Object Model

Object Model

- Captures static structure of system
- Objects, relationships, attributes & operations
- Most important
- Intuitive graphic representation
- Valuable for communication & documentation

Objects

- Decomposing problem into objects
    - depends on judgement & nature of problem
- No one correct representation
- Objects have identity

Class

- Often appear as *nouns* in problem descriptions
- Has semantic
- Interpretation of semantics
    - depends on application and matter of judgement
- Each class may have zero, one or more objects
- Each object knows it class

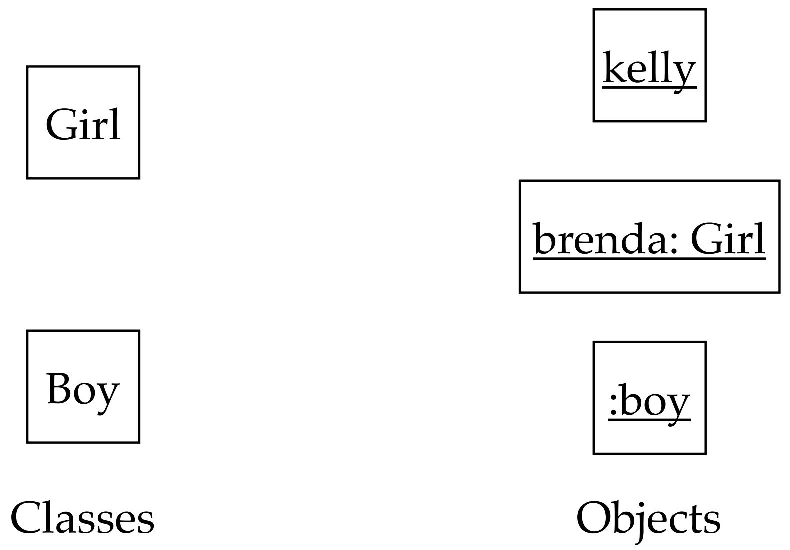
Class Diagrams

- Provide formal graphic notation for modeling
- Concise, easy to understand, practical
- Describes many possible instances

Object Diagrams

- Describes how set of objects relate
- Useful for
  - documenting test cases
  - Clarification of complex class diagrams
- Class Diagram corresponds to infinite set of object diagrams

Notation: Class, Objects



Attributes

- Data value held by objects of a class
- Objects may have same/different values for attribute
- Attribute name unique within a class
- Adjectives often represent specific enumerated attribute values : "red car"

Attributes...

- Attribute is a pure data value - not an object
- Internal identifiers must not be shown as attributes
- *Show only important attributes*

Derived Attributes

Base Attribute :

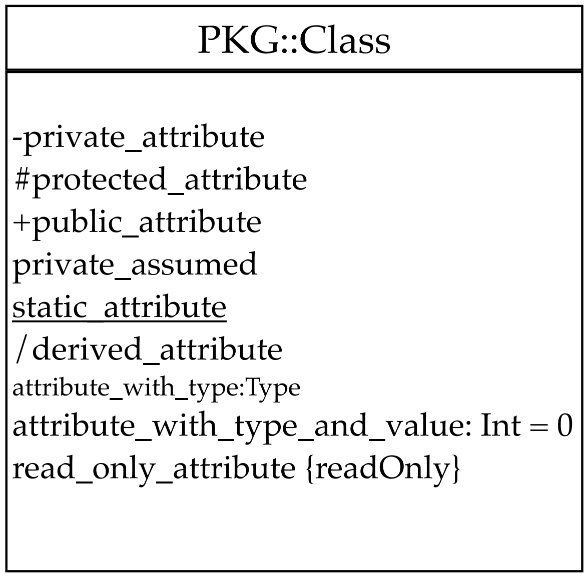
- primitive, not dependent on other attributes

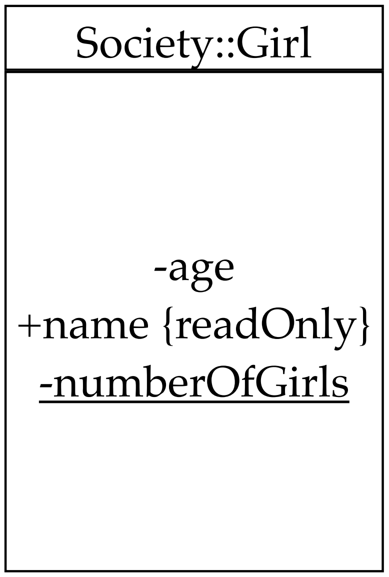
Derived Attribute :

