CT2109 – Assignment 3 – Luke Gibbons - 15553883:

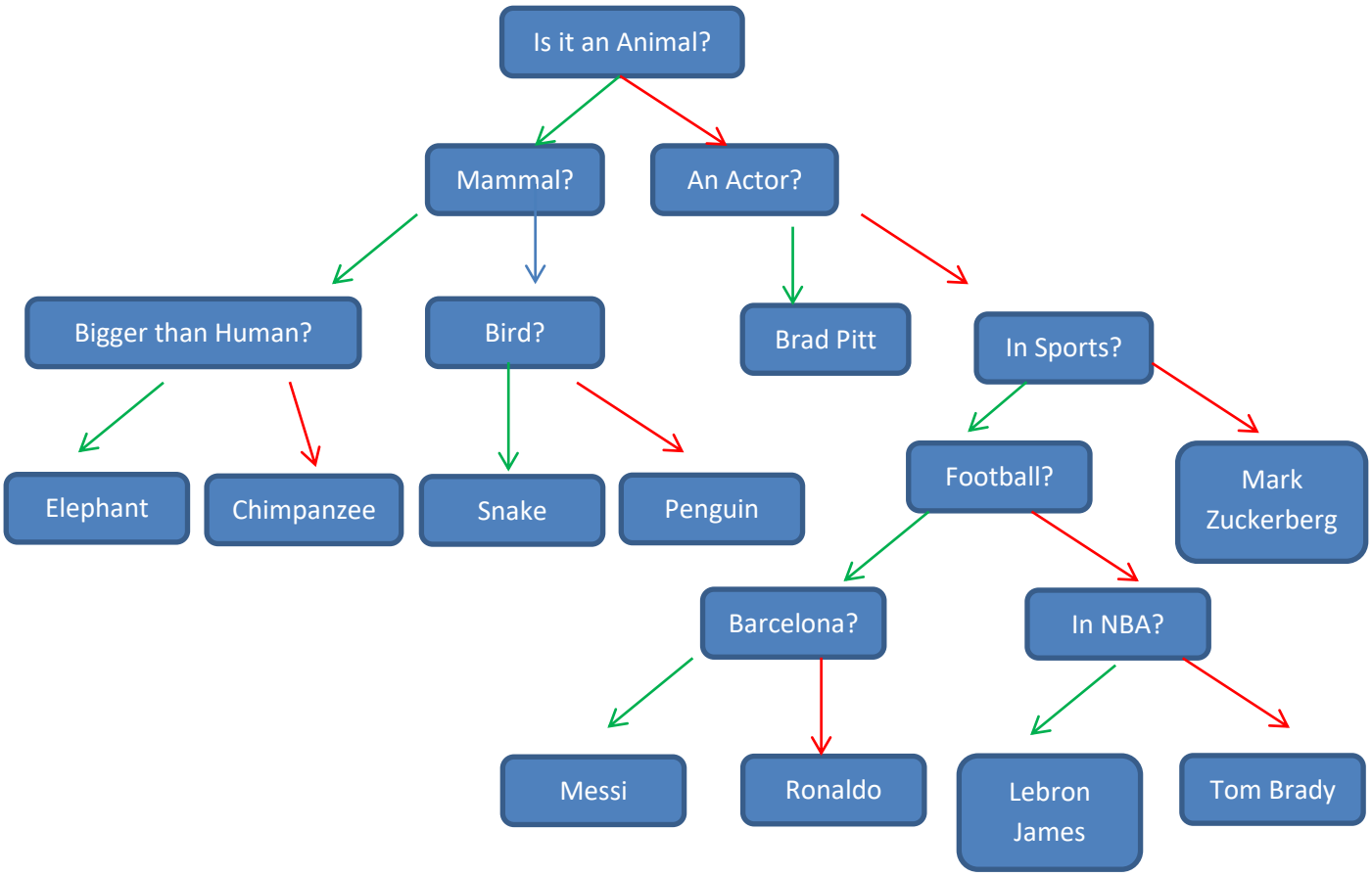
Expandable Binary Tree Guessing Game

Problem Statement:

The problem requires me to code a guessing game that works via a binary tree. The game will use yes/no questions and travel the path based on the answers given before reaching a leaf node and making this as a guess. The program also needs to implement 2 methods, one to save the binary tree to a file and the other to load it from a file.

Analysis and Design:

To begin I will need create a tree with at least 4 levels that I can use as a base.



A method to create this initial tree

The very first method I will need is one to create the initial tree. Using some of the above as an example the method could be similar to the following:

```
Public void createTree(BinaryTree tree)

        BinaryTree elephant = new BinaryTree('elephant')
        BinaryTree chimp = new BinaryTree('chimpanzee')
        BinaryTree big? = new BinaryTree('Bigger than…', elephant,
chimp)
        BinaryTree mammal = new BinaryTree('mammal?', big?, bird?)
        Tree.setRoot('Animal?', mammal, actor)
```

Going through the Nodes

The next part will be going through the nodes. I will need to get the node data, display it to the user and alter the path depending on the response.

