CSCI 204 - Introduction to Computer Science II

Lab 12- Tree

1 Objectives

- Write a recursive function to traverse and modify a binary tree.

2 Introduction to the Animal Game

In this lab you are to write a Java program *using recursion* to allow an individual to play an interactive computer game called *Animal*. The computer plays by trying to guess the name of an animal the player imagines. Although an animal may have many characteristics, considering only one at a time — "Is the animal furry?" or "Does it have horns?" — reduces its description to a series of binary (two way) choices. Some sample output from an *Animal* run will help you picture the game. The human types only "yes" or "no" for each question at this point in game.

```
Play again? yes
Think of an animal.  Does it have fur? yes
Does it have horns? no
Is it a lion? no
```

The program uses a binary tree to hold the questions and the animals. The interior nodes of the tree will always be questions, and the leaves will always contain animal names.

The program begins at the binary tree's root and asks the question stored as a string in that node. Whether the left or right node is visited next depends on the answer. If a "yes" we take the right link, otherwise we take the left. We either reach a further question or a leaf (final node with an animal in it).

The *Animal* program learns as it plays. If it reaches a leaf and guesses wrong, the following interaction takes place:

```
Is it a cow? no
I guessed wrong.  What animal were you thinking of? sheep
Type in an additional question I should have asked.
Does it say bah?
Is the correct answer yes or no? yes
```

Internally, two new nodes are added to the tree after the above conversation, along with the implication that cows don't say bah. The new animal and the current node's animal are placed in the two new leaves. The new question is placed in the current node and the node changes from one that contains an animal to one that contains a question.

Your main program should initialize the binary tree by creating three nodes — one question node "Does it have feathers?" and two animal nodes, "tiger" on left for "no" response and "chicken" for "yes" response on right as shown below:

3 Code in the Animal Class

```
public static void main(String[] args) {
    boolean debug = args.length == 1;

    // Start with three nodes
    TreeNode<String> left = new TreeNode<String>("tiger", null, null);
    TreeNode<String> right = new TreeNode<String>("chicken", null, null);
    TreeNode<String> root = new TreeNode<String>("Does it have feathers?",
            left, right);

    String response;
    do {
        System.out.print("Think of an animal.  ");
        play(root);
        if (debug)
            root.print();
        System.out.print("\nPlay again? ");
        response = input.nextLine();
    } while (responseIsYes(response));
}
```

You are to write the `play()` method *recursively*.

Create a `lab12` project and import all the files for today's lab from

`~csci204/2011-spring/student/labs/lab12`

into your `src` directory. Commit the project to your repository.

Among the files copied, you should see the complete `TreeNode` class as discussed in the lectures. You will also find an `AnimalExample` class file containing my solution. You may want to run this program a few times and observe the behavior of the program so you have an idea how the results should look. It's probably easiest to run it from the command line. Open a terminal window and change into the `src` directory for today's lab.

```
cd ~/csci204/lab12/src
```

There you will see `TreeNode.class` and `AnimalExample.class` that you can try out. Execute the program with

```
java AnimalExample
```

You may run the program in a debugging mode by providing a command line parameter as follows.

```
java AnimalExample debug
```

Running the program in debugging mode asks the program to print the existing game tree. The tree is printed from left to right with the root on the left, instead of having the root at the top. Each question is an interior node and the animals are the leaves. Here is an example of the output from debug mode.

```
       +--chicken
       |
  +--Does it fly?
  |    |
  |    +--emu
  |
  |
Does it have feathers?
  |
  |
```

```
|    +--giraffe
|    |
+--Does it have a long neck?
     |
     +--tiger
```

This can help you debug the program. When you complete your program correctly, you can run the program without the debugging mode.

Currently the printing method is called in the `main()` method within the loop. You may add it somewhere else if you need to.

The basic idea is that you need to traverse the existing tree based on the answer(s) from the user. If you are looking at a leaf node (How do you determine if a node is a leaf node?), then the node contains the name of an animal. The program should print this name and ask the user if this is the correct answer. If it is not, the algorithm needs to replace this node with a new tree branch (three nodes, one with the question, the other two are children nodes). If it is the correct answer, this round of the game is finished. If the current node is not a leaf node, it means it is an interior node which means it contains a question. Your algorithm needs to print the question and ask an answer from the user. If the answer is a 'yes', the program follows the right branch, otherwise follow the left branch.

4 Upon Completion

Commit everything to your Subversion repository.