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

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.

Selection of Analysis Classes from Potential Classes

1. ***Retained information.*** The potential class will be useful during analysis only if information about it must be remembered so that the system can function.
2. ***Needed services.*** The potential class must have a set of identifiable operations that can change the value of its attributes in some way.
3. ***Multiple attributes.*** During requirement analysis, the focus should be on “major” information; a class with a single attribute may, in fact, be useful during design, but is probably better represented as an attribute of another class during the analysis activity.
4. ***Common attributes.*** A set of attributes can be defined for the potential class and these attributes apply to all instances of the class.
5. ***Common operations.*** A set of operations can be defined for the potential class and these operations apply to all instances of the class.
6. ***Essential requirements.*** External entities that appear in the problem space and produce or consume information essential to the operation of any solution for the system will almost always be defined as classes in the requirements model.

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

2 Kinds of Classes During Analysis

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

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

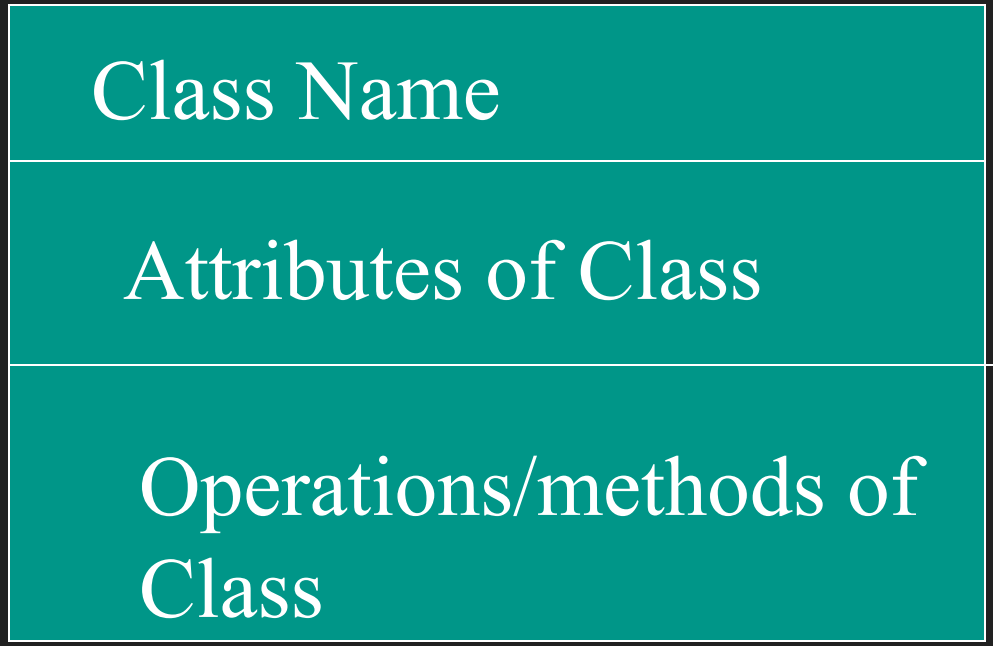
Attributes in a Class

- 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

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

UML Representation of Class



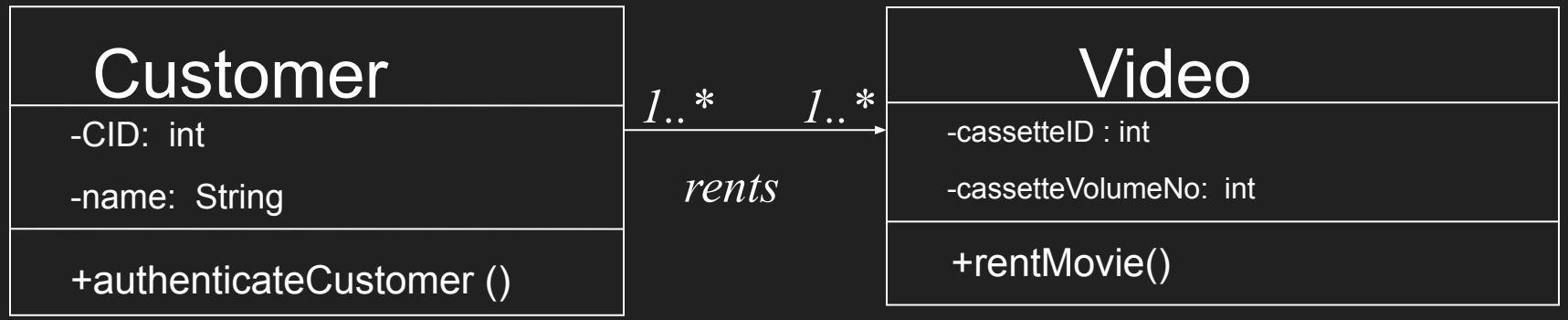
Example of a Class Diagram

Video Rental System

visibility

multiplicity

class name



attributes

relationship

methods

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.

