```java
package examples;

/**
 * This class represents a single sentence. It is based on an example from
 * section 13.2 of Horstmann's Big Java, 3rd ed.
 *
 * @author Cay Horstmann
 */
public class Sentence {

    private final String text;

    /**
     * Creates a sentence object for the given string.
     *
     * @param text
     */
    public Sentence(String text) {
        this.text = text;
    }

    /**
     * Main entry point for example.
     *
     * @param args
     *            ignored
     */
    public static void main(String[] args) {
        String str = "Go hang a salami, I'm a lasagna hog.";
        Sentence sent = new Sentence(str);
        System.out.println(sent.isPalindrome());
    }

    /**
     * Checks whether this sentence is a palindrome. Palindromic sentences are
     * considered to be those the read the same forward or backward, ignoring
     * case, punctuation, and spaces.
     *
     * Examples:
     *
     * new Sentence("deified");
     *
     * new Sentence("I prefer Pi");
     *
     * new Sentence("A man, a plan, a canal -- Panama!");
     *
     * new Sentence("Madam, I'm Adam");
     *
     * new Sentence("Go hang a salami, I'm a lasagna hog.");
     *
     * @return true iff this sentence is a palindrome
     */
    public boolean isPalindrome() {
        // FIXME: delete body for template
        int length = this.text.length();

```

```
57          if (length <= 1) {
58              return true;
59          }
60          // Checks first and last
61          char first = Character.toLowerCase(this.text.charAt(0));
62          char last = Character.toLowerCase(this.text.charAt(length - 1));
63          if (!Character.isLetter(first)) {
64              Sentence shorter = new Sentence(this.text.substring(1));
65              return shorter.isPalindrome();
66          }
67          if (!Character.isLetter(last)) {
68              Sentence shorter = new Sentence(this.text.substring(0, length - 1));
69              return shorter.isPalindrome();
70          }
71          // first and last are both letters
72          if (first == last) {
73              Sentence shorter = new Sentence(this.text.substring(1, length - 1));
74              return shorter.isPalindrome();
75          } else {
76              return false;
77          }
78      }
79
80      // FIXME: delete for template
81      /**
82       * Checks whether the substring of this.text between start and end,
83       * inclusive, is a palindrome.
84       *
85       * @param start
86       * @param end
87       *
88       * @return true iff the substring between start and end is a palindrome
89       */
90      public boolean isPalindrome(int start, int end) {
91          if (end - start <= 1) {
92              return true;
93          }
94
95          // If start character is not a letter, throw it out
96          char firstChar = this.text.charAt(start);
97          if (!Character.isLetter(firstChar)) {
98              return isPalindrome(start + 1, end);
99          }
100
101         // If end character is not a letter, throw it out
102         char lastChar = this.text.charAt(end);
103         if (!Character.isLetter(lastChar)) {
104             return isPalindrome(start, end - 1);
105         }
106
107         // Both start and end characters are letters, if they match, throw both
108         // out
109         if (Character.toLowerCase(firstChar) == Character.toLowerCase(lastChar)) {
110             return isPalindrome(start + 1, end - 1);
111         }
112
```

```
113         // Both start and end characters were letters and they didn't match
114         return false;
115     }
116 
117     /**
118      * @return a NEW sentence object whose text is the reverse of this one
119      */
120     public Sentence reverse() {
121         /*
122          * TODO: implement and JUnit test this method. Your solution must be
123          * recursive.
124          */
125         return new Sentence(reverseHelper(0));
126     }
127 
128     // FIXME: delete for template
129     /**
130      * @param i
131      * @return the reverse of this.text.substring(i)
132      */
133     private String reverseHelper(int i) {
134         if (i == this.text.length())
135             return "";
136         return reverseHelper(i + 1) + this.text.charAt(i);
137     }
138 
139     @Override
140     public String toString() {
141         return this.text;
142     }
143 
144 }
145 
```