Assignment: Choose Your Own Adventure

### Learning Objectives

* Use tree traversals to construct and enumerate tree structures
* Navigate a tree structure using recursion.

### External Resources and Code

This assessment is open book-- you can look at text or online sources as you see fit. You may copy **small pieces of code** (1-3 lines that contain semicolons) from from the internet so long as you understand how it works and cite the source in a comment. You may use Apache commons, but no other external libraries. As per the academic honesty policy, you may not copy code (even with modification) from other students, or from other sites on the internet. You may not consult other students or look at their code. You may not share your code with other students. Any submission that violates the academic honesty policy will receive an automatic 0, and may also result in more severe repercussions.

### Overview and Submission Requirements

Your task is to create a Java program to assist in the creation and use of a choose your own adventure book using tree data structures.

### Choose Your Own Adventure Overview

A choose your own adventure (CYOA) book is a type of interactive novel designed to allow the reader to play as the protagonist of the story. The reader makes choices to navigate to different parts in the story, eventually leading to an ending. CYOA books have inspired simple story-based video games, as well as narrative fiction such as the Netflix experience “Bandersnatch.”

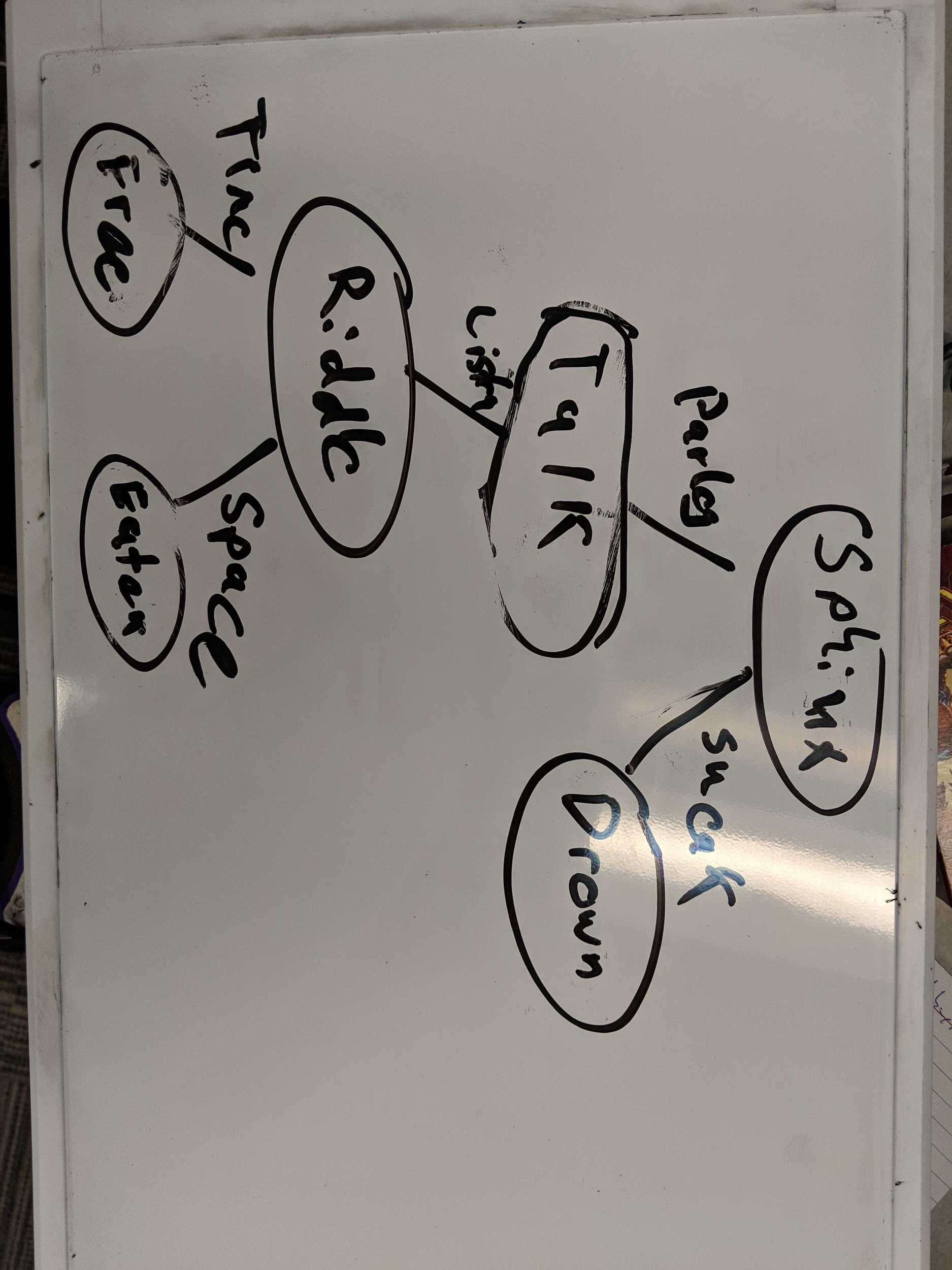
CYOA stories are typically organized in a binary tree structure of story nodes. The root node is the first page of the book, and leaf nodes are the various endings available in the story. Each internal story node of the tree has four elements:

1. A title
2. A text description of the current situation faced by the story’s protagonist.
3. One or two option descriptions, with an associated story node for each option.

