```
// joi/4/dictionary/Dictionary.java
//
//
// Copyright 2003 Bill Campbell and Ethan Bolker

import java.util.*;

/**
 * Model a dictionary with a TreeMap of (word, Definition) pairs.
 *
 * @see Definition
 *
 * @version 4
 */

public class Dictionary
{
    private TreeMap entries;

    /**
     * Construct an empty Dictionary.
     */

    public Dictionary()
    {
        entries = new TreeMap();
    }

    /**
     * Add an entry to this Dictionary.
     *
     * @param word the word being defined.
     * @param definition the Definition of that word.
     */

    public void addEntry( String word, Definition definition )
    {
        entries.put( word, definition );
    }

    /**
     * Look up an entry in this Dictionary.
     *
     * @param word the word whose definition is sought
     * @return the Definition of that word, null if none.
     */

    public Definition getEntry( String word )
    {
        return (Definition)entries.get(word);
    }

    /**
     * Get the size of this Dictionary.
     *
     * @return the number of words.
```



```
     */

    public int getSize()
    {
        return entries.size();
    }

    /**
     * Construct a String representation of this Dictionary.
     *
     * @return a multiline String representation.
     */

    public String toString()
    {
        String str = "";
        String word;
        Definition definition;
        Set allWords = entries.keySet();
        Iterator wordIterator = allWords.iterator();
        while ( wordIterator.hasNext() ) {
            word = (String)wordIterator.next();
            definition = this.getEntry( word );
            str += word + ":\n" + definition.toString() + "\n";
        }
        return str;
    }
}
```

```
 1  // joi/4/dictionary/Definition.java
 2  //
 3  //
 4  // Copyright 2003 Bill Campbell and Ethan Bolker
 5
 6  /**
 7   * Model the definition of a word in a dictionary.
 8   *
 9   * @see Dictionary
10   *
11   * @version 4
12   */
13
14  public class Definition
15  {
16      private String definition;    // the defining string
17
18      /**
19       * Construct a simple Definition.
20       *
21       * @param definition the definition.
22       */
23
24      public Definition( String definition )
25      {
26          this.definition = definition;
27      }
28
29      /**
30       * Construct a String representation of this Definition.
31       *
32       * @return the definition string.
33       */
34
35      public String toString()
36      {
37          return definition;
38      }
39  }
```

```
// joi/4/dictionary/Lookup.java
//
//
// Copyright 2003 Bill Campbell and Ethan Bolker

/**
 * On line word lookup.
 *
* @see Dictionary
 * @see Definition
 *
 * @version 4
 */

public class Lookup
{
    private static Terminal t = new Terminal();
    private static Dictionary dictionary = new Dictionary();

    /**
     * Helper method to fill the dictionary with some simple
     * Definitions.
     *
     * A real Dictionary would live in a file somewhere.
     */

    private static void fillDictionary()
    {
        dictionary.addEntry( "shape",
            new Definition(  "a geometric object in a plane" ) );
        dictionary.addEntry( "quadrilateral",
            new Definition(  "a polygonal shape with four sides" ) );
        dictionary.addEntry( "rectangle",
            new Definition(  "a right-angled quadrilateral" ) );
        dictionary.addEntry( "square",
            new Definition(  "a rectangle having equal sides" ) );
    }

    /**
     * Helper method to print the Definition of a single word,
     * or a message if the word is not in the Dictionary.
     *
     * @param word the word whose definition is wanted.
     */

    private static void printDefinition(String word)
    {
        Definition definition = dictionary.getEntry(word);
        if (definition == null) {
            t.println("sorry, no definition found for " + word);
        }
        else {
            t.println(definition.toString());
        }
    }

```



```
    /**
     * Run the Dictionary lookup.
     *
     * Parse command line arguments for words to look up,
     * "all" prints the whole Dictionary.
     *
     * Then prompt for more words, "quit" to finish.
     *
     * For example,
     * <pre>
     *
     * %> java Lookup shape square circle
     * shape:
     * a geometric object in a plane
     * square:
     * a rectangle having equal sides
     * circle:
     * sorry, no definition found for circle
     *
     * look up words, "quit" to quit
     * word> rectangle
     * a right-angled quadrilateral
     * word> quit
     * %>
     * </pre>
     *
     * @param args the words that we want looked up, supplied as
     *             command line arguments.  If the word "all" is
     *             included, all words are looked up.
     */

    public static void main( String[] args )
    {
        // fill the dictionary (not a big one!)
        fillDictionary();

        // look up some words
        String word;

        // words specified on command line
        for (int i = 0; i < args.length; i++ ) {
            word = args[i];
            if (word.equals("all")) {
                t.println("The whole dictionary (" +
                          dictionary.getSize() + " entries):");
                t.println("-------------------");
                t.println(dictionary.toString());
                t.println("-------------------");
            }
            else {
                t.println(word + ":");
                printDefinition(word);
            }
        }

        // words entered interactively
```

```
   113          t.println("\nlook up words, \"quit\" to quit");
   114          while (true) {
   115              word = t.readWord("word> ");
   116              if (word.equals("quit")) {
   117                  break;
   118              }
   119              printDefinition(word);
   120          }
   121      }
   122  }
```