Object-Oriented Design and Programming

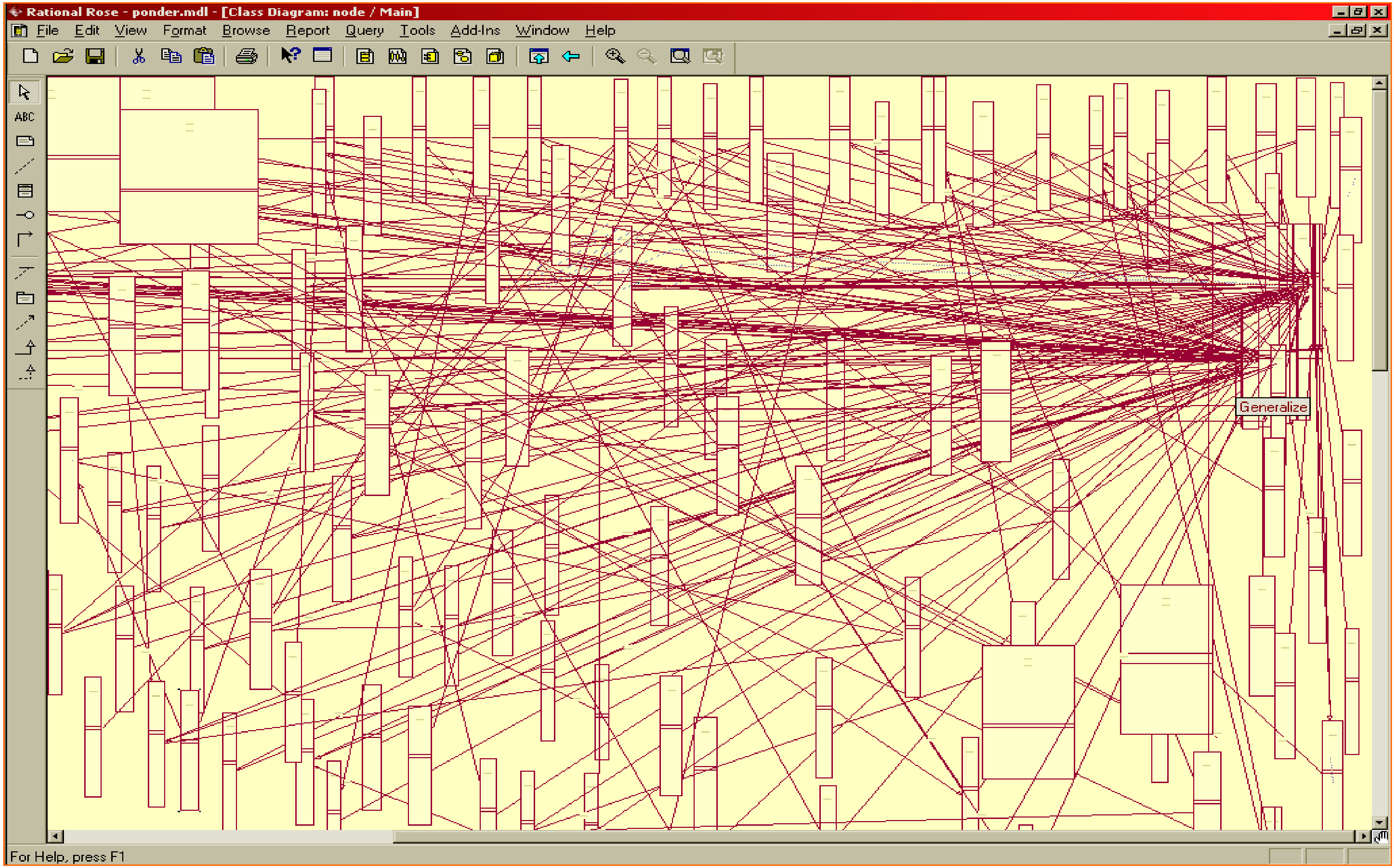
William Knottenbelt

and

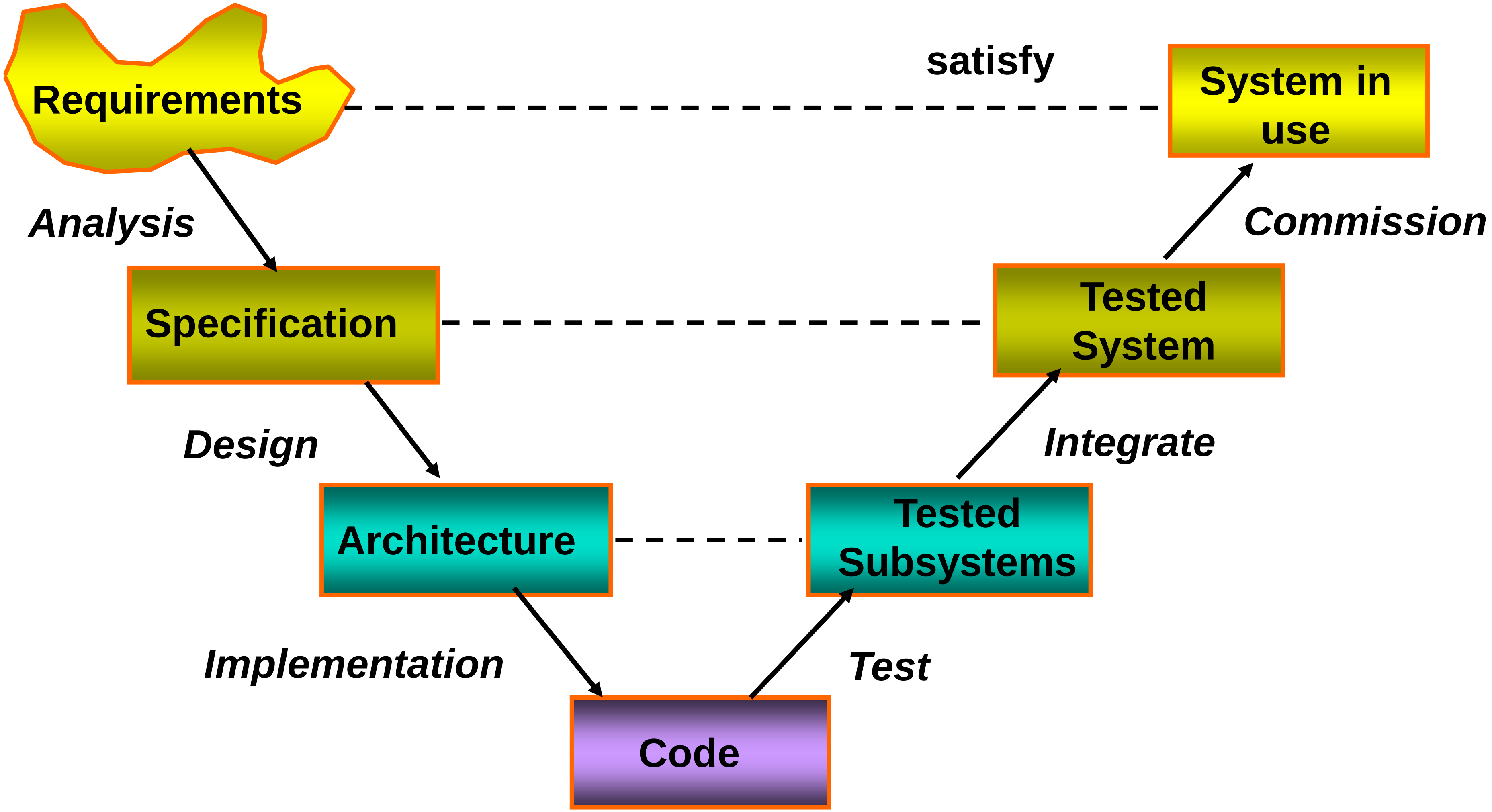
Fidelis Perkonigg

Based on original notes by
Sophia Drossopoulou and Emil Lupu

Objective of this course


Rational Rose - ponder.mdl - [Class Diagram: node / Main]
File Edit View Format Browse Report Query Tools Add-Ins Window Help
ABC
Generalize
For Help, press F1

The Software Production Process



We are concerned with the following two parts of the software production process

- Design (in UML)
- Implementation (in C++); Always starting from a UML diagram

Exam Questions

1. UML design for a simplified real life situation
2. Map UML to C++
3. Write (part of) C++ code

Course Operation

- Lectures
  - Mondays at 11h00 and Fridays at 14h00
  - Notes pretty complete – but sometimes with holes
- Tutorials
  - Fridays at 15h00
  - Pen and paper exercises with solutions
- Assessment
  - 1 coursework (in groups)
  - 2 lab exercises
  - 2 hour exam (together with Logic)

Reading Material (UML, C++)

- C++, any book on C++, e.g.,
  - *Problem Solving with C++,* Walter Savitch, Addison-Wesley.
  - *Navigating C++,* Anderson and Anderson, Addison-Wesley.
  - C++: How to program, Deitel & Deitel, Prentice Hall.
- UML Modelling, any book on UML, e.g.,
  - *Practical Object-Oriented Design with UML*. M. Priestley, McGraw-Hill, 2000
  - *The Unified Modeling Language User Guide*. G. Booch, J. Rumbaugh & I. Jacobson, Addison-Wesley, 2nd Edition, 2005.
  - *Learning UML 2.0.* Russ Miles, O'Reilly, 2006.

