CS 213 – Spring 2016

Lecture 11 – Feb 23

UML Class Diagram - I

UML Diagrams

- UML stands for Unified Modeling Language, which is a (mainly) graphical notation used to express object-oriented design
- There are three kinds of UML diagrams that are used in practice:
  - **Class diagram**, used to show classes and the relationships between them
  - **Sequence diagram**, used to show sequences of activity when methods are invoked on classes
  - **State diagram**, used to represent state-based designs

Class Diagram

- A class diagram shows classes and the relationships between them

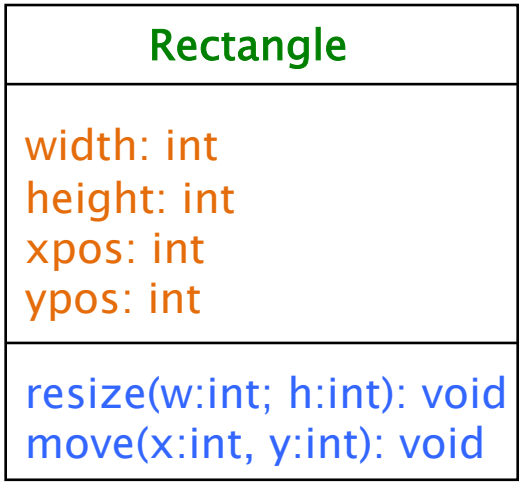
- The simplest way to draw a class is like this:

- Details may be added to the class like this:

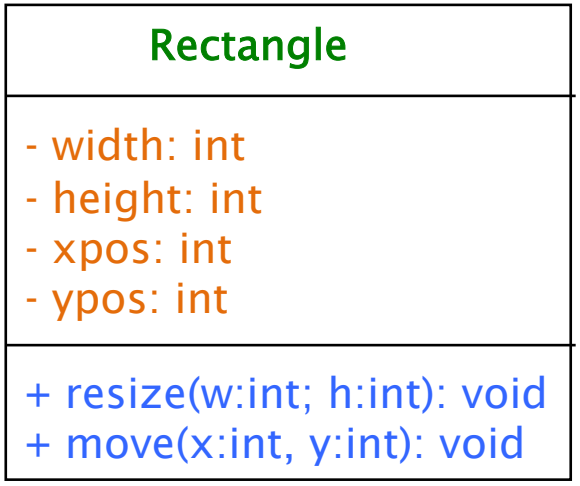
Attributes appear below the class name, and operations (methods) appear below the attributes

Class Diagram with Attributes and Methods

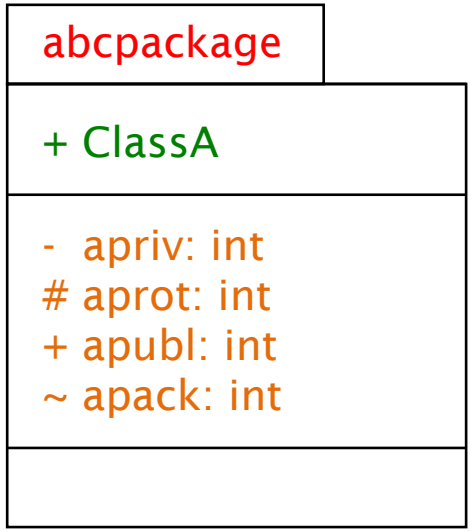
- An even greater level of detail can be added by specifying the type of each attribute, as well as type of each parameter and return type for each method:



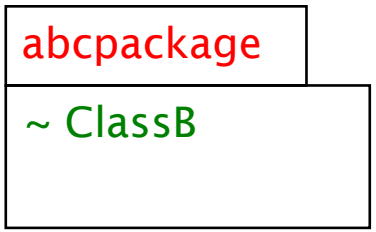
- And the access level (private, public, etc.) for each member:



