UML Essentials
Static Modeling

Excerpts from: *Object Oriented Software Engineering*
by Lethbridge/Laganière
and
Applying UML and Patterns
by Larman, C.

Class model (diagram) elements

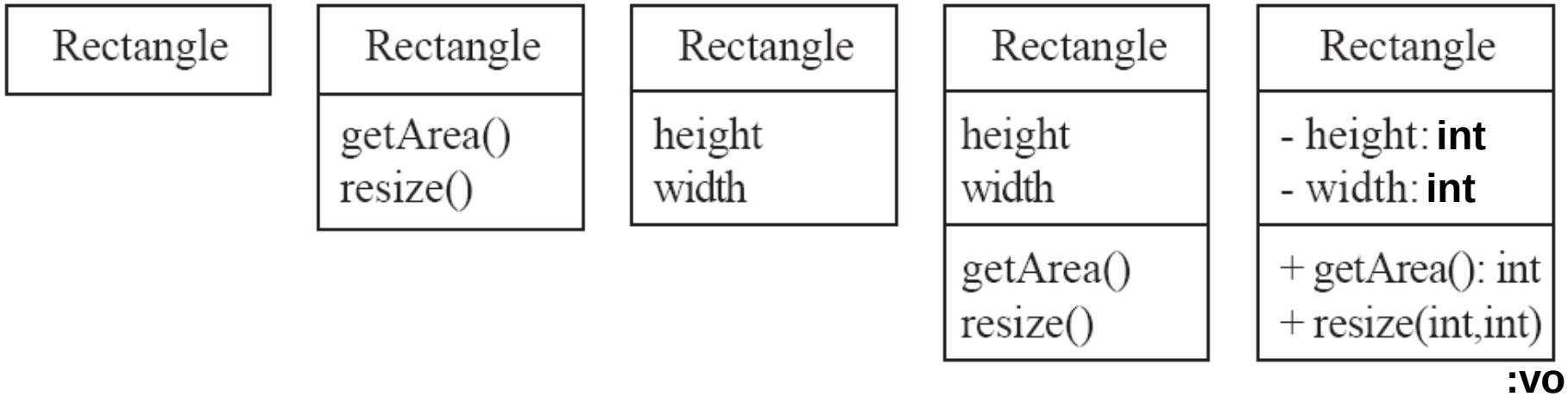
- *Classes*
  - represent the types of data themselves
- *Associations*
  - represent linkages between instances of classes
- *Attributes*
  - are simple data found in classes and their instances
- *Operations*
  - represent the functions performed by the classes and their instances
- *Generalizations*
  - group classes into inheritance hierarchies

Classes

•A class is simply represented as a box with the name of the class inside

- The diagram may also show the attributes and operations
- The complete signature of an operation is:

