Course Application Design

Creating beautiful and reliable applications
UML

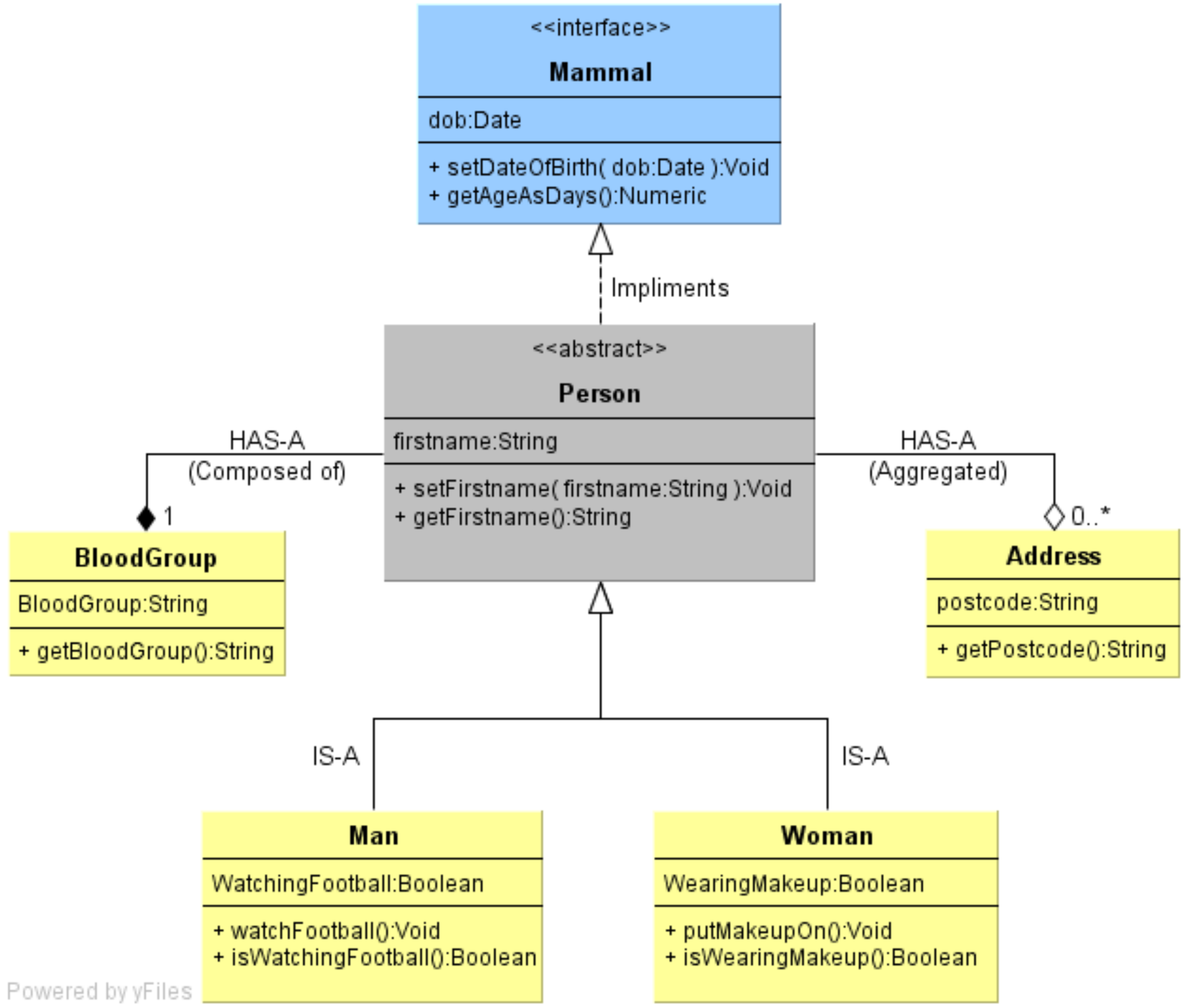
Michiel Noback
Institute for Life Sciences and Technology
Hanze University of Applied Sciences