UML Notation for Access Levels

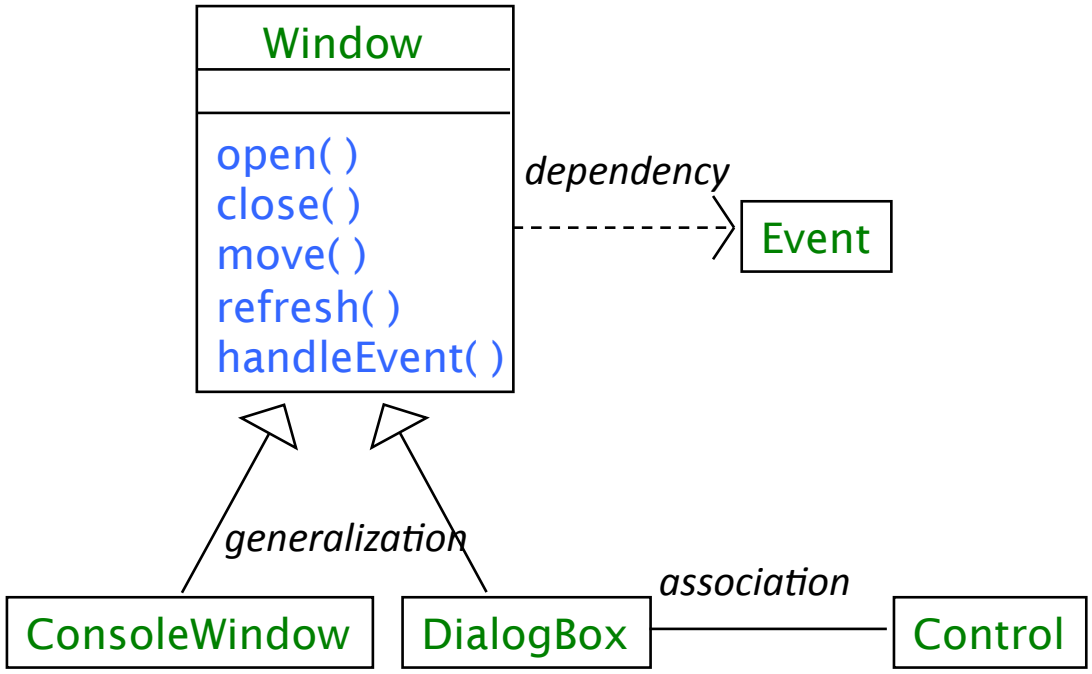


+ : public
# : protected
~ : package
– : private



Class Diagram: Relationships

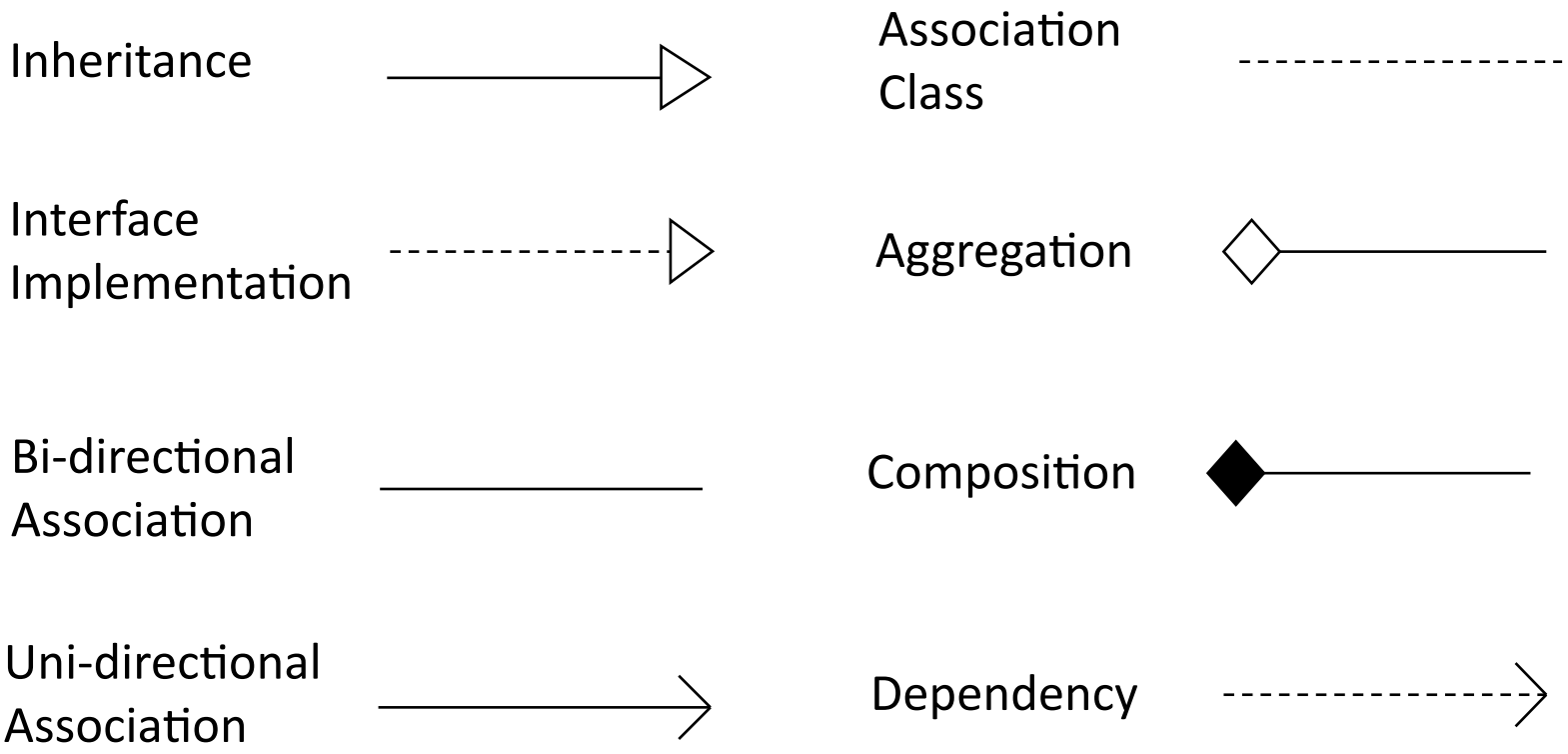
- Relationships between classes are essentially of three kinds: *generalization/specialization* (super/sub), *association*, and *dependency*