Introduction to UML -- object and class diagrams

Analysis and Design

- Analysis and Design require a *modelling notation* and a *methodology*.
- There are many *methodologies*
  - Structured methods (80s) emphasise separation between functions and data.
  - Object-Oriented Methodologies emphasise the encapsulation of data and behaviour, Coad-Yourdon (91), Booch (94), JSD (Jackson – 83), OOSE, Objectory (Jacobson – 92-), OMT (Rumbaugh 91), Fusion (Coleman), Catalysis (Wills & D'Souza), RUP (Rational Unified Process) … *moving towards component-oriented methodologies*

UML designs consist of…

- Class and object diagrams
- Use case diagrams
- Sequence diagrams
- Collaboration diagrams
- State charts
- Deployment diagrams
- …and many others

In this course we only discuss class and object diagrams.

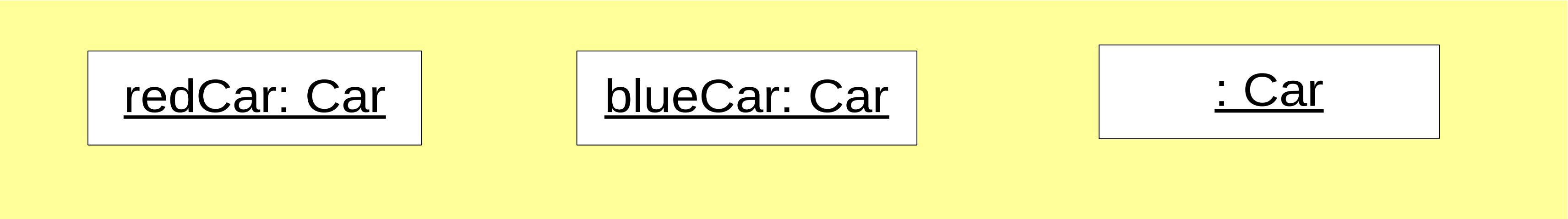
UML class and object diagrams may have:

- Objects (of certain class), with
  - attributes
  - operations
- Links between Objects,
- Aggregations between Objects,
- Association classes.
- Inheritance between classes.

Note: the attributes and operations have types.

Classes and Objects

- Objects have crisp boundaries within a problem domain; anonymous objects possible.

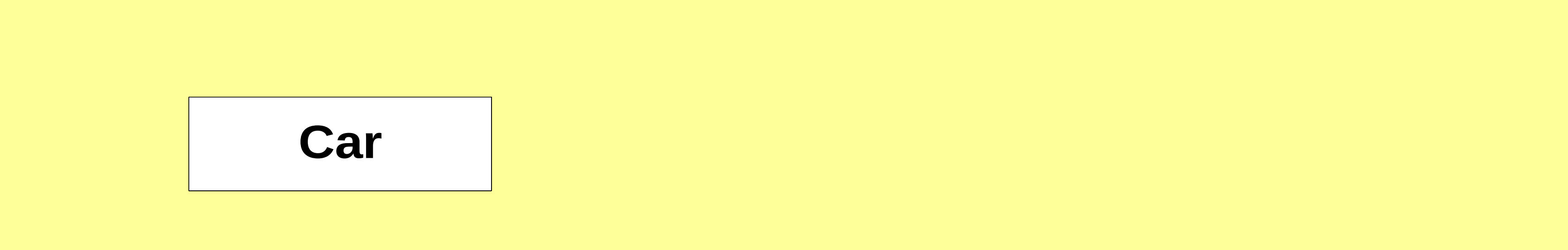


redCar: Car

blueCar: Car

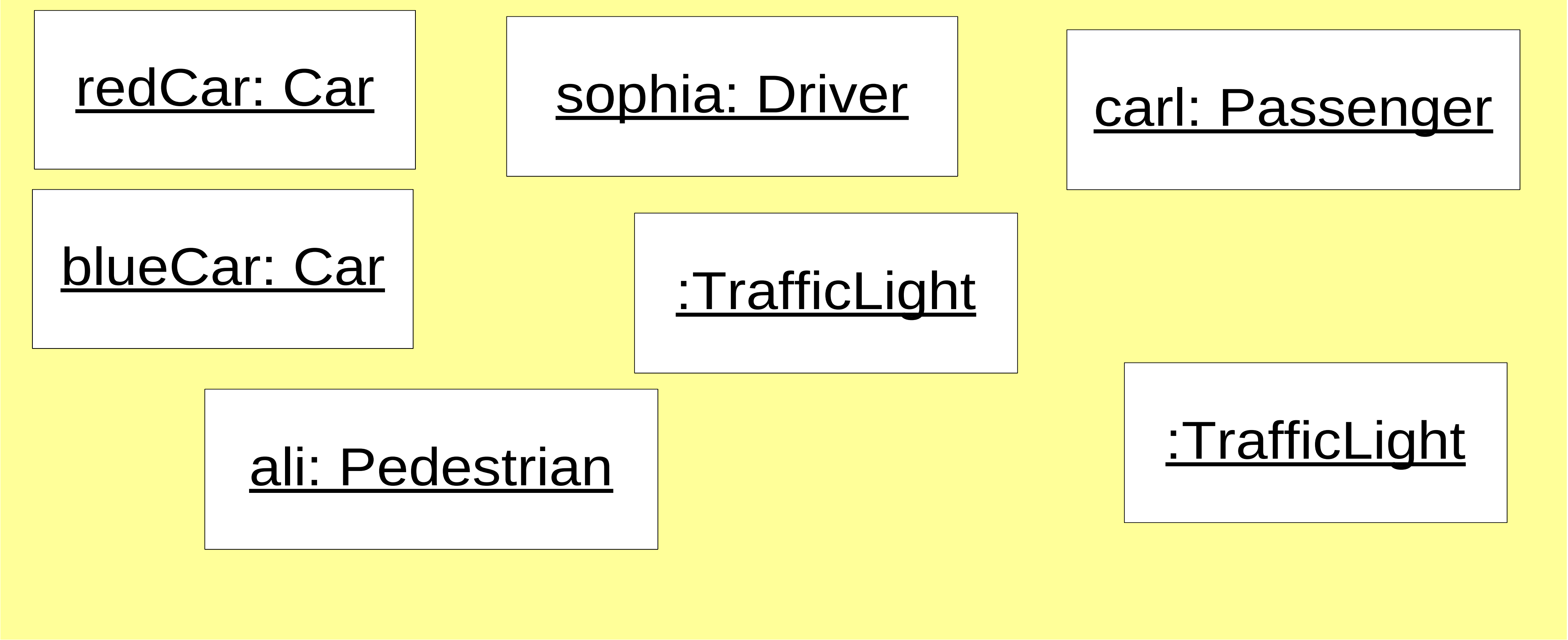
: Car

- Classes group objects with similar properties – objects are instances of classes.



