**Architecture Overview**

**Key Components and Responsibilities**

* **NGram**
  + **Purpose:** Immutable value object representing a word (as a sequence of characters).
  + **Key Operations:** get(index), size(), equals(), and hashCode().
  + **Relationship:** Used by all other classes; no inheritance.
* **Corpus**
  + **Purpose:** Dictionary of valid NGrams (all same length).
  + **Key Operations:** Filtering (via feedback), scoring guesses (worst-case and average-case), best-guess selection.
  + **Relationship:** Contains a collection of NGrams.
* **Filter**
  + **Purpose:** Encapsulates feedback rules (constraints) derived from comparing a guess to the key (e.g., exact match, misplaced, absent).
  + **Key Operations:** test(NGram) to decide if a candidate word satisfies the feedback.
* **NGramMatcher**
  + **Purpose:** Implements multi-pass matching logic (exact, misplaced, and absent) between two NGrams, producing a Filter.
  + **Key Operations:** match() (split into helper methods for each pass).
  + **Improvements:** Each pass is refactored into its own small method so each has low cyclomatic complexity (≤ 4).

A diagram of a flowchart

AI-generated content may be incorrect.

**Defensive Programming and Code Quality**

* **Defensive Checks:**
  + Every public method validates inputs using Objects.requireNonNull(...).
  + Methods throw clear exceptions (e.g., IllegalStateException when trying to operate on an empty Corpus).
* **Immutability:**
  + NGram objects and final data structures (e.g., internal collection in Corpus) are immutable.
  + Exposed collections are unmodifiable.
* **Low Complexity:**
  + Methods (especially in NGramMatcher) are split into small helpers (exact, misplaced, and absent phases) to keep cyclomatic complexity low (≤ 4).
  + DRY principle applied by reusing match() in scoring logic; no duplicated code.

**Testing Strategy**

* **Unit Tests:**
  + Test individual classes (NGram, Corpus, Filter, NGramMatcher) with edge cases, valid inputs, and error conditions.
  + Verify that each method’s defensive checks work (e.g., null inputs, empty corpus).
* **Integration Tests:**
  + Simulate a game scenario: apply feedback from NGramMatcher and then narrow down the Corpus to see if correct words remain.
* **Stress Tests:**
  + Run with a large corpus to ensure performance is acceptable.
  + Simulate repeated game iterations to check memory and state consistency.

**Pseudocode Summary for NGramMatcher.match()**

function match(guess, target) -> Filter:

if guess.length != target.length:

return Filter.FALSE

for each index i:

if guess[i] == target[i]:

mark exact match at i

add exact filter for i

for each index i not matched:

for each index j not matched in target:

if guess[i] == target[j]:

mark misplaced match for i

add misplaced filter for i

break

for each index i not matched:

add absent filter for guess[i]

combine all filters with logical AND

return final Filter