```
/*
 * Class name:  Word.h
 * Class description:  Allows the use of strings as values for the BSTree binary search tree.
 *
 * Programmer:  Chad Philip Johnson
 * Date Created:  Friday, April 26th, 2013
 * Last Date Modified:  Thursday, May 09th, 2013
 *
 * Sources Used:
 *      N/A
 */

#include <string>
#include <iostream>

using namespace std;

#ifndef WORD_H
#define WORD_H

class Word
{
    public:
        /**
         * Default constructor for the Word class.  Set count to 1.
         */
        Word();

        /**
         * Overloaded constructor for the Word class.  Set count to 1 and assign the passed value to
         *      the string variable representing the word value of the object.
         * @param strWord The string value to be held by the object.
         */
        Word( string strWord );

        /**
         * Destructor for the Word class.  Currently unused.
         */
        virtual ~Word();

        /**
         * Overloaded output operator converts the word to uppercase (if not already uppercase) and feeds
         *      it to an output stream.
         * @param objostreamOut The output stream to receive the value.
         * @param objWordToOutput The word object to have its value sent to the output stream.
         */
        friend ostream& operator << ( ostream& objostreamOut, const Word*& objWordToOutput );

        /**
         * Overloaded equal comparison operator checks whether two Word objects contain the same values.
```

```
51          * @param objWordCompare The Word object value to be compared with the current object value.
52          */
53         bool operator == ( Word& objWordCompare );
54
55         /**
56          * Overloaded less than operator checks whether the right object value is greater than the current object value.
57          * @param objWordCompare The Word object value to be compared with the current object value.
58          */
59         bool operator < ( Word& objWordCompare );
60
61         /**
62          * Overloaded greater than operator checks whether the right object is less than the current object value.
63          * @param objWordCompare The Word object value to be compared with the current object value.
64          */
65         bool operator > ( Word& objWordCompare );
66
67         /**
68          * Return the value of the word contained within the Word object.
69          * @return String value of the word.
70          */
71         string          getWord() const;
72
73         /**
74          * Return the value of the word contained within the Word object (has the same use as the getWord function but
75          *      is a generic function name for Word class compatability with templated trees).
76          * @return String value of the word.
77          */
78         string          getValue() const;
79
80         /**
81          * Change the value of the word contained within the Word object.
82          * @param strWord The new value of the word.
83          */
84         void            setWord( string strWord );
85
86         /**
87          * Get the number of times the word value has been added to the tree.
88          * @return The unsigned int value of the number of times the word has been added to the tree.
89          */
90         unsigned int    getCount() const;
91
92         /**
93          * Increase the count by one (count represents the number of times the word value of the object has been added to a tree).
94          */
95         void            incrementCount();
96
97     private:
98         string strWord;
99         unsigned int uintCount;
100 };
```

```
#endif
```