handout #26

**CSE143—Computer Programming II**

**Programming Assignment #7**

**due: Friday, 5/17/13**

This assignment will give you practice with binary trees.  You are to implement a yes/no guessing game.  The idea is that you construct a binary tree where each leaf has the name of an object and each branch node has a yes/no question that distinguishes between the objects.  It’s like a big game of 20 questions where each question is a yes/no question.

Initially you start out with a tree that has a single leaf node with “computer” in it.  This is, after all, a computer science class, so we should include computer as a possible answer.  Then your program learns new objects by interacting with the user.  You ask the user to think of an object and then you ask a series of questions.  Starting at the root of your tree, you ask the questions in the branch nodes and go to either the left or right subtree depending upon whether the user answers yes or no.  We will adopt the convention that yes is left and no is right.  Once you hit a leaf node, you ask the user if that is the object they were thinking of.  If the user says yes, you announce that they got it right.  Otherwise you ask the user for the name of their object and a yes/no question that would distinguish their object from the one you guessed.  You then ask what the answer is for their object (yes or no) and you replace the old leaf with a branch node containing the question from the user and containing the old object and the new object as children.

You also have to be able to write the tree to an output file and read it back in.  That way your question tree can grow each time a user runs the program.  To be able to read and write the tree, we need a set of rules for how to represent the tree which we’ll refer to as the *standard format* for a question tree.  A tree is specified by a nonempty sequence of line pairs, one for each node of the tree.  The first line of each pair should contain either the text “Q:” to indicate that it is a question node (i.e., a branch node) or the text “A:” to indicate that it is an answer node (i.e., a leaf node).  The second line of each pair should contain the text for that node (the question or answer).  The nodes should appear in preorder (i.e., in the order produced by a preorder traversal of the tree).

This program involves interaction with the user.  As always, we will use a Scanner linked to System.in to do this.  But with interactive programs, you don’t want to have more than one Scanner linked to System.in.  For that reason, in this program your class will construct the console Scanner.  It turns out that all that the main method needs is the ability to ask a yes/no question of the user.  So the interface for your class will include a method yesTo that main will call when it wants to interact with the user.  You must construct a single console Scanner that you store in a data field and use throughout your class.

The main program is called QuestionMain.java.  You are to write a class called QuestionNode.java (your tree node class) and QuestionTree.java.  In defining your tree node class, you should decide what data fields you need for solving this particular problem. In other words, you don’t have to use a generic binary tree; you can use one that is highly specific to this program.  Your node class should have at least one constructor.  It can have more than one, but don’t include any constructors that you don’t actually use in your QuestionTree class.

Your QuestionTree class should store the binary tree and should have the following public methods:

|  |  |
| --- | --- |
| Method | Description |
| QuestionTree() | This method should construct a question tree with one leaf node representing the object “computer”. |
| 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.  Read entire lines of input using calls on nextLine. |
| void write(PrintStream output) | This method will be called if the client wants to store the current tree to an output file.  The given PrintStream will be open for writing. |
| void askQuestions() | In this method you should use the current tree to ask the user a series of yes/no questions until you either guess their object correctly or until you fail, in which case you expand the tree to include their object and a new question to distinguish their object from the others. |
| boolean yesTo(String prompt) | This method asks the given question until the user types “y” or “n.”  It returns true if “y”, false if “n”. |

A log of execution is provided at the end of this write-up showing how the program should interact using your QuestionTree class.  You are to exactly reproduce the format of this log.

The details of the yesTo method are fairly uninteresting, so it is being provided for you (this code assumes a data field called “console” has been initialized).  You are to include this method without modification in your QuestionTree class.

// 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) {

    for (;;) {

        System.out.print(prompt + " (y/n)? ");

        String response = console.nextLine().trim().toLowerCase();

        if (response.equals("y"))

            return true;

        else if (response.equals("n"))

            return false;

        else

            System.out.println("Please answer y or n.");

    }

}

In terms of correctness, your class must provide all of the functionality described above.  In terms of style, we will be grading on your use of comments, good variable names, consistent indentation and good coding style to implement these operations.  Remember that you will lose points if you declare variables as data fields that can instead be declared as local variables.  You should also avoid extraneous cases (e.g., don’t make something into a special case if it doesn’t have to be).

You might be tempted in writing this program to “morph” what used to be an answer node of the tree into a question node.  This is considered bad style.  Question nodes and answer nodes are fundamentally different kinds of data.  You can rearrange where they appear in the tree, but you shouldn’t turn a former answer node into a question node just to simplify the programming you need to perform.

You should name your files QuestionNode.java and QuestionTree.java and you should turn it in electronically from the “assignments” link on the class web page.  A collection of files needed for the assignment is included on the web page as ass7.zip.  You will need to have QuestionMain in the same directory as your files in order to run QuestionMain.  The folder contains a version of question.txt, although you should be able to generate such a file from your own program.  The folder also contains a file called bigquestion.txt which contains a large data file of approximately 10 thousand animal names obtained from <http://animalgame.com/>. Also keep in mind that you can once again use the output comparison tool to check your solution.

**Sample execution (short file)**

Welcome to the cse143 question program.

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

Please give me a yes/no question that

distinguishes between your object

and mine--> Is it an animal?

And what is the answer for your object? (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

Please give me a yes/no question that

distinguishes between your object

and mine--> Does it hop?

And what is the answer for your object? (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

**Sample data file question.txt (after above log)**

Q:

Is it an animal?

Q:

Does it hop?

A:

frog

A:

dog

A:

computer