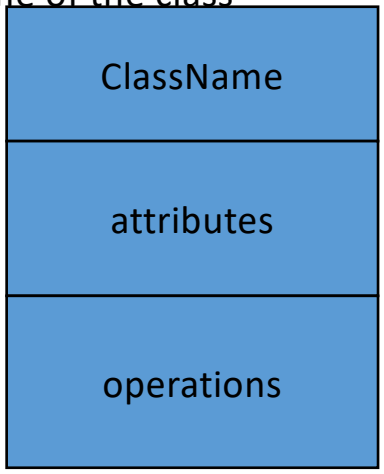
UML

Class Diagram

UML Class Diagram

- Model the structure of a domain
- Represent the entities
  - Fields and methods
  - Each entity is represented by a rectangle with three sections:
    - Name
      - Can include the name of the package
      - For abstract classes, the name is in *italic,* or use **<<abstract>>** before name of the class
      - The name is the only section that is mandatory
    - Fields or attributes
    - Methods or operations
- And their relationships
  - Inheritance
  - Association

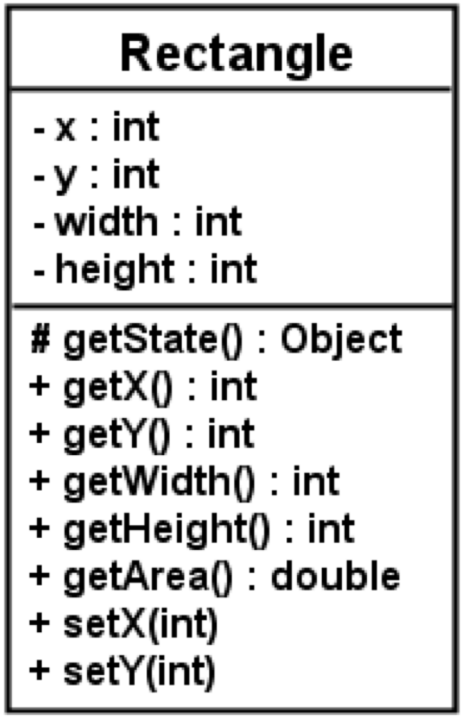


Class Diagram: Field Declaration

- Each field is represented as a line in the second section
- Sintax:
  [***visibility]*** **[/]** ***name*** **:** **[*type]***
- ***visibility*** is one of
  - **-** for private
  - **+** for public
  - **#** for protected
  - **~** for package-private
- ***/*** means derived attribute
- Static fields are underlined

