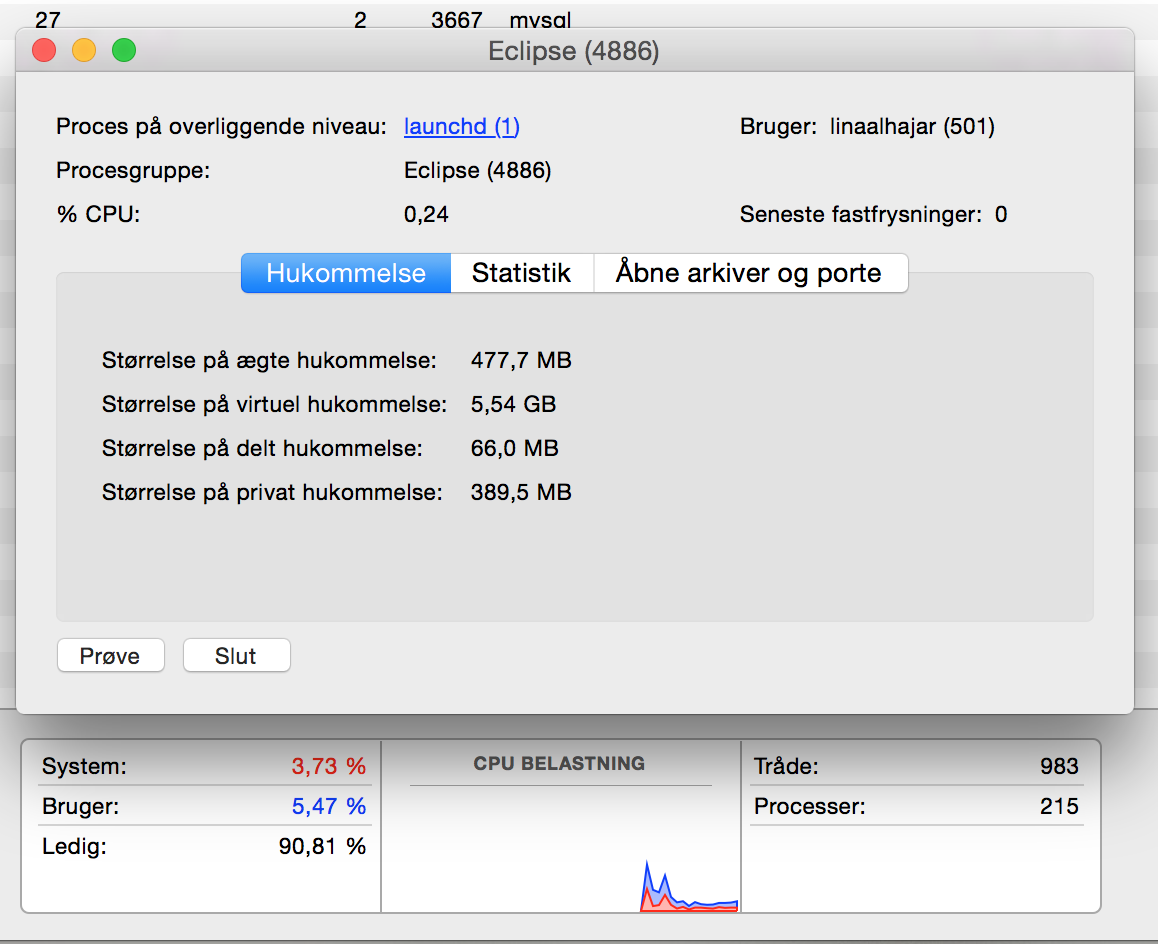
This is why we have test our program by written System.***out***.println()in the code.

