

LEC 10

CSE 122

Introduction to Objects

BEFORE WE START

*Talk to your neighbors:
Favorite study spot on campus?*

Instructors **Tristan Huber & Hunter Schafer**

TAs

Ambika
Andrew
Audrey
Autumn
Ayush
Ben
Colton
Di
Eesha
Elizabeth

Evelyn
Jacob
Jaylyn
Jin
Joe
Kevin
Leon
Megana
Melissa
Mia

Poojitha
Rishi
Rucha
Shivani
Shreya
Steven
Suhani
Yijia
Ziao


Questions during Class?

Raise hand or send here

sli.do #cse122



Lecture Outline

- **Announcements** 
- Nested Collection Practice
- Bias in Data Discussion
- OOP Review
- Example
- Abstraction


Announcements

- Quiz 1 yesterday
 - Retakes starting next week
 - Quiz 1 grades & Quiz 0 Retake grades out soon
- P2 Absurdle out
 - get started early!
 - due 5/11

Lecture Outline

- Announcements
- **SearchEngine Recap** 
- Bias in Data Discussion
- OOP Review
- Example
- Abstraction

Lecture Outline

- Announcements
- SearchEngine Recap
- **Bias in Data Discussion** 
- OOP Review
- Example
- Abstraction

Data Bias

- Common Misconception: Models or Artificial Intelligence (AI) are somehow “less biased” or “more objective” than humans. **Not true.**
- The programs we use operate on real-world data, and will often reflect the biases that data contains
- Have to carefully consider the context and limitations of the data we gather. If the data an algorithm is built on is vastly different than the context in which it’s used, some pretty awful outcomes can happen

Data Bias

In modern artificial intelligence, data rules. A.I. software is only as smart as the data used to train it. If there are many more white men than black women in the system, it will be worse at identifying the black women.

Color matters in Computer vision

Facial recognition algorithms made by Microsoft, IBM and Face++ were more likely to misidentify the gender of black women than white men.



Gender was misidentified in **up to 1 percent of lighter-skinned males** in a set of 385 photos.



Gender was misidentified in **up to 7 percent of lighter-skinned females** in a set of 666 photos.

One widely used facial-recognition data set was estimated to be more than 75 percent male and more than 80 percent white, according to another research study.



Gender was misidentified in **up to 12 percent of darker-skinned males** in a set of 318 photos.



Gender was misidentified in **35 percent of darker-skinned females** in a set of 271 photos.

citation:

<http://gendershades.org/>

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 the bias in the real world...
- AI/Models aren't "neutral" or "more objective", they just quickly 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 work to fight bias
- Ask ourselves:
 - What biases might be present in my data?
 - What assumptions 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
- Bias in Data Discussion
- **OOP Review** ◀
- Example
- Abstraction

(PCM) Object Oriented Programming (OOP)

- **object-oriented programming (OOP):** Program as interaction between *things* (ie. objects)
 - Code that *represent* things
 - We're going to start writing our own objects!

compare to:

- **procedural programming:** Program as a list of steps
 - Code that *does* things

(PCM) State & Behavior

- Object - a grouping of related *state* and *behavior*
- *State* - information the object holds
 - *ArrayList state: elements, size*
- *Behavior* - things the object can do
 - *ArrayList behavior: add an element, remove an element, get size, etc*



(PCM) Syntax

```
public class MyObject {  
    // fields (aka instance variables)  
    type1 fieldName1;  
    type2 fieldName2;  
    ...  
  
    // instance methods  
    public returnType methodName(...) {  
        ...  
    }  
}
```

} State

} Behavior


(PCM) Classes & Objects

- Classes can define the *template* for an object
 -  Like the blueprint for a house!
- Objects are the actual *instances* of the class
 -  Like the actual house built from the blueprint!
 - Each *instance* has its own state

We create a new object of a class with the **new** keyword

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

Lecture Outline

- Announcements
- SearchEngine Recap
- Bias in Data Discussion
- OOP Review
- **Example** 
- Abstraction

Representing a 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 mistakes!

`int[]`

- 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

Let’s make a class instead!

Lecture Outline

- Announcements
- SearchEngine Recap
- Bias in Data Discussion
- OOP Review
- Example
- **Abstraction** 

(PCM) Abstraction

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

You could use the Scanner class without understanding how it worked internally!

(PCM) Client v. Implementor

We have been the *clients* of many objects this quarter!

Now we will become the *implementors* of our own objects!