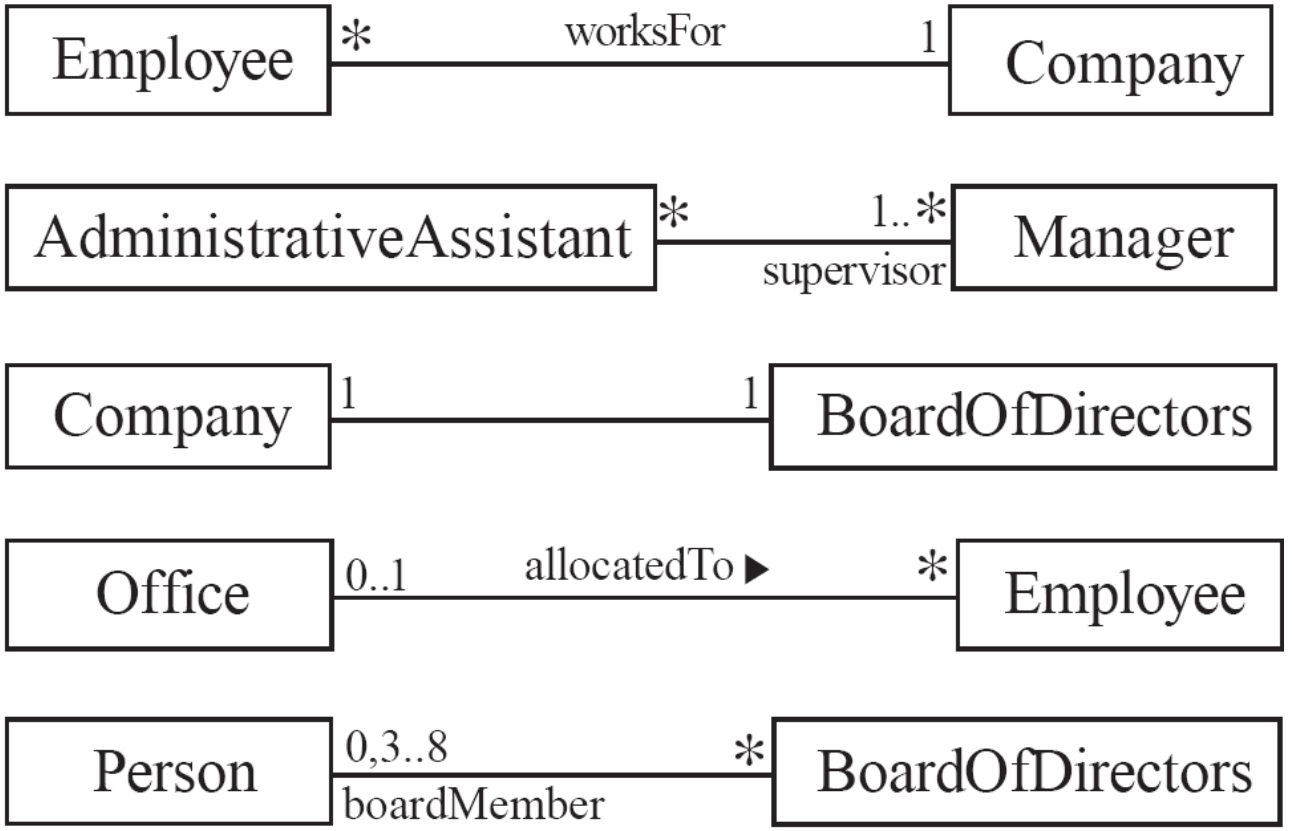
  operationName(parameterName: parameterType …): returnType



Associations and Multiplicity

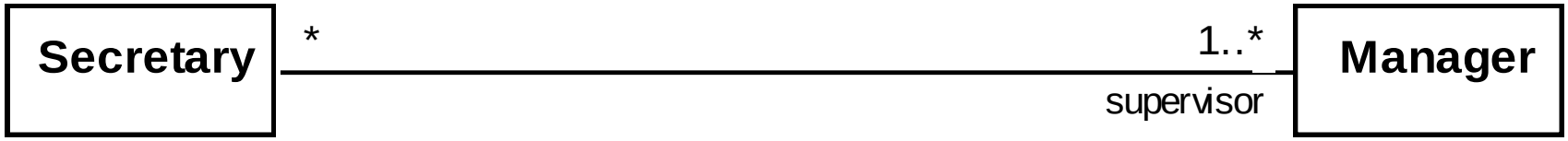
An *association* is used to show how two classes are related to each other

- Symbols indicating *multiplicity* are shown at each end of the association
- Each association can be labelled, to make explicit the nature of the association



Analyzing and validating associations

- **Many-to-many**
  - A secretary can work for many managers
  - A manager can have many secretaries
  - Secretaries can work in pools
  - Managers can have a group of secretaries
  - Some managers might have zero secretaries.
  - Is it possible for a secretary to have, perhaps temporarily, zero managers?