- computing not considered to change state of an object
- dependent on base attributes
- may be stored or computed upon a query operation

Example : Area of a circle, age of a person

Attributes Notation



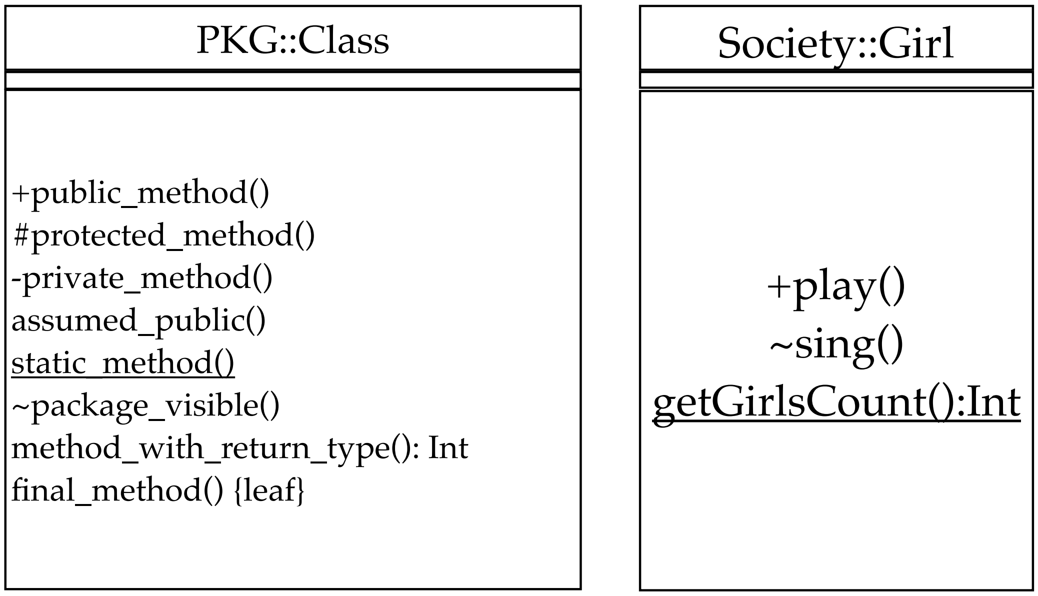


Operations & Methods

- Operation : Function that may be applied to or by objects
- Same Operations applying to different classes: Polymorphic
- Method is implementation of an operation for a class
- Operation has a target object and may have arguments
- Same operations on different classes should have
  - same signature and consistent intent
- Query Operation : Does not affect the state of object

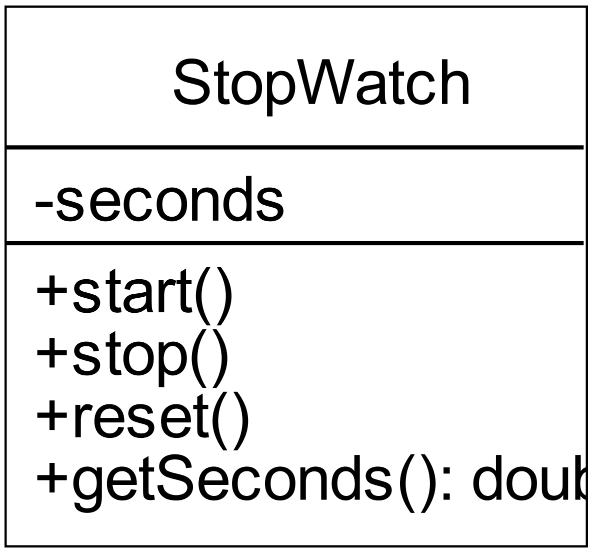
- *Show only important methods*

Operation Notation



MDL- 13

Example



MDL- 14

Associations and Links

- Link is physical or conceptual connection between objects
- Link is an instance of an Association

Example:

Link : Susan is-wife-of Robert

Julie is-wife-of John

Association: Woman is-wife-of Man

Associations and Links appear as *Verbs*

Associations & Links...

