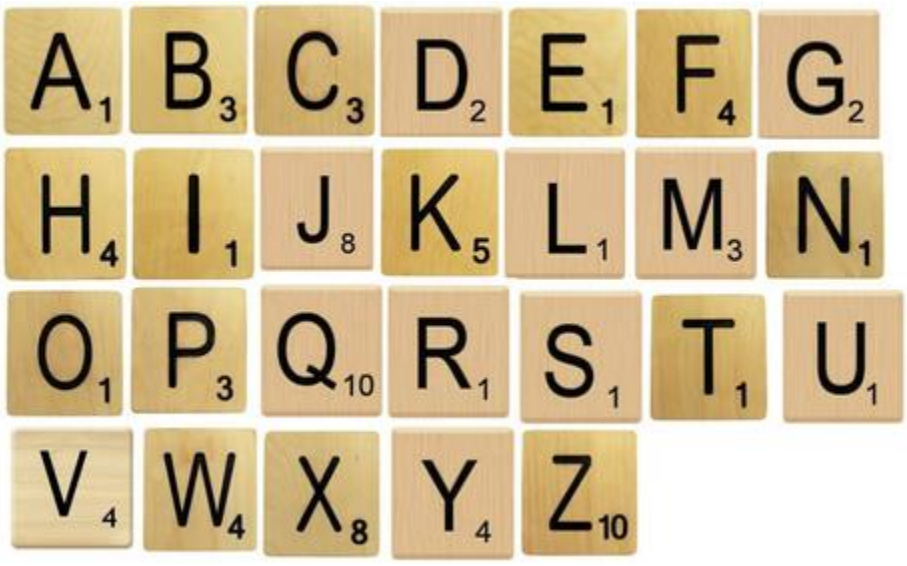
Learning Outcomes

- Getting used to programming in Java
- Overloading constructors
- Implementing equals(), toString(), getters, and other methods
- Working with Arrays and ArrayLists
- Using loops and conditionals

Introduction

Most of us are probably familiar with the beloved family game 'Scrabble'. In Scrabble, players collect seven random tiles, each consisting of a letter A through Z (for the sake of simplicity, let us that assume the 'blank' tile does not exist in this version of the game). Players then use these tiles to create words that consist of one or more of the letters they've randomly selected.

When a word is created using one or more of the seven selected tiles, the value of each of the tiles is summed, and that score is added to the player's cumulative score. The individual value of each tile is presented below.



In this assignment, you are provided with a text file consisting of all 279,496 words published in the 2019 version of the Collins Scrabble Word dictionary.

Given the letters of seven randomized Scrabble tiles, you must determine the set of scores that a player could possibly obtain by placing these tiles. We will be assuming three traits that differ slightly from the traditional game of Scrabble:

1. There is an unlimited amount of every letter within the scrabble bag, thus it is reasonable to assume (though incredibly unlikely) that you could obtain something like [‘A’, ‘A’, ‘A’, ‘A’, ‘A’, ‘A’, ‘A’].
2. You will be placing these tiles during the first turn on the board. Thus, you do not need to worry about combining your word with any pre-existing letters that might have already been placed on the Scrabble board.
3. Assume that there exist no word or letter bonuses. We will simply be basing the score off the tile values.

As an example, assume that your seven randomized tiles read:

{‘A’, ‘C’, ‘A’, ‘A’, ‘B’, ‘A’, ‘H’}

We are interested in determining:

1. A list of length n of words that can be created using these letters.
2. A sorted list of length n containing the scores (integers) for each of the words of length $1 \leq m \leq 7$ that can be created using these letters (like ‘AA’, or ‘AAH’, which are both words within the Collins Scrabble Word dictionary, surprisingly). So, two of the scores that will be in your score set for the above tile set are 2 (A (1) + A (1) = 2), and 6 (A (1) + A (1) + H (4) = 6).

Provided Files

The following is a list of files provided to you for this assignment. **Do NOT alter these files**.

- CollinsScrabbleWords2019.txt
- TestGame.java to test your solution
- Sample Tiles.java file with hints
- Sample Scrabble.java file with hints

Classes to Implement

For this assignment, you must implement two java classes: **Tile** and **Scrabble**. Follow the guidelines for each one below.

In these classes, you can implement more private (helper) methods if you want to. You may not, however implement more public methods.

You may not add instance variables other than the ones specified below nor change the variable types or accessibility (i.e. making a variable public when it should be private). Penalties will be applied if you implement additional instance variables or change the variable types of modifiers from what is described here.

Tile.java

This class represents a single Scrabble tile that will be used in the game.

The class must have the following **private** variables:

- letter (char)

The class must have the following **public** methods:

- public Tile() [constructor]
  - Set 'letter' to a random letter.
  - This constructor should call the private 'generateLetter' method provided in the Tile.java template.
- public Tile(char c) [constructor]
  - Initialize letter to the given argument
- public char getLetter() [getter]
  - Returns the value of the letter instance variable

The class has the following **private** methods (provided to you):

- private char generateLetter()
    - Generates a random char between A and Z (inclusive) and returns the char.
    - We use the 'Random' class to generate random numbers
    - random.nextInt(26) generates a random integers between 0 and 25 (inclusive)
    - Adding the generated random number to the ASCII value of 'A' (which is 65) maps the number to a corresponding uppercase letter between 'A' and 'Z'.