Using the assignment skeleton the code could look something like the following:

```
While(true)
    Node currentNode = tree.rootNode
    While(!currentNode.isLeaf)

        Answer = JOptionPane(Display currentNode data to user)

        If(yes)
            currentNode = currentNode.leftChild
        else
            currentNode=currentNode.rightChild
```

If a leaf node is reached

When a leaf node is reached I will need to display the guess and ask whether the answer is right or wrong.

```
    int Correct = JOptionPane("Is currentNode data correct?")
```

If correct

If(correct == 0)

Choice = JOptionPane(Choose an option – 1.play again, 2. Store tree, 3. Load tree, 4. Quit)

If(Store tree)

Call method to store tree

Else if(load tree)

Call method to load tree

Else if(quit)

System.exit(0)

//Choosing anything else (play again) will automatically start over from top of while loop

If not correct

Else()

Guess = currentNodes data

Answer = JOptionPane("Input correct answer")

Question = JOptionPane("Input a distinguishing question")

Set current nodes data = question

Set current nodes leftChild = answer

Set current nodes rightChild = guess

Storing and loading the tree

This part will prove the most difficult. I initially thought that I could create an Array of all the nodes data and use a binary heap order. However, when I tested this on the test tree above it was clear it would not work as it must be a complete tree. This led me to serialization and deserialization. I had initially ignored this method as when I first looked up storing a binary tree these options seemed difficult, however, after some time I found the webpage https://www.tutorialspoint.com/java/java_serialization.htm which explained it very clearly and helped me understand that this would be the most efficient method for the task.

The saving method will take the tree and the file as its arguments. The code would be similar to the following:

```
FileOutputSteam = new FileOutputStream(fileName)
ObjectOutputStream out = new ObjectOutputStream(fileOut)
Out.writeObject(tree)
Out.close()
fileOut.close();
```

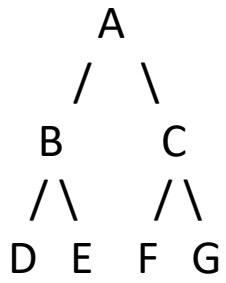
The loading method will return the tree and take the file as its argument. Its body will be similar to:

```
FileInputSteam fileIn = new FileInputStream(fileName)
ObjectInputStream in = new ObjectInputStream(fileIn)
Tree = (cast to BinaryTree object)in.readObject()
In.close()
fileIn.close()

return tree
```

Printing out the trees contents for testing

I initially wanted to print them out similar to:

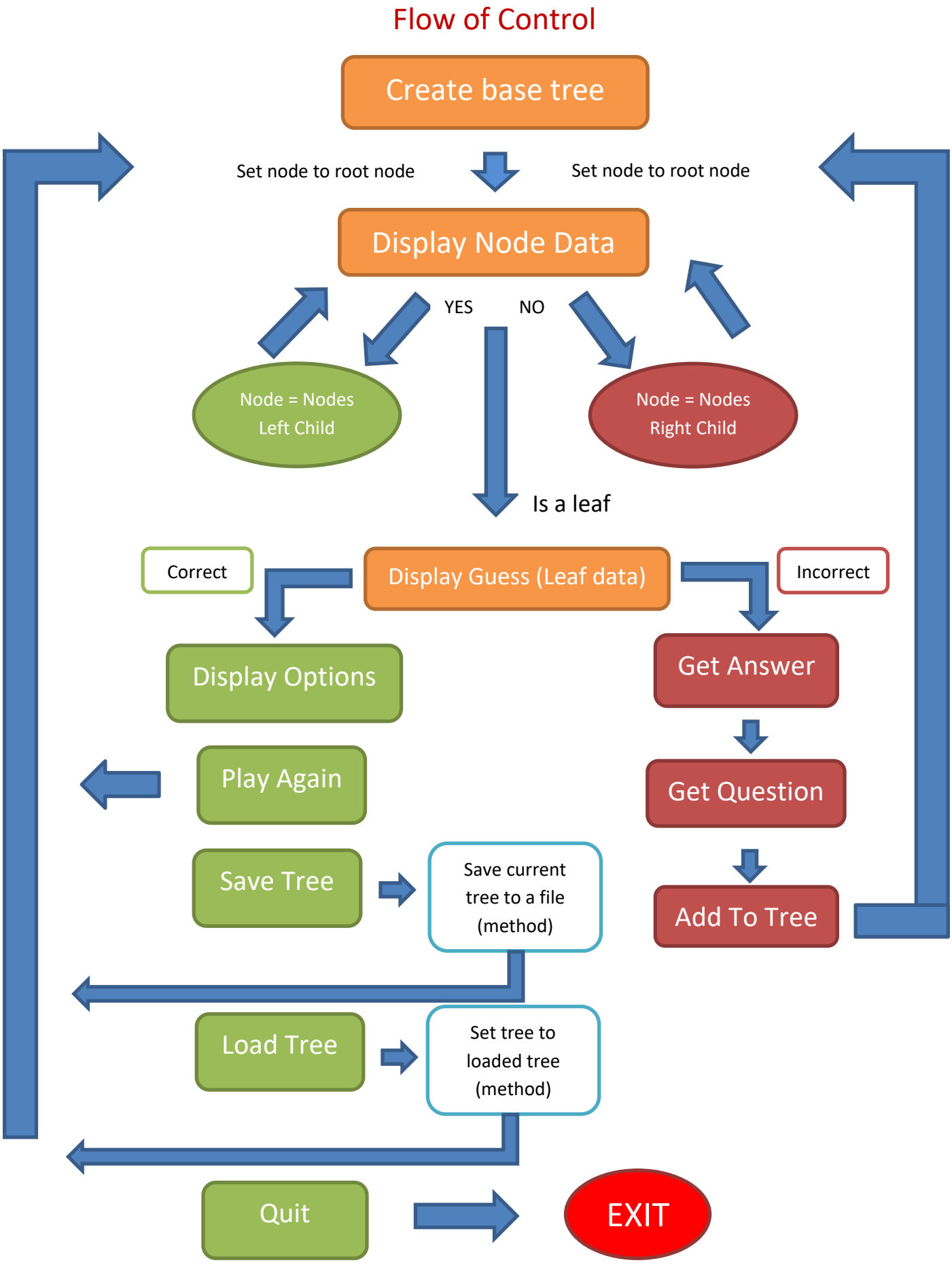


However, I soon realised that this would be too hard to accomplish due to the length of the data strings. The best way would therefore be to go through each node from top to bottom, left to right and list its child nodes as well as if it is a leaf node.

I will first put the binary nodes into an ArrayList using a for loop:

```
For(numberOfNodes)
        Add current node to array
        If(node is not a leaf)
                Add nodes left child to array
                Add nodes right child to array
        currentNode = get next node in array
```

I will then print the results to the console.



