CS 143 Lab 5 Program Specifications

Checkpoint Due 11:59PM Fri Aug 3
Lab Due 11:59PM Mon Aug 6

Problem Overview: This assignment will give you practice with *binary trees* and *recursion*. You will implement a variation of the yes/no guessing game known as 20 Questions. Each round of the game begins by you (the human player) thinking of an object for the computer to guess. The computer will try to guess your object by asking you a series of yes/no questions. Eventually the computer will guess what the object is. If the guess is correct, the computer wins the round; if not, you win the round.

Initially the computer is not very intelligent, but it grows more intelligent after each round that it loses. This is because you must give it a new question that it can ask in future rounds of the game. This collection of yes/no questions and guesses will be stored in a binary tree, and eventually stored in a file for later retrieval. You will implement a `QuestionTree` class to manage the binary tree, and a `QuestionTreeClient` class to manage the guessing game.

Lab 5 Logistics: Download and unzip the `Lab5Files` folder. In addition to source code from examples in BJP chapters 12 and 17, the folder contains files named `QuestionTree.java` and `QuestionTreeClient.java` which are templates for the classes you will be implementing in this assignment.

Lab 5 Checkpoint Specifications: Use the template `QuestionTreeClient.java` and `QuestionTree.java` files to implement your classes. For this checkpoint you will declare the data, and implement a default constructor and `write()` method for the `QuestionTree` class. You will write a `main()` method for the `QuestionTreeClient` class, to test the `QuestionTree` checkpoint implementation. Detailed specifications for the classes are given in tables below.

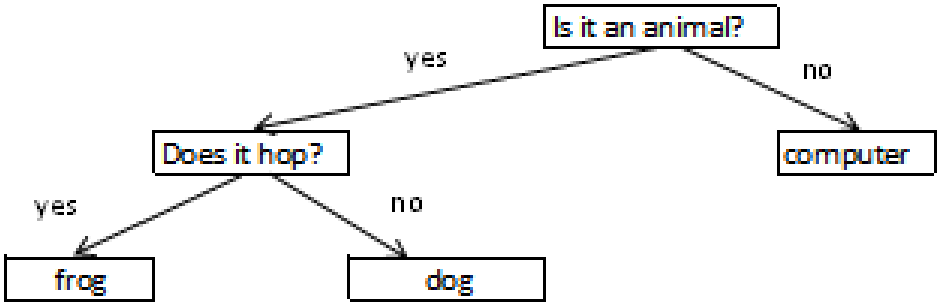

Figure 1 Sample `QuestionTree` Object
(The `root` points to "Is it an animal?" node.)

`QuestionTree` class	
Data Field	**Description**
`private static class QuestionNode` NOTE: A `QuestionNode` holds ***only a single `String`*** variable that will hold either a question or an answer/guess, depending on the node's position in the tree.	This nested inner class should include • A private `String` for holding the question/answer data value • Two private `QuestionNode` references to point left/yes and right/no • A constructor that accepts a string parameter which should become the data value for the node, unless the parameter value is `null` or empty, in which case throw an `IllegalArgumentException`.
`private QuestionNode root`	Will refer to the root `QuestionNode` of the tree. Initialize to `null`.
`private static Scanner console`	Should be connected to `System.in`. This object will be used for all console input performed by this class.
Method	**Description**
`public QuestionTree()` NOTE: The final version of your default constructor will have the `root` field point to a single node containing the answer/guess "computer." For the checkpoint we need a more complicated tree on which to test the write() method described below.	For the purposes of the checkpoint, write the code needed to have the `root` field point to a binary tree of `QuestionNode` objects that reflect the tree depicted in Figure 1 above. HINT: This requires a series of 5 assignment statements. Use the same strategy that the `Construct1` program following Table 16.1 in BJP 16.1 used to construct a linked list of three nodes.

`public void write(PrintStream output)`
 	
Data sent to the `output PrintStream` by this method for the tree containing the nodes shown in Figure 1 above is shown below.
 	
`Q:`
`Is it an animal?`
`Q:`
`Does it hop?`
`A:`
`frog`
`A:`
`dog`
`A:`
`computer`
	This method will be called by the client if they want to save/write the current tree to a file, or display it on the console screen.
	
	• This method should perform its job by correctly calling an appropriate `private` recursive helper method as discussed in BJP 17.2.
	• The `private` recursive helper method does all the work to produce the output such as that shown at the left. It does this by performing a *preorder traversal* of the tree.
	• Notice that before writing the node's data string to the `PrintStream`, a "Q:" is written prior to question nodes, while answer/guess nodes are preceded by "A:"
	• You can tell a "Q:" node from an "A:" node because "Q:" nodes have non-null left/yes and right/no pointers, while "A:" nodes have null left/yes and right/no pointers.
	
	The "Q:" and "A:" values written to a `PrintStream` connected to a file will aid in the subsequent retrieval of the node data from the file later.

