API Guide for D&D Character Simulator

Introduction

The D&D Character Simulator API provides functions to retrieve character details like name, class, ancestry, and various attribute scores (strength, intelligence, etc.). This guide outlines the available functions and their expected outputs, allowing you to interact with and customize your D&D character programmatically.

Available API Functions

Basic Information

These functions provide general character information.

- `get_character_name()`: Returns the name of the character as a string.
- `get_character_class()`: Returns the character's class (e.g., "Wizard", "Fighter") as a string.
- `get_character_ancestry()`: Returns the character's ancestry (e.g., "Human", "Elf", "Dwarf") as a string.

Character Attributes

These functions return attribute scores for the character, with values ranging from 1 to 20. You can use these scores to calculate modifiers, which help determine a character's strengths and weaknesses.

- `get_strength()`: Returns the strength score as an integer.
- `get_dexterity()`: Returns the dexterity score as an integer.
- `get_constitution()`: Returns the constitution score as an integer.
- `get_intelligence()`: Returns the intelligence score as an integer.
- `get_wisdom()`: Returns the wisdom score as an integer.
- `get_charisma()`: Returns the charisma score as an integer.

Attribute Modifiers

Each attribute has an associated modifier, calculated using the formula:

$$\text{modifier} = \frac{\text{score} - 10}{2}$$

You can write a function in Rust to calculate modifiers for each attribute score using this formula.

Example Usage

Here's an example of how to use the API functions in your code:

```
fn main() {
    let name = get_character_name();
    let class = get_character_class();
    let ancestry = get_character_ancestry();
    let strength = get_strength();
    let dexterity = get_dexterity();

    println!("Character Name: {}", name);
    println!("Class: {}", class);
    println!("Ancestry: {}", ancestry);
    println!("Strength: {}", strength);
    println!("Dexterity: {}", dexterity);
}
```

Additional Notes

- Each function returns a single value based on pre-set or randomized attributes. - Use the character attributes and classes to design unique abilities or simulate game scenarios.

This API guide provides the basic building blocks to interact with your D&D character through Rust code, allowing for both fun and learning opportunities in programming.