- Lab 1. Running Eclipse.
- Lab 2. class diagrams in Jude

UML tutorials and references

www.sintef.no/time/ELB40/ELB/UML/UML.pdf

dn.codegear.com/article/31863

www.dotnetcoders.com/web/learning/uml

www.ibm.com/developerworks/rational/library/769.html

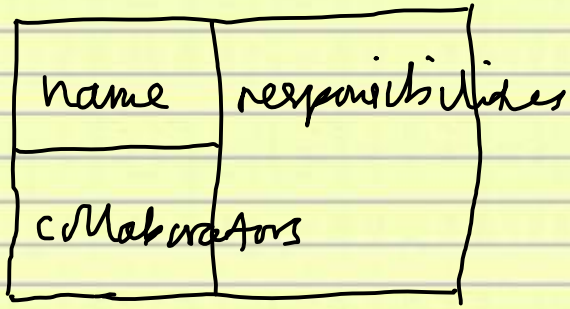
jude.change-vision.com/jude-web/download/try_uml.html

Precursor to UML called CRC cards.
- a way to develop thinking about objects, classes, and their interaction

C - class
R - responsibility
C - collaboration

This is a group activity (the development team)
- everyone sits round a table - no leader - note cards and pencil (eraser)
- Someone proposes a class which abstracts some object that takes part in the application.
- each class gets a name (descriptive)
- other team members propose responsibilities for the class - think about methods or functions as well what data the object (class) keeps inside it

- the class will in general have a relationship with other classes either through data or through methods — collaborators
- team members can criticize (constructively) trying to build concensus on the content of each CRC card
- any member can change anything after discussion
- eventually a team member can take charge of a set of classes and "act out" the behavior of each class
- the end result is a conceptual model in terms of classes and their relationships

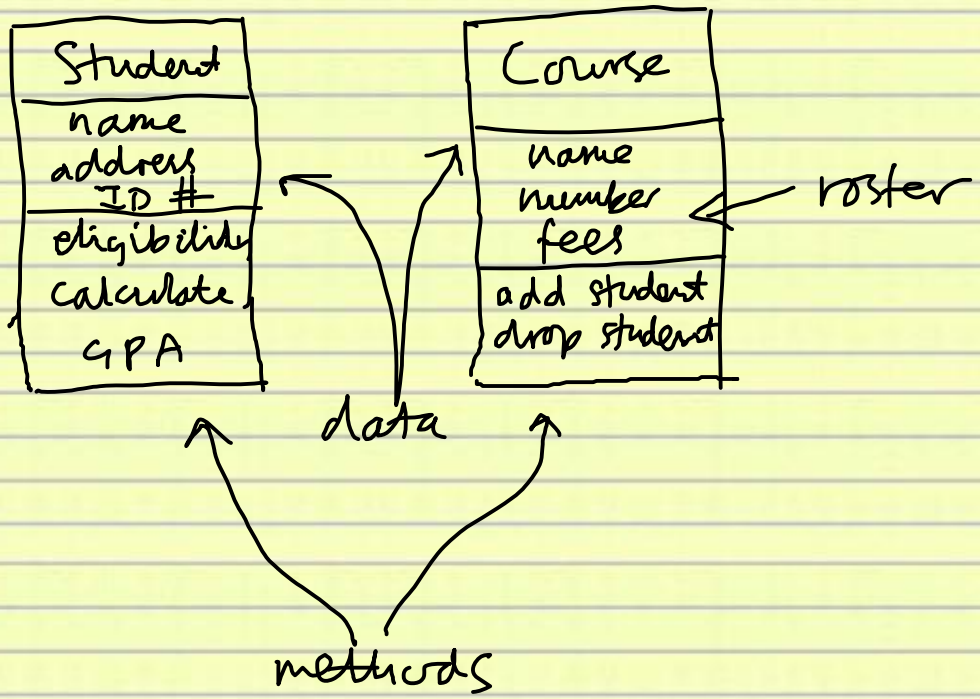


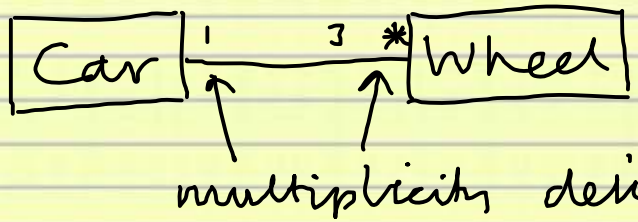
It's easy to turn a CRC card into a class in a UML class diagram


A class represents a set of objects – people, places, things, concepts, events etc.

Objects know about attributes

Objects carry out operations on the data (execute methods)

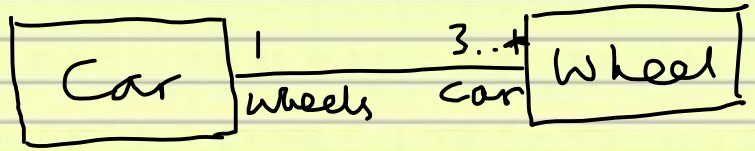

Student
name
address
ID #
eligibility
calculate
GPA
Course
name
number
fees
roster
add student
drop student
data
methods

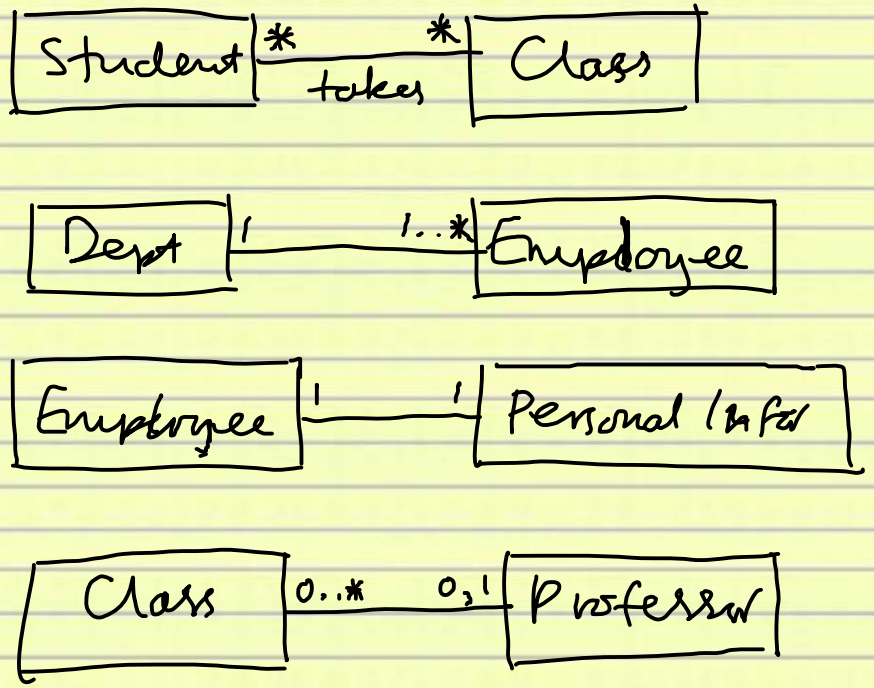



```
class Car {
  List<Wheel> wheels;
}
```

```
class Wheel {
  Car car;
}
```

We invent names but there are also roles for the class

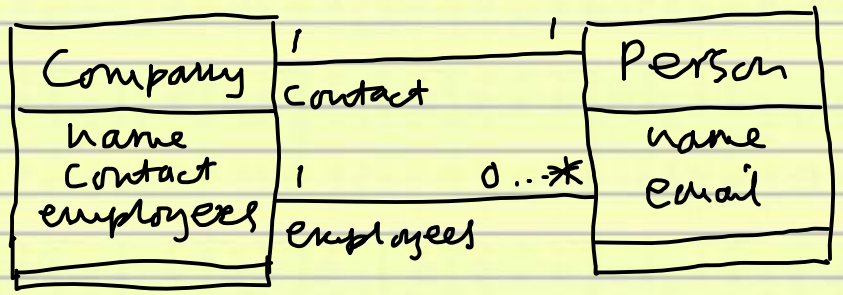




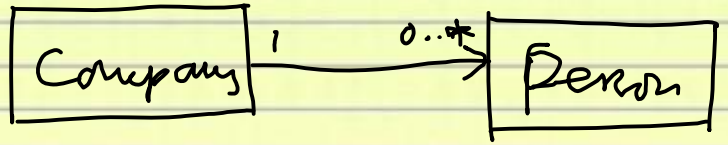

In general the line between classes expresses a HAS-A relationship.

In code this ends up being a contained object

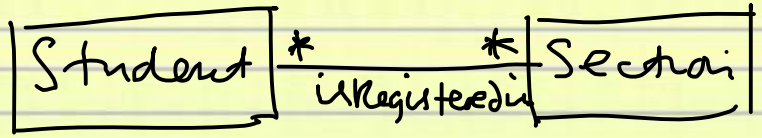
Multiplicities define dependencies one to one, one to many etc.



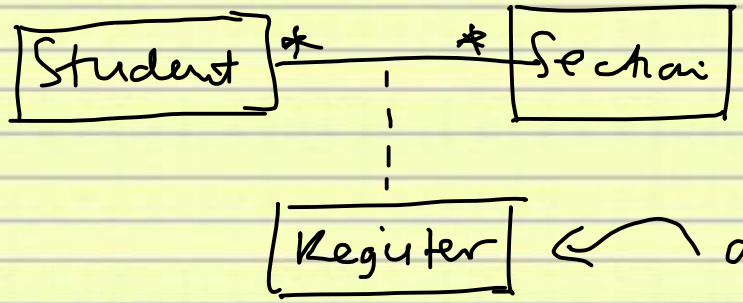

Express navigability. Without arrows, both ways are possible

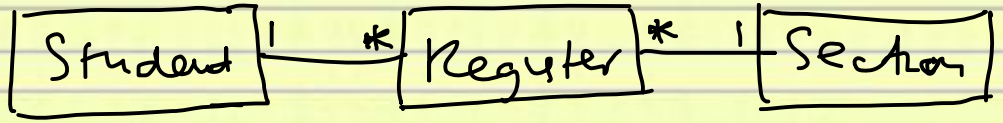



Company knows about its employees (persons) but a person does not know about its company.

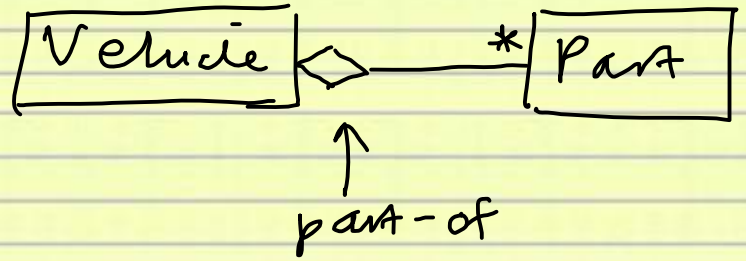


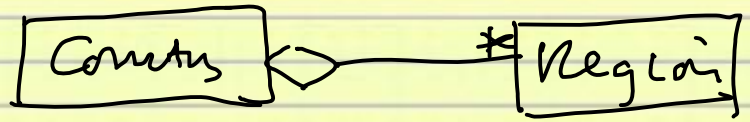

Where to put student grades

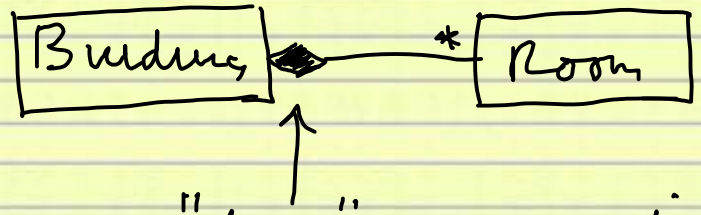


Aggregation – modeled with collections




"strong" aggregation – the building is a member of rooms