Assignment 5 – Oisin Mc Laughlin – 22441106

Problem Statement

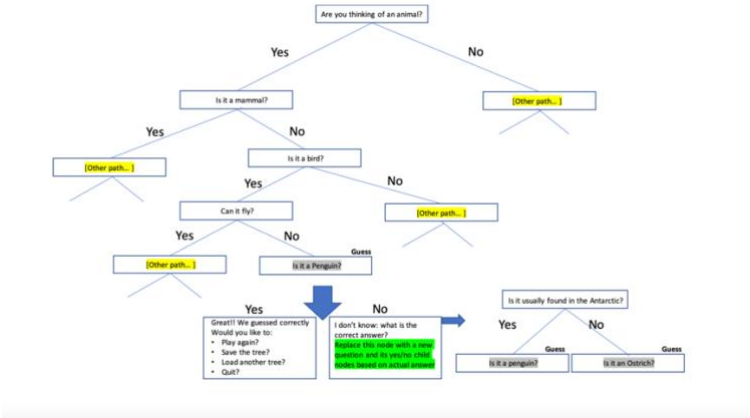
Using the code from canvas, I will need to create a guessing game that will implement the use of a binary tree. Each binary node will store different options to guess and if a user wants to add more guessable options, they can expand that tree. Expanding the tree will be done using serialising to save the tree and deserializing to load the tree.

The complex part of this is expanding on the binary tree, this makes the code robust and expandable and is a great way to adapt our knowledge from the lectures.

Analysis and Design Notes

First of all, I will use the binary node and binary tree code provided on canvas. If there is no tree already, one will be created. The tree will have a yes/ no question and each leaf will be a guess. When the tree is created/ loaded it, the program will traverse the tree and when it reaches a leaf, it asks the user if it guessed correct or to provide a new answer to expand a tree as well as asking for a question that would differentiate it.

The first method I will make will create the tree and populate it in the same format as the one on canvas. This will be done by creating leaf's for each question, branch's above that and then the animal root.



The next method will expand the tree when a guess is incorrect, it will then create a new node for the correct and incorrect animals. It will prompt the user, If yes for animal, current question of node to user, attach correct animal as left child and previous as right. Otherwise it will attach previous animal as left and correct animal as right nodes.

The third method will store the tree as a txt file and the fourth method will load it from the file path.

The fifth method will be where the actual guessing game happens. It will get the root node and prompt the user with a question as well as scanning the response each time as it loops through the tree to the leaf node. If what the user guesses is not at the leaf, it will prompt the user to add a new animal and ask a question that differentiates it.

In main the actual game loop is run until the user no longer wants to play again, it may also contain some additional logic to create the tree and also storing the tree.

