



# CS 315

## Software Design

### Homework 7

#### Find the Abstraction

#### Due: Nov. 5, 11:30 PM

### Objectives

- To get practice refactoring
- To build a class that defines a clean abstraction

### Assignment

An earlier homework assignment, the Anagram assignment, had you improve the style used in a working Java program. For this assignment, I want you to take those improvements further. In particular, I want you to focus your efforts on the Word class. Here is the Word class as it appears in the original assignment.

```
/**
 * Manage one word.
 * @author Peter van der Linden
 *
 */
public class Word {
    int count[] = new int[26]; // count of each letter in the word
    int total; // number of letters in the word
    String aword; // the word

    /**
     * Convert a string into the form convenient for further processing.
     * @param s
     */
    public Word(String s) { // construct an entry from a string
        int ch;
        aword = s;
        total = 0;
        s = s.toLowerCase();
        for (int i = 'a'; i <= 'z'; i++) count[i-'a'] = 0;

        for (int i = s.length()-1; i >= 0; i--) {
            ch = s.charAt(i) - 'a';
            if (ch >= 0 && ch < 26) {
                total++;
            }
        }
    }
}
```

```

        count[ch]++;
    }
}
}

```

This class does not provide a clean abstraction. In particular, there is no information hiding since all of the instance variables are accessible outside of the Word class.

In this assignment, you should clean up the Word class so that it provides a better abstraction. Callers should not need to understand how the Word class is implemented in order to be able to use it successfully. In particular, they should not need to know about the count array.

In addition, once you understand how a word is represented in this class, it should seem clear that there is a relationship between the values of the three instance variables. To ensure that a class with such relationships works properly, there should be a wellFormed method that checks the invariants, along with assert statements that call the wellFormed method to ensure that the invariants are never violated.

For this assignment, you should do the following:

- Study the implementation of the Word class to understand what its invariants are
- Define a wellFormed method to check the invariants. Add assert statements at appropriate places to check the invariants. (Remember to turn assertion checking on using the -ea VM argument when running your program.)
- Modify the instance variables to be private.
- Add new methods to the Word class that allow the rest of the program to get the information they need or to update the Word object without violating the Word abstraction or invalidating any of the Word invariants. Pay particular attention to hiding the count array.

To do this assignment, you can either use the original version of the program or the version you submitted in Homework 4 as your starting point.

## Grading

wellFormed method	15 points
assert statements	10 points
Providing a good Word abstraction	25 points
Style	10 points
Comments	10 points
Correctness	30 points

## Turning in your work

To turn in your work, create a jar file that contains your Anagram project, including the source code of all the classes and submit it on Ella.