CSC 207 **Assignment 9: Questions** **Spring 2011**

Assigned: Friday 22 April 2011

Due: Tuesday 3 May 2011

Objectives:

- Gain experience developing recursive data structures, namely **trees**
- Practice using **polymorphism**
- Remember how to use **recursion** as a problem-solving strategy
- Sharpen skills in reasoning about **encapsulation**
- Optionally learn how to use **serialization** to save and restore objects
- Build a fun game

Collaboration: This homework assignment may be completed individually or in pairs.

Submission: Follow the instructions for submitting programs via P-Web and handing in a printed copy. Generate unit tests and a demonstration run along with your statement justifying their sufficiency.

Background

As a child, you may have played the game twenty questions, in which one player thinks of an object, and the other player gets up to twenty yes/no questions in to determine what the object is. [1] Machine learning has been applied to this task and a handheld device for playing the game was developed by 20Q.net Inc and sold by Mattell. We can build a simple imitation of the game that simultaneously plays and learns by building a tree of answers and yes/no questions.

Infrastructure

`Node` Create an abstract class `Node` with no members.

`Guess` Create a class `Guess` that extends `Node`. It should simply store a string that would contain the "object" a player has thought of.

`Question` Create a class `Question` that also extends `Node`. A `Question` should store a string containing the text of the question, as well as two other `Nodes`, one representing the next play (e.g. another question or a guess) if the player's answer to the question is yes, and the other representing the next play if the player's answer is no. You should provide mutators and accessors for the two child nodes, and an accessor (only) for the question text.

Game

Play

Create a class called `Game` that stores a `Question` node as its root action. The game should have a public method `play` that takes no arguments. This method should contain an infinite loop that asks a user to think of an object that the program will then guess. If the root is null, the program must give up and ask for the object. Then, to help the program learn, it should ask the user for a question whose answer is "yes" for the item they just gave. With this information, a new root can be installed and another game can be played.

Add a private method, also called `play`, that takes a `Question` as its argument and asks the user the question appropriate to its argument. Given the user's response, if there is no next play, the program should give up and again ask for the object. The program should learn from this and install a new guess appropriately.

[1] See http://en.wikipedia.org/wiki/Twenty_questions for more details.

If the next play is to guess, the program should guess and ask for confirmation. If correct, the program can celebrate and terminate the current round. (Thus, allowing the user to think of another object). If the guess is incorrect, the program should ask for the correct object as well as a question whose answer is yes for that guess but no for the user's object. The program should learn from this and install a new play accordingly.

If the next play is to ask a question, the method should call itself recursively.

In case you have not yet noticed, you are recursively traversing and building a binary tree. The class `~weinman/courses/CSC207/code/questions/Game.java` contains two `static` methods you may use to prompt a user. for boolean or string information. In addition to the methods described above, add a `main` method that allows the user to begin playing the game.

Stats

Add a method to your `Game` that allows the program to print the following properties of the game tree when the user is finished:

- Total number of questions in the tree
- Total number of guesses in the tree
- Longest series of questions possible
- Shortest series of questions possible

Note that there are two ways to solve this problem. One is to place the computation solely in the `Game` class and the other is to add methods to the `Node` hierarchy. You may use either, but your statement must justify your choice.

Bonus

It can get tiring to start with the same dumb program every time. You can (optionally) save your learned game to disk using Java's built-in serialization interface and routines.

1. Declare that `Node implements Serializable` . This means that `Node` and any descendent classes can be automatically written to disk with the right methods. However, these classes must have a protected or public zero-argument constructor.

2. At any point (such as when the user states they are done playing the game), you can write a serializable object *`theObj`* to a stream (assuming all of its member fields are `Serializable`) with something like the following:

   ```
   (new ObjectOutputStream(new FileOutputStream(fileName))).writeObject(theObj);
   ```

   Note that `IOException`s may need to be caught.

3. The corresponding object can be read from a stream (and will need to be cast to the correct "known" type, e.g. `Question`) with something like the following:

   ```
   (Question)(new ObjectInputStream(new FileInputStream(fileName))).readObject();
   ```

4. Because Java also stores details about the state of the class when it wrote the objects, you'll likely want to store a "serial number" with the class file that will allow you to reliably read the object. Note, this number will allow you to read the object in the case you've re-compiled the class, but reading previously serialized objects will likely break unpredictably if you add, remove, or modify fields of the class.

   ```
   static final long serialVersionUID = 42L;
   ```

   The number you pick is up to you. The `L` modifier instructs the compiler to interpret the constant value as a `long`, rather than an `int`.

Example Interaction

```
$ java Game
Welcome to the questions game!
Starting a new game. Think of something and I will guess it.
I give up! What is the answer? a duck
Please type a question whose answer is YES for a duck
Is it an animal?
Play again? y
Starting a new game. Think of something and I will guess it.
Is it an animal? y
Is it a duck? n
I give up! What is the answer?  a cat
Please type a question whose answer is YES for a duck and NO for a cat
Is it a bird?
Play again? y
Starting a new game. Think of something and I will guess it.
Is it an animal? n
I give up! What is the answer?  a Christmas tree
Play again? y
Starting a new game. Think of something and I will guess it.
Is it an animal? y
Is it a bird? n
Is it a cat? n
I give up! What is the answer?  a giraffe
Please type a question whose answer is YES for a cat and NO for a giraffe
Does it live inside?
Play again?  y
Starting a new game. Think of something and I will guess it.
Is it an animal? y
Is it a bird? y
Is it a duck? n
I give up! What is the answer?  a pigeon
Please type a question whose answer is YES for a duck and NO for a pigeon
Does it live on water?
Play again? y
Starting a new game. Think of something and I will guess it.
Is it an animal? y
Is it a bird? n
Does it live inside? y
Is it a cat? y
Hooray!
Play again? n
Thanks for playing!
Total number of questions: 4
Total number of guesses: 5
Longest series of questions: 3
Shortest series of questions: 1
```

Acknowledgments Adapted from Chapter 11, Programming Problem 3 (pp. 636–637) in *Data Abstraction & Probelm Solving with* JAVA (3rd Edition; Addison Wesley) by J. J. Prichard and F. M. Carrano.