Dalhousie University

Faculty of Computer Science

CSCI 3132 – Object Orientation and Generic Programming

Week 2 – Class Diagrams

Class Diagrams

- Represents the static view of an application
  - Used for describing the attributes and operations of a class
  - Provides an initial set of notation elements
- Class diagram shows the types (classifiers) being modelled within the system including:
  - Classes
  - Interfaces
  - Data types
  - Components

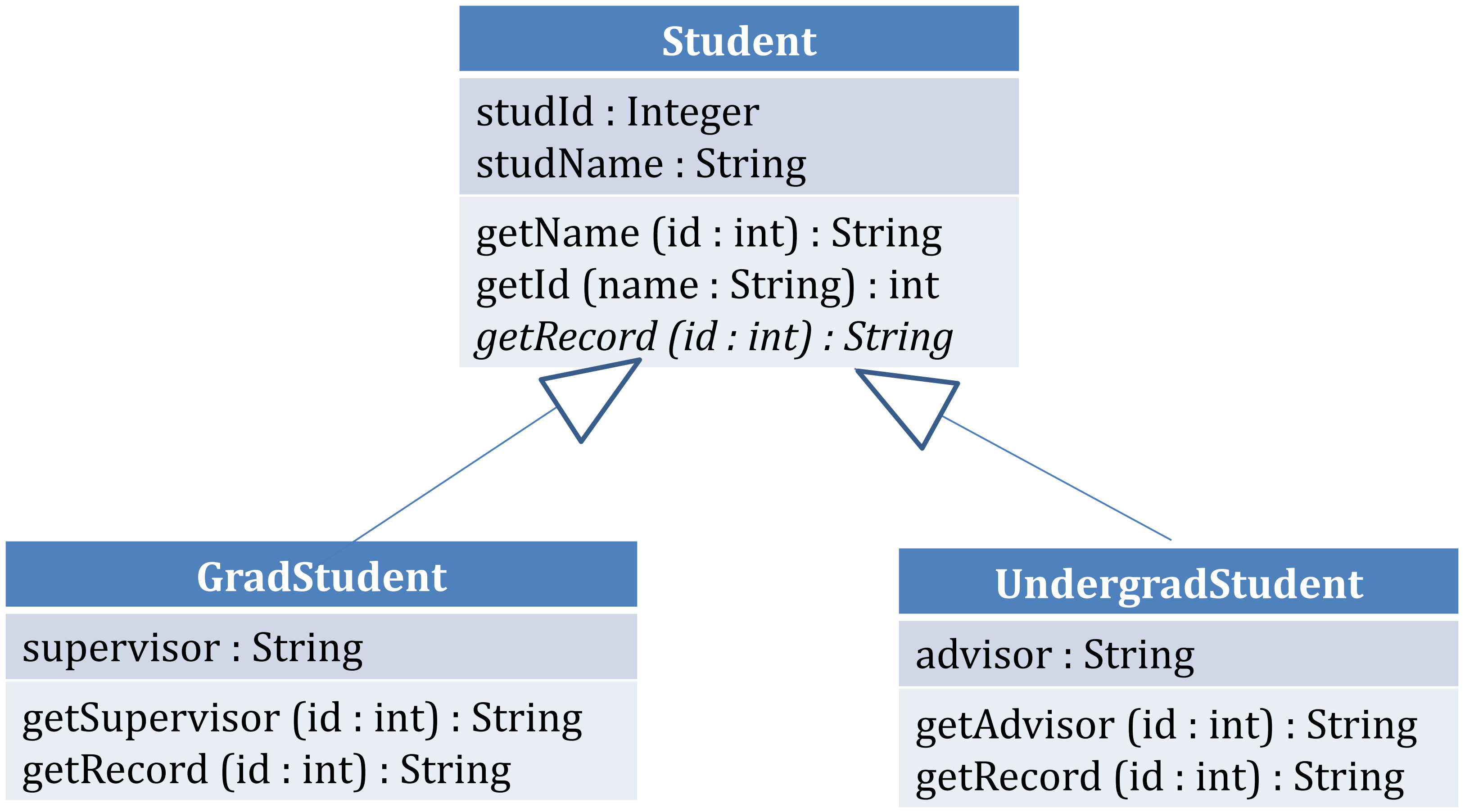
UML Representations

- Represented by a box with three sections consisting of:
  - Name of the class
  - List of attributes
  - List of operations
- Attribute section is optional
  - Written as *name : attribute type*
  - Use types provided by the programming language if class diagram is used to generate code
  - Default values can be shown
    name : attribute type = default value
- Operations list is optional
  - Written as *name(parameter list) : type of return value*
