BRAC UNIVERSITY
Inspiring Excellence

CSE470: Software Engineering

Class Diagram

What is a Class

- A general template that we use to create specific instances or objects in the application domain
- Represents a kind of person, place, or thing about which the system will need to capture and store information
- Abstractions that specify the attributes and behaviors of a set of objects

What is an Object

- Entities that encapsulate state(attributes) and behavior(methods)

- Each object has an identity
  - It can be referred individually
  - It is distinguishable from other objects (separate memory location for each object)

Potential Classes

- External entities (e.g., other systems, devices, people) that produce or consume information to be used by a computer-based system.
- Things (e.g., reports, displays, letters, signals) that are part of the information domain for the problem.
- Occurrences or events (e.g., a property transfer or the completion of a series of robot movements) that occur within the context of system operation.
- Roles (e.g., manager, engineer, salesperson) played by people who interact with the system.
- Organizational units (e.g., division, group, team) that are relevant to an application.
- Places (e.g., manufacturing floor or loading dock) that establish the context of the problem and the overall function of the system.
- Structures (e.g., sensors, four-wheeled vehicles, or computers) that define a class of objects or related classes of objects.

Types of Classes during Analysis

Concrete

- Class from application domain
- Example: Customer class and Employee class

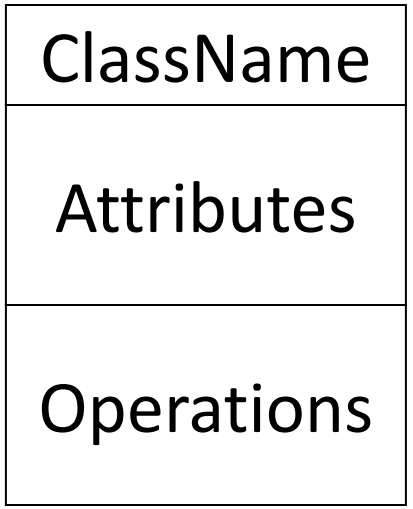
Abstract

- Useful abstractions
- Example: Person class

Classes

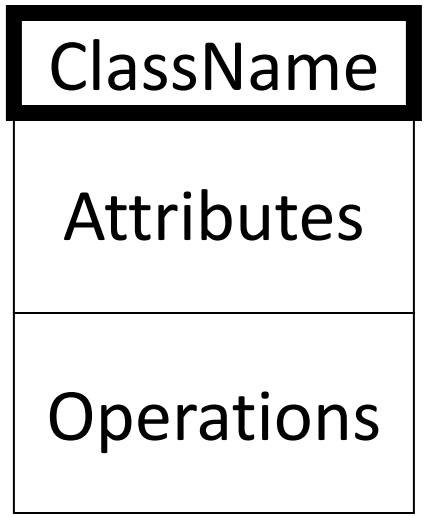
A class is a description of a set of objects that share the same attributes, operations, relationships, and semantics.

Graphically, a class is rendered as a rectangle, usually including its name, attributes, and operations in separate, designated compartments.



Class Name

The name of the class is the only required tag in the graphical representation of a class. It always appears in the top-most compartment.

