CompSci 201, First Day
What is Computer Science?

What is computer science?

Computers are machines…

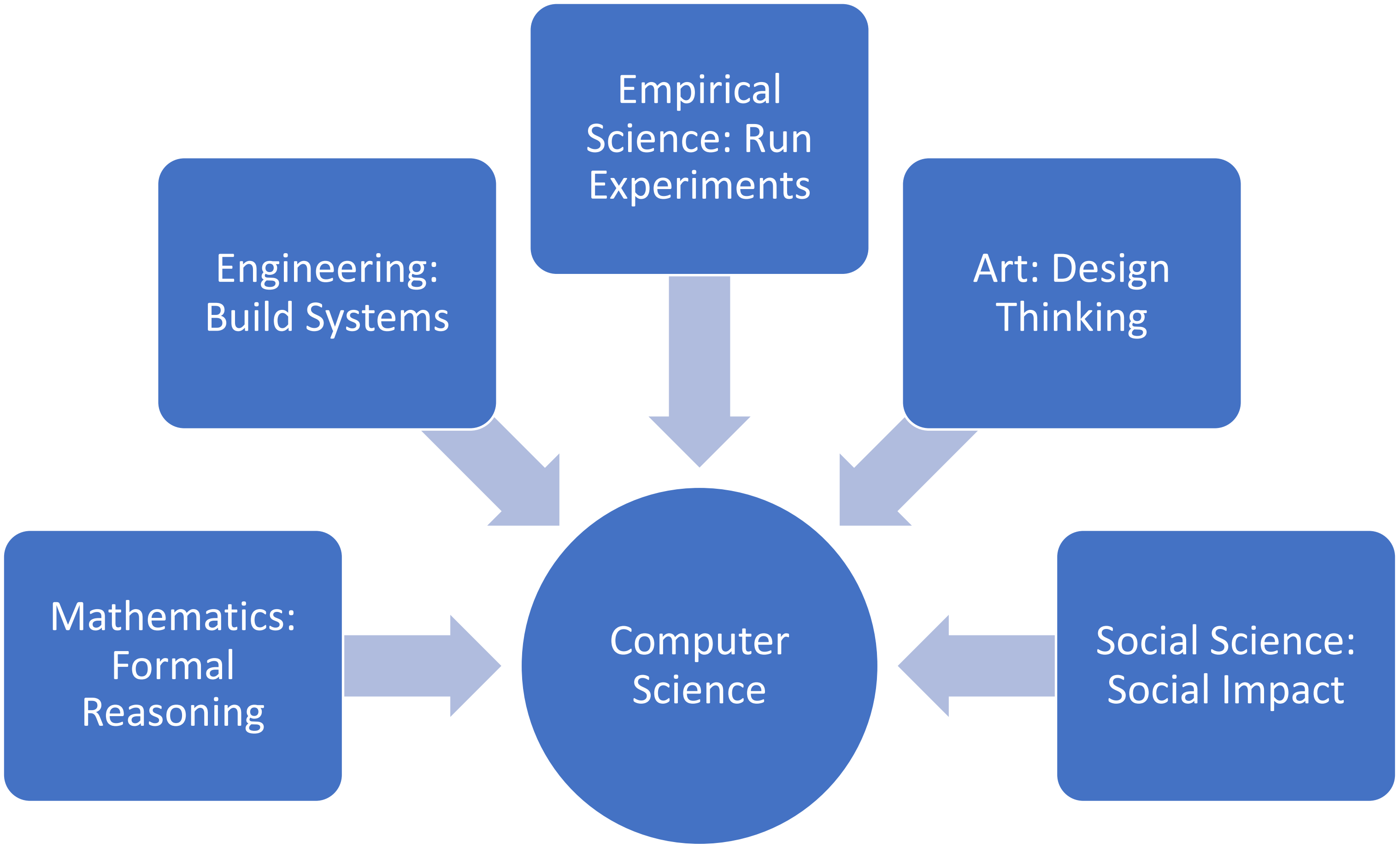
That execute algorithms…

Using data…

To do…something?.