This example from "The Unified Modeling Language User's Guide" by Booch, Rumbaugh, Jacobson

Class Diagram: Relationships

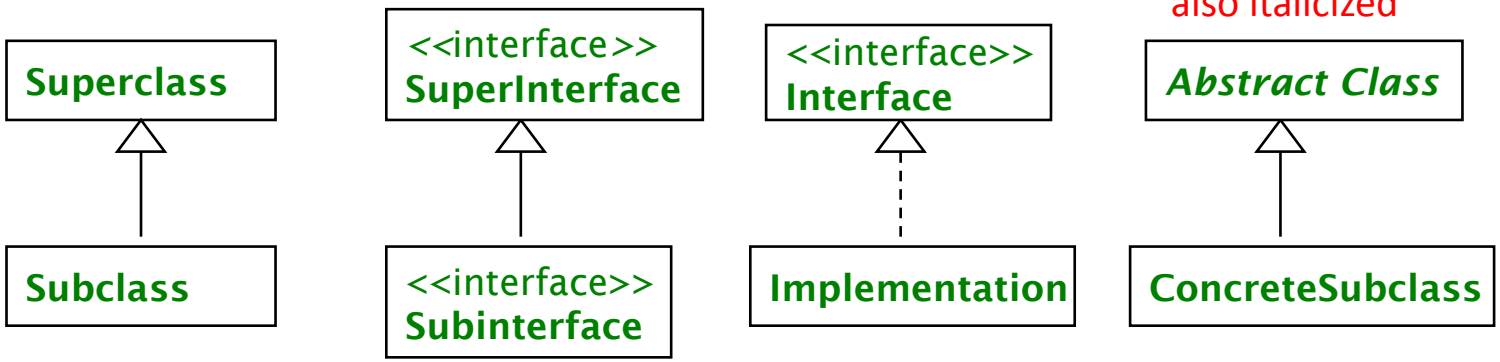
- Relationships between classes are represented by various kinds of lines



