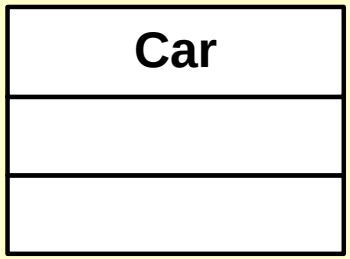
Object-Oriented Design

UML 2.0 Basics

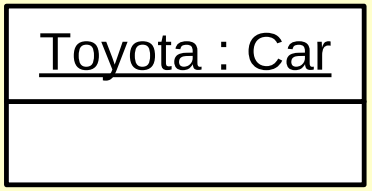
UML 2.0: Classes



The name of the class must be

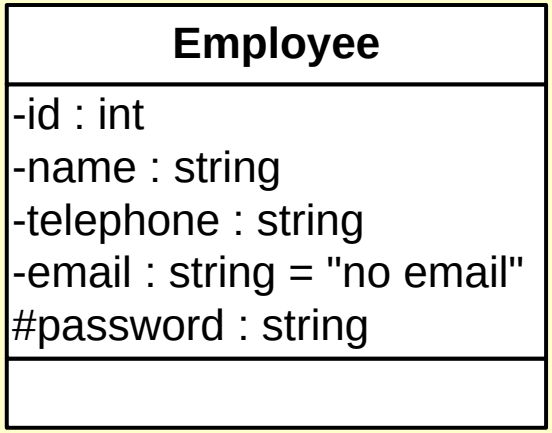
- centered
- capitalized
- bold

UML 2.0: Objects



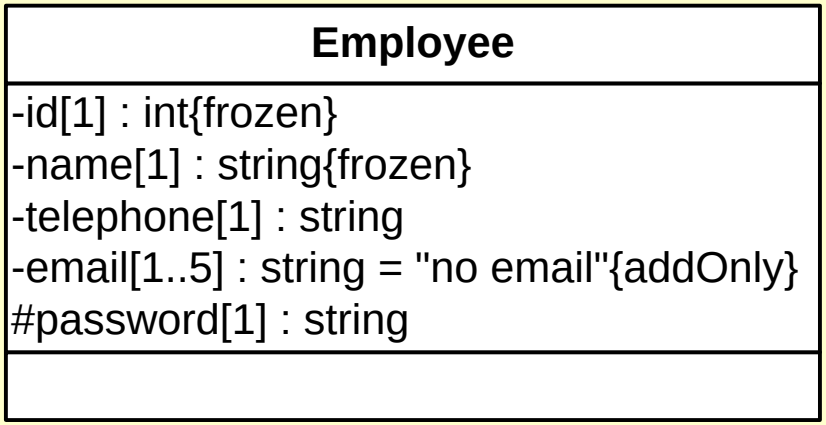
- object Toyota is an *instance of* the class Car

UML 2.0: Attributes



- Attributes must start with lowercase letters
- Are of two kinds:
  - inline attributes
  - attributes by relation (more on this later)

UML 2.0: Inline Attributes' Syntax




```
visibility / name : type [multiplicity] =
  default_val {properties & constraints}
visibility ::= {+|-|#|~}
multiplicity ::= [min..max]
```

Inline Attributes' structure

```
visibility name [multiplicity] : type = default
  {properties, constraints}
```

- visibility (optional): specifies if the attribute is accessible from outside the class
- name (mandatory): the name of the attribute
- multiplicity (optional): specifies the number of values the attribute can store. Default value: 1
- type (optional): specifies the data type of the attribute. Basic data types are: bool, int, float, string.
- default value (optional): specifies the initial value of an attribute
- properties, constraints (optional): contains additional information about the attribute (more on this later)

Visibility (or accessibility)

- Indicates if the attribute can be accessed from outside the class. It can be :
  - public (+) : accessible from outside
  - private (-) : accessible only by the class' methods
  - protected (#) : accessible only by the class' methods and by the class' descendants
  - package (~): accessible only from within the same package

Important!