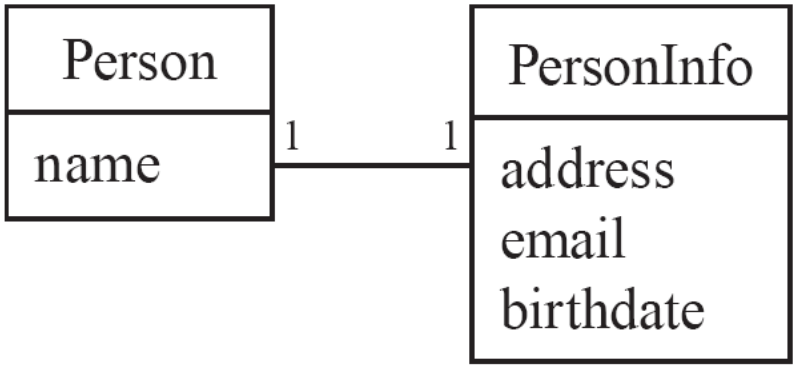


Analyzing and validating associations

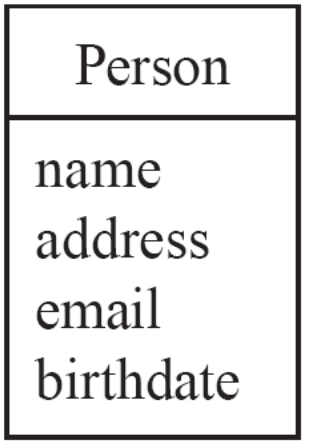
•Avoid unnecessary one-to-one associations

•

Avoid this



do this

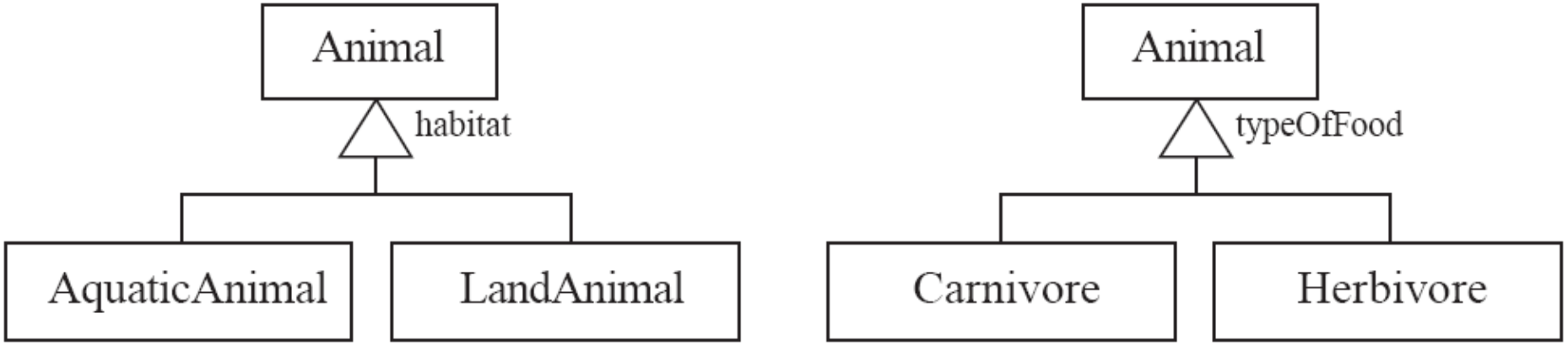


Directionality in associations

- Associations are by default are undefined, though many tools treat these as *bi-directional*.
- It is possible to limit the direction of an association by adding an arrow at one end

Generalization

- Specializing a superclass into two or more subclasses
  - The *discriminator* is a label that describes the criteria used in the specialization