Code

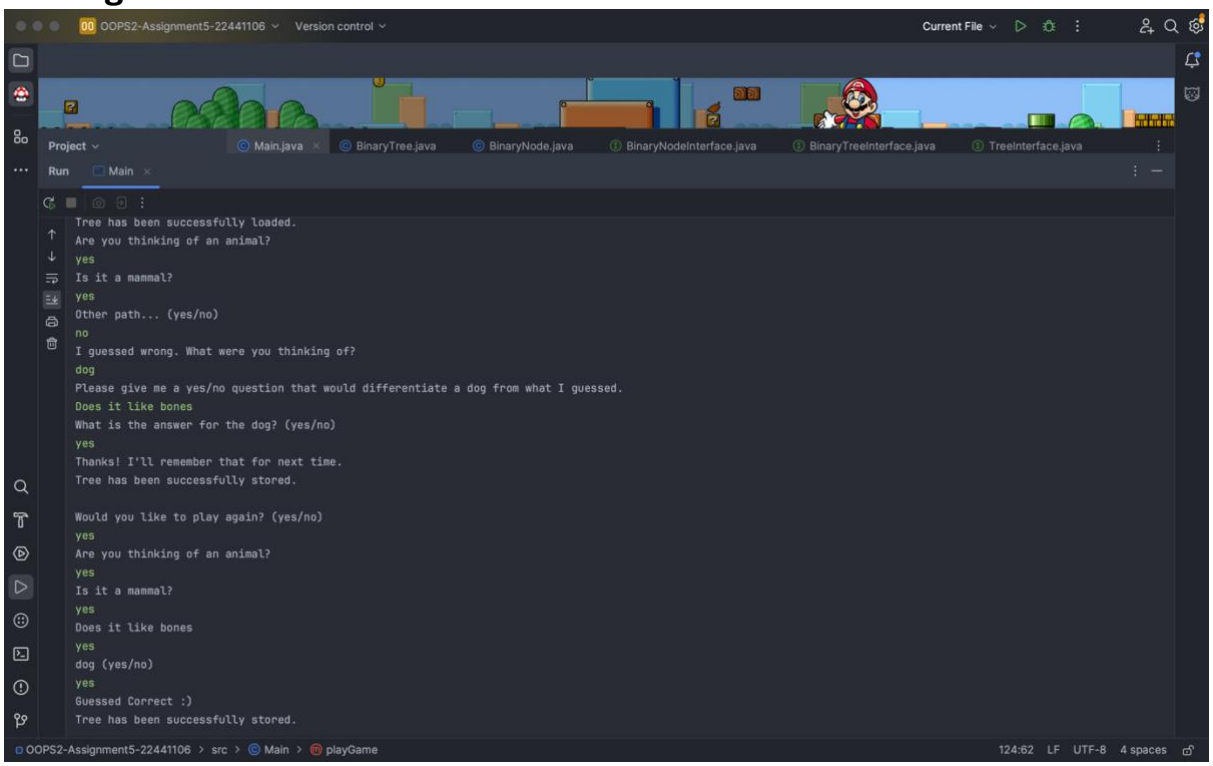
```
import java.io.*;
import java.util.Scanner;

public class Main {
    //Scanner obj and file path where tree is saved and loaded from
    private static Scanner scanner = new Scanner(System.in);
    private static final String filePath = "/Users/oisinmcl/Java
Code/OOPS2-Assignment5-22441106/src/Tree.txt";

    //Creates initial tree structure like from lecture slides
    public static void createTree(BinaryTree<String> tree) {
        //Creates leaf for each question
        BinaryTree<String> penguinTree = new BinaryTree<>("Is it a
penguin?");
        BinaryTree<String> ostrichTree = new BinaryTree<>("Is it an
ostrich?");
        //Creates branch for bird question
        BinaryTree<String> canItFlyTree = new BinaryTree<>("Can it fly?",
penguinTree, ostrichTree);
        //Is it a bird branch
        BinaryTree<String> birdTree = new BinaryTree<>("Is it a bird?",
canItFlyTree, new BinaryTree<>("Other path..."));
        //Is it mammal branch
        BinaryTree<String> mammalTree = new BinaryTree<>("Is it a mammal?",
new BinaryTree<>("Other path..."), birdTree);
        //Creates root
        tree.setTree("Are you thinking of an animal?", mammalTree, new
BinaryTree<>("Other path..."));
    }

    // Expands tree when a binary guess is incorrect
    public static void expandTree(BinaryNodeInterface<String> currentNode,
String correctA, String question, String yesCorrect) {
        //Correct animal new node
        BinaryTree<String> correctAnimalTree = new BinaryTree<>(correctA);
        //New node for incorrect guessed animal
        BinaryTree<String> guessedAnimalTree = new
BinaryTree<>(currentNode.getData());
        //If yes for animal, current question of node to user, attatch
correct animal as left child and previous as right
        if (yesCorrect.equals("yes")) {
            currentNode.setData(question);
            currentNode.setLeftChild(correctAnimalTree.getRootNode());
            currentNode.setRightChild(guessedAnimalTree.getRootNode());
        }
        else {
            //Set current node to new question to user, attatch previous
animal guess as left, attatch correct to right
            currentNode.setData(question);
            currentNode.setLeftChild(guessedAnimalTree.getRootNode());
            currentNode.setRightChild(correctAnimalTree.getRootNode());
        }
    }

    //Stores current binary tree to the filepath
    public static void storeTree(BinaryTree<String> tree) {
```