**1. Launching the Application**



**2. Starting the Server**

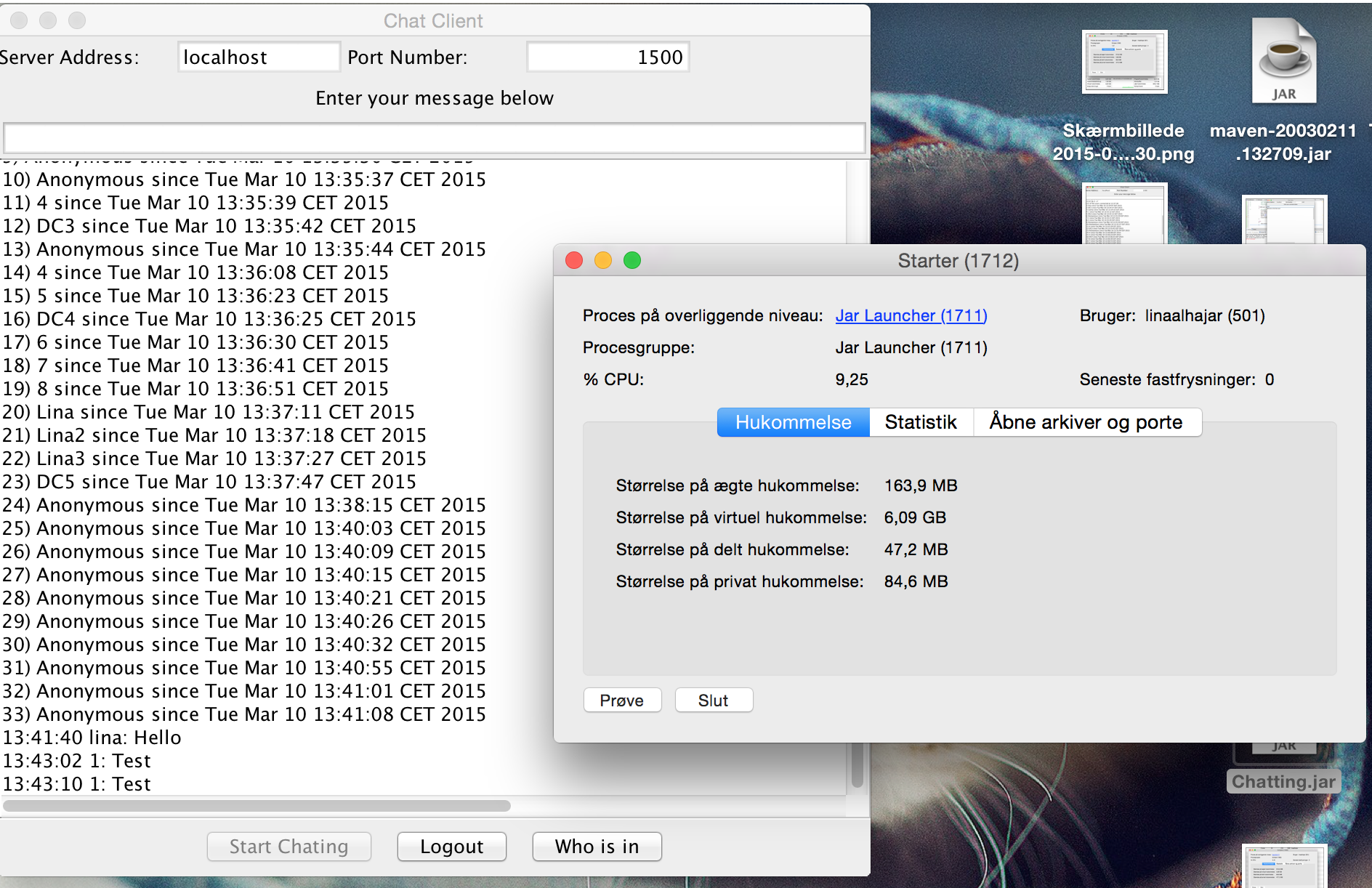


**3. Starting the Client**

**Spike Test**

Spike testing is observing the system behavior by abruptly increasing load on the system. The purpose of this test is to check how system responds whether it crashes or performs well when suddenly two many users try to connect.

We executed 30+ instances of the client on our system and tried to connect to the server to see the results but out system performed very well and it didn’t crash or slow down.



**Soak Test**

No system degradation over time - e.g. memory utilization!!!

This test is to test how the system with a significant load extended over a significant amount of time.

# **User Acceptance Test**

The Client or the user can sign out of the application at anytime.

