CSE 223 Spring 2019 Programming Assignment #4

For this assignment, you'll be writing code to implement, use and extend a decision tree. This is similar in spirit to the "20 Questions" game (though we're not restricting this to 20 questions).

This assignment will be discussed in detail in class, but here are some additional details:

- you'll need to implement a binary tree, where each node contains 2 pieces of data: a string of text, and a flag indicating whether this is a question or an answer;

- To play the game, begin at the root (it will be a question), and ask the question;

- based on the user's response, travel to the left (YES) or right (NO) sub-tree, and repeat, until you reach an answer;

- if the answer is correct, no action is necessary;

- if the answer is wrong, ask for the answer and a new question to discriminate the correct answer from the incorrect one you guessed; then make a new node from that questions and those two answers, and insert it into the tree in place of the incorrect answer.

Additionally, the initial decision tree should be read from a file, and, after being modified, re-written to the same file. **The name of this database should be modifiable by specifying a single command-line argument**. Remember to traverse in PRE-ORDER (NLR) to simplify the reading/writing of this file.

Here's a sample initial database file (between the ------------------ lines):

```
------------------------
Q:
Is it an animal
A:
Pig
A:
Rock
-----------------------
```
(this file should have 6 lines total, with no empty lines).

Additionally:

- make sure your code is well-commented;

- feel free to embellish your project using Swing, etc. At a minimum though, it should play the game, enhance the decision tree if it guesses wrong, and read/write the decision tree to disk.

As always, I strongly recommend approaching this in pieces: make a binary tree class, then extend it to support this decision tree; then hard-wire some nodes and get it to traverse/play the game. Work on the File I/O separately; work on extending the tree separately; and so on.

You may use built-in primitives from Java such as Scanner, PrintWriter, etc. but create your own tree class, your own file parser, your own traversal code, and so on.