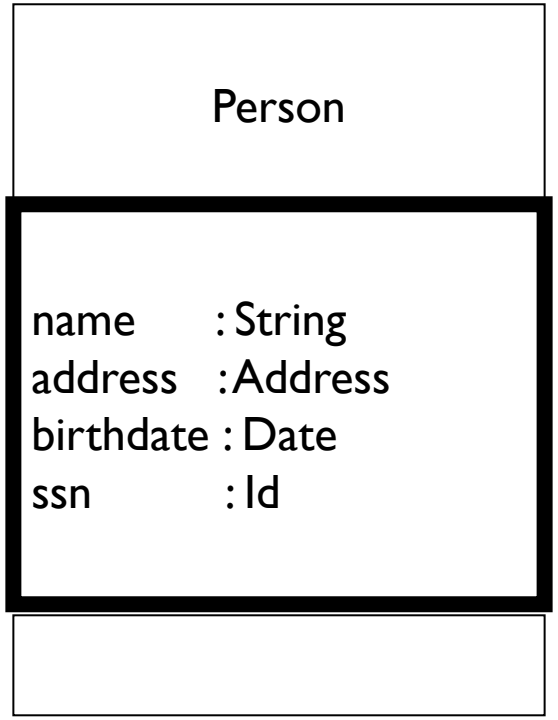


Attributes in a Class

- Properties of the class about which we want to capture information
- Represents a piece of information that is relevant to the description of the class within the application domain
- Only add attributes that are primitive or atomic types like Boolean, string, char, integer, float, double, date etc.
- Derived attribute
  - Attributes that are calculated or derived from other attributes
  - Denoted by placing slash (/) before name

Class Attributes

An attribute is a named property of a class that describes the object being modeled.
In the class diagram, attributes appear in the second compartment just below the name-compartment.



Class Attributes

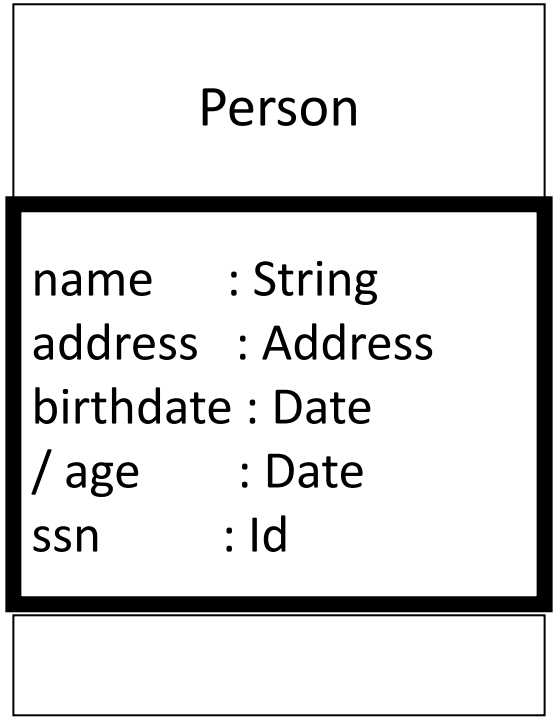
Attributes are usually listed in the form:

- attributeName : Type

A derived attribute is one that can be computed from other attributes, but doesn't actually exist. For example, a Person's age can be computed from his birth date.

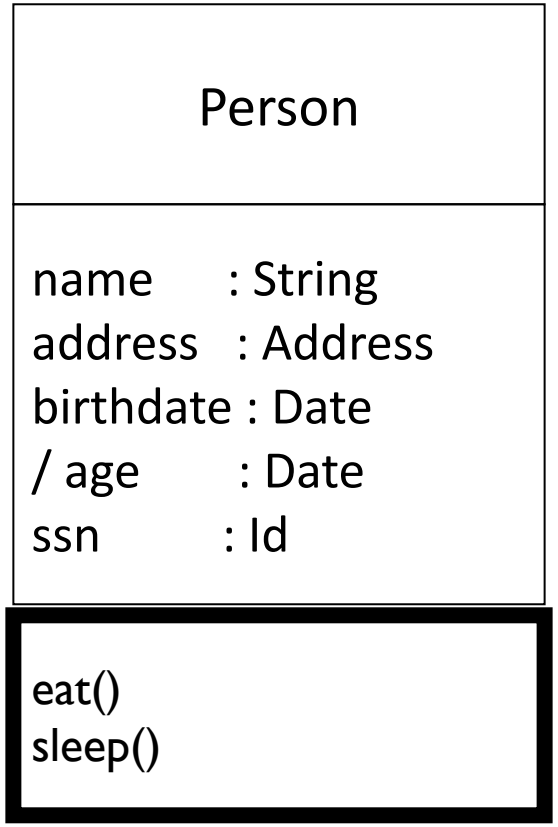
A derived attribute is designated by a preceding '/' as in:

- / age : Date



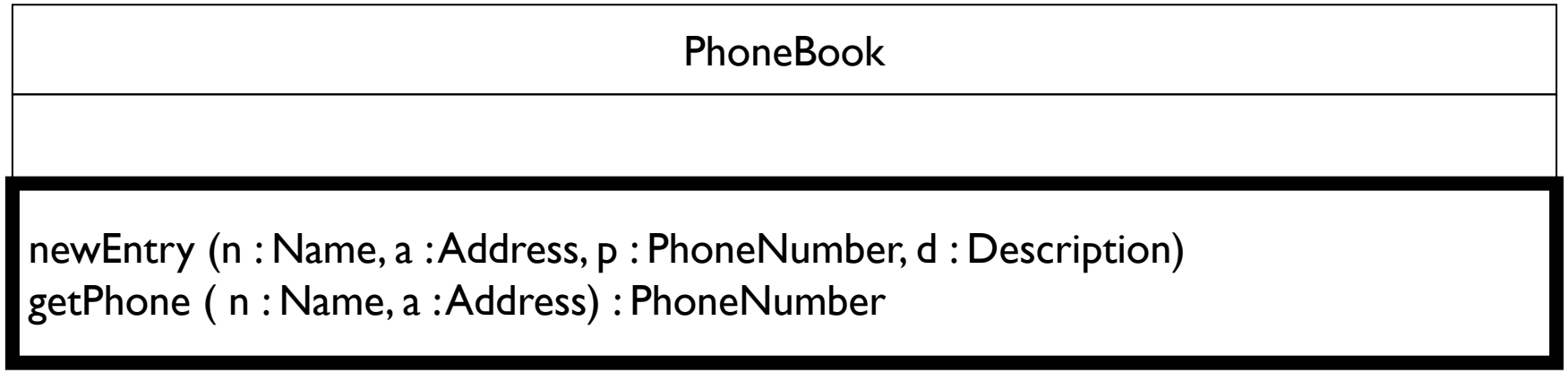
Class Operations

- Represents the actions or functions that a class can perform
- Describes the actions to which the instances of the class will be capable of responding
- Can be classified as a constructor, query, or update operation
- Operations describe the class behavior and appear in the third compartment.



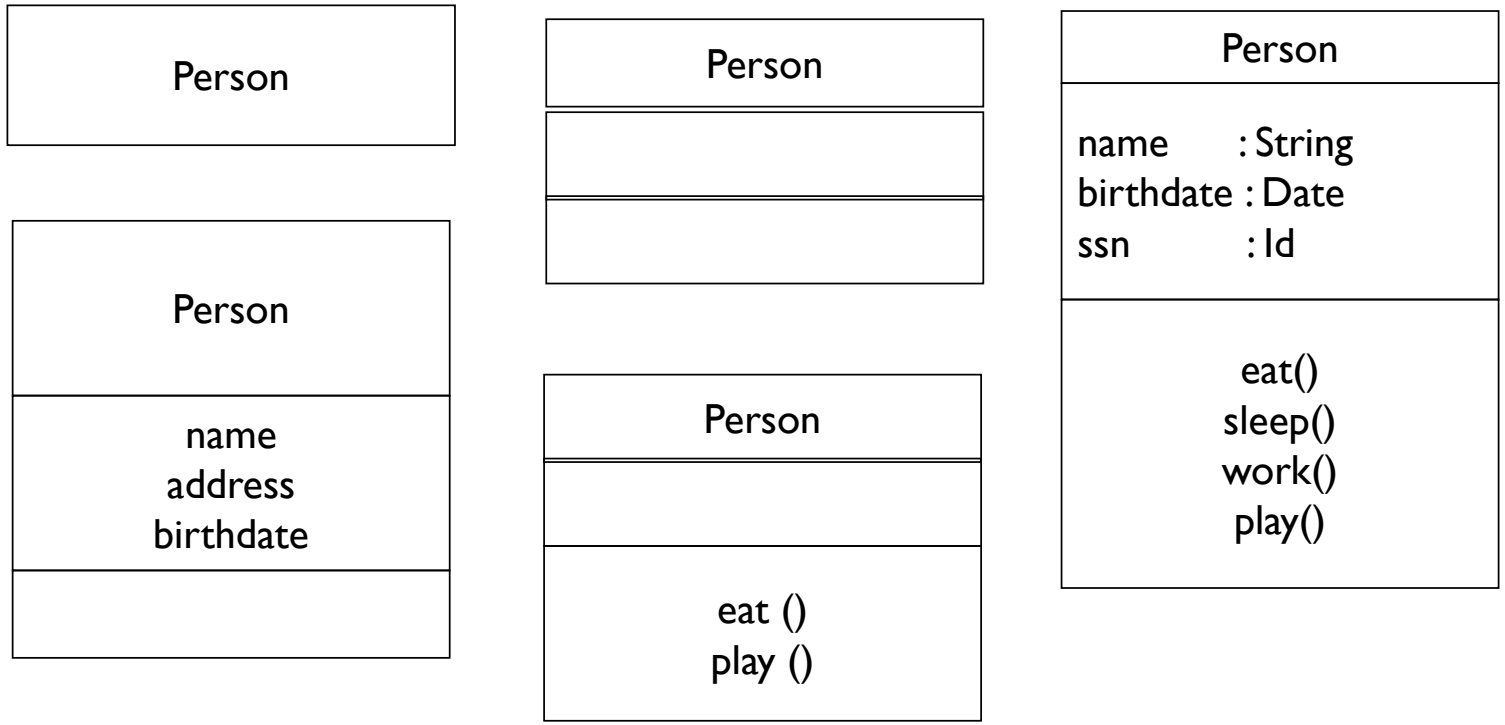
Class Operations

You can specify an operation by stating its signature: listing the name, type, and default value of all parameters, and, in the case of functions, a return type.



Drawing Classes

When drawing a class, you need not show attributes and operation in every diagram.

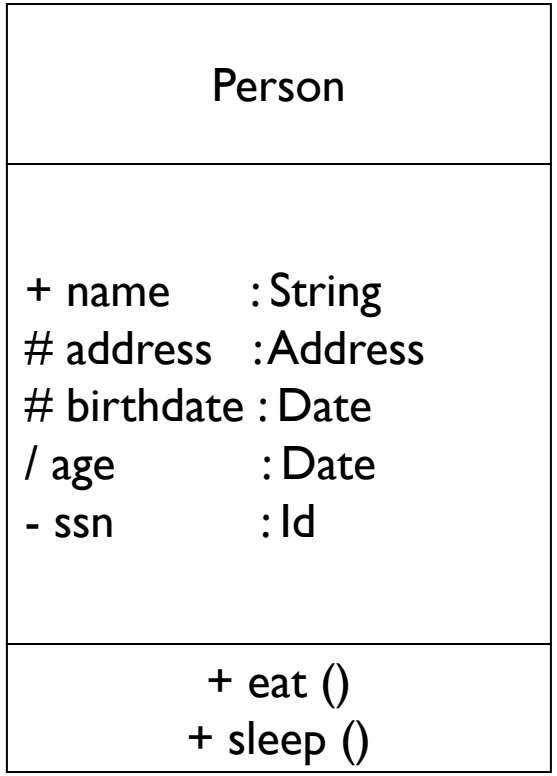


Visibility of Attributes and Operations

You can specify an operation by stating its signature: listing the name, type, and default value of all parameters, and, in the case of functions, a return type.

Visibility	Symbol	Accessible to
Public	+	All objects within your system
Protected	#	Instances of the implementing class and its subclasses
Private	-	Instances of the implementing class
Package	~	Instances within the same package

Visibility of Attributes and Operations



Relationships among Classes

Represents a connection between multiple classes or a class and itself
2 basic categories:

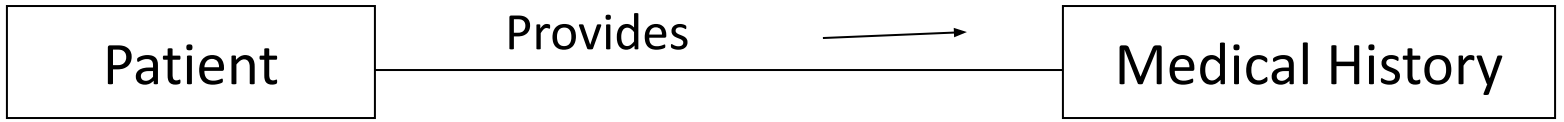
- Association relationships
  - Aggregation
  - Composition
  - Reflexive
  - Dependency
- Generalization relationships

Association Relationship

If two classes in a model need to communicate with each other, there must be link between them. An association denotes that link.

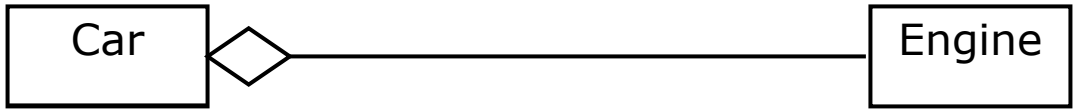
Name of relationship type shown by:

- drawing line between classes
- labeling with the name of the relationship
- indicating with a small solid triangle beside the name of the relationship the direction of the association



Aggregation Relationship

- Specialized form of association in which a whole is related to its part(s)
- Represented by a-part-of relationship
- Specifies a whole-part relationship between an aggregate (a whole) and a constituent part, where the part can exist independently from the aggregate.
- Denoted by a hollow diamond on the association



Aggregation Relationship

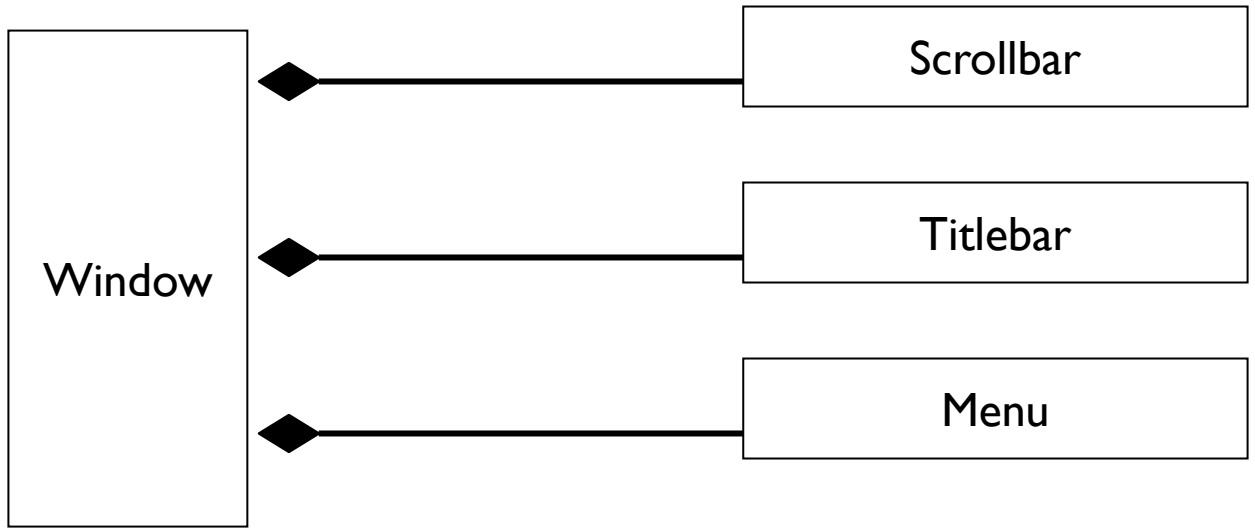
```
class Office:
    def __init__(self, room_number, phone_number):
        self.room_number = room_number
        self.phone_number = phone_number

class Employee:
    def __init__(self, office):
        self.office = office

office_100 = Office("100", "423-434")
e1 = Employee(office_100)
e2 = Employee(office_100)
```

AGGREGATION

Composition Relationship

A composition indicates a strong ownership and coincident lifetime of parts by the whole.

- Represented by a "is entirely made of" relationship
- stronger version of aggregation
- the parts live and die with the whole
- symbolized by a black diamond



Composition Relationship

```
class Office:
    def __init__(self, room_number, phone_number):
        self.room_number = room_number
        self.phone_number = phone_number

class Employee:
    def __init__(self, room_number, phone_number):
        self.office = Office(room_number, phone_number)
```

Employee owns office

COMPOSITION

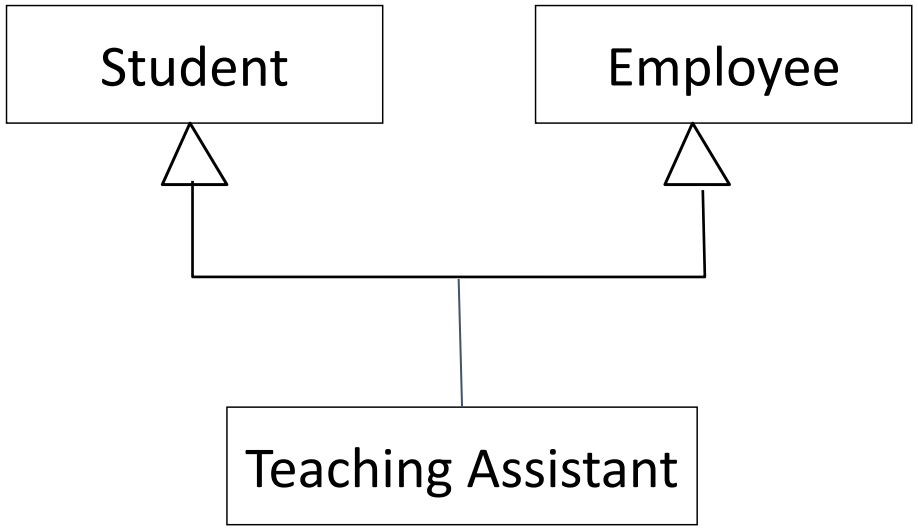
Generalization Relationship

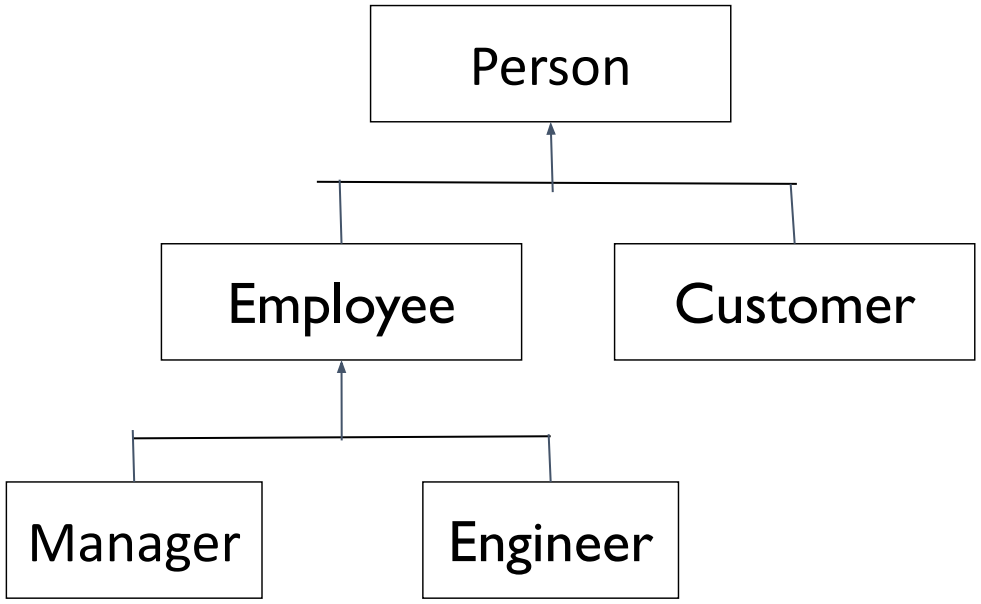
A generalization connects a subclass to its superclass. It denotes an inheritance of attributes and behavior from the superclass to the subclass and indicates a specialization in the subclass of the more general superclass.

- Represented by a-kind-of relationship
- Dog is a kind of Animal

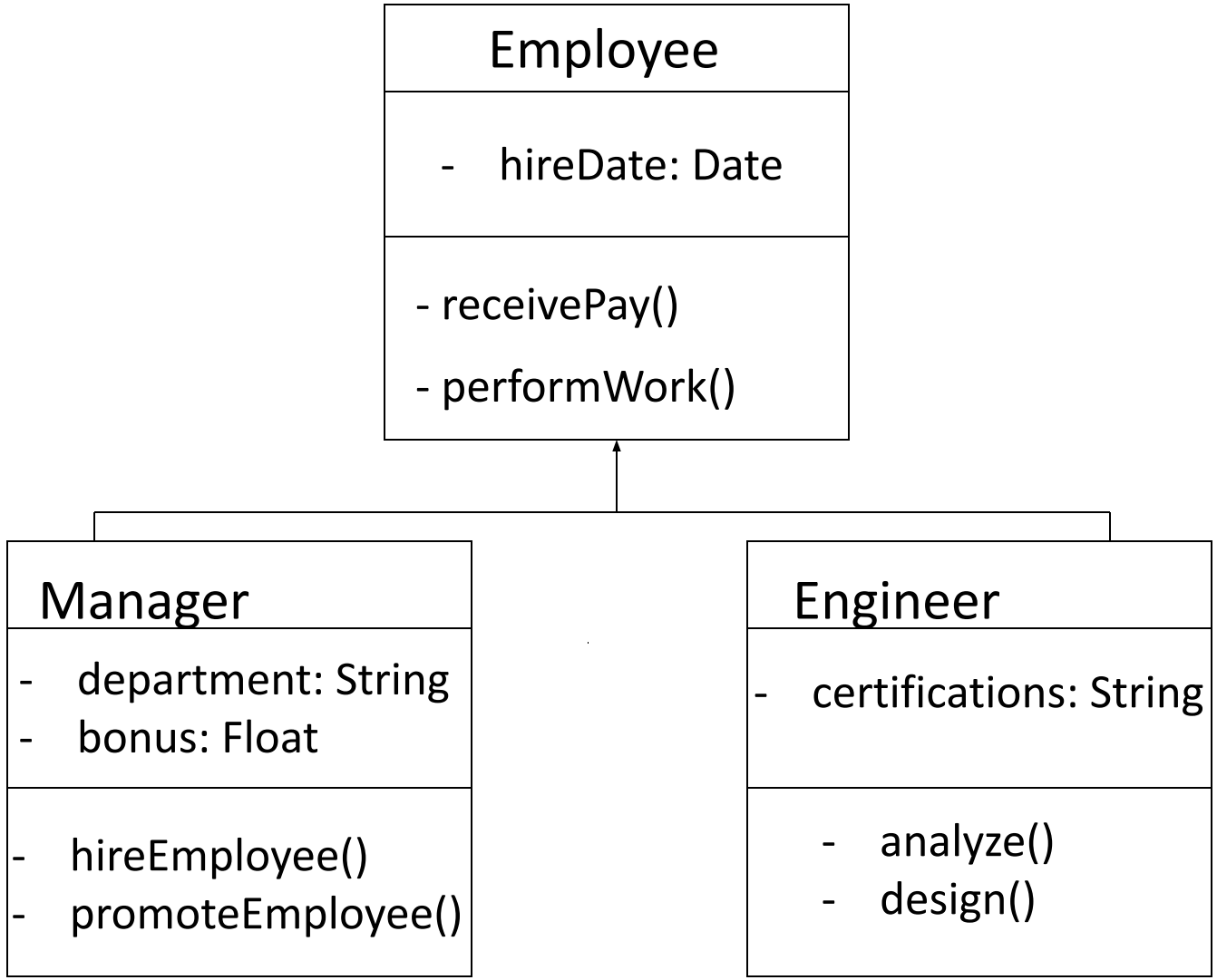
Generalization Relationship

UML permits a class to inherit from multiple superclasses, although some programming languages (e.g., Java) do not permit multiple inheritance.





Generalization Relationship



Multiplicity

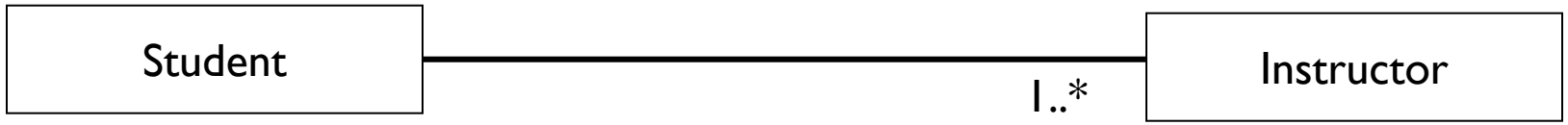
The multiplicity of a property is an indication of how many objects may fill the property. The most common multiplicities you will see are:

- **1** (An order must have exactly one customer.)
- **0..1** (A corporate customer may or may not have a single sales rep.)
- ***** (A customer need not place an Order and there is no upper limit to the number of Orders a Customer may place—zero or more orders.)

Multiplicity

We can indicate the multiplicity of an association by adding multiplicity adornments to the line denoting the association.

The example indicates that a Student has one or more Instructors:



The example indicates that every Instructor has one or more Students:

Multiplicity

The following denotes the minimum number.. maximum number of instances

- Exactly one 1
- Zero or more 0..* or 0..m
- One or more 1..* or 1..m
- Zero or one 0..1
- Specified range 2..4
- Multiple, disjoint ranges 1..3, 5

Class Task

You've been assigned the development of a Library Management System. The library system is composed of multiple branches, each housing a collection of books and multimedia resources. The library has 3 types of items: Books, DVDs, and Journals. Each library item is identified by a distinct code, contains information about its acquisition date and a specific author or creator.

Library patrons can engage in activities such as borrowing items, returning them, and checking the availability of resources. In addition to the general library items, special features are implemented for different types of items. For instance, books can be reserved, DVDs have a return date, and Journals provide access to specific articles.
Library patrons are allowed to have multiple borrowed items, but each item is uniquely associated with one patron. The library is managed by a team of employees, each having a unique ID, name, hiring date, and expertise level indicating their experience in the library system.

The library staff is divided into two categories: Librarians and Assistants. Both roles have distinct attributes and methods tailored to their specific responsibilities within the library system. It is essential to ensure that the attributes and methods are appropriately configured for Librarians and Assistants to efficiently manage the library resources and provide quality service to patrons.

Create a class diagram based on the given scenario.

Class Task - Solve