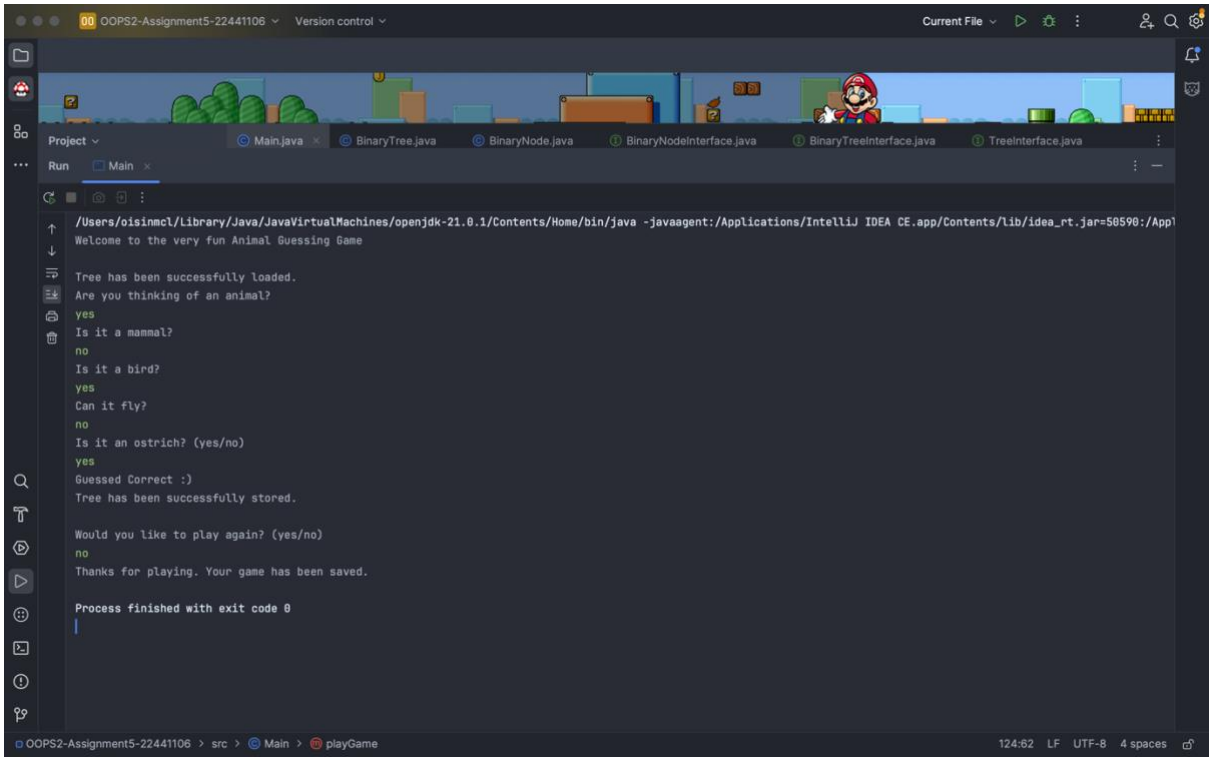
```java
        //Open stream and write obj data to file
        try (ObjectOutputStream outputStream = new ObjectOutputStream(new
FileOutputStream(filePath))) {
            outputStream.writeObject(tree);
            System.out.println("Tree has been successfully stored.");
        }
        //Print error if something goes wrong
        catch (IOException e) {
            System.out.println("Error storing the tree: " +
e.getMessage());
        }
    }

    //Loads binary tree from the filepath
    public static BinaryTree<String> loadTree() {
        //Open stream and read obj data from file, read binary tree, close
inout and return tree
        try (ObjectInputStream inputStream = new ObjectInputStream(new
FileInputStream(filePath))) {
            BinaryTree<String> tree = (BinaryTree<String>)
inputStream.readObject();
            System.out.println("Tree has been successfully loaded.");
            return tree;
        }
        //Print error if something goes wrong (hopefully not)
        catch (IOException | ClassNotFoundException e) {
            System.out.println("Error loading the tree: " +
e.getMessage());
            return null;
        }
    }

    //Where the game is actually played that runs the logic for the
guessing and decides what to do with it
    public static void playGame(BinaryTree<String> tree) {
        //Get root node which contains first question, ask user and scan
response
        BinaryNodeInterface<String> currentNode = tree.getRootNode();
        System.out.println(currentNode.getData());
        String answer = scanner.nextLine().trim().toLowerCase();

        //If no, offer exit or try something fancy
        if (answer.equals("no")) {
            System.out.println("Ok, let's try something else or you can
exit.");

            return;
        }
        //If answer is not yes, put the user in their place and to stop
messing around with the incorrect inputs
        else if (!answer.equals("yes")) {
            System.out.println("Please answer 'yes' or 'no' ONLY.");
            return;
        }

        //If user response is yes, proceed next quesiton in tree
        currentNode = currentNode.getLeftChild();

        //Loop through question nodes of tree until leaf, ask question and
scan response
        while (!currentNode.isLeaf()) {
```

```
            System.out.println(currentNode.getData());
            answer = scanner.nextLine().trim().toLowerCase();

            //If yes, navigate left responses
            if (answer.equals("yes")) {
                currentNode = currentNode.getLeftChild();
            }
            //If no, navigate right responses
            else if (answer.equals("no")) {
                currentNode = currentNode.getRightChild();
            }
            //Otherwise user input isn't correct and put them in their
place
            else {