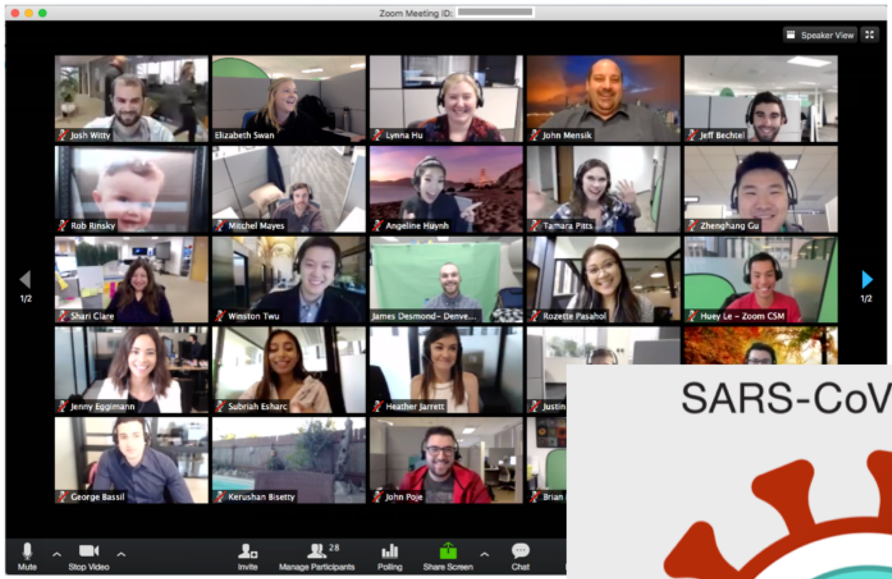
Computer Science is the systematic study of computer systems, algorithms, data, and applications/impacts.

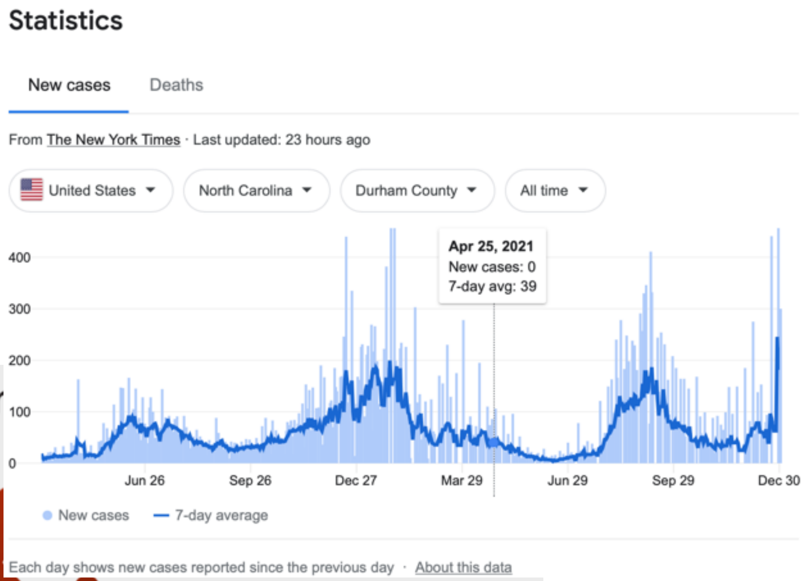
Computer science is interdisciplinary.

