OO

- object: collection of data with associated behaviors
- class: general template for object, logical entity
- instance: specific object of a certain class with its own set of data and behaviors
- attributes: variables stored inside object with unique values for each instance (aka, properties, members, instance variables)
- methods: behaviors taht objects of a certain class can perform

OO SOFTWARE DEVELOPMENT

Three stages:

1. object-oriented analysis
    - looking at problem or system
    - identifying object sand interations
    - about "what needs to be done"
    - trying to identify requirements

OO SOFTWARE DEVELOPMENT

2. object-oriented design
    - turning requirements into implementation specification
    - going from waht should be done to "how it should be done"
    - identifying classes and interfaces, methods, etc.
3. object-oriented programming
    - converting design into a working program

UML

- Umified Modeling Language
- Not associated with a specific programming language
- Depicts structure of OO system
- Show classes and interfaces and relationshsips between them

DEPICTING A CLASS

- Classes (and interfaces) are represented as rectangles
- Rectangle has 3 sections
  - Name
  - Instance variables
  - Methods
- Each method/variable has visibility indicator
  - + public (we'll only use this one)
  - technically also – (private) and # (protected)

DEPICTING A CLASS

- One instance variable per line
- Each instance variable lists type
- Example

```
+ name : str
```

DEPICTING A CLASS

- One method per line
- Each method lists parameters (and type for each), followed by return type
- Example

```
+ __init__(self, arg1, arg2)
+ getName() -> str
```

MORE GENERALLY

Instance variables

```
vis name : type [= default_value]
```

Methods

```
vis name(param_name1, param_name2) -> return_type
```

DEPICTING RELATIONSHIPS: ASSOCIATION

- When one object "has-a" different object
- `A` has-a `B` if `B` is type of field(s) in `A`
- Example: Book class has instance variable that is Publisher
- Use a solid, directed line from A to B

DEPICTING RELATIONSHIPS: ASSOCIATION

- Two forms of association
  - Aggregation (solid line, open diamond) ("has-a" relationship)
  - Composition (solid line, closed diamond) ("own" relationship)
- Diamond goes at side of "whole" / "owner"
- Composition is stronger than aggreggation
  - Doesn't make sense for the contained object to exist outside
  - Ex: Person has a head (closed diamond at Person)

DEPICTING RELATIONSHIPS: DEPENDENCY

- indicates a "uses" relationship
- Examples: `A` uses `B` if
  - `A` has method(s) with local variable of type `B`
  - `A` has method(s) with parameter of type `B`
  - `A` has method(s) with return type `B`
  - `A` has method(s) that invoke methods in `B`
- Use a dashed, directed line from A to B