- Associations are bi-directional
  - may be implemented as unidirectional
- Implemented usually as pointers
  - important ***not*** to think as pointers
- Associations may be
  - one-to-one
  - one-to-many
  - many-to-one
  - many-to-many

Ternary & High Order Associations

Binary Association : Relates two classes

Woman is-wife-of Man

Ternary Association : Relates three classes

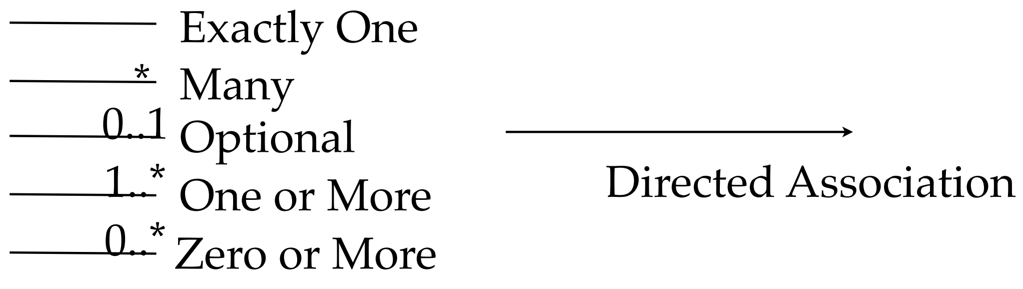
Nancy is-daughter-of Susan and Robert

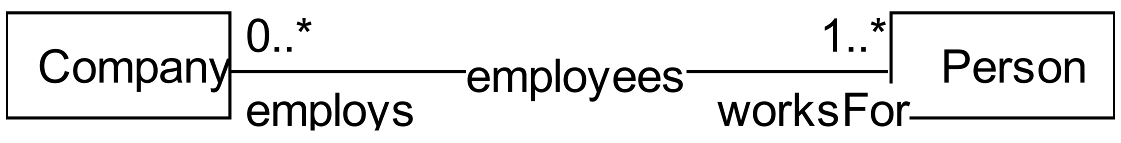
n-ary Association : Relates *n* classes

Higher Order Associations

- complicated to draw, implement and think
- try to avoid if possible

Associations Notations



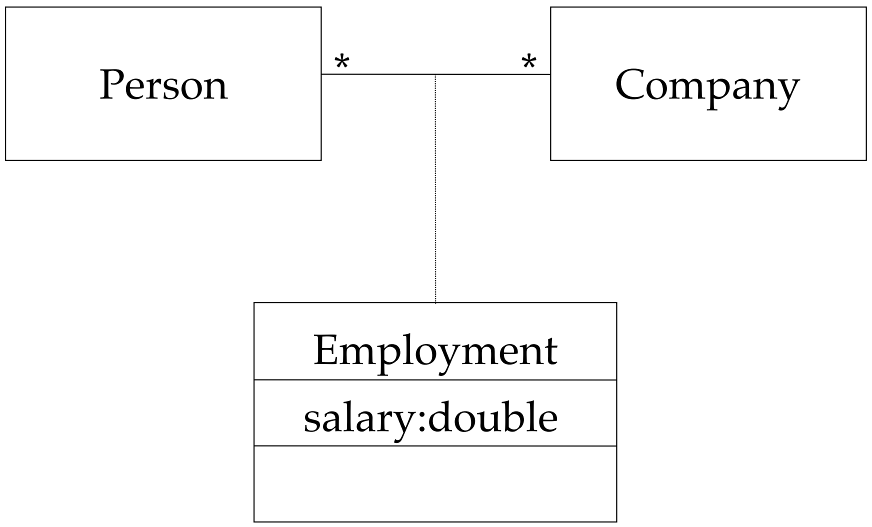


Link Attributes & Association Classes

- Attributes that belong to association of object rather than one object

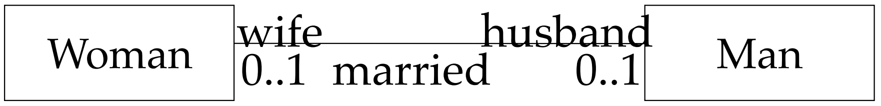
- Link Attributes belong in Association Classes
- Ex: Salary received by Employee from Company
- In an one-to-one association you may try to make it attribute of one of the objects
  - Leads to extensibility problems

