**Assignment 6**

**Due, Sunday, July 23, 2017 for maximum 100**

**Monday, July 24, 2017 for maximum 90**

**Tuesday, July 25, 2017 for maximum 80**

**Wednesday, July 26, 2017 for maximum 70**

**Assignment Scope**

1. Writing Action or Item Listeners

**Deliverables**

To complete this assignment you must submit your **compressed Netbeans project** to Webcourses.

**Tasks and Rubric**

|  |  |  |
| --- | --- | --- |
| Activity | | |
| userInterface package |  |
| BoggleUi.java |  |
|  | 1. ~~Add member variables~~    1. ~~Primitive data type int for the player’s score~~ |
|  | 1. ~~Write an inner class to create an ActionListener that is registered to the JButton with text “Submit Word”; it should~~    1. ~~Validate if the word can be used based on the dictionary text file provided~~       1. ~~if it is not included in the dictionary notify the user the word was not valid and do not add it to the JTextPane~~       2. ~~if it is included in the dictionary, update the JTextPane by adding the word fromv the JLabel representing the Current Word~~    2. ~~Clear the JLabel representing the Current Word~~ |
|  | 1. ~~Write an inner class to create an ActionListener that is registered to the 16 JButtons that represent the dice on the board; when the JButton is clicked it should~~    1. ~~Update the JLabel representing the current word with the letter on the die and concatenate it to existing text~~    2. ~~Update the UI so only the available letters are enabled, all other letters are disabled based on the rules of Boggle; reference the PDF document for the rules of the game.~~ |
|  | 1. ~~Update the inner class that created the ActionListener for the javax.swing.Timer event handler; it should do the following:~~    1. ~~Stop the timer~~    2. ~~Randomly determines how many of the player’s word the computer found as well~~    3. ~~Randomly select which words of the player’s were found by the computer~~    4. ~~Strike through the words in the JTextPane~~    5. ~~Update the JLabel representing the player’s score for their final game score~~ |
| Boggle application |  |
| Test Case 1 | Test Case 1 passes |
| Test Case 2 | Test Case 2 passes |
| Test Case 3 | Test Case 3 passes |
| Test Case 4 | Test Case 4 passes |
|  | Source compiles with no errors |
|  | Source runs with no errors |
|  | Source includes comments |
| Total |  |

**Perform the following test cases**

|  |  |  |
| --- | --- | --- |
| Test Cases | | |
|  | **Action** | **Expected outcome** |
| Test case 1 | **User clicks letters on Boggle Board** | 1. Letter is displayed in the JLabel representing current word 2. Letters not eligible for use are disabled |
| Test case 2 | **User clicks the Submit Word button** | 1. Word is checked for validity 2. If valid, word is displayed in the Text Area 3. If not valid, is not added 4. JLabel of current word is cleared out 5. JLabel of Score is updated based on the rules |
| Test case 3 | **Time is 0** | 1. Game ends 2. Computer checks its words against the player 3. Same words are lined through in the player’s word list 4. Player’s score is updated to reflect final game score |

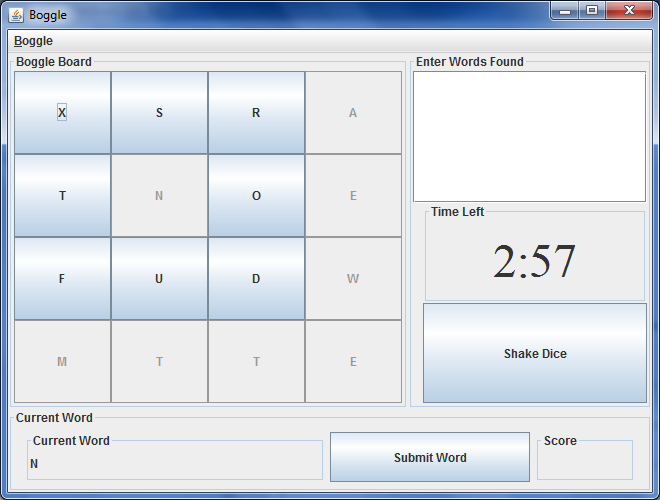


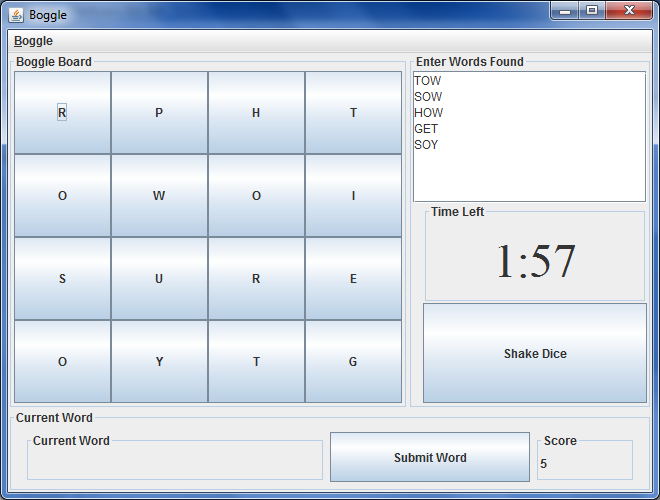
Figure 1 User Interface Display

Figure 2 Valid words

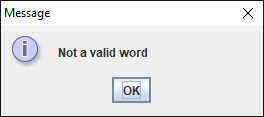


Figure 3 Invalid world

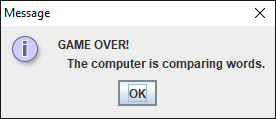


Figure 4 Time over

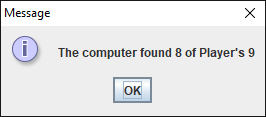


Figure 5 Computer’s words

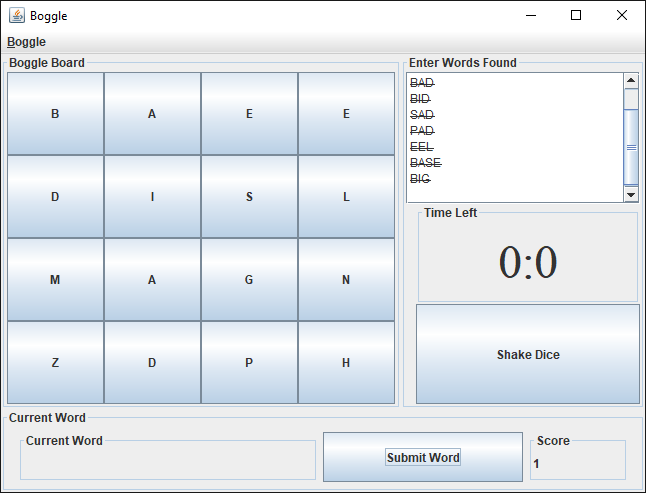


Figure 6 Final UI