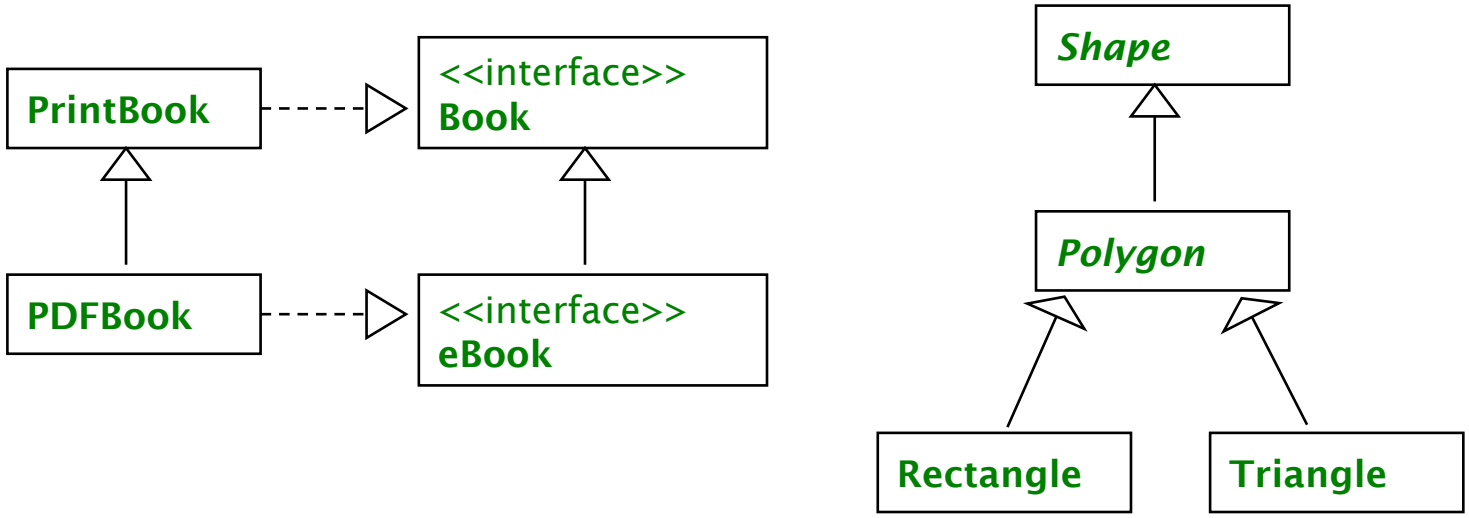
Generalization (and Interface Implementation)

- Notation

abstract methods are also italicized

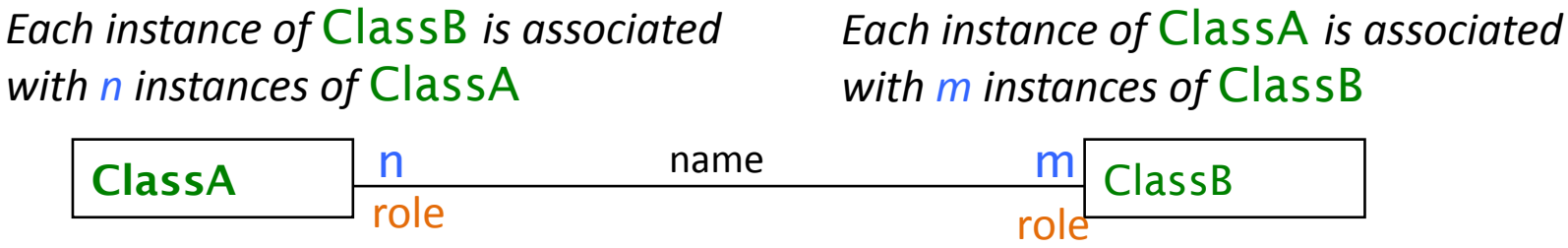


- Examples

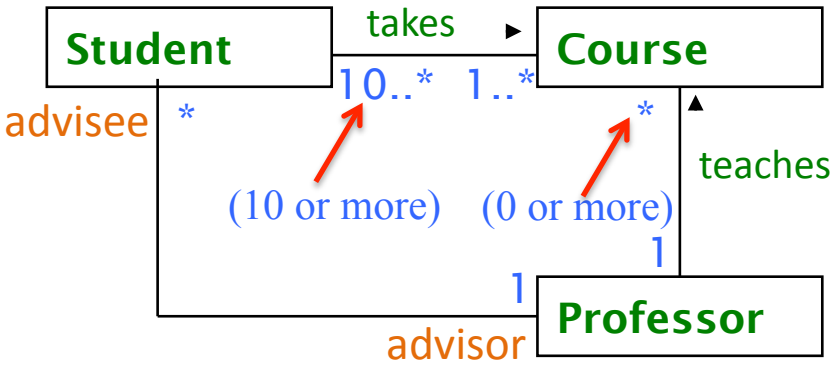


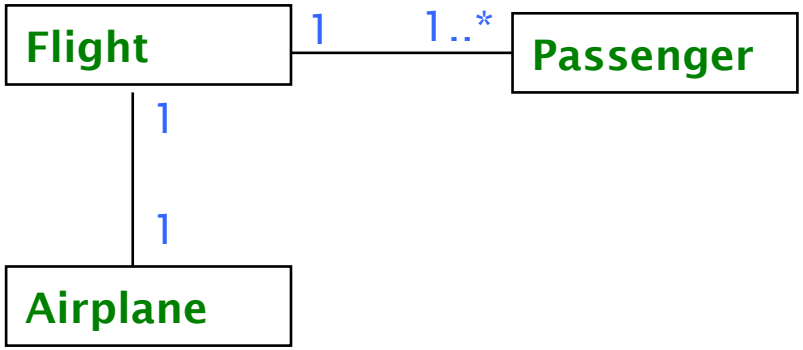
Association and Multiplicity

- An association is a general relationship between two classes, with options for name of association, and number of instances (multiplicity) of participation of each class