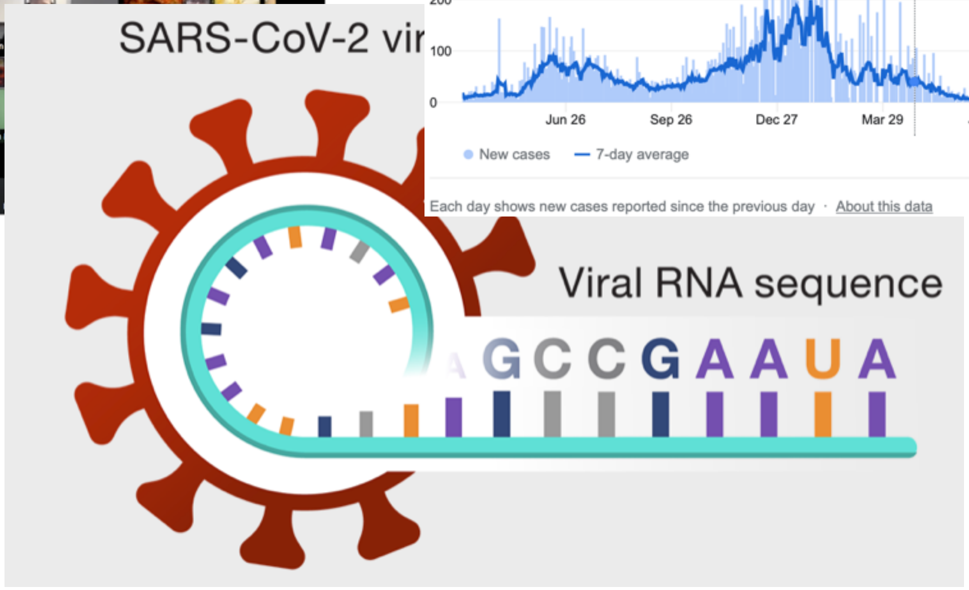

Empirical Science: Run Experiments
Engineering: Build Systems
Art: Design Thinking
Mathematics: Formal Reasoning
Computer Science
Social Science: Social Impact

Computer science in the real world?

What does computer science make possible?







Why computer science?

“Our species needs, and deserves, a citizenry with minds wide awake and a basic understanding of how the world works.”

- Carl Sagan

Lies we tell ourselves

“Computer science is only for people who want to work in software engineering at tech companies.”

“Computer science is only for people who want to program for 8+ hours a day.”

“Some people are just ‘natural’ computer scientists.”

“If I struggle with computer science, it means I’m bad at this and will always be bad at this.”

Truths that bear repeating

Computer science often goes best when paired with other interests.

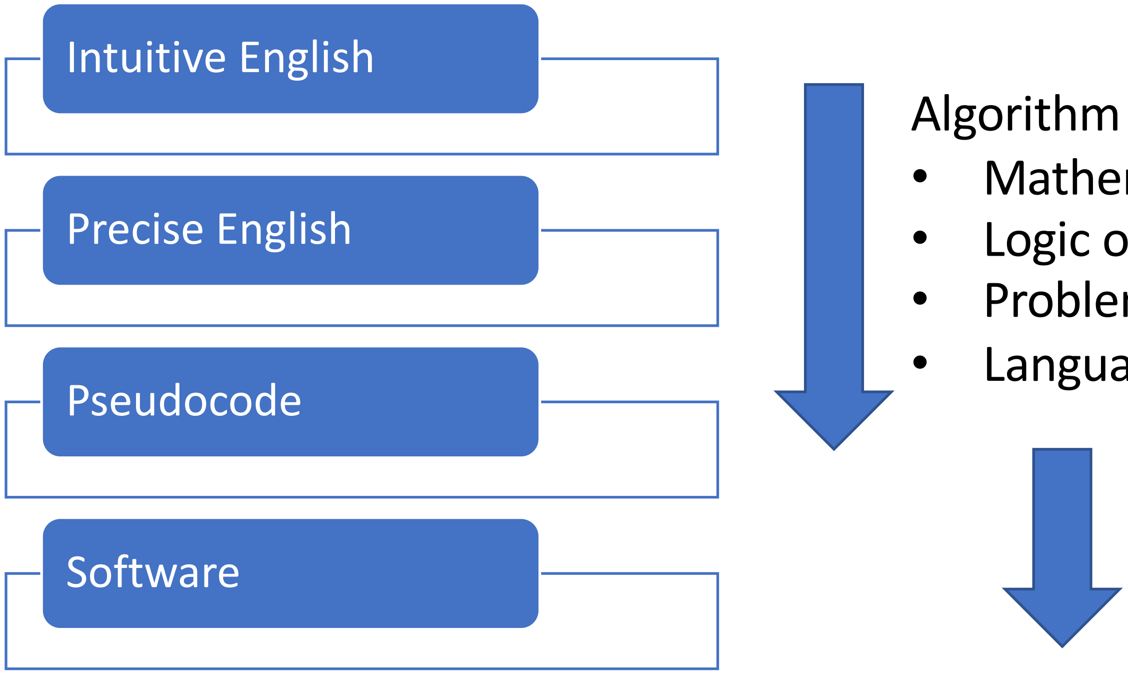
Computer science is a *fascinating* and *relevant* thing to learn, even if you don't want to code for a job.

