Class Diagram

CSE 3223

Mir Tafseer Nayeem

Faculty Member, CSE AUST

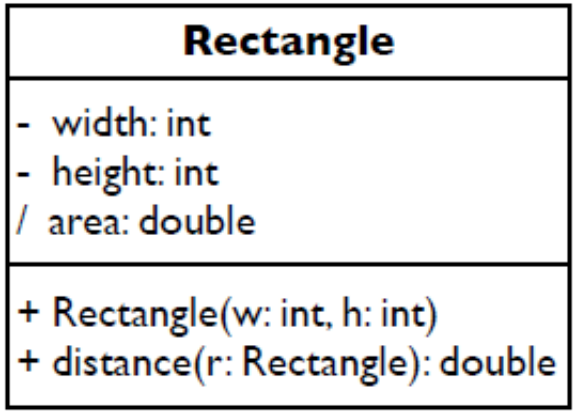
tafseer.nayeem@gmail.com

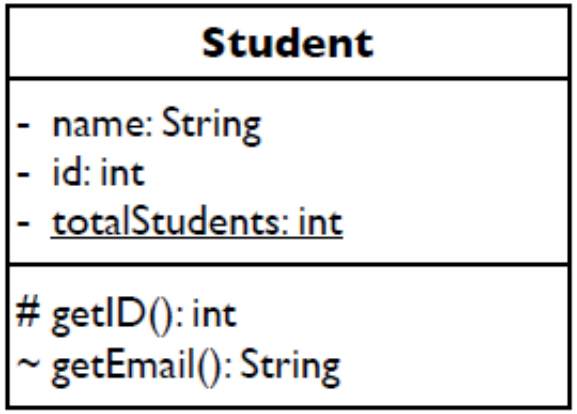
How to design classes?

- Class diagram shows classes and relationships among them.
- Identify classes and interactions from project requirements:
  - **Nouns** are potential classes, objects, and fields
  - **Verbs** are potential methods or responsibilities of a class
  - **Relationships** between nouns are potential interactions (generalization, dependence, etc.)

Diagram of a Single Class

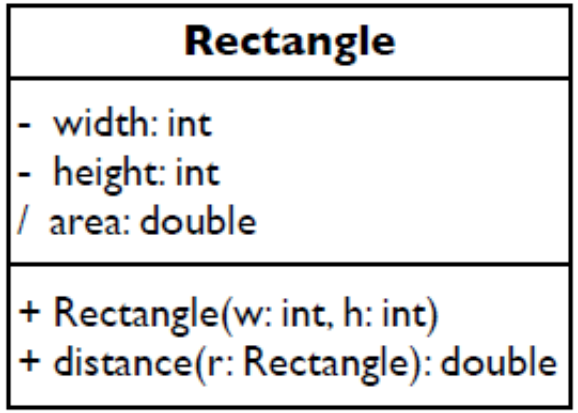
- Class name
  - write «interface» on top of interfaces' names
  - use *italics* for an abstract class name
- Attributes
  - fields of the class
- Operations / methods
  - may omit trivial (get/set) methods
  - but don't omit any methods from an interface!
  - should not include inherited methods

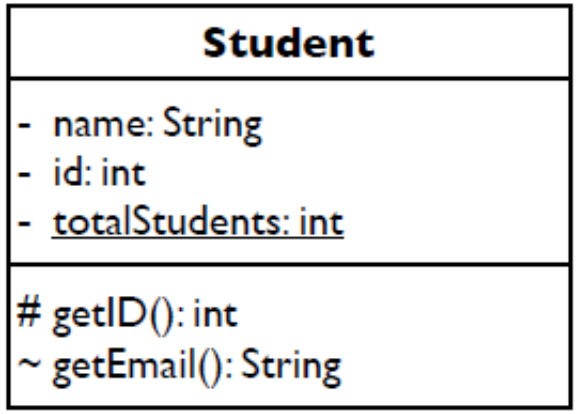




Class attributes (fields, instance variables)

