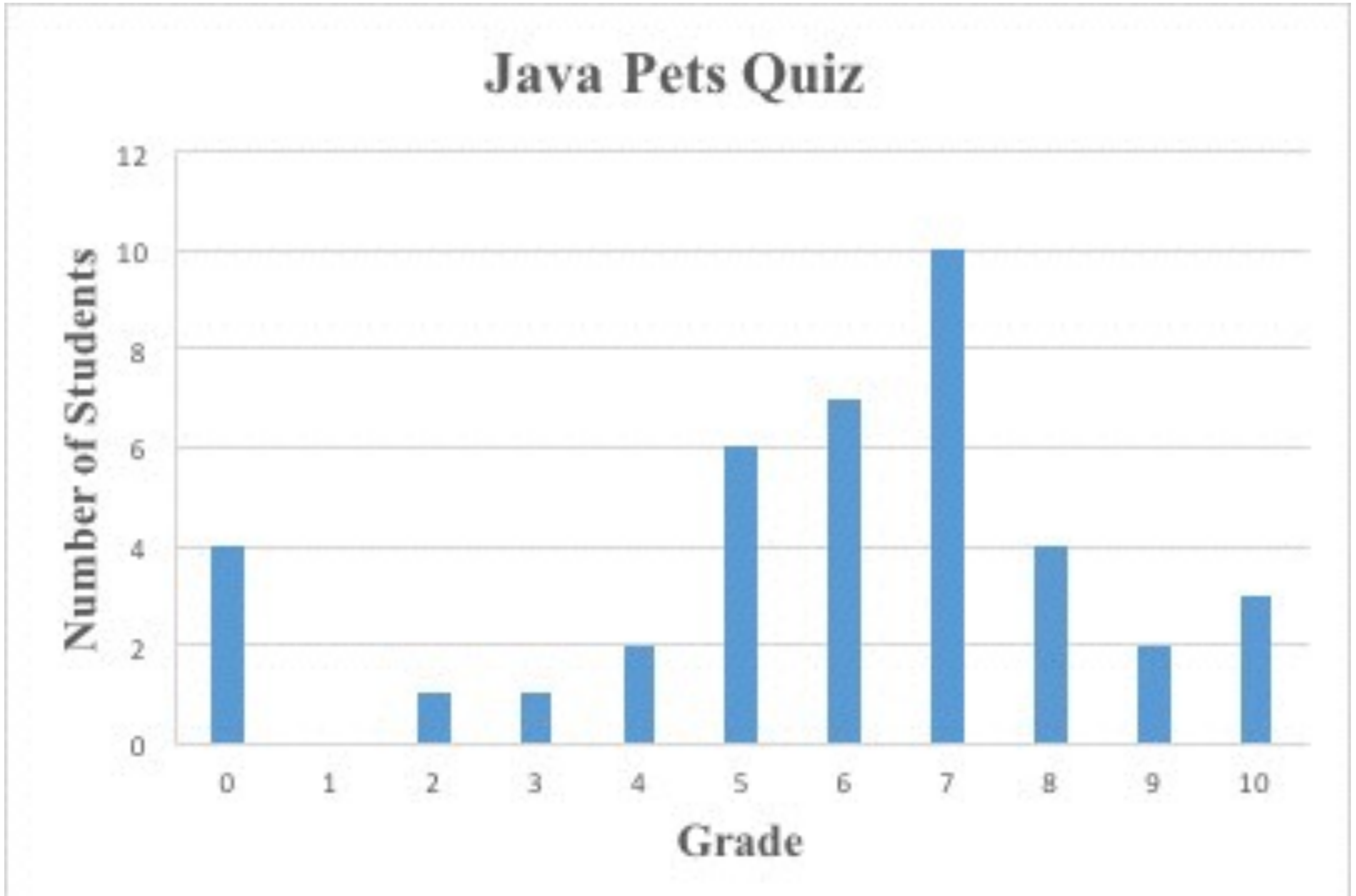
CS 2113
Software Engineering

Lecture 8: Practice, Practice!

Professor Tim Wood - The George Washington University

Last Week's Quiz

- Store two types of pets---cats and dogs
  - When you create a pet, constructor takes a name
  - All pets have a printName() function that prints the name
  - All pets have a makeNoise() function
    - Cats say "meow" and dogs say "woof"



Good OOP

- Find redundancy in a problem description
  - Cats and Dogs sound very similar —> Use a parent class to provide the basic functionality
- Child classes only need to add new functionality
  - Cats need a different constructor
  - Cats and Dogs need different makeNoise functions
- ArrayLists can store multiple types of objects
  - ArrayList<Pet> list = new ArrayList<Pet>();
  - list.add(new Dog(“Fido”));
  - list.add(new Cat(“Mowzer”));

Exam!