Anyone can learn computer science! It's challenging, but not more so than mastering any other discipline.

Struggling is part of learning; remember that your future self will be more skilled than you can currently imagine.

What are algorithms?

Loosely speaking: A precise sequence of unambiguous steps that effectively compute an output given an input.



Algorithm Design

- Mathematical
- Logic of program
- Problem-solving
- Language independent

Implementation

- Semantics and Syntax
- Language dependent
- Programming on a real machine

What is code?

In order to execute an algorithm on a real computer, we must write the algorithm in a formal language. An algorithm so written is a **program**.

In this class we explore both:

Theory

- Design an algorithm
- Analyze performance
- Data structure tradeoffs

Practice

- Write a Java program
- Debug/test
- Measure performance

What are data structures?

Different data structures store different kinds of information in different ways, with algorithmic tradeoffs.

Often relates to how **efficiently** you can access or transform data during the operation of a program. For example, in Java:

- **Array**: Stores a *fixed* number of entries of a single type (e.g., integers). Fast lookup (e.g., get value of the 4th entry) and low memory use, but must know total number of entries in advance.
- **List**: Stores a *dynamic* number of entries of a single type. Uses more runtime or more memory than an array.

Why does efficiency matter?

- You wrote the next big social media app:
  - Will it work if it has 1 billion users?
  - What about on a phone with limited memory?
- In the sciences, discovery depends on computing with big data:
  - Sequencing the human genome
  - Surveying millions of images in astronomy
  - Processing data logs from the CERN collider
- Pushing the limits of current technology:
  - Virtual / augmented reality?
  - Deep neural networks for large scale machine learning?

Efficiency? Data Structures?

Live Coding

Learning goals for the course

- Given a problem statement & a real data source, design, develop, debug, and test a Java program that uses appropriate standard libraries to efficiently solve the problem.
- Write programs that effectively implement and use data structures such as: arrays, maps, linked lists, stacks, queues, trees, and graphs.
- Evaluate the time and space complexity of iterative and recursively-defined algorithms using empirical and mathematical analysis.

Some specifics you will learn

Data Structures

- Arrays
- Lists: ArrayList and LinkedList
- Sets: HashSet and TreeSet
- Maps: HashMap and TreeMap
- Stacks, Queues, Priority Queues / Heaps
- Trees: Binary Search Trees
- Graphs

Algorithms

- Iterative
- Hashing
- Big O Asymptotic Analysis
- Recursive
- Sorting
- Greedy

Software

- Java API
- Objects, Classes
- Interfaces, inheritance
- Testing, Debugging

Expected background for the course

- Introductory programming experience at the level of Computer Science 101 or equivalent.
- Following should be familiar:
    - source code, development environment, running code
    - integers, floats/doubles, characters, strings
    - printing and output
    - if/else statements and conditions
    - iteration with For/While loops
    - functions/methods, parameters, arguments, returns
    - debug your program

Do I need to know Java?

Java experience helpful but not required.

- Many of you studied Java
- Many of you never studied Java

This course about data structures and algorithms.

- We will implement them in Java,
- and you will learn about Java,
- but we could have used a different language.

Informal goals for the course

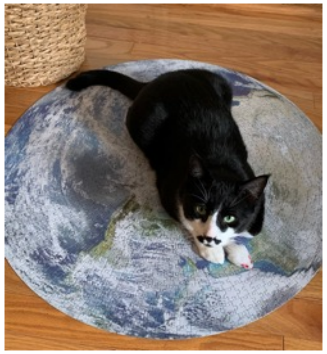
- Make or deepen a friendship with someone else passionate about computer science.
- Develop a new appreciation of computing phenomena you see in the real world.
- Experience joy when your program *works*, even if it took a while to get it there.
- WOTO: WOrking TOgether

Course Staff

- Instructor: Brandon Fain
- Teaching Associate: Kate O'Hanlon
- Graduate Teaching Assistants:
  - Zhekun Wu
  - Yilun Song
- Undergraduate Teaching Assistants: Many! Will add pictures and names to website.