Part two

UML

A really simple intro so you are able to read the schematics

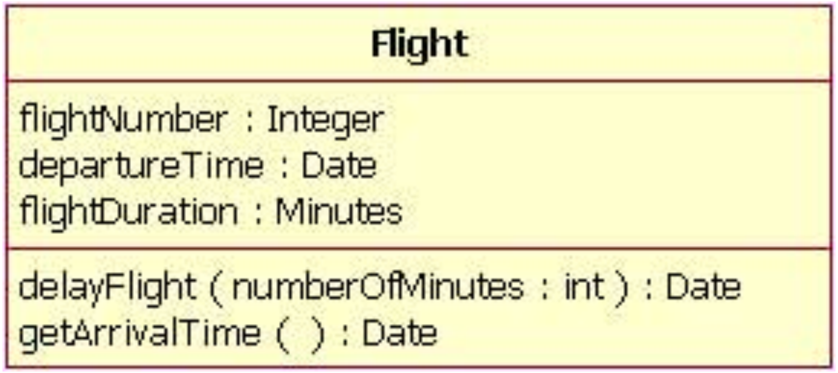
UML - Unified Modeling Language



UML

- UML is used to create graphical representations of your design *–any design-*
- Because there are strict rules/conventions, other designers can “read” your model easily
- Here we only deal with OO design UML

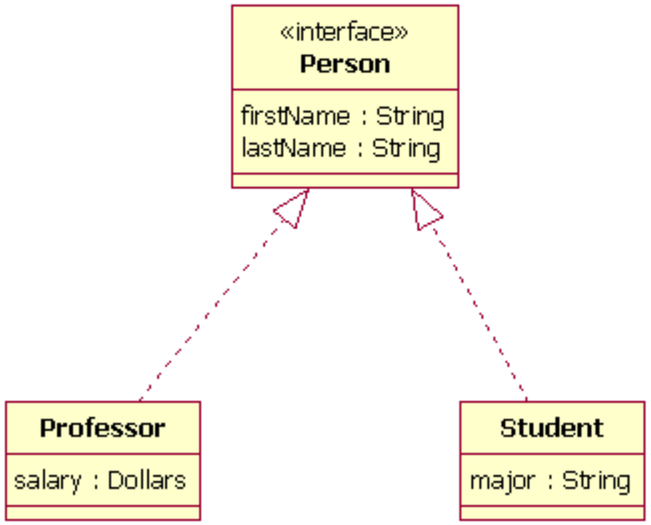
UML symbols: class



- A rectangle containing three compartments stacked vertically
- Top compartment: class name
- Middle compartment: class attributes/properties
- Bottom compartment: class operations/methods

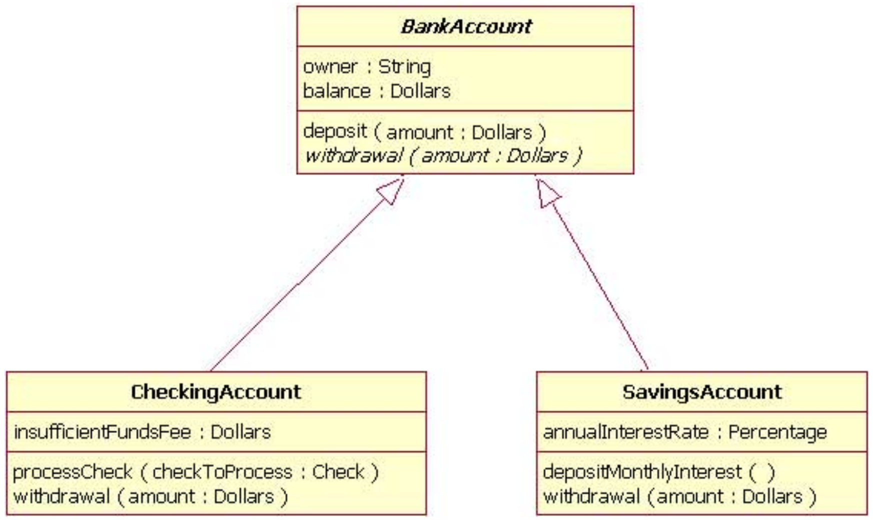
This and the following slides rely heavily on http://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/