Associations versus generalizations in object diagrams

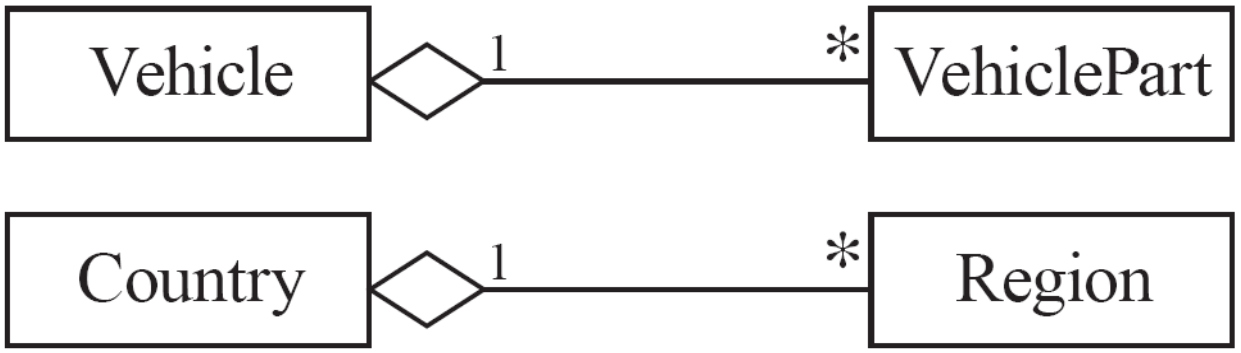
- Associations describe the relationships that will exist between *instances* at **run time**.
  - When you show an instance diagram generated from a class diagram, there will be an instance of *both* classes joined by an association
- Generalizations describe relationships between *classes* in class diagrams.
  - They do not appear in instance diagrams at all.
  - An instance of any class should also be considered to be an **instance** of each of that class's **superclasses**

More Advanced Features: Aggregation

- Aggregations are special associations that represent ‘part-whole’ relationships.
  - The ‘whole’ side is often called the *assembly* or the *aggregate*
  - This symbol is a shorthand notation association named `isPartOf`

•As a general rule, you can mark an association as an aggregation if the following are true:

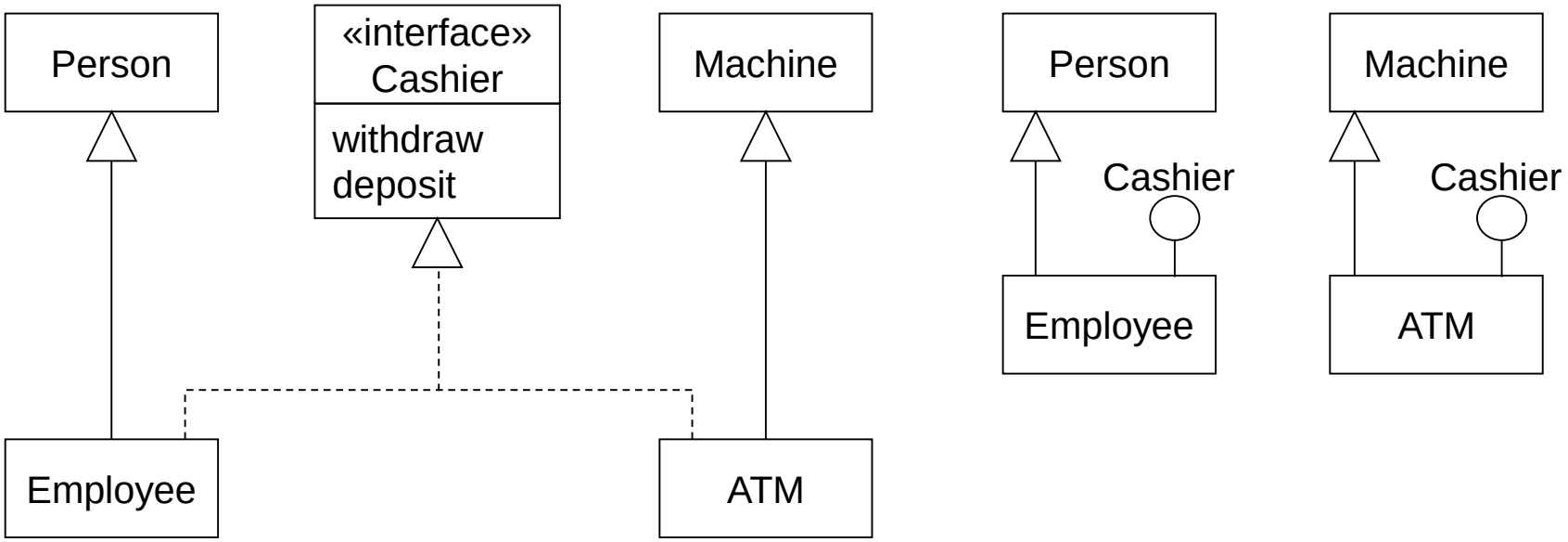
- You can state that
  - the parts ‘are part of’ the aggregate
  - or the aggregate ‘is composed of’ the parts
- When something **owns** or controls the aggregate, then they also own or control the parts