The Code:

```
import javax.swing.*;
import java.io.*;
import java.util.ArrayList;

public class BinaryTreeGuessingGame {


    private static String title = "Binary Tree Guessing Game"; //global variable - title for JOptionPanes

    public static void main(String[] args){

        System.out.println("Constructing a tree ... ");
        BinaryTree<String> testTree = new BinaryTree<String>(); //Empty tree object created
        createTree(testTree); //Method to fill tree with nodes - creates the base tree
        int j;
        Object[] options = {"Play Again", "Store This Tree", "Load a Tree", "Quit"}; //Options after correct guess
        String saveFile = "C://Users/Luke//Desktop/test.ser"; //File to save and load from - change to suit

        //Infinity loop unless user quits(Option 4)
        while (true){

            printTree(testTree); //For testing
            BinaryNodeInterface<String> currentNode = testTree.getRootNode(); //set current node to root node
            while(! currentNode.isLeaf()){

                //Ask user the question
                j = JOptionPane.showConfirmDialog(null,
                        currentNode.getData(), title, JOptionPane.YES_NO_OPTION);

                //if yes - set left child as currentNode else set right child as current
                if(j==0){
                    currentNode = currentNode.getLeftChild();
                }
                else{
                    currentNode = currentNode.getRightChild();
                }
            }

            //Display guess - this is the leaf node that is reached
            j = JOptionPane.showConfirmDialog(null,
                    "You're thinking of \t" + currentNode.getData() + "!" + "\n\nAm I Correct?", title, JOptionPane.YES_NO_OPTION);

            //if correct
            if(j==0){
                //Display message and options
                Object choice = JOptionPane.showInputDialog(null,
                        "I Win!\nWhat would you like to do?", title,
                        JOptionPane.INFORMATION_MESSAGE, null,
                        options, options[0]);

                //If store file - try and catch for exception handling
                if(choice.equals(options[1])){
                    try{
```

```
                        toFile(testTree, saveFile); //calls method to save the tree
to the file inputted
                    }
                    catch(IOException io){
                        JOptionPane.showMessageDialog(null, "I/O Error - tree
failed to save", title, JOptionPane.ERROR_MESSAGE);
                    }
                }

                //if load file - try and catch for error handling
                else if(choice.equals(options[2])){
                    try{
                        testTree = loadFile(saveFile); //calls method to load file
from file inputted
                    }
                    catch(IOException io){
                        JOptionPane.showMessageDialog(null, "I/O Error - tree
failed to load", title, JOptionPane.ERROR_MESSAGE);
                    }
                    catch(ClassNotFoundException c){
                        JOptionPane.showMessageDialog(null, "Class Not Found Error
- tree failed to load", title, JOptionPane.ERROR_MESSAGE);
                    }
                }
                //if Quit - exit program
                else if(choice.equals(options[3])){
                    System.exit(0);
                }
                //Otherwise Play Again
            }

            //If guess is wrong
            else{
                //Get correct answer
                String answer = JOptionPane.showInputDialog("Oh no, what is the
correct answer?");

                //Set the leftNode to this answer and the rightNode to the wrong
guess
                BinaryNodeInterface<String> leftNode = new BinaryNode<>(answer);
                BinaryNodeInterface<String> rightNode = new
BinaryNode<>(currentNode.getData());

                //Get the question to replace guess
                String question = JOptionPane.showInputDialog("What question
differentiates " + currentNode.getData() + " from " + answer + "?");

                //Replace the current guess with the question and set its children
to correct answer(left) and wrong guess(right)
                currentNode.setData(question);
                currentNode.setLeftChild(leftNode);
                currentNode.setRightChild(rightNode);

                JOptionPane.showMessageDialog(null, "Your question and answer have
been added!", title, JOptionPane.INFORMATION_MESSAGE);



            }

        }


    }

    //Method to store the inputted tree into the inputted File
    public static void toFile(BinaryTree<String> tree, String fileName) throws
IOException {
```

```
        //Create File and object output streams
        FileOutputStream fileOut = new FileOutputStream(fileName);
        ObjectOutputStream out = new ObjectOutputStream(fileOut);
        out.writeObject(tree); //write tree to file
        out.close(); //close file and object streams
        fileOut.close();
        JOptionPane.showMessageDialog(null, "The tree has been saved into " +
fileName, title, JOptionPane.INFORMATION_MESSAGE);


    }

    //Method to load the tree from the inputted file - the tree is returned
    public static BinaryTree<String> loadFile(String fileName) throws IOException,
ClassNotFoundException{

        //Create new file and object input streams
        FileInputStream fileIn = new FileInputStream(fileName);
        ObjectInputStream in = new ObjectInputStream(fileIn);
        BinaryTree<String> tree = (BinaryTree<String>) in.readObject(); //set tree
as the object in the file
        in.close(); //close file and object streams
        fileIn.close();

        JOptionPane.showMessageDialog(null, "The tree has successfully loaded!",
title, JOptionPane.INFORMATION_MESSAGE);
        return tree; //return the tree

    }

    //Method to print out the BinaryTree in text format to the console
    public static void printTree(BinaryTree<String> tree){
        ArrayList<BinaryNodeInterface<String>> arr = new ArrayList<>(); //create
new arrayList of Node interfaces
        BinaryNodeInterface<String> currentNode = tree.getRootNode(); //set
currentNode to root
        arr.add(currentNode); //add the root node to the array

        //Loop to add nodes to arrayList
        for(int i=1; i<tree.getNumberOfNodes(); i++){
            if(!currentNode.isLeaf()) { //If node has children - add them to array
                arr.add(currentNode.getLeftChild());
                arr.add(currentNode.getRightChild());
            }
            currentNode = arr.get(i); //set the current node to the next iteration
        }

        //Print out tree in readable format
        System.out.println("A Representation of the tree");
        System.out.println("**********************");
        for(int i=0; i<arr.size(); i++){ //For all nodes

            if(!arr.get(i).isLeaf()){
                if(i==0){
                    System.out.print("The root node is "); //If its the root node -
display this
                }

                System.out.println(arr.get(i).getData());

                if(arr.get(i).getLeftChild().isLeaf()){
                    System.out.println("It's left child is a leaf: " +
arr.get(i).getLeftChild()); //if its a leaf - display this
                }
                else {
                    System.out.println("It has the left child: " +
arr.get(i).getLeftChild());
```

```
                }
                if(arr.get(i).getRightChild().isLeaf()){
                    System.out.println("Its right child is a leaf: " +
arr.get(i).getRightChild()); //if its a leaf display this
                }
                else {
                    System.out.println("It has the right child: " +
arr.get(i).getRightChild());
                }

                System.out.println("**********************");
            }
        }
    }

    //Method to create the Base Tree
    public static void createTree(BinaryTree<String> tree){
        //create the leaves first
        BinaryTree<String> guessElephant = new BinaryTree<String>("an elephant");
        BinaryTree<String> guessChimp = new BinaryTree<String>("a chimpanzee");
        BinaryTree<String> guessPenguin = new BinaryTree<String>("a penguin");
        BinaryTree<String> guessSnake = new BinaryTree<String>("a snake");
        BinaryTree<String> guessBrad = new BinaryTree<String>("Brad Pitt");
        BinaryTree<String> guessMessi = new BinaryTree<String>("Messi");
        BinaryTree<String> guessRonaldo = new BinaryTree<String>("Ronaldo");
        BinaryTree<String> guessLebron = new BinaryTree<String>("Lebron James");
        BinaryTree<String> guessBrady = new BinaryTree<String>("Tom Brady");
        BinaryTree<String> guessZuck = new BinaryTree<String>("Mark Zuckerberg");

        //Create all the inner nodes and join them together
        BinaryTree<String> qBig = new BinaryTree<String>("Is it bigger than a
human?", guessElephant, guessChimp);
        BinaryTree<String> qBird = new BinaryTree<>("Is it a bird?", guessPenguin,
guessSnake);
        BinaryTree<String> qBarca = new BinaryTree<String>("Does he play for
Barcelona?", guessMessi, guessRonaldo);
        BinaryTree<String> qNba = new BinaryTree<String>("Does he play in the
NBA?", guessLebron, guessBrady);
        BinaryTree<String> qFootball= new BinaryTree<>("Does he play football?",
qBarca, qNba);
        BinaryTree<String> qSportsman = new BinaryTree<>("Does he play sports?",
qFootball, guessZuck);
        BinaryTree<String> qMammal = new BinaryTree<>("Is it a mammal?", qBig,
qBird);
        BinaryTree<String> qActor = new BinaryTree<>("Is he an actor?", guessBrad,
qSportsman);

        //Finish the tree by setting up the root
        tree.setTree("Is it an Animal?", qMammal, qActor);


    }
}
```