Association and Multiplicity

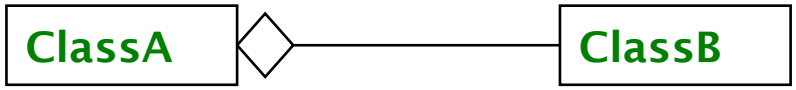




- Multiplicity can also be specified as one of the values an enumerated set such as 1, 3..5

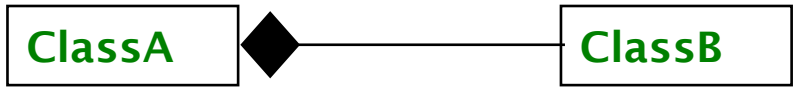
Aggregation and Composition

- Aggregation is a special kind *of association that represents a has-a or* whole-parts relationship – the *whole* is the aggregate class instance, and the *parts* are the component class instances



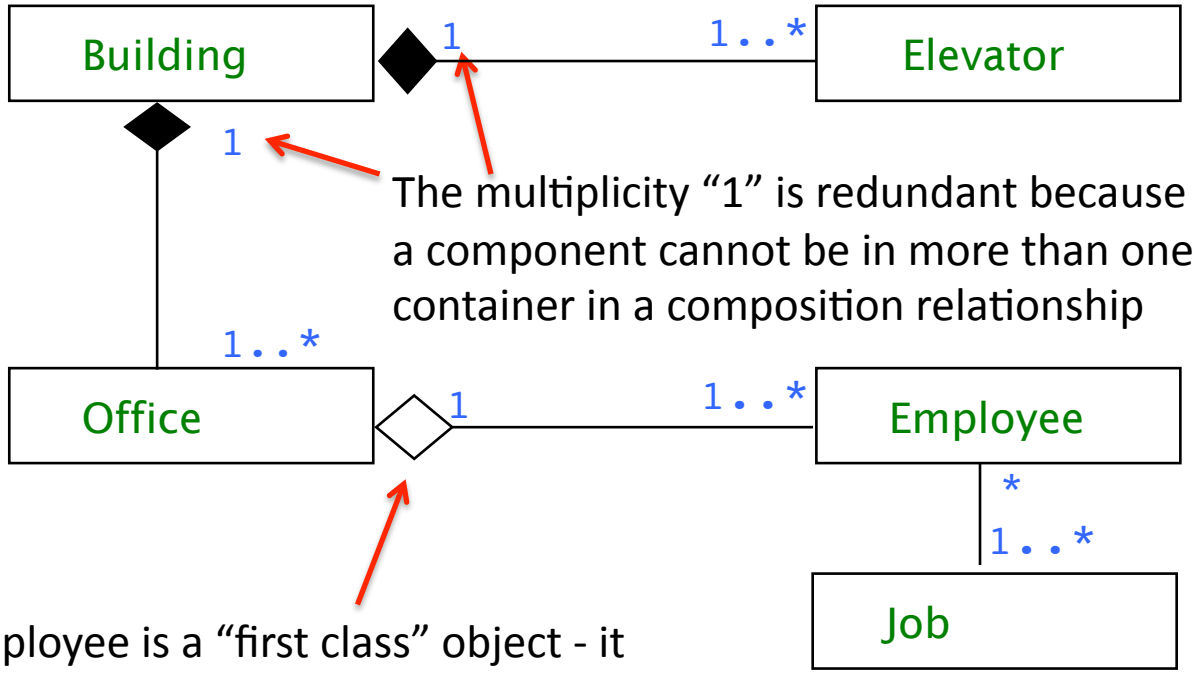
Each instance of ClassA *aggregates one or more instance of* ClassB

- Composition is a stronger form of aggregation, in which the *components* live or die with the containing class (the whole)—a deletion of the whole will lead to the deletion of the parts (an object may be a part of only one composite at a time)



Each instance of ClassA *is composed of one or more instance of* ClassB

Aggregation and Composition



Employee is a "first class" object - it has an independent existence. Even if the Office object goes away, Employee objects will still remain. Hence aggregation instead of composition.