Project 3

Due date: April 26, 2020, 11:59pm

Objective

Model a problem using binary trees and implement a solution.

The Animal Game

View a demonstration of the animal game in this video (this is the same video as posted on the course website under the Project 3 link). You will implement the animal game in this project.

Requirements and Guidelines

Read all requirements and guidelines before starting on the design and implementation.

The game logic and its data structures

The questions and animals used in the game have to be stored as a binary tree, called *game tree* in the following. Each non-leaf node stores a question that has to be answered with “yes” or “no”. Each leaf node represents an animal. The first question asked by the game is the question stored in the root node. If the human player answers with “yes” to a question, then the game moves to the left child of the current node and continues asking the question in that left child. If the answer is “no”, the game moves to the right child. If the game reaches a leaf node, the game guesses the animal stored in the leaf.

When implementing nodes, decide whether or not the nodes in the tree need a reference to their parent node.

By default, the game should start with a game tree consisting of a single node that represents an animal. The animal should be randomly selected from a hard-coded list of animals.

Learning new questions and animals

If the human player wins, the game should extend the game tree to include the animal that the human player was thinking of.

- The application should ask the human player for
  - the animal they were thinking of
  - a question that distinguishes the guessed animal from the animal the human player was thinking of
  - The answer to the question for the human player’s animal.
- The leaf node with the animal that was incorrectly guessed by the game should be replaced by a question that distinguishes the guessed animal from the animal the human

player was thinking of. The guessed animal and the correct animal should be the leaves of the new question node.

For example, assume the game tree looks as follows:

```
      Does it live in water?
         /            \
 Is it a mammal?       horse
    /      \
whale   goldfish
```

Further assume that the human player is thinking of a dog. The game will first ask the question “Does it live in water?” and then, after the human player answers with “no”, it will guess “horse”.

Now the human player has to enter the requested data, i.e.

1. the animal “dog”
2. a question distinguishing a dog from a horse, like “Is it a carnivore?”
3. the answer to the question for dog, i.e. “yes”.

In the game tree, the leaf node “horse” is replaced by the question “Is it a carnivore?”. The new node has as left child “dog” (since the answer for dog is “yes”) and as right child “horse”:

```
       Does it live in water?
         /            \
 Is it a mammal?     Is it a carnivore?
    /      \            /       \
whale   goldfish       dog      horse
```

Saving the game tree

The user-interface must have an option to save the current game tree to a text file. The save operation has to traverse the nodes of the game tree using one of the four traversal methods level-order, inorder, preorder, or postorder. Each node value, i.e. the question or animal, should be written to file in the traversed order with a tag indicating whether the node is a question or an animal. You can decide on the detailed representation of the nodes and tags in the text file.

For example, in my sample solution, the game tree is traversed in preorder. Each question and animal is written to a separate line. Each line with a question or animal is followed by a line containing the single letter `Q` if the previous line is a question and containing the letter `A` if the previous line is an animal. The game tree below

```
        Does it live in water?
         /            \
 Is it a mammal?     Is it a carnivore?
    /      \            /       \
whale   goldfish   Can it fly?  horse
                    /    \
                eagle    dog
```

is written as follows to file

```
Does it live in water?
Q
Is it a mammal?
Q
whale
A
goldfish
A
Is it a carnivore?
Q
Can it fly?
Q
eagle
A
dog
A
horse
A
```

Loading a game tree

The user-interface must have an option to load a game tree from a file. The load operation has to use the same file format as your save operation. In particular, the load operation has to be able to load a file that has been previously saved by your application.

The User Interface

Use JavaFX to create a graphical user-interface. I used a single scene and a single FXML file with different panes for different parts of the game, where panes where hidden as needed. The user-interface should be functional, but keep it simple since the focus of this project is on the data structure and related algorithms.

Submission

Rename the Eclipse project to `CS3151Project3`*`FirstNameLastName`* where *`FirstNameLastName`* is your first and last name. Submit a zipped solution to the course website before the due date on Sunday, April 26, 11:59pm.

Grading

A project will receive 0 points if one or more of the following holds

- the code contains syntax errors
- the program crashes
- the program does not use a binary tree structure.

Points will be assigned as follows:

- 30 points: Suitable design of the game tree and the operations on the game tree.
- 20 points: Functional game play, including user interface.
- 20 points: The game learns new questions and animals while playing
- 15 points: The game tree can be saved to a file
- 15 points: The game tree can be saved to a file