                System.out.println("Please answer 'yes' or 'no' ONLY.");
            }
        }

        //Once at leaf, make guess and ask, scan user response
        System.out.println(currentNode.getData() + " (yes/no)");
        answer = scanner.nextLine().trim().toLowerCase();

        //If user confirms correct, print nice message
        if (answer.equals("yes")) {
            System.out.println("Guessed Correct :)");
        }
        //Else, ask what they were after and ask more questions to
distingush what kind of animal they want to add to the tree and add it
        else if (answer.equals("no")) {
            System.out.println("I guessed wrong. What were you thinking
of?");
            String correctAnimal = scanner.nextLine().trim();

            System.out.println("Please give me a yes/no question that would
differentiate a " + correctAnimal + " from what I guessed.");
            String newQuestion = scanner.nextLine().trim();

            System.out.println("What is the answer for the " +
correctAnimal + "? (yes/no)");
            String yesForCorrectAnimal =
scanner.nextLine().trim().toLowerCase();

            expandTree(currentNode, correctAnimal, newQuestion,
yesForCorrectAnimal);
            System.out.println("Thanks! I'll remember that for next
time.");
        }

        else {
            System.out.println("Invalid response. Please answer 'yes' or
'no'");
        }
    }

    public static void main(String[] args) {
        System.out.println("Welcome to the very fun Animal Guessing
Game\n");
        //Initialise binary tree that will hold questions and guesses, also
check file
        BinaryTree<String> gameTree = null;
        File treeFile = new File(filePath);
```

```java
        //If tree exists, load it otherwise create new tree
        if (treeFile.exists()) {
            gameTree = loadTree();
        }
        if (gameTree == null) {
            gameTree = new BinaryTree<>();
            createTree(gameTree);
        }

        //While loop runs game, saves it and checks if they want to play
again, otherwise stops running code
        while (420==420) {
            playGame(gameTree);

            storeTree(gameTree);

            System.out.println("\nWould you like to play again? (yes/no)");
            String choice = scanner.nextLine().trim().toLowerCase();

            if ("no".equals(choice)) {
                System.out.println("Thanks for playing. Your game has been
saved.");
                break;
            }
        }
    }
}
```

Testing





As you can see from my testing, the game works if it first of all guesses correctly.
The part also works where new leaf's are created when what the user wants to guess isn't there and prompts them about a question for it. In this example the part that is added to the tree goes from:
Are you thinking of an animal -> (yes) -> Is it a mammal -> (yes) -> Does it like bones -> (yes) -> dog.