Next Week
Wednesday October 26th in class

What have we covered?

- C and Java Basic Programming
  - Syntax and structure
- Memory Management in C and Java
  - Stack vs Heap
  - Pointers in C
  - Garbage collection in Java
- Objected Oriented Design
  - Inheritance, Polymorphism
  - Organizing program structure with UML Class diagrams
- Java Data Structures

How to Study

- Review the lecture notes
    - Focus on concepts, not low level syntax or details
- Review the labs and in-class worksheets
    - You should be familiar with the Java library functions and classes used in the demos
    - Check out the self-quiz sections on each worksheet
- Look at the books
    - Chapters 1-9 in Head First Java (not interfaces or abstract)
    - Parts of 1-6, 8-9, 12-13, 17 in Practical C Programming
- Look over the exercises
    - Coding on exam may be similar (but shorter)

Exam Components

- **Short answer**: explaining concepts

- **Programming**: applying concepts
  - Coding in Java and C using CodeAnywhere
  - Strict academic honesty requirements

- Goal: 1.5 hour exam

On paper...

- Differences between C and Java
- Memory and pointers
- Draw a UML diagram to represent a complex piece of software
- Explain some principles of OOP

Programming...

- Mini-programs:
  - codingbat.com style programs to ensure you can do simple things with loops/logic/arrays/strings
- Basic data structures like Linked Lists
- Manipulate arrays and strings
- Use inheritance, Java Collections

You don't need to know

- Syntax to read / write to files
- You will get:
  - C reference sheet
  - Java reference sheet
- **You can bring:**
  - One sheet (front and back) of handwritten notes

One more data structure to learn…

Data Collections

The Java library already includes many data structures called Collections

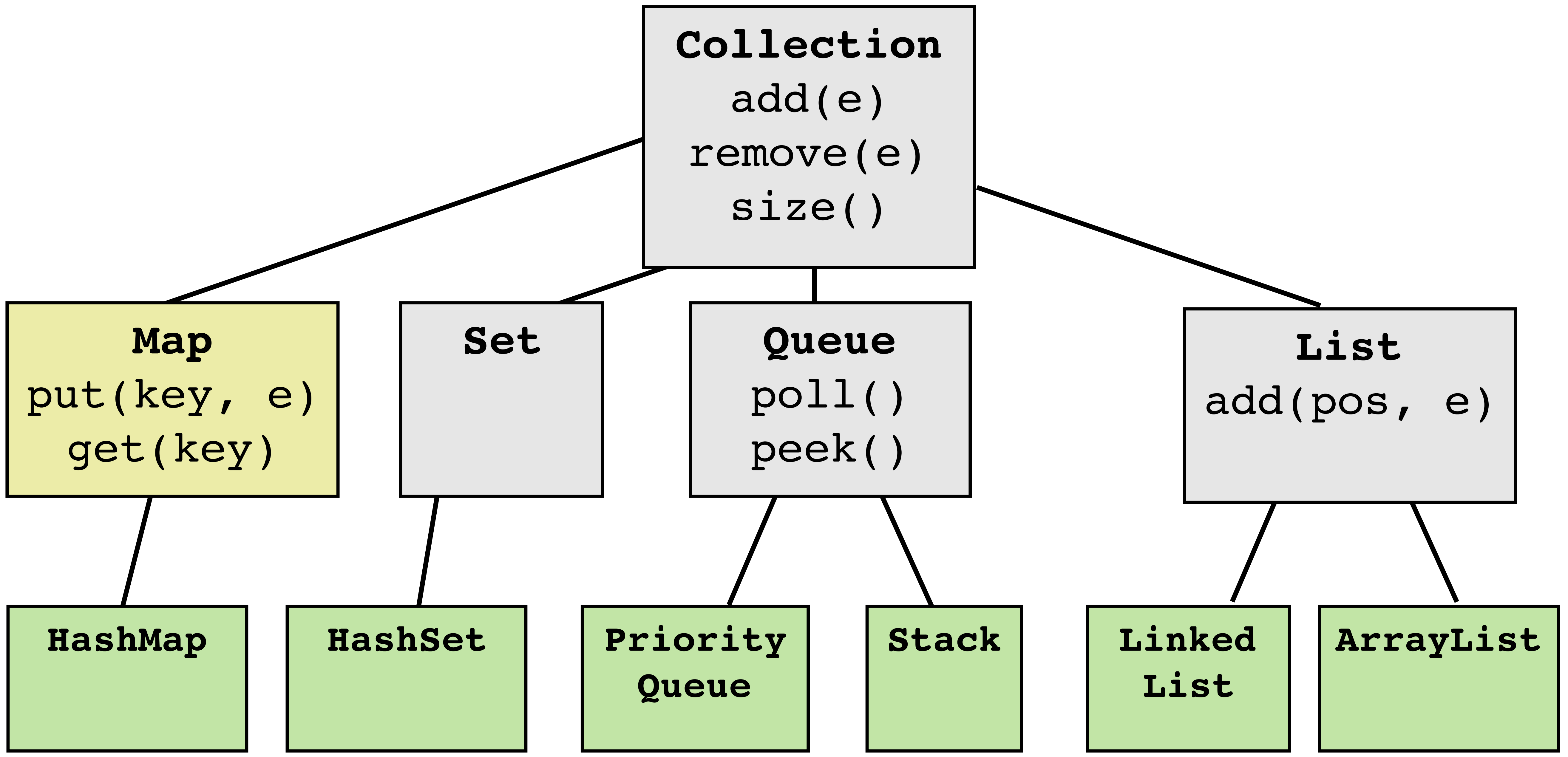
Java supports:

