Assignment 5: Expandable Binary Tree Guessing Game

For this assignment, you will use the BinaryTree implementation on Blackboard to program a guessing game. Firstly, you will manually build an initial tree in which each internal node is a yes/no question.

Yes goes to left side (left child), **No** goes to right side (right child).

Each *leaf node* in the tree is a guess.

If the user arrives at a leaf node and the guess is wrong, get the user to provide you with what the correct answer actually was and to provide a new yes/no question which can be added to the tree.

Example:

Are you thinking of an animal? Yes.

Is it a mammal? No.

Is it a bird? Yes.

Can it fly? No.

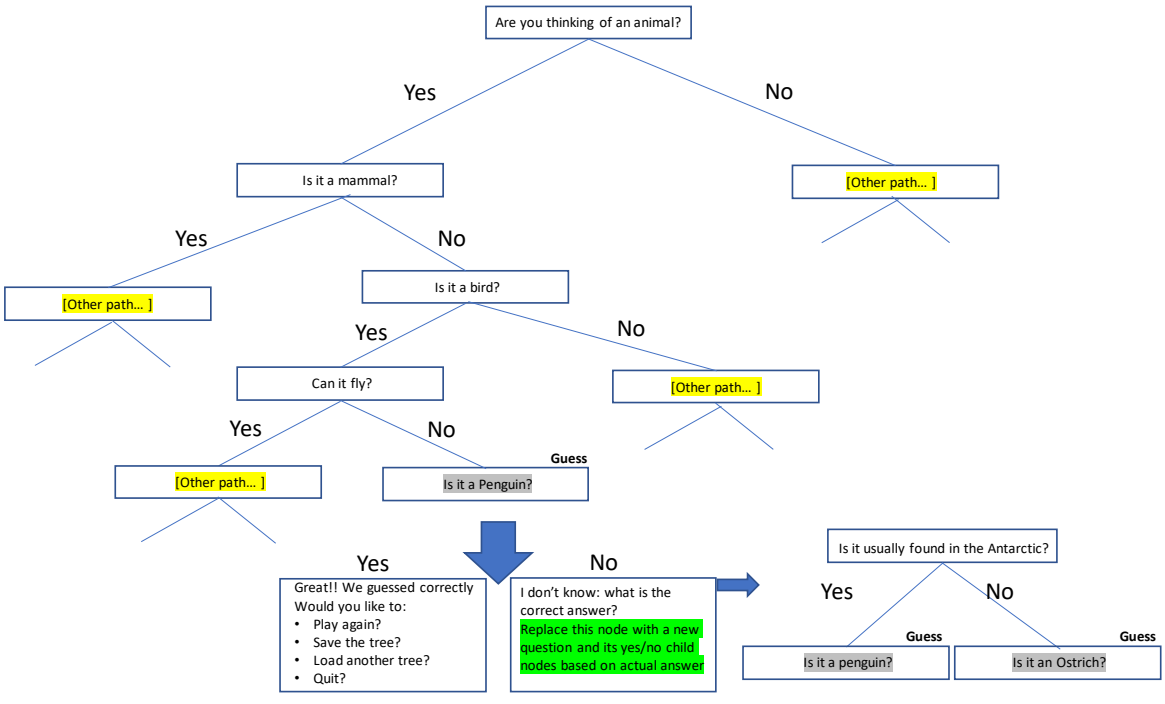
I think you are thinking of a Penguin? No.

I don't know: what is the correct answer? Ostrich.

[We need the user to provide a question that differentiates the guessed answer from the correct one]

Distinguishing question? Is it usually found in the Antarctic?

Right answer for Penguin: Yes.



Notes:

- You should start with populating an initial binary tree with at least four levels
- Internal nodes are all yes/no questions (yes: go left; no: go right)
- Leaf nodes are guesses at the correct answer
- We need to traverse the tree based on the users answers until we reach a leaf node.
- At a leaf node we do the following…
- **Incorrect answers**:
    - Ask user for a new question to distinguish the guess from the actual answer
    - Replace this leaf node with the new question and its yes/no answers
- **Correct answers**:
    - The tree guessed correctly!!
    - Give the user the following options:
        - Play again?
        - Store the tree?
        - Load a stored tree?
        - Quit?
- For testing purposes, you should also create a method that takes a binary tree as a parameter and then prints out a text representation of its content.

Storing and loading tree data

- For this assignment, you will need to devise a method for storing the contents of the tree in a file so that it can be loaded at a later date.
- You can use any method that you choose, from once the full information from the tree can be persisted. This can be anything from a specifically formatted text file to a serialized object.
    - For this part, you will have to carry out your own research to find a suitable method.
- This will ensure that you can continually add to the tree and that the new information won't be lost when you quit the program.

What will make this assignment challenging is that we will have to work with the BinaryNode items, not the BinaryTree, because our BinaryTree code implementation doesn't allow the tree to be changed directly. For this reason, it will be important to study the implementation code from Blackboard and understand exactly what each method can provide.

An example outline of the main steps:

```
//Variable to store user answer here
//Create an initial binary tree using a createTree method that you
//define yourself
while (true)
{
	BinaryNodeInterface<String> currentNode = tree.getRootNode();
	while (! currentNode.isLeaf())
	{
		//Ask the question and update current node
		//based on the answer
	}

	//At the leaf: got an answer that is either right or wrong

	//Present guess to user, and store their answer

	//Compare answer to guess. Two possibilities to proceed:
	//1. Answer is correct. Display options for user to continue
		//	Play again?
		//	Store the tree?
		//	Load a stored tree?
		//	Quit?
	//2. The answer was wrong. We need to expand the tree.
		//Get new question from user
		//Replace the current node with this question, and add
		//left & right children with the answers
}
```

Submission Notes:

- You should submit a *single PDF document* with the following sections:
  - Problem Statement (3 Marks)
    - Describe the problem. You can carry out your own research to describe the overall problem in sufficient detail to demonstrate that you fully understand it.
  - Analysis and Design Notes (5 Marks)
    - Before you begin coding, you should analyse the overall problem and create design notes for yourself. This can include identifying methods that will be required, writing basic pseudocode and outlining the flow of control for the program. You can then use this as a guide when you begin programming.
  - Code (17 Marks)
    - **IMPORTANT**: You must copy and paste your code as text into this section
    - **If you submit a screenshot of the code you will receive 0 marks.**
    - Basic full game implementation *without* storing/loading trees: 11 marks
    - Code to store the tree content to a file **and** to load from a file: 6 marks
    - Your code must also contain plenty of *meaningful* comments to fully describe the functionality of each part in your own words.
  - Testing (5 Marks)
    - You should extensively test each aspect of the code and provide screenshots of the testing output.

Due date: **29th of March**