LEC 11

CSE 122

Introduction to Objects

BEFORE WE START

Talk to your neighbors:
Best places to study on campus?

Music: 122 24sp Lecture Tunes ❁

Instructors:	Miya Natsuhara and Kasey Champion			
TAs:	Ayush	Kyle	Colin	Chaafen
	Poojitha	Jacob	Ronald	Smriti
	Chloe	Atharva	Saivi	Ambika
	Ailsa	Rucha	Shivani	Elizabeth
	Jasmine	Megana	Kavya	Aishah
	Lucas	Eesha	Steven	Minh
	Logan	Zane	Ken	Katharine

Lecture Outline

- **Announcements**
- SearchEngine Recap
- OOP Review
- Example
- Abstraction

Announcements

- Programming Assignment 2 (P2) out
  - Due Thursday, May 9th by 11:59 PM
  - Which means… no assignment releasing tonight!

- Quiz 0 grades released yesterday
  - Check them out and use results to calibrate how you should study over the weekend!

- Quiz 1 on Tuesday, May 7th in your registered quiz section

- Resubmission Cycle 3 (R3) out
  - Due Tuesday, May 7th by 11:59 PM
  - Eligible assignments: *P0*, C1, P1

Lecture Outline

- Announcements

- **SearchEngine Recap** ◀

- OOP Review

- Example

- Abstraction

searchEngine & Inverted Index

- An **inverted index** is a Mapping from possible query words to the set of documents that contain that word
  - Answers the question:
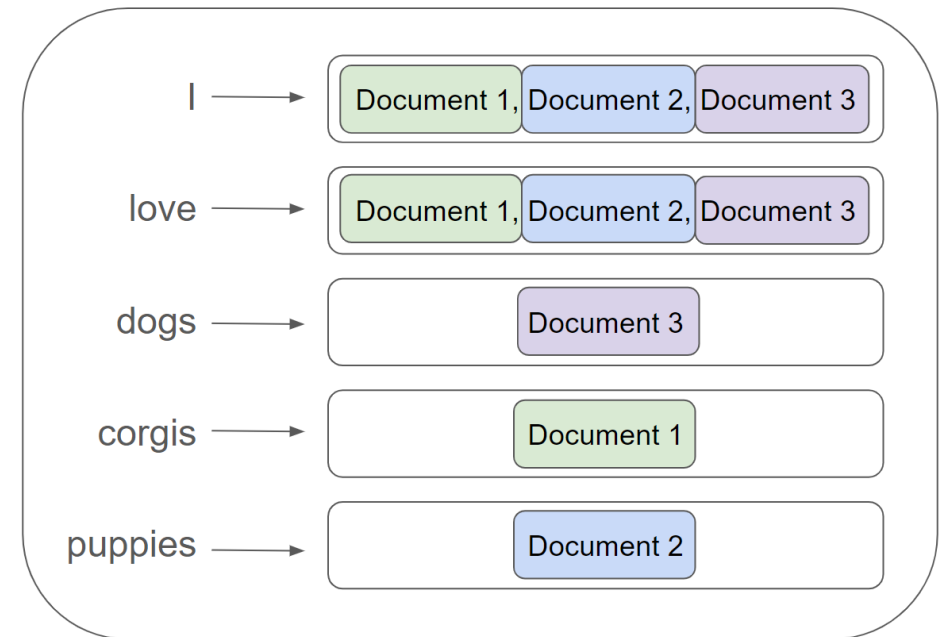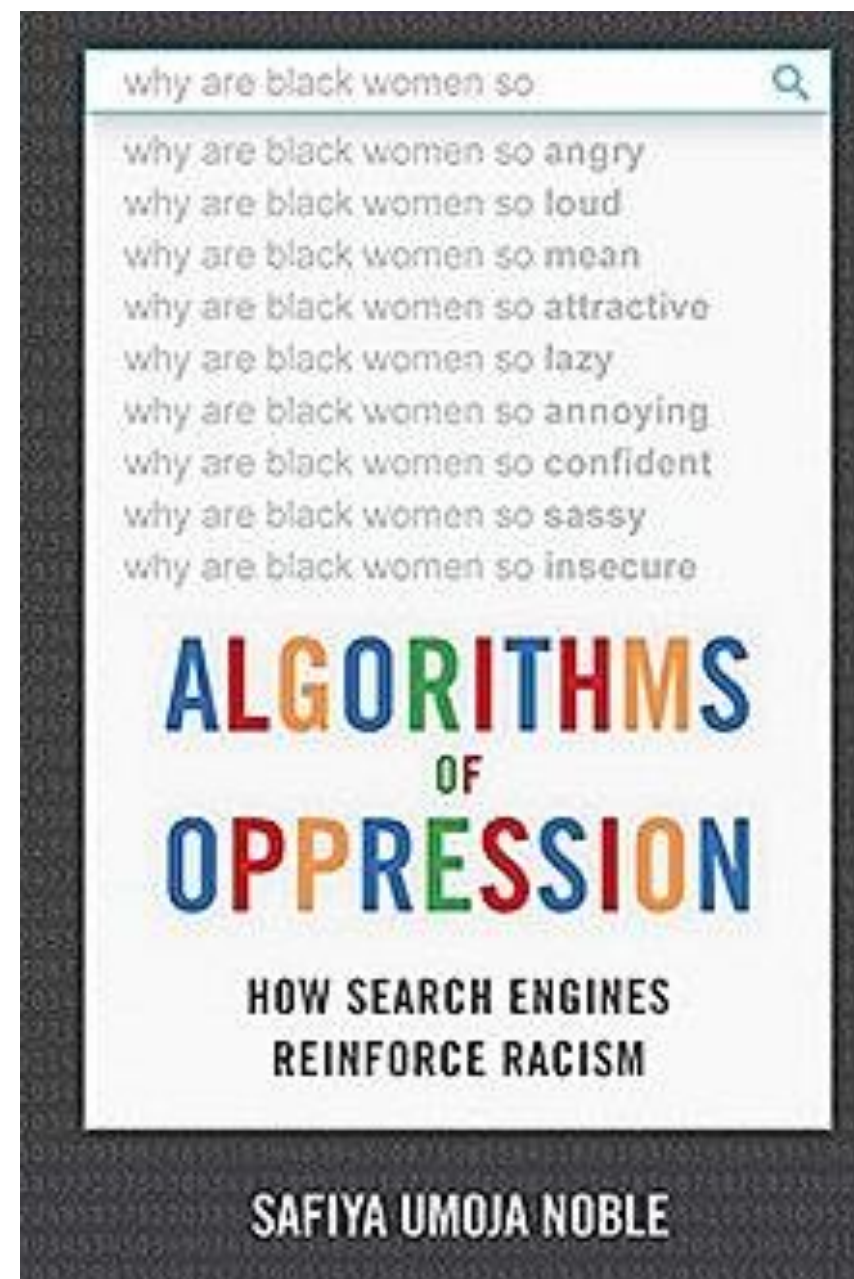   "What documents contain the word 'corgis'?"

Data Bias

- Google's autocomplete recommendations used to actually look like this
  - Fix: Don't display autocomplete results for phrases like "why are [group] _____"

Are these changes fixing the right thing?

Btw, this is a great book that you should check out if you're interested ->

What to do?

- Obviously, ideal to have datasets that aren't biased in the first place.
  - But might not always be possible if we can't fix the sources of bias in the real world...
- AI/Models aren't "neutral" or "more objective", they just quicky and automatically codify the status quo (and perpetuate biases)
  - Garbage in -> Garbage out
- Lots of work going into how to de-bias models *even if* they are trained on biased data. Active area of research!
  - Key take-away: None of this comes "for free", requires hard word to fight bias
- Ask ourselves:
  - What biases might be present in my data?
  - What assumption might I be making about who is using my program?
  - How can I write code to be more inclusive?
  - What happens when *(not if)* mistakes happen? Who potentially benefits and who is potentially harmed?