Also used code from blackboard:
(Unchanged)

TreeInterface.java, BinaryTreeInterface.java, BinaryTree.java, and BinaryNodeInterface.java

(Changed)

BinaryNode.java – see below for added code to class:

```
@Override
public String toString(){
    return((String)getData());
}
```

Testing

Is the initial tree created successfully?

Using the method I created to print out the tree contents I will be able to tell if the base tree is successfully created using the createTree method.

Expected Outcome:

A text version of the base tree I devised at the beginning of the analysis section

Actual Outcome:

```
Constructing a test tree ...
A Representation of the tree
**********************
The root node is Is it an Animal?
It has the left child: Is it a mammal?
It has the right child: Is he an actor?
**********************
Is it a mammal?
It has the left child: Is it bigger than a human?
It has the right child: Is it a bird?
**********************
Is he an actor?
It's left child is a leaf: Brad Pitt
It has the right child: Does he play sports?
**********************
Is it bigger than a human?
It's left child is a leaf: an elephant?
Its right child is a leaf: a chimpanzee
**********************
Is it a bird?
It's left child is a leaf: a penguin
Its right child is a leaf: a snake
**********************
Does he play sports?
It has the left child: Does he play football?
Its right child is a leaf: Mark Zuckerberg
**********************
Does he play football?
It has the left child: Does he play for Barcelona?
It has the right child: Does he play in the NBA?
**********************
Does he play for Barcelona?
It's left child is a leaf: Messi
Its right child is a leaf: Ronaldo
**********************
Does he play in the NBA?
It's left child is a leaf: Lebron James
Its right child is a leaf: Tom Brady
**********************
```

Pass or fail? Pass

Does the tree navigate in accordance to answers?

I will try and get 3 results from the tree to see if it goes through it correctly.

Test 1 – Brad Pitt:

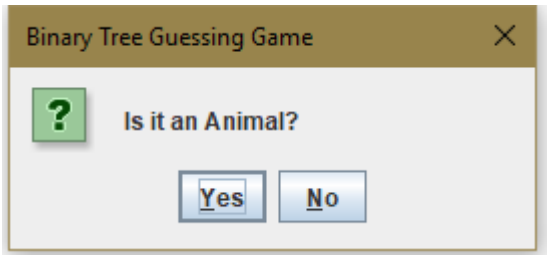
Expected outcome:

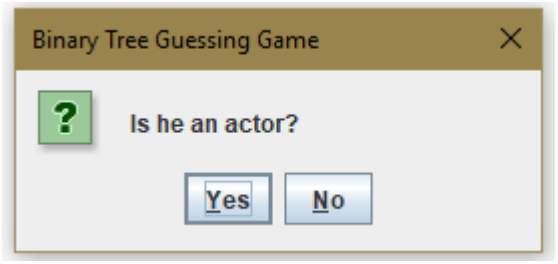
Is it an animal? No

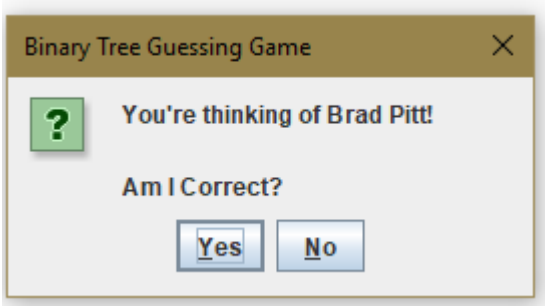
Is he an actor? Yes

It is Brad Pitt

Actual Outcome:







Test 2 – Snake:

Expected Outcome:

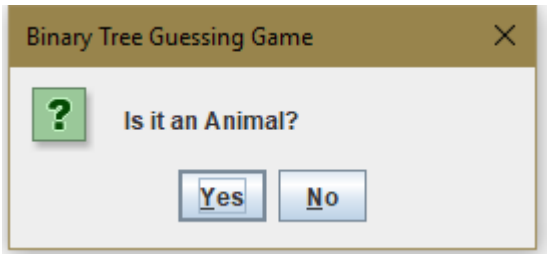
Is it an Animal? Yes

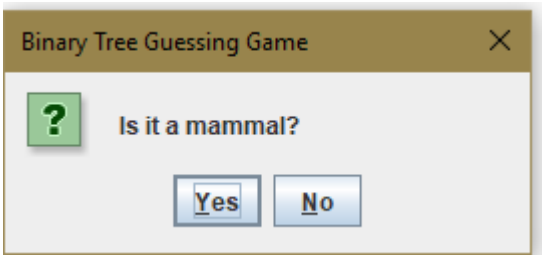
Is it a mammal? No

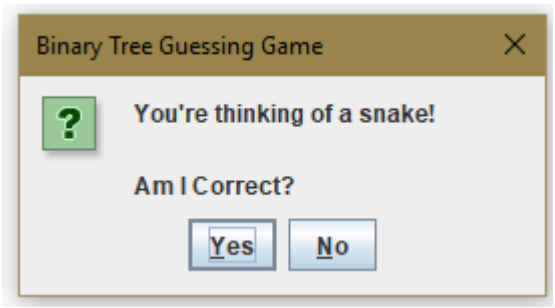
Is it a Bird? No

It is a Snake

Actual Outcome:







Test 3 – Ronaldo

Expected Outcome:

Is it an animal? No

Is he an actor? No

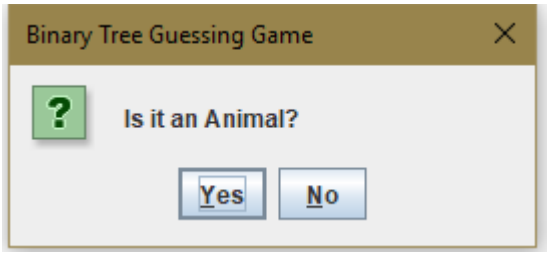
Is he a sportsman? Yes

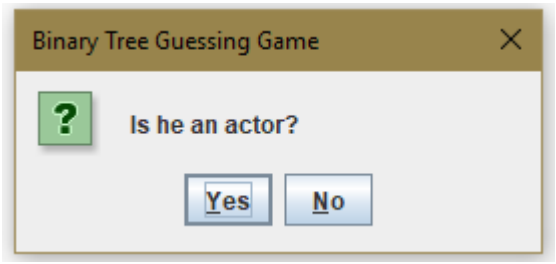
Does he play football? Yes

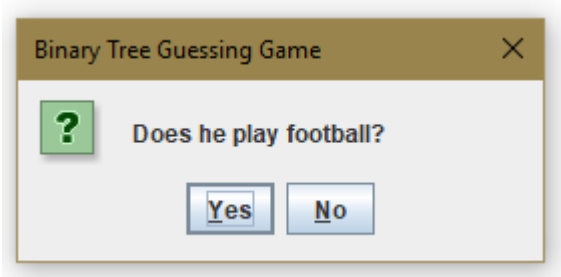
Does he play for Barcelona? No

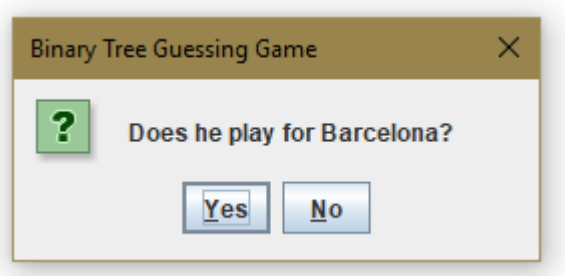
It is Ronaldo

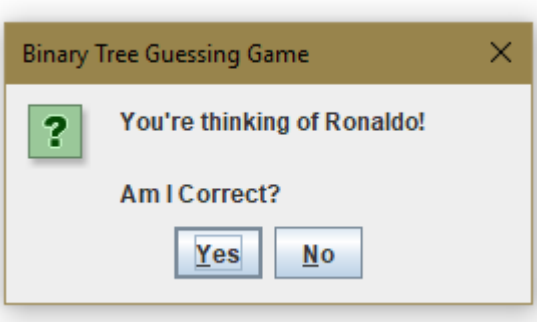
Actual outcome:











Pass or fail? All Tests Pass

Do the inputs work if the guess is wrong?

I will now test if the inputs that user enters (answer and distinguishing question) when the guess is wrong are added to the tree.

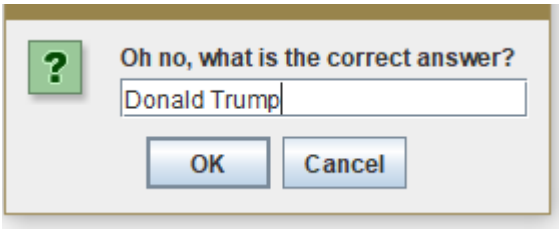
Test – try to get Donald Trump

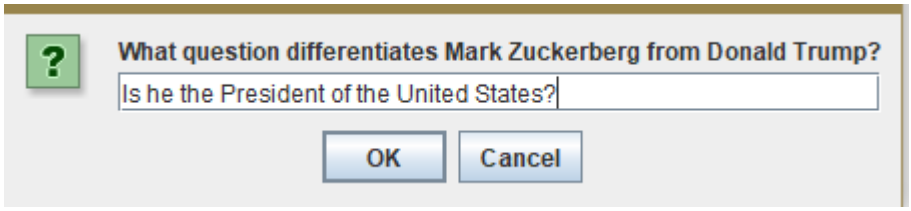
Expected Outcome:

The test result is impossible to achieve so when wrong result prompts for input – 'Donald Trump' and question 'Is he the current president of the United States?' are added to the tree. On the second try it guesses correctly.