Who is Prof. Fain?

My **research** is in the theory of algorithms, especially for problems in algorithmic game theory, AI/ML, and fairness.



Who are you? What do you think about computer science?

Let’s WOTO (WOrk TOgether)

Go to duke.is/g7epx
or scan the QR code

Some Administrivia

Compsci 201 Website

All material online, accessible from the website

cs.duke.edu/courses/fall22/compsci201

which redirects to ...

sites.duke.edu/compsci201f22/

Or you can see it embedded on the Sakai overview.

Setup & Resources

- ZyBook: Online interactive textbook
  - Reading schedule on course website schedule
- Java 17 Open JDK
  - Programming language for the course
- Visual Studio Code (VS Code) + Extensions
  - Development environment, edit and run code
- Git
  - Version control + use for project submission workflow

Follow directions from the website setup and resources page to get started as soon as possible.

Assignments and Grades

Large multi-file programming projects; explore object-oriented programming and algorithmic tradeoffs.

3 midterms and final exam

Small programming practice problems

Programming quiz with small problems

Participation and forms during lecture and discussion

Exams	45%
Projects	27%
APTs	10%
APT Quizzes	8%
Discussion	6%
In Class WOTO Questions	4%

Getting Technical Help

sites.duke.edu/compsci201f22/getting-help/

- Ed Discussion: (linked from Sakai)

ed COMPSCI 201 Sp22 – Discussion

- Helper/Office Hours: Sunday-Thursday evenings, drop-in via zoom. Starting next week.

- OIT Co-lab: Drop-in tech support (Java, Git, etc.)

  colab.duke.edu/resources

Getting Other Help

- Blue Devils Care: 24/7 mental telehealth support to all students at no cost. To get started, visit BlueDevilsCare.duke.edu.

- Counseling & Psychological Services

- DukeReach: Submit a report to DukeReach if you're concerned about the physical or mental well-being of yourself or another student.

Collaboration Policy

sites.duke.edu/compsci201f22/policies/

Exams, APT Quizzes: **NONE**

Projects, regular APTs: Discuss ideas, but **the code must be your own**.

- Help each other out, discuss problems and ideas
- Do not directly share code with other students
- Do not write the code line by line together
- Don’t deprive yourself or anyone else of the chance to learn by just telling someone the answer

