

# C Sc 227 Final Project: Boggle

**Collaboration** This final project may be completed with one other registered Fall 2008 C Sc 227 Student in a pair programming mode (see Rick if you need to work solo).

## Due Date:

- Iteration 1: Tuesday 9-Dec 10:00pm (50pts)
- Iteration 2: Wednesday 10-Dec 5:00pm (50pts)

For this project you are asked to complete a computer version of Boggle.

## Boggle the Game

Boggle is a word game designed by Allan Turoff and trademarked by Parker Brothers and Hasbro. The game of Boggle has a container that holds 16 dice that can be "rolled". Each six-sided die has letters from which players try to find words. Words are possible if letters line up next to each other horizontally vertically or diagonally without being reused. There are 3 letters that can be connected to any letter on a corner 5 for the other six letters on the borders and 8 for the middle four letters (there is no wraparound). To make up one word no letter may be used more than once (this include the "Qu"). A word is accepted and the score increased if all of the following are true:



- The word made up from the dice follows the letter connection rules of Boggle
- The word exists in our official list of words: [BoggleWords](http://www.cs.arizona.edu/classes/cs227/fall108/projects/BoggleWords)  
<http://www.cs.arizona.edu/classes/cs227/fall108/projects/BoggleWords>
- The word contains 3 to 16 characters (one and two letter words are not allowed)
- The word is not already in the accepted list (no duplicate words allowed)

The player's score is determined by summing the points for all words found in a dictionary according to these Boggle rules.

Word Length	Points
3	1
4	1
5	2
6	3
7	5
8+	11

Simulate Boggle's dice rolls with the 16 dice. The same die must not always appear in the same location. Here are the letters of each of the 16 six-sided dice where each:

L R Y T T E	V T H R W E	E G H W N E	S E O T I S
A N A E E G	I D S Y T T	O A T T O W	M T O I C U
A F P K F S	X L D E R I	H C P O A S	E N S I E U
Y L D E V R	Z N R N H L	N M I H U Qu	O B B A O J

## Iteration 1

Dicetrays with public `DiceTray(char[][])` and `Boolean stringFound(String)`

Due Tuesday 9-Dec @ 10:00 via WebCat turnin

As with other projects you are asked to first develop a well-tested model. To get started immediately begin with a type named `DiceTray` that determines if a given `String` can be found in the collection of the current dice values using the rules of Boggle. `DiceTray` must have these 2 methods:

```
// Constructor takes a 2D array of characters that represents the
// Boggle DiceTray with 16 dice already rolled in a known fixed state.
public DiceTray(char[][] array)

// Return true if search is in the Boggle DiceTray according to Boggle rules.
// Note: This method does NOT check to see if the word is in the list of words.
public boolean stringFound(String search)
```

Here is a start of a unit test java that asserts some strings can be found that begins in the upper left corner. After you get these working you can change `stringFound` to try to find a word beginning at any of the 16 locations.

```
import static org.junit.Assert.*;
import org.junit.Test;
public class DiceTrayTest {

    private char[][] tray = {
        {'A' 'B' 'C' 'D' },
        {'E' 'F' 'G' 'H' },
        {'I' 'J' 'K' 'L' },
        {'M' 'N' 'O' 'P' }};

    @Test
    public void testStringFindWhenThereStartingInUpperLeftCorner() {
        DiceTray dt = new DiceTray(tray);
        assertTrue(dt.stringFound("ABC"));
        assertTrue(dt.stringFound("ABF"));
        assertTrue(dt.stringFound("ABC"));
        assertTrue(dt.stringFound("ABCD"));
        // ...
        assertTrue(dt.stringFound("ABFEJINM"));
        assertTrue(dt.stringFound("ABCDHGFEIJKLPONM"));
        assertTrue(dt.stringFound("ABCDHLPQJNMIEFG"));
    }

    @Test
    public void testStringFindWhenNotThere () {
        DiceTray dt = new DiceTray(tray);
        assertFalse(dt.stringFound("ACB"));
        assertFalse(dt.stringFound("AIE"));
        // ...
    }

    @Test
    public void testStringFindWhenAttemptIsMadeToUseALetterTwice () {
        DiceTray dt = new DiceTray(tray);
        assertFalse(dt.stringFound("ABA"));
        assertFalse(dt.stringFound("ABB"));
        assertFalse(dt.stringFound("AEA"));
        // ...
    }
}
```

## Grading Criteria Iteration 1 Due Tuesday 9-July: 50 pts

\_\_\_\_/ +50 For `DiceTray(char[][])` and `stringFound(String)` Graded by WebCat

## Iteration 2

Class DiceTray with a 2<sup>nd</sup> Constructor and toString plus Boggle.java (below)