Car

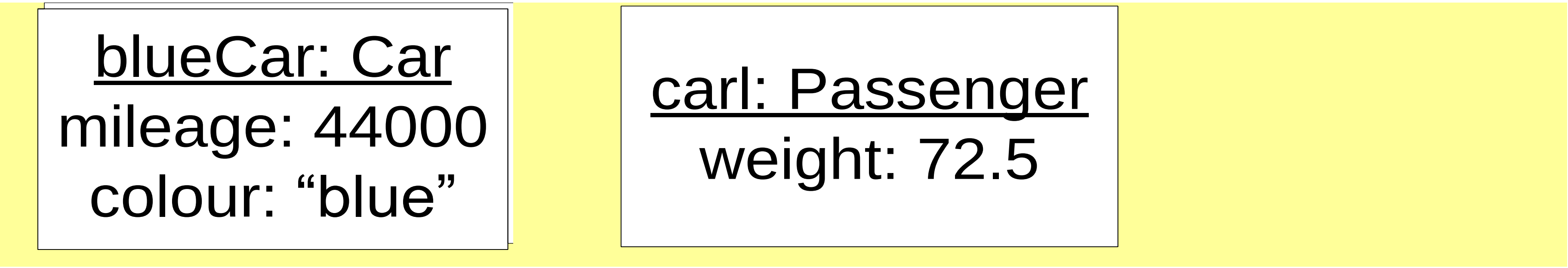
Problems usually have several different classes, and several objects of given class – depending on domain of interest.



Questions: 1) Which classes appear above? 2) What situation might be described above? 3) Think of some additional classes.

Attributes

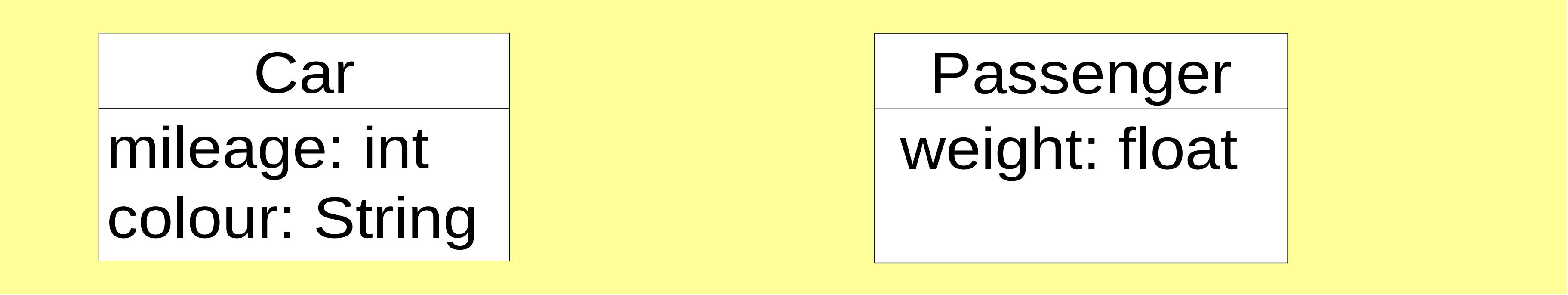
Attributes are properties of, or data held in, an object.



blueCar: Car
mileage: 44000
colour: “blue”

carl: Passenger
weight: 72.5

Attributes are declared in the corresponding classes:

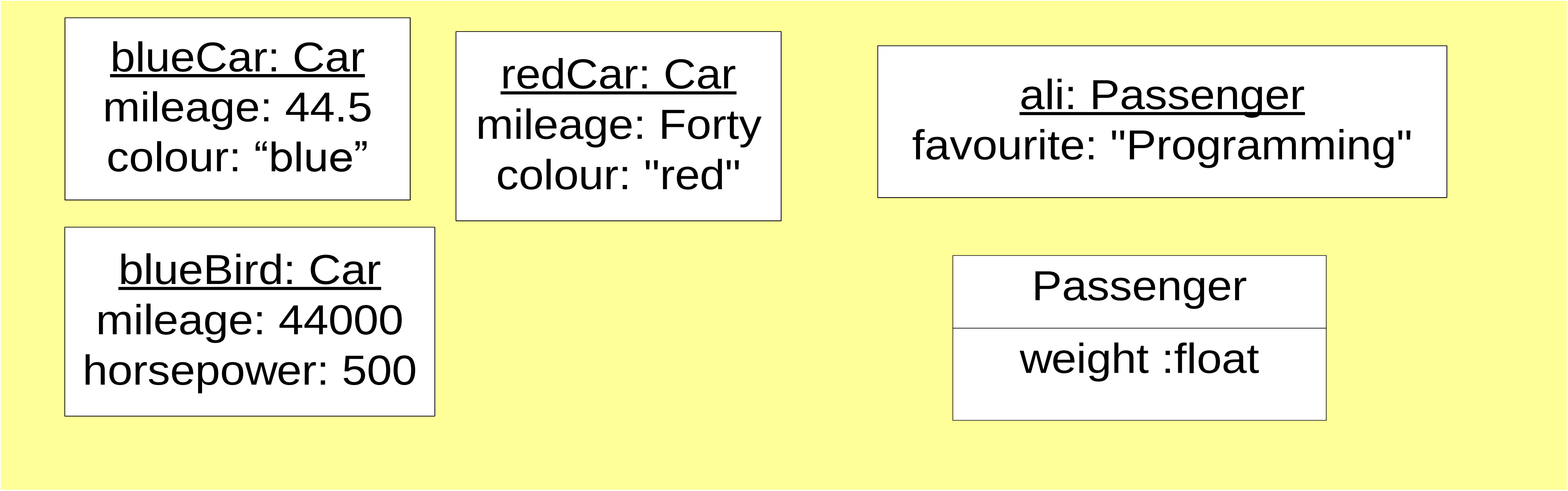


Car
mileage: int
colour: String

Passenger
weight: float

Objects of the same class have the same kind of attributes.

Therefore, the following situations are illegal:



blueCar: Car
mileage: 44.5
colour: “blue”

redCar: Car
mileage: Forty
colour: "red"

ali: Passenger
favourite: "Programming"

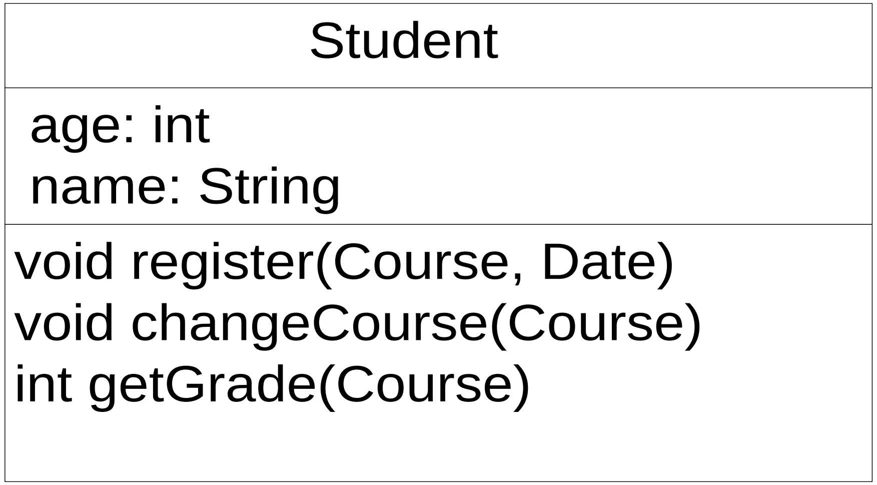
blueBird: Car
mileage: 44000
horsepower: 500

Passenger
weight :float

Operations

Operations are functions that may be applied to objects. Each operation has target object as implicit argument. Behaviour of object depends on its class (remember each object “knows” its class). Operations take parameters of certain type, and return result of certain type.

For example:



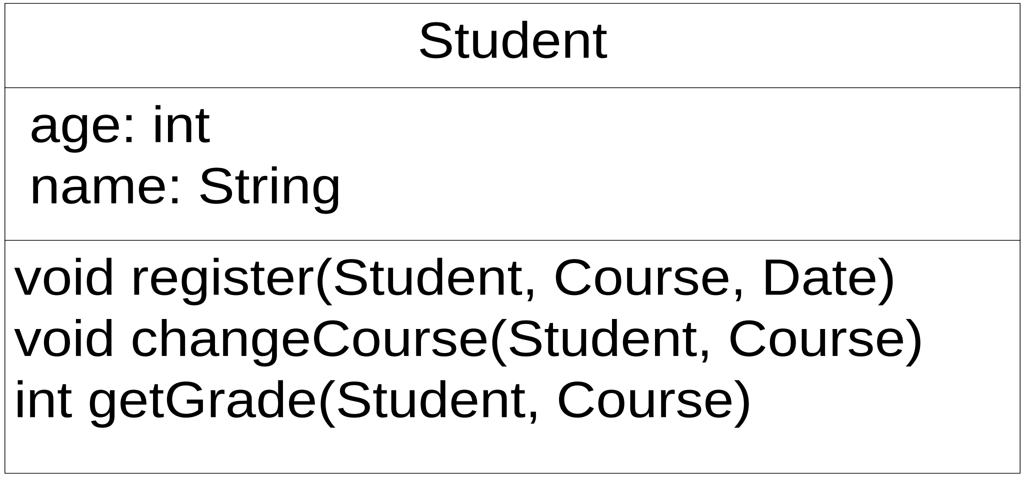
Student

age: int
name: String

void register(Course, Date)
void changeCourse(Course)
int getGrade(Course)

Operations (2)

- What is difference between an operation and an attribute?
- Why not have



Student

age: int
name: String

void register(Student, Course, Date)
void changeCourse(Student, Course)
int getGrade(Student, Course)

- **Note: always give types of attributes and operations**.

1st Exercise: Libraries

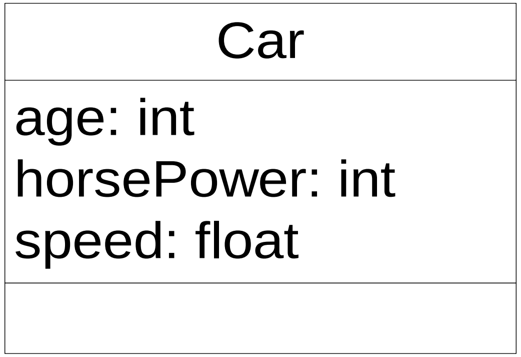
- A book called “C++ without tears”, at location 1, may be borrowed for up to 10 days,
- A book called “The Hitchhiker’s Guide to IC”, at location 2, may be borrowed for up to 13 days,
- A periodical called “Around the WWW in 80 sec”, at location 3,
- A periodical called “20 m under Queen’s Tower”, at location 4.
- All periodicals may be borrowed for up to 5 days.
- Library users are “georgia” and “ali”.

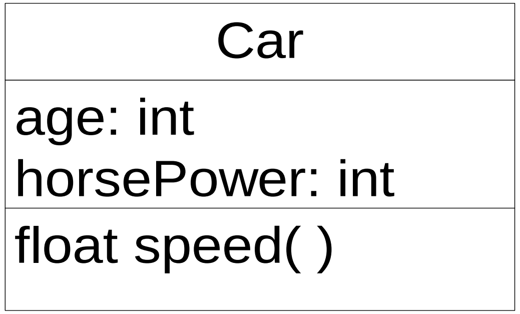
Library – UML instance diagram

Library – UML class diagram

Operations (3)

- Properties depending on other properties or attributes should be represented as and not as because,
- *For example,* A car has a horse power, and an age. Its speed is the product of its horse power and 100, divided by its age. Which of the following two designs is more appropriate?





Operations (4)   -- Who

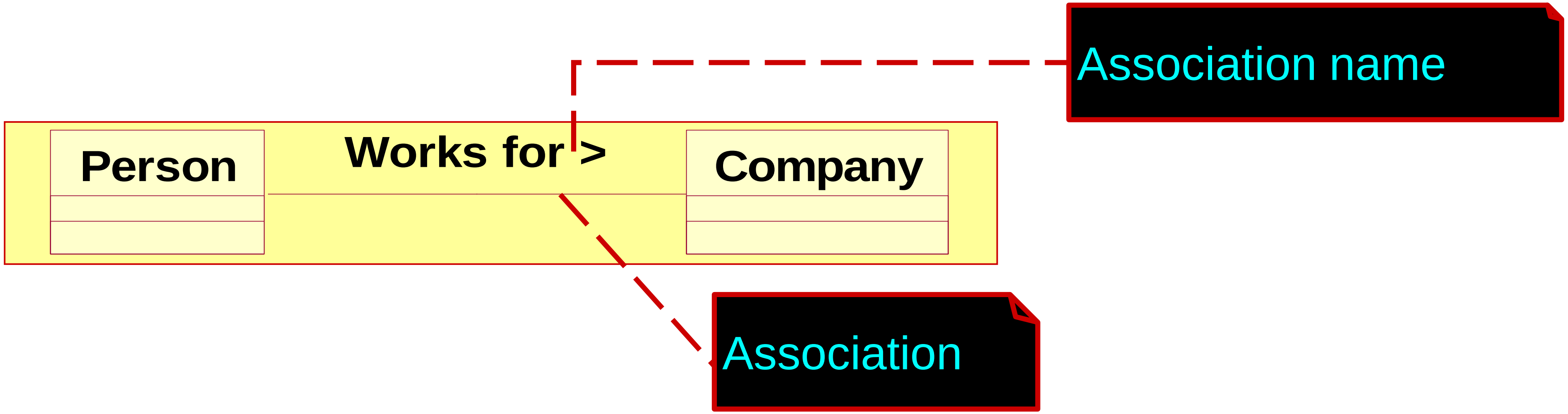
- Some operations may involve two or more objects. In such cases the question arises as to which class should contain that operation. To answer that, one considers:
  - **Who** does the processing,
  - **Who** holds the information,
  - **Who** decides the outcome of the operation.
- 1st example: When a person inflates a tyre, his blood pressure increases by 10, his temperature goes up by 2, his heartbeat increases to 110, and the tyre’s pressure goes up by 3.
- 2nd example: When a person inflates a tyre, the tyre’s pressure goes up by 3, its temperature goes up by 10, and if the maximal tyre pressure is reached, then the tyre explodes.

Operations (5) -- Parameters

- Operations may require further information, e.g. the tyre’s maximal pressure in the previous example. This information will be passed as a parameter, unless the object has a way of accessing/calculating this information.
- 1st Example: A tyre will explode, if when it is inflated, it reaches above its maximal pressure.
- 2nd Example: When an amount is paid into a bank account, then the balance of the account increases by the amount.
- 3rd Example: When a man and a woman get married, the registrar enters their names into his registry book, and sends a letter to their parents.

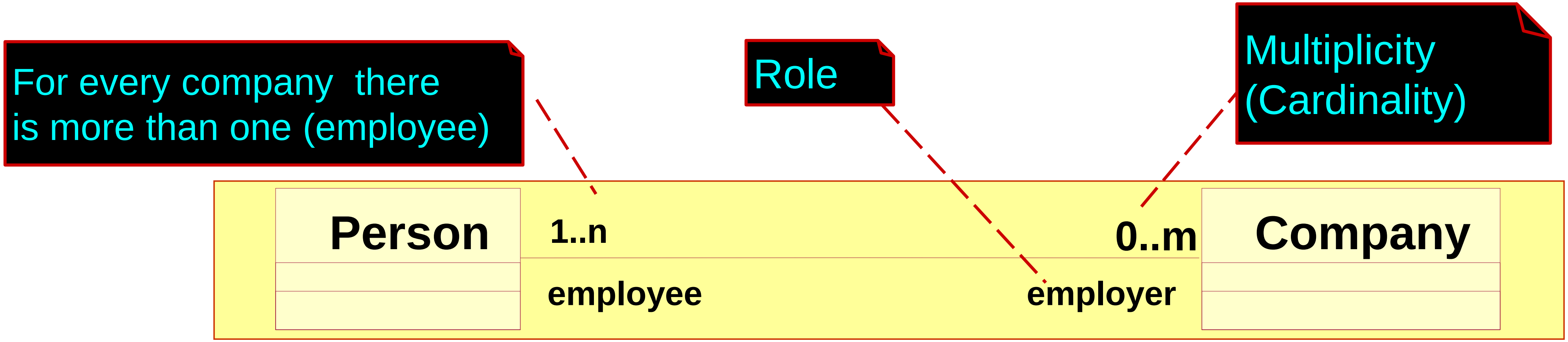
Associations

- Associations denote a relationship between concepts (classes).
- Associations often correspond to verb phrases including: physical location (next to, part of, contained in), directed actions (drives), ownership (has, part of), or satisfaction of some condition (works for, married to, manages).



Associations: roles and multiplicity

- Classes play a specific *role* in an association. The role is a named endpoint of the association.
- *Multiplicity*
  - * – any number, also written as 0..*, or 0..n
  - n – an exact number e.g., 1, 3, …
  - 0..1 – zero or one
  - 1..* – one or more, also written as 1..n



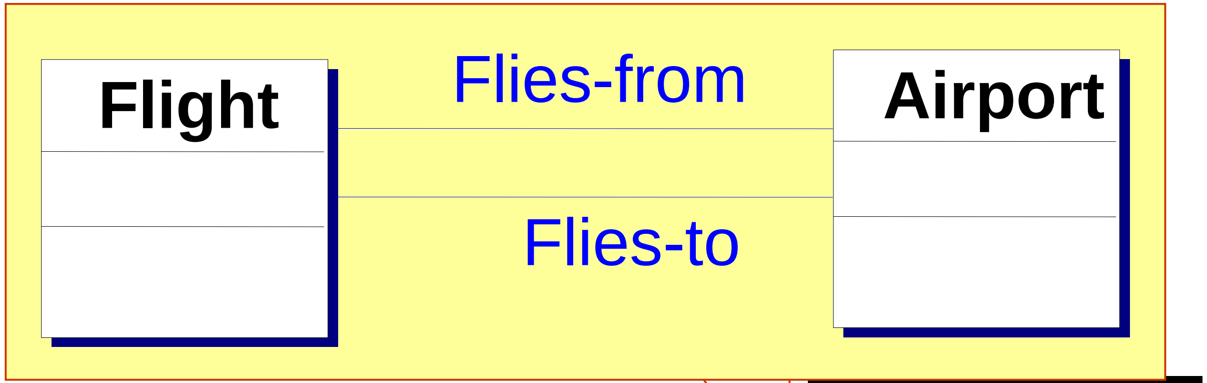
For every company there is more than one (employee)

Role

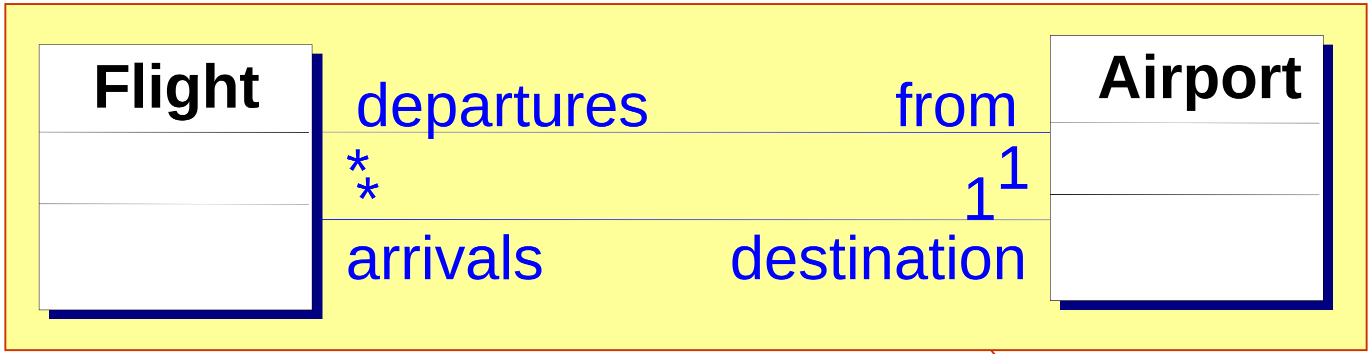
Multiplicity (Cardinality)

Multiple associations

- There can be multiple associations between the same classes.



association names



association roles

2nd Exercise, Library revisited

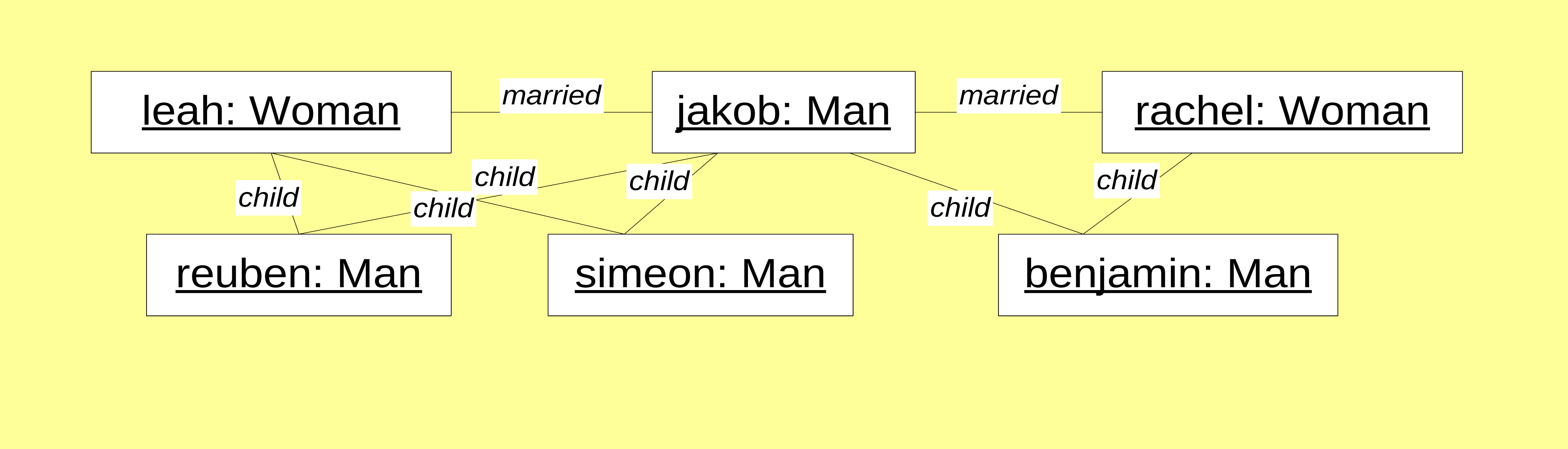
Georgia has borrowed “C++ without tears”, and “Around the WWW in 80 sec” .

Modify the instance diagram accordingly.

Modify the class diagram accordingly.

3rd Exercise: Families

Consider the following instance diagram and draw the corresponding class diagram.



Note: The instance diagram might lead to conclusions which do not hold in the real world.

Associations vs Attributes

Associations exist between objects; attributes have simple type (not objects).

Therefore

- Being mother of a person is represented through …

- Students having an age is represented through …

Note: attributes *and* associations might be represented through fields in C++. This decision does *not* concern the design phase, and is *not* reflected in the UML diagram.

Associations vs Operations

Associations represent persistent relationships; operations may establish, or break, associations between objects.

Which of the following are operations, which are associations?

- being married to
- applying for a driver's license
- meeting a colleague
- being friends with
- owning a car
- riding a bicycle
- a plane landing at airport

- marrying somebody
- divorcing somebody
- belonging to a club
- working for a company
- selling a house
- owning a house
- moving into a flat

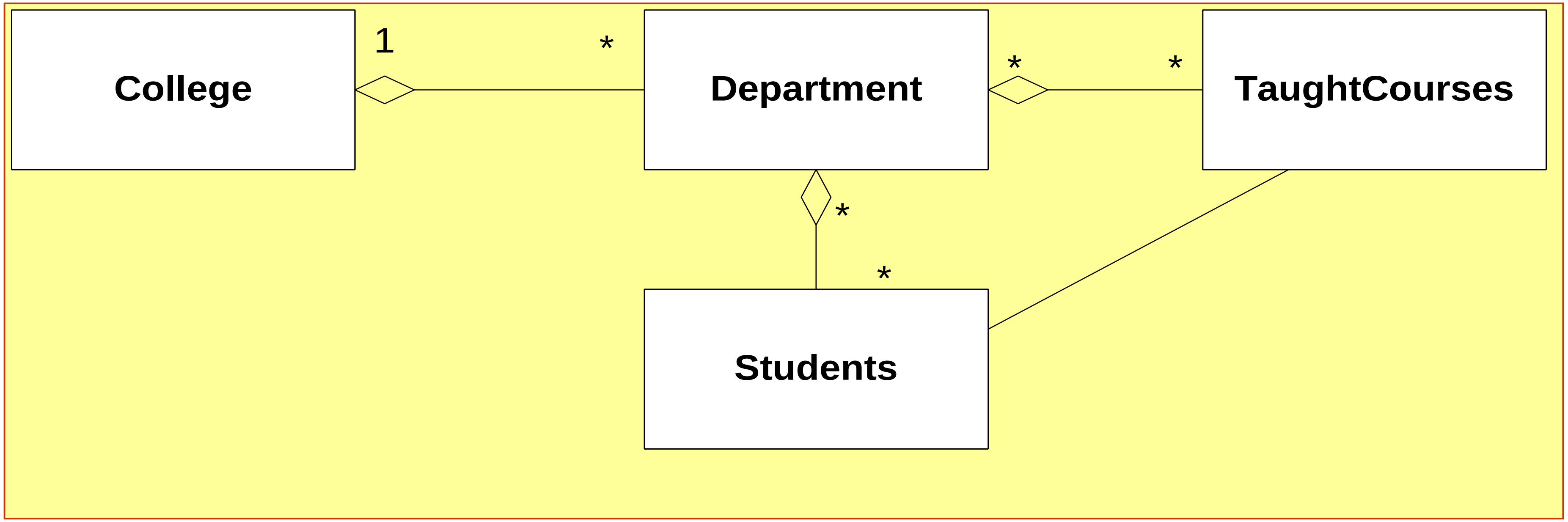
Aggregation

- An *aggregation* is a specialisation of an association denoting a ‘part of’ (or 'has-a') relationship between the objects of the respective classes.
- Two different types of aggregation:
  - *Shared Aggregation*
  - *Composition*

Note: For this course, we do not insist on the difference between shared aggregation and composition .

We do insist on the difference between aggregation and association.

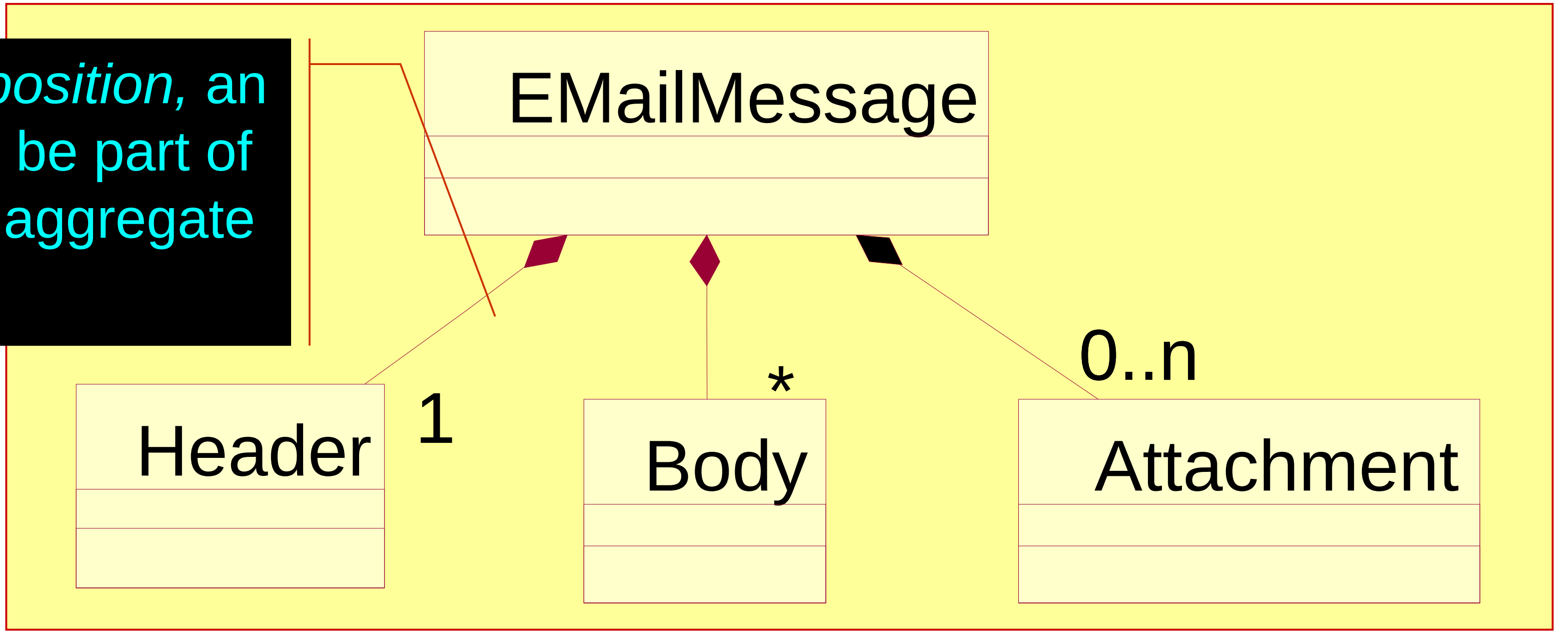
Shared Aggregation



Note: A student can be in more than one department and courses can be taught in more than one department

Composition

- *Composition* is a type of aggregation which links the lifetimes of the aggregate and its parts.
- In other words:
  - The *parts* cannot exist if the *aggregate* no longer exists
  - An entity can exist as part of only one aggregate



Under *Composition*, an element can be part of at most one aggregate

4th Exercise PCs

Write a UML class diagram to represent that a microprocessor consists of the following:

- At least one monitor
- One system box, which consists of
  - a chassis
  - a CPU
  - several RAMs
  - a fan
- possibly a mouse
- a keyboard

Associations vs Aggregations

Aggregation is a special kind of association, and usually links the lifetime of two objects.

Therefore, we represent

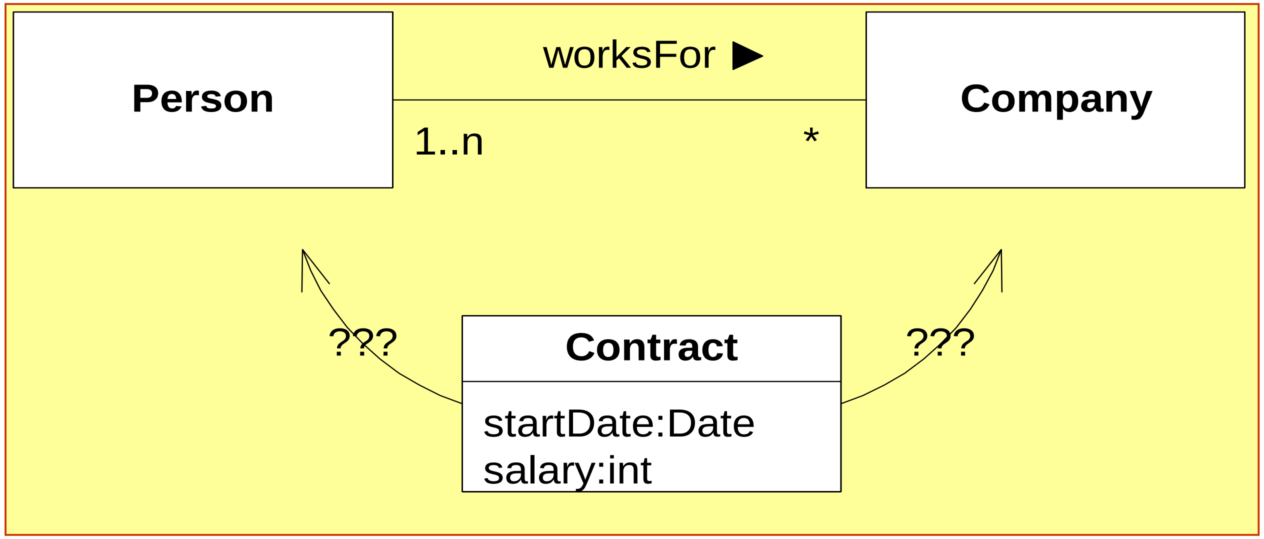
a department as part of a College as :

a woman married to a man as:

a car with steering wheel, an engine, and 4 wheels

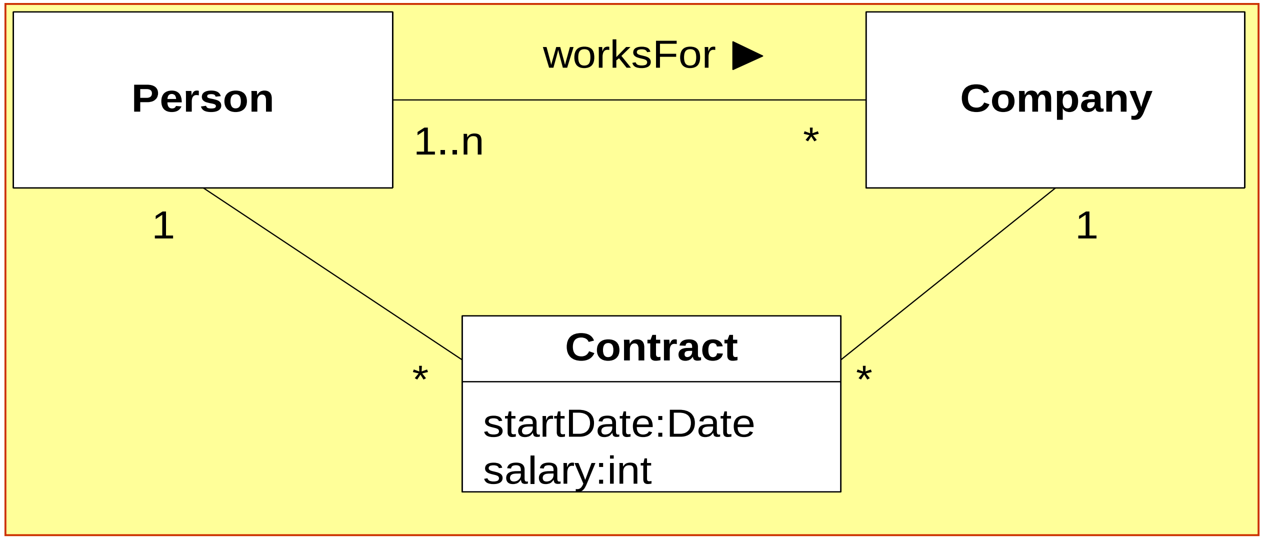
Association Classes

- Associations may have their own properties.
- An association class is an element that has both association and class properties.
- Occurs frequently in *many-to-many* associations because it is difficult to position the property at either end of the association.



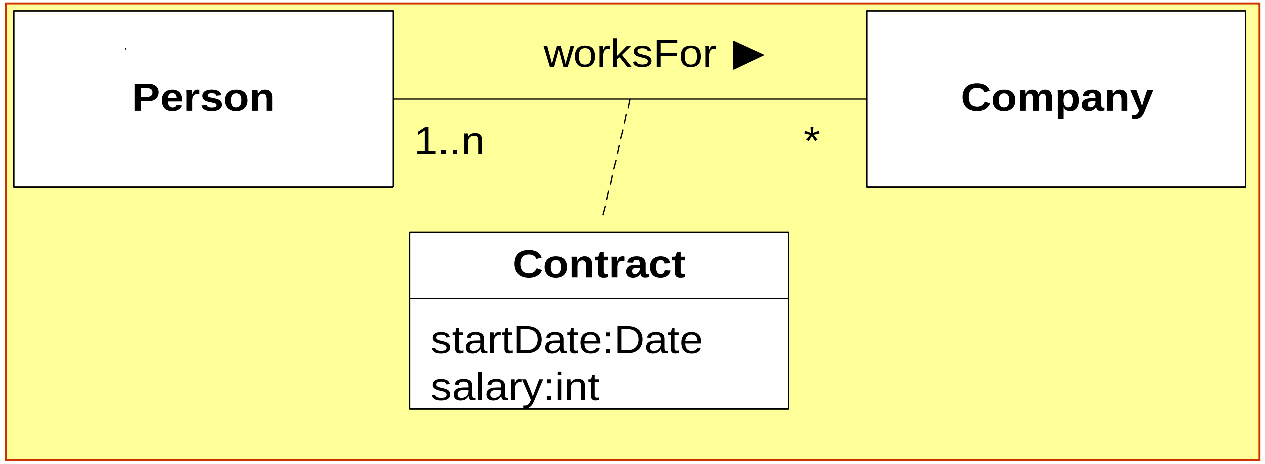
The employment is subject to a contract specifying start date, salary, etc.

Association Classes (2)



However, each *contract* instance depends on an instance of the *worksFor* association.

Thus the *contract* depends on the association. The second diagram is preferable!



5th Exercise: Families revisited

Modify the families class diagram from the 3rd exercise, to express that each child is born out of a particular marriage relationship.

6th Exercise: User authorizations

Develop a UML class diagram to express the following:

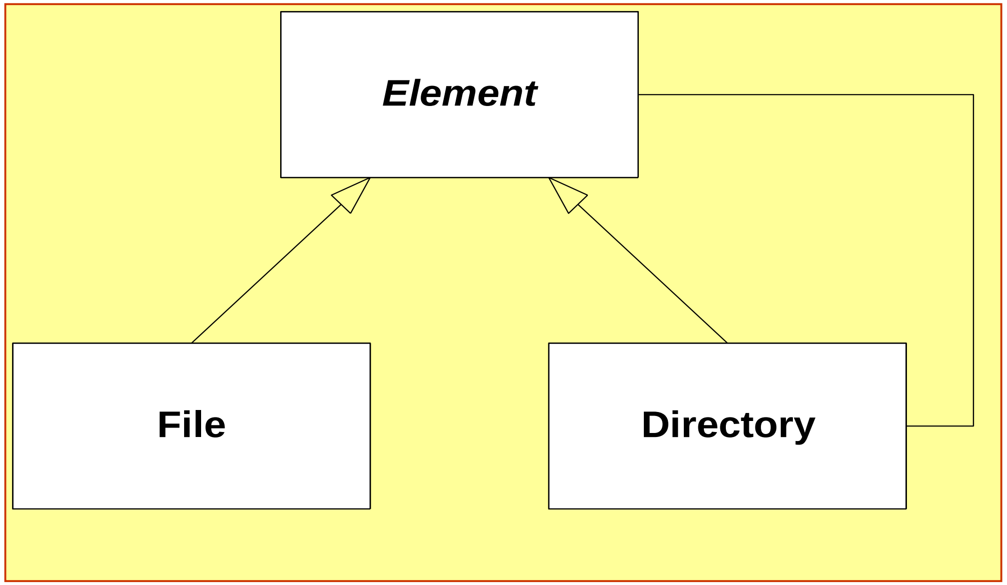
- A user may be authorized on several workstations
- Several users may be authorized on one workstation
- Each authorization has a priority (a number), a start session which is run when user logs on, an a directory which is the default directory when the user logs on.
- A directory has a name, and is contained in another directory.

Generalization

- Generalization = relationship between a class and its more refined versions
- Subclasses inherit all features of their ancestors (ie attributes, operations, aggregations, and associations}
- Generalization is important:
  - modelling: captures similarities;
  - software: supports re-use

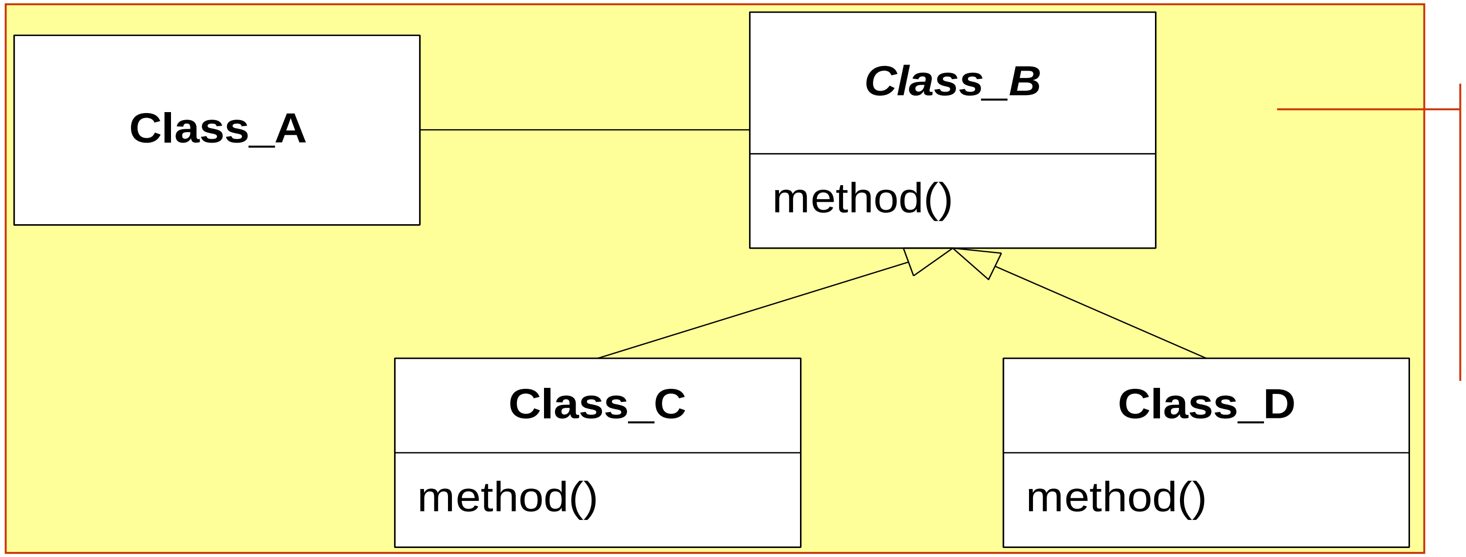
- **Note*:* a super class should not have elements which are applicable only to some of its subclasses.**

Generalization – Example



A file is contained in a directory.
A directory is contained in another directory.

Generalization and Polymorphism



Class_C and Class_D implement method() in different ways

7th Exercise: Employees

Express the following in a UML class diagram:

- Employees may be managers or executives.
- Employees have a salary, and may be given a payrise.
  - When a manager is given a payrise, his salary is incremented by 100,

    but
  - When an executive is given a payrise, his salary is incremented by 300.

8th Exercise: Library re-revisited

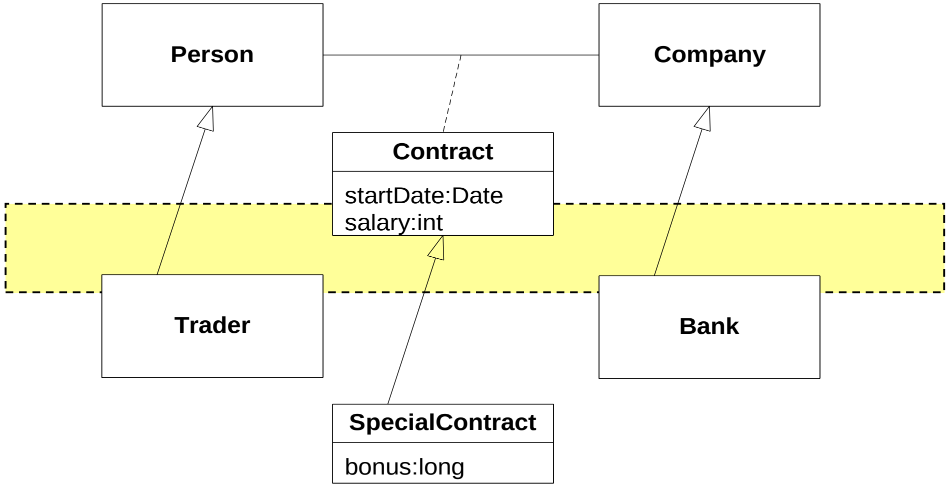
Develop a more elegant version of the library class diagram.

Parallel Hierarchies

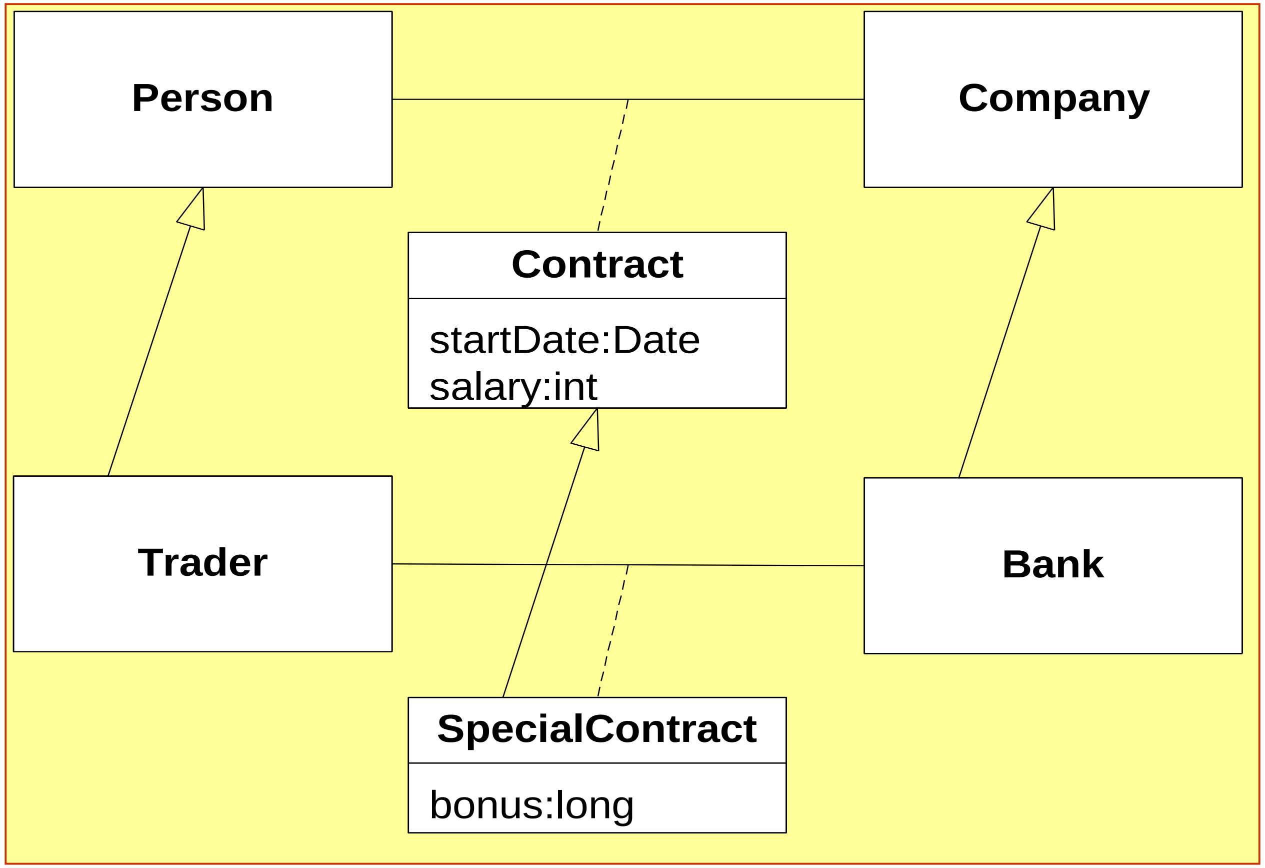
Sometimes an association is refined in a subclass.

For example, a person working for a company has a contract, but traders working for banks have a special contract.

This may be expressed as:



… and may also be expressed as:



9th Exercise: Flights

Develop a UML class diagram to express the following:

- A flight has a code number, a date and a price.
- Passengers have a name and an address, and may be booked on flights.
- There are firm bookings, where the whole price of the flight has been paid, and flexible bookings, where only a deposit (smaller than the flight price) has been paid.
- A Flexible booking may be cancelled up to 10 days before the date of the flight; then 80% of the deposit paid will be returned.

(continued overleaf…)

9th Exercise: Flights (2)

- A firm booking may only be cancelled up to 20 days before the date of the flight, but then the complete flight price will be returned.

- Information about a flight may be printed: The fight number, and date will be printed, followed by the name, address and the outstanding amount (ie price of flight) for each of the passengers booked.