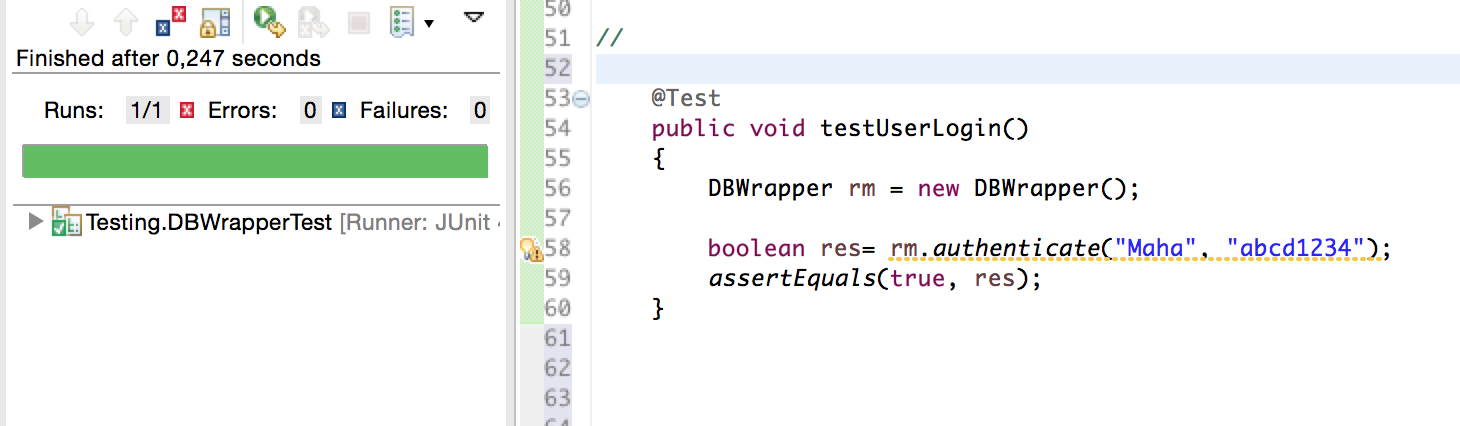
**JUnit Test**

We have used JUnit test to test the functionality required for certain methods of our project to ensure that the code is implemented, and that there are no breaking changes that have not been taken into consideration.

We have tested the following cases:

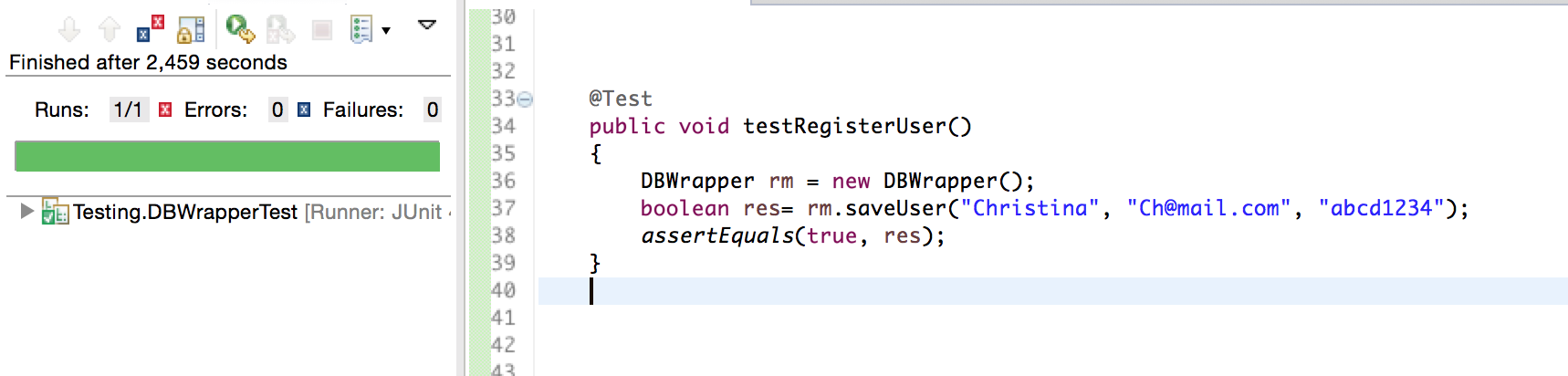
* User log in.

This test is testing if the user name and password is existing in the Mysql Database.