  - *in* or *out* maybe used in parameter list to indicate input or output parameter. Defaults to in

Class Name
Class attributes
Class operations

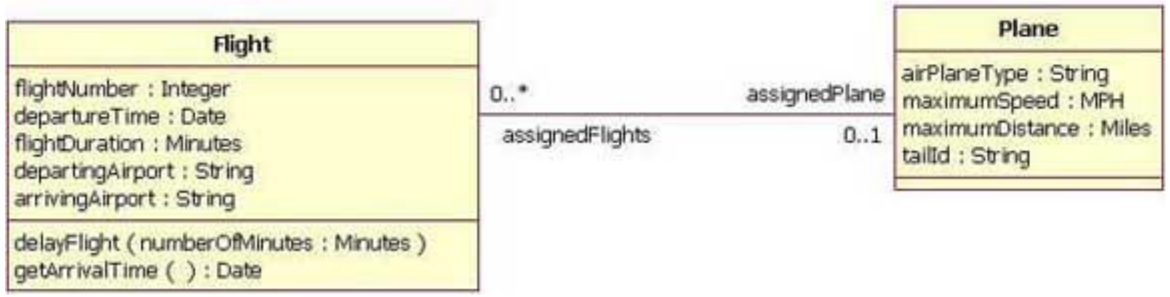
Student
studId : Integer studName : String
getName (id : int) : String getId (name : String) : int

Inheritance

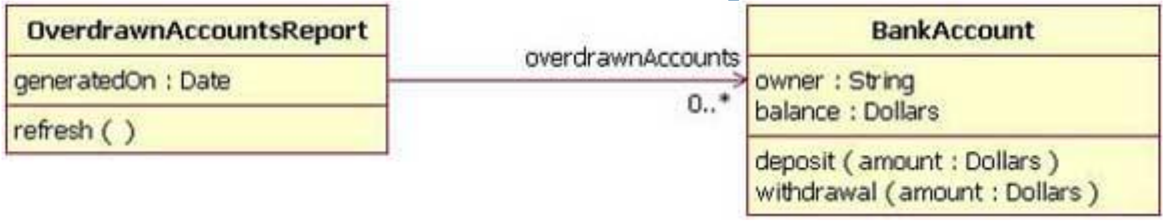

Student
studId : Integer
studName : String
getName (id : int) : String
getId (name : String) : int
getRecord (id : int) : String
GradStudent
supervisor : String
getSupervisor (id : int) : String
getRecord (id : int) : String
UndergradStudent
advisor : String
getAdvisor (id : int) : String
getRecord (id : int) : String

Associations

- Bi-directional association
  - Linkage between two classes showing role names and multiplicity values
  - Shown as a connecting line with multiplicity shown.



- Uni-directional association
  - Only one class knows that the relationship exists



Figures source: CC-BY-SA-3.0 Wikimedia Commons

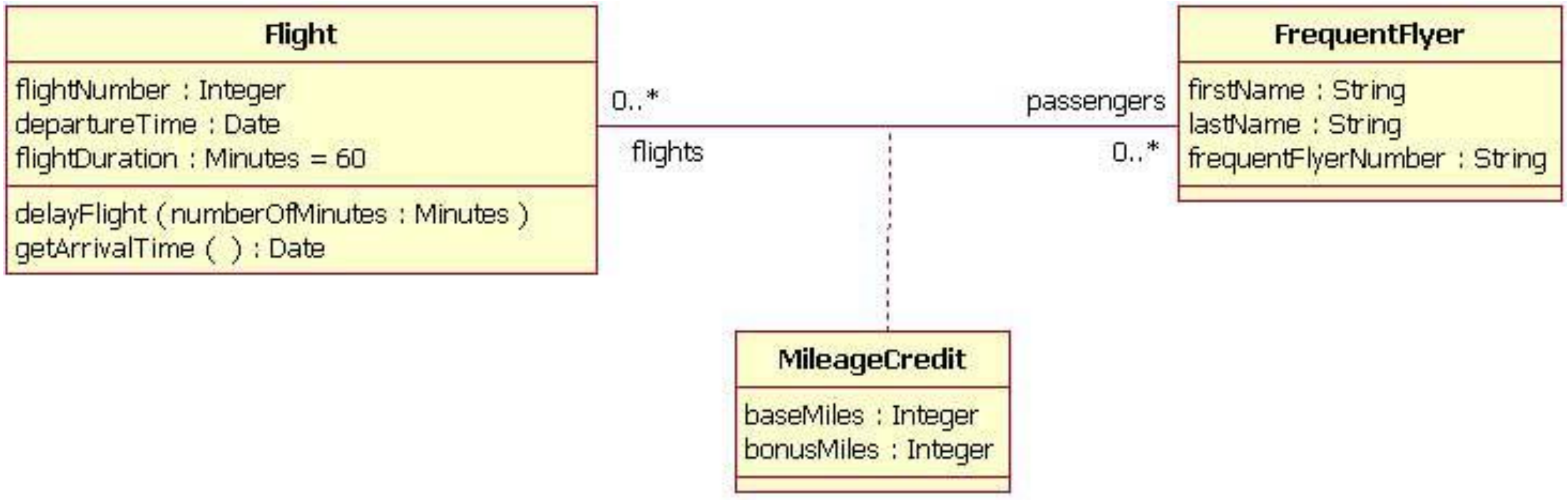
Multiplicity

- Some multiplicity examples are shown below

Indicator	Meaning
0..1	Zero or one
0..* or *	Zero or more
1..*	One or more
1	Exactly one
3	Exactly three
2..4	Two to four

Associations

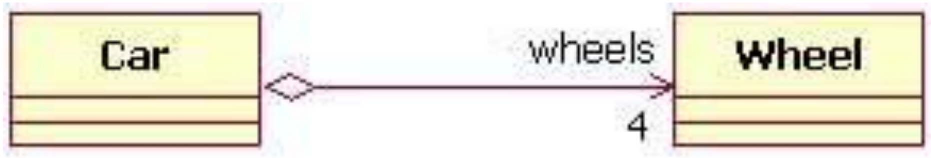
- Association class
  - Used when there is a need to include another class because it includes valuable information about the relationship



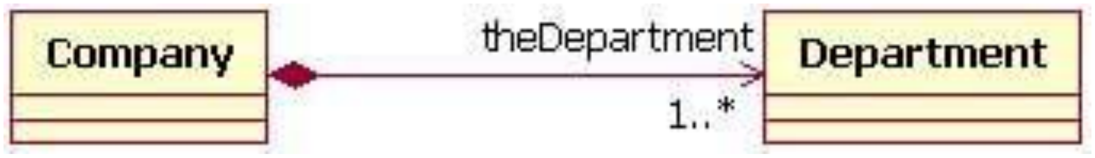
Source: CC-BY-SA-3.0 Wikimedia Commons

Aggregation

- Models a whole to its part or has-a relationship
- Mainly two types:
  - Basic aggregation
    - Child class instance is independent of parent class instance
    - E.g. car and wheel



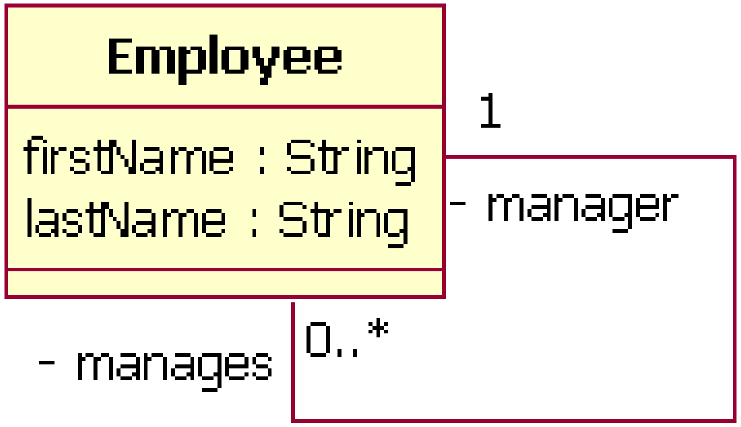
  - Composition aggregation
    - Child class instance is dependent on parent class instance
    - E.g. company and department



Source: CC-BY-SA-3.0 Wikimedia Commons

Reflexive Association

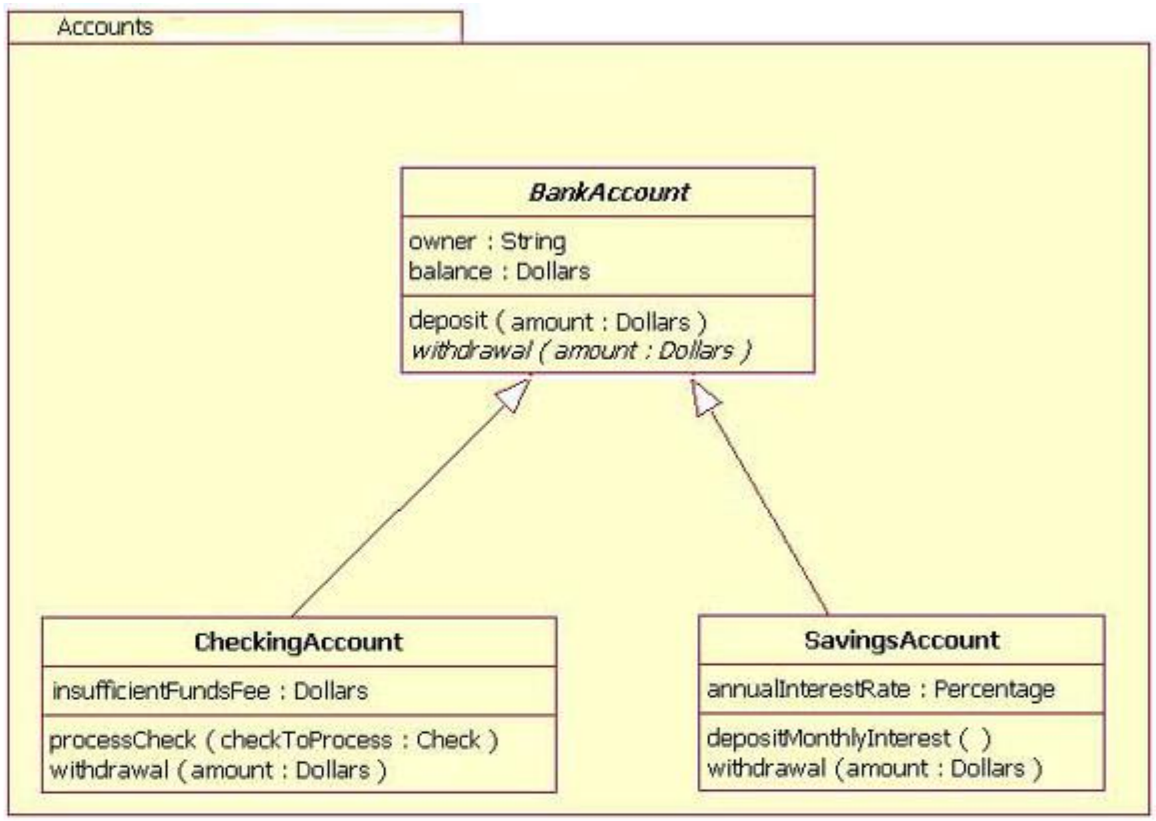
- Class is associated with itself
  - One instance of the class is related to another instance of the same class
  - E.g. Employee class can relate to itself through manager role
  - Does not mean class' instance is related to itself



Source: CC-BY-SA-3.0 Wikimedia Commons

Packages

- Packages are used to organize the model's classes
  - Organized as namespaces

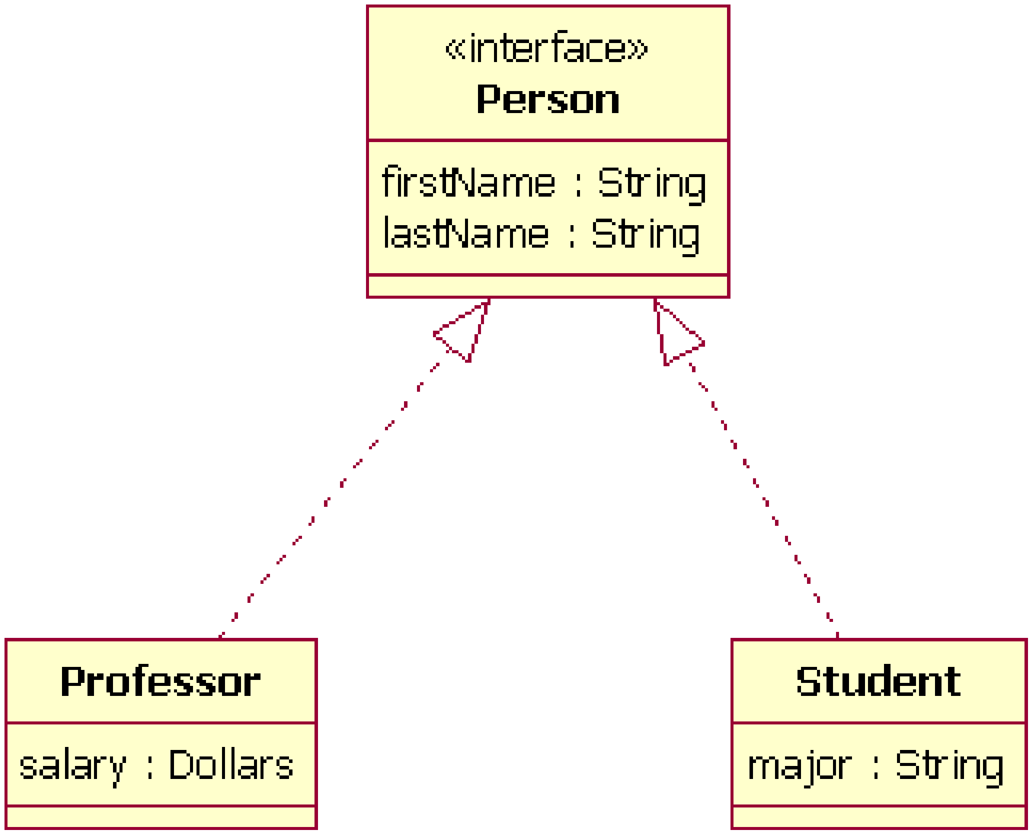


Accounts
BankAccount
owner : String
balance : Dollars
deposit (amount : Dollars)
withdrawal (amount : Dollars)
CheckingAccount
insufficientFundsFee : Dollars
processCheck (checkToProcess : Check)
withdrawal (amount : Dollars)
SavingsAccount
annualInterestRate : Percentage
depositMonthlyInterest ( )
withdrawal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

Source: CC-BY-SA-3.0 Wikimedia Commons

Interfaces

- A class can have an actual instance of its type, while an interface must have at least one class to realize or implement it



Source: CC-BY-SA-3.0 Wikimedia Commons

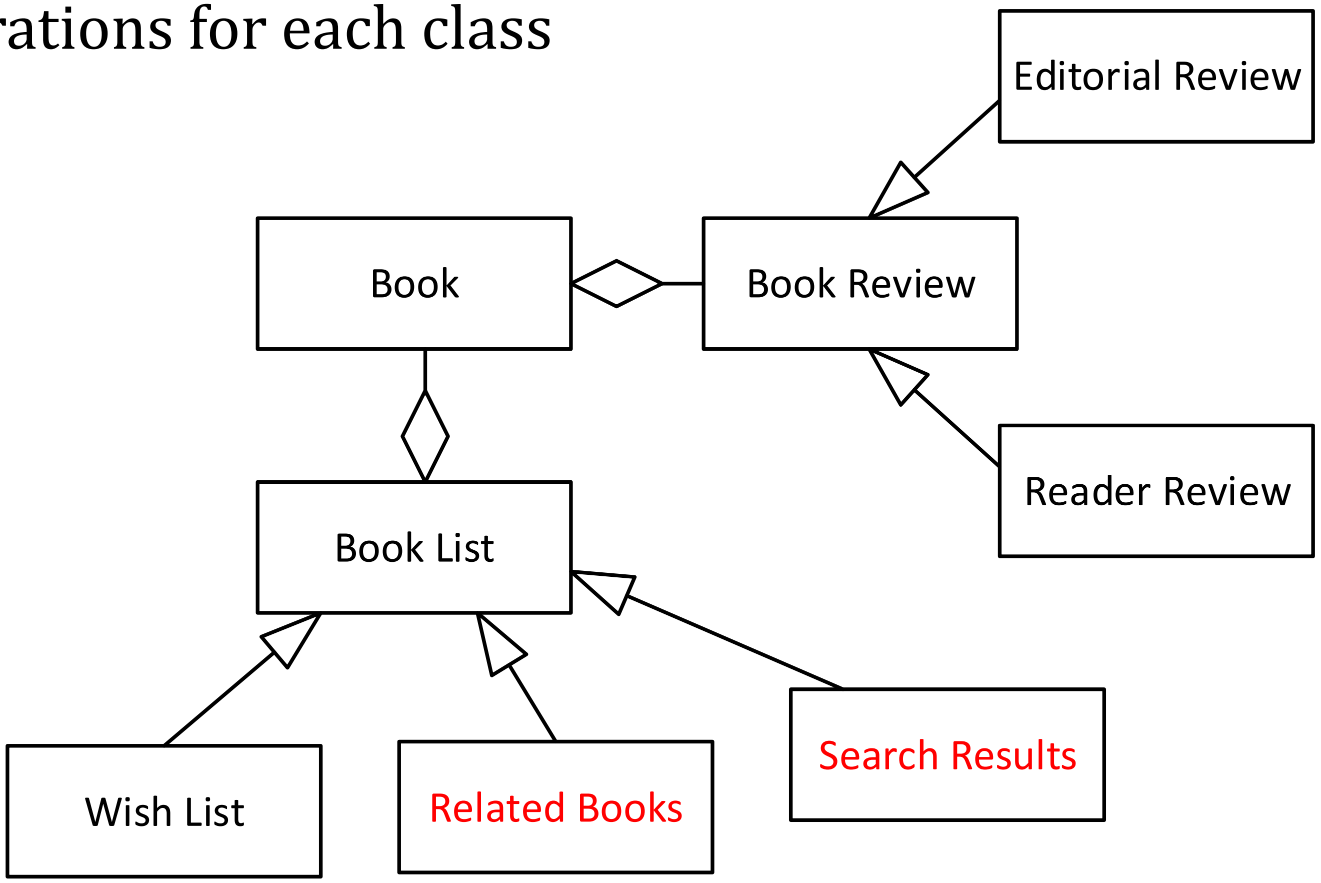
Visibility

- Marks for UML supported visibility types

Indicator	Meaning
+	Public
#	Protected
-	Private
~	Package

Exercise

- Draw a class diagram from the following domain model using appropriate relationships. Identify a few attributes and operations for each class



References

- https://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/