Leaf nodes work similarly, but do not have any options for further play-- they simply describe how the story ends.

### CYOA Tree Example

As example, suppose you have the following adventure tree:



The story nodes might look as follows:

1. Title: Sphinx
   1. Description: Walking home one day, you approach a sphinx guarding a bridge in the distance. The water rushes quickly beneath the bridge. What do you do?
   2. Option: Parley with the sphinx.
      1. Leads to Talk, but reader does not know this.
   3. Option: Swim through the river.
      1. Leads to Drown, but reader does not know this.
2. Title: Talk
   1. Description: The sphinx’s pupils dilate as you approach the bridge. “Halt, traveller,” it purrs at you. You are transfixed by its voice, and can only listen.
   2. Option: Listen to the sphinx. (Leads to Riddle)
3. Title: Drown
   1. Description: The river is too strong! You are swept away by the current, never to be seen again.
4. Title: Riddle
   1. Description: “Answer my riddle correctly and you may pass. Answer incorrectly and I’ll gobble you up!” yawns the Sphinx. “What is harmless but can kill you?”
   2. Option: Answer “Time” (Leads to Free)
   3. Option: Answer “Space” (Leds to Eaten)
5. Title: Free
   1. Description: “You are correct! You may pass.” The sphinx steps aside and you continue home, unharmed.
6. Title: Eaten
   1. Description: “Wrong!” shouts the sphinx, and it eats you whole.

### File Format

For this assessment, CYOA books are stored .cyoa files with the following format:

1. Each line represents a single story node, with tab separated values representing the following node attributes. You may assume that text values do not contain tabs themselves.
   1. Node title.
   2. Node description
   3. Left branch choice description, or \* if there is no option. (NB: References to the next node are not directly stored here)
   4. Right branch choice description, or \* if there is no option. (NB: References to the next node are not directly stored here)
2. Empty nodes are indicated by four, tab-separated \*’s.
3. Nodes are stored using an pre-order tree traversal.

The above example has been provided as a file in the sample repository.

Note: For this assessment, you may use the Apache commons CSV library for parsing, located here: <https://commons.apache.org/proper/commons-csv/>  
Be sure that your project can build and run after cloning the repo. See this link for help using Apache commons CSV: <https://www.baeldung.com/apache-commons-csv>

### Program Requirements and Starter Code

Starter code has been provided in D2L for this assignment.

Use the starter code to implement the following methods in CYOAGame.java:

*/\*\**

*\* Factory method to create a game using the .cyoa format.*

*\**

*\* Note: This method MUST remain static! Implement a separate constructor method of your own design.*

*\**

*\** ***@param fileName****: The name of a file located in the root program directory*

*\* that contains the CYOA adventure in the .cyoa format.*

*\*\*/*

**public static** CYOAGame createGame(String fileName)

*/\*\**

*\* Return the title of the current node.*

*\*\*/*

**public** String getTitle()

*/\*\**

*\* Return the description of the current node*

*\**

*\** ***@return***

*\*/*

**public** String getDescription()

*/\*\**

*\* Return the description of one action for the current node. E.g. "Swim through the river"*

*\* Return null if that action does not exist.*

*\**

*\** ***@param choice*** *the choice direction in the binary tree*

*\*\*/*

**public** String getOptionDescription(ChoiceType choice)

*/\*\**

*\* Choose an option of the current node and move through the story.*

*\**

*\** ***@param option****: The option that the user chooses*

*\*\*/*

**public void** chooseOption(ChoiceType option)

*/\*\**

*\* Find and return the depth of a given node title. You may assume that the title is*

*\* unique.*

*\**

*\** ***@param title*** *title of the story node to find*

*\** ***@return*** *depth of the node with a given title, or -1 if the node does not exist*

*\*\*/*

**public int** getDepth(String title)

*/\*\**

*\* Returns the titles of each node in the CYOAGame, separated by spaces, using an*

*\* in-order traversal of the root node.*

*\**

*\** ***@return*** *a String of each title, separated by spaces, of an in-order traversal of*

*\* the book.*

*\*\*/*

**public** String getInOrderTitles()

### Marking Criteria

This assessment has a maximum of 75 points.

1. Functionality (50 points) Unit tests pass for the following methods:
   1. Parsing (20 points): createGame can parse any valid .cyoa file.
   2. Navigation (10 points): getters and chooseOption work for any valid .cyoa file.
   3. Depth (10 points): getDepth works for any valid .cyoa.
   4. Traversal (10 points): getInOrderTitles works for any valid .cyoa.
2. Design and Style (25 points)
   1. Trees are used effectively. (10 points)
   2. Recursion is used effectively. (10 points)
   3. Standard Java best practices are followed. (5 points)