Programming Assignment Unit 1

My Rhyme game is a text-based game that utilizes core programming concepts for user interaction, random number generation, and basic game logic. Here's a breakdown of the technical aspects:

- User Interaction: The `Scanner` class from the `java.util` package (Eck, Chapter 2, page 48)[^1] serves as the foundation for user input. The program utilizes the `scanner.nextLine()` (Eck, Chapter 11, page 572)[^1] method to capture the player's guess for the rhyming word.
- Random Word Selection: The Random class provides functionalities for generating pseudo-random numbers. Here, an instance of Random (Eck, Chapter 5, page 226)[^1] is used to select a random index within the predefined wordList array. This retrieved index corresponds to the word presented to the player for rhyming.
- Game Loop and Conditional Statements: The core game logic is implemented within a while loop (Eck, Chapter 3, page 76)[^1]. This loop continues as long as the player chooses to keep playing (indicated by a "y" input). Inside the loop, an if statement (Eck, Chapter 3, page 79)[^1] checks if the user's guess rhymes with the chosen word (determined by the isRhyme function). Based on the outcome, the program displays appropriate messages and updates the score.
- Basic Rhyming Check: The `isRhyme` function demonstrates a simplified approach to identifying rhyming words. It employs string manipulation methods (toLowerCase and substring) (Eck, Chapter 2, page 34)[^1] to compare the last two characters of the player's guess and the chosen word (converted to lowercase for case-insensitive comparison). This is a basic implementation, and more sophisticated algorithms can be employed for advanced rhyming detection.

Code comments are included to help understand the created logic.

```
// java.util.Scanner is for user input
// and java.util.Random to choose random words.
// Eck, Chapter 2, page 48
import java.util.Scanner;
// Eck, Chapter 5, page 226
import java.util.Random;

public class RhymeRider {

  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    Random random = new Random();

    // Define a list of words. Could be improved
    String[] wordList = {
      "glove",
      "love",
      "cat",
      "hat",
      "tree",
```

```
    "sea",
    "fun",
    "sun",
    "age",
    "wage",
  };

  int score = 0;
  int totalQuestions = 0;
  boolean keepPlaying = true;

  System.out.println("Let's play a rhyme game!");
  // The game loop keeps running
  // until the user decides to stop (keepPlaying = false).
  while (keepPlaying) {
    // In each loop, a random word is chosen,
    // and the user is prompted to guess a rhyming word.
    // A basic isRhyme function checks if the last
    // two characters of the words match (a more sophisticated
    // rhyming check can be implemented).
    int randomIndex = random.nextInt(wordList.length);
    String word = wordList[randomIndex];

    System.out.println("What rhymes with " + word + "?");
    String guess = scanner.nextLine();

    // Check if the guess rhymes
    // Eck, Chapter 3, page 79
    if (isRhyme(word, guess)) {
      System.out.println("Great rhyme!");
      // The score is kept track of,
      // and the game displays messages based on the guess.
      score++;
      totalQuestions++;
    } else {
      System.out.println("Hmm, not quite. Try again!");
      totalQuestions++;
    }

    System.out.println("Continue riding? (y/n)");
    String choice = scanner.nextLine().toLowerCase();
    keepPlaying = choice.equals("y");
  }

   double percentage = (double) score / totalQuestions * 100;
  System.out.println("Thanks for playing Rhyme Game! Your've made "
  + score + " out of "
  + totalQuestions +
  " (" + percentage + "%)");
}

/**
 * Checks if two words rhyme (basic implementation)
```

```
     *
     * @param word1 First word
     * @param word2 Second word
     * @return True if the words rhyme, false otherwise
     */
    public static boolean isRhyme(String word1, String word2) {
      // Simple check based on the last two characters (can be improved)
      // Eck, Chapter 2, page 34
      return word1.toLowerCase().
             endsWith(word2.
             toLowerCase().
             substring(word2.length() - 2));
    }
}
```

Code run instructions

As I'm a linux user, to run this code I created a `gabriel_game.java` file, copy and paste the code, and run it using the `java gabriel_game.java` command.

[^1]: Eck, David J. (2022). Introduction to Programming Using Java. https://math.hws.edu/javanotes/index.html