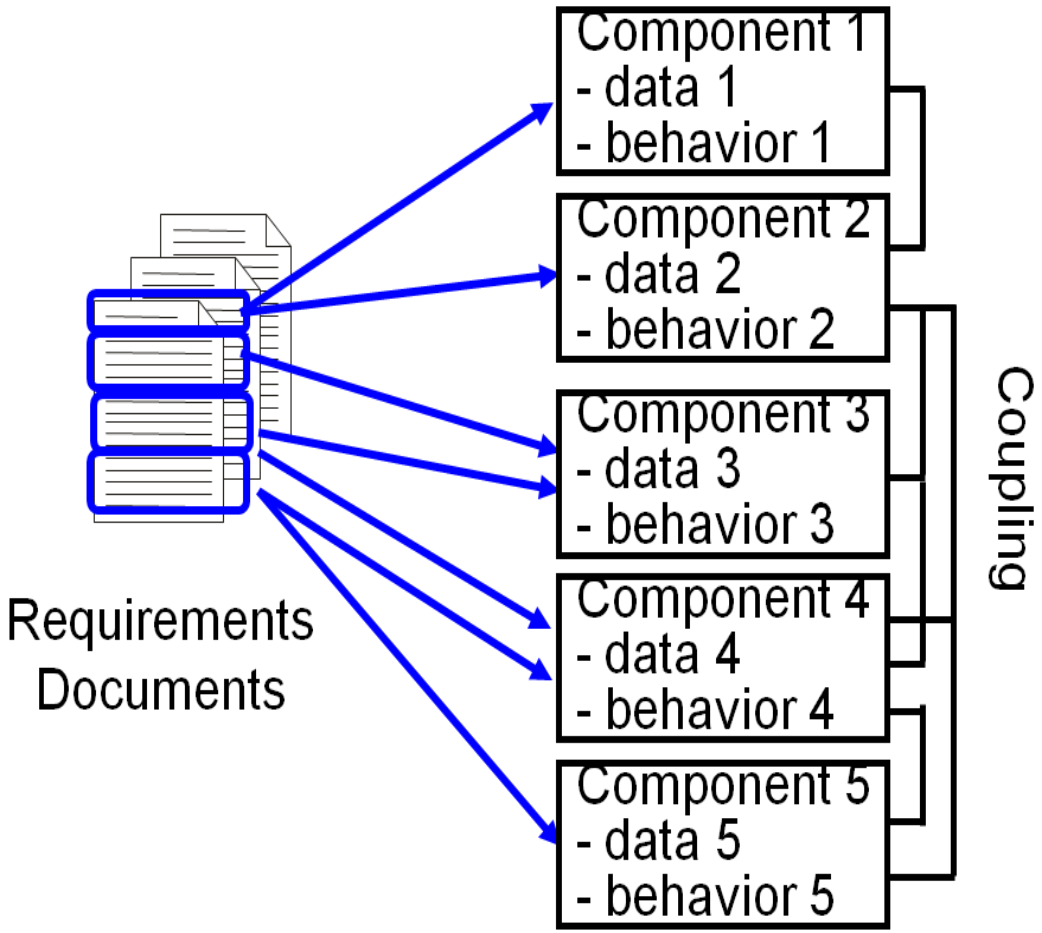
Interfaces

•An interface describes a *portion of the visible behaviour* of a set of objects.

- An *interface* is similar to a class, except it lacks instance variables and implemented methods



Mapping Requirements to Design Components



- Design must satisfy requirements
  - Everything (data and behavior) in the requirements must be mapped to the design components
  - Decide what functionality goes into which component
- As you do the mapping, assess functional cohesion and coupling
  - Strive for **low** coupling and **high** cohesion