Questions

Your name:

By the time you are done with this activity, you should be able to:

- use recursive algorithms to interact with binary trees.
- perform complex reference (pointer) manipulations.

After you complete this activity, please fill out the short survey at

http://goo.gl/forms/HXjyuUb2ou

to improve this project for future users.

Playing the game

Included in the Questions project is a file Questions.jar. This is a compiled version of the working game. To play it, use a terminal to navigate to the directory containing the file and type this on the command line:

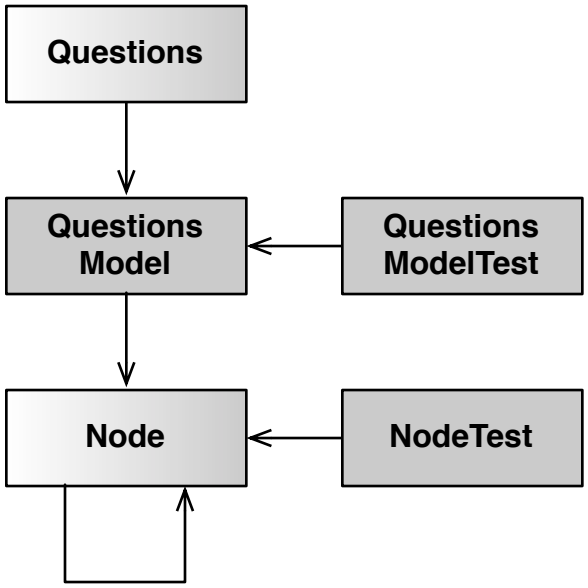
```
java -jar Questions.jar
```

Questions is essentially the parlor game *20 Questions*, but (for simplicity) without a limit on the number of questions. The user thinks of something and the program tries to figure out what it is. By being given explanations when it fails, the program becomes more capable with repeated plays.

This is a one-player game. Play a few times to get a sense of how the game works. After your copy of the program has built up a body of knowledge, try switching computers with another student to play against a program that someone else has "trained". Be careful not to close the program, though; the program does not remember things it has learned in previous sessions.

Overview

The UML class diagram below shows the relationships between classes. The shaded classes have been provided for you; you should not alter them. The others are partially shaded because you have been given incomplete skeletons of these classes.



Implementation

Looking at the UML class diagram, decide in which order to tackle the missing and incomplete classes. For each one with a test, work through the tests *in the order in which they appear in the test class*.

Take notes as you work. What bugs and conceptual difficulties did you encounter? How did you overcome them? What did you learn?

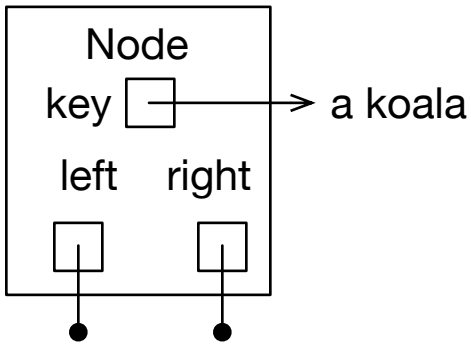
Students tend to find three parts of this project particularly challenging:

`toString` (Node)