- visibility name : type *[count] = default_value*
- **visibility**
  - + public
  - # protected
  - - private
  - / derived
- **derived attribute**: not stored, but can be computed from other attribute values
- underline <u>static attributes</u>



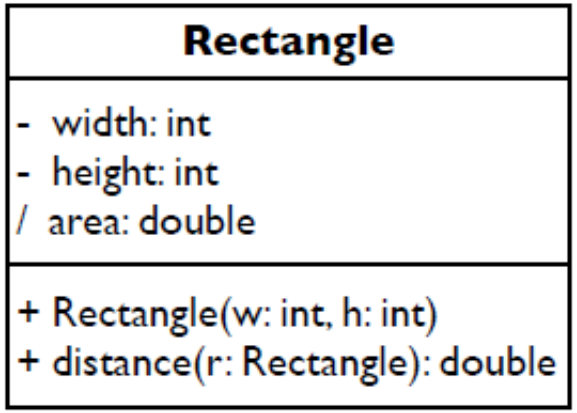


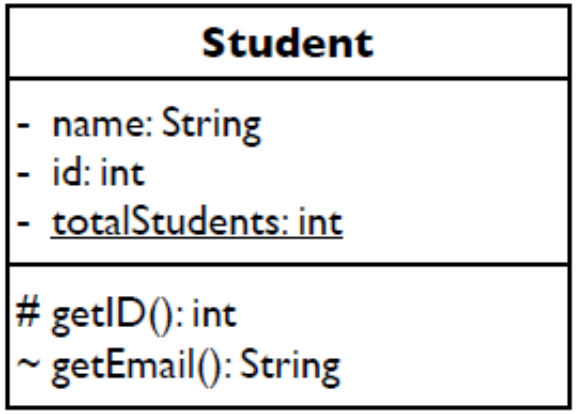
Class operations / methods

- visibility name(*parameters*) : return_type
- **visibility**
  - + public
  - # protected
  - - private
  - ~ package (default)
- **parameters** listed as name : type
- omit return_type on **constructors** and when