Scrabble.java

This class represents the Scrabble game in which there are seven randomly selected tiles, and scoring is performed for each possible word (this will be the tougher class to implement).

The class must have the following **private** variables:

- tiles (Tile[])

The class must have the following **public** methods:

- public Scrabble() [constructor]
    - Initialize the Tile array (tiles) with seven randomized Tile objects.
- public Scrabble(Tile[] t) [constructor]
    - Initialize the tile array with the given array of Tile objects passed as an argument to this constructor.
    - You should ensure that the array passed as an argument is of length seven. Otherwise, throw an IllegalArgumentException.
- public String getLetters()
    - Return a string that is comprised of the letters for all of the tile characters (for example, "ABFEODL")

- public ArrayList<String> getWords()
    - Create an ArrayList of Strings with n elements. Each element should represent a word that can be created using the current tiles.
    - The algorithm for this method should reference the provided file CollinsScrabbleWords2019.txt
    - ** do NOT put this file somewhere on your local machine and hardcode the local directory in your code. This will likely cause your tests to fail on GradeScope.
    - A portion of this method is provided to you to make things easier (as well as some hints in the comments to figure out a working algorithm).
- public int[] getScores()
    - Create an int array with n elements. Each element in this list should represent each individual score for each word that can be created using the current tiles. This should be returned in ascending order.
    - You should call the getWords method in this method!
- public Boolean equals(Scrabble s)
    - Compare the given Scrabble object from the argument with the 'this' object to see if they are equal (do they have the same tiles, assuming that the order of the tiles doesn't matter?).
    - Assuming we are comparing two Scrabble objects, s1 and s2, then if s1.getLetters() == "**AAAAAAB**" and s2.getLetters == "**BAAAAAA**", then s1.equals(s2) should be **true**.
    - Conversely, if s1.getLetters() == "**AAAAAAA**" and s2.getLetters == "**BAAAAAA**", then s1.equals(s2) should be **false**.

The class has the following **private** methods (provided to you):

- private int getLetterValue(char letter)
    - Returns the value of a letter (based on the figure on page 1 of this document).
    - To be used in the getScores() method.

Marking Notes

Functional Specifications

- Does the program behave according to specification?
- Does the program produce the correct output and pass all tests?
- Are the classes implemented properly?
- Does the code run on another machine (i.e., is anything hardcoded? It shouldn't be)
- Does the code produce compilation or runtime errors?
- Does the program follow all stated rules and match the specifications laid out in the 'Classes to Implement' section?

Non-Functional Specifications

- Are there comments throughout the code?
- Are variables within the methods given appropriate and meaningful names?
- Is the code clean and readable with proper indenting and white-space?
- Is the code consistent regarding formatting and naming conventions?
- Submission errors (i.e., missing files, too many files, etc.) will receive a penalty of 5%.
- Including a 'package' line at the top of a file will receive a penalty of 5%.

** Remember you must do all the work on your own. Do not copy or even look at the work of another student. All submitted code will be run through similarity detection software.

Rules

- Please only submit the files specified below.
- Do not upload the .class files! A 5% penalty will be applied for this.
- Submit your assignment on time. Late submissions will receive a penalty of 10% per day up to three days.
- Forgetting to submit is not a valid excuse for submitting late.

- Submissions must be done through Gradescope.
- Assignment files are not to be emailed to the instructor or TAs. They will not be marked if sent by email.

Testing your Code

The TestGame class tests the functionality of your methods (so you can see if your code is working properly). These tests are similar, but not identical, to the tests that will be used to grade your work. You do not need to understand all the code presented in the TestGame.java file. Just compile and execute it.

Test 1 and test 2 test the getLetters method

Test 3 and test 4 test the equals method

Test 5 and test 6 test the getWords method

Test 7 and test 8 test the getScores method

To test your code, compile and execute TestGame.java. If everything is working properly, you should see:

Test 1 Passed

Test 2 Passed

Test 3 Passed

Test 4 Passed

Test 5 Passed

Test 6 Passed

Test 7 Passed

Test 8 Passed

Otherwise, check while tests you are failing and that should give you an indication of where you should be focusing your attention.

Files to Submit

- Tile.java
- Scrabble.java

** Note that you do not need to submit the CollinsScrabbleWords2019.txt file.

Using the Tests

To use the TestGame.java file provided, place the TestGame.java file in the same location on your machine as your Scrabble.java, Tile.java, and CollinsScrabbleWords.txt files and compile and run the TestGame.java file. You should be passing all tests.

Grading Criteria

Total Marks [20]

Passing custom tests (similar but not identical to the ones provided) **[17]**

- There will be 8 tests of increasing value.

Following document specifications **[3]**

- Does your code comply with all 'non-functional specifications' discussed above?

Good luck, and have fun!