QuestionTree class	
Method	**Description**
`public boolean yesTo(String prompt)` **NOTE: All code requiring a yes/no response from the user should perform that task by calling this method appropriately.**	This method asks the question indicated by the `prompt` parameter until the user types a valid y/n response. It then returns `true` if the user entered 'y' and `false` otherwise. The code for this method is provided in the `QuestionTree.java` template posted with this assignment. **Do not modify this code in any way.** *Call this method whenever you need to ask the user a yes/no question.*

Once you have implemented the constructor and `write()` methods for the `QuestionTree` class, write code in the `main()` method of the `QuestionTreeClient` class to perform the following tasks:

- Cause a `QuestionTree` object to be constructed.
- Call the `QuestionTree` object's `write()` method to display the tree to the `System.out` stream.
- The resulting output should look like that shown for the `write()` method in the specification table above.

End Lab 5 Checkpoint Specifications

Growing the Computer's Intelligence: The binary tree will be comprised of nodes whose data value is a string that will represent either a question or the name of an object. Each leaf node contains the name of an object. Each non-leaf (branch) node contains a yes/no question, and has left (yes) and right (no) pointers to non-null subtrees. Initially the tree contains a single leaf node with the value `computer`. Questioning begins at the root of the tree. When a leaf node is reached the computer makes a guess. Each time the computer guesses incorrectly, it takes the new information from the lost round and uses it to replace the old incorrect answer node with a new question node that has the old incorrect answer and the new correct answer as its children.

I/O Session: During the early stages of your program, an I/O session might look like the following, where user input is shown underlined only for easy identification:

```
Please think of an object for me to guess.
Would your object happen to be computer? (y/n)? n
What is the name of your object? dog
Give me a yes/no question to distinguish
     object dog from object computer
Is it an animal?
What is the answer for object dog? (y/n)? y

Do you want to go again? (y/n)? y
Please think of an object for me to guess.
Is it an animal? (y/n)? y
Would your object happen to be dog? (y/n)? n
What is the name of your object? frog
Give me a yes/no question to distinguish
     object frog from object dog
Does it hop?
What is the answer for object frog? (y/n)? y

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

```
Please think of an object for me to guess.
Is it an animal? (y/n)? y
Does it hop? (y/n)? y
Would your object happen to be frog? (y/n)? y
Great, I got it right!

Do you want to go again? (y/n)? n
Here is the final question tree:
Q:
Is it an animal?
Q:
Does it hop?
A:
frog
A:
dog
A:
computer
```

Lab 5 Remaining Specifications: Make the following additions or edits to the `QuestionTreeClient.java` file.

QuestionTree class	
Method	**Description**
`public QuestionTree()`	The constructor should now construct only a single node containing "computer" for the `root` to point to. That is the total of the knowledge that the tree begins with.
`void askQuestions()`	This method should use the current tree accessed through `root` to ask the user a series of yes/no questions until the method reaches a leaf node and tries to guess the object the user is thinking of. • If the guess is correct, display the "Great, I got it right!" message. • If the guess is wrong, expand the tree to include the object the user was thinking of and the question that will distinguish that object from the wrongly guessed one, as indicated in the sample I/O session above. **DRAW PICTURES TO FIGURE OUT HOW TO CONNECT THE NEW NODE TO THE TREE**.
`void read(Scanner input)`	This method will be called by the client if they want to replace the current tree with the data from the file indicated by the `input` parameter. Assume `input` has been connected to a file that is formatted as described in the `write()` specs above. Read entire lines of input using the `nextLine()` method. Expect to create and call a `private` recursive helper method to perform the preorder construction of the replacement tree.

QuestionTree class	
Private Methods	**Description**
You may have additional private methods to help your public methods, or other private methods.	Methods exceeding 25 non-blank, non-comment, non-{ and non-} lines should be subdivided into smaller cooperating methods that communicate through appropriate parameter-passing if necessary. You may want helper methods to simplify logic. For example, you may find it helpful for `askQuestions()` to call a method named `createNewSubtree()` that is responsible for creating the new subtree that replaces the wrong-guess leaf node. Finally, recursive solutions often require helper methods.

The final version of your program will allow the tree to be saved and retrieved using a file named `questions.txt`. The final program should be able to produce the following sample I/O session. Lines preceded by an asterisk and highlighted in yellow where produced by the client program. The asterisks are not part of the actual output. They are there for in case you can't see the yellow.

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

*Please think of an object for me to guess.
Would your object happen to be computer? (y/n)? n
What is the name of your object? dog
Give me a yes/no question to distinguish
     object dog from object computer.
Is it an animal?
What is the answer for object dog? (y/n)? y

*Do you want to go again? (y/n)? y
Please think of an object for me to guess.
Is it an animal? (y/n)? y
Would your object happen to be dog? (y/n)? n
What is the name of your object? frog
Give me a yes/no question to distinguish
     object frog from object dog.
Does it hop?
What is the answer for object frog? (y/n)? y

Do you want to go again? (y/n)? y
Please think of an object for me to guess.
Is it an animal? (y/n)? y
Does it hop? (y/n)? n
Would your object happen to be dog? (y/n)? y
Great, I got it right!

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

*Do you want to save this tree?  (y/n)? y
*Tree has been saved.
```

In subsequent runs of the `QuestionTreeClient` program, if the user answers y to the first question, the tree stored in the `questions.txt` file should replace the default single-node

“computer” tree. Be sure your program meets the Good Style Specs posted in the Quick Links module.