LEC 10

CSE 122

Introduction to Objects

Questions during Class?

Raise hand or send here

sli.do #cse122

BEFORE WE START

Talk to your neighbors:
Favorite study spot on campus?

Music: Hunter/Miya's Playlist

Instructor Hunter Schafer / **Miya Natsuhara**

TAs

Ajay	Gaurav	Melissa
Andrew	Hilal	Noa
Anson	Hitesh	Parker
Anthony	Jake	Poojitha
Audrey	Jin	Samuel
Chloe	Joe	Sara
Colton	Joe	Simon
Connor	Karen	Sravani
Elizabeth	Kyler	Tan
Evelyn	Leon	Vivek

Lecture Outline

- Announcements
- Quiz Retake Policies
- OOP Review
- Example
- Abstraction

Announcements

- Programming Assignment 2 is due tomorrow (Thurs, Nov 3)
- IPL Staffing
  - Reduced staffing next week (11/7 – 11/10)
  - IPL Closed 11/11 for University Holiday
- No assignment released on Friday!
- Quiz 2 next week
  - Tuesday 11/8 *or* Thursday 11/10
  - (Go vote!)

Lecture Outline

- Announcements
- **Quiz Retake Policies**
- OOP Review
- Example
- Abstraction

Quiz Retake Policies

- Quiz grades used for course grade calculation will be the **best** grade on a quiz problem (original || retake) instead of a retake completely replacing your original quiz grade.
  - More details in the announcement from Friday

- Only sign up for a quiz retake if you are actually going to show up
  - We had several time slots overbooked on Tuesday
  - And several students who didn't show up
  - Quiz retakes will increase in demand as the quarter goes on
  - **Updated policy**: If a student no-shows or attempts to cancel their retake slot at the last minute without an instructor-approved absence, their quiz grade will be **UUU** (no "best grade" application).

Lecture Outline

- Announcements
- Quiz Retake Policies
- **OOP Review** 
- Example
- Abstraction

(PCM) Object Oriented Programming (OOP)

- **procedural programming**: Programs that perform their behavior as a series of steps to be carried out
  - Classes that *do* things

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

(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!

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

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

(PCM) 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.*

(PCM) Syntax

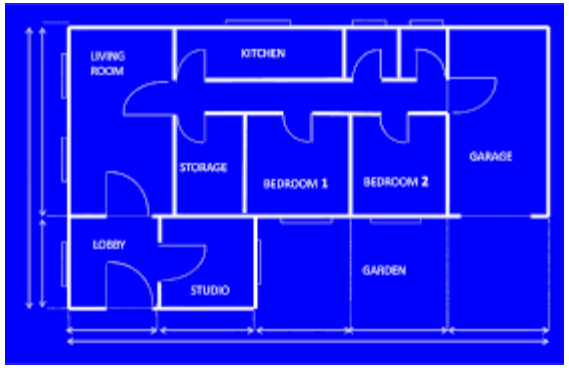
```
public class MyObject {
    // fields
    type1 fieldName1;
    type2 fieldName2;
    ...

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

(PCM) 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!



Lecture Outline

- Announcements
- Quiz Retake Policies
- 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 `int`s floating around – easy to make mistakes!

`int[]`

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

Let's make a class instead!

Lecture Outline

- Announcements
- Quiz Retake Policies
- 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!

Practice : Think

sli.do #cse-122

What do p and p2 hold after the following code is executed?

```
Point p = new Point();
p.x = 3;
p.y = 10;
Point p2 = p;
p2.y = 100;
p = new Point();
p.y = -99;
```

A. p: (3, 10) p2: (3, 10)
B. p: (3, -99) p2: (3, 100)
C. p: (0, -99) p2: (3, 100)
D. p: (3, -99) p2: (0, 100)
E. p: (0, -99) p2: (3, 10)

Practice : Pair

sli.do #cse-122

What do p and p2 hold after the following code is executed?

```
Point p = new Point();
p.x = 3;
p.y = 10;
Point p2 = p;
p2.y = 100;
p = new Point();
p.y = -99;
```

A. p: (3, 10) p2: (3, 10)
B. p: (3, -99) p2: (3, 100)
C. p: (0, -99) p2: (3, 100)
D. p: (3, -99) p2: (0, 100)
E. p: (0, -99) p2: (3, 10)