- **Sets**: no-duplicates allowed, unordered
  - (a, b, c) === (b, c, a)
- **Lists**: ordered list of elements
  - (a, b, c, b)
- **Queues**: ordered list of elements waiting for processing
  - Optimized for accessing the ends of the list
- **Maps:** store both a "key" and a matching "value" entity
  - Useful for looking up an object by a unique name, not position

Many different implementations:

- HashSet, ArrayList, LinkedList, Vector, PriorityQueue, Stack

Java Collections Class Tree



(plus many other implementations)

HashMap

- A type of **`Map`** - stores {**Key, Value**} pairs
    - Lookup a **Value** object by presenting a **Key** object
- Need to be careful if value is an int, float, or double
    - These are basic types in Java, not objects!
    - Need to use Integer, Float, or Double objects

```
HashMap<String, Integer> hmap = new HashMap<String, Integer>();
hmap.put("Rahul", 20);
hmap.put("Chen", 15);
int c = hmap.get("Chen");
c++
hmap.put("Chen", c); // replaces old value
```

- Example usage:
    - http://beginnersbook.com/2013/12/hashmap-in-java-with-example/

Why are Hash tables Magic?

- `ArrayList<Student>` stores every GW student (about 25,000 entries)
- How long does it take to find yourself in the list by searching for your ID #?
- What if you use `HashMap<ID#, Student>`?
- A LOT of CS interview questions have to do with hash tables

Let’s do something useful!

Debate Analysis

- There is an election coming up!
- Let's get to know the candidates…

- The Policy Institute you work for has a transcript of the first presidential debate. They want you to present the transcript in a visual way and analyze it for information about the different candidates—how much did each of them talk? What kinds of words did they use? What terms did they think were the most important?

Debate Analysis

- Print out the debate with colors indicating party of the speaker. Or print out only the lines said by a specific candidate.
- How many words were said by each candidate?
- What was the complexity of words used by each candidate?
  - Provided function will give you the "grade level" of a sentence
- What were the most frequent words used by each candidate?
- **Use good OOP practices and store data so we can display the debate in different ways**

Data Source

Text file containing transcript of the first presidential debate

- Some new lines indicate a new speaker with "Name:"

```
HOLT: Let me let Secretary Clinton get in here.
CLINTON: Well, let's stop for a second and
remember where we were ....
In fact, Donald was one of the people who rooted
for the housing crisis. He said, back in 2006....
TRUMP: That's called business, by the way.
CLINTON: Nine million people — nine million
people lost their jobs. Five million people lost
their homes. And $13 trillion in family wealth
was wiped out.
```

Parsing the file

Split the file up by speaker

- Associate different lines to the candidate (or moderator) who said them
- Print them out in color