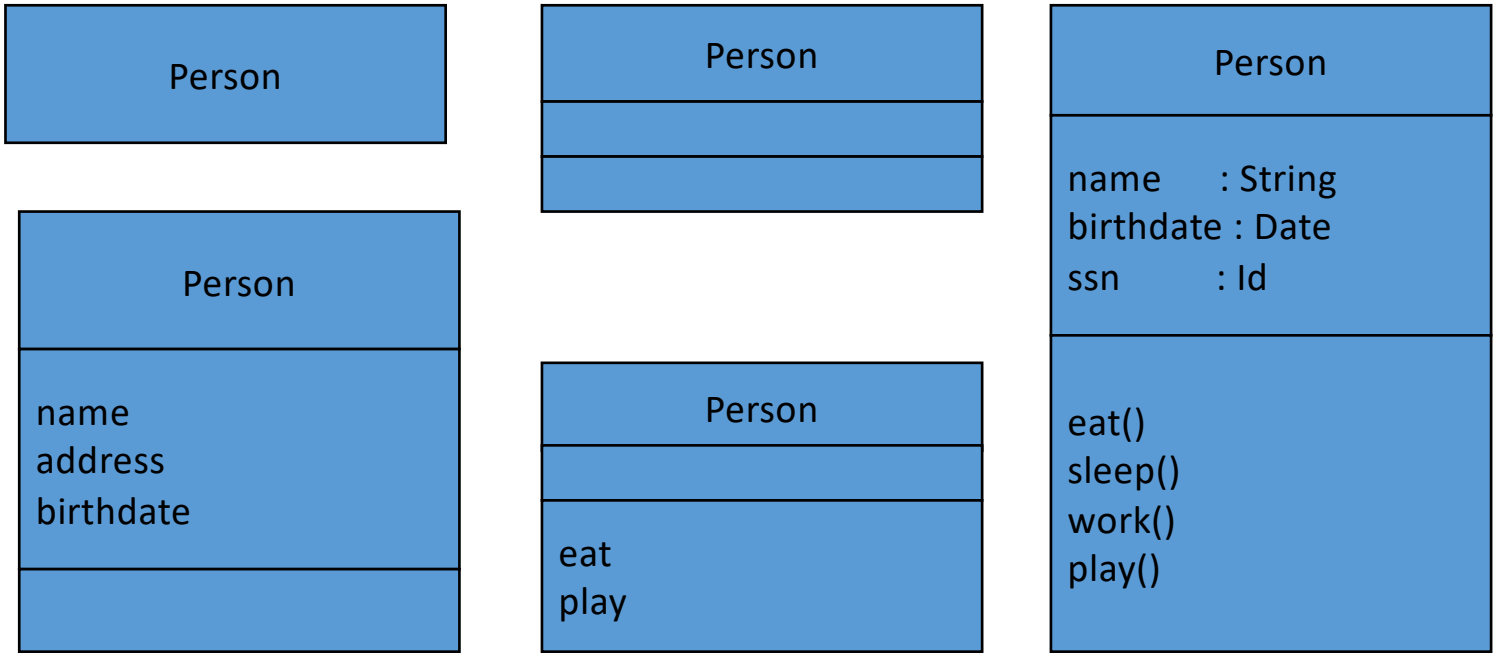
Class Diagram: Method Declaration

- methods are written as
  [visibility] name [(parameter-list)] [:return-type]
  - omit *:returnType* for constructors and methods that return `void`
  - ***[(parameter-list)*** -> (arg1: type1, arg2: type2, …)
  - Static methods are <u>underlined</u>
  - Abstract methods are in *italic* or start with **<<abstract>>**
  - Abstract classes have name in *italic* or start with **<<abstract>>**



Depicting Classes

- The same class can appear in several diagrams
  - Or several times in the same diagram
- Do not need to show all attributes and operations of a class in every diagram.

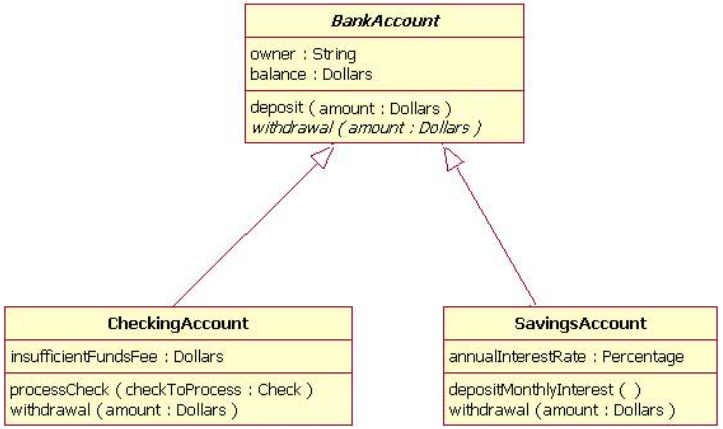


Relationships

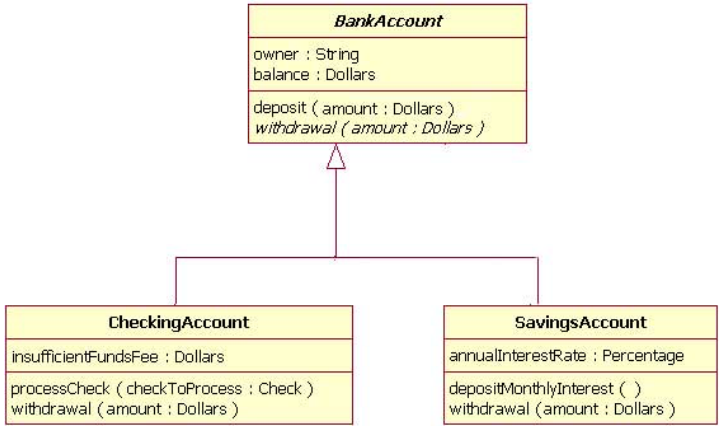
- There are three kinds of relationships in UML:
  - Dependencies
  - Generalizations
  - Associations

Generalization Relationship - Inheritance

- Hierarchies drawn top-down
  - With arrows from child to parent
  - The subtype points to the super type
  - Inheritance is represented as a solid line with a closed, unfilled arrowhead pointing at the superclass

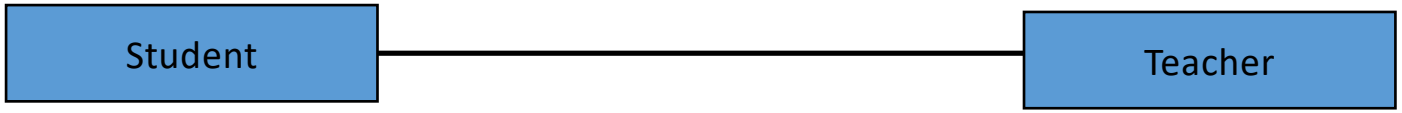


  - Interfaces
    - Name section starts with «interface»
    - Representing implementation is similar to inheritance but lines are dashed
    - Extension of interfaces represented as inheritance in classes

