UML Class Diagram

Associations
- Player submits Guess → A Player makes multiple Guess objects.
- Game provides Feedback → Game calculates bulls & cows for Guess.
- GameManager creates Player → GameManager is responsible for player creation.
- GameManager starts and tracks multiple Games → GameManager manages multiple Game instances.

Aggregation
- GameManager manages Leaderboard → Leaderboard exists separately but is managed by GameManager.
- Game contains multiple Guesses → Guess objects exist independently but are tracked by Game.
- Leaderboard tracks multiple Players → Leaderboard maintains Player rankings.

Composition
- Game owns Player → Player is directly tied to a Game, and they exist together.

Generalization
- Different game modes inherit from Game → NumberGame and CharacterGame extend Game.

C GameManager

□ active_games: list[Game]
□ leaderboard: Leaderboard

● create_player(name: str): Player
● start_game(game_type: str, difficulty: int, player: Player): Game
● show_leaderboard(n: int)

1   1   1
manages   manages
*   1

A Game

□ mode: str
□ difficulty: int
□ secret_sequence: list[str]
□ player: Player
□ guess_history: list[Guess]
□ attempts: int

● validate_guess(guess: str): bool
● process_guess(guess: str): Guess
● is_game_over(): bool
● generate_secret_sequence(): list[str]

creates

C Leaderboard

□ rankings: dict[str, int]

● update_score(player: Player, score: int)
● get_top_players(n: int): list[Player]

1

contains

1

plays

tracks

1   *   *   1

C CharacterGame

□ word_list: dict[int, list[str]]

● generate_secret_sequence(): list[str]
● validate_guess(guess: str): bool

C NumberGame

● generate_secret_sequence(): list[str]
● validate_guess(guess: str): bool

C Player

□ name: str

● make_guess(guess: str): Guess

1

makes

*   *

C Guess

□ guess: str
□ bulls: int
□ cows: int
□ timestamp: datetime