Word N-Grams

Summary

Duke
UNIVERSITY

WordGram

- Extends predictive, Markov text generation
  - Word at-a-time instead of character at-a-time
  - Implements same IMarkovModel interface
- Developed MarkovWordOne first
  - Easy transition from MarkovModel with letters
- Take a small step in changing design firsts
  - After that works, take another step
  - Keep using seven-step process as warranted!

WordGram Internals

- Developed MarkovWordOne with strings, then implemented WordGram for 2,3, …,N
- Familiar methods and design, new classes
  - Tested WordGram independently of Markov, but designed with Markov in mind
  - Tested `.toString()` and constructor first
  - Make sure `.wordAt()` and `.length()` work
- Understand `.equals()`, `.hashCode()`
  - Needed for more advanced design

New Ideas

- Throw exceptions when "bad" calls made
  - Can't index array with -1 or array.length
  - Make WordGram behave similarly, like String
  - Must understand exceptions
- Writing functions you may not call!
  - Other methods will call them: **`.toString()`**
  - Inserting into HashMap calls both **`.hashCode()`** and **`.equals()`**