return type is **void**

- method example:
  + distance(p1: Point, p2: Point): double



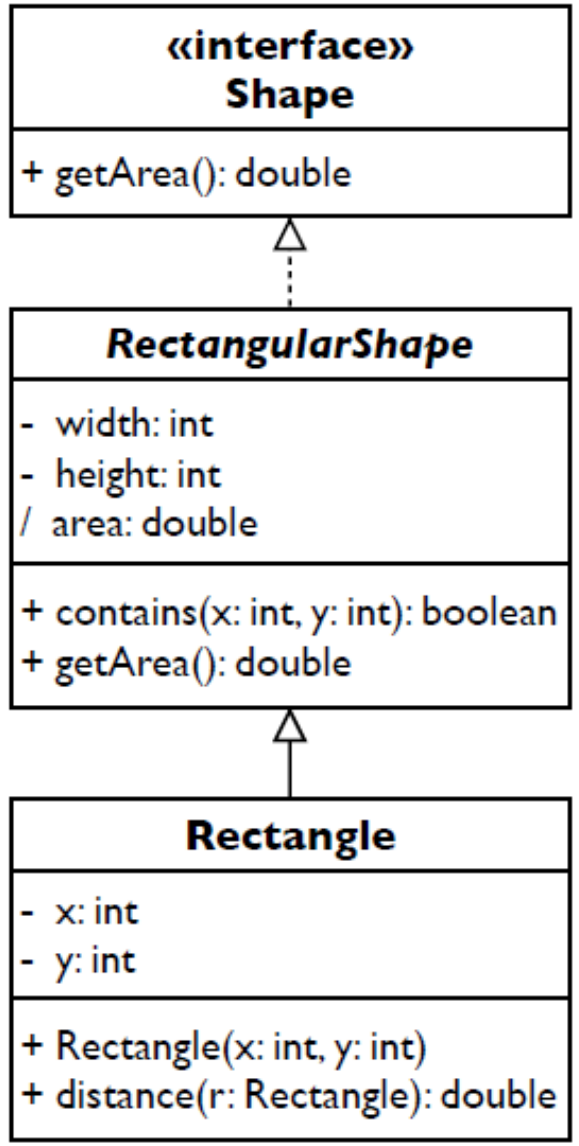


Relationships between classes

- **Generalization**: an inheritance relationship (***is a***)
  - inheritance between classes
  - interface implementation
- A**ssociation**: a usage relationship
  - Dependency (***uses a***)
  - Aggregation (***has a***)
  - Composition (***has a***)

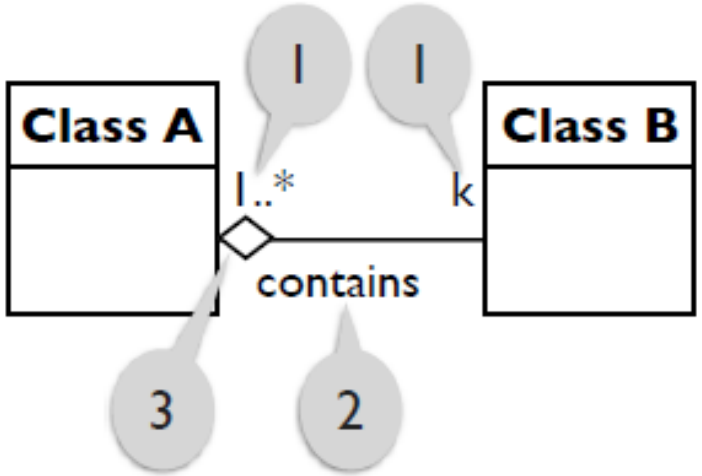
Generalization relationships

- Hierarchies drawn top-down
- Arrows point upward to parent
- Line/arrow styles indicate if parent is a(n):
  - **class**: solid line, black arrow
  - **abstract class**: solid line, white arrow
  - **interface**: dashed line, white arrow



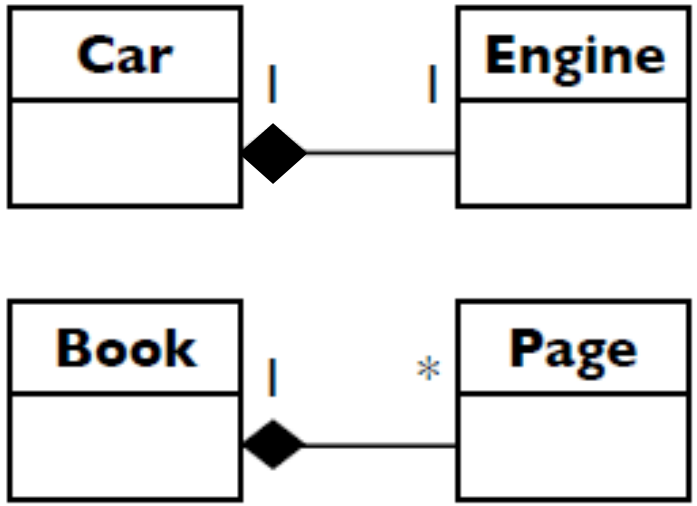
Associational (usage) relationships

1. Multiplicity (how many are used)
    - * (zero or more)
    - 1 (exactly one)
    - 2..4 (between 2 and 4, inclusive)
    - 3..* (3 or more)
- 2. Name (what relationship the objects have)
- 3. Navigability (direction)