Attendance Policy

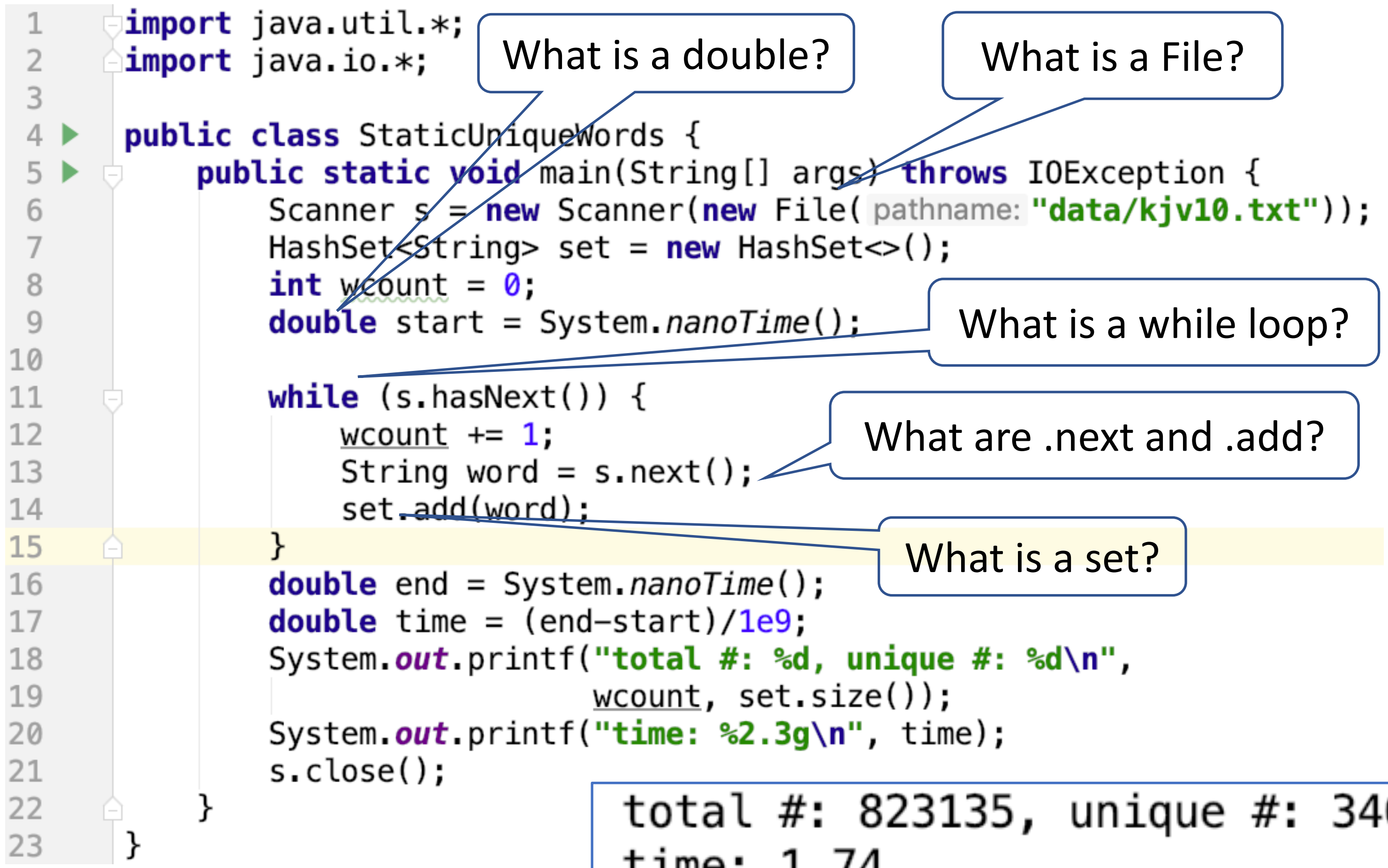
Generally expected, in-person.

- Can miss 6 lectures and 3 discussions with no penalties, no questions asked.
- Do not need to do anything for this, please do not email/STINF when you need to miss a class.
- Up to you to manage within this: Don’t skip 6 times for fun and ask us for more.

Reasoning about programs

Alternative definition of the course: CS2, a second course in computer science. Not just getting code to work, but reasoning about *how* it works.

What does this code do?



```java
import java.util.*;
import java.io.*;

public class StaticUniqueWords {
    public static void main(String[] args) throws IOException {
        Scanner s = new Scanner(new File( pathname: "data/kjv10.txt"));
        HashSet<String> set = new HashSet<>();
        int wcount = 0;
        double start = System.nanoTime();

        while (s.hasNext()) {
            wcount += 1;
            String word = s.next();
            set.add(word);
        }
        double end = System.nanoTime();
        double time = (end-start)/1e9;
        System.out.printf("total #: %d, unique #: %d\n",
                          wcount, set.size());
        System.out.printf("time: %2.3g\n", time);
        s.close();
    }
}
```

What is a double?

What is a File?

What is a while loop?

