Assignment 2

Group 20 Bejeweled

Exercise 1 Part 1

We chose to implement a game mode into Bejeweled (modeled after Bejeweled Blitz) where the player only has 60 seconds to reach a certain score. If the goal is reached, the level is incremented. Otherwise the game is lost.

The requirements for this implementation can be found on the following two pages.

Bejeweled Blitz requirements

Functional requirements

1.1 Must Haves

1. Upon starting a new Bejeweled game, a timer of 60 seconds and a score limit X (X will be specified later during the implementation process) will be initialized. We think this will add more fun to the game.
2. Upon successfully acquiring the required amount of points, the level will be incremented.
3. When the level gets incremented, a new field of jewels will be created on the board, the timer will be reset to 60 seconds and the score limit X will be increased by Y * Level. The game will start immediately.

1.2 Should Haves

1. If the timer runs out before the player achieves the required score, the player loses the game immediately.
2. There should not be a level limit. A game will end when acquiring the required score exceeds either the player's abilities or a human's capabilities in general.

1.3 Could Haves

1. The user will be able to pause the game by clicking a button, which will stop the timer and make the board invisible (to prevent cheating).
2. The game will start in a "paused" state and the user will be able to start it by clicking a button.

1.4 Won't Haves

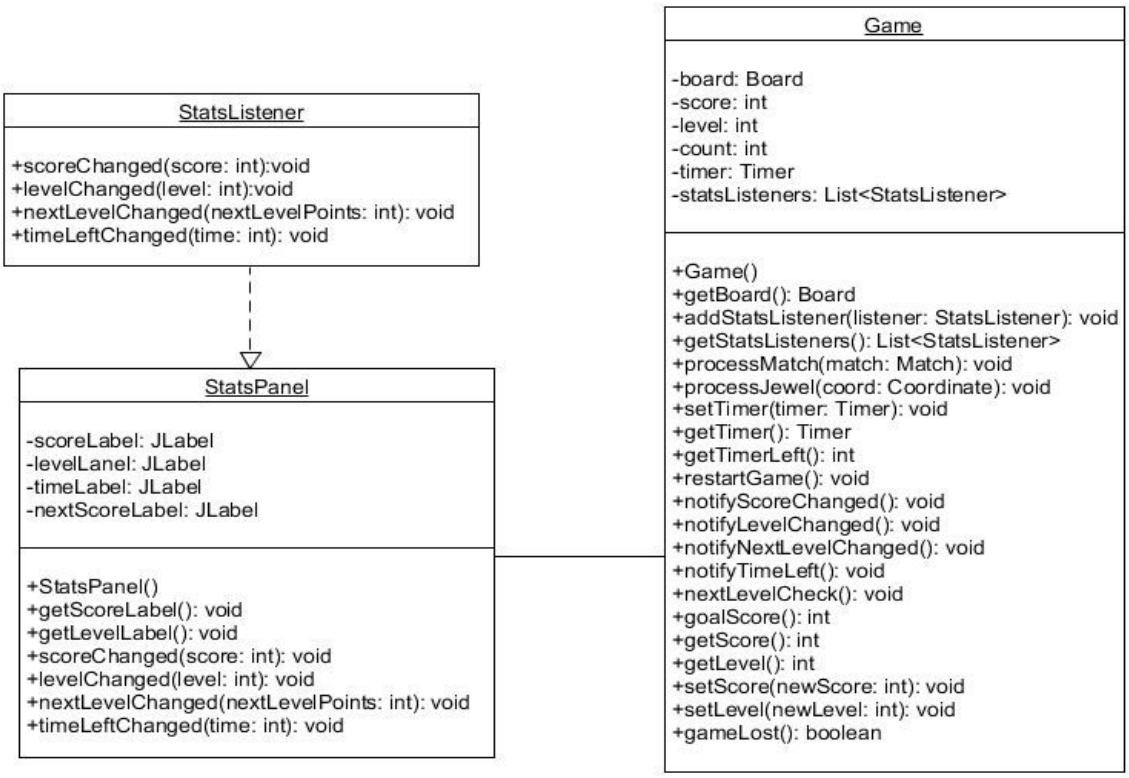
1. The user will not be able to choose other game modes besides this one.

Nonfunctional requirements

1. A fully functional version of the time/level will be delivered on 25-09-2015.
2. The time/level function will be developed using the Scrum software development methodology. (The development will take one Scrum iteration)
3. The time/level function will be developed using the Responsibility Driven Design technique.
4. The implementation of the time/level function shall have at least 75% of meaningful line test coverage. (where meaningful means that the tests actually test the functionalities of the time/level function and for example do not just execute the methods involved)

Exercise 1 Part 2

For responsibility driven design, see requirements as well. UML was made as follows:



Exercise 2

Our TA gave us the assignment to implement a “save game” feature, where a player would be able to quit a game and start again where he/she left off upon restarting the game. The specified requirements are on the page below.

Save game requirements

Functional requirements

1.1 Must Haves

1. When the user closes the game, its state must be automatically saved to a xml file. The state of a game includes:
    - The position of all the jewels.
    - The time left (A feature from exercise 1).
    - The level number.
    - The score.

2. When a user opens the game, the state which was saved upon closure of the game must automatically be read from the xml file and loaded.

1.2 Should Haves

1. No “Should Haves” were specified for this implementation.

1.3 Could Haves

1. The user is able to start a new game by pressing a “New Game” button.

1.4 Won’t Haves

1. The user won’t have the option to save the game ‘on the fly’ with a save game button or something like that.

Nonfunctional requirements

1. A fully functional version of the save function will be delivered on 25-09-2015.
2. The save function will be developed using the Scrum software development methodology. (The development will take one Scrum iteration)
3. The save function will be developed using the Responsibility Driven Design technique.
4. The implementation of the save function shall have at least 75% of meaningful line test coverage (where meaningful means that the tests actually test the functionalities of the savegame and for example do not just execute the methods involved)

The UML document for this feature is as follows: