UML

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
    - – private
    - # protected

DEPICTING A CLASS

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

```
+ name : String
```

DEPICTING A CLASS

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

```
+ main(args : String[]) : void
+ getName() : String
```

MORE GENERALLY

Instance variables

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

Methods

```
vis name(param_name1 : type, param_name2 : type, ...) : return_ty
```

DEPICTING A CLASS (CONT.)

- Abstract classes: italicize class name
- Interface: `<<interface>>` placed above name
- Static methods/variables: underline

DEPICTING RELATIONSHIPS: GENERALIZATION

- Relationship between general thing and more specific kind of it
- "is-a" relationship indicated through inheritance
- Use solid line with open arrowhead pointing from child to parent

DEPICTING RELATIONSHIPS: REALIZATION

- When one thing specifies contract another must carry out
- aka, interface implemented by a class
- Use dashed line with open arrowhead pointing from class to interface

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