Quizmaster

Assignment 5

Kai Arne S. Myklebust, Silvan Adrian

Handed in: October 18, 2018

Contents

1 Solution

1.1 Files

All Files are situated in the **src/** folder:

- **src/quizmaster.erl** The quizmaster Server implementation
- **src/quizmaster_helpers.erl** The greetings module implementation
- **tests/quizmaster_conductor.erl** Conductor Implementation for testing
- **tests/quizmaster_player.erl** Player implementation for testing
- **tests/quizmaster_tests.erl** Counter module implementation

1.2 Running the programm

Out of convenience we used a Emakefile which compiles all the erlang files in one go then rather compile each file on it's own. This can be done by using the erlang shell and run:

```
1 make:all([load]).
```

1.3 Running the tests

The tests are no eunit tests but rather a fe functions in quizmaster_tests.erl, which test the functionality of the quizmaster server by playing a game or other functionality.

2 Implementation

2.1 Gen-Statem

Since the Quizmaster can be seen as a simple State machine we chose gen_statem. The Quizmaster has overall 3 important states:

- editable
- between_questions
- active_question

2.2 Data Structure

Data with which we loop is a map with following entries:

- **conductor** The Pid of the Conductor (Gamemaker)
- **questions** all questions which belong to a quiz
- **players** all joined players (we chose to either have active or inactive players (left the game) to get through the tests of OnlineTA)
- **active_question** The index of the active question (in the questions list)
- **answered** Saving the first guess for each player, to be sure only one guess can be made for each question
- **distribution** the distribution of the current active question which shows which index has be chosen how many times

2.3 All states

Messages which get accepted in all states.

2.3.1 get_questions

Get all added questions.

2.4 Editable state

In the editable state the quiz can be modified, meaning new questions can be added, join is not allowed.

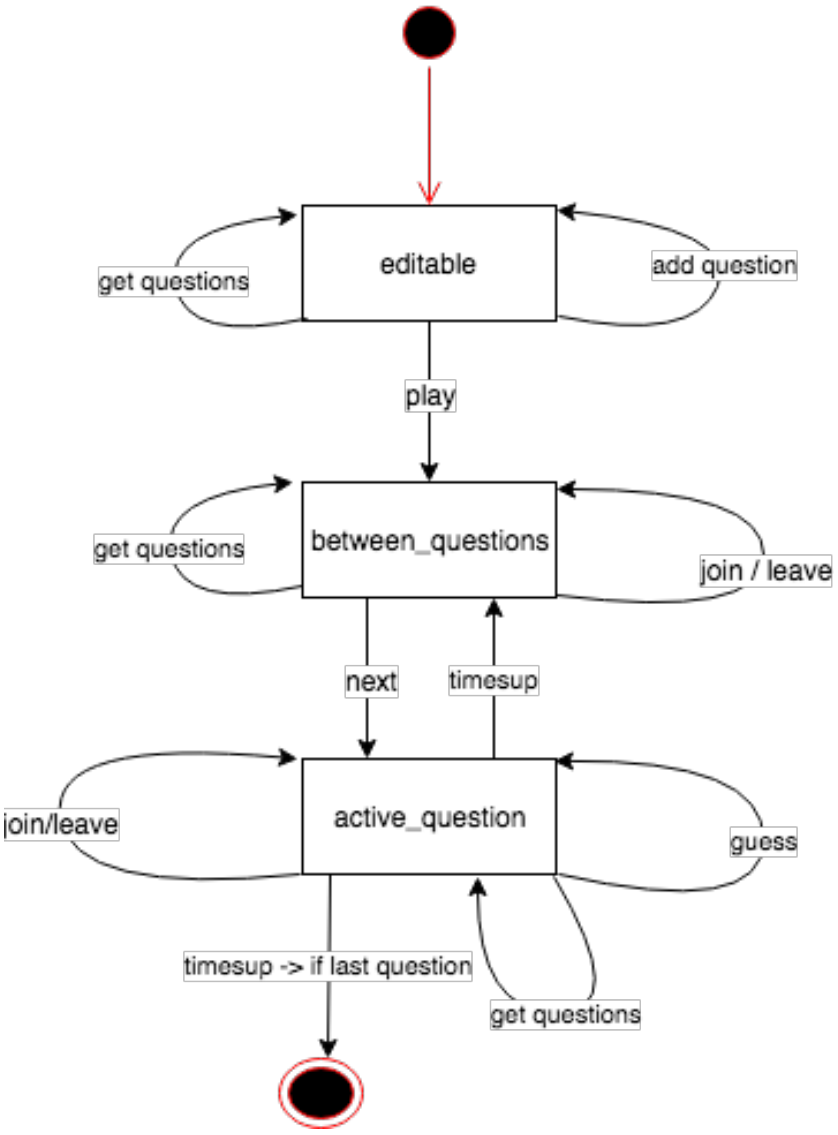

Figure 1: Simple drawing of the quizmaster state machine

```
[Description, [Answers]]
```

Figure 2: Question Format

```
{error, "Question is in wrong format"}
{error, "Wrong Format"}
```

Figure 3: Error Messages

2.4.1 add_question

So the message add_question adds a new question to the server state, with whom we loop further. The question does have to in the format:

In case the question doesn't fit the format we will get an error back that we tried to add a question in the wrong format, but we don't check if there is a correct answer added (so technically can add a question without any correct answers).

2.4.2 Play

When all Questions have been added a Quiz can be played, which means it changes it's state to between_questions and no more questions can be added. Additionally the Conductor gets set (any process which calls play first).

2.4.3 Other Messages

All other messages get ignored in the editable state.

2.5 Between_questions state

In between_questions new Players can join/leave and the next question can be set to active.

2.5.1 join

When joining we check if the nickname already exists, is this is the case then we set the existing player to active again otherwise add it as a new player to the players map in data. And send a message to the conductor by announcing a new player has joined.

2.5.2 leave

When a player leaves the nickname will still exist in the players map, only his status will be set to inactive so that the player still shows in the Report after timesup (this way the distribution still makes sense).

2.5.3 next

Select the next active question (according to the index in data), the state also changes to active_question. Next can be only called by the conductor, which gets checked against the one saved in data. Additionally the next question get broadcasted to all joined players, with the Description and the Answers (removing the correct).

2.5.4 timesup

Return a error in between_questions since there is no active question.

2.6 Active_question state

In Active question a player can join or leave, and the players can make guesses. The conductor then can run timesup to finish up the round and change back to between_questions state.

2.6.1 join

Same as in between_questions.

2.6.2 leave

Same as in between_questions.

2.6.3 guess

A guess can be made on a specific index which then either gives a point (if correct answer) or just gets counted into the distribution of the guesses. If the guess is on a wrong index then the guess is going to be ignored.

2.6.4 timesup

Timesup get called by the conductor and only by him, by doing that the state gets changed back to in between_questions and a reports gets sent out (with distribution, score of all players etc.). In case it was the last question then all players get a Message with quiz_over.

3 Assessment

3.1 OnlineTA

OnlineTA gave ok results back but only the last test wasn't able to fully run through and the end we ended up with guessing what exactly is wished what we implement to get through all test cases, so we gave up in the end.

3.2 Scope of Test Cases

3.3 Correctness

3.4 Code Quality

A Code Listing