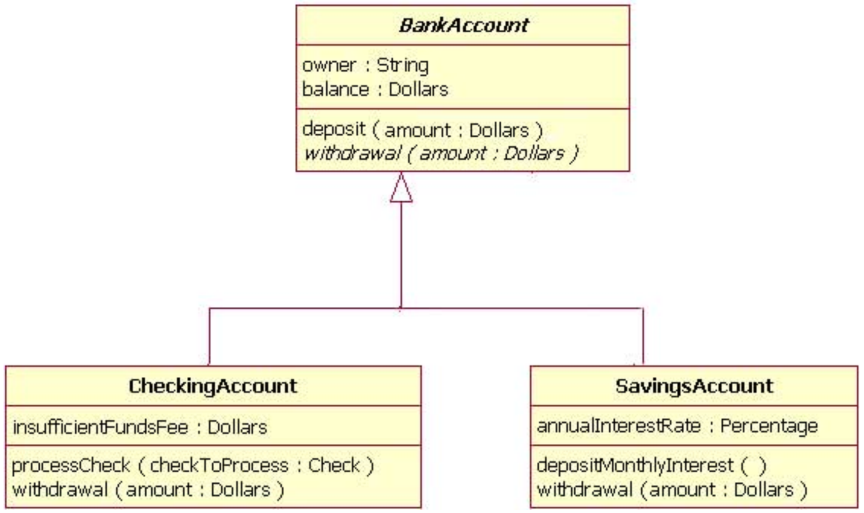
UML symbols: interface



- A class and an interface differ: A class can have an actual instance, whereas an interface must have at least one class to implement it.
- An interface is drawn just like a class, but the top compartment of the rectangle also has the text "«interface»"

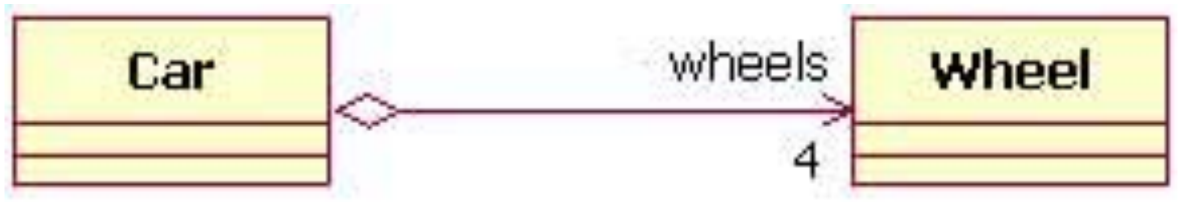
UML symbols: inheritance


BankAccount
owner : String
balance : Dollars
deposit (amount : Dollars)
withdrawal (amount : Dollars)
CheckingAccount
insufficientFundsFee : Dollars
processCheck (checkToProcess : Check)
withdrawal (amount : Dollars)
SavingsAccount
annualInterestRate : Percentage
depositMonthlyInterest ( )
withdrawal (amount : Dollars)


BankAccount
owner : String
balance : Dollars
deposit (amount : Dollars)
withdrawal (amount : Dollars)
CheckingAccount
insufficientFundsFee : Dollars
processCheck (checkToProcess : Check)
withdrawal (amount : Dollars)
SavingsAccount
annualInterestRate : Percentage
depositMonthlyInterest ( )
withdrawal (amount : Dollars)

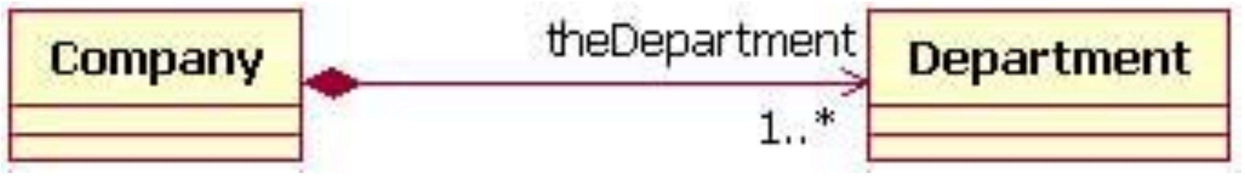
- Inheritance is indicated by a solid line with a closed, unfilled arrowhead pointing at the super class
- It can be drawn with separate lines or using a tree notation

UML symbols: Basic aggregation



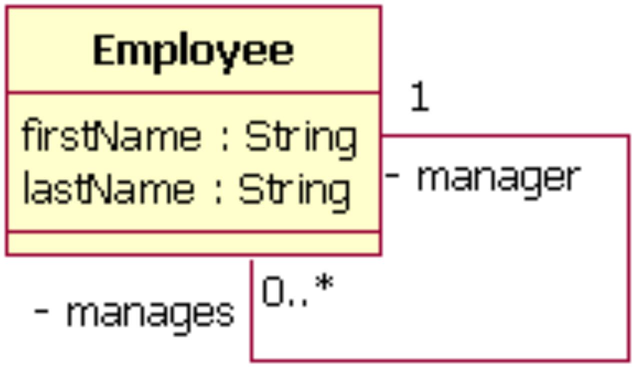
- One class is a part of another class
- In *basic* aggregation, the child class instance can outlive its parent class
- Drawn with a solid line from the parent class to the part class, with an unfilled diamond shape on the parent class's association end