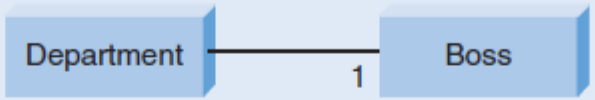
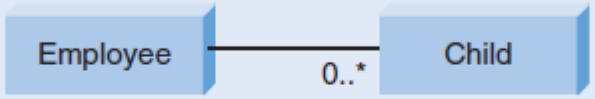
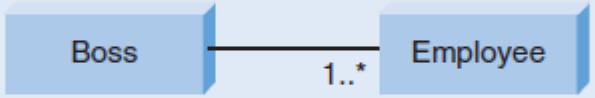
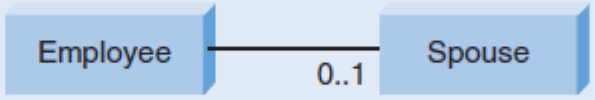
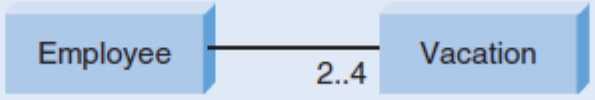
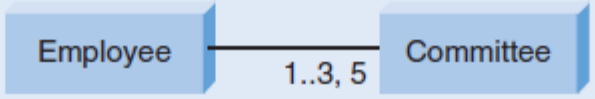


Association multiplicities

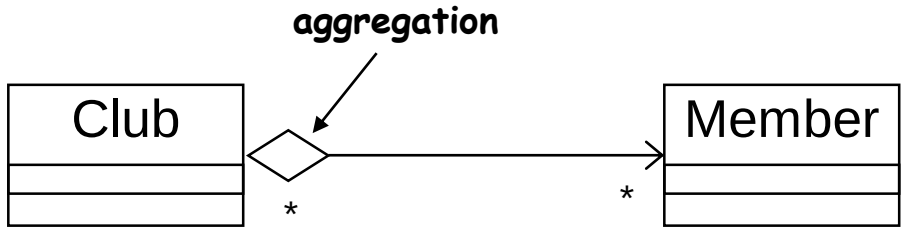
- **One-to-one**
  - Each car has exactly one engine.
  - Each engine belongs to exactly one car.
- **One-to-many**
  - Each book has many pages.
  - Each page belongs to exactly one book.



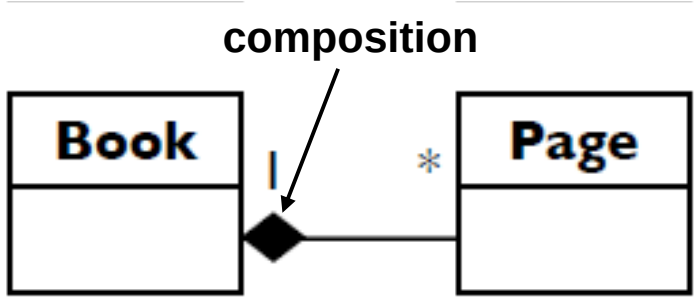
Multiplicities Examples

Instance(s)	Representation of Instance(s)	Diagram Involving Instance(s)	Explanation of Diagram
Exactly one	1		A department has one and only one boss.
Zero or more	0..*		An employee has zero to many children.
One or more	1..*		A boss is responsible for one or more employees.
Zero or one	0..1		An employee can be married to zero or one spouse.
Specified range	2..4		An employee can take between two to four vacations each year.
Multiple, disjoint ranges	1..3, 5		An employee is a member of one to three or five committees.

