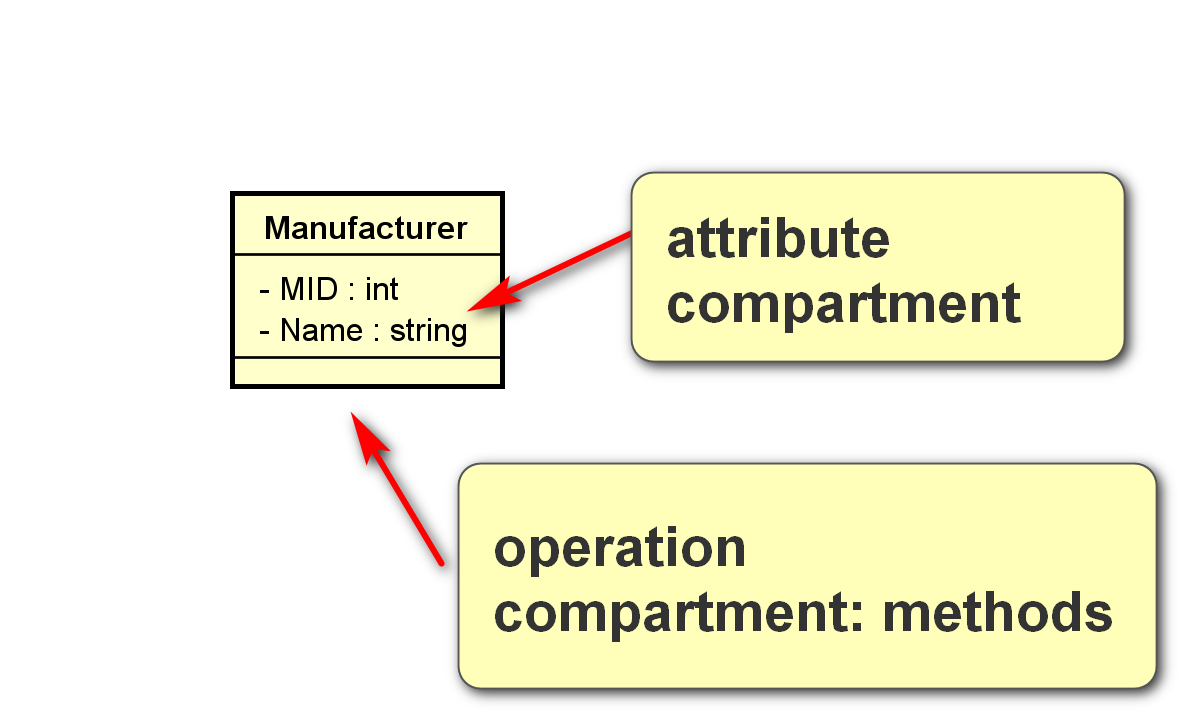
**9/7/2021**

Self-annotation:

Class diagram:

IN data modeling, operation compartment is more important than the attribute compartment.



Software modeling:

Operation compartment: many methods public.

Attributes compartment: many data private.

Data modeling: attributes mostly.

**CSCI 5333.1 DBMS  
Spring 2020  
Homework #1**

**Database Modeling in UML**

Download and install the *community* version of Astah's UML Editor: <http://astah.net/download>. If Astah provides free version for students, that should also be acceptable. This is one of the best free UML editors available. It has some restrictions but should be more than sufficient for our class.

The homework assignment is to model a drastically simplified toy Online Task Management System Application (Taskster). Multiplicities for attributes and associations should be as specific as possible. Attribute and association documentation are optional, but they are encouraged when not trivial. For example, the roles or names of associations should be displayed when appropriate. Operations for classes are not necessary. Use the stereotypes <<PK>>, <<CK>>, <<unique>> and <<derived>> for attributes when appropriate. Create suitable data types for SQL, such as SQL::Datetime, SQL::time, SQL::PhoneType, SQL::Decimal, etc. For string types, you may use 'string', 'varchar', 'char' or user-defined types when appropriate.

Save your files as <<last-name>>\_<<student>>\_h1\_class.asta (Astah's file format) and <<last-name>>\_<<student>>\_h1\_documentation.docx (optional; for additional information). Examples: bajaj\_0007007\_h1\_class.asta. Also include a PDF versions of your class diagram: <<last-name>>\_<<student>>\_h1\_class.pdf.

Submit your homework through Blackboard.

**Taskster**

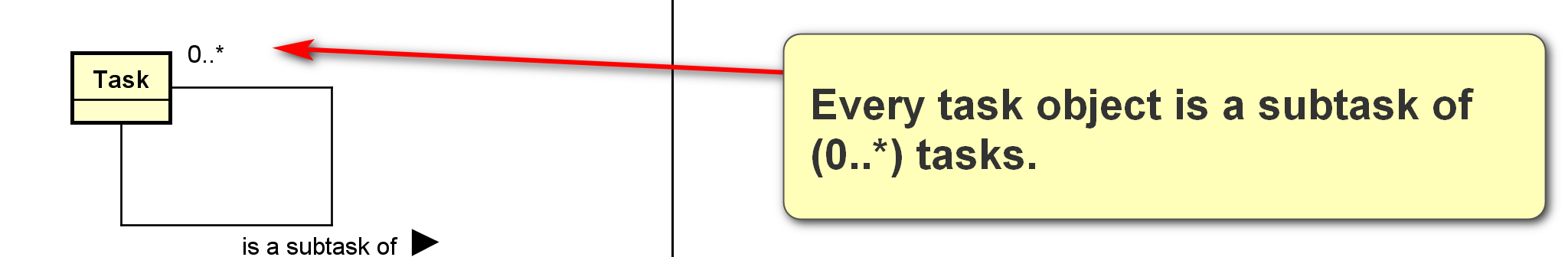
Create a data model to support a part of an online team project task application. This is a drastically simplified system that does not support many functions. Make reasonable assumptions. (it is a summary)

The basic idea is to support teams to define, store, access, label, and update tasks and sub-tasks for a team project. Access to the system can be made through multiple types of devices such as computers, smartphones, web services, etc.

Users must be registered as members to use Taskster. The system should store the last name, first name, a unique screen name, and a working email address for every member. The email address does not need to be unique. A unique member id is assigned by the system and the start time of the membership should be recorded. A member may be referred by another member to join Taskster. For example, Bun and Joe may be member.

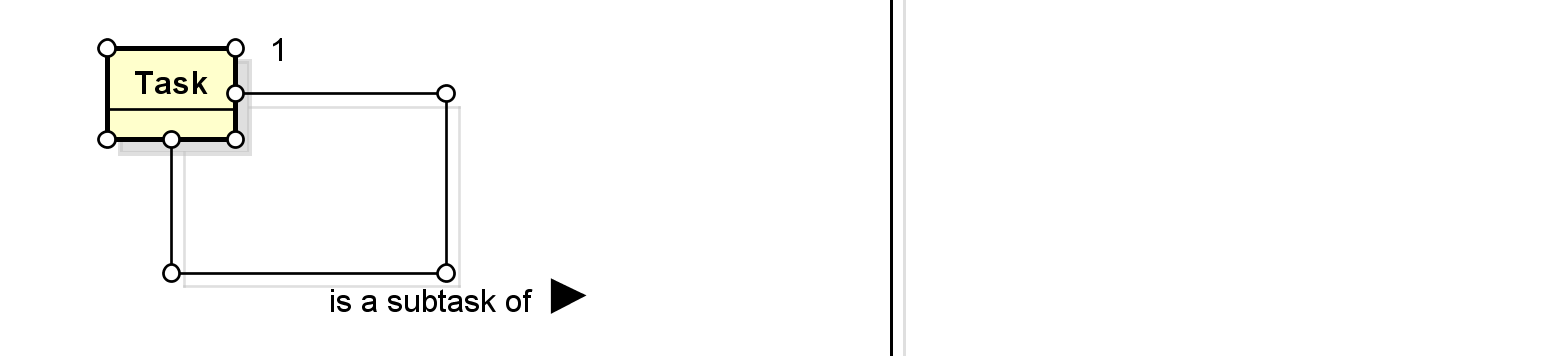
Concepts:

1. Member:
   1. Definition: A member can create any number of projects.
   2. Choice: class
2. Bun, Joe: instance of the class Member; member object. No need to model.
3. User:
   1. Definition: cannot come with definition -> may not be important concept. Differentiate between user and members in term of functions.
   2. Choice: no need to model.
4. Project:
   1. Definition: a project is created by a member to manage its tasks. Synonyms: team project. (Example: one concept many terms).
   2. Class
5. Team project
   1. Same as a project.
6. Device:
   1. Definition: an user agent for accessing data.
   2. Choice:
      1. Class
      2. No need to model: for the time being. Questions: do we want to keep the device information of the member?
7. computers, smartphones, web services:
   1. No need to model if device is not modeled.
   2. If device is a class -> attributes (attribute name, or attribute value) -> attribute name: device\_name
8. Task: class.
9. System:
   1. Choice: no need to model (the entire class diagram is the system.)
10. Subtask:
    1. Definition:
    2. Choice:
       1. No need: likely need to model.
       2. Class:
       3. Attributes:
       4. Associations: use this for the moment.
    3. Example: Task T1. Subtask T2.
       1. T1 is a task. OK. -> Task is more likely a class (exist by itself.)
       2. T2 is a sub-task. Not OK -> likely not a class.
          1. T2 (object) is a sub-task of (associate) T1 (object)
          2. T3 (Task object) is a sub-task of T2 (task objection).
       3. LName is a property of a member. OK -> more likely an attribute of the member class.



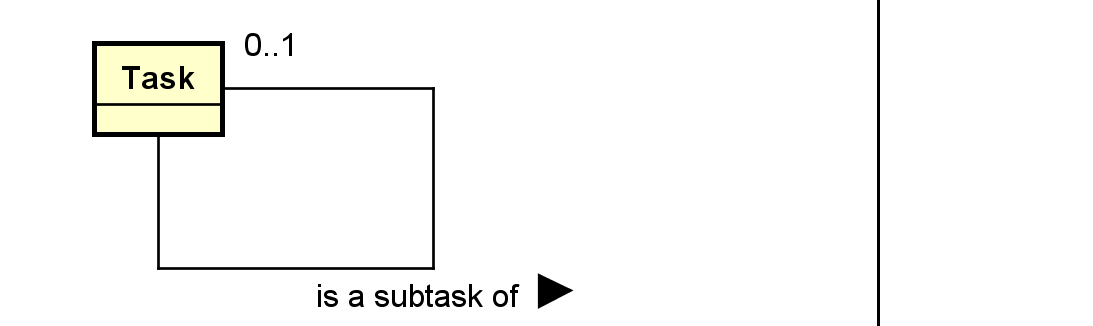
(0..\*) not correct because a task can have only one parent task.

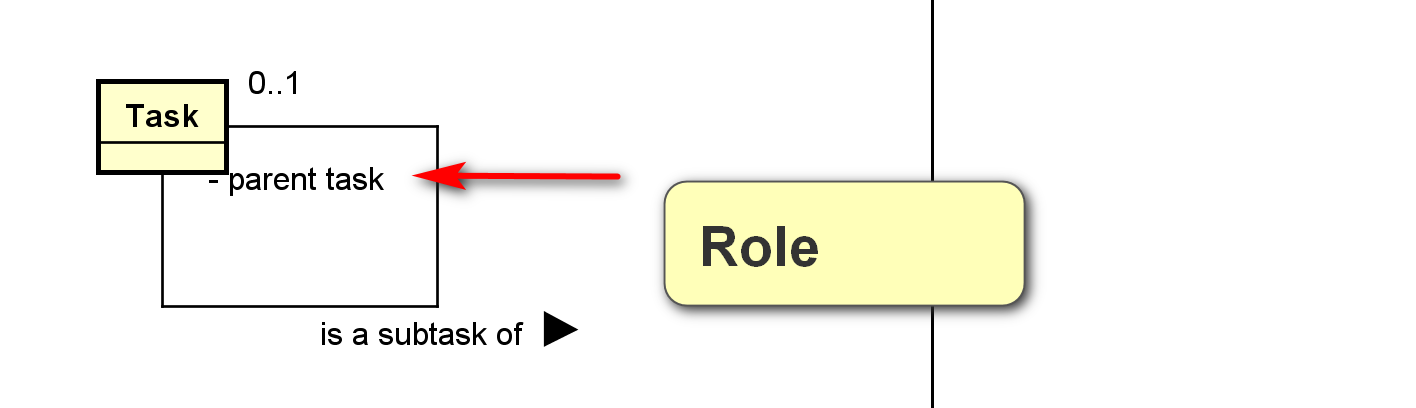
1?

? No

Meaning: every task object is a sub-task of (1..1) task.

Correct answer is (0..1) as a task may not have a parent task.





Every task has 0 to 1 parent task.

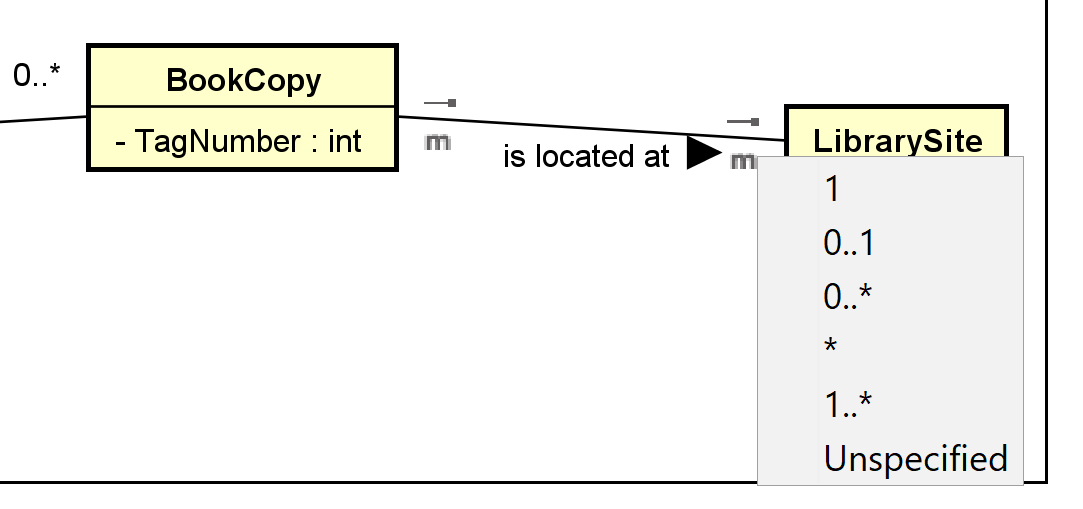


Multiplicity m of sub-task:

1. 1..\*
2. 0..\*: likely more reasonable.

[1] Every task object must have 1..\* sub-task objects associated with it.

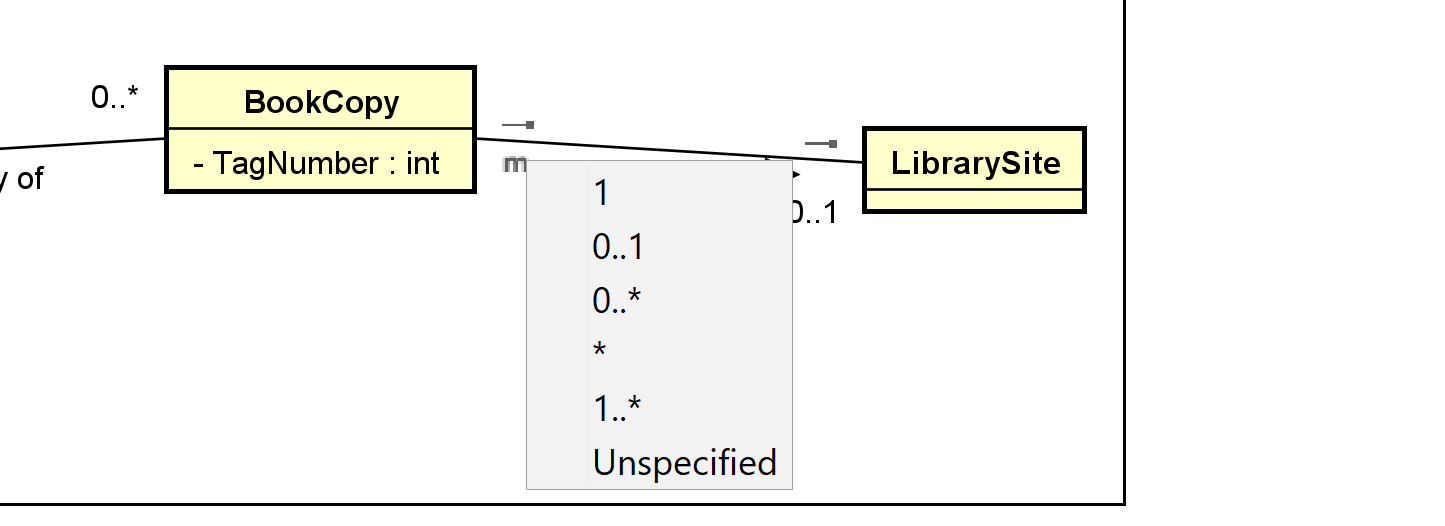
[2] Every task object must have 0..\* sub-task objects associated with it.



1. 1: 2
2. 0..1: 1
3. 0..\*: 2
4. \*:
5. 1..\*: 3
6. I don’t know: I don’t know the vote count.

A book copy object is located at m: likely 1 or 0..1 (in case lost is modeled), library site object.

Is located at: a book copy is physically located in a library site for check out.



1. 1:
2. 0..1:
3. 0..\*: 8
4. \*:
5. 1..\*:
6. I don’t know:
7. Last name: attribute of Member
8. First name: attribute of Member
9. Email: attribute of Member
10. Member Id attribute of Member
    1. Unique

An object does not need a primary key.

Built-in object identity.

Java is more object-oriented than Python.

1. Yes: 13
2. No: mostly correct.
3. I don’t know: 1

UML extension: stereotypes.

1. Author: Yue
2. PK: DB extension towards the relational model.
3. Start time: attribute of Member
4. Screen name: attribute of Member
   1. Unique

<<PK>> for screen name?

1. Yes: 3
2. No: 4 <- right choice
3. I don’t know:

IN the relational model, every relation has 1 and only 1 PK. Every relation has 1 or more candidate keys.

1. define, store, access, label, and update: operations drive the data requirements.
   1. Input and Output requirements -> drive data requirements.
2. ~~Taskster: system: see above.~~
3. Reference: A member may be referred by another member. Member M1 (object of the class Member) refers (verb: possibly an association) member M2 (object of the class Member).
   1. Operation.
   2. Class
   3. Attributes
   4. Association (class):
   5. No need

Choices:

1. Class: have attributes, form associations; exists by themselves, …
2. Attributes: do not have attributes, cannot form asociations; not exists by themselves. Simple.
3. Associations
4. No need to model

“One term many concepts”

Book:

1. Book copy: A physical copy of a book.
2. Book: The generic book with its content: “Elmasri, R. and Navathe, S. Fundamentals of Database Systems, 6th or 7th Edition, Addison-Wesley, Boston, MA.”

Each member must have a username and a password to access the system. However, an account may be used by a person other than a member, such as an Taskster's manager or a system administrator. The management and administration module of the system is not modeled in this assignment.

A member can create any number of projects. The creator of a project is known as the owner of the project. Each project may also has managers who have heightened privileges relative to other members of the project. However, only the owner can change the manager of a project. The owner and the manager can be the same person. A project can have any number of members. The time when a member joins a project should be recorded.

A member may have many roles in a project. Some of these role types are predefined by Taskster, for example, librarian, developer, team leader, supervisor, consultant, etc. They are standard role types. Some roles are project specific and can be defined by a manager of the project. For example, a software project may include roles such as modeler, programmer, technical writer, etc. When a project-specific role type is defined, an optional description and a creation time should be stored together with the role name. The manager who defined the role should also be noted. Every role, standard or user-defined, has a role level. The role level is an integer with a definition. Many roles can have the same role levels.

A project has an unique id. The project name and logo (an image) should be stored with a description and the creation time. A project has any number of tasks (to-do tasks). A task has a name and a description. A task may have any number of sub-tasks. There is no limit on the level of sub-tasks. A task can be created by any project member. The creator of a task should be noted. A task may have an expected completion time and an actual completion time. A task may be assigned to a team member. The creator and the assignee may or may not be the same project member.

A task has a current status. Taskster keeps track of status changes. A status has a name. There are standard predefined status names, such as 'start' or 'completed'. A standard status has a definition and an abbreviation that should be stored. Status may also be project-specific and user defined, such as 'first draft completed'. In this case, a description of the status should also be noted together with its name. For example, a status history for a task may be:

8/1/2019 13:22:05: 'started'  
8/1/2019 13:23:15: 'assigned'  
8/4/2019 17:24:15: 'first draft completed'  
8/4/2019 17:25:12: 'routed to audit'  
8/6/2019 09:24:25: 'second draft completed'  
8/4/2019 09:25:32: 'routed to audit'  
8/10/2019 11:11:28: 'completed'

Questions and discussions:

[1] What is the word "owner" stands in Member class relation

The owner of a projector is its creator and has the highest privilege in the project. (Synonym: creator). A project can only have one owner. No multiple ownership of projects.

[2] For our assignment, what's the difference between username and screen name? Why are they separate?

“The username is used for logon and thus a password should also be stored.

A screen name is the name that shown up in the screen when you log ong and use the Website.

E.g. username: “bunyue2021”; screen name: “UHCL DB evangelist”

[3] How many different types of relationships/associations exist in the class diagram?

Number of participants:

1. Binary association: 2 objects.
2. Tertiary association: 3 objects.
   1. E.g. Paul (student object) is assigned (verb: association) to the dorm Hunter (object: dorm) for Fall 2021 (semester object). -> promote to a class (dorm assignment). Add three associations to student, dorm and semester each.
3. N-ary association: n objects.

Natures: special kinds of associations:

1. A kind of: superclass and subclass.
2. A part of: component-whole association; aggregation and composition.

[4] Values of multiplicities in the class diagram?

Six supported by ASTA.

1. 1: {1}
2. 0..1{ {0,1}
3. 0..\*
4. \*
5. 1..\*
6. Unspecified
7. 2..5: {2,3,4,5}
8. [1,2,4,6,10.12]: {1,2,4,6,10.12}

Generalization discussion:

Example:

[1] two copies of A, B, C, D: redundant, potential inconsistency.

Class Y: A, B, C, D, E, F

Class Z: A, B, C, D, G

[2] Better reuse: one copy of ABCD. (More number of classes)

Class X as superclass: A, B, C, D

Class Y as a subclass of X: E, F

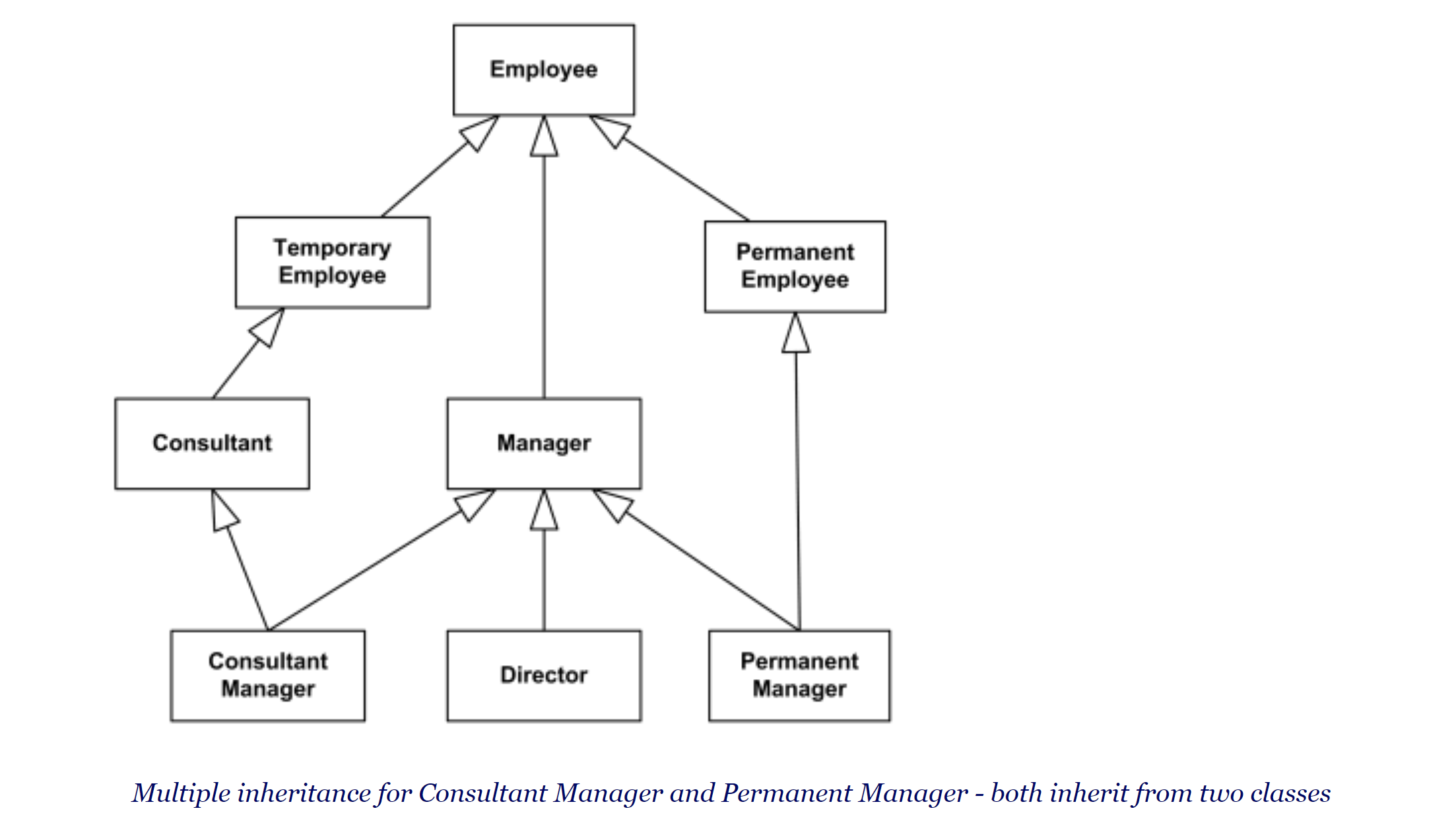
Class Z as a subclass of X: G

Another example:

Class Y: A, D, E, F

Class Z: A,, G, H, I

Likely not use generalization.



Distinguish between generalization-specialization with roles.

Employee:

1. SSN
2. LName
3. FName

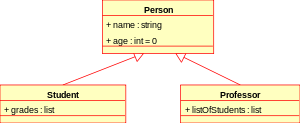
Management is a role/job position that an employee may take:

Employee – has (0..\*) job positions.

**Generalization/Inheritance**

From wiki:

The generalization relationship is also known as the [*inheritance*](https://en.wikipedia.org/wiki/Inheritance_(computer_science)) or *"is a"* relationship.



A student is a person.

A person takes on the role of a student.

Is-a -> a-kind-of

Bun is an ok person

Bun is a person <> Bun is a kind of person.

Bun is an instance of a person.

Better not modeled as generalization.