```java
import java.io.IOException;

/************************************************************
* Dalton Nofs                                               *
* Login ID: nofs5491                                        *
* CS-102, Summer 2017                                       *
* Programming Assignment 4                                  *
* Term class: High level binary tree of terms for database  *
************************************************************/
public class Term implements TermInterface
{
    Tree<Course> courseTree;        // Tree of all the courses
    String term;                    // The term of the courses

    /************************************************************
    * Method: Term()                                            *
    * Purpose: default constructor for linkedList obj           *
    *                                                            *
    * Parameters:               N/A                             *
    * Returns: void:            N/A                             *
    ************************************************************/
    Term()
    {
        courseTree = new Tree<Course>(); // init an empty tree
        term = null;
    }
    /************************************************************
    * Method: Term()                                            *
    * Purpose: constructor passed a string                      *
    *                                                            *
    * Parameters:               N/A                             *
    * Returns: void:            N/A                             *
    ************************************************************/
    Term(String term)
    {
        courseTree = new Tree<Course>(); // init a empty tree
        this.term = term;
    }

    /************************************************************
    * Method: isEmpty()                                         *
    * Purpose: check to see if linkedList is empty              *
    *                                                            *
    * Parameters:               N/A                             *
    * Returns: boolean:         if list is empty                *
    ************************************************************/
    public boolean isEmpty()
    {
        if(courseTree.isEmpty())
            return true;
        return false;
    }

    /************************************************************
    * Method: getSearched()                                     *
    * Purpose: get object from linked list at index             *
    *                                                            *
    * Parameters:               N/A                             *
    * Returns: Object:          Object stored in index          *
    ************************************************************/
    public Course getSearched()
    {
        return(courseTree.getSearched().getDatum());
    }

    /************************************************************
    * Method: search()                                          *
```

```
68      * Purpose: searches tree for target                                *
69      *                                                                  *
70      * Parameters: T:         target                                    *
71      * Returns: boolean:      if found or not                        *
72      ********************************************************************/
73      public boolean search(Course target)
74      {
75          return courseTree.search(target);
76      }
77
78      /*******************************************************************
79      * Method: add()                                                     *
80      * Purpose: add a object at specified index                        *
81      *                                                                  *
82      * Parameters:                                                      *
83      *               Course:         Object to be placed               *
84      *                                                                  *
85      * Returns: void:                N/A                               *
86      ********************************************************************/
87      public void add(Course item) throws IOException
88      {
89          courseTree.add(item);
90      }
91
92      /*******************************************************************
93      * Method: remove()                                                    *
94      * Purpose: remove item from tree                                 *
95      *                                                                  *
96      * Parameters: Course:      course to remove                        *
97      * Returns: void:           nothing is returned                    *
98      ********************************************************************/
99      public void remove(Course item)
100     {
101         courseTree.remove(item);
102     }
103
104     /*******************************************************************
105     * Method: removeAll                                                 *
106     *                                                                 *
107     * Purpose: removes all nodes from array                            *
108     *                                                                  *
109     * Parameters:              N/A                                    *
110     * Returns: void:           N/A                                    *
111     ********************************************************************/
112     public void removeAll()
113     {
114         courseTree.removeAll();
115     }
116
117     /*******************************************************************
118     * Method: removeAll                                                 *
119     *                                                                 *
120     * Purpose: removes all nodes from array                            *
121     *                                                                  *
122     * Parameters:              N/A                                    *
123     * Returns: String           the term                               *
124     ********************************************************************/
125     public String getTerm()
126     {
127         return(this.term);
128     }
129
130     /*******************************************************************
131     * Method: getRoot()                                                 *
132     *                                                                 *
133     * Purpose: gets tree root node                                     *
134     *                                                                  *
135     * Parameters:              N/A                                    *
```

```
136     * Returns: TreeNode<T>:       the root node                      *
137     ************************************************************/
138     public TreeNode<Course> getRoot()
139     {
140         return(this.courseTree.getRoot());
141     }
142 }
```