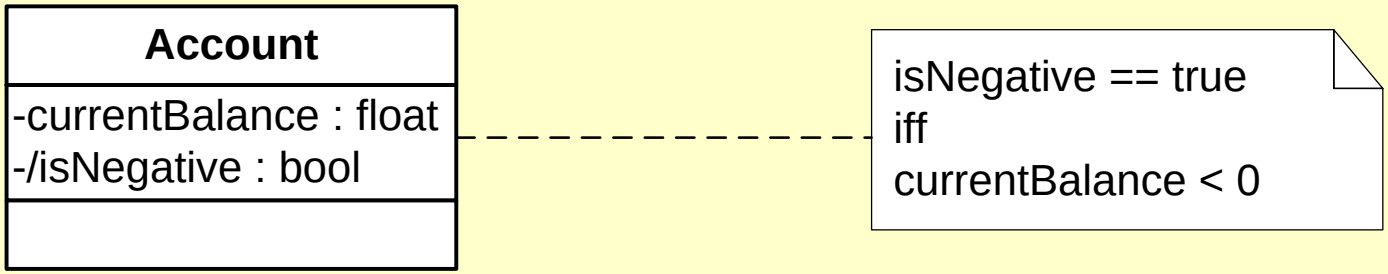
Attributes must (almost) always be

PRIVATE

or PROTECTED at most

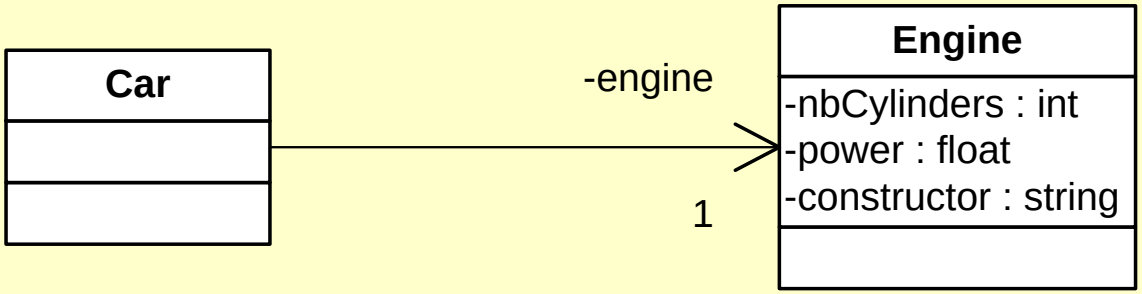
see: encapsulation

Derived attributes



- attributes which can be computed from other attributes
- preceded by a “/”

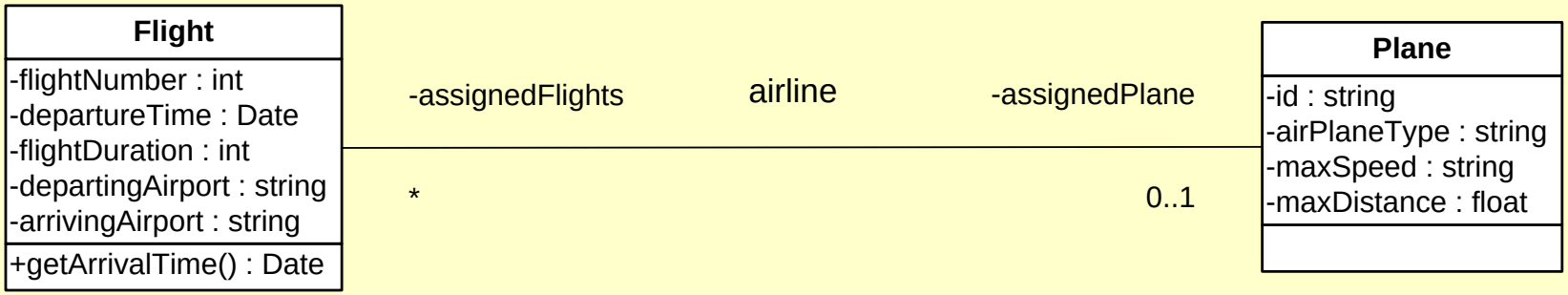
Attributes by relation



- engine is called an attribute by relation
- In the resulting code:

```
public class Car {
        …
        Engine engine;
        …
}
```