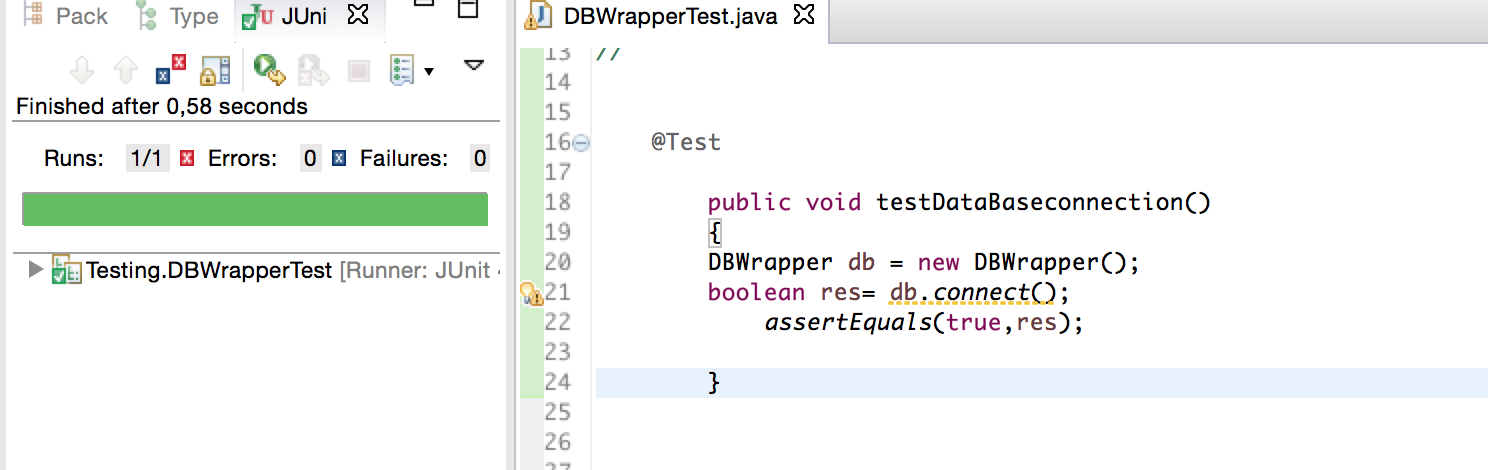
* Register User

Testing if the saveUser() method is saving the user with the given data.



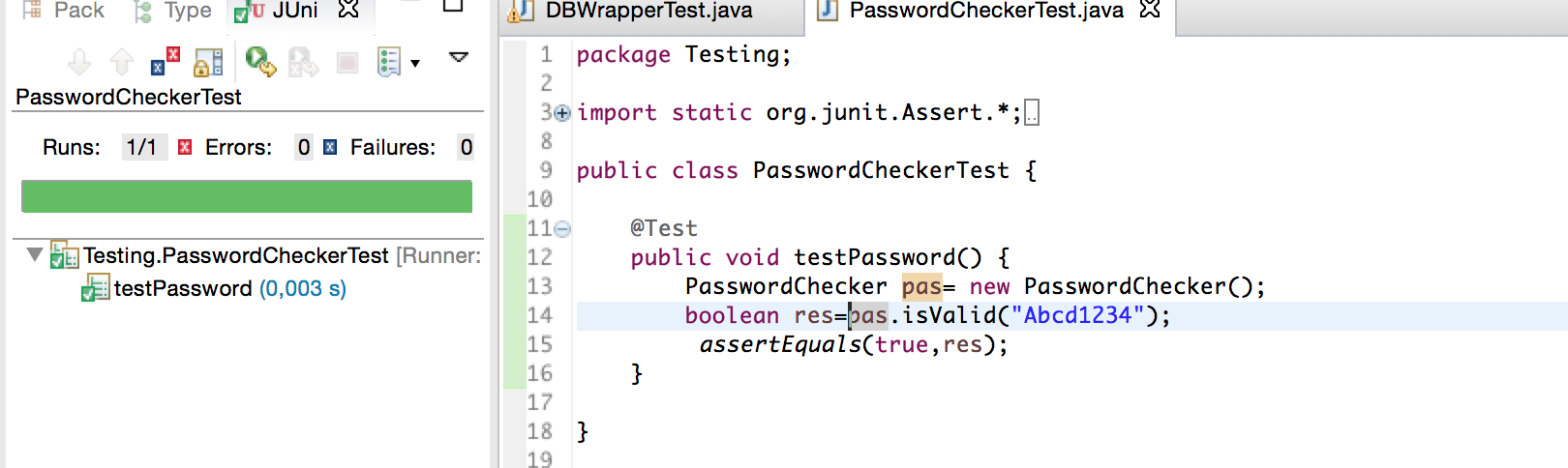
* Database connection.

Testing if the Database connection is established.



* Password validation.

Testing if the password is valid according on the requirements



* Email Validation

We have tested to cases the first one testEmailValidation() where the email is written right and valid, and the second one TestEmailValidation2() is where the email is not valid.