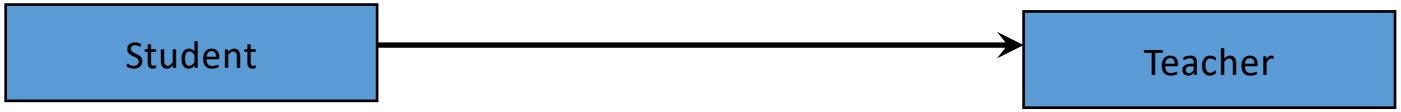


Association Relationship

- If two classes in a model need to communicate with each other, there must be link between them.
- Instances of one class are associated with instances of another class
- Represented as a solid line between the two entities
- Two types of *association*
- ***Bi-directional*** (default)
  - Both classes are aware of each other and their relationship

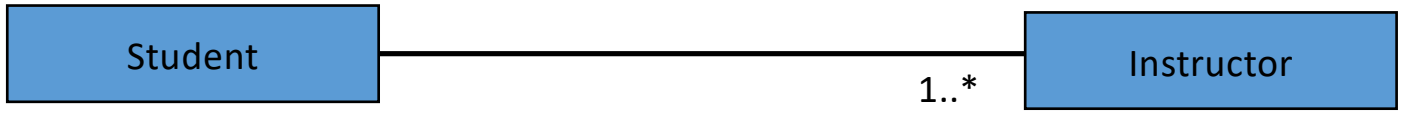


- Uni-directional
  - Both classes are related, but only one class knows that the relationship exists
  - Represented as a solid line with an open arrowhead

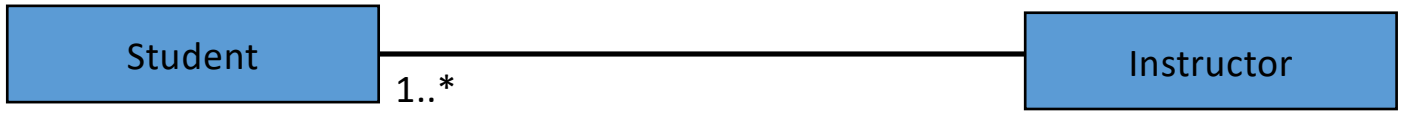


Association Relationships - Multiplicity

- We can indicate the *multiplicity* of an association by adding *multiplicity adornments* to the line denoting the association.
  - E.g., a *Student* has one or more *Instructors*:



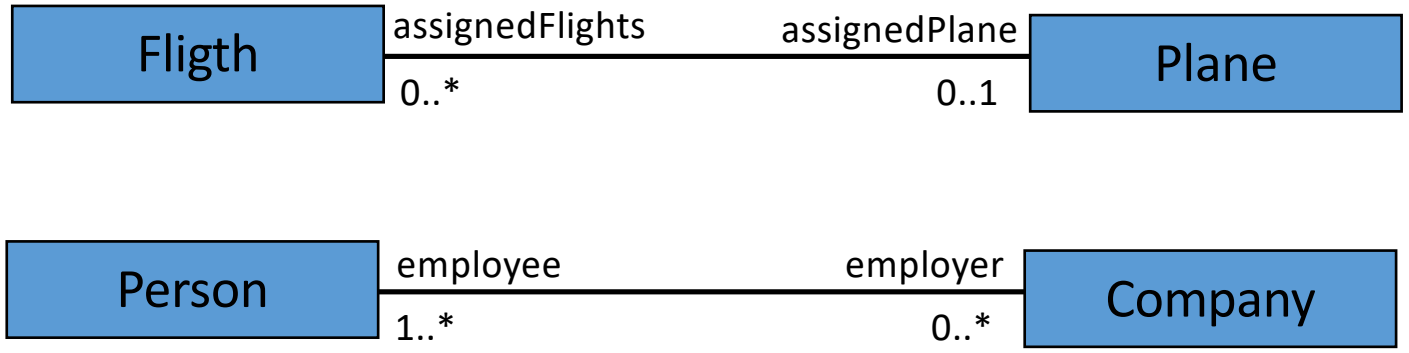
- To show that every *Instructor* has one or more *Students*:



- Potential Multiplicity Values
  - 0..1 – zero or one
  - 1 (default) – exactly one
  - 0..* - zero or more (or just *)
  - n – exactly n
  - n..p – between ***n*** and ***p***

