Word N-Grams

Equals and HashCode Methods

Duke
UNIVERSITY

Developing WordGram

- We'll use WordGramTester.java in developing, coding, and testing **`WordGram`**
  - We'll test **`.toString()`** and constructor, code already written
  - We'll see why **`.equals()`** is needed, developed for **`MarkovWord.getFollows()`**
- Enough for us to implement MarkovWord, but one advanced method to go!
  - Implement **`.hashCode()`**, work with HashMap

Writing .equals()

- You've seen why **`.equals()`** with Strings
  - Can't use ==, that tests something different
- Writing **`.equals()`**: Java requirements

```
public boolean equals(Object o) {
    WordGram other = (WordGram) o;
    // compare me to other
    if (length() != other.length()) return false;
    …
```

Writing .equals()

- You've seen why **`.equals()`** with Strings
  - Can't use ==, that tests something different
- Writing **`.equals()`**: Java requirements
  - Parameter has type Object

```
public boolean equals(Object o) {
    WordGram other = (WordGram) o;
    // compare me to other
    if (length() != other.length()) return false;
    …
```

Writing .equals()

- You've seen why **`.equals()`** with Strings
  - Can't use ==, that tests something different
- Writing **`.equals()`**: Java requirements
  - Parameter has type Object
  - Cast object to type in class being designed

```
public boolean equals(Object o) {
    WordGram other = (WordGram) o;
    // compare me to other
    if (length() != other.length()) return false;
    …
```

Writing .equals()

- You've seen why **`.equals()`** with Strings
  - Can't use ==, that tests something different
- Writing **`.equals()`**: Java requirements
  - Parameter has type Object
  - Cast object to type in class being designed

```
public boolean equals(Object o) {
    WordGram other = (WordGram) o;
    // compare me to other
    if (length() != other.length()) return false;
    …
```

Looking Ahead

- We could use a HashMap to store the follow lists for each *key*: "the herd then …"
  - Works with letter or word Markov
  - Avoid scanning the text many times for "he"

Looking Ahead

- We could use a HashMap to store the follow lists for each *key*:

  - Works with letter or word Markov
  - Avoid scanning the text many times for "he"

Looking Ahead

- We could use a HashMap to store the follow lists for each *key*:

  - Works with letter or word Markov
  - Avoid scanning the text many times for "he"

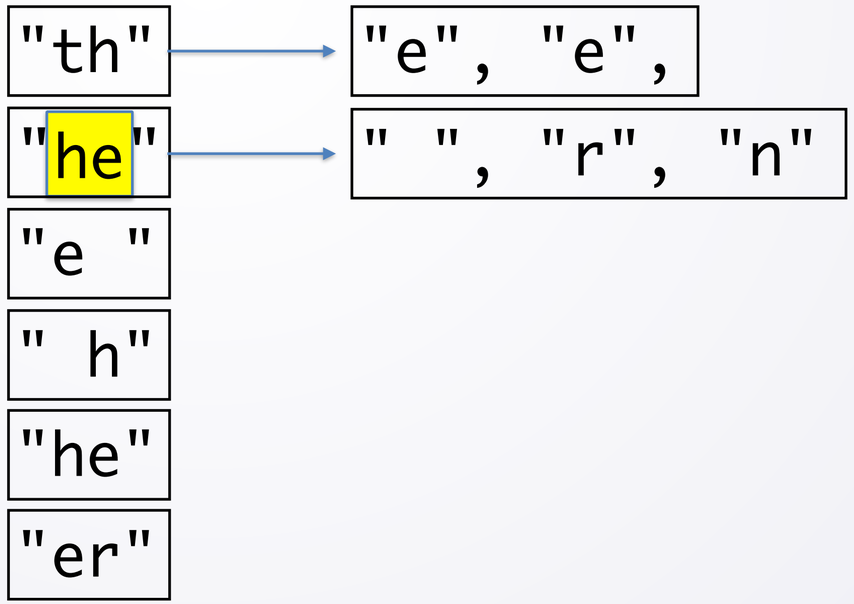
Looking Ahead

- We could use a HashMap to store the follow lists for each *key*:

  - Works with letter or word Markov
  - Avoid scanning the text many times for "he"

Looking Ahead

- We could use a HashMap to store the follow lists for each *key*: "the herd then …"
  - Works with letter or word Markov
  - Avoid scanning the text many times for "he"
  - look up in map!
  - Could be efficient
  - Need .hashCode



Values in HashMap

- Must convert object to integer: **hash code**
  - This "hash code" is like index into map
  - Simple idea? hash code is 17 for everything
- With .equals working, everything correct
  - Bad performance, everything in same "bucket"
  - Details in further studies
- Simple .hashCode
  - Add string hash codes