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

Association 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

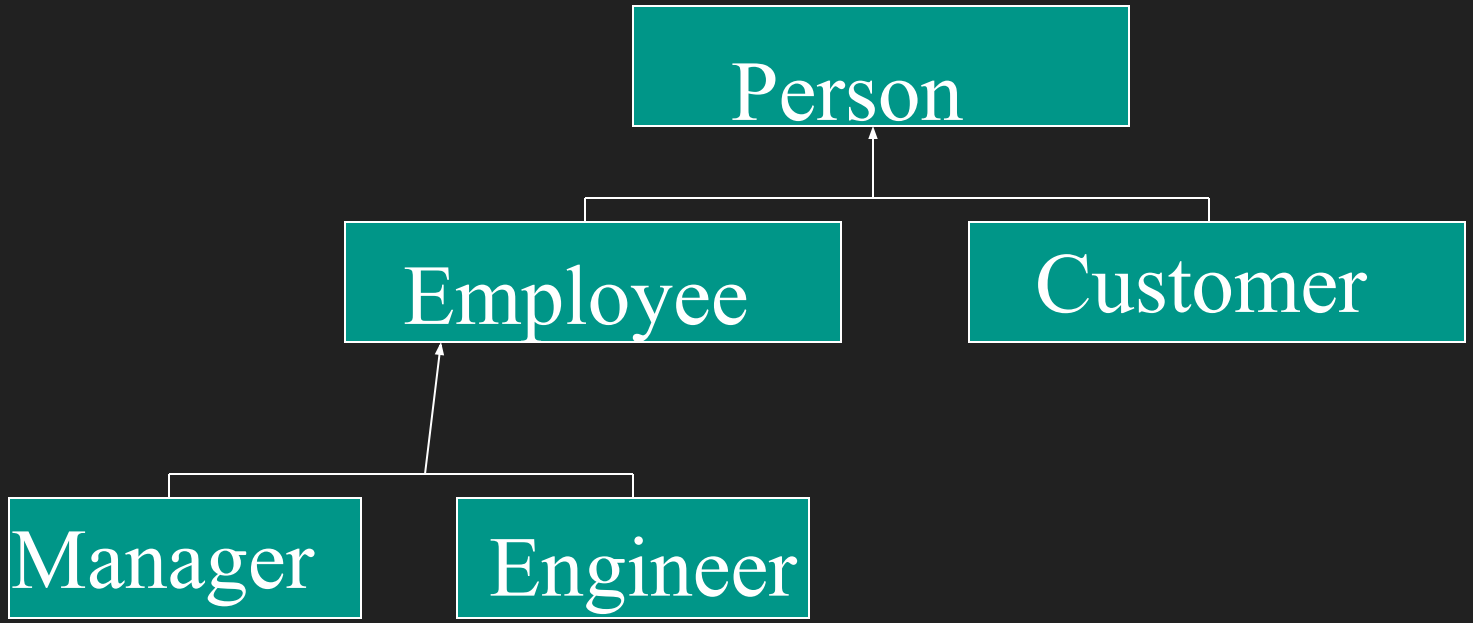
Association Relationship

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

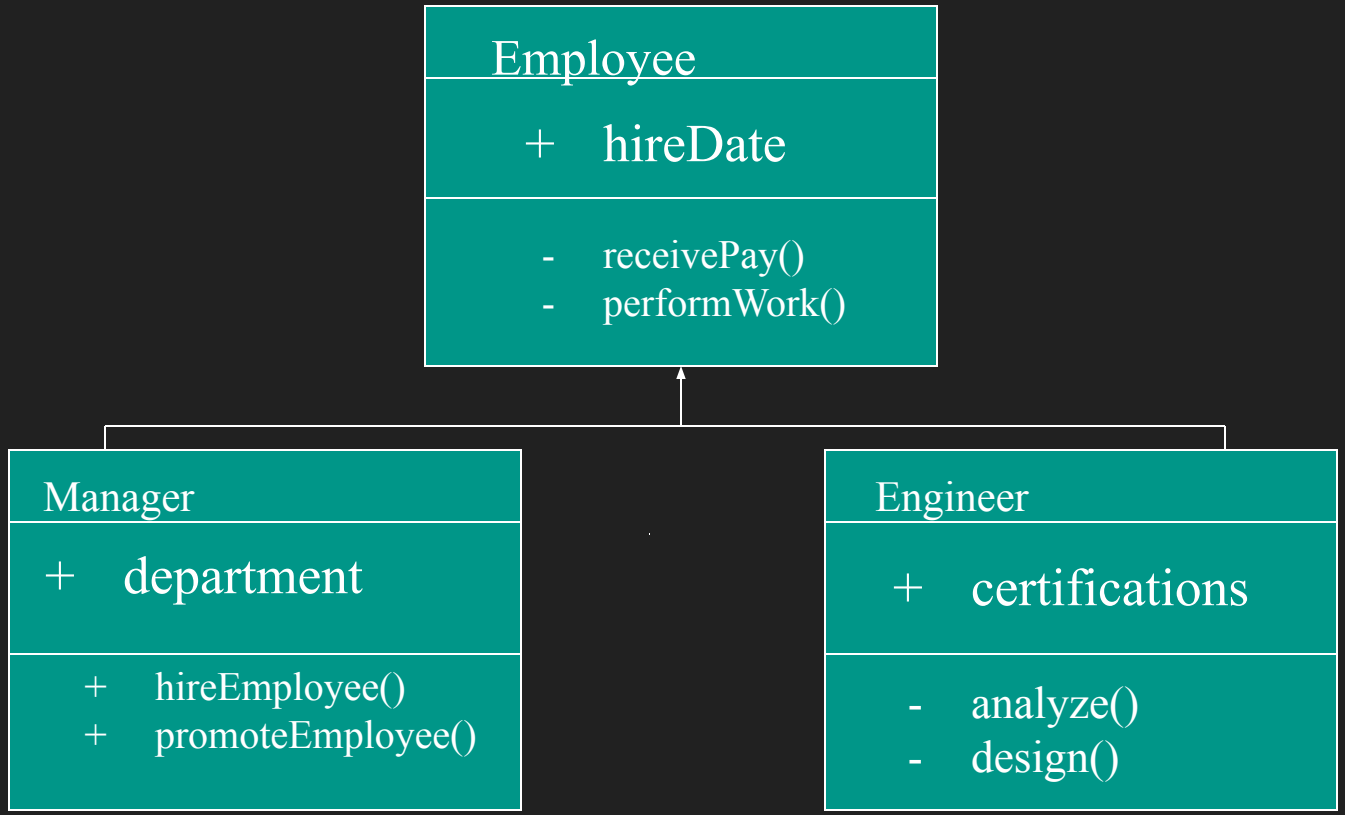
Generalization Relationship

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

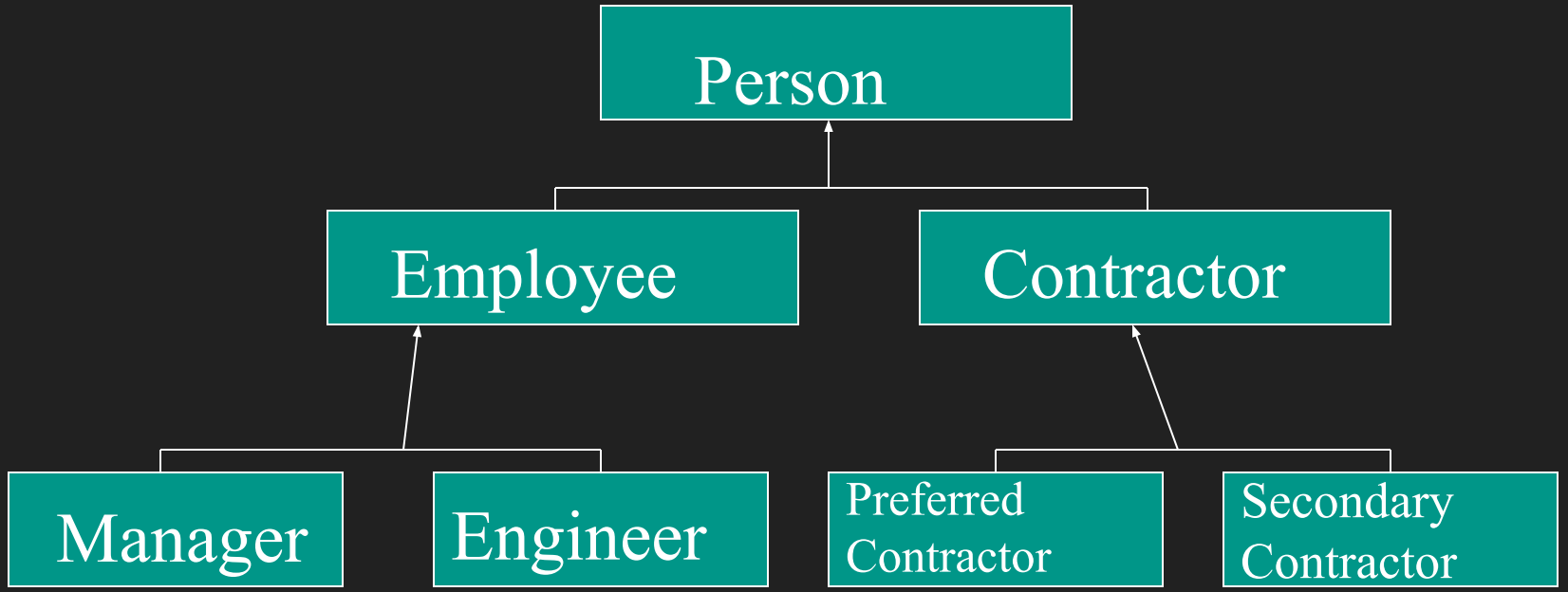
Generalization Relationship



Generalization Relationship



Generalization Relationship

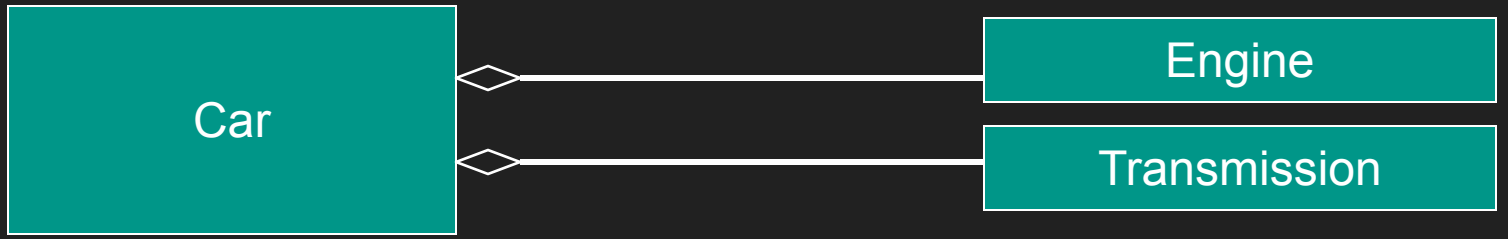


Aggregation Relationship

- Specialized form of association in which a whole is related to its part(s)
- Represented by *a-part-of* relationship

Aggregation Relationship

- Denoted by placing a diamond nearest the class representing the aggregation



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

Multiplicity

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