UML symbols: Composition aggregation



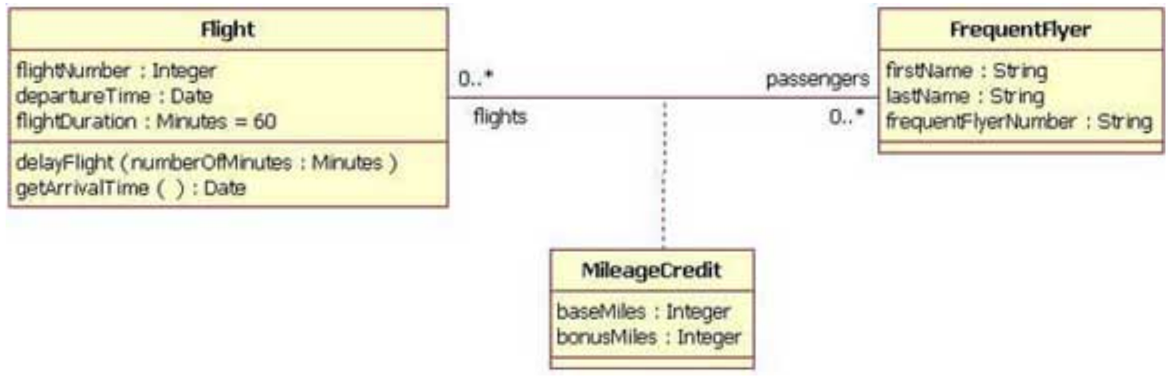
- One class is a part of another class
- In *composition* aggregation, the child class's instance lifecycle is dependent on the parent class's instance lifecycle. Also, the part class can only be related to one instance of the parent class
- Drawn with a solid line from the parent class to the part class, with a filled diamond shape on the parent class's association end

UML symbols: Reflexive aggregation



- One class is associated with itself
- Here it means that an instance of Employee can be the manager of other (0 to many) Employee instances
- Drawn with a solid line

UML symbols: association



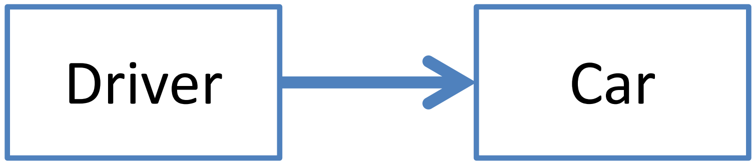
- the association line between the primary classes intersects a dotted line connected to the association class
- Here: when an instance of a Flight class is associated with an instance of a FrequentFlyer class, there will also be an instance of a MileageCredit class

Association
- aggregation –
composition

- They seem a bit like the same thing
- Here some more detail

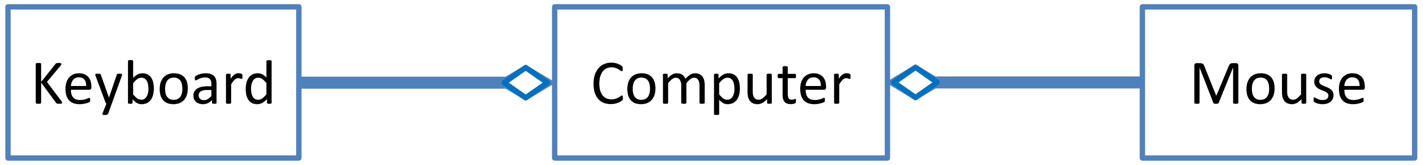
Association

- It is a relationship between objects.
- One object is connected to the other.
- Usually called as “has-a” relationship.
- Both objects have independent life-cycle.
- Each object owns their actions and will not affect other object.



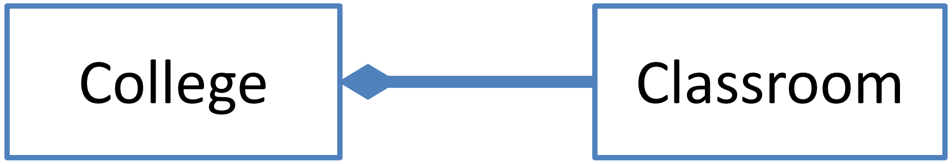
Aggregation

- Specialized form of Association.
- Usually called as “has-a” relationship.
- Each object has an independent life-cycle.
- A whole-part relationship between a component object and an aggregate object.
- Sense of ownership between objects.



Composition

- Specialized form of Aggregation.
- Usually called as “has-a” relationship.
- Child Object has dependent life-cycle. This is what separates it from aggregation.
- A whole-part relationship between a component object and an aggregate object.
- Sense of ownership between objects



UML symbols: visibility

- Visibility symbols are

Symbol	Visibility / Scope
+	Public
#	Protected
-	Private
~	Package (default)