Association Class Notation

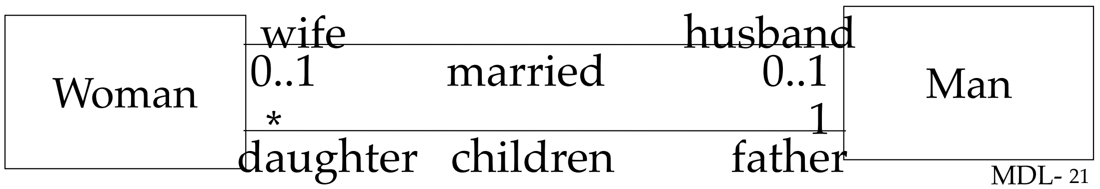


Role Names

- Name given to either end of an association
- Helps to navigate from one object to related objects

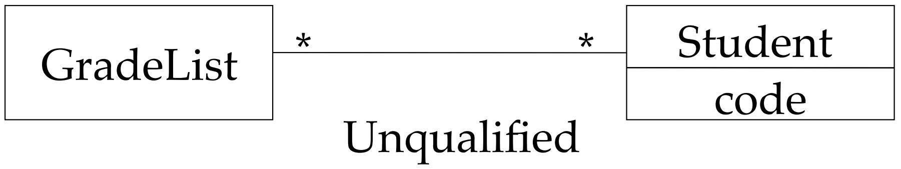


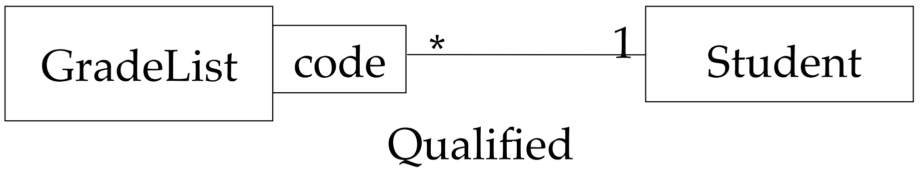
- Helps clarify when two classes have several associations between them



Qualifiers

- Distinguishes among set of associated objects
- Models associative arrays, dictionaries
- **Qualifiers may be wrongly modeled as attribute of associated class**





Aggregation & Composition

Aggregation:

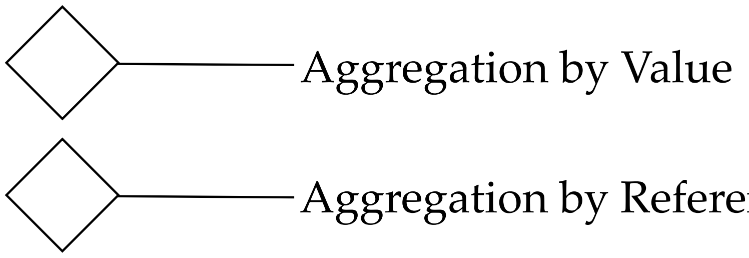
- Part-of or part-whole relationship (by reference)
- Example : Car has Engine and Transmission
- Assembly of objects with aggregate and component parts
- Component existence may or may not depend on aggregate

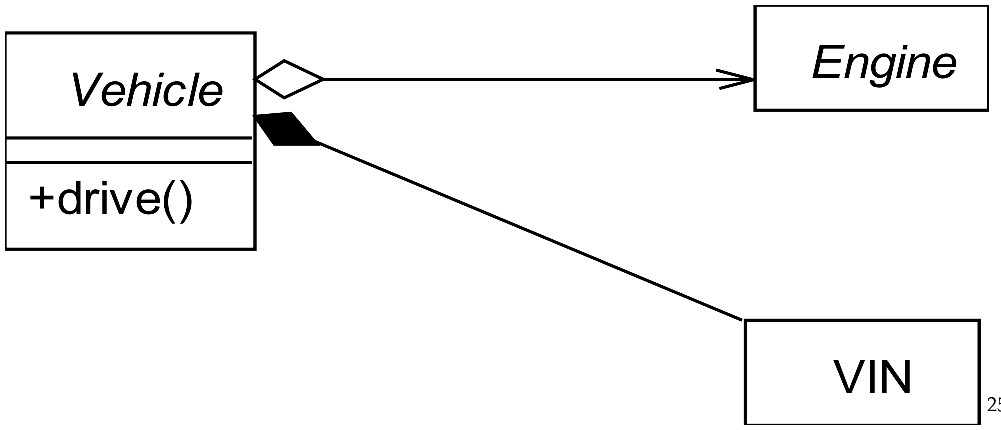
Aggregation & Composition...