Due Wednesday 10-Dec at 5:00 pm

The Dicetray is responsible for storing a set of characters that represent the showing sides of the 16 Boggle dice. It must have the following methods:

```
// Constructor takes a 2D array of characters that represents the
// Boggle DiceTray with 16 dice already rolled in a known fixed state.
public DiceTray(char[][] array)

// Return true if search is in the Boggle DiceTray according to the Rules.
public boolean stringFound(String search)

// Default constructor creates a random dice tray. It must simulate rolling
// of the actual 16 boggle dice where each die ends up in a random location and each die
// contains the letters that come with the avctual game (see page 1)
public DiceTray()

// Return the dicetray on 4 lines spaces between letters. Like this
// H A H I
//
// V Qu O S
//
// E E S E
//
// T I P W
public String toString()
```

## Class Boggle

Boggle should encapsulate all the Boggle rules and the list of words considered to be our dictionary. Therefore class Boggle has these instance variables:

```
private Dicetray dicetray;
private List<String> boggleWords;
```

Boggle must have these methods

```
// Initialize a game with 16 dice that mimic the real game with the actual dice.
public Boggle()

// Return the dicetray as a string.
public String getDiceTrayAsString();

// Added: Set Boggle to have a hard coded dice tray so this can be tested.
// @Test public void testGetWordsFoundAfterPrepareResultsCalledWithSetDiceTray() {
//     char[][] tray = { { 'H', 'A', 'H', 'I' },
//                       { 'V', 'W', 'O', 'S' },
//                       { 'E', 'E', 'S', 'E' },
//                       { 'T', 'I', 'P', 'W' } };

//     DiceTray dt = new DiceTray(tray);
//     Boggle game = new Boggle();
//     game.setDiceTray(dt);
//     ArrayList<String> userInput = new ArrayList<String>();
//     userInput.add("tip");
//     userInput.add("tips");
//     userInput.add("spit"); // add all other user guesses and prepareReport

public void setDiceTray(DiceTray dt) {
    this.diceTray = dt;
}
```

```

// Given the list of words entered by the user let methods getScore getWordsFound
// getWordsIncorrect getWordsNotGuesssed all return the correct values.
public void prepareResultsReport(List<String>)

// Score after the user quit.
public int getScore()

// All words the user entered that count for the score.
public Set<String> getWordsFound()

// All words the user entered that do not count for the score.
public Set<String> getWordsIncorrect()

// All words the user could have guessed but didn't
public Set<String> getWordsNotGuessed()

```

### Your Computer Version of Boggle

One version of Boggle will be console based with console IO using Scanner. By using the given design (set of methods) you will also be able to use the GUI provided by Rick for an event driven version that has a timer. After the user decides to quit show all words guessed correctly and the score. Also show all words not guessed that are in BoggleWords and the current "dice". These are the words the user did not find (and there are usually many). Your output should appear like this:

Play one game of Boggle:

```

H A H I
V W O S
E E S E
T I P W

```

Enter words or ZZ to quit:

```

tip tips spit see how sHoW notInTheBoard
tie tee tees shows spite
have have have
shave shaves
Aardvark ZymUrgy zZ

```

must be case insensitive

incorrect words have no point reduction

duplicates only count as one

Score: 19

You found these words correctly:

```

have how see shave shaves show shows spit spite tee
tees tie tip tips

```

show 10 per line max

Word you tried that were incorrect:

```

aardvark notinthetheboard zymurgy

```

Total number of word you could have listed: 68

```

awe awes epee eve eves ewe ewes hah haves haw
haws his hiss hoe hoes hos hose hoses hows ohs
owe owes pee peso pesos pet pew pews pie pies
pis pit psi set sew sews shah shoe shoes
sieve sip sit site sos sow sows spew ssh sweep

```

show 10 per line max

sweet ties tis veep veeps vet wave waves wee weep  
weeps wees wet who whoa whose woe woes

The above dialog indicates you must show the dice tray and ask the user to enter words found in the dice tray. When the user enters ZZ ((or zZ or Zz or zz) show the following information

1. The score
2. All words the user found in the dice tray that are also in our list of words (BoggleWords)
3. All incorrect words the user entered that were either not in the board or not in the list of words
4. All words the user missed

### **Grading Criteria Iteraton 2 Due Wednesday 10-July: 50 pts**

\_\_\_/ +20 Console Based Game produces output like that above

\_\_\_/ +30 WebCat turnin: Correctness and CodeCoverage

**Optional** Run your code with [BoggleGUI.java](#)

<http://www.cs.arizona.edu/classes/cs227/fall08/projects/BoggleGUI.java>