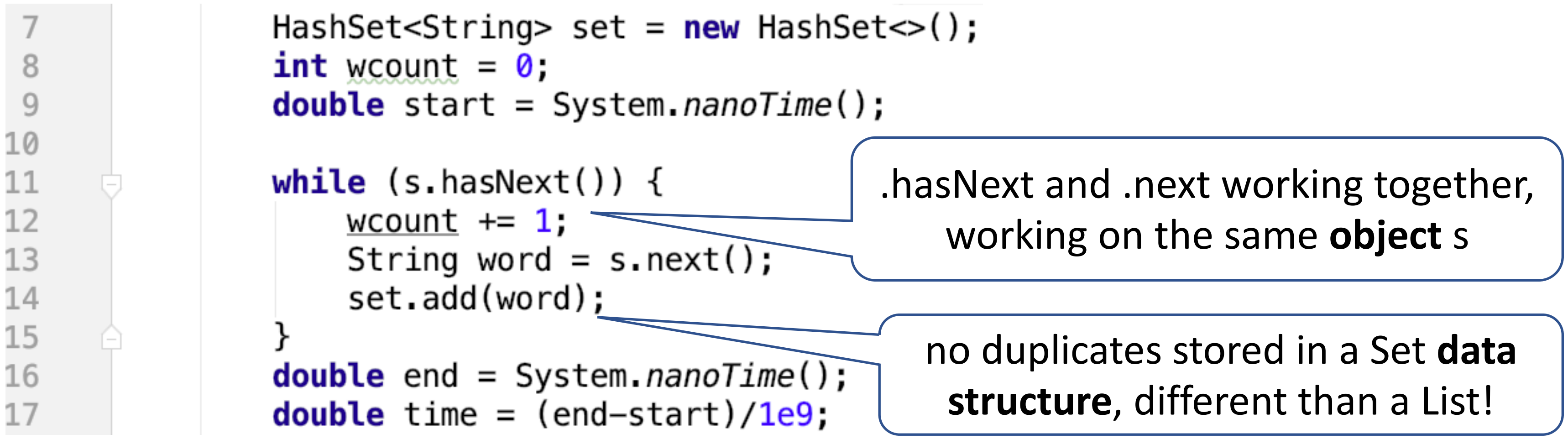
What are .next and .add?

What is a set?

```
total #: 823135, unique #: 34027
time: 1.74
```

Understanding Repetition

- When does loop terminate?
- What takes time when code runs?



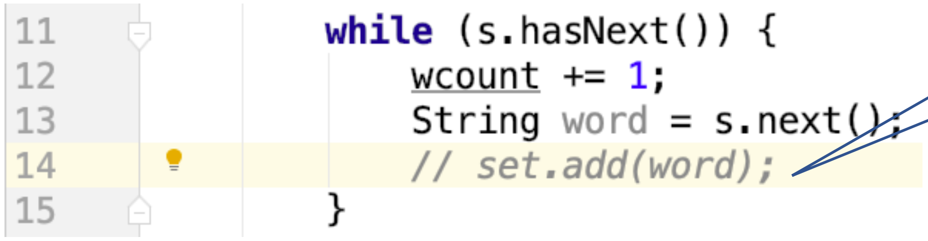
```
HashSet<String> set = new HashSet<>();
int wcount = 0;
double start = System.nanoTime();

while (s.hasNext()) {
    wcount += 1;
    String word = s.next();
    set.add(word);
}
double end = System.nanoTime();
double time = (end-start)/1e9;
```

.hasNext and .next working together, working on the same **object** s

no duplicates stored in a Set **data structure**, different than a List!

Analyze Code, Algorithm, Structures

- What file of a million strings results in fastest execution? Slowest?
  - How do we find code bottlenecks?
  - How do we experiment with code?



```
    while (s.hasNext()) {
        wcount += 1;
        String word = s.next();
        // set.add(word);
    }
```

To answer, need to understand the **implementation** of the set.

Lots of Java/Language details! But …

- This code/algorithm scales
    - Time to count is about the same as time to read

- Some Java concepts familiar from previous programming
    - You'll learn vocabulary and practice!

Latanya Sweeney

As Professor of Government and Technology in Residence at Harvard University, my mission is to create and use technology to assess and solve societal, political and governance problems, and to teach others how to do the same.

Former CTO of the FTC,

First African-American Women to earn CS PhD from MIT (2001)

Latanya Sweeney

I am a computer scientist with a long history of weaving technology and policy together to remove stakeholder barriers to technology adoption. My focus is on "computational policy" and I term myself a "computer (cross) policy" scientist. I have enjoyed success at creating technology that weaves with policy to resolve real-world technology-privacy clashes.

What to do this week

- Go to website setup and resources page
  - Get the ZyBook
  - Star reviewing Java as needed (chapters marked optional)
  - Start tech installation

- See you at…
  - Class Wednesday: Intro Java
  - Discussion Friday: Algorithmic Problem Solving