LEC 10

CSE 122

Introduction to Objects

Questions during Class?

Raise hand or send here

sli.do #cse122

BEFORE WE START

Slido vote & chat with neighbors:

What are your favorite places to study on/near campus?

Music: 122 26sp Lecture Jams

Instructor: Elba Garza

TAs:

David	Caleb	Cole	Yang
William	Neha	Blake R.	Cady
Dani	Wesley	Carson	Diya
Rohan	Isis	Sushma	
Andrew	Colin	Connor	
Ava	Naomi	Mahima	
Shreyank	Hanna	Nicolae	
Nicole	Blake P.	Ivory	

Lecture Outline (1/3)

- Announcements
- OOP
- Example

Announcements

- Programming Assignment 2 (P2)
  - Due Tuesday, May 12th by 11:59pm PT
  - Note: This is an extended deadline
- Quiz 1 tomorrow, May 7th in your registered quiz section!

Lecture Outline (2/3)

- Announcements
- **OOP**
- Example

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!

Classes & Objects

- **Classes** can define the template for an object
  -  Like the blueprint for a house!

    "What does it mean to be this thing?"

- **Objects** are the actual instances 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);`

Object 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.*
  - ***...Next time!***

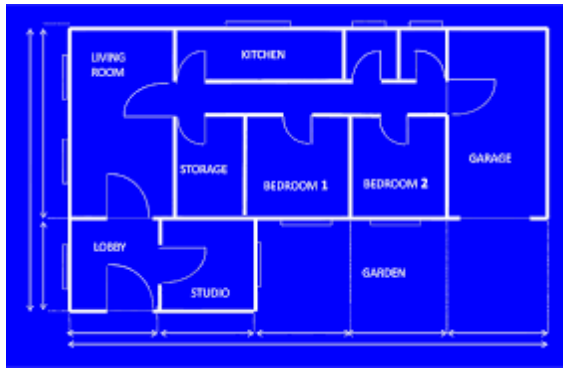
Object Class Syntax

```
public class MyObject {
    // fields (or instance variables)
    type1 fieldName1;
    type2 fieldName2;
    ...

    // constructor
    public MyObject() {
        ...
    }

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

Instance Variables

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

- Fields are defined in a class, where 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 only on that instance

Lecture Outline (3/3)

- Announcements
- OOP
- **Example** 

Representing a Shopping Cart (1/2)

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

Maybe `int capacity,` ~~`List<String>`~~ `items?`

Maybe `Set<String>?`

Representing a Shopping Cart (2/2)

How eek?

May ?

Let's make a class instead!

Maybe `Set<String>`?

Constructors

- **Constructors** create (or construct) a new instance of a class

- Java provides a default constructor “for free” that sets all fields to a default value

We’ll talk more about constructors next time too!