Composition:

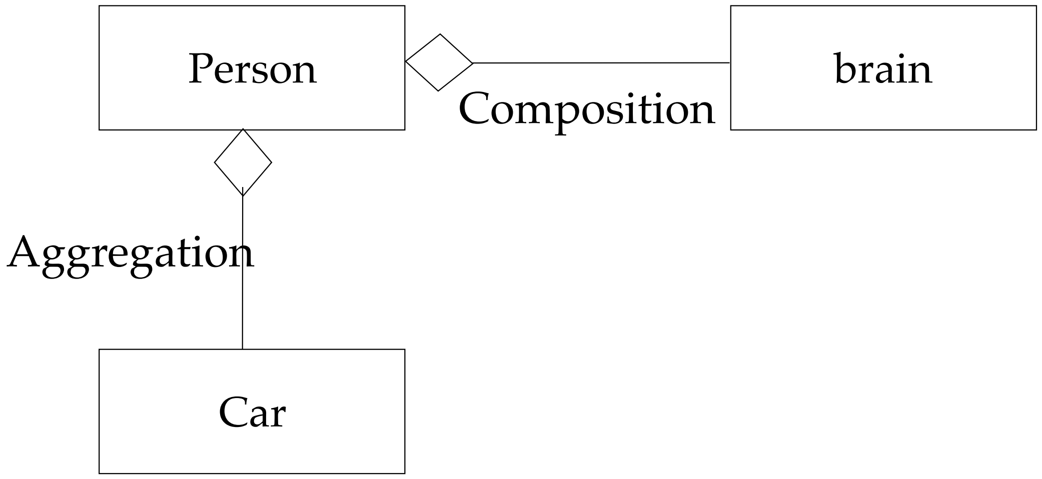
- Part belongs to only one whole (by Value)
- Part lives and dies with the whole
- Whole cannot replace the part

