Keaton Spiller  
03/06/2024  
CS 302 Winter 2024  
Program 4/5

**Riddle Specification**A diagram of a company

Description automatically generated

***Figure 1****: UML Describing Proposed Logic*

The Idea of my proposed specification is the game of riddles. There are three characters, the Riddler, Ridlee, and the Hint Guide. The game is about answering questions, and as you answer questions correctly you go down different paths. Your score dictates which path you follow, and the path you end depends on your ability to guess correctly. I am planning on starting the user off with a score of 50. When the user reaches below 0, above 100, or runs out of questions, the game ends.

This game will be built using a Binary Search Tree Architecture, the application from a menu system will prompt the user, and run the program, and a separate exception handling class that will catch any place in the program that could result in a segmentation fault, error, or invalid/unwanted response. The three characters will each have their own class, and each hold their own responsibility to play.

The main functions of the code will follow the diagram from above. Read questions from the tree based on the user score, prompt the Riddler, give the user the option for a hint, but deduct from their score based on if they take a hint. Validate the users answer with the correct answer and update the score + or – based on if they got the correct answer. If the game is over, and there are either no more questions to ask, the user has a high enough score to win, or they have too low and lose the game, otherwise this process repeat for the next question.

Although the user doesn’t have an inventory similar to a medieval, fantasy or adventure game, they have a series of questions and answers to help them navigate the riddles. In general I decided to keep this program simple, and avoid using multiple trees. Although multiple trees could have created more variation in the possibilities and added more depth into the game my main focus for this assignment was to understand the high level schematic process creating a clean black box UML architecture, and the understanding what I am programming before I begin.

This process includes unit testing, before code has been written into implementation I’ll be using pytest in a black box suite to test the flow diagrams above and validate which inputs and outputs should result from each function given edge case test scenarios, without having written the underlying code.

The data inserted into the Tree, and the functions that run the application will have to be validated to have appropriate input. Unit tested to continue to follow the same rules and regulations and stay the same when large changes in code are made. Ideally this will keep us accountable for unknown bugs in logic, and keep the code up to industry standards.

* Is the data null?
* Did the user answer with a string?
* Did the string get normalized appropriately to compare with the answer?
* Is the score of the game within 0 and 100?
* Is the score of the game positive?
* When we ask for a question, are we given a question?
* Are we at a leaf of the tree?
* Do we win the game when the score is greater than 100
* Lose when less than 0
* Or End when we ran out of questions.

Besides these unit tests I will ideally check each function to validate when I enter in a true argument value I should receive a true output, and vice versa testing invalid arguments, and catching these invalid arguments in the appropriates places.