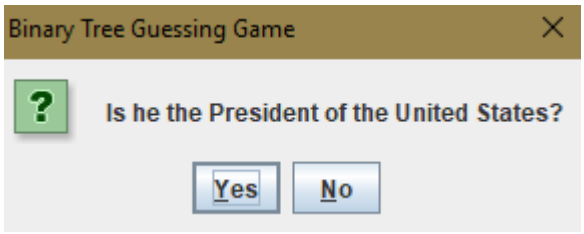
Actual Outcome:

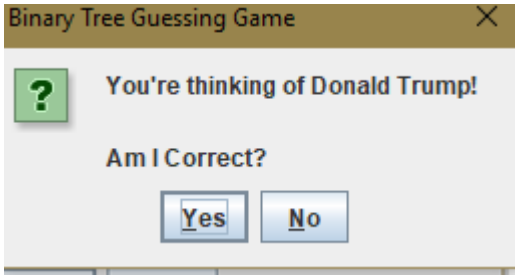
On first try it prompts for inputs:





On second try:





Pass or fail? Pass

Do the 4 options after correct guess work correctly?

I will test each option to determine if all of them work correctly

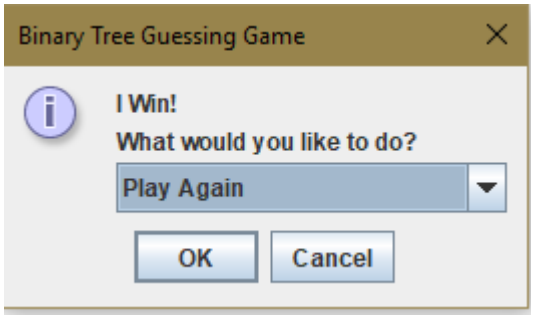
Option 1: play again

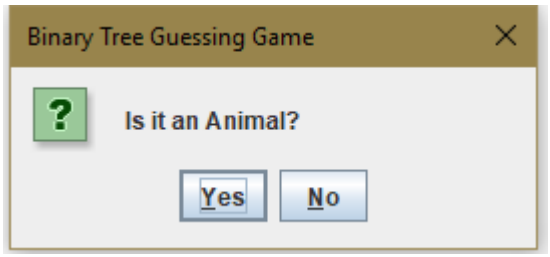
Expected outcome:

Program runs again from start

Actual Outcome:

Program runs again from start:





Option 2 and 3: Stores a file and loads it

Test 1: It stores the file and loads it

Expected Outcome:

I will input an answer into the tree when it gets it wrong.

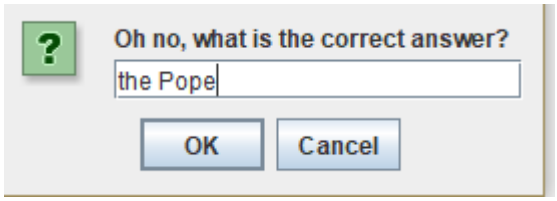
Actual answer? 'The pope'

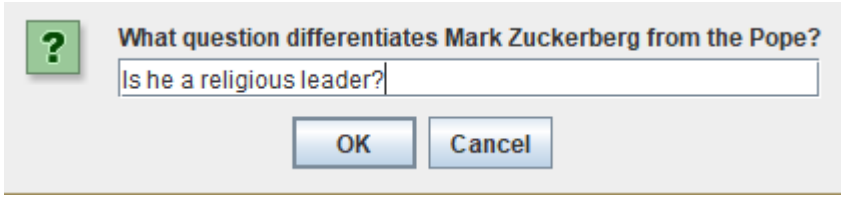
Distinguishing Question? 'Is he a religious leader?'

I will then run through the tree again to get 'the Pope' and save the tree. Next, I will exit the program and start it again. Then I will get to the correct screen and choose to load a tree. Finally I will run through the program again and get the pope.