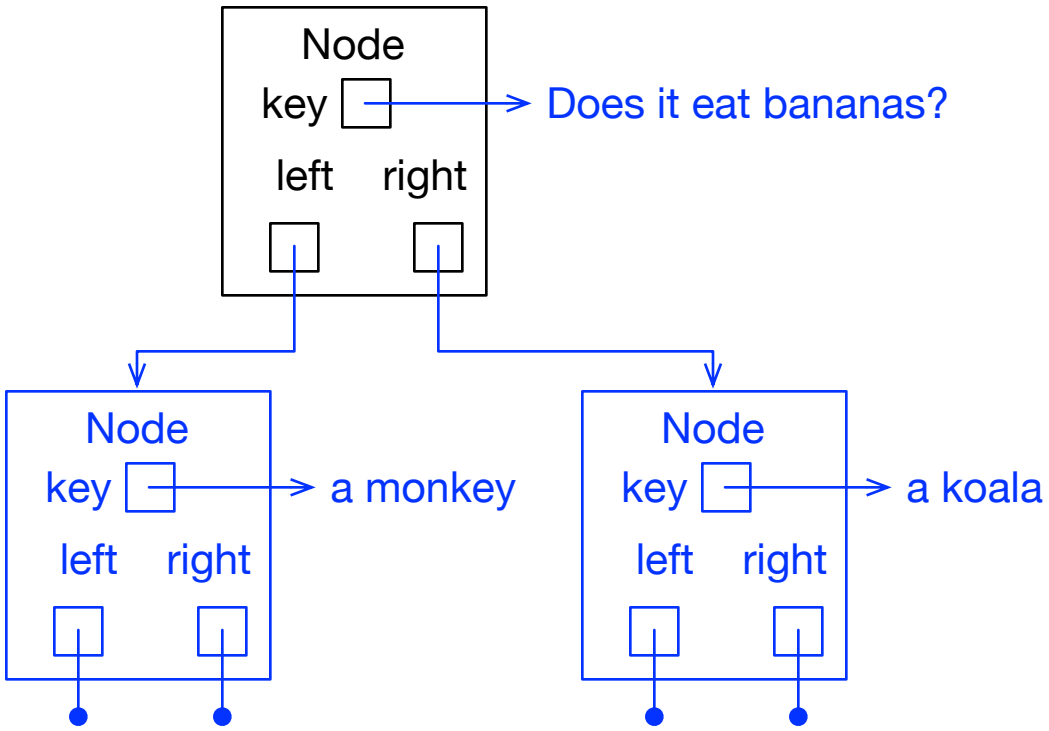
Notice that there are two overloaded versions of `toString`. The first, with the usual signature, is a trivial method that simply calls the other version. The second version does all the work. Read the comment carefully. What will your recursive calls look like?

`learn` (Node)

Suppose `this` is the node shown below, `correct` is `"a monkey"`, and `question` is `"Does it eat bananas?"`.



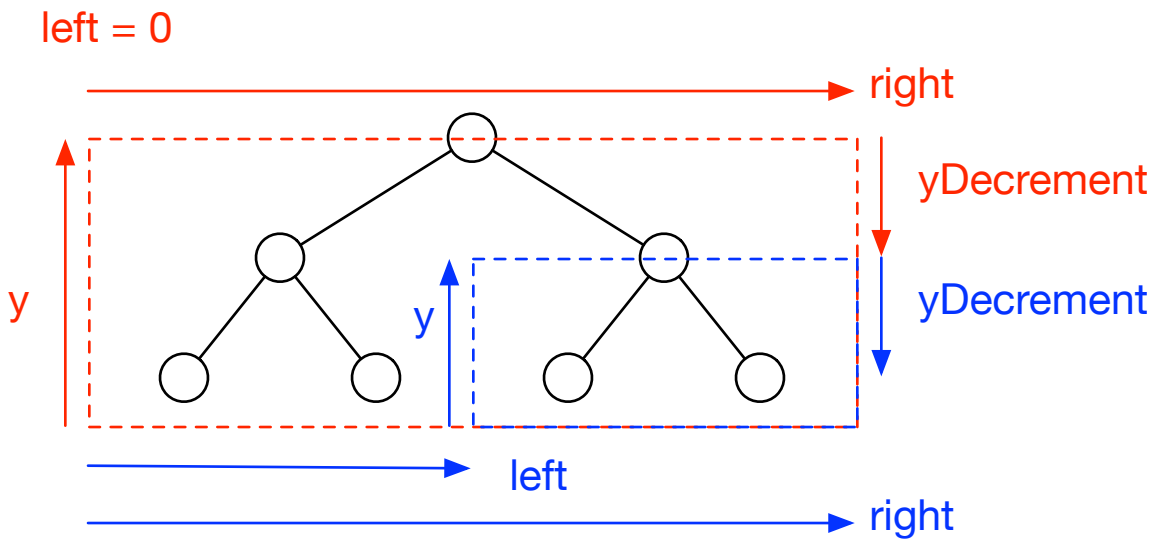
After you're done, `this` should look like the diagram below.



As you debug, drawing diagrams of intervening states of your data structure may be very useful.

`drawSubtree` (Questions)

The diagram below may help you figure out what arguments to pass to your recursive calls.



Be sure to play the game (by running Questions.java) a few times after you're done to make sure the entire system is working correctly.

If you're feeling really ambitious, you might look into making the program remember what it has learned between sessions. Reading about the Serializable interface could help with this (optional!) improvement.

Please fill out the survey at http://goo.gl/forms/HXjyuUb2ou.

This work was supported by the Google Education and Relations Fund's CS Engagement Small Grant Program grant #TFR15-00411.

The principal investigator, Peter Drake, wishes to thank the following for their useful comments: the members of the CS-POGIL project, specifically Clif Kussmaul and Helen Hu; Maggie Dreyer; and various anonymous reviewers.