Homework Turnin

Name:	Xuqing Wu
Account:	xw88 (xw88@uw.edu)
Student ID:	1933202
Section:	AD
Course:	CSE 143 20wi
Assignment:	a7
Receipt ID:	dcffdbaea954e6fa7048dd68342de0c0

Warning: Your turnin is 3 days late. Assignment a7 was due Thursday, March 5, 2020, 9:00 PM.

Turnin script completed with output:

Turnin Successful!

The following file(s) were received:

QuestionNode.java (677 bytes, sha256: fdd48d1a70420e2a43a805569bcfe917)

```
 1. // Xuqing Wu
 2. // 3/5/2020
 3. // CSE143
 4. // TA: Eric Fan
 5. // Assignment #7 part 1
 6. //
 7. // Class QuestionNode for storing a single node of a binary tree of Strings
 8.
 9. public class QuestionNode {
10.     public String data;
11.     public QuestionNode left;
12.     public QuestionNode right;
13.
14.     // constructs a leaf node with given data
15.     public QuestionNode(String data) {
16.         this(data, null, null);
17.     }
18.
19.     // constructs a branch node with given data, left subtree, right subtree
20.     public QuestionNode(String data, QuestionNode left, QuestionNode right) {
21.         this.data = data;
22.         this.left = left;
23.         this.right = right;
24.     }
25. }
```

QuestionTree.java (4688 bytes, sha256: e24061a30dbec087f92103e05e67c1ab)

```
 1. // Xuqing Wu
 2. // 3/5/2020
```

```java
// CSE143
// TA: Eric Fan
// Assignment #7 part 2
//
// Class QuestionTree allows client to play a yes/no guessing game.
// The class constructs a binary tree that distinguishes between
// the objects. The computer will try to guess the object clients
// think by asking a series of yes or no questions. It updates the
// tree when it asks a question. Eventually the computer will have
// asked enough questions that it thinks it knows what object clients
// are thinking of. It will make a guess. If this guess is correct, the
// computer wins; if not, clients win.


import java.util.*;
import java.io.*;

public class QuestionTree {
   private QuestionNode overallRoot;
   // binary tree which stores questions and answers
   private Scanner console;
   // scanner to get the response of user

   // post: Construct a binary tree with one leaf node representing computer
   public QuestionTree() {
      overallRoot = new QuestionNode("computer");
      console = new Scanner(System.in);
   }

   // post: asks the user a question, forcing an answer of "y " or "n";
   // returns true if the answer was yes, returns false otherwise
   public boolean yesTo(String prompt) {
      System.out.print(prompt + " (y/n)? ");
      String response = console.nextLine().trim().toLowerCase();
      while (!response.equals("y") && !response.equals("n")) {
         System.out.println("Please answer y or n.");
         System.out.print(prompt + " (y/n)? ");
         response = console.nextLine().trim().toLowerCase();
      }
      return response.equals("y");
   }

   // post: replace the current tree by reading another tree from a file.
   // A Scanner that link to the file is passed.
   public void read(Scanner input) {
      overallRoot = readHelper(input);
   }

   // post: the helper method that read information from scanner passed
   // and construct a new tree
   private QuestionNode readHelper(Scanner input) {
      QuestionNode root = null;
      if(input.hasNextLine()) {
         String type = input.nextLine();
         String qOrA = input.nextLine();
         if(type.contains("Q:")) {
            root = new QuestionNode(qOrA, readHelper(input),
               readHelper(input));
         }
         else {
            root = new QuestionNode(qOrA);
         }
      }
      return root;
   }

   // post: store the current tree to an output file using passed PrintStream
   public void write(PrintStream output) {
      write(output, overallRoot);
   }

   // post: helper method that store the current tree to an output file
   private void write(PrintStream output, QuestionNode root) {
      if(root != null) {
         if(root.data.contains("?")) {
            output.println("Q: ");
```

```
79.            }
80.            else {
81.               output.println("A: ");
82.            }
83.            output.println(root.data);
84.            write(output, root.left);
85.            write(output, root.right);
86.         }
87.      }
88.
89.      // post: use the current tree to ask the user a series of yes/no questions
90.      // until computer either guess their object correctly or until fail,
91.      // in which case expand the tree to include their object and a new question
92.      // to distinguish their object from the others
93.      public void askQuestions() {
94.         overallRoot = askQuestions(overallRoot);
95.      }
96.
97.      // post: the helper method asks user questions and get response from
98.      // users and update the tree with qustions and answers from user when it
99.      // fails to guess the object
100.     private QuestionNode askQuestions(QuestionNode root) {
101.        if(root.left == null || root.right == null) {
102.           if(yesTo("Would your object happen to be " + root.data + "?")) {
103.              System.out.println("Great, I got it right!");
104.           }
105.           else {
106.              System.out.print("What is the name of your object? ");
107.              String animal = console.nextLine();
108.              QuestionNode curr = new QuestionNode(animal);
109.              System.out.println("Please give me a yes/no question that");
110.              System.out.println("distinguishes between your object");
111.              System.out.print("and mine--> ");
112.              String question = console.nextLine();
113.              if(yesTo("And what is the answer for your object?")) {
114.                 root = new QuestionNode(question, curr, root);
115.              }
116.              else {
117.                 root = new QuestionNode(question, root, curr);
118.              }
119.           }
120.        }
121.        else {
122.           if(yesTo(root.data)) {
123.              root.left = askQuestions(root.left);
124.           }
125.           else {
126.              root.right = askQuestions(root.right);
127.           }
128.        }
129.        return root;
130.     }
131.  }
132.
```