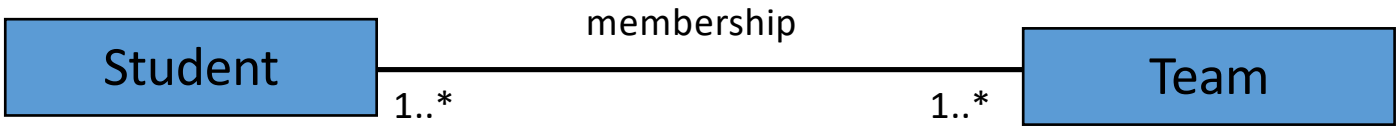
Association Relationship - Role

We can also indicate the behavior of an object in an association (*i.e.,* the *role* of an object) using *rolenames*



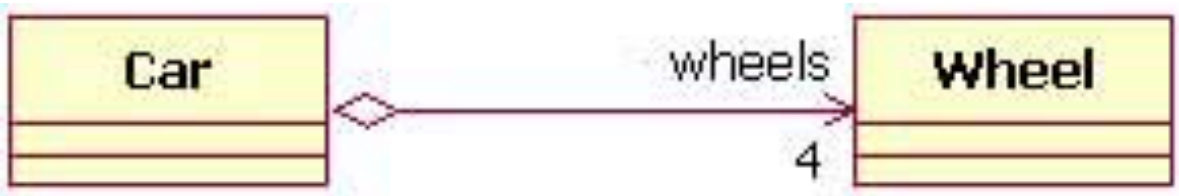
Association Relationships - Name

- We can also name the association
  - Give semantic information about the association
  - It is optional (same for role and multiplicity)



Basic Aggregation

- A special type of association
  - Represents a relationship of the type *is-part-of*
  - The lifecycle of a *part* class is independent from the *whole* class's lifecycle

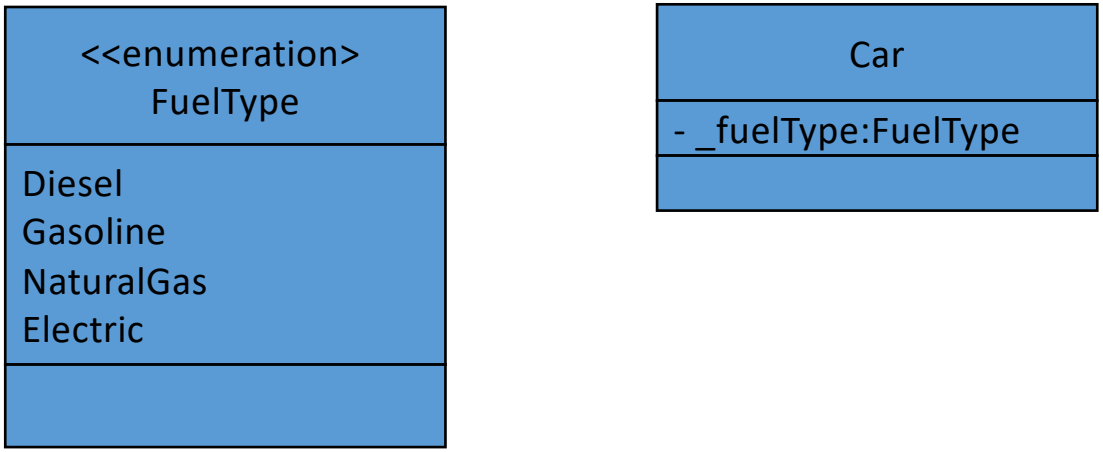


Composition Aggregation

- Another special type of association
  - Represents a relationship of the type *is-part-of*
  - The child class's instance lifecycle is dependent on the parent class's instance lifecycle

Enum type in UML

- Use <<enumeration> stereotype
- Specify the distinct values in the attribute section
- Not mandatory to use the association
  - Can specify an attribute of the enum type



Exemplo: Gestão de Voos

- Pretende-se desenvolver uma aplicação para a gestão de voos de companhias aéreas. A aplicação gere as várias companhias aéreas existentes. Cada companhia aérea tem 0 ou mais voos e tem um ou mais funcionários. A aplicação sabe ainda os vários aeroportos existentes
- Um voo tem sempre associado um número (único no contexto da companhia aérea que o gere), um aeroporto de origem e outro de destino, bem como a hora de partida (hh:mm). Cada Aeroporto sabe os voos de partida e de chegada.
- Um aeroporto é identificado quer pelo seu código IATA, quer pelo seu código ICAO. O aeroporto está instalado numa dada cidade e tem associado um custo de utilização por voo. O código IATA e ICAO de um aeroporto são únicos dentro do contexto da aplicação de gestão de companhias aéreas
- Por fim, uma companhia aérea tem um nome e um código IATA próprio, únicos na aplicação de gestão de companhias aéreas. Cada companhia tem ainda um determinado conjunto de trabalhadores. Cada trabalhador tem um nome. Existem diferentes tipos de trabalhador(pessoal de terra, piloto e assistente de bordo) e todos trabalham mas o trabalho realizado é distinto para cada tipo.

UML Class Diagram