Aggregation/Composition Notation





Aggregation / Composition Example

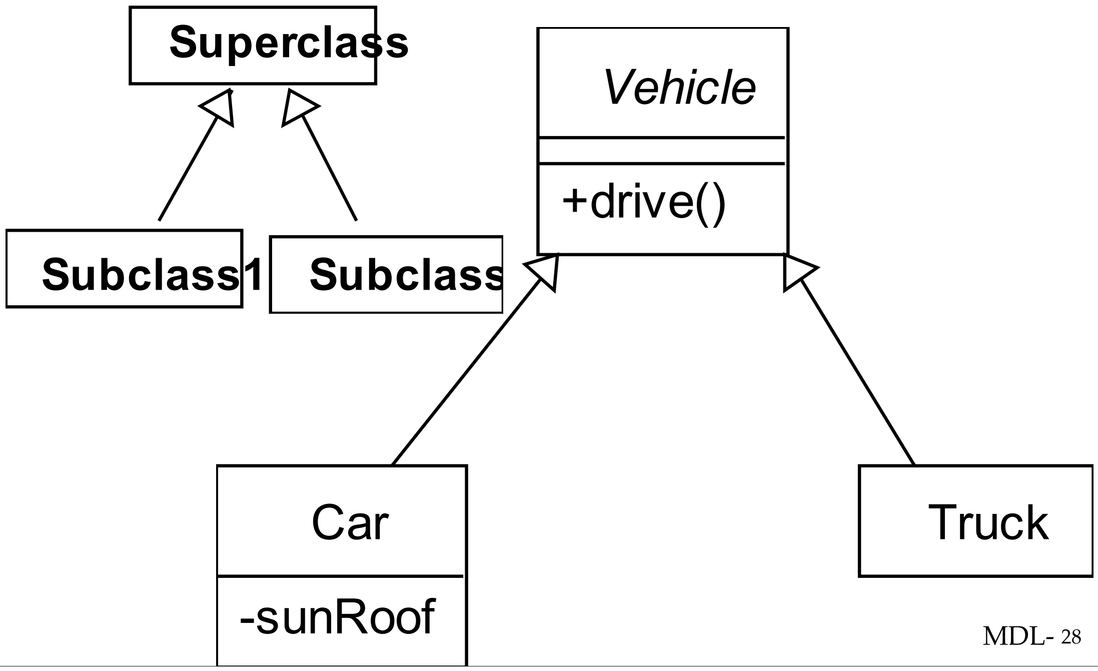


Inheritance

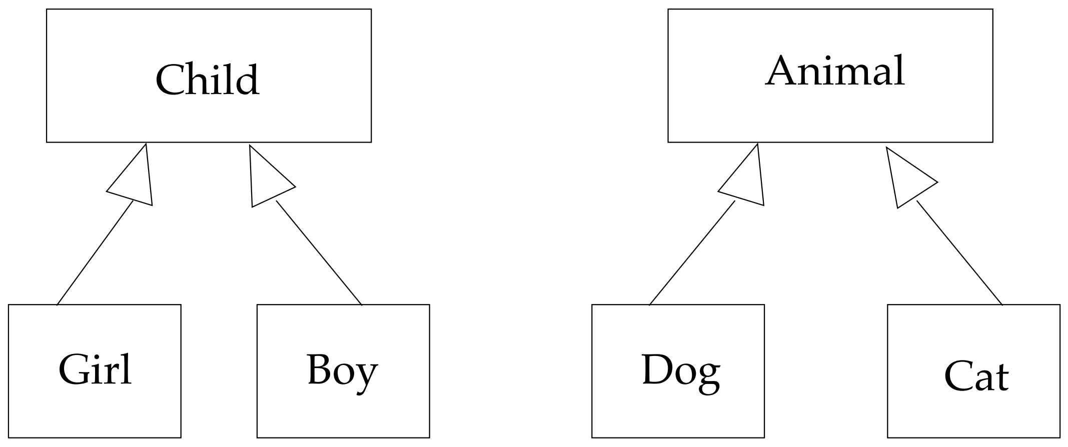
- Models *is-a* relationship
- Relationship between a class and its refined versions
- Superclass or Base class
- Subclass or Derived class
- Inheritance is transitive
- Discriminator : The property being abstracted by a particular inheritance
- Breath Vs. Depth of inheritance

