# Game of Twenty Questions

QuestionsGame.java – A class that represents a tree of questions and answers

You will need the support files QuestionMain.java, spec-questions.txt, and big-questions.txt; place these in the same folder as your program or project. You should not modify the provided files. Your code must work properly with the unmodified versions of the provided files.

# The Game of Twenty Questions

20 Questions is a guessing game in which the objective is to ask yes/no questions to determine an object. In our version, the human begins a round by choosing some object, and the computer attempts to guess that object by asking a series of yes/no questions until it thinks it knows the answer. Then, the computer makes a guess; if its guess is correct, the computer wins, and otherwise you win. If the computer loses, it will add the object you were thinking of to its knowledge base.

In this assignment, you will create a class named QuestionsGame to represent the computer’s tree of yes/no questions and answers for playing games of 20 Questions. Your QuestionsGame class **MUST** contain a private static inner class named QuestionNode.

You are provided with a client QuestionMain.java that handles user interaction and calls your tree’s methods to play the games.

# QuestionNode

The contents of the QuestionNode class are up to you. Though we have studied trees of ints, you *should* create nodes specific to solving this problem. Your QuestionNode class should have at least one constructor used by your tree. Don’t include constructors that are not actually used in your program. Your QuestionNode class must be private static inner class within QuestionsGame. Your node’s fields *must* be public. QuestionNode should not contain *any actual game logic*. It should only represent a single node of the tree.

# QuestionsGame

This class represents a game of 20 Questions. It keeps track of a binary tree whose nodes represent questions and answers. (Every node’s data is a string representing the text of the question or answer.) Note that even though the name of the game is “20 questions”, the computer will *not* be limited to only *twenty* ; the tree may have a larger height.

The *leaves* of the tree represent possible answers (guesses) that the computer might make. All the other nodes represent questions that the computer will ask to narrow the possibilities. The left branch indicates the next question the computer asks if the answer is *yes*, and the right branch is the next question if the answer is *no*. The game is played by starting at the root and asking questions for each of the branch nodes, going down the the tree based on the user’s answer. Once a leaf node is reached, the computer will ask if that answer is the correct one. Page 3 walks through a full example of a game.

public **QuestionsGame**()

This constructor should initialize a new QuestionsGame object with a *single leaf node* representing the object “computer”.

public void **read**(Scanner input)

This method will be called if the client wants to replace the current tree by reading another tree from a file. Your method will be passed a Scanner that is linked to the file and should replace the current tree with a new tree using the information in the file. Assume the file is legal and in standard format. Make sure to read entire lines of input using calls on Scanner’s nextLine.

public void **write**(PrintStream output)

This method should store the current questions tree to an output file represented by the given PrintStream. This method can be used to later play another game with the computer using questions from this one.

public void **askQuestions**()

This method should use the current question tree to play one complete guessing game with the user, asking yes/no questions until reaching an answer object to guess. A game begins with the root node of the tree and ends upon reaching an answer leaf node.

**If the computer wins the game**, this method should print a message saying so.

**Otherwise**, this method should ask the user for the following:

* what object they were thinking of,
* a question to distinguish that object from the player’s guess, and
* whether the player’s object is the yes or no answer for that question.

public boolean **yesTo**(String prompt)

This method asks the given question until the user types “y” or “n”. Returns true if “y”, false if “n”.

**User Input: Yes and No**

At various points in this assignment, you will need to get a yes or no answer from the user.

The details of for asking the user to input yes/no are fairly uninteresting, so it is being provided for you as a method called yesTo (the code assumes there is a data field called console has been initialized). You are to include this method without modification to your QuestionsGame class and use it whenever possible to read user input. **The code for yesTo can be found in the starter code on the homework page of the course website**

# Question Tree “Standard Format”

In read and write, your class will be interacting with files containing questions. Just like with BNF where we used a common format so everyone could share the same grammars, here, we specify a common question tree format.

A single QuestionNode should be represented as a non-empty sequence of line pairs. The first line of the pair should contain either “Q:” or “A:” to differentiate between questions (branches) and answers (leaves). The second line of the pair should contain the text for that node (the *actual* question or answer).

The nodes of the tree should appear in *pre-order*. The readTree and writeTree methods from section will be *very helpful* for any method that deals with this format.

# Full Example Walk-Through

Q:

questions.txt

Is it an animal? Q:

Can it fly? A:

bird Q:

Does it have a tail? A:

mouse A:

spider Q:

Does it have wheels? A:

bicycle Q:

Is it nice? A:

TA A:

teacher

Is it an animal?

yes

no

Can it fly?

Does it have wheels?

yes

no

yes

no

bird

Does it have a tail?

bicycle

Is it nice?

yes

no

yes

no

mouse

spider

TA

teacher

The following output log shows one game being played on the above tree:

>> Welcome to the cse143 question program.

>>

>> Do you want to read in the previous tree? (y/n)? y

>>

>> Please think of an object for me to guess.

>> Is it an animal? (y/n)? n

>> Does it have wheels? (y/n)? y

>> Would your object happen to be bicycle? (y/n)? y

>> Great, I got it right!

>>

>> Do you want to go again? (y/n)? n

Initially, the computer is not very intelligent, but it grows smarter each time it loses a game. If the computer guesses incorrectly, it asks you to give it a new question to help in future games. For example, suppose in the preceding log that the player was thinking of a car instead. You might get this game log:

>> Welcome to the cse143 question program.

>>

>> Do you want to read in the previous tree? (y/n)? y

>>

>> Please think of an object for me to guess.

>> Is it an animal? (y/n)? n

>> Does it have wheels? (y/n)? y

>> Would your object happen to be bicycle? (y/n)? n

>> What is the name of your object? car

>> Please give me a yes/no question that

>> distinguishes between your object

>> and mine--> Does it get stuck in traffic?

>> And what is the answer for your object? (y/n)? y

>>

>> Do you want to go again? (y/n)? n

The computer takes the new information from a lost game and uses it to replace the old incorrect answer node with a new question node that has the old incorrect answer and new correct answer as its children. After the preceding log, the computer’s overall game tree would be the following:

As usual, your output must match the output in the specification exactly. The output comparison tool will be helpful to make sure that your output is identical.

Q:

questions.txt

Is it an animal? Q:

Can it fly? A:

bird Q:

Does it have a tail? A:

mouse A:

spider Q:

Does it have wheels? Q:

Does it get stuck in traffic? A:

car A:

bicycle Q:

Is it nice? A:

TA A:

teacher

Is it an animal?

yes

no

Can it fly?

Does it have wheels?

yes

no

yes

no

bird

Does it have a tail?

Does it get stuck in traffic?

Is it nice?

yes

no

yes

no

yes

no

mouse

spider

car

bicycle

TA

teacher

Note that QuestionMain will always read and write to a file named questions.txt. If you want to start with the tree from spec-questions.txt or big-questions.txt, then you should copy the contents of those files to a file named questions.txt. Be careful, since the program will write the tree to this file every time.