Actual outcome:

Entering details into tree:

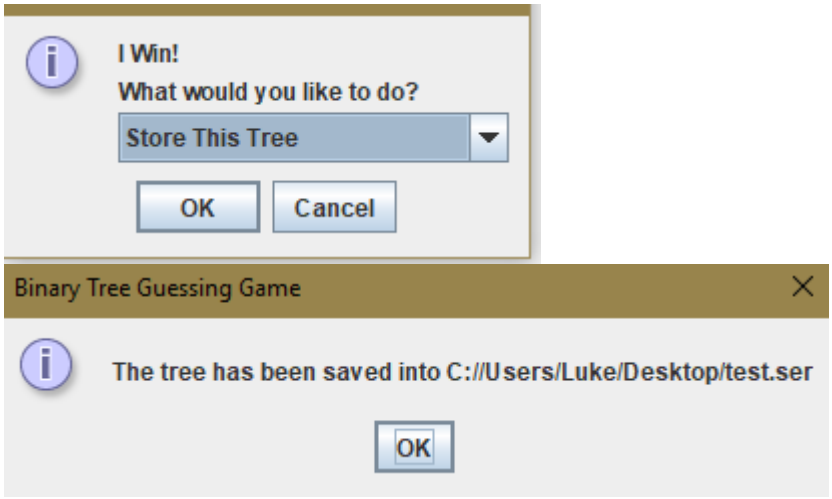




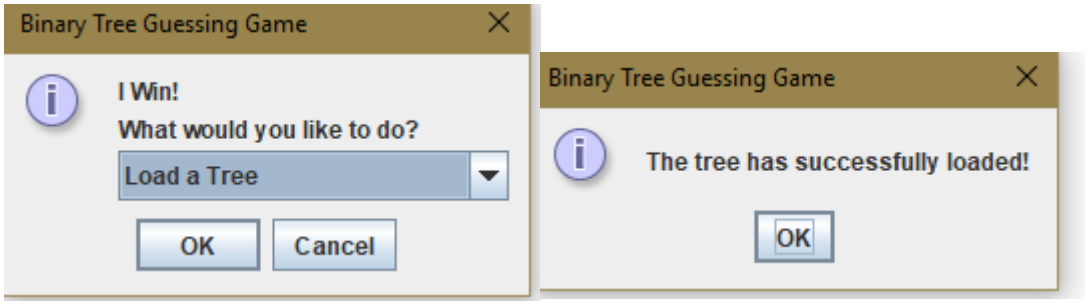
printTree output:

```
**********************
Is he a religious leader?
It's left child is a leaf: the Pope
Its right child is a leaf: Mark Zuckerberg
**********************
```

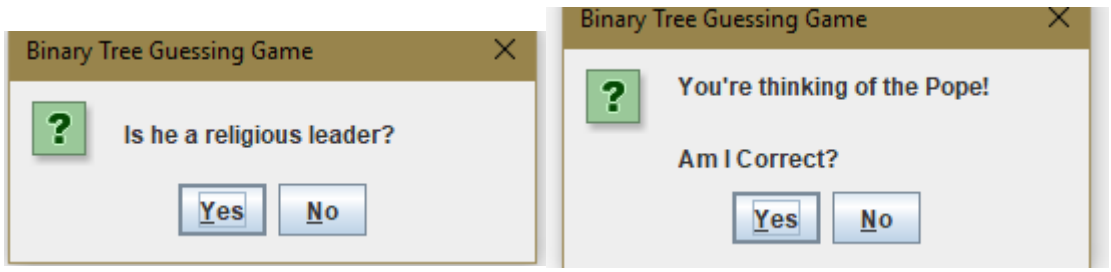
Saving the tree to a file and exiting:



Loading the tree from the file:



Getting the answer added at the beginning:



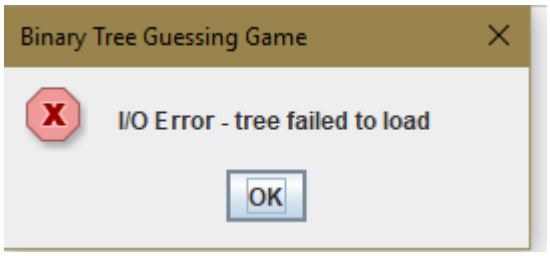
Test 2: Throws error if unknown folder location

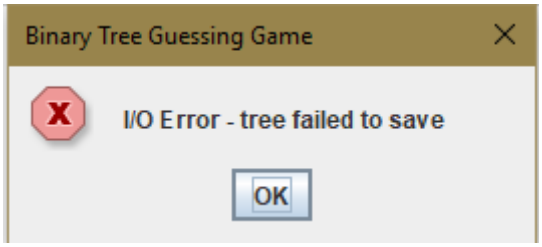
Expected outcome:

Error alert

Actual outcome:

Error alert





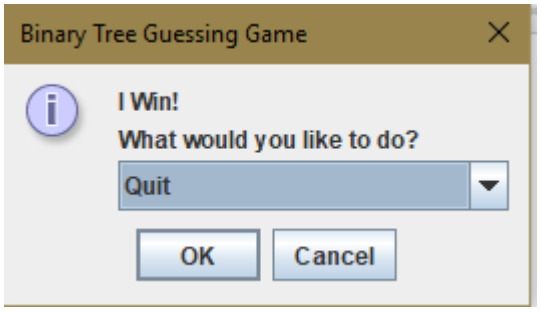
Option 4: Quit

Expected output:

Program exits with exit status 0

Actual output:

Program exits with exit status 0




```
Process finished with exit code 0
```

Pass or fail? All Options/Tests Pass