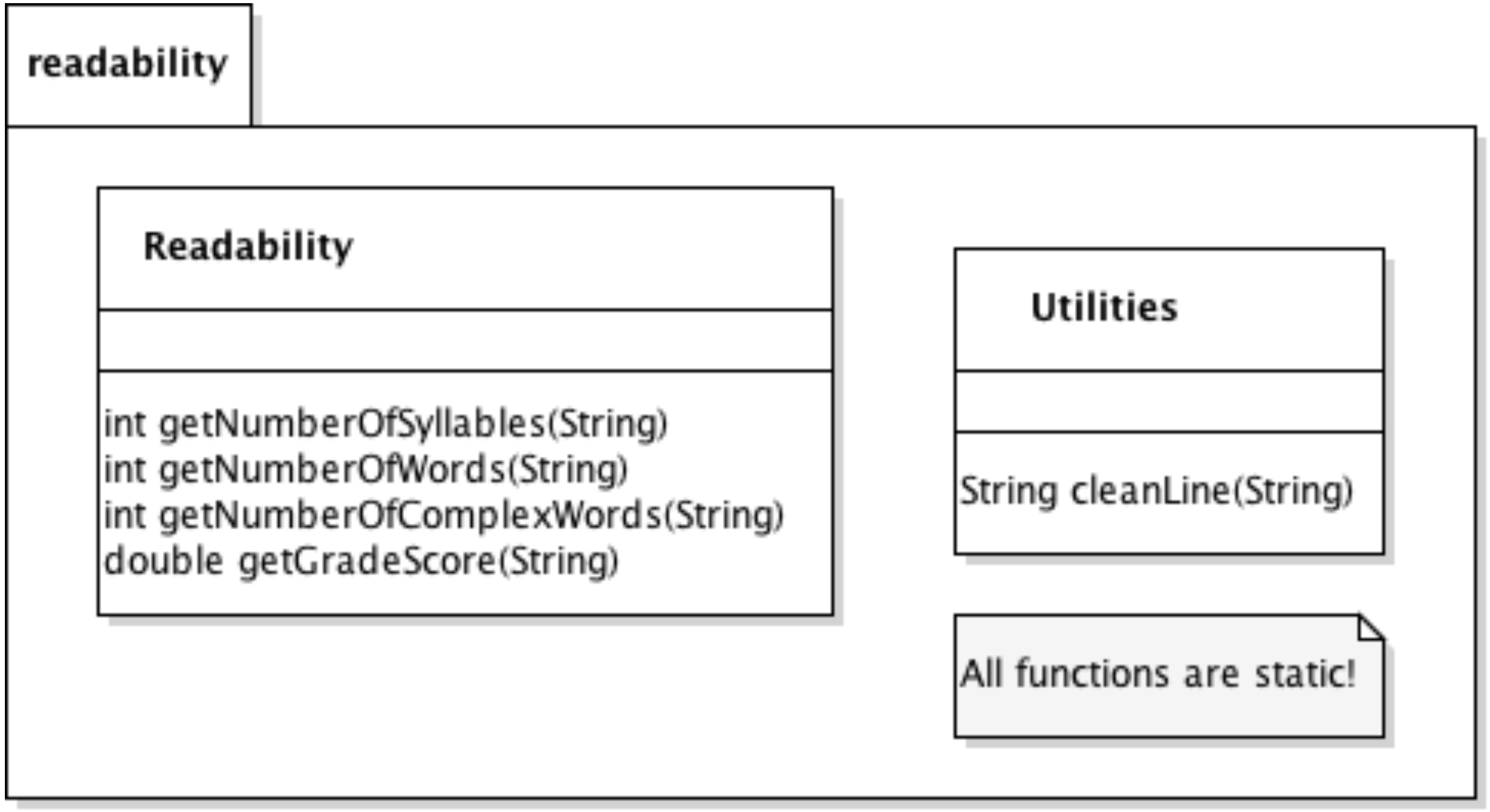
```
HOLT: Let me let Secretary Clinton get in here.
CLINTON: Well, let's stop for a second and
remember where we were ....
In fact, Donald was one of the people who rooted
for the housing crisis. He said, back ...

TRUMP: That's called business, by the way.

CLINTON: Nine million people — nine million
people lost their jobs. Five million people lost
their homes. And $13 trillion in family wealth
was wiped out.
```

Quotes

Analyzing Text

readability package has some useful functions for analyzing sentences

- All functions in these classes are static!
- Utilities.cleanLine(string) will remove all odd characters


readability
Readability
int getNumberOfSyllables(String)
int getNumberOfWords(String)
int getNumberOfComplexWords(String)
double getGradeScore(String)
Utilities
String cleanLine(String)
All functions are static!

Roles

Data I/O

- Input: text file
- Get strings from the debate file and figure out which candidate said each one.
- Print strings with color coding

Basic Word Stats

- Input: String (may want to change this later)
- Count the number of words in the string
- Use Readability class to find complexity

Favorite Words

- Input: String (may want to change this later)
- Count the frequency of all words
- More advanced: ignore stop words and use word stem

Architect

Design UML class diagram to represent program

Must use at least 3 classes:

- **Debate**: represents overall debate, reads from file, has main()
- **Quote**: a possibly multi-line quotation said by a candidate
- **Speaker**: represents a candidate and what they have said

What Java Collections data structures will you use?

Meet with at least one other Architect to discuss plan

Get approval from Josh, Bo, or Prof. Wood!

Get starter code

FIRST: Architect goes here and creates a new team

https://classroom.github.com/group-assignment-invitations/d168ab97059eb00210712a9b224b074e

THEN: other students join the team by name with the same link