Attributes and roles

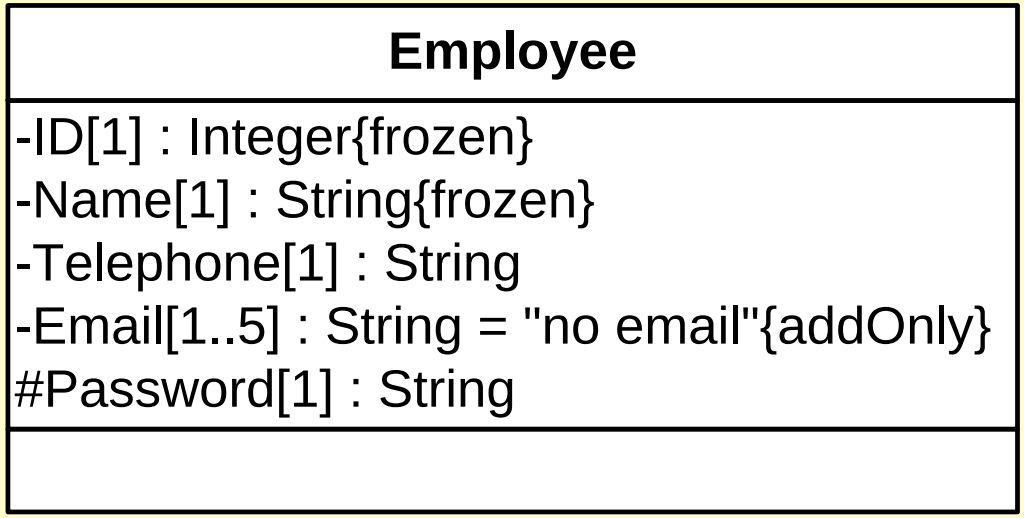


- assignedFligts and assignedPlane are the roles of the relation (or association) airline
- assignedFlights and assignedPlane are attributes by relation

Attribute Properties

- An attribute can be
  - {changeable}: default, can be read and written
  - {frozen}: a constant
  - {addOnly}: when the attribute is a container (multiplicity > 1). Elements can only be added.

Attribute properties: example

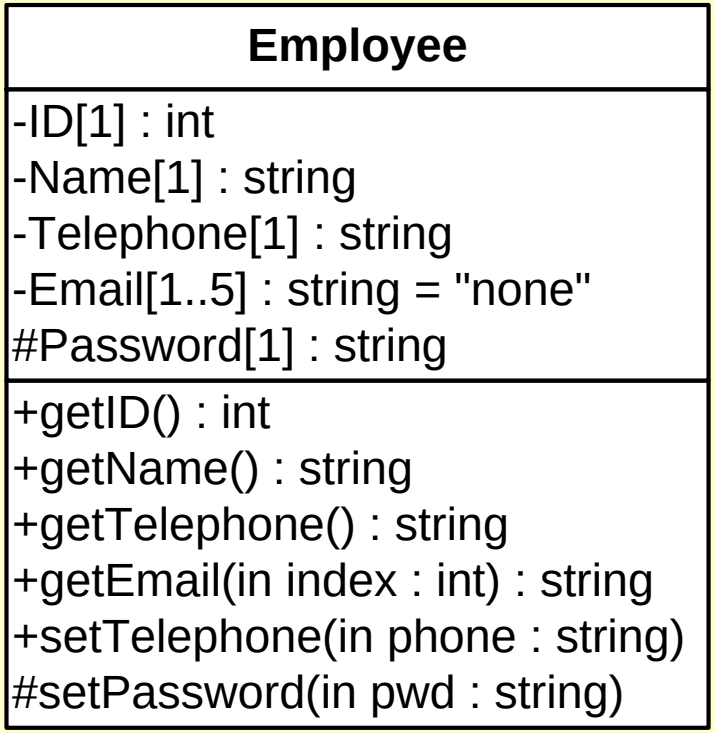


The syntax of methods

```
visibility name (param_list): return_type
```

- visibility (optional): specifies if the method is *callable* from outside the class
- name (mandatory): the name of the method
- param_list (optional): the list of parameters for the method
- return_type (optional): specifies the data type which is returned by the method. Basic data types are: bool, int, float, string.

The syntax of methods

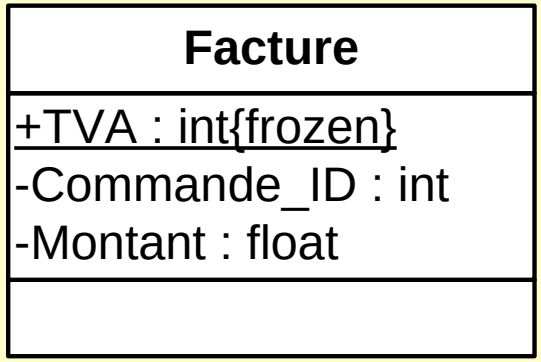


On visibility of methods

- Usually methods are made public
- Private methods are used to implement a complex public method without exposing internal details

Class (or static) attributes

- A class (or static) attribute is an attribute common to all instances of a class



Class (or static) methods

- A class (or *static*) method is a method which can be invoked without instantiating a class
- Example : methods which generate new classes