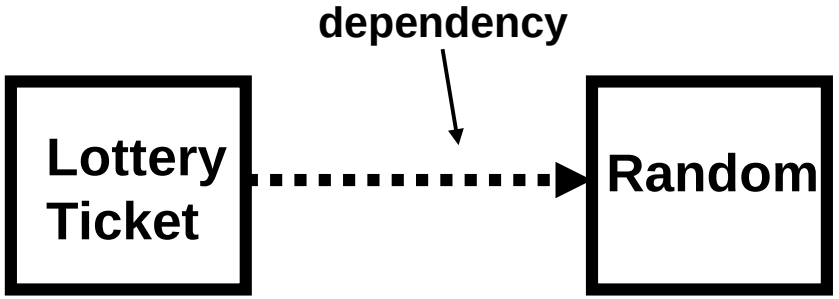
Association types



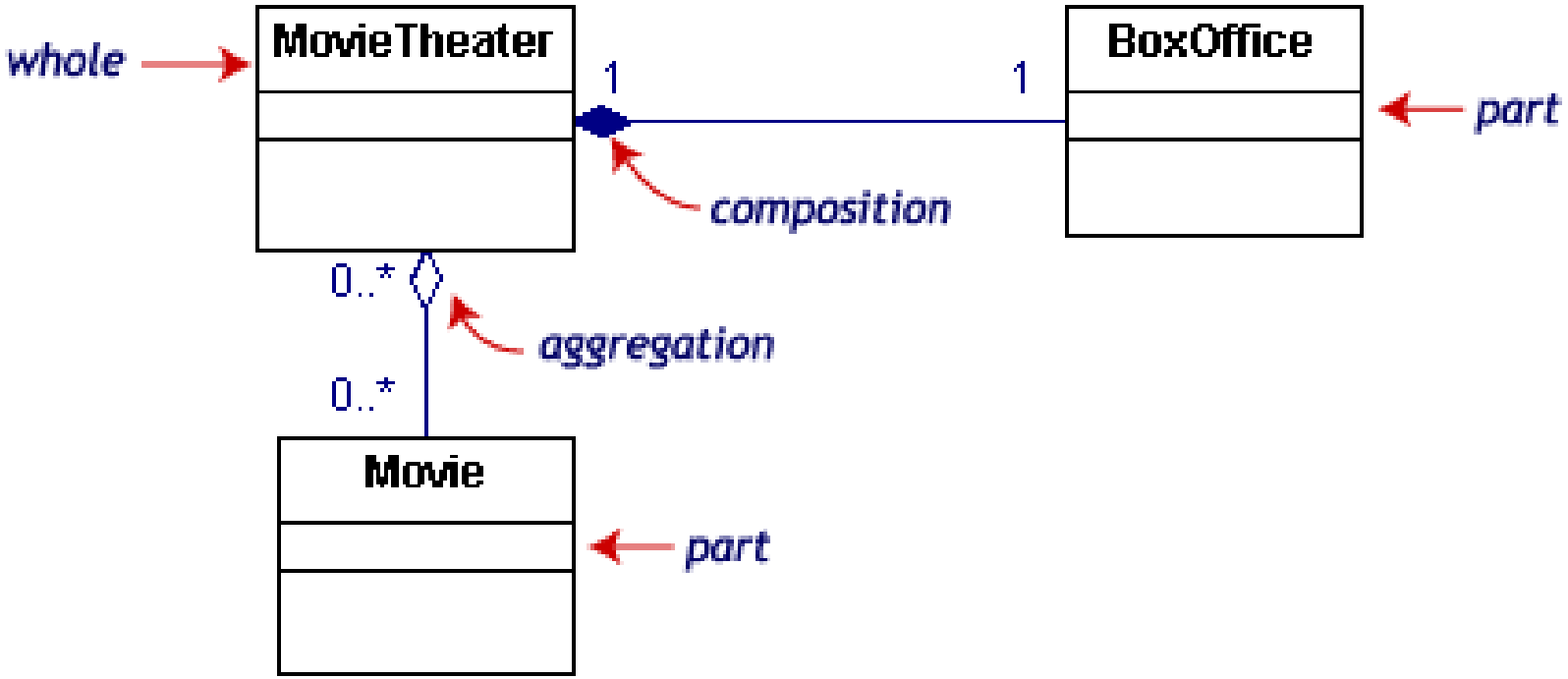
- **Aggregation**: “is part of”
  - symbolized by a clear white diamond
- **Composition**: “is entirely made of”
  - stronger version of aggregation
  - Expected to live and die with the whole
  - the parts live and die with the whole
  - Delete whole → Delete part
  - symbolized by a black diamond



- **Dependency**: “uses temporarily”
  - symbolized by dotted line
  - often is an implementation detail, not an intrinsic part of the object's state



Composition/aggregation example

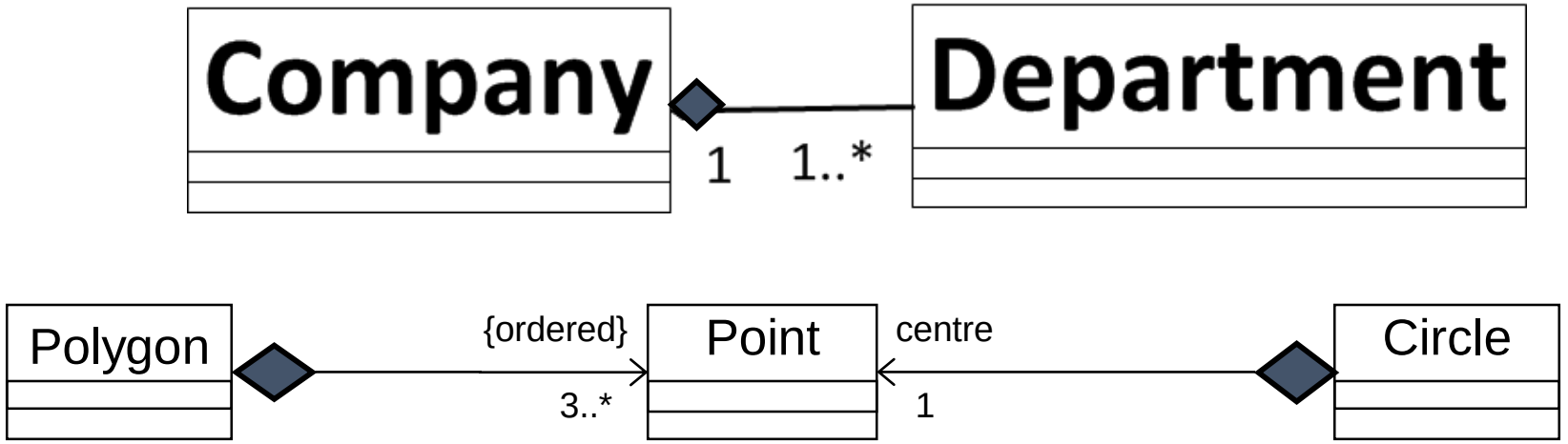


If the movie theater goes away

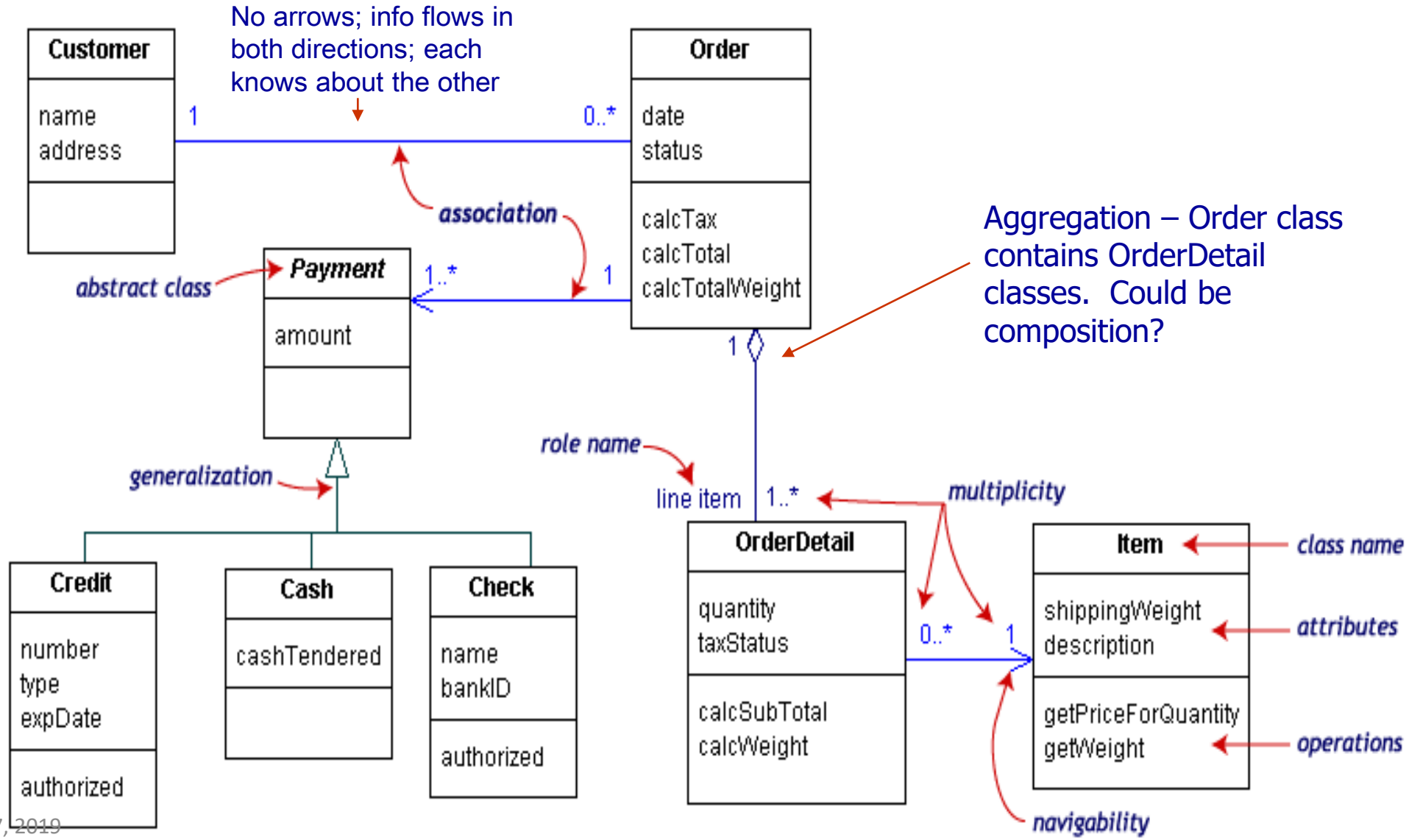
so does the box office => composition

but movies may still exist => aggregation

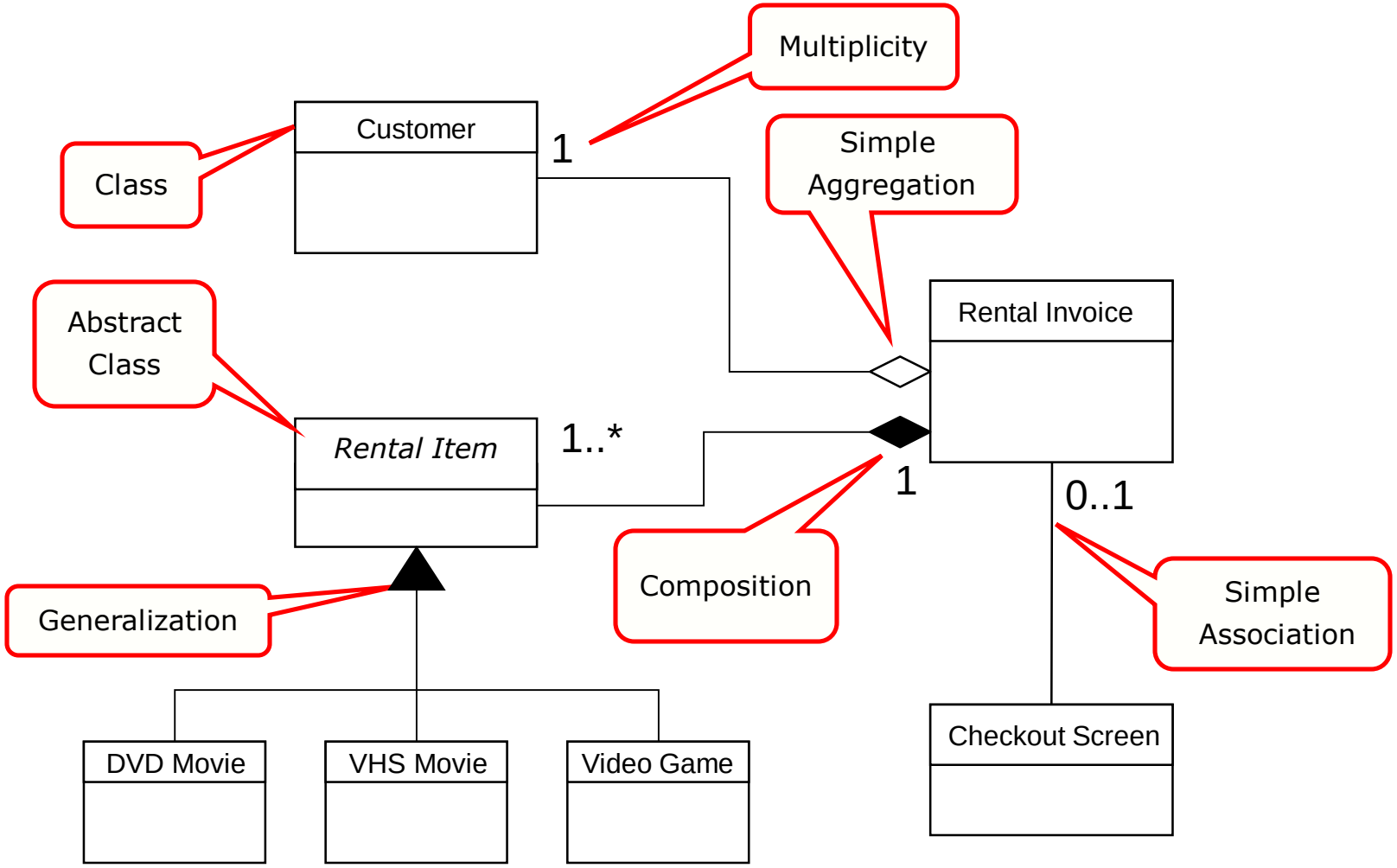
Composition/aggregation example



Class Diagram Example



Class Diagram Example: Video Store



Guidelines for designing Class Diagram

Guideline	Example
Nouns → imply *objects* or *classes*.	
A common or improper noun implies a class of objects.	"An employee serves the customer" implies two classes of objects, employee and customer.
A proper noun or direct reference implies an instance of a class.	"John addressed the issues raised by Susan" implies two instances of an object, John and Susan.
A collective noun implies a class of objects made up of groups of instances of another class.	"The list of students was not verified" implies that a list of students is an object that has its own attributes and methods.
Verbs → imply *associations* or *operations*.	
A doing verb implies an operation.	"Don files purchase orders" implies a "file" operation.
A being verb implies a classification association between an object and its class.	"Joe is a dog" implies that Joe is an instance of the dog class.
A having verb implies an aggregation or association relationship.	"The car has an engine" implies an aggregation association between car and engine.
A transitive verb implies an operation.	"Frank sent Donna an order" implies that Frank and Donna are instances of some class that has an operation related to sending an order.
A predicate or descriptive verb phrase implies an operation.	"If the two employees are in different departments, then …" implies an operation to test whether employees are or are not in different departments.
Adjectives → imply *attributes* of a class.	
An adjective implies an attribute of an object.	"All 55-year-old customers are now eligible for the senior discount" implies that age is an attribute.
Adverbs → imply an attribute of an *association* or *operation*.	
An adverb implies an attribute of an association or an attribute of an operation.	"John drives very fast" implies a speed attribute associated with the driving operation.

END