Inheritance Notation

Generalization



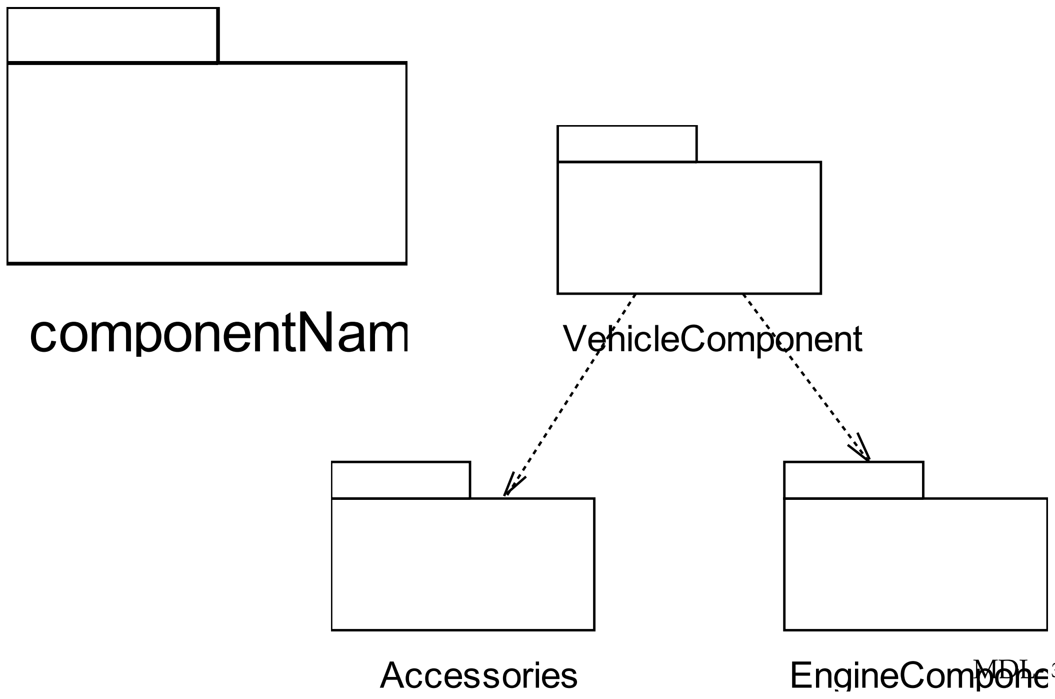
Inheritance Example



Grouping Mechanism : Package

- Grouping classes together into higher-level units
- Package diagram with dependency
- Dependency between packages exists if
  - class in one package depends on a class in the other
  - definition change of one package may change other

Package Notation



Aggregation Vs. Association

- Special form of Association
- May be Confusing
- Aggregation represents “part-of” relationship
- Some operations on whole automatically applied to its parts
- Aggregate is asymmetric : part is subordinate to the whole
- Association is symmetric : objects involved are of equal stature

Aggregation Vs. Inheritance

- Aggregation represents part-of relationship
- Inheritance represents kind-of relationship
- Aggregation refers to object relationships
- Inheritance refers to class relationships

Fixed, Variable & Recursive Aggregates

Fixed :

- Fixed structure
- Number & types of parts pre-defined

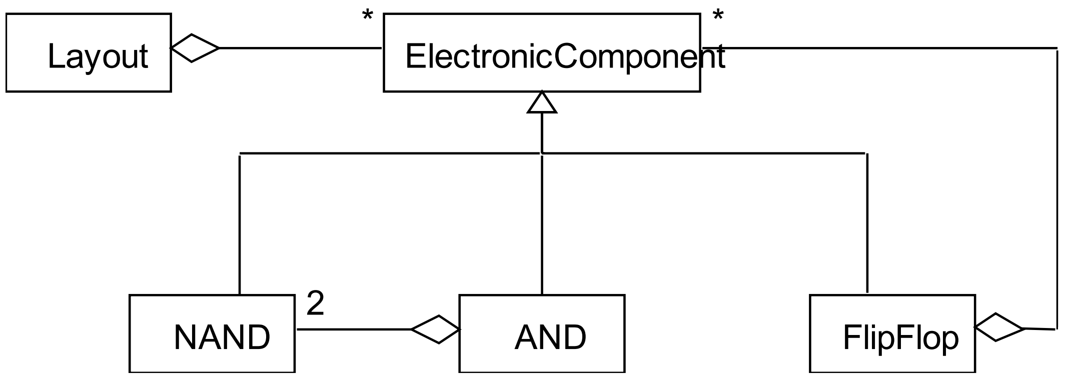
Variable :

- Finite number of levels - Number of parts vary

Recursive :

- Contains instances of the same kind of aggregate
- number of potential levels unlimited

Example : Fixed, Variable, Recursive Aggregation



Operations & Aggregation

- Operation or Triggering :
  - automatic application of an operation to network of objects when applied to some starting object
- Shallow Copy vs. Deep Copy

Abstract Classes

Representing an Abstraction that is Abstract.

- Abstract classes represent
  - concepts
  - not real objects
- ABCs used only to create other “Concrete” classes

Abstract Classes ...

Example: Shape, Employee, Animal

Whether a class in Abstract or not depends on

- judgement
- application on hand

Inheritance : Extension & Restriction

Extension :

- Subclass adds new features
- Subclass inherits all properties & operations of ancestor

Restriction :

- Subclass constrains ancestor attributes
- Subclass may not inherit all properties & operations of its ancestor
- Often leads to trouble (Liskov Substitutability Principle)

Inheritance : Extension & Restriction ...

- Proper Extension:
  - A Subclass may override the internal implementation of an operation
  - No problem as long as external protocol remains the same