**SimpleRandomSentences Documentation**

**Abstract**

The SimpleRandomSentences project is a Java application that leverages recursive syntax rules to generate random sentences. The program follows a set of defined grammar rules expressed in BNF (Bachus-Naur Form) style, providing a playful and illustrative example of sentence structure. This documentation aims to provide a comprehensive understanding of the project, including its purpose, usage, configuration options, internal structure, grammar rules, and considerations on recursion.

**1. Introduction**

**1.1 Purpose**

The purpose of the SimpleRandomSentences project is to demonstrate the application of recursive syntax in generating random sentences. The program serves as an educational tool to showcase how programming concepts, such as recursion and grammars, can be applied to mimic natural language structures.

**1.2 Features**

* Recursive sentence generation based on defined grammar rules.
* Configuration options for controlling recursion depth and the number of generated sentences.
* Randomized selection of words from predefined lists for nouns, verbs, adjectives, etc.

**2. Configuration**

You can configure the behavior of the program through the following parameters:

* **MAX\_DEPTH:** Set a maximum recursion depth to control sentence complexity and prevent infinite recursion.
* **numSentences:** Adjust the number of sentences to be generated in each run of the program.

**4. Internal Structure**

**4.1 Main Class**

* **Class:** **SimpleRandomSentences**
  + **Methods:**
    - **main**: Initializes the generation process and prints the generated sentences.

**4.2 Randomization**

* **Class Variable:** **Random random**
  + Utilized for generating random choices.

**4.3 Sentence Generation Methods**

* **Methods:**
  + **generateSentence(int depth)**
  + **generateSimpleSentence(int depth)**
  + **generateNounPhrase(int depth)**
  + **generateVerbPhrase(int depth)**
  + **generateConjunction()**

**4.4 Word Generation Methods**

* **Methods:**
  + **generateProperNoun()**
  + **generateCommonNoun()**
  + **generateDeterminer()**
  + **generateAdjective()**
  + **generateIntransitiveVerb()**
  + **randomItem(String [] list)**

**4.5 Utility Methods**

* **Methods:**
  + **randomItem(String [] list)**: Selects a random item from a given array.

**5. Grammar Rules**

The program follows a set of grammar rules for generating sentences. The rules define the structure of sentences, incorporating elements such as nouns, verbs, adjectives, conjunctions, and more. The provided grammar rules are expressed in BNF style.

**6. Notes on Recursion**

The program utilizes recursion to create sentences, providing a natural and expressive syntax. To prevent infinite recursion and manage sentence complexity, the **MAX\_DEPTH** parameter is introduced. Careful consideration should be given when adjusting this parameter to balance the desire for diverse sentences with the need to avoid excessively long or complex outputs.

**7. Program**

import java.util.Random;

public class SimpleRandomSentences {

    private static Random random = new Random();

    private static final int MAX\_DEPTH = 15; // Set a maximum recursion depth

    public static void main(String[] args) {

        int numSentences = 15; // You can adjust the number of sentences to generate

        for (int i = 0; i < numSentences; i++) {

            String sentence = generateSentence(0);

            System.out.println(sentence);

            try {

                Thread.sleep(2000);

            }

            catch (InterruptedException e) {

            }

        }

    }

    private static String generateSentence(int depth) {

        if (depth > MAX\_DEPTH || random.nextBoolean()) {

            return generateSimpleSentence(depth);

        } else {

            return generateSimpleSentence(depth) + " " + generateConjunction() + " " + generateSentence(depth + 1);

        }

    }

    private static String generateSimpleSentence(int depth) {

        return generateNounPhrase(depth) + " " + generateVerbPhrase(depth);

    }

    private static String generateNounPhrase(int depth) {

        if (random.nextBoolean()) {

            return generateProperNoun();

        } else {

            String determiner = generateDeterminer();

            String adjective = random.nextBoolean() ? " " + generateAdjective() : "";

            String commonNoun = generateCommonNoun();

            String whoClause = random.nextBoolean() ? " who " + generateVerbPhrase(depth) : "";

            return determiner + adjective + " " + commonNoun + whoClause;

        }

    }

    private static String generateVerbPhrase(int depth) {

        if (random.nextBoolean()) {

            return generateIntransitiveVerb();

        } else if (random.nextBoolean()) {

            return "is " + generateAdjective();

        } else {

            return "believes that " + generateSimpleSentence(depth + 1);

        }

    }

    private static String generateConjunction() {

        String[] conjunctions = {"and", "or", "but", "because"};

        return randomItem(conjunctions);

    }

    private static String generateProperNoun() {

        String[] properNouns = {"Fred", "Jane", "Richard Nixon", "Miss America"};

        return randomItem(properNouns);

    }

    private static String generateCommonNoun() {

        String[] commonNouns = {"man", "woman", "fish", "elephant", "unicorn"};

        return randomItem(commonNouns);

    }

    private static String generateDeterminer() {

        String[] determiners = {"a", "the", "every", "some"};

        return randomItem(determiners);

    }

    private static String generateAdjective() {

        String[] adjectives = {"big", "tiny", "pretty", "bald"};

        return randomItem(adjectives);

    }

    private static String generateIntransitiveVerb() {

        String[] intransitiveVerbs = {"runs", "jumps", "talks", "sleeps"};

        return randomItem(intransitiveVerbs);

    }

    private static String randomItem(String[] list) {

        return list[random.nextInt(list.length)];

    }

}

**8.Output**

