UML for design: Class Diagrams

ENGR 110 #15 2016

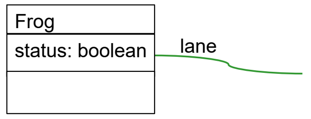
Peter Andreae

Computer Science

Victoria University of Wellington

Class diagrams

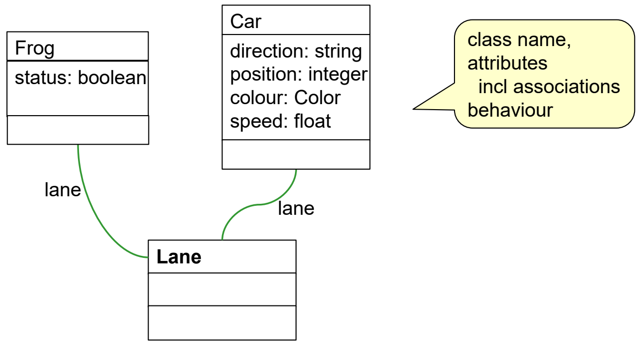
- More useful for real design
- Abstract from individual objects to classes of objects.



- What is the ontology of Class diagrams?
  - Classes:
    - categories of objects all of the same type, with the same behaviour
  - Attributes of the objects in the classes
    - Associations (relationships) between objects in the classes
  - Actions/behaviour of objects in the classes

Class Diagrams

- Describe the categories of the objects, rather than the objects themselves:



Notes
- Associations can have additional information on them
- Behaviour specifies the actions that can be done on the objects

Designing Class Diagrams

- How do you work out what classes there should be?
  - No mechanical algorithm: it is a design issue
  - Starting point: the nouns/noun phrases in the specification
    eg, the initial text description, the use cases, etc.
    - note: this is not enough:
      - Some nouns don't need to be classes
      - Some classes aren't explicitly mentioned in the specification.

Group Project sign-up system

- System should allow students to sign up for group projects for their courses.
  - Each course will have one or more group projects.
  - Each project will have a task, a lab place, a maximum size, and a leader.
- A course administrator needs to be able to set up the projects for their course.
- Students should be able to specify the projects that they would like to do, and a preference order.
- When sufficient students have entered their preferences, the system should show which projects students are likely to be assigned to, along with the likelihood.
- The course administrator can "commit" to an allocation, at which point, the system will permanently allocate signed up students to projects.
- Students should be able to withdraw from projects, and enter new preferences, but they won't be actually assigned until the next time the administrator "commits".
- Administrators should be able to get lists of students in each project of their course.

What are the objects and classes?

- Classes, attributes, associations:

Course

Project

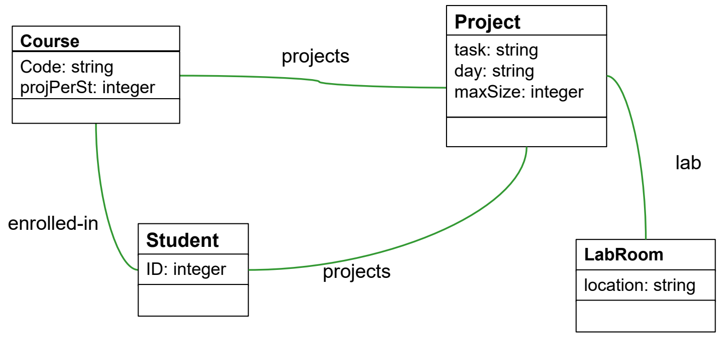
Student

Administrators

Lab rooms

More on Associations

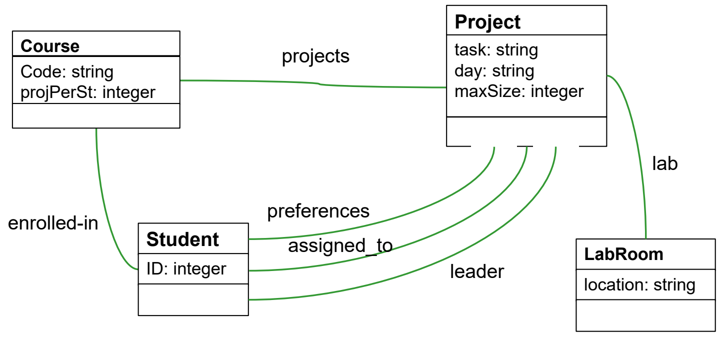
- What does an association mean:
  - Objects of one class need to know about objects of the other class
  - label describes the relationship



More on Associations: Multiple

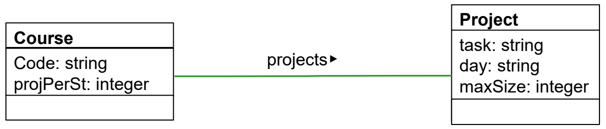
- May have multiple associations between two classes

  if there are different relationships.

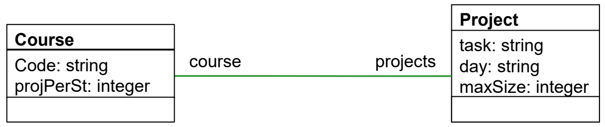


More on Associations: roles

- How do we read the label?
  - just the name of relationship:
    (typically left to right, can add a ▸ or ◂ to clarify)



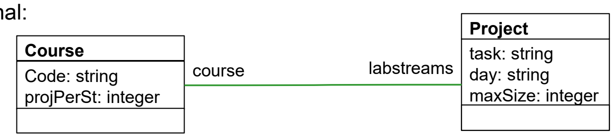
  - roles of classes at each end of the association:
    describe what role the other objects play in this object



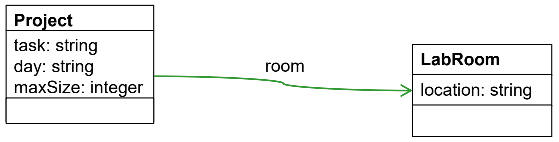
Role on far end of association will turn into field name in the code!

More on Associations: navigability

- Do both classes need to know about the other?
  - bidirectional:



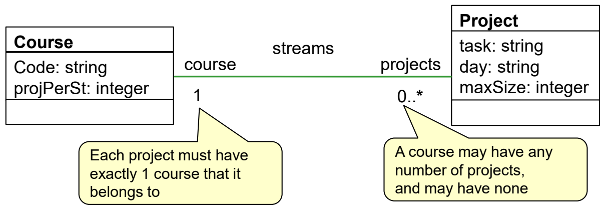
  - unidirectional:
    - the Project needs to know which room;
    - the room doesn't need to know which projects are in it;

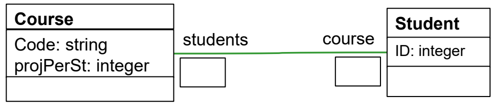


    - Note: roles are not necessary here

More on Associations: multiplicity

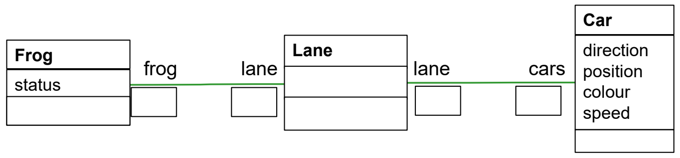
- How many objects involved:
  - specify the range of the number of objects on each end:
  - eg: 1..1, 0..1, 0..5, 1..5, 1..*, 0..*

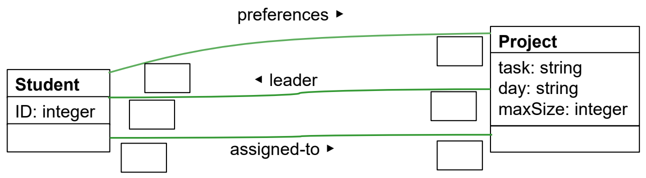

Course
Code: string
projPerSt: integer
course
streams
projects
Project
task: string
day: string
maxSize: integer
1
0..*
Each project must have exactly 1 course that it belongs to
A course may have any number of projects, and may have none



More on Associations: multiplicity

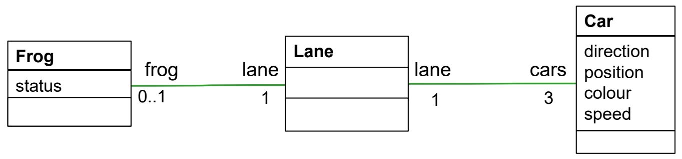
- How many objects involved:

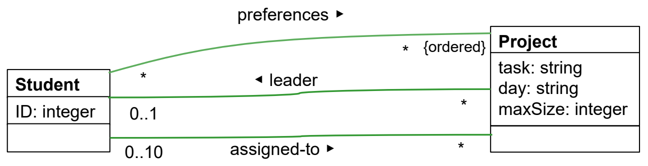




More on Associations: multiplicity

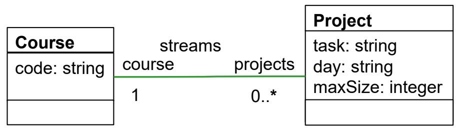
- A set of items may be ordered





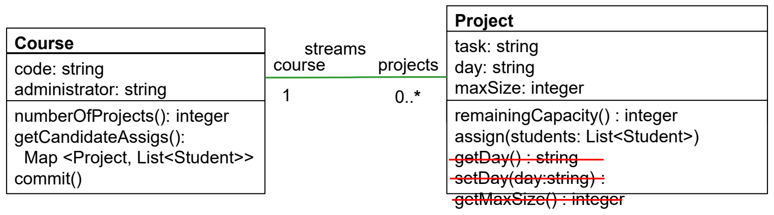
Summary so far

- Class diagrams show
  - Classes
    - name
    - properties
      - attributes (*name* : *type*)
      - associations
    - behaviour
  - Associations between classes
    - name (and/or role descriptions of each end)
      - describes the relationship between objects of the two classes
    - navigability
      - which object "knows about" the other
        (bidirectional —— $\longleftrightarrow$ or unidirectional $\longrightarrow$)
    - multiplicity (min : max)
      - how many other objects one object can be related to
        eg: 1 * 0:1 1:* 0:n n:m
      - set or ordered list? {ordered}



Class Behaviour

- 3rd component of a class:
  - specifies the actions that can be performed on objects of the class
  - ie, the methods
  - Specify
    - name of action,
    - parameters and types,
    - return type
  - Don't specify actions for getting and setting properties


Course
code: string
administrator: string
numberOfProjects(): integer
getCandidateAssigs():
Map <Project, List<Student>>
commit()
streams
course
projects
1
0..*
Project
task: string
day: string
maxSize: integer
remainingCapacity() : integer
assign(students: List<Student>)
~~getDay() : string~~
~~setDay(day:string) :~~
~~getMaxSize() : integer~~