Lecture Outline

- Announcements

- SearchEngine Recap

- **OOP Review** ◀

- Example

- Abstraction

Object Oriented Programming (OOP)

- **Procedural programming**: Programs that perform their behavior as a series of steps to be carried out
  - Classes that <u>do</u> things

- **Object-oriented programming (OOP):** Programs that perform their behavior as interactions between objects
  - Classes that <u>represent</u> things
  - We're going to start writing our own objects!

Classes & Objects

- **Classes** can define the <u>template</u> for an object
  - Like the blueprint for a house!
    "What does it mean to be this thing?"

- **Objects** are the actual <u>instances</u> of the class
  - Like the actual house built from the blueprint!
    "It is an example of this thing!"

We create a new instance of a class with the new keyword

e.g., `Scanner console = new Scanner(System.in);`

State & Behavior

- **Objects** can tie related *state* and *behavior* together

- **State** is defined by the object's *fields* or *instance variables*
  - *Scanner's state may include what it's scanning, where it is in the input, etc.*

- **Behavior** is defined by the object's *instance methods*
  - *Scanner's behavior includes "getting the next token and returning it as an* `int`*", "returning whether there is a next token or not", etc.*

Syntax

```java
public class MyObject {
    // fields
    type1 fieldName1;
    type2 fieldName2;
    ...

    // instance methods
    public returnType methodName(...) {
        ...
    }
}
```

UNIVERSITY *of* WASHINGTON

Instance Variables

- Fields are also referred to as **instance variables**

- Fields are defined in a class

- Each instance of the class has their own copy of the fields
  - Hence *instance* variable! It's a variable tied to a specific instance of the class!
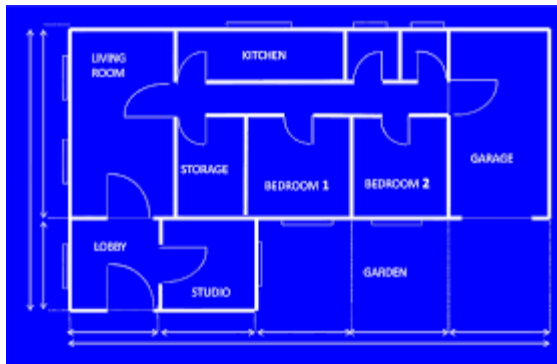
Instance Methods

- **Instance methods** are defined in a class
- Calling an instance method on a particular *instance* of the class will have effects on <u>that</u> instance

Lecture Outline

- Announcements

- SearchEngine Recap

- OOP Review

- **Example** ◀

- Abstraction

Representing a Coordinate Point

How would we do this given what we knew last week?

Maybe `int x, int y`?

Maybe `int[]`?

Representing a point

`int x, int y`

- Easy to mix up x, y
- Just two random ints floating around – easy to make mis~~take~~

`int`

Let's make a class instead!

- Not really what an array is for
- Again, just two `ints` – just have to "trust" that we'll remember to treat it like a point

Instance Methods

- **Instance methods** are defined in a class
- Calling an instance method on a particular *instance* of the class will have effects on <u>that</u> instance

Instance Methods

- **Instance methods** are defined in a class
- Calling an instance method on a particular *instance* of the class will have effects on <u>that</u> instance

Lecture Outline

- Announcements

- SearchEngine Recap

- OOP Review

- Example

- **Abstraction** ◀

Abstraction

The separation of ideas from details, meaning that we can <u>use</u> something without knowing exactly <u>how</u> it works.

You were able use the `Scanner` class without understanding how it works internally!

UNIVERSITY *of* WASHINGTON

Client v. Implementor

We have been the <u>clients</u> of many objects this quarter!

Now we will become the <u>implementors</u> of our own objects!