CSE 470 – Class Diagram in UML

BRAC University

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 and behavior
- Each object has an identity
  - It can be referred individually
  - It is distinguishable from other objects

Types of Classes

- Ones found during analysis:
  - people, places, events, and things about which the system will capture information
  - ones found in application domain
- Ones found during design
  - specific objects like windows and forms that are used to build the system

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.
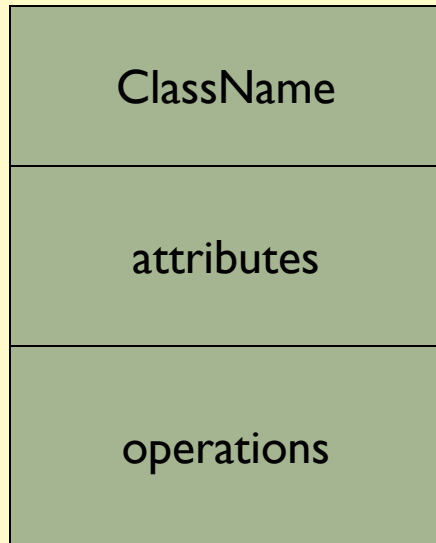
2 Kinds of Classes during Analysis

- Concrete
  - Class from application domain
  - Example: Customer class and Employee class
- Abstract
  - Useful abstractions
  - Example: Person class

BRAC
UNIVERSITY

Inspiring Excellence

Classes

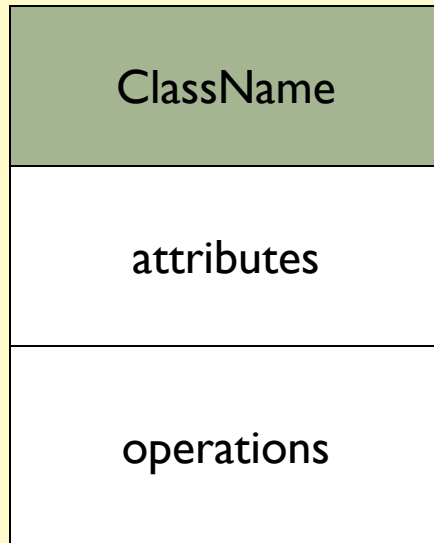| ClassName |
|-----------|
| attributes |
| operations |

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 Names

ClassName
attributes
operations

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.

BRAC
UNIVERSITY

Inspiring Excellence

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
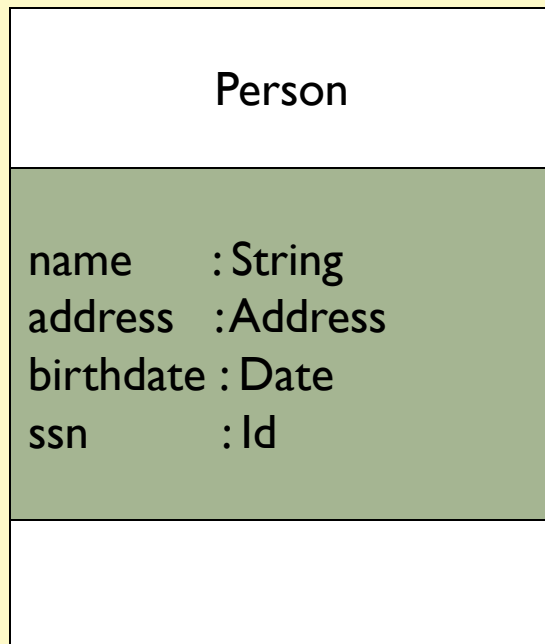
Attributes in a Class

- Only add attributes that are primitive or atomic types
- Derived attribute
  - attributes that are calculated or derived from other attributes
  - denoted by placing slash (/) before name

Class Attributes

Person
name : String address : Address birthdate : Date ssn : Id

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 (Cont'd)

Person
name   : String address   : Address birthdate : Date / age   : Date ssn   : Id

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

BRAC UNIVERSITY
Inspiring Excellence
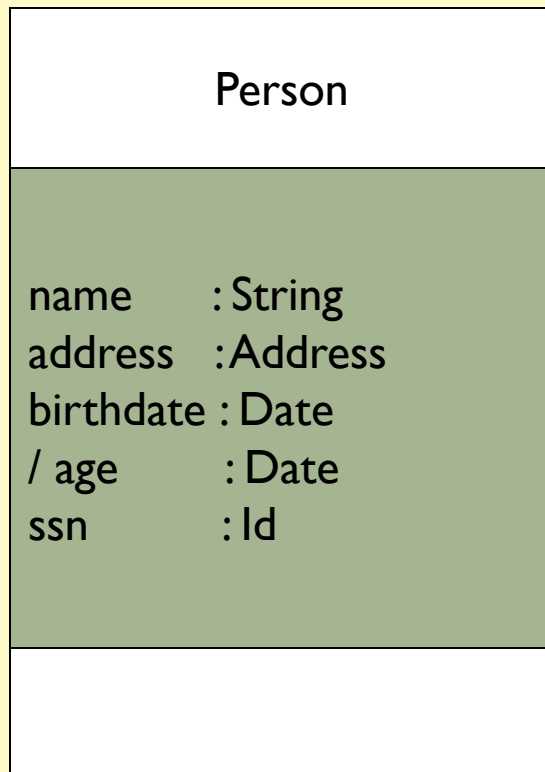
Operations in a Class

- 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

Class Operations

Person
name   : String address   : Address birthdate : Date ssn   : Id
eat () sleep () work () play ()
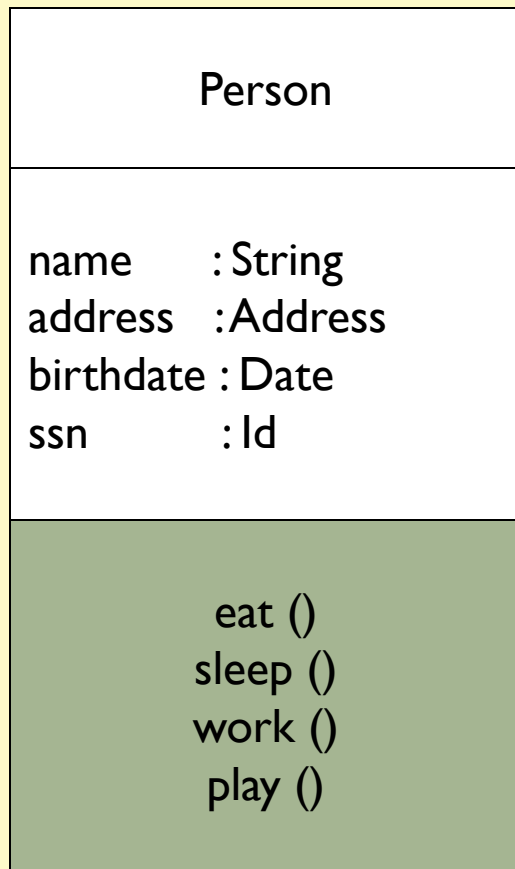
Operations describe the class behavior and appear in the third compartment.

Class Operations (Cont'd)

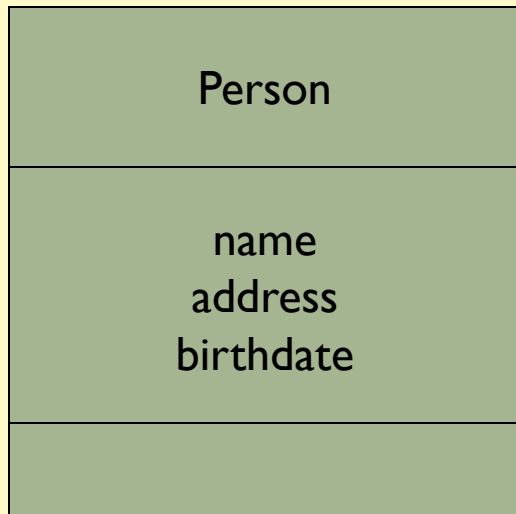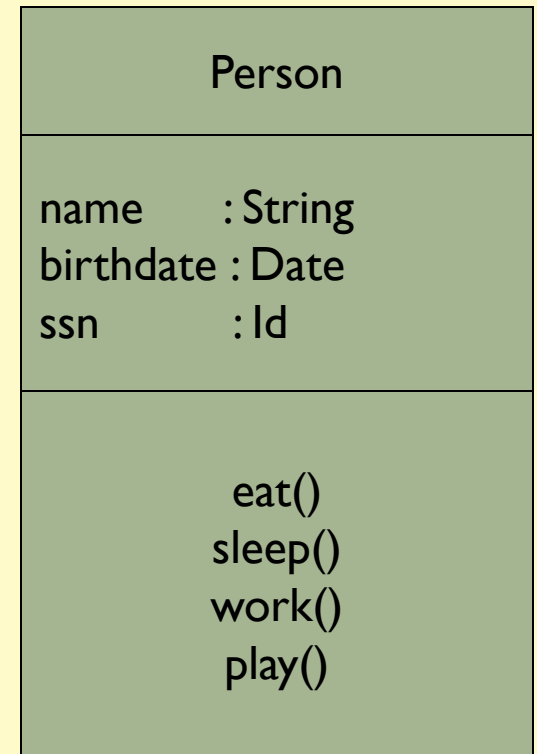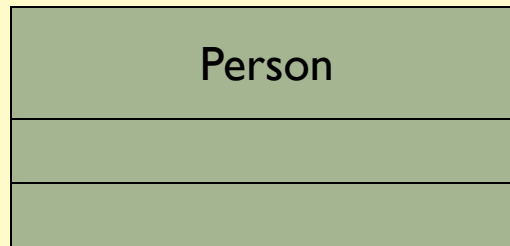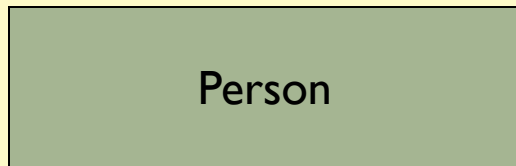| PhoneBook |
|---|
| |
| newEntry (n : Name, a : Address, p : PhoneNumber, d : Description)
getPhone (n : Name, a : Address) : PhoneNumber |

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.
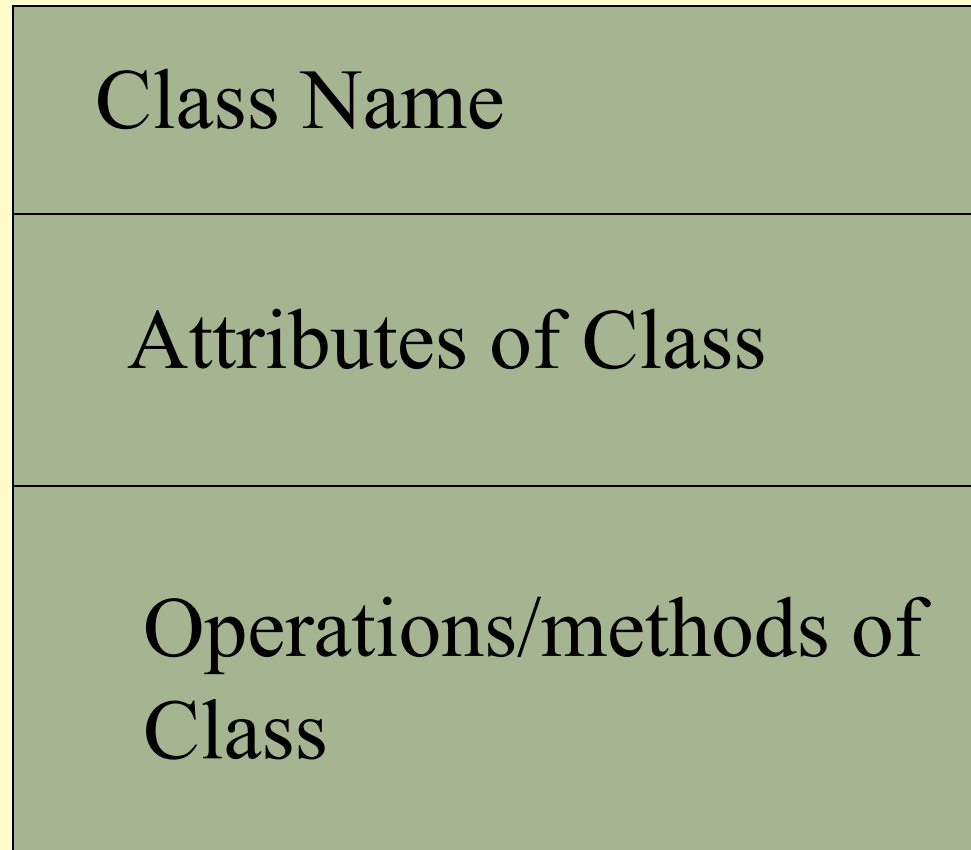
Depicting Classes

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

```
┌─────────────────────┐
│                     │
│       Person        │
│                     │
└─────────────────────┘
```

```
┌─────────────────────┐
│       Person        │
├─────────────────────┤
│                     │
├─────────────────────┤
│                     │
└─────────────────────┘
```

```
┌─────────────────────┐
│       Person        │
├─────────────────────┤
│ name       : String │
│ birthdate  : Date   │
│ ssn        : Id     │
├─────────────────────┤
│       eat()         │
│      sleep()        │
│      work()         │
│      play()         │
└─────────────────────┘
```

```
┌─────────────────────┐
│       Person        │
├─────────────────────┤
│       name          │
│      address        │
│     birthdate       │
├─────────────────────┤
│                     │
└─────────────────────┘
```

```
┌─────────────────────┐
│       Person        │
├─────────────────────┤
│                     │
├─────────────────────┤
│       eat ()        │
│      play ()        │
└─────────────────────┘
```

BRAC
UNIVERSITY

Inspiring Excellence
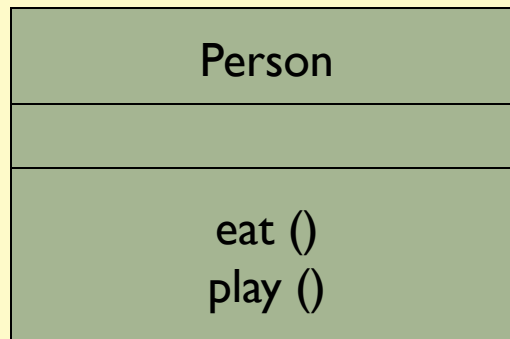
UML Representation of Class

Class Name
Attributes of Class
Operations/methods of Class

Visibility
of Attributes and Operations

- Relates to the level of information hiding to be enforced

Visibility
of Attributes and Operations

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.

Visibility (Cont'd)

Person
+ name   : String # address   : Address # birthdate : Date / age   : Date - ssn   : Id
+ eat () + sleep ()
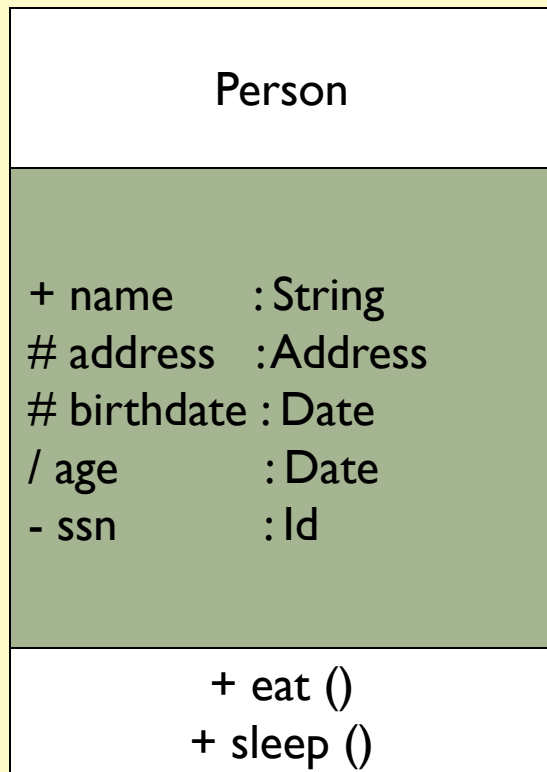
Attributes can be:
    + public
    # protected
    - private
    / derived

BRAC
UNIVERSITY

Inspiring Excellence

Relationships among Classes

- Represents a connection between multiple classes or a class and itself

- 2 basic categories:
  - association relationships
    - Aggregation
    - Composition
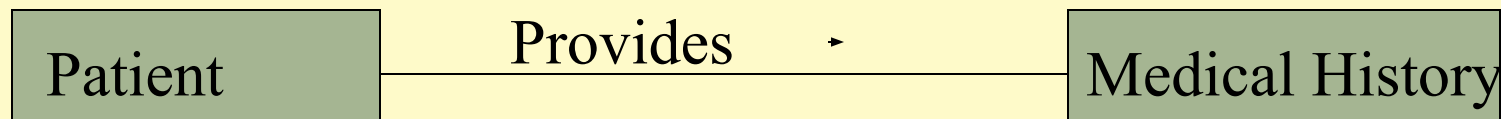 - generalization relationships

Association Relationship

- A bidirectional semantic connection between classes

- Type:
  - name of relationship
 - role that classes play in the relationship

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

| Patient | Provides ▸ | Medical History |

Association Relationships

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

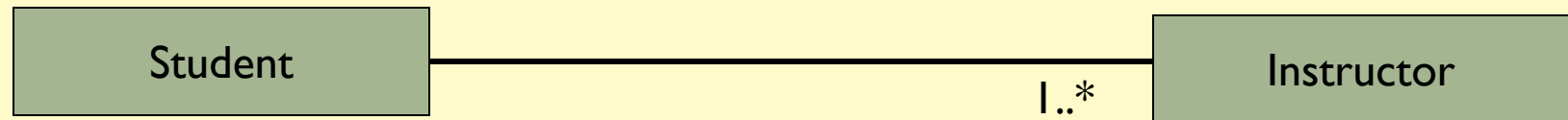An *association* denotes that link.

| Student	Instructor
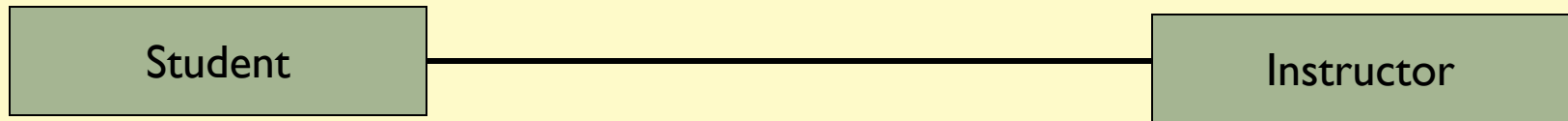
Association Relationships (Cont'd)

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*:

Association Relationships (Cont'd)

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

Student

1..*

Instructor
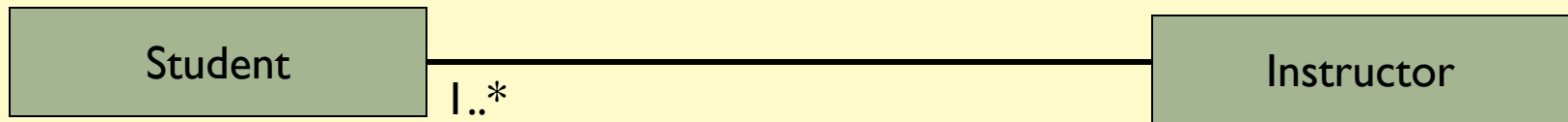
Association Relationship

- Role type shown by:
  - drawing line between classes
  - indicating with a plus sign before the role name

```
┌──────────┐
│ Patient  │───────────┐
└──────────┘           │
     │                 │        + primary insurance
     │                 │        carrier
     └─────────────────┘
```
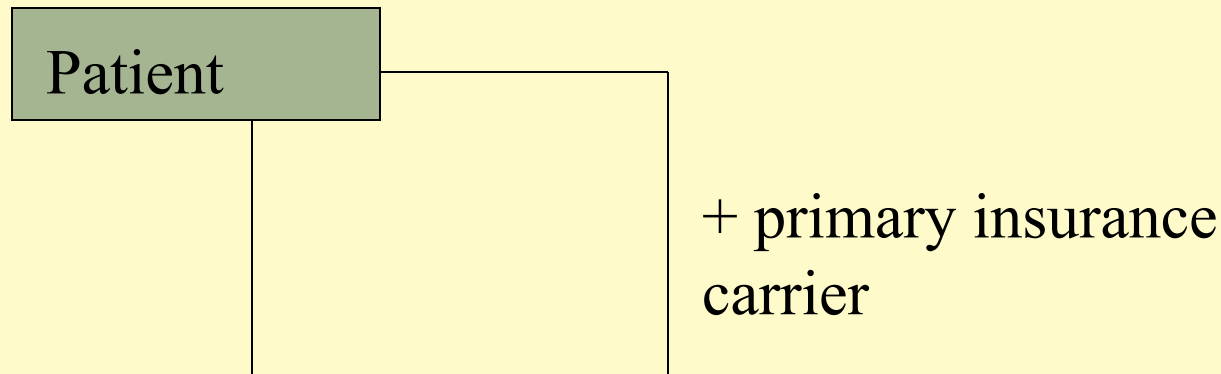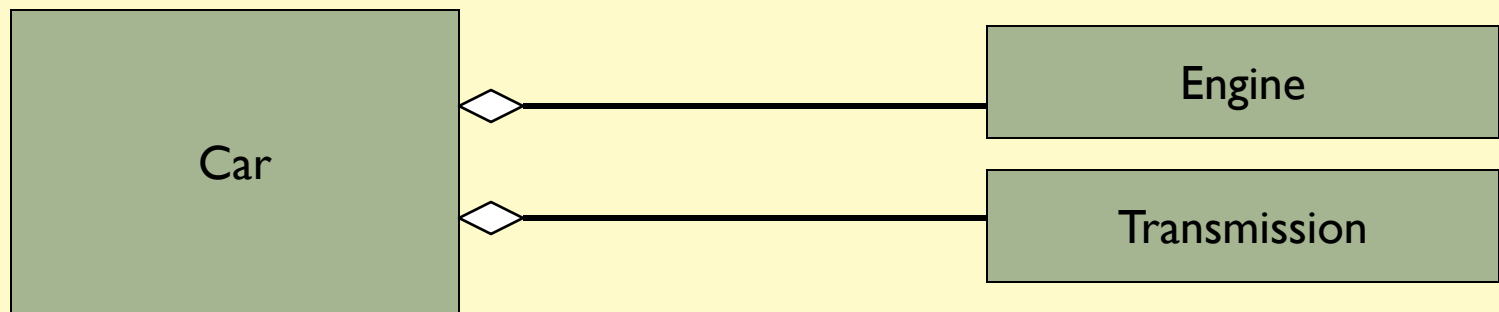
Aggregation Relationship

- 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.

BRAC
UNIVERSITY

Inspiring Excellence

Aggregation Relationship

☐ Aggregations are denoted by a hollow-diamond adornment on the association.

```
┌──────────────┐            ┌──────────────────┐
│              │◇───────────│      Engine      │
│     Car      │            └──────────────────┘
│              │◇───────────┌──────────────────┐
│              │            │   Transmission   │
└──────────────┘            └──────────────────┘
```

Composition Relationship

☐ A *composition* indicates a strong ownership and coincident lifetime of parts by the whole (*i.e.,* they live and die as a whole). Compositions are denoted by a filled-diamond adornment on the association
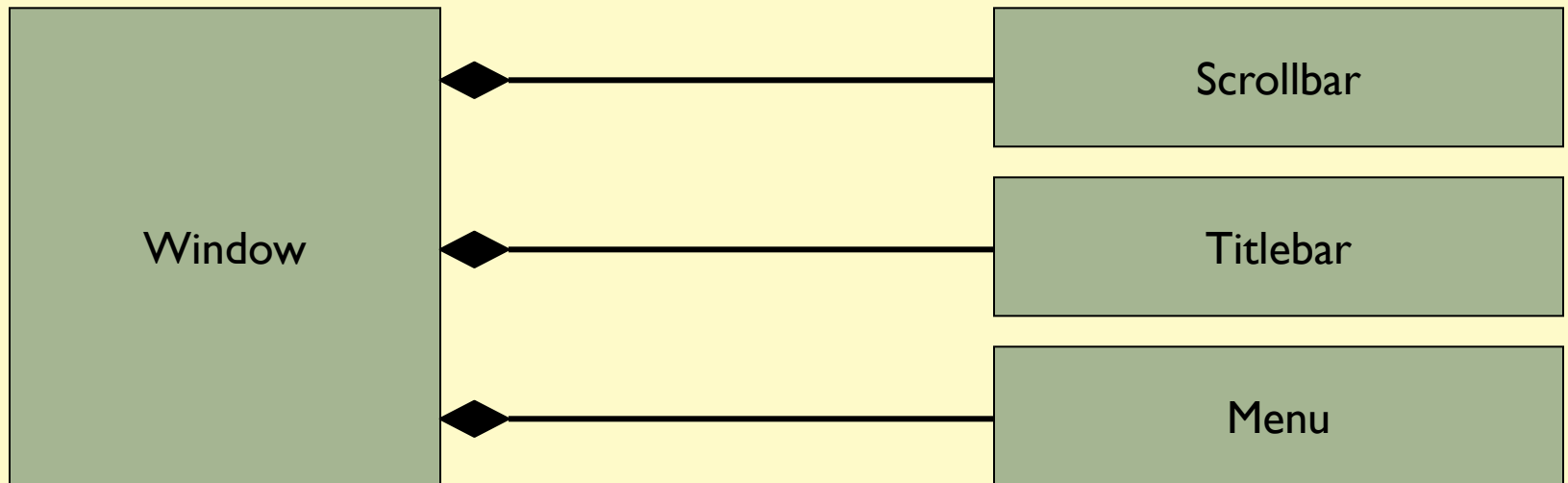
Generalization Relationship

- Enables the analyst to create classes that inherit attributes and operations of other classes
- Represented by *a-kind-of* relationship

Generalization Relationships

Person

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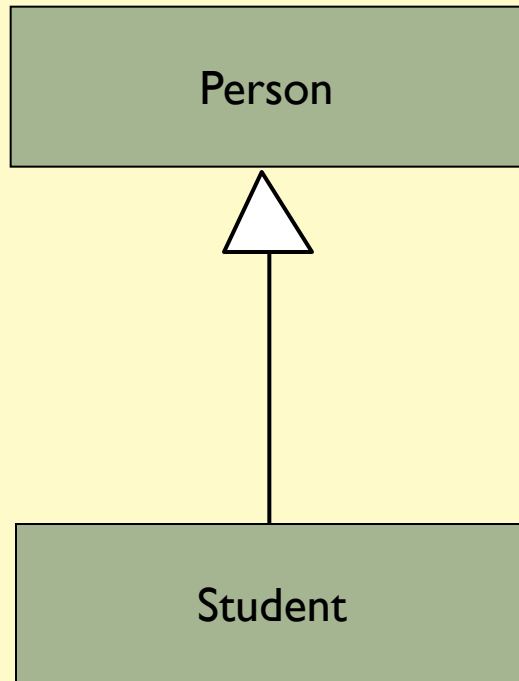
A *generalization* connects a subclass to its superclass. It denotes an inheritance of attributes and behavior from the superclass to the subclass and indicates a specialization in the subclass of the more general superclass.

Generalization Relationships (Cont'd)

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

BRAC
UNIVERSITY
Inspiring Excellence

Generalization Relationship

Generalization Relationship


```
                    ┌──────────────────────────┐
                    │ Employee                 │
                    ├──────────────────────────┤
                    │ hireDate                 │
                    ├──────────────────────────┤
                    │ receivePay               │
                    │ performWork              │
                    └──────────────────────────┘
                                 △
                    ┌────────────┴────────────┐
    ┌───────────────────────────┐   ┌───────────────────────────┐
    │  Manager                  │   │ Engineer                  │
    ├───────────────────────────┤   ├───────────────────────────┤
    │ department                │   │ certifications            │
    │ bonus                     │   │                           │
    ├───────────────────────────┤   ├───────────────────────────┤
    │ hireEmployee              │   │ analyze                   │
    │ promoteEmployee           │   │ design                    │
    └───────────────────────────┘   └───────────────────────────┘
```
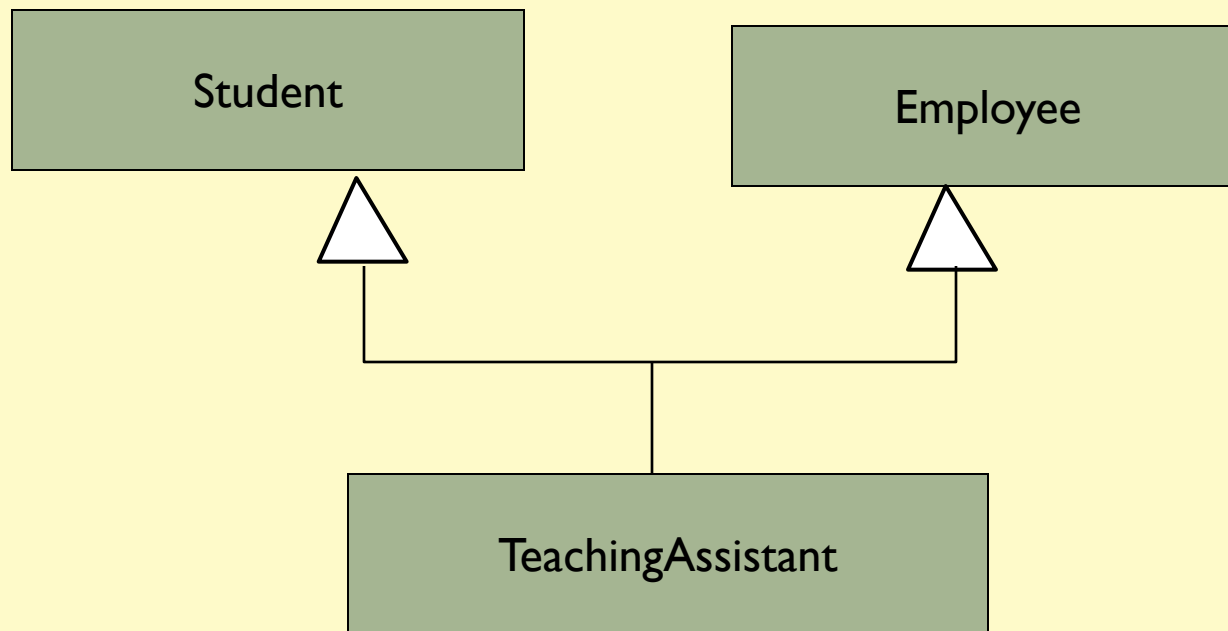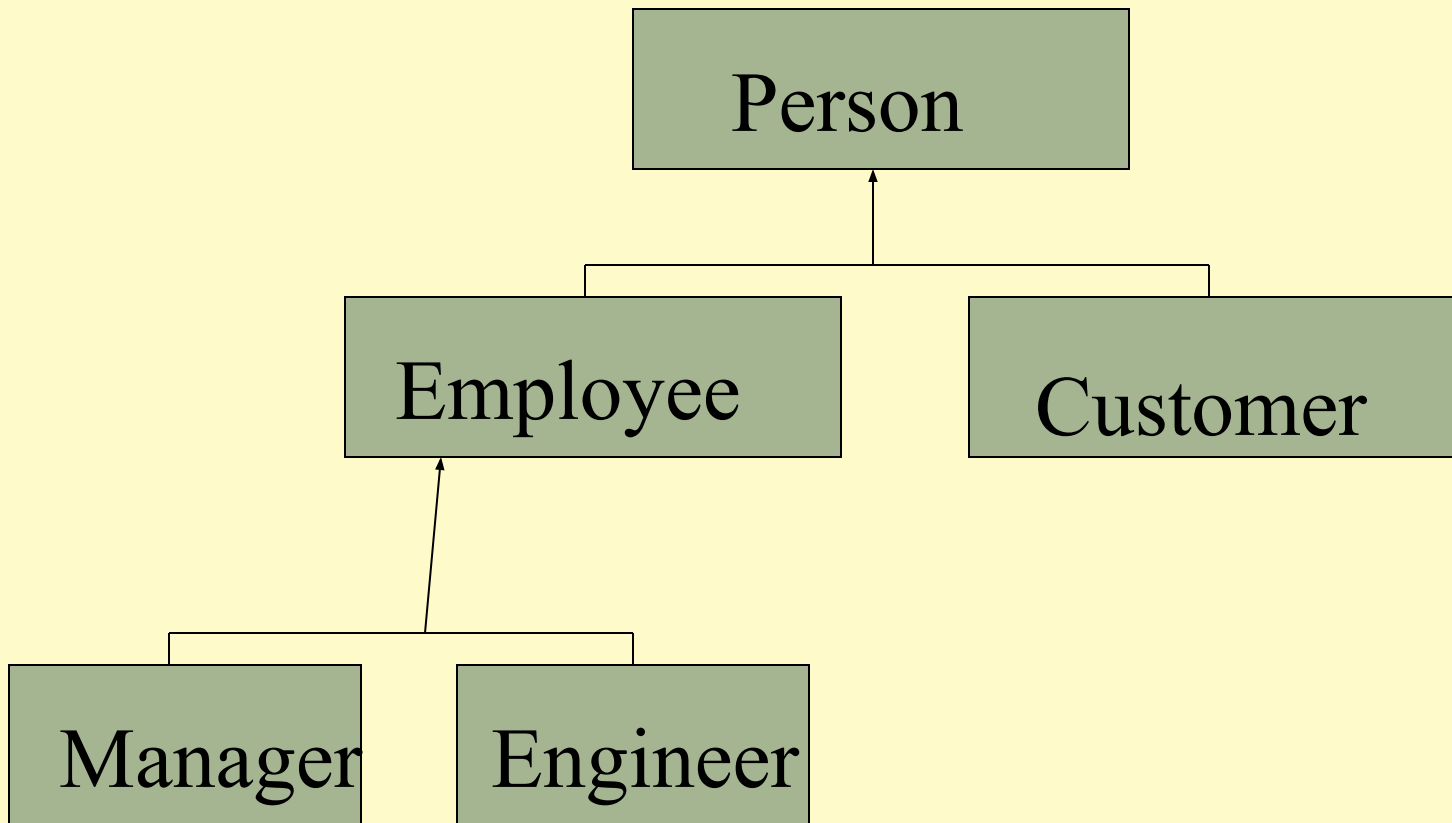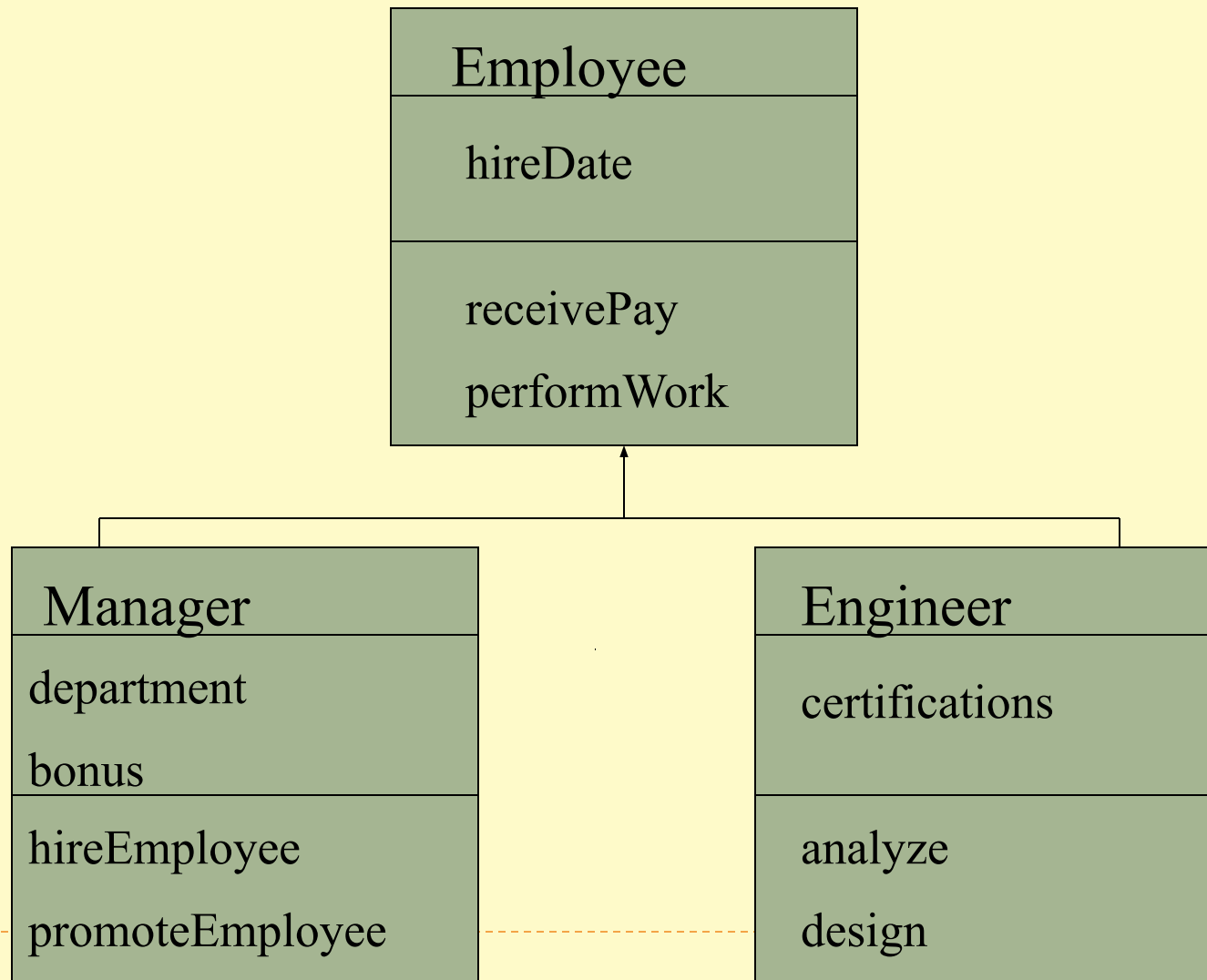
Multiplicity

- Documents how many instances of a class can be associated with one instance of another class

```
┌─────────────┐ 1   provides →  0..1 ┌──────────────────┐
│  Patient    │◆─────────────────────│ Medical History  │
└─────────────┘                      └──────────────────┘
```
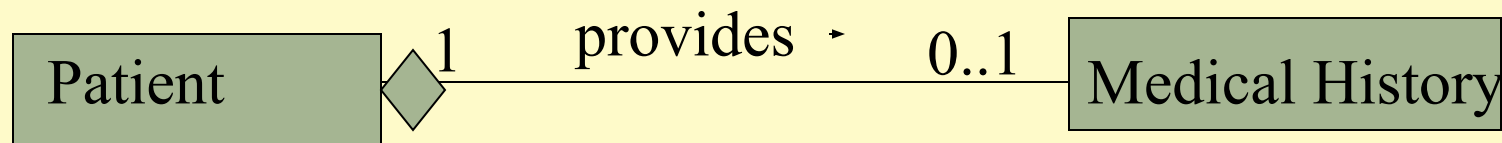
Multiplicity

- 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