Homework Turnin

Name:	Timothy J Ha
Email:	junkwan@uw.edu
Student ID:	1367917
Section:	BE
Course:	CSE 143 14sp
Assignment:	a6
Receipt ID:	8d11b073fca4470891b33a523d57d694

Turnin Successful!

The following file(s) were received:

Anagrams.java (3311 bytes)

```
// Timothy Ha
// 05.22.14 Spring
// CSE 143B, BE
// TA: Caitlin Schaefer
// Assignment #6: Anagrams
// Anagrams.java

// Finds all possible anagrams with a given phrase and an initial dictionary of words
// and prints them to the console with or without a maximum number of words restriction
// Also allows the user to get a list of words that the phrase is made up of

import java.util.*;

public class Anagrams {

   // word dictionary of all possible words
   private Set<String> dictionary;

   // an alphabetically sorted word bank with words
   // for possible anagrams
   private Set<String> wordbank;

   // Pre: the passed in dictionary is not null
   // otherwise, throw an IllegalArgumentException
   // Post: create an empty word bank
   // store the dictionary words
   public Anagrams(Set<String> dictionary) {
      if (dictionary == null) {
         throw new IllegalArgumentException();
      }
      wordbank = new TreeSet<String>();
      this.dictionary = new HashSet<String>(dictionary);
   }

   // Pre: phrase is not null
   // otherwise, throw an IllegalArgumentException
   // Post: stores and returns all possible words that can be made
   // with the letters of the passed in phrase
   public Set<String> getWords(String phrase) {
      if (phrase == null) {
```

```
            throw new IllegalArgumentException();
        }
        createWordbank(new LetterInventory(phrase));
        return wordbank;
    }

    // Pre: phrase is not null
    // otherwise, throw an IllegalArgumentException
    // Post: prints to the console all anagrams
    // created from the passed in phrase
    public void print(String phrase) {
        print(phrase, 0);
    }

    // Pre: phrase is not null AND max >= 0
    // otherwise, throw an IllegalArgumentException
    // Post: prints to the console every anagram created from
    // the passed in phrase with a restriction of max number of words per anagram
    public void print(String phrase, int max) {
        if (phrase == null || max < 0) {
            throw new IllegalArgumentException();
        }
        LetterInventory letters = new LetterInventory(phrase);
        if (wordbank.isEmpty()) {
            createWordbank(letters);
        }
        print(new Stack<String>(), max, letters);
        wordbank.clear();
    }

    // Post: prints to the console every anagram created from
    // the passed in letters with a restriction of max number of words per anagram
    private void print(Stack<String> anagram, int max, LetterInventory letters) {
        if (letters.isEmpty() && (max == 0 || anagram.size() <= max)) {
            System.out.println(anagram);
        } else {
            for (String word : wordbank) {
                if (letters.contains(word)) {
                    letters.subtract(word);
                    anagram.push(word);
                    print(anagram, max, letters);
                    letters.add(word);
                    anagram.pop();
                }
            }
        }
    }

    // Post: with the given letters, stores all possible words that
    // can be made up from the letters within the dictionary
    private void createWordbank(LetterInventory letters) {
        for (String word : dictionary) {
            if (word.length() <= letters.size()) {
                if (letters.contains(word)) {
                    wordbank.add(word);
                }
            }
        }
    }
}
```