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		

Guidelines for Analyzing Use Cases

- A common or improper noun implies a class of objects
- A proper noun implies an instance of a class
- A collective noun implies a class of objects made up of groups of instances of another class

Guidelines for Analyzing Use Cases (2)

- An adjective implies an attribute of an object
- A doing verb implies an operation
- A being verb implies a relationship between an object and its class
- A having verb implies an aggregation or association relationship

Guidelines for Analyzing Use Cases (3)

- A transitive verb implies an operation
- An intransitive verb implies an exception
- A predicate or descriptive verb phrase implies an operation
- An adverb implies an attribute of a relationship or an operation

Class Diagram

- Ensure that the classes are both necessary and sufficient to solve the underlying problem
  - no missing attributes or methods in each class
  - no extra or unused attributes or methods in each class
  - no missing or extra classes

Discarding Unnecessary and Incorrect Classes

- Redundant Classes: Some potential classes differ only in name.
- Irrelevant Classes: Classes that have nothing to do with the system. Example: *computer connection*
- Vague Classes: Classes whose meaning is not clear at all. Examples: *system* and *software*

Discarding Unnecessary and Incorrect Classes

- Attributes: Some nouns in the list above are likely to be modeled as attributes rather as classes. Examples: *author*, *title*
- Operations: Some nouns are likely to be operations rather than classes. Example: *book search*.
- Roles: Some nouns are roles of objects involved in associations rather than